UNIVERSITÀ DEGLI STUDI DI CAGLIARI

SCUOLA DI DOTTORATO IN INGEGNERIA DELL'INFORMAZIONE

DOTTORATO IN INGEGNERIA ELETTRONICA E INFORMATICA

Settore Scientifico Disciplinare ING-INF/05

Adaptive and Power-aware Mechanisms in Wireless Sensor Networks

Dottorando: Romolo Camplani

Tutor: Giulio Concas

"Wow, a whole bucket full o' mud, AND IT'S MINE, ALL MINE!"
Guybrush Threepwood,"Monkey Island 2: LeChuck's Revenge"

Ringraziare tutte le persone che, in divesra misura, mi hanno consentito di arrivare dove sono ora forse la cosa pi difficile. Le persone da ringraziare, inoltre, sono tantissime e ricodarle tutte in poche righe impossibile. Per non scordarmi di nessuno, inanzitutto ringrazio $[a - zA - Z]+$ di vero cuore.

Permettetemi, però, di ringraziare in particolare alcune persone che mi sono state sempre vicine nei momenti fondamentali. Innanzitutto Roberta, la mia ragazza, che ha sempre appoggiato le mie scelte e mi ha incoraggiato quando mi sono sentito triste. Ringrazio, ovviamente, la mia famiglia che mi ha sostenuto in questi anni e che pazientemente ha lasciato che ricercassi la mia strada.

Ringrazio Giulio, perch mi ha sempre appoggiato e difeso anche quando non sono stato "friendly" con lui (come invece avrei dovuto essere). Ringrazio Cesare, con il quale lavoro e dal quale ho imparato tanto (eccetto l'inglese sfortunatamente...). Ringrazio Mariagiovanna, che mi ha voluto nella "squadra" e che, professionalmente parlando, considero come una madre. Ringrazio Fabio, con il quale collaboro per alcuni interessanti progetti, che ha permesso che ampliassi i miei orizzonti nella ricerca. Ringrazio Luigi, i cui consigli mi hanno consentito di scegliere per il meglio (quando li ho seguiti, ovviamente...).

Ringrazio Oreste, per me come un fratello, che é sempre stato prodigo di consigli e che mi ha sostenuto tanto. Ringrazio Manuel, amico prima che collega, con il quale ho trascorso intere domeniche di lavoro per rispettare la deadline. Ringrazio Cristian, amico e compagno di avventura agli antipodi del mondo. Come dimenticarsi anche di Umberto, lavoratore infaticabile e ingegnere coi fiocchi. Ringrazio anche i miei amici di una vita, Alessandro e Andrea, che mi hanno sempre appoggiato, sostenuto e sopportato in tutti questi anni.

Ringrazio Gigionz, JK, Paolino ed Antony che mi hanno accolto come membro di una grande famiglia. Un ringraziamento speciale va ai fratelli Monkiers per l'amicizia che mi hanno sempre mostrato. Ringrazio il mitico Ing. Penta, amico

fraterno e compagno di bisboccia nella Milano da bere...

Ricordo e ringrazio anche gli amici come il perfido matematico Boracchi, l'ing. Loiacono, prof. Roby e prof. Lazaric; i colleghi cagliaritani Alessandra, Alessio, Nicola, Sandro e Selene (quest'ultima mio braccio armato per la modulistica).

Ringrazio tutti i miei tesisti: Delle Canne, Cortellazzi, Capitanio, i tesisti fidanzati, Briggi, e il fin troppo mitico ingegnere jr. Capocefalo (il vero Gattuso dei tesisti), i tesisti Fortunato e Marelli (aka Garrone e De Rossi), Big Indian e Little Indian, e infine Alfredo. Per ultimi ma non per questo meno importanti il Santa, il Saccente, l'esperto di serial killer, Luciano e Mr. CiBi.

Tutte le persone che ho citato non sono che una piccola parte di quelle che ho conosciuto in questi anni e con le quali ho vissuto piccole e grandi esperienze, che mi hanno permesso di crescere come uomo ed ingegnere. E' questa la cosa più importante. Il resto é mancia.

sinceramente vostro,

`romolinux`

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In a very short time, the interest about Wireless Sensors Networks (WSN) and their applications has grown both within the academic community and in the industry. At the same time, the complexity of the envisaged WSN-based systems has grown from a handful of homogeneous sensors to hundreds or thousands of devices, possibly differing in terms of capability, architecture, and operating system.

Recently, some deployments of WSNs have been suggested in the literature both addressing the energy-aware and the adaptability to environmental changes issues; in both cases limitations arise which either prevent a long lifetime or/and the Quality of Service (QoS) of the envisaged applications.

Energy is one of the scarcest resources in WSNs; energy harvesting technologies are thus required to design credible autonomous sensor networks. In addition to energy harvesting technologies (which convert different forms of environmental energy into power supply), energy saving mechanisms play an important role to reduce energy consumption in sensor nodes.

Moreover, in wireless sensor networks, the network topology may change over time due to permanent or transient node and communication faults (units live in a harsh, highly non-stationary environment), energy availability for the nodes (despite the possible presence of energy harvesting mechanisms) and environmental changes (e.g., a landslide phenomenon, solar power density made available to

the nodes, presence of vegetation subject to seasonal dynamics). Development of smart routing algorithms is hence a must for granting effective communications in large scale wireless networks with adaptation ability combined with energy-aware aspects.

1.1 Research aim and thesis goal

This thesis aims at developing a methodology for the design of WSNs oriented towards a credible deployment. Here, credible has to be intended as an energy-aware system, robust with respect to perturbations affecting the normal operational life that adapts itself to face a change in the network topology. The need to develop a methodology for the design of a credible deployment of WSN-based applications has pushed the research to deal with technological and applicative issues. As regards the technological issues, the work has been carried out both at the network and at the node level.

In the specific domain of the *network level*, the thesis focuses on MAC and routing layers.

