A Flexible Approach to WSN Development and Deployment

Luis D. Pedrosa
Instituto de Telecomunicações & Laboratory of Excellence in Mobility – LEMe,
Instituto Superior Técnico, Technical University of Lisbon,
Av. Prof. Dr. Cavaco Silva, 2744-016 Porto Salvo, Portugal
E-mail: luis.pedrosa@lx.it.pt

Pedro Melo
Laboratory of Excellence in Mobility – LEMe
Instituto Superior Técnico, Technical University of Lisbon,
Av. Prof. Dr. Cavaco Silva, 2744-016 Porto Salvo, Portugal
E-mail: pedro.melo@tagus.ist.utl.pt

Rui M. Rocha*
Instituto de Telecomunicações & Laboratory of Excellence in Mobility – LEMe,
Instituto Superior Técnico, Technical University of Lisbon,
Av. Prof. Dr. Cavaco Silva, 2744-016 Porto Salvo, Portugal
E-mail: rui.rocha@lx.it.pt
*Corresponding author

Rui Neves
Instituto de Telecomunicações & Laboratory of Excellence in Mobility – LEMe,
Instituto Superior Técnico, Technical University of Lisbon,
Av. Prof. Dr. Cavaco Silva, 2744-016 Porto Salvo, Portugal
E-mail: rui.neves@lx.it.pt

Abstract: A flexible Wireless Sensor Network platform for easier implementation of diverse applications has been developed and deployed at the Instituto Superior Técnico / Technical University of Lisbon (IST-TUL). This test-bed integrates multiple projects into a single common network, thus creating an expandable platform that facilitates the development of future applications. To achieve this flexibility, a dedicated software framework was developed that not only provides a centralized configuration panel that is accessible over the Internet, allowing the administrator to configure common network parameters, but also supports application programmability, enabling fine-grained control of in-network sensing, processing, and actuation. On top of this platform, four initial applications have been developed and are currently coexisting within the same network, thus demonstrating the new platform's capabilities. The article discusses the main issues related with the test-bed architecture and the development of the environmental interaction and the vibration monitoring applications, with an illustrative purpose, along with the deployment challenges. Results of the experimental evaluation of the test-bed are also shown, focusing on the vibration monitoring application's capacity limits and the performance of the environmental interaction application's in-network processing system. The vibration monitoring application's experimental results show that remote processing on sensor nodes is needed to successfully perform multipoint vibration measurements in real-time, as the high frequency sampling used by this system can easily consume all of the network's resources, if compression schemes are not used. As for the Environmental Interaction Application, a particularly relevant result is denoted by the minimum time it needs to complete its processing tasks within the network (approximately 200 ms in our test topology).

Reference to this paper should be made as follows: Pedrosa, L.D., Melo, P., Rocha, R.M. and Neves, R. (2009) 'A Flexible Approach to WSN Development and Deployment', *Int. J. Sensor Networks*, Vol. x, Nos. x, pp.1–12.

Keywords: deployment; environmental interaction, high fidelity sampling; in-network processing, performance measurement, multihop networks, vibration monitoring, wsn test-bed.

Biographical notes: Luis D. Pedrosa (IEEE S'08) received both his B.Sc and M.Sc Degrees at the Instituto Superior Técnico – Technical University of Lisbon (IST-TUL), in Portugal. He is currently doing research at the Instituto de Telecomunicações (IT), in Lisbon, Portugal, within the broad area of networked embedded systems (a.k.a. Wireless Sensor Networks - WSNs), more specifically around transport layer and middleware issues, as well as opportunistic networking.

Pedro Melo received his B.Sc and is finalizing his M.Sc Degrees at the Instituto Superior Técnico – Technical University of Lisbon (IST-TUL), in Portugal. He is currently doing research in wireless sensor networks, specifically in structure health monitoring applications and high fidelity sampling. He is also working in Next Generation Networks at Nokia Siemens Networks.

Rui M. Rocha (IEEE S'89, M'94, SM'07) graduated in electrical engineering at the Instituto Superior Técnico – Technical University of Lisbon (IST-TUL) in 1981, and received the M.Sc. and Ph.D. degrees in electrical and computer engineering from the same Institute in 1987 and 1995, respectively. Since 1986 he has been teaching at IST-TUL, involved in courses on computer architecture, broadband communications and wireless systems. He coordinated the Information and Communication's Networks Engineering graduation programme at IST-TUL from 2001 until 2006. He was affiliated with INESC, having participated in several national and international research projects until 1997. He is now an Associate Professor with the Electrical and Computer Engineering Department of IST-TUL, and a researcher at the Instituto das Telecomunicações (IT), in Lisbon. His current research interests are focused on the heterogeneous communications networks area, namely on self-organized networks and wireless sensor systems.

Rui Neves has been a professor at the Instituto Superior Técnico – Technical University of Lisbon (IST-TUL) since 2005. He received the Eng. Diploma and the PhD degrees in Electrical and Computer Engineering from IST-TUL in 1993 and 2001, respectively. In 2006 he joined the Instituto de Telecomunicações (IT), as a Research Associate. His research activity deals with sensor networks, embedded systems and mixed signal integrated circuits. During his research activities he has collaborated / coordinated several EU and National projects.

1 INTRODUCTION

Advances in networking for embedded systems have brought about new challenges to tackle and new problems to solve. The special needs related to these new kinds of networks and the paradigms that evolved from them created the concept of Wireless Sensor Networks (WSNs), as described by Akyildiz et al. (2002). These networks are built upon low cost, battery powered, wireless nodes, commonly referred to as motes, whose processing abilities, albeit limited, go beyond mere sensing, thus enabling the creation of intelligent data centric networks.

Although WSNs have generally been developed in academic environments, industrial and other commercial applications are becoming more common. Hence, traditional research, mainly based on simulation and lab work, must be complemented with applied experimentation, based on real deployed sensor networks (Tanenbaum et al. 2006).