As regards the MAC layer, the TDMA approach is particularly interesting in applications where sensor nodes transmit at predefined time slots hence granting an efficient control of the radio module. Unfortunately, TDMA is not specific for WSNs, since it does not support adaptation to topological changes and, moreover, it is not power-aware (e.g., in a traditional TDMA the radio module of the gateway remains active for all available time slots even when less sensor nodes are available). In the thesis an adaptive power-aware TDMA is suggested which includes power-aware and network scalability issues (in Chapter 2). In particular, the adaptive power-aware TDMA provides an efficient radio management both of the sensor nodes and the gateway. A robust registration phase has been included in the developed protocol to allow an efficient insertion/removal of sensor nodes in/from the network.

As far as the routing layer is concerned, many self-organizing routing algorithms have been proposed in recent years for wireless sensor networks, which exploit different features and *a priori* information about the deployment. Among these, hierarchical routing algorithms such as Low Energy Adaptive Clustering Hierarchy (LEACH), Localized Clustering (LC) and Low-energy Localized Clustering (LLC) are particularly interesting for scalability, power-efficiency, extended network life and intrinsic adaptability. Chapter 3 provides a k-level hierarchical extension to Low-Power Localized Clustering (LLC). The first novel content of the suggested approach is that the proposed routing algorithm takes into account the effects introduced by finite unit bandwidth on the routing capabilities. Moreover, as a second novel content, differently from the existing algorithms the proposed routing algorithm provides a uniform distribution of alive units in the deployment area (feature associated with the QoS of the network over time).

The contributes of this thesis at the *node level*, presented in Chapter 4, can be summarized as follows: a framework for the start and restart of the nodes that use both detailed information about residual energy and a major change in core part of the nodes OS to outperform the standard power management policy, hence, to increase the node life. The suggested improvements, together with ad-hoc hardware, allow the development of a novel generation of power manager systems for WSN units. In fact, a joint use of specialized hardware and super-capacitors technology allows one to know the residual energy of the node. This information has been used to trigger the power manager. Instead of a complete node shutdown, the proposed power manager can turn-off system modules (e.g., radio). However, if the energy level is critically low, the power manager hibernates the node by saving its internal state (e.g., tasks, memory). The specialized hardware wakes up the node when the energetic level in the super-capacitors allows a normal operational life of the node. Then, the node restores its previous internal state. To achieve this goal essential parts of the node OS (i.e., scheduler, power manager) have been modified and the support for the MPPT hardware has been integrated.

At the *application level*, the research focuses on the definition of a language and a middleware for managing data in highly adptive and pervasive systems, such as WSNs, made of very different devices as to their technology and functional capabilities (Chapter 5). The language has been defined in order to manage both functional features, comprising the definition of operations which manipulate raw data to generate the query output and statements for the setting of sampling parameters, and non-functional features, which account for constraints on the offered functionalities and on the Quality of Service (QoS); in WSNs the QoS is mainly related to power management, however node latency and sensors availability are considered as well. Functional and non-functional requirements are dealt with in a transparent mode by a SQL like interface.

A wide experimental campaign, which is presented in Chapter 6, has validated the proposed design methodology. We considered two real monitoring applications:

- a real monitoring application envisaging a prototypal deployment of the WSN designed for monitoring the Australian Coral Reef. The prototype application aims at monitoring temperature and lightness in the water at different depths in Moreton Bay (Brisbane, Australia).

- a real civil buildings and unstable cliffs monitoring application for the evaluation of the geological risk in a rock faces collapse scenario. The micro acoustic signal detection and compression algorithms, suggested in this thesis, perfectly fits in this application.

1.2 Novel contributions

The main novel contributions of this thesis are:

1. a power-aware and adaptive TDMA-based MAC layer for the local transmission (i.e., in clusters) that guarantees robustness and adaptability to network

changes in terms of topology (e.g., due to insertion or removal of new nodes caused by energy availability). This work has been accepted per publication at the IEEE IMTC 2008 international conference [1];

2. an extension of the LLC routing algorithm by introducing a hierarchical structure in the network management. The proposed algorithm is particularly appealing for its ability to maintain a uniform distribution of alive nodes in the deployment area, feature associated with the monitoring QoS, where other routing algorithms introduce vast areas not covered by sensor nodes. This work has been presented at the IEEE International Workshop on RObotic and Sensors Environments (ROSE 2007) [2]. An extended version of this work has been submitted to the IEEE Transactions on Instrumentation and Measurement [3];

3. a framework for the start and restart of nodes that, by modifying essential parts of the node OS (i.e., scheduler, power manager) and integrating the support for specialized hardware (i.e., MPPT), allows to save the internal node state (e.g., tasks, memory) when residual energy prevents the normal operation to the nodes. Conversely, when the energetic level overcomes a threshold, the node re-start and restore its previous internal state;

4. a language and a middleware for managing data in highly adaptive and pervasive systems, such as WSNs, made of very different devices as to their technology and functional capabilities. Functional and non-functional requirements are dealt with in a transparent mode by a SQL like interface. This work has been accepted to the IEEE PERCOM 2008 international conference [4].

5. a novel detection and compression technique for micro acoustic signals. This technique allows to reduce the energy consumption by transmitting the signal at the base station through an event selection and compression algorithm. An hardware implementation of the proposed algorithms on a FPGA is also

presented. Effectiveness of the algorithms shows a transmission reduction up to 95%. This work has been presented at the IEEE International Workshop on RObotic and Sensors Environments (ROSE 2007) [5].

1.3 Thesis organization

The thesis organization is as follows.

Chapter 2 presents the TDMA-based MAC layer with the node insertion/removal issues. Chapter 3 reviews the network level mechanism at routing layer, in particular suggested extension to LCC with the multi-level hierarchical architecture with bandwidth, power consumption and load balancing constraints.

In Chapter 4, the framework for the start/restart of nodes (together with the novel power management policy) is presented.

Chapter 5 proposes a middleware architecture for the integration of a pervasive system (i.e., a WSN). In particular, the language functional and non-functional features with the query processing mechanism is introduced.

In Chapter 6, the proposed framework is considered in two real monitoring applications (an environmental monitoring -Australian coral reef- and a homeland security application -unstable rock cliffs-). Finally, the future works and the conclusion are discussed in Chapter 7.

Chapter 2

Network Level mechanism: MAC layer

2.1 Introduction

Environmental monitoring with wireless sensor networks (WSNs) is one of the most challenging research activities faced by the intelligent and distributed measurement community in last decades [6]. The need to have pervasive and accurate monitoring systems pushes the research towards the realisation of credible deployments able to survive in harsh environments for long time. As a consequence, basic research issues have to be integrated with technological constraints requiring multi-disciplined competences.

In recent years some deployments of WSNs have been suggested in the literature both addressing the adaptability to environmental changes and energy-aware issues; in both cases limitations arise which either prevent a long lifetime or/and the QoS of the application.

For instance, [7] and [8] present a star-based topology for seabirds habitat and volcano monitoring, respectively (the gateway collects data from the sensor nodes and forwards them to a remote control station for further processing). The first perception that energy scavenging is a fundamental aspect for a credible deploy-

ment has been pointed out in [7], where simple solar harvesting mechanisms have been envisaged (only at the gateway level, leaving sensor nodes battery powered). In the proposed WSN thirty-two sensor units are deployed for four weeks while the authors estimate in six months the lifetime of the network. In [8] a system for monitoring volcanic eruptions has been suggested; no energy harvesting solutions are provided neither for the gateway nor for the three sensor nodes. The deployment of the proposed WSN (involving the gateway and three sensor nodes) was active only to collect 54 hours of infrasonic signals. A more complex WSN architecture is proposed in [9] where a multi-hop WSNs for wildland fire environment Monitoring is proposed. The limited adaptation ability of the presented monitoring system requires human intervention for introduction of new nodes, which are battery powered.

It immediately arises that a credible deployment requires sensor nodes and gateways to be equipped with energy harvesting mechanisms. In this direction, since the maximum power density obtainable from a modern solar cell is about $5 - 20mW/cm^2$ (outdoor, sun at the zenith) whereas all other sources provide an energy gain far below $1mW/cm^2$ [10], solar energy is the most adequate energy supply mechanisms for most of outdoor applications. In this direction, [10], [11], and [12] suggest to use commercial silicon solar cells and an on-off charging scheme based on the solar power. While the system is effective in optimal sun conditions, the efficiency drastically falls (the charging mechanism goes off) when a direct strong light is not granted (e.g., in presence of a partly cloudy sky, mist, morning and late hours, etc). To grant an effective energy harvesting for the most relevant energy source it is therefore necessary to consider a Maximum Power Point Tracker (MPPT), a circuit which continuously monitors, forecasts the light conditions and consequently adapts the solar cell working point to maximise energy transfer to accumulation means. The system allows the unit for harvesting energy even if the cell is not directly exposed to the optimal radiation, as it happens in outdoor applications where the panel surface may become dusty or covered

with water. We implemented, in our units, the adaptive mechanism for tiny solar cells suggested in [13]. In turn, consideration of energy harvesting aspects implies that units need to be disconnected from the network for lack of energy and reconnected once energy goes back to a sufficient level. These adaptation aspects have an immediate impact on the network node clustering and the local communication protocol.

In its simplest architecture a WSN is characterised by a star topology with sensor nodes sending their measurements directly to the gateway (single-hop transmission). More sophisticated architectures would involve a hierarchical structure for the network (e.g., see Figure 2.1) with nodes locally organized in clusters, each characterised by a star topology. Gateways forward the collected information to a second level gateway; a multi-hop approach would constitute a different option.

The first (local star topology) and the second level of the network may use the same transmission protocol (e.g., the one presented in Section 2.2) or may differentiate themselves based on the application needs. A traditional hierarchical TDMA-based solution would be particularly appealing for simplicity and energy efficiency but surely it is not adequate in a network topology adaptive framework and it might suffer from intra-cluster frequency interferences. While the latter issue can be solved by considering the frequency allocation mechanism proposed in [14], the former requires design of an ad hoc TDMA-based protocol able to deal with adaptation in the network topology, yet keeping in mind energy savings aspects. Particular attention has been devoted to the design of a novel low power transmission protocol for local robust transmission (sensor nodes to gateway) able to support node connection and disconnection as well as deal with a large class of faults. Such a protocol will be presented in Section 2.2.