The Instituto Superior Técnico – Technical University of Lisbon (IST-TUL) has already cultivated some experience in WSNs, namely through the participation in several projects both at internal and external levels (e.g. CRUISE (IST NoE CRUISE, 2006a) and NEWCOM++ (IST NoE NEWCOM++, 2008)). However, most projects were developed as isolated setups, providing solutions to single specific problems (e.g. the LEMe Room Project for determining a user's position within a room (IST NoE CRUISE, 2006b)).

The effort involved in the setup of every new WSN-based application led to the idea of building a common test-bed that not only integrates multiple projects into a single sensor network, but also provides a basic software platform, based on the TinyOS operating system (Hill et al., 2000), that eases the development of future applications. On the other hand, our previous deployment experience taught us that putting sensor nodes to work out of a lab-controlled environment is not an easy task. As such, we decided to follow an incremental deployment strategy which provides an ongoing insight into the challenges of deploying these networks, clearly assessing the differences between simulating or using WSNs in the lab and on the field.

Using this common test-bed – the Tagus-SensorNet – two parallel development efforts have been pursued. On one hand, a specialized software framework has been developed to allow multiple concurrent applications to coexist within the network and operate in an integrated fashion. This framework is used as a set of common libraries and APIs, providing key services and enabling the use of modular design principles in current and future applications. On the other, the following applications were developed, not only to exercise the new test-bed, but also to provide a useful service to the campus community:

- Environmental Interaction Application - a generic data retrieval, processing and actuation system that allows users to configure what data is collected within the network, how it should be processed, which

decisions should be made based on the results, and how these decisions should be acted upon.

- Vibration Monitoring Application - allowing the user to either collect raw vibration measurements or receive periodic peak and RMS aggregate values. Raw data may be further analyzed to extract such information as the structure's vibration modes; however, the required bandwidth limits its use to only a few sensor nodes within the network. Peak and RMS values, on the other hand, use far less network resources and, as such, can be collected from many more sensor nodes at once, while still providing essential structural health data.
- Temperature Gradient Map - implementing a distributed algorithm used to determine the temperature gradient map within a room; since nodes collaborate to calculate the room temperature gradient, the system offers a more efficient solution than simply collecting the data from each individual node.
- Remote Monitoring and Control - enabling remote access to sensors in one or more wireless sensor networks through machines or devices that aren't directly connected to them. This way, users may use the Internet to know the remaining battery charge on any given node, or to manually initiate local actuation. This service is composed of a Gateway that gets the queries from the user over the Internet and translates them into native network messages that convey the query directly to the desired sensor node.

Although all four applications are now fully functional, the scope of this article is focussed on the environmental interaction and the vibration monitoring applications, due to their particular relevance within IST-TUL, as well as the wireless sensor network community at large. Aside from demonstrating the network's capabilities, these applications were also used to evaluate some performance metrics, thus providing a further insight into the technical trade-offs that characterize this type of network.

This new platform brings several new challenges to overcome. On one hand, the network must operate with minimal operational effort, automating common management tasks whenever possible. This way, the network is configurable through a centralized panel that is accessible remotely over the Internet. On the other hand, individual applications must also be easily programmable, allowing the network administrator to control in-network sensing, processing, and actuation procedures with ease. Finally, the platform must be expandable, allowing future applications and sensors to be easily integrated with current ones, not only enabling the use of heterogeneous sensor technologies, but also heterogeneous applications within the same network.

The network deployment, as mentioned before, poses a challenge of its own (Camilo et al., 2006). Several factors have to be taken into account when deciding where to locate each sensor (e.g. physical sensing range, wireless radio range, accessibility for occasional maintenance and battery replacement, sensor physical security, and minimal aesthetic intrusiveness). Therefore, this particular aspect of the current work is, by no means, a less important issue to tackle and lessons learnt from it won't be of second order for people involved in WSN deployment.

The remainder of the article is organized as follows: section two provides an overview of the network architecture used in the test-bed; section three explains the services that have been developed over the network; section four describes some of the applications that have been developed using this system; section five provides an insight into the deployment procedure; section six shows the experimental results of the performance tests that have been conducted; finally, section seven draws some final conclusions and lays out the foundation for future work.

2 TEST-BED ARCHITECTURE

Given the high level of flexibility that the test-bed requires, a traditional single-sink homogeneous network would not suffice. Hence, an irregular multi-sink / multi-tier network, as illustrated in Figure 1, was envisioned from the beginning. This flexible topology uses traditional Ethernet networks to connect multiple wireless sensor sub-networks that are deployed in physically separated locations, forming a new tier that allows all of the separated sub-networks to be operated and managed as a single entity.

Additionally, although the network was initially built using a single sensor hardware platform, the integration of different platforms is planned for the near future. Taking all of these factors into consideration, the Tagus-SensorNet test-bed was deployed, spreading a total of 23 nodes, forming 3 connected regions, around the IST-Taguspark main building, as shown in Figure 2. The main connected region lies in the middle of the building and includes 5 hallway motes, used as general purpose routing/monitoring nodes, 4 bridge motes, used to measure structural vibrations, and a sink node. The connected region towards the south end of the building (towards the left, in Figure 2) is made up of 7 motes installed within the chemistry lab. These nodes will be used to monitor the air quality within the lab, generating alerts whenever the concentration of certain dangerous gases exceeds a safety threshold. Finally, the 6 motes that are deployed within a single room, just to the right of the main area within Figure 2, are a part of the LEMe Room project (described in IST NoE CRUISE, 2006b) and, as such, are used to locate users within the room.

Figure 1 Test-bed multi-tier / multi-sink architecture

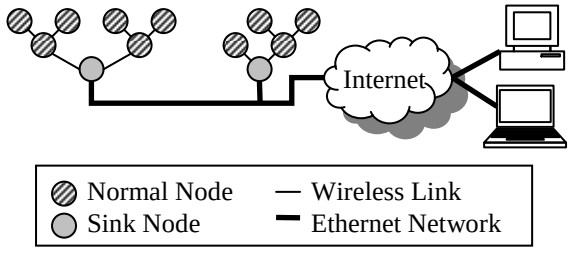


Figure 2 IST-Taguspark main building map and Tagus-SensorNet topology

As for the software platform, TinyOS was used as the common foundation. TinyOS is an operating system for networked embedded systems that uses nesC (Gay et al., 2003), an extension of the traditional C language, to provide component based abstractions. The use of this particular platform not only promotes modular and extensible design, but also facilitates the development of multi-platform software that can be implemented once, and deployed on multiple hardware platforms, both characteristics that were seen as fundamental requirements of our common test-bed.