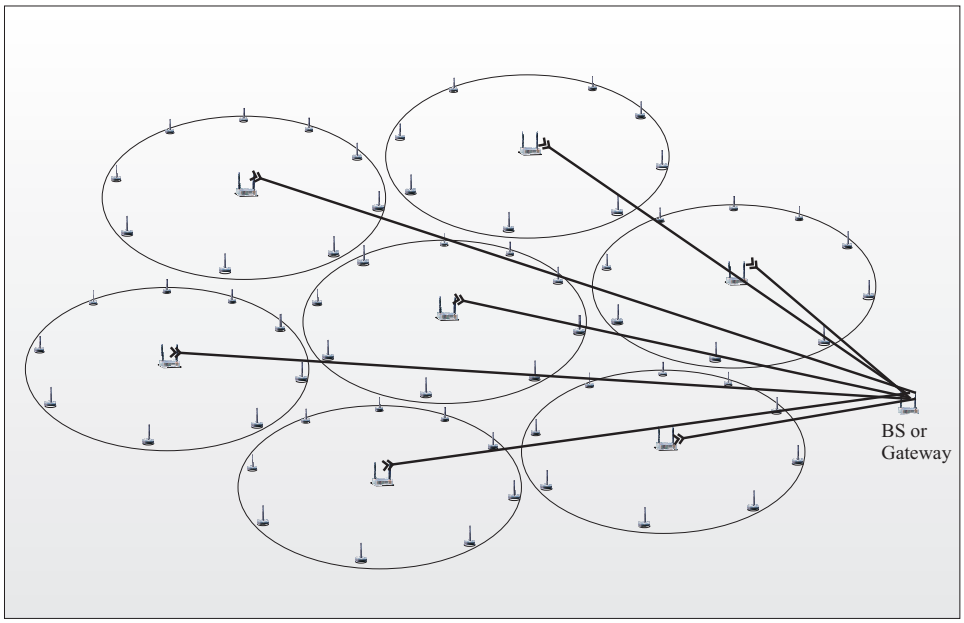


Figure 2.1: Hierarchical Topology

2.2 Local transmission protocol

Measurements acquired by each sensor node have to be collected by the gateway for the remote transmission. Unfortunately, the transmission could be corrupted by noise and the network topology could change due to the unpredictable removal or introduction of nodes (e.g., due to the extension of the deployment are or to persistent or transient fault on one sensor node or, again, to lack or availability of energy).

Moreover, the limited energetic resources present in the sensor nodes (e.g., batteries or supercapacitors) require simple power-aware routing algorithms able to guarantee a reduced and optimised access to the radio module.

In the considered framework, robustness and efficient energy management are thus two fundamental elements of the of local transmission protocol for the wireless sensor network.

Several routing algorithms are present in the literature. In SMAC (Self Orga-

nization Medium Access Control) [15] all nodes select a transmission frequency to communicate with *adjacent* nodes. Unfortunately, the most common off-the-shelf sensor units (e.g., MICA units [16]) cannot receive simultaneously more than one frequency (no FDMA) and the overhead introduced by the SMAC protocol is not justified for the topology we are considering. Eavesdrop And Register (EAR) [15] is an interesting protocol able to manage both fixed and mobile sensor nodes but its use would be overdimensioned for the proposed network. A Hybrid TDMA-FDMA [17] allows us for combining TDMA and FDMA approaches by transmitting more data in one time slot acting on different frequencies. Again, the FDMA cannot be considered on MICAz units. The Carrier Sense Multiple Access (CSMA) avoids message collisions by listening the channel before each transmission; the approach is not power-aware (the energy consumption of the radio in the receiving mode is comparable to the one in transmission and the radio must be kept on). The TDMA approach [17] is particularly interesting in applications where sensor nodes transmit at predefined time slots hence granting an efficient control of the radio module. Unfortunately, TDMA is not specific for WSNs, it does not support adaptation to topological changes and is not power-aware oriented (in a traditional TDMA the radio module of the gateway remains active for all available time slots even when less sensor nodes are available).

Alternatively, we could have considered the ZigBee protocol [18]. Unfortunately, this off-the-shelf protocol does not support a fine management of the radio module, which is autonomously managed by the protocol itself. To reduce the transmission/reception power consumption of the WSN units we require to access the radio module with the lowest admissible frequency and simplify the protocol by minimizing the size of the message and the volume of message exchange.

What is here proposed is a modified TDMA method by including power-aware and network scalability issues. In particular, the suggested *power-aware TDMA* provides an efficient radio management both of the sensor nodes and the gateway (the gateway only listens the messages of the sensor nodes connected to the net-

work). A robust registration phase has been included in the developed protocol to allow an efficient insertion/removal of sensor nodes to the network.

The designed power-aware TDMA can be formalized through the finite state machines of figures 2.2 (sensor nodes) and 2.3 (gateway). The protocol acts as follows.

2.2.1 Sensor node

Each sensor node starts from the INIT state. In this initial state the node has no information about the state of the gateway and its transmission time slot. The sensor node turns on the radio in a transmission (TX) mode and sends a SUBSCRIBE message to the gateway to signal its presence and be included in the gateway TDMA table. Once the message has been sent, the sensor node commutes the radio to the reception (RX) mode and waits for an ACK message from the gateway. If the ACK message does not arrive within ACK_TIMEOUT seconds, the sensor node turns off the radio, sleeps for RETRY_TIMEOUT seconds and returns to the initial INIT state.

If the sensor node receives the ACK message, the gateway has registered the sensor node in the network and modified the TDMA table accordingly. Moreover, in the ACK message, the gateway signals to the node the amount of time (DUTY_DELTA) it can sleep (turning off the radio) up to the next synchronization phase (which is the first phase of each TDMA cycle). After DUTY_DELTA seconds, the sensor node passes to the WAIT_SYNC state and turns on the radio in RX mode. If the SYNC message does not arrive within SYNC_TIMEOUT seconds, the sensor node moves into the LOST_CYCLE state, turns off the radio, sleeps for CYCLE_TIMEOUT seconds and finally wakes up again in the WAIT_SYNC state. If a sensor node misses three consecutive SYNC messages, it disconnects itself from the network and starts from the INIT state.

When the sensor node receives the SYNC message, it passes to the SYNCHRONIZED state. The gateway includes in the SYNC message the information