3 SERVICES

Tagus-SensorNet test-bed was designed to consolidate multiple heterogeneous applications. To be able to cope with such heterogeneity in an integrated environment, a set of common services was developed. These services were implemented as shared libraries and made available to all applications within the network, providing a shared functionality that is obtained collaboratively through all nodes. These services are described in further detail below.

3.1 Centralized Graphical Management Console

To allow users to interact with different network applications through a single integrated computer application a common graphical management console was developed. This console is also capable of communicating with the sensor network remotely, over the Internet, allowing network users to manage all of the network's applications from outside the University Campus.

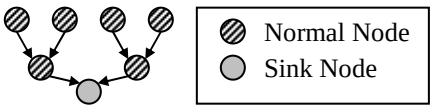
3.2 Local User Interface Panel

Whereas the centralized Graphical Management Console provides a convenient way for users to interact with the network using a computer, there are some scenarios where it would be more convenient to access the network directly from within the monitored environment. This is the case of e.g. the wireless sensor sub-network installed within the chemistry lab where the concentration of dangerous gases is monitored. A direct access to an interface capable of setting up local safety thresholds along with the possibility to generate corresponding alarms is a valuable asset in this application. Hence, a local interface panel was developed, allowing users to interact with individual sensor nodes directly through an LCD / keypad set.

3.3 Convergecast Data Collection

Convercast routing is used to collect data from multiple source nodes within a network, forwarding it across multiple hops towards one of possibly many sink nodes, as Figure 3 illustrates. This basic routing model, albeit simple in nature, is quite common within WSNs (Karl & Willig, 2005), as it may be used to collect such kinds of data as sensory measurements, application state, or debug information.

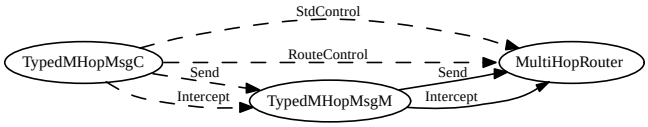
Figure 3 Convergecast routing model



Although TinyOS natively supports this routing model, the original implementation lacks some of the flexibility required by Tagus-SensorNet As such, a common convergecast network layer was developed, extending TinyOS' native multi-hop convergecast routing capabilities so to multiplex data from multiple applications, while also allowing fixed routes to be configured into the network in run-time, enabling users to change the networks routing topology on the fly.

Since the new convergecast common network layer was designed to extend TinyOS' native one, rather than to replace it, two modules were developed, following the Facade and Decorator design patterns (Gay et al., 2005): TypedMHopMsg and MultiHopFixedRoute. The TypedMHopMsg component is wired according to Figure 4, extending the native system to support multiple applications. This way, all the application has to do to use the new extended network layer is to wire its Send and Intercept interfaces to the TypedMHopMsgC component, rather than TinyOS' original MultiHopRouter. When this happens, the interfaces are diverted through the TypedMHopMsgM module that adds an 8-bit application identifier alongside the message payload, thus allowing the central management console to distinguish which sensor application generated the data.

Figure 4 TypedMHopMsg component wiring



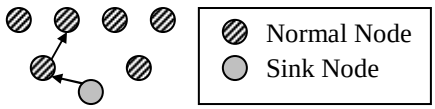
The MultiHopFixedRoute component, in turn, uses a similar wiring technique, illustrated in Figure 5, to selectively bypass the native routing protocol. Under these circumstances, TinyOS' MultiHopEngineM is rewired to get its routing information from the new MultiHopFixedRouteM module. This module then gets the static routing configuration from the MoteAppM module and, if a manual route is defined, it will take precedence over the automatic one. On the other hand, if no such route is set, the native automatic system, provided by MultiHopLEPSM, will prevail.

Figure 5 MultiHopFixedRoute component wiring.

3.4 Unicast Data Delivery

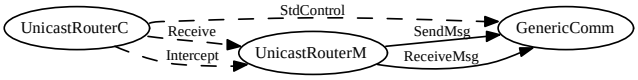
Unlike the convergecast data collection service, the unicast data delivery service provides a new routing model not frequently found in WSNs, where data can be forwarded from a sink to any specified destination node within the network, as depicted in Figure 6. This allows an application running on the central management console to communicate with individual nodes, conveying orders or new run-time parameters.

Figure 6 Unicast routing model



As TinyOS provides no equivalent implementation for the Unicast routing mechanism, the system had to be implemented entirely from scratch, using the wiring illustrated in Figure 7. This system lies directly between the application and the link-layer, hence the UnicastRouterM module is wired directly to TinyOS' GenericComm link-layer component, ultimately exporting the Receive and Intercept interfaces to the application, via the UnicastRouterC component. This routing library uses a simple source-routing mechanism, allowing data to be sent from the sink to any single node within the network by appending the packets entire route after its payload. This way each forwarding node needs only to look at the packet's final bytes to discover the next hop. Additionally, the packet is truncated prior to being sent, thus removing the current hop. Eventually, the destination node will receive the packet, now with no hops specified, thus delivering it to the appropriate local application.

Figure 7 UnicastRouter component wiring



3.5 Time Synchronization

Many sensor network applications can benefit from, or even depend on, the use of synchronous operations that execute periodically and at the same time across all nodes within a region of the network. These operations may be used to enable such activities as periodic measurements that coherently take snapshots of the measured reality, or to synchronously duty cycle the motes' radios, saving precious energy and extending the networks life-time.