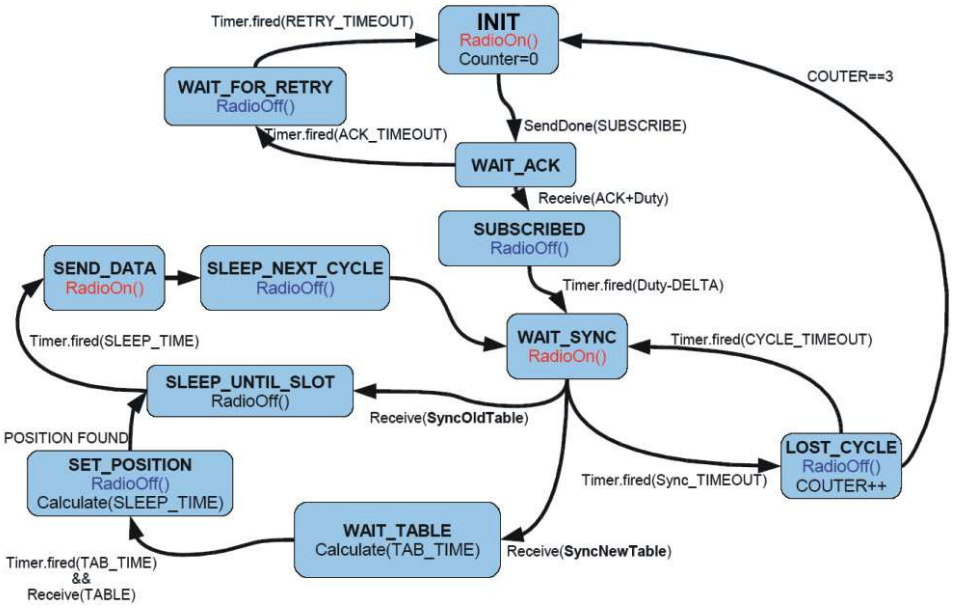


Figure 2.2: FSM of the sensor node protocol

whether a new TDMA table is arriving (due to a network topology change in the previous TDMA cycle) or not (no change happened). In case of a new TDMA table, the sensor node moves to the WAIT_TAB state and waits the TAB message from the gateway for TAB_TIMEOUT seconds. If the TAB message is not arrived at the end of the timeout, the sensor node disconnects itself from the network and passes to the initial INIT state. When the TAB message arrives, each sensor node identifies its own transmission time slot and computes the amount of time (SLEEP_TIME) it can sleep up to the next transmission (SLEEP_UNTIL_SLOT state). If the SYNC message does not signal the arrive of a new TDMA table, the sensor node passes directly to SLEEP_UNTIL_SLOT state.

After SLEEP_TIME seconds, each sensor node turns on the radio and transmits its own message. Then, it turns off the radio and sleeps (SLEEP_NEXT_CYCLE) up to the next synchronization phase.

2.2.2 Gateway

The gateway starts in the INIT state, turns on the radio and waits for a SUBSCRIBE message from a sensor node. This approach is not power-aware for the gateway that may remain for long time in RX mode. On the contrary, this limitation is compensated by the power-aware approach of sensor nodes that minimize the energy consumption of the radio in the registration phase. This approach is also justified by the fact that the gateway has higher energy capabilities (larger solar panels) than sensor nodes due to the augmented energy needed by the radio link for the remote transmission.

When the gateway receives a SUBSCRIBE message, it activates a timer that will fire after PERIOD seconds, moves into the REGISTER_NODE state, updates the TDMA table with the just registered sensor node and sends back the ACK message. When the gateway reaches the WAIT_FOR_SUBSCRIPTION state it sets the radio in RX mode waiting for subscription of other sensor nodes. If a SUBSCRIBE message arrives, the gateway returns to the REGISTER_NODE state and the registration of the new sensor nodes is the same per the first one.

After PERIOD seconds from the subscription of the first node, the gateway moves to the SEND_SYNC state, sets the radio to a TX mode, broadcasts the SYNC message (that contains the information regarding the necessity to send a TDMA table or not) and activates a TAB_TIMEOUT timer. Then, in case of TDMA table transmission, the gateway reaches the SEND_TABLE state, broadcasts the updated TDMA table, turns off the radio and moves to the RADIO_SLEEP state. When the gateway does not need to send a new TDMA table, it achieves directly the RADIO_SLEEP state.

When the TAB_TIMEOUT timer generated an interrupt, the gateway moves into the WAIT_FOR_DATA state, sets the radio to RX mode, and waits for the data messages coming from the registered nodes. Each time a message arrives, the gateway moves to the REGISTER_DATA state, stores received data in a specific memory location and returns to the WAIT_FOR_DATA state.

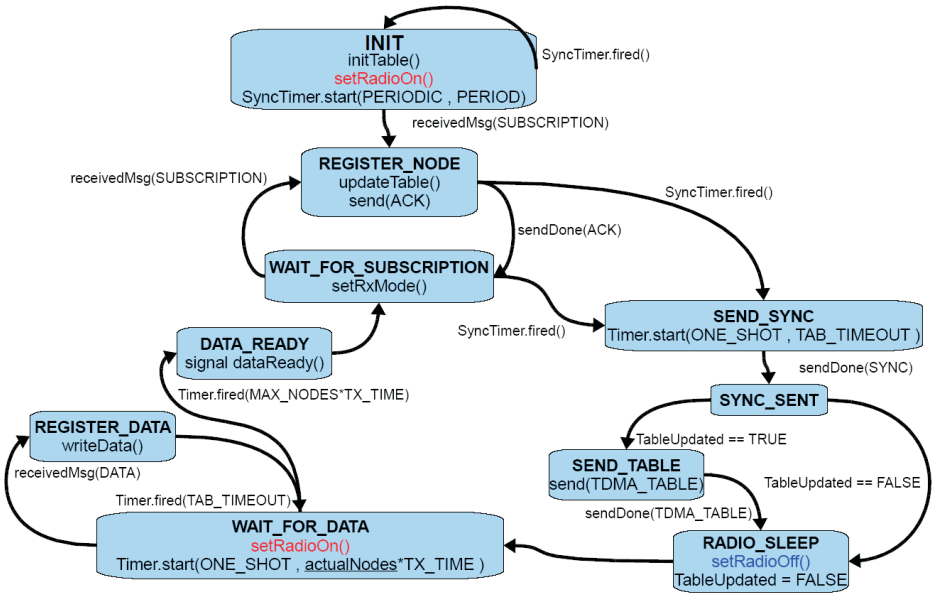


Figure 2.3: FSM of the gateway protocol

After CURR_NODES*TX_TIME seconds (i.e., the sum of all the time slots of the registered nodes), the gateway gains the DATA_READY state which means that all data have been collected by the sensor nodes in this TDMA cycle. Then, the gateway returns to the WAIT_FOR_SUBSCRIPTION state to allow new sensor nodes for registering to the network.