To provide this service, TinyOS' FTSP protocol implementation was extended, creating a new *round synchronization* abstraction. Whereas FTSP (Maroti et al., 2004) provides a global reference clock-like abstraction, the round synchronization concept creates a higher level of abstraction, where applications may set the round period (i.e. the periodicity of the synchronous events) and the round synchronization service ensures that each round starts at the same time on each node, while also globally identifying the round with an incremental sequence number.

4 APPLICATIONS

Once the initial library of common services was ready to be used, four initial applications were developed with an illustrative purpose: an Environmental Interaction Application, a Vibration Monitoring Application, a Remote Monitoring and Control Application, and a Temperature Gradient Map. As previously referred, the following sections will focus specifically on the Environmental Interaction and the Vibration Monitoring Applications.

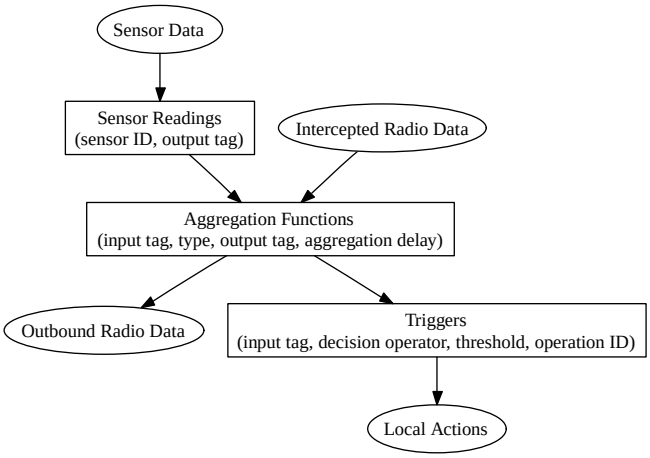
4.1 Environmental Interaction Application

The environmental interaction application is a generic data measuring and processing tool. By configuring each node individually, the network can be programmed to collect data from specific sensors, process and act upon it, in a completely autonomous fashion.

The idea behind the underlying system is simple: using the time synchronization service, all nodes operate in multiple synchronous measuring / processing / actuating cycles in a sequence of multiple *rounds* of a configurable period, which begin with all nodes performing the specified measurements simultaneously.

The basic structure of the environmental interaction application is depicted in Figure 8. The system is built around three main sub-systems: *Sensor Readings*, *Aggregation Functions*, and *Triggers*.

Figure 8 Environmental interaction application data acquisition and processing scheme



The *Sensor Reading* subsystem allows the user to define which sensors are measured and to associate with each measurement a configurable *Output Tag* (an 8-bit identifier). During the rest of the round, this Tag is used to refer to the data as an input to any further in-network processing or decision making.

Once this sensor reading procedure has been configured, the user can then setup multiple *Aggregation Functions* at each node. This way, instead of sending all of the measured data towards the sink to feed a centralized processing mechanism, the data may be consumed within the network, producing summarized aggregate values. Just as before, this sub-system also provides a versatile configuration, allowing the user to select the function type (minimum, maximum, average, etc.) and the function's Input and Output Tags, as well as its aggregation delay (the size of the time window, starting at the beginning of the round, during which the function accepts incoming data and after which it generates the aggregate result). Just as in the sensor readings, the output of an aggregation function can also be forwarded across the network or otherwise be consumed as an input for another aggregation function on any downstream node.

Finally, the user may also define one or more *Triggers* at each node that monitor the data associated with a specified Input Tag, comparing it with a defined threshold according to a specified decision operator (greater than, lesser than, etc.). If this Boolean decision becomes true, a configurable operation is set-off, ultimately acting upon the decision.

This system is best explained through an example. At the beginning of each round, multiple sensor nodes within an industrial facility may be configured to measure the temperature and forward the data, associating it with Tag 1, towards a conveniently placed forwarding node. This forwarding node is configured to wait 500 ms for any incoming data associated with Tag 1 and then to calculate the maximum of all the received values, associating Tag 2 to the result. All Tag 2 data is then examined and, if it exceeds a configured critical limit, an audible alarm is set-off, alerting anyone present to evacuate the room immediately. This maximum temperature value is also forwarded towards a sink node, where it may be used for further logging or off-line analysis.

Although the use of aggregation functions and simple decision making within sensor networks is not new, the innovative aspect of this application relies in the great flexibility it offers in its motes' configuration. As shown in Figure 9, each mote can be configured individually using the centralized graphical console, allowing the user to configure all of the above mentioned functionality with ease. Under these circumstances, aside from being able to create powerful sensor monitoring systems, the user is also capable of defining advanced autonomous systems that can sense environmental data, process and act upon it, entirely within the network, without ever having to write a single line of code or reprogram any of the motes.

4.2 Vibration Monitoring Application

The vibration monitoring application was developed to monitor the movements of the pedestrian bridges inside the campus. It uses several sensor nodes to measure accelerations at high sampling rates and to transmit the measurement data to the sink node. This data is then displayed graphically in a custom interface, which is also used to control and configure each sensor node independently and remotely. The architecture of this application is illustrated in Figure 10.