2.2.3 The power-aware TDMA protocol: Robustness issues

As presented in Section 2.2, robustness is a key aspect in the local transmission phase. With this goal in mind we designed the suggested TDMA protocol so as to be robust both to error transmissions (e.g., one or more messages did not reach the recipient) and to topology changes (e.g., one node or the gateway is momentarily not reachable; node fault).

In particular, the suggested power-aware TDMA protocol is robust w.r.t. transmission errors that can cause the loss of the SUBSCRIBE message from a sensor node (see Figure 2.4.a), loss of the SYNC message by a sensor node (see Figure

2.4.b), loss of the TDMA table by a sensor node (see Figure 2.4.c) and loss of a DATA message by the gateway (see Figure 2.4.d). In case of loss of the SUBSCRIBE message (Figure 2.4.a), the node unit relies on a retry mechanism that keeps on sending the subscription in the next cycles up to the final accomplishment. On the contrary, if the node unit does not receive the SYNC message, it retries the reception for RETRY times (in our case we set RETRY $= 3$) and it returns to the INIT state. When a node unit waits the TDMA table and a transmission error corrupts the transmission (Figure 2.4.c), it disconnects itself from the WSN since it would be not aware of the new transmission order. After the disconnection, the node unit returns in the INIT state. If the DATA message does not reach the gateway (Figure 2.4.d), the protocol changes over the next node without affecting the whole data acquisition of the TDMA cycle.

Moreover, the power-aware TDMA protocol is able to manage those cases in which the gateway switches on after the sensor nodes (see Figure 2.5.a), the gateway temporarily switches off while the network is synchronized (see Figure 2.5.b) and the gateway switches off before sending the TDMA table (see Figure 2.5.c).

The protocol does not require the gateway to be switched on before the nodes since the registration phase is activated periodically at the nodes up the complete subscription (SUBSCRIBE message sent and ACK message received (Figure 2.5.a). As explained above, when the node does not receive the SYNC message, it aims at receiving RETRY times and then returns to the INIT state. Thus, if the gateway switches off when the network is synchronized, the nodes will wait for the SYNC for RETRY times and then they will return to the INIT model (Figure 2.5.b). When the gateway wakes up, the network is exactly in the situation presented in Figure 2.5.a (the gateway switches on after the sensor nodes). If the gateway switches off before sending the TDMA table (Figure 2.5.c), the sensor nodes disconnect themselves from the network and return to the initial registering phase.

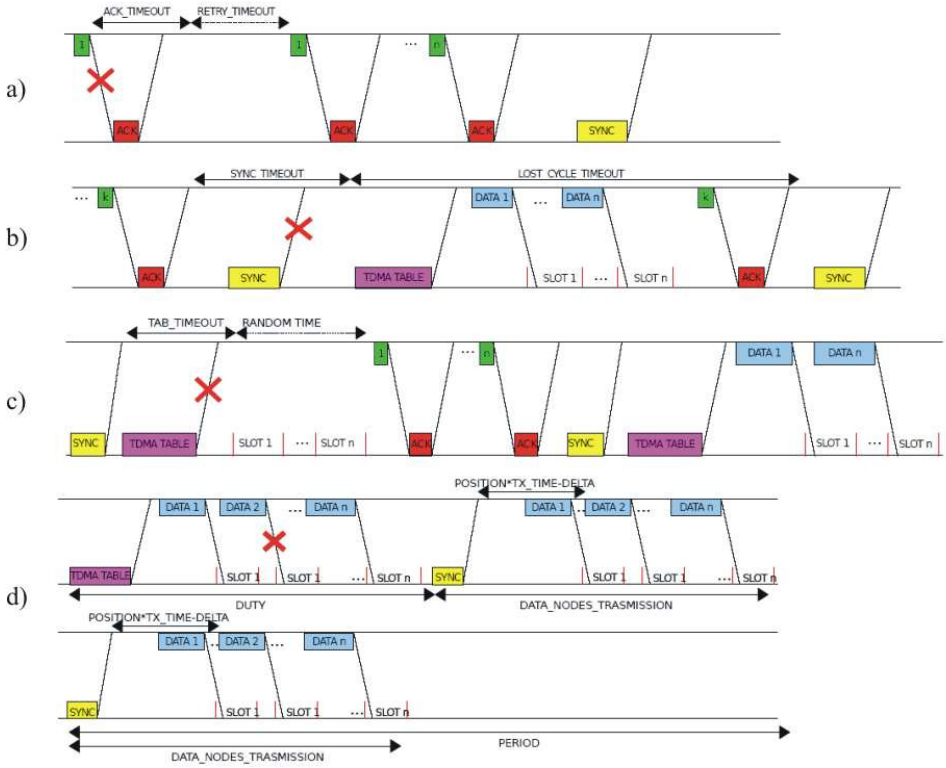


Figure 2.4: Protocol robustness to transmission errors: a) loss of the SUBSCRIBE message; b) loss of the SYNC message; c) loss of the TABLE message; d) loss of the DATA message.