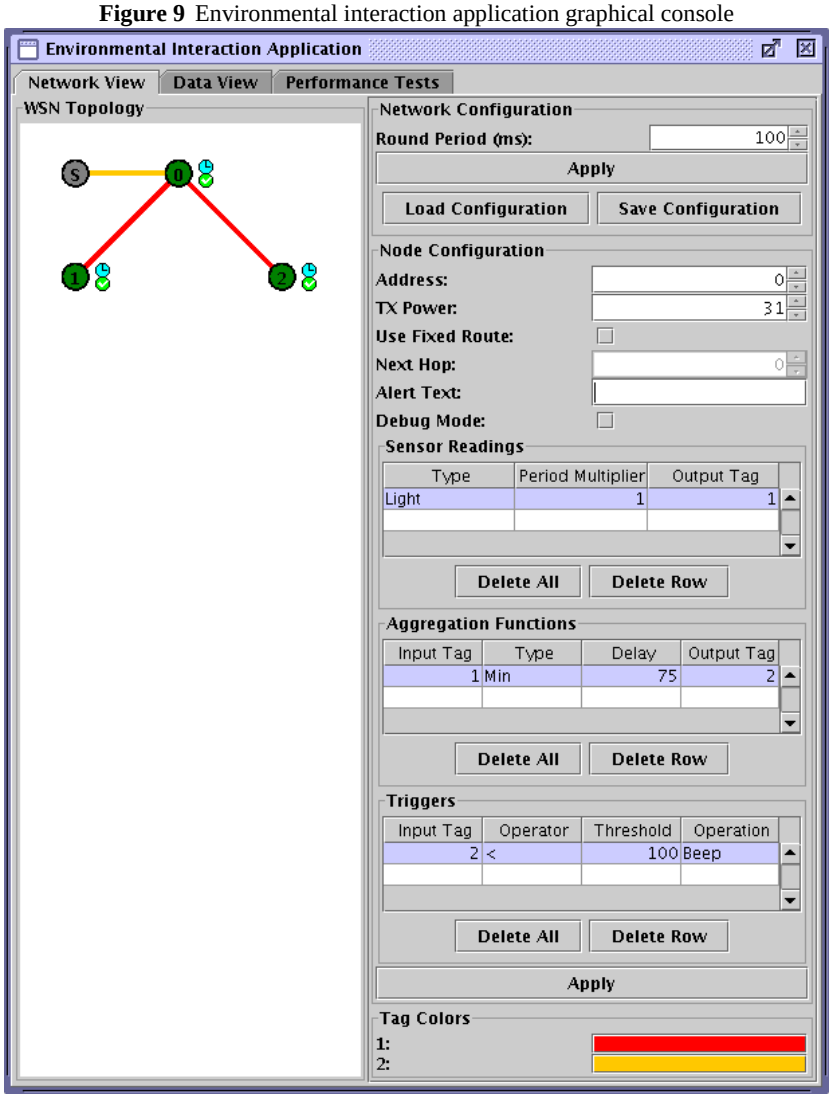
Each sensor node measures the structure's vibrations using one of two different types of sensor boards: either the MTS310 (Crossbow, n.d.b) or the LEMe Sensor Board (LEMe-SB). While the former is a Crossbow commercial product, equipped with a built-in bi-axial accelerometer, the latter is a custom sensor board developed at IST-TUL. Aside from the additional functionalities (ultra-sound transducers, and light sensors) that are used in other projects (IST NoE CRUISE, 2006b), the LEME-SB uses a tri-axial MEMS accelerometer (Kionix, Inc., 2007), and has a built-in PIC co-processor (Microchip Technology, Inc., 2003) to off-load complex or low-level signal processing tasks off the mote's main processor.

Having measured the raw unprocessed acceleration values, a remote calibrated conversion procedure is then performed, allowing the nodes to report actual acceleration values in G-force units. This conversion consists in transforming the raw sensor information, $a_{measured}$, into a calibrated value, $a_{calibrated}$, according to (1).

$$a_{calibrated} = (a_{measured} - a_0) \cdot n \quad (1)$$

where a_0 is the measured value for a null acceleration force, and n is the number of *mg* units equivalent per measured unit. These two calibration parameters are individually established for each node during an initial calibration procedure and programmed into the nodes using the unicast data delivery service. Aside from calibration, the unicast routing system is also used to configure the sensor nodes' measurement parameters, such as the sampling frequency and the enabled axis.

Figure 9 Environmental interaction application graphical console

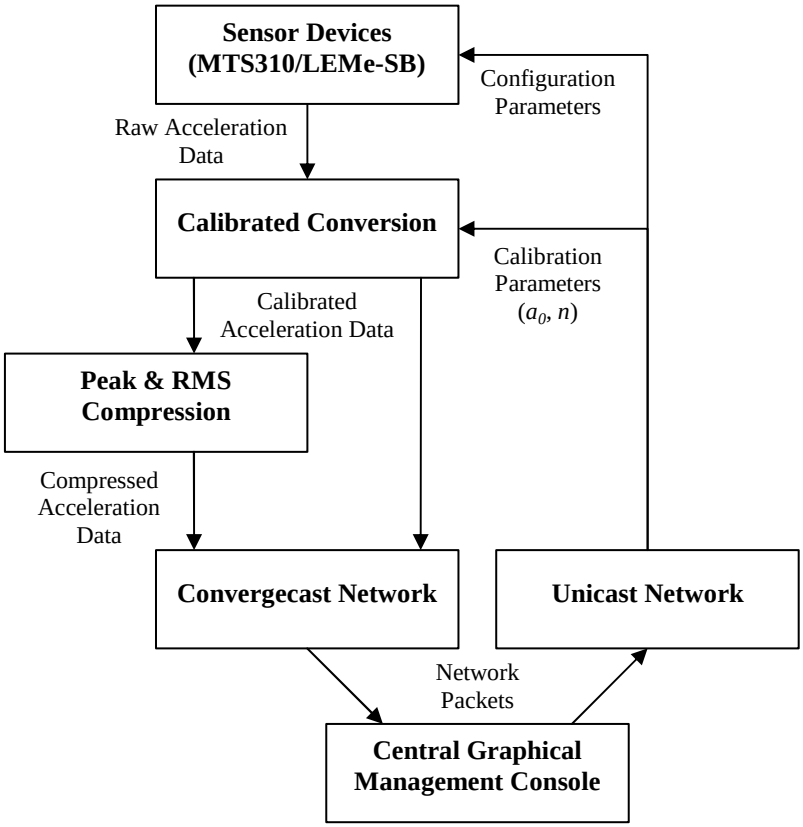


For high sampling frequencies e.g. 200 Hz (Kim, S., 2007), each sensor node outputs a huge amount of data that needs to be delivered to a sink node. Since these measurements can easily saturate the entire network capacity, each sensor node can also be configured to compress the measurement data into periodic peak and RMS values. The latter are calculated, for each axis, using a custom implementation of (2).

$$RMS = \sqrt{\frac{\sum_{i=1}^{N} a_i^{\,2}}{N}} \qquad (2)$$

Where a_i are the axis acceleration values, and N is the number of these values per each calculation. Instead of storing the N acceleration values, which implies a high usage of the nodes' RAM memory, and calculating both the peak and RMS values in a single iteration, instantaneously peaking the processor's load, the node updates the peak acceleration and accumulates its square value after each sensor reading, minimizing the calculation footprint. Once the desired number of sensor readings is accumulated, the node calculates the square root of the mean squared accelerations, and outputs both of the calculated values. Using this compression scheme, the amount of data that is simultaneously generated by each sensor node can be transferred through the entire multihop network to the sink. However, as will be shown in the experimental evaluation of the network, it is still possible to measure accelerations without this compression scheme if only a few sensor nodes are operating at the same time and in close proximity to the sink, or if the sensor nodes are otherwise configured to use a lower sampling frequency (e.g. 50 Hz per axis).

Figure 10 Vibration monitoring application architecture



Regardless of the use of compression, the sensor nodes also use the time synchronization service to timestamp their data. Although they do not cooperate with each other, beyond the relaying of messages, this synchronization is used to align the generated data from each source at the graphical application. Beyond allowing the user to identify the same structure excitement event on measurement graphics from different sources, this functionality also allows the generation of custom graphics, composed with data measured by the different sensor nodes. Finally, the graphical application also provides an offline processing functionality, allowing the generation of frequency spectrums for the measured vibrations, on demand.

5 DEPLOYING THE WSN

WSN deployments are, by nature, a relatively complex operation as one must take into account multiple, often contradicting, factors at once (e.g. the ideal sensor location for physical phenomena detection and measurement may not be ideal for radio communication, and vice-versa). The addition of these extra factors into the decision process, allied with the unreliability of current wireless channel models when applied to these networks, makes the WSN deployment phase a laborious and time consuming task.

As for the sensor hardware itself, the WSN deployed at IST-TUL was created mostly from commercially available, off-the-shelf, sensor hardware, namely the Crossbow MICAz platform (Crossbow, n.d.a). Comprising solely the processing and communications units, the MICAz motes must be coupled with additional sensing or actuating daughter boards (e.g. MTS300 and MTS310 or our own LEMe-SB boards). Furthermore, the sink node is connected to a standard Ethernet network through a specialized interface board, the Crossbow MIB600 (Crossbow, n.d.c). Finally, all of this equipment was installed within Crossbow's own plastic housing modules, the MIH2400CA and the MBH600CA.

Once all of the required mote hardware was fully acquired or built, the sensor network deployment procedure commenced. Although traditional site-surveying techniques, commonly used when deploying 802.11 networks, may be used as a general approach to the problem, in practice, we found that many of the steps that these site surveys contemplate are of little consequence in WSN deployments. On one hand, any attempt to typify sections of the network according to rigid radio link models proved to be unreliable and ultimately pointless. On the other hand, we found that, whereas the behaviour of an 802.11 radio on an access point will predictably not vary too much within a small area, the low-power 802.15.4 radios used by our sensors may go from 0% to 100% connectivity within mere centimetres. Additionally, in our case, other factors also had to be considered, as the motes were installed indoors. Aside from all the trouble that typical indoor objects were found to cause (e.g. metallic objects and thick glass were found to

considerably attenuate the radio signal, while indoor plants scattered it, leading to significant packet loss), special care had to be taken not to damage walls or structures in any possible way. Moreover, since several motes were installed in public areas, an additional effort had to be made to ensure they were beyond any curious observer's reach, while still being accessible for future battery changes. Having all of this in mind, the following steps were taken to choose each sensor node's ideal location:

1. The area where the measured phenomena occur was identified, defining the region where sensors must be installed to effectively capture ongoing events (e.g. gas sensors should be installed within the chemistry lab's store room, above the shelves where dangerous chemicals are kept). This procedure is particularly important, hence its position as the first step, since the ability to effectively measure the targeted phenomena is the network's primary purpose. If this procedure is poorly executed, the networks most basic reason to exist may be severely compromised.
2. Using a simple radio link testing application, developed specifically for the purpose, the sensor nodes were moved around the identified area, detecting the spots that allow them to best communicate with their neighbours, using a site-survey-like approach. Although the previously defined regions may optimize the physical sensing process, they may not be the ideal locations to facilitate the nodes' ability to communicate with each other. This step is used as an initial approximation to finding the nodes' optimal position, while also identifying the need to add the occasional routing node to extend the networks connectivity towards any particularly isolated regions.
3. Once the overall number of sensing and forwarding nodes was settled upon, and their initial positions approximately determined, the complete multi-hop network was tested, optimizing each sensor's position to maximize its goodput to the sink. This step was particularly complicated as the multi-hop network suffered some adverse effects that were not apparent when testing the individual hops in an isolated fashion. Additionally, when adjusting the position of a forwarding node, one must not forget that all of the upstream nodes that depend on it, as well as any neighbouring nodes, may be affected. Under these circumstances, the entire network's performance had to be systematically rechecked whenever any changes were enacted. Figure 11 shows a portion of the deployed network.

6 EXPERIMENTAL RESULTS

As previously mentioned, both the environmental interaction and the vibration monitoring applications, aside from demonstrating the network's capabilities, also evaluate its performance. The experimental results obtained from this performance evaluation are presented and discussed below.

6.1 Environmental Interaction Application

This application continuously performs a synchronous sensing / processing cycle that begins with the actual sensor readings, followed by a chain of cascading aggregation operations within the network that ultimately lead up to the final results reaching the sink node.

While the advantages of using this kind of in-network processing, over simply sending the raw data towards the sink, are self-evident, there are, in fact, some tradeoffs and inconveniences that are commonly forgotten. In the first place, when data is being aggregated, an additional delay comes into effect, namely while aggregator nodes wait for intermediate results from their children nodes, before sending out the aggregated result. Since the application is not supposed to mix together data from different rounds and due to the processing limitations and the relatively small amounts of memory available on the network nodes, these delays end up limiting the overall network performance. This imposes an upper limit to the amount of data that the network can collect and process.