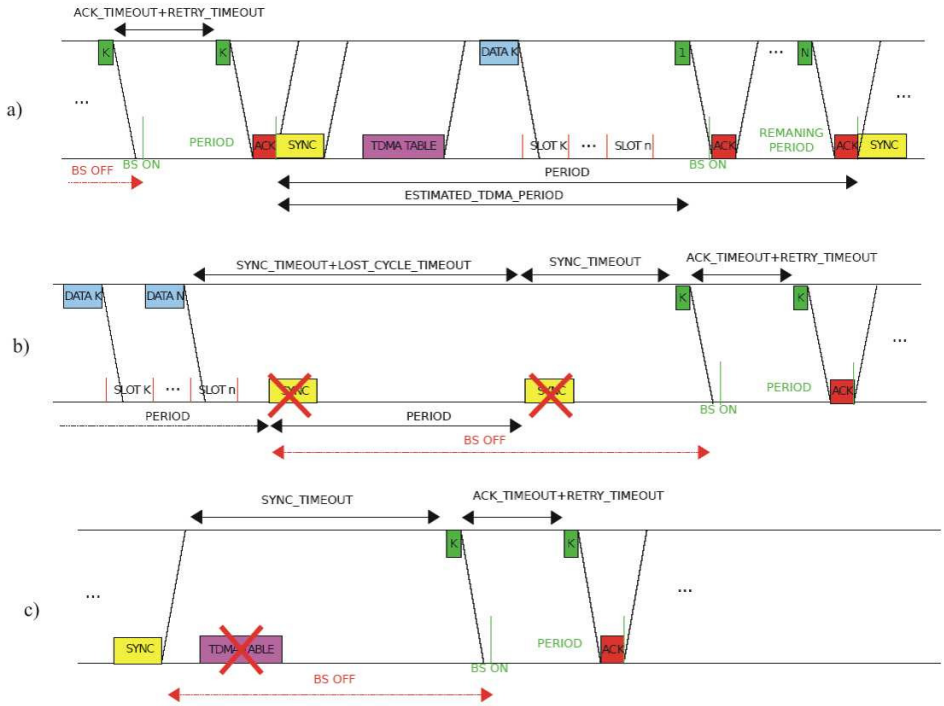


Figure 2.5: Protocol robustness to gateway faults: a) the gateway switches on after the sensor nodes; b) the gateway temporarily switches off while the network is synchronized; c) the gateway switches off before sending the TDMA table.

Chapter 3

Network Level mechanism: Routing layer

In the previous Chapter, the thesis presented the issues of energy-awareness and of adaptivity to network topological changes at the MAC layer level; in this Chapter the focus will be on the Routing layer.

In general, the network topology of a Wireless Sensor Network (WSNs) changes over time due to unit and communication faults (permanent or transient), energy availability in the units and environmental/electromagnetic changes.

As already stated in the previous Chapters, an adaptive communication mechanism among units becomes the necessary action for granting a credible and reliable distributed wireless measurement system which, in turn, requires smart -energy aware- routing algorithms to provide communication in large WSNs.

In this direction, many self-organizing routing algorithms have been proposed in recent years; among these, hierarchical routing algorithms such as Low-energy Localized Clustering (LLC) are particularly interesting for scalability, power-efficiency, extended network life and intrinsic adaptability. This chapter suggests a k-level hierarchical extension to LLC. Differently from existing routing algorithms, the proposed one takes into account the effects of a finite bandwidth for the units and grants a more uniform distribution of alive units within the deployment area (fea-

ture associated with the QoS of the distributed measurement system). The effectiveness of the proposed solutions has been validated through simulations.

3.1 Introduction

In wireless sensor networks the network topology may change over time due to permanent or transient node and communication faults (units generally live in a harsh, highly non-stationary environment), energy availability for the nodes (despite the possible presence of energy harvesting mechanisms [13]) and environmental changes (e.g., a landslide and terrain assessment, solar power density seasonality, presence of wind, vegetation ...).

In turn, all these -possibly combined- phenomena affect the performance of the network ensemble by impairing its communication ability, energy consumption and nodes lifetime. In other words, the quality of service of the environmental monitoring network is impaired and acquired measurements cannot reach the remote control station (in current deployments, since a reliable communication protocol is not envisaged due to the power consumption overhead, acquired data are sent with a best effort modality and not received measurements are lost data).

Development of smart routing algorithms is hence a must for granting effective communications in large scale wireless networks with adaptation ability being the key issue combined with energy-aware aspects. In this direction, source-initiated [19] routing algorithms become excellent candidates for their intrinsic ability to adapt to changing situations even if, in their traditional form, they cannot be accepted for the complexity of the routing algorithm and the associated high power consumption. Moreover, they do not exploit information related to the node status (e.g., residual energy of the node, computational complexity of the code in execution and memory usage), e.g., see [20] and [21] or locality indications of the node position (e.g., Received Signal Strength Indicator (RSSI) sensor [22] or Global Position System (GPS) [23]) which would allow us for designing location

aware, power-efficient protocols.

To deal with this issue self-organizing routing algorithms (a source-initiated routing algorithm family) have been specifically developed for WSNs. These algorithms grant adaptability to environmental changes by providing a quick reaction to topological modification at the network level and, particularly interesting, can support an easy node add-on and removal modality with a tunable frequency. In particular, routing algorithms that aim at reducing the overall consumption of the network energy, defined as the sum of residual energies of nodes [24] are particularly appealing solutions for effective distributed measurement systems. Here, the specific literature has suggested LEACH [25], LC [26] and LLC [27], HEED [28] and Min Max D-Cluster [29].

These power-aware algorithms share a common basic philosophy: the network topology is partitioned into clusters (divide et impera philosophy). Each cluster is ruled by a *cluster head* that coordinates (impera) the cluster nodes, e.g., by receiving messages from the cluster nodes with a single hop communication modality (star network topology). Each cluster head is then directly connected to the base station with a star topology and sends acquired data through a single hop communication mechanism. Since the network is organized without any hierarchy, the communication power delimits the coverage area. The side effect, which might become a limit, is that only small/medium monitoring areas can be controlled since the cluster heads have to reach the base station with a direct connection. However, whenever not addressed by the protocol, we could adopt a multi-hop algorithm to connect cluster heads at the highest hierarchical level to deliver data to the base station.