To quantitatively assess this limit, a special test scenario was prepared where all of the nodes within the network were configured to read their sensor data and send it towards the sink. Additionally, each intermediate node was configured to aggregate its own data, alongside the data from its children, ultimately assuring that the sink would only receive one single aggregated value for the entire network. Under these circumstances, two distinct system parameters were varied to detect when the network was no longer capable of processing all of the data: the round period (P), which represents the duration of each sensing / processing cycle, and a special factor k, that indicates which percentage of each round may be used for processing the data. Under these circumstances, each node's aggregation delay (D) is calculated by (3), where n is the node's depth (the number of hops between the node and the sink) and N the network's depth (the depth of the most distant node within the network). The results, measured as the average number of messages dropped in each round due to a late arrival, are presented in Figure 12.

$$D = k \cdot P \cdot \frac{(N - n)}{N} \tag{3}$$

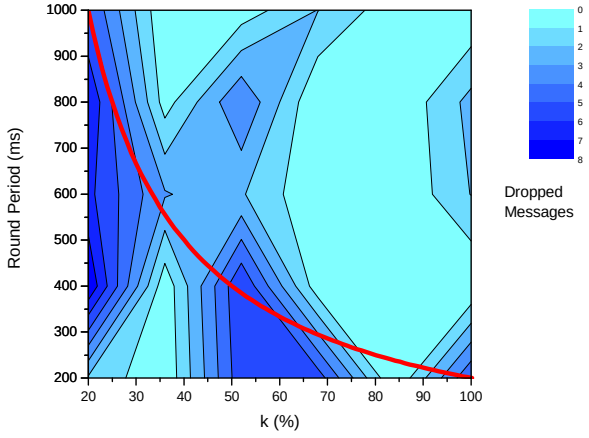
Figure 11 Some sensors deployed at IST-TUL

As seen in Figure 12, there are marginal amounts of dropped packets for scenarios with a large aggregation delay. But, once the aggregation delay falls below a certain point, be it due to a smaller round period or due to a smaller *k*, the number of dropped packets starts growing quickly, indicating that data packets are being discarded due to a late arrival. The boundary at which this occurs, roughly when (4) is true (also shown in Figure 12 as a bold line), is a key performance limitation. Basically, the network needs at least 200 ms to aggregate data. If we force it to do so in less time, information will be lost.

$$k \cdot P = 200ms \quad (4)$$

The lower left area of the graph, where the number of dropped packets begins to decrease again, corresponds to the truly extreme cases where most of the raw data is never actually aggregated at all, thus precluding the generation of further results. This scenario represents a total system breakdown as no meaningful results are being conveyed to the sink.

Figure 12 Number of dropped messages as a function of the *round period* and the *k* factor



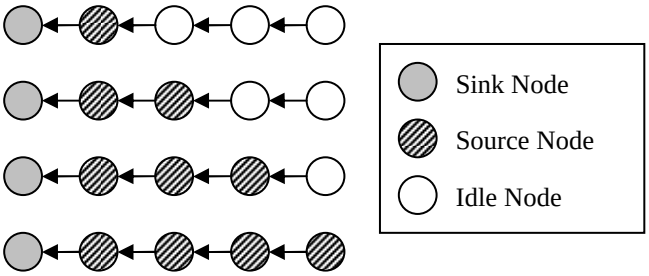
6.2 Vibration Monitoring Application

To analyze the performance of the vibration monitoring application, an experiment was performed with several sensor nodes deployed as a multihop network. In order to

better evaluate the limitations of this type of application, four sensor nodes were deployed in a linear fashion, with the first node being in range with the sink node, and the last being distanced at four hops. In this experiment, several different scenarios were considered, where the number of simultaneous active sources changed from a single node to all network nodes, as Figure 13 illustrates.

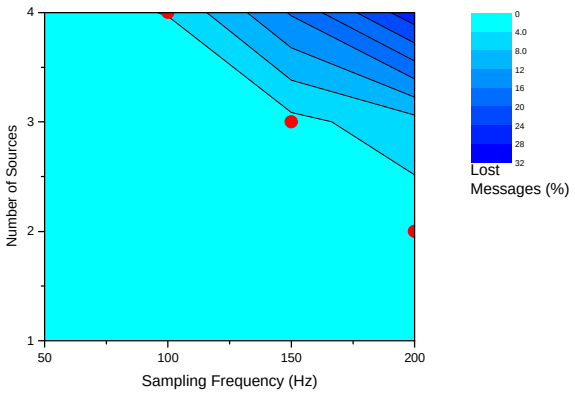
For these scenarios, four tests were performed using a different sampling frequency for all the active sources, with a duration of two minutes for each measurement. Beyond this, the two measurement modes, raw acceleration values and peak and RMS values, were used for each test case, and both the generated and lost data message statistics were logged.

Figure 13 Vibration monitoring application experimental test setup



As expected, the raw acceleration measurement mode experiments demonstrated that the maximum number of simultaneous sources greatly depends on the sampling frequency used by the sensor nodes. In Figure 14, it is possible to see that the overall network measurement data can only be reliably delivered to the sink node when using equal or "inferior" combinations for the number of simultaneous sources and sampling frequency to the ones represented as bold dots. Beyond these limits, the overall amount of lost data starts increasing at a high rate, due to the limited available bandwidth of the radio channel in the multihop network.

Figure 14 Number of lost messages as a function of the number of sources and the sampling frequency



On the contrary, the peak and RMS measurement mode behaved consistently for all the tests in each scenario, making its statistical results for the overall measurement data loss irrelevant. Due to its high data compression rate, this mode allowed all the active sensor nodes in the network to deliver all their data to the sink node, as otherwise expected.

7 CONCLUSIONS AND FUTURE WORK

Although our initial sensor deployment is not considered a large-scale WSN (23 motes installed throughout the entire campus), its installation still required a considerable effort. Not only is the deployment procedure complex and time-consuming but, as we came to find, several objects commonly found indoors (e.g. indoor plants, thick glass, etc.) interfere with the sensor network's radio communications. Under these circumstances, some of the network protocols, namely TinyOS' native routing system, may operate erratically, leading to poor connectivity or unstable network topologies. This led to the need to develop the previously mentioned static routing feature.

With the initial ordeal of installation surpassed, four applications were developed to demonstrate the network's capabilities and measure its performance: an Environmental Interaction Application, a Vibration Monitoring Application, a Temperature Gradient Map, and a Remote Monitoring and Control Application. Both the Environmental Interaction Application and the Vibration Monitoring Application were fully explained and their performance results analyzed and discussed.

Even though the Environmental Interaction Application is already fully-functional, some open issues still remain for future work. Although the motes are already capable of performing several kinds of in-network processing, as well as to make simple decisions, these operations are currently limited to simple aggregation functions and threshold checking. It would be interesting if, in the future, this framework could be extended to use more elaborate processing techniques. Ideally, the system should be powerful enough to completely automate any standard procedure entirely within the network. Regarding the vibration monitoring application, the data processing performed on the sensor nodes is still limited to simple local data compression schemes. Application performance and usability could possibly be improved with the introduction of more complex distributed data processing schemes that would generate cluster measurement values rather than independent sensor readings. Ideally, the sensors deployed at each bridge would provide the structure's natural frequencies or even its vibration modes.

As for the network's performance assessment, the results uncover some interesting technical trade-offs. The environmental interaction application has shown that there is, in fact, a limit to the amount of data that the in-network aggregation system can handle, ultimately indicating that, for our particular topology, no less than 200 ms must be used to process the data, otherwise information may be lost. The vibration monitoring application, on the other hand, has shown that, given the limited network resources that are available in typical WSNs, applications that require high

data rate multi-point vibration monitoring with real-time constraints will have serious operational difficulties. In fact, the experimental results show that data processing schemes are needed in such scenarios to guarantee acceptable data loss rates even in relatively small multihop networks.

Even though the network test-bed is, in its current form, already powerful enough to support most common WSN applications, several extensions are already being planned. On one hand, a new Power Management system is being developed to manage the nodes' energy resources while also enabling energy harvesting, where applicable. On the other, the network's functionality is being extended to support new networking paradigms, namely opportunistic networking (Pelusi et al., 2006). Once this new system is developed, mobile nodes may be used to opportunistically collect data from remote disconnected regions within the network, physically carrying it towards a connected region that will then promptly deliver the data to a sink.

8 ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers, whose valuable input and comments contributed to the article's overall quality, and the students of the IST-TUL Wireless Sensor Network's course, whose hard work contributed to the overall success of this project. A special word of thanks to the members of the Group of Embedded networked Systems and Heterogeneous Networks – GEMS, for their support.

This project was partially funded by the Instituto de Telecomunicações – Networks and Multimedia Groups.

REFERENCES

Akyildiz, I.F., Su, W., Sankarasubramaniam, Y. & Cayirci, E., 2002. Wireless Sensor Networks: A Survey. *Computer Networks*, 38(4), pp.393-422.

Camilo, T., Melo, P., Rodrigues, A., Pedrosa, L., Silva, J.S., Neves, R., Rocha, R. & Boavida, F., 2006. Wireless Sensor Network Deployment: an Experimental Approach. In: G. Aggelou, ed. Forthcoming. *Handbook of Wireless Mesh and Sensor Networking*. McGraw-Hill International

Crossbow Technology, Inc., n.d.a. MICAz Datasheet. Document Part Number 6020-0060-04 Rev. A.

Crossbow Technology, Inc., n.d.b. MTS/MDA Datasheet. Document Part Number 6020-0047-03 Rev. A.

Crossbow Technology, Inc., n.d.c. MIB600 Datasheet. Document Part Number 6020-0055-04 Rev A.

Gay, D., Levis, P. & Culler, D., 2005. Software design patterns for TinyOS. In: ACM (Association for Computing Machinery), SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems, Chicago, IL, USA 11-15 June 2005.

Gay, D., Levis, P., Behren, R., Welsh, M., Brewer, E. & Culler, D., 2003. The nesC language: A holistic approach to networked embedded systems. In: ACM (Association for Computing Machinery), SIGPLAN 2003 conference on Programming language design and implementation. San Diego, CA, USA 8-11 June 2003.

Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D. & Pister, K., 2000. System Architecture Directions for Networked Sensors. In: The 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000). Cambridge, MA, USA 12-15 November 2000.

IST NoE CRUISE, 2006a. CRUISE Handbook. Deliverable D000.1.

IST NoE CRUISE, 2006b. Report on WSN applications, their requirements, application-specific WSN issues and evaluation metrics. Deliverable D112.1.

IST NoE NEWCOM++, 2008. Core Contract, Annex I - Description of Work. Proposal Number 216715.

Karl, H. & Willig, A., 2005. *Protocols and Architectures for Wireless Sensor Networks*. Wiley

Kim, S., 2007. Wireless Sensor Networks for High Fidelity Sampling. Technical Report No. UCB/EECS-2007-91, 20 July 2007

Kionix, Inc., 2007. KXP74 Series Accelerometers and Inclinometers. Rev. 4, 24 January 2007

Maroti, M., Kusy, B., Simon, G. & Ledeczi, A., 2004. The Flooding Time Synchronization Protocol. In: ACM (Association for Computing Machinery), Second International Conference on Embedded Networked Sensor Systems (SenSys 04), Baltimore, MD, USA 3 November 2004.

Microchip Technology, Inc., 2003. PIC16F7X7 Data Sheet. Document DS30498B, 28 July 2003

Pelusi, L., Passarella, A. & Conti, M., 2006. Opportunistic Networking: Data Forwarding in Disconnected Mobile Ad Hoc Networks. *IEEE Communications Magazine*, 44(11), pp.134-141.

Tanenbaum, A., Gamage, C. & Crispo, B., 2006. Taking Sensor Networks from the Lab to the Jungle. *IEEE Computer*, 39(8), pp.98-100.