In this chapter we propose the *X-LLC* algorithm which extends LCC by considering a k-level hierarchy for the sensor nodes. The algorithm guarantees a more uniform distribution of the alive nodes in the monitoring area (the environment can be monitored even if some units disappears and maps of the studied entities can be reconstructed at the base station with a suitable data processing) and, at the same

time, reduces the energy consumption of the whole network (the distributed measurement system can prolong its operational activity in time). The basic tree-like structure is such that sensor nodes send data to the first level cluster heads that, in turn forward them (together with their acquisitions) to the 2-nd level cluster heads up to the k-the level where, finally, cluster heads send data collected over the tree to the base station. The structure of the cluster heads is hierarchical: a cluster head belongs only to one level of clusters and is associated only to one upper level cluster head. The introduction of intermediate levels between simple nodes and base station reduces the energy consumption since the distance between cluster heads of different levels is always smaller than that between traditional cluster heads and base station (short range activity). A further significant advantage of a multilevel solution is that cluster heads at certain levels can envisage compression techniques with a further power consumption reduction.

In the suggested algorithm identification of clusters and election of cluster heads (at the different levels) is performed with a simple -yet effective- low power consuming distributed algorithm which provides the network with high adaptation ability to the changing environment. Such self-organization of the network nodes is dynamic and allows nodes for being inserted in or removed from the network on the fly; detection of new topology configuration is synchronous, with a frequency tunable by designers.

A second novel content of the proposed algorithm is that, differently from the existing literature, we take into account effects posed by finite node bandwidth which traditional algorithms unrealistically assume to be infinite (hypothesis very far from being satisfied). The bandwidth limit is critical in WSNs nodes up to the point that available routing algorithms are unfeasible on current days node technology for a large class of applications.

The structure of the chapter is as follows. Section 3.2 reviews and compares the LEACH, the LC and the LCC routing algorithms. Section 3.3 presents the suggested extension to LCC with the multi level hierarchical architecture with band-

width, power consumption and load balancing constraints. Experimental results are finally given in section 3.4.

3.2 Hierarchical Routing Algorithms for WSNs

Cluster-based routing algorithms are a subset of the self-organizing routing ones designed to grant scalability, power-efficiency, and long system life for the network units. The philosophy onto which these algorithms rely (and make them particularly suitable to address communication in a changing environment) is characterised by three independent operational steps:

1. clustering: nodes communicate with neighbours to dynamically identify cluster heads;
2. network topology creation: neighbourhood information is used to generate a hierarchical tree-based communication structure where leaves are the nodes and the base station is the root of the tree;
3. operational: acquired data are dispatched along the tree structure to the base station.

We decided to consider LEACH, LC, LLC for their adaptability to the changing environment (making them effective in the dynamic insertion or deletion on nodes associated with the first two phases) and for their scalability (in terms of number of nodes). Before introducing the suggested extension to LLC we briefly review the selected algorithms and discuss their main features.

3.2.1 LEACH: Low Energy Adaptive Clustering Hierarchy

In the clustering phase LEACH activates an election process for identifying cluster heads within the network. To do this, each active node of the network generates a random number between 0 and 1; such a value is then compared with a time-varying threshold: the j-th node is elected cluster head when the generated value is below threshold T. At clustering round r, the threshold is

$$T(j) = \begin{cases} \frac{P}{1-P(r \, mod(1/P))} & \text{if } n \in G, \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

where P is the expected percentage of cluster heads (a priori fixed by the network's designer) and G is the set of nodes which did not become cluster head in the last $\frac{1}{P}$ iterations (when $r = 0$ each node has probability P to become cluster head). Once elected, cluster heads broadcast an advertisement message to contact simple nodes in the neighborhood (defined by the fixed communication power) and wait for their answers. Each node decides which cluster belonging to based on the signal power associated with reachable cluster heads (i.e., a node selects the closest - in terms of receiving power- cluster head). At this stage, cluster heads create, for each cluster, a TDMA schedule which is forwarded to the nodes composing the cluster (the cluster creation is performed without taking into consideration bandwidth limitations). At the end of the clustering phase and the network topology creation, nodes are ready to work: data are acquired from sensors and forwarded to the cluster head according to the defined TDMA schedule.

The main limit of this algorithm is that the number of nodes composing a cluster differs from cluster to cluster since clusters are not balanced: some cluster heads coordinate a very large number of nodes where others do not. The problem of low quality clusters may even worsen when bandwidth is an issue.

3.2.2 LC: Localized Clustering

In LC all nodes start the clustering phase by broadcasting an initial advertisement message according to a predefined communication power. The election phase follows and each node activates a *promotion timer* inversely proportional to its residual energy and the number of advertisements received from neighbours (we have a longer countdown for nodes receiving few advertisement messages or characterised by low energy).

If a node terminates the countdown without receiving messages during the elec-

tion phase, it becomes cluster head and, as a consequence, sends an advertisement broadcast message to its neighbours. Those nodes that receive such a message while still in an active countdown modality (i.e., they received few advertisements and/or have low energy) interrupt the countdown and label themselves as simple nodes. Simple nodes belong to a cluster based on the Received Signal Strength Indicator (RSSI) (and preference is given to closer cluster heads for reducing energy consumption). [27] has shown that LC guarantees a good network partitioning (or good quality clusters) and grants that all nodes within the network are reachable (isolated nodes become cluster heads). The main drawback of LC is the communication overhead associated with the advertisement messages of the clustering phase.

3.2.3 LLC: Low-energy Localized Clustering

LLC, which is an extension of LC, reduces the impact of messages overhead by imposing a maximum value x on the percentage of nodes that may become cluster heads w.r.t. the total amount of nodes (the parameter must be fixed at design time). The probability that node j is considered in the election mechanism at time t_r is proportional to the residual energy e_j [27]:

$$p_j = \frac{e_j tx}{e(t - t_r)} \tag{3.2}$$

where e is the initial energy of the node and t is the expected network lifetime. Probability p_i has the same role of the threshold value T in LEACH.

All other nodes wait for the end of the election mechanism, receive an advertisement message from elected cluster heads and select the closest one.

3.2.4 Main limits of LEACH, LC, LLC

The considered routing algorithms assume the unrealistic hypothesis of infinite (or very large) bandwidth: this is a main issue since it is hardly satisfied in many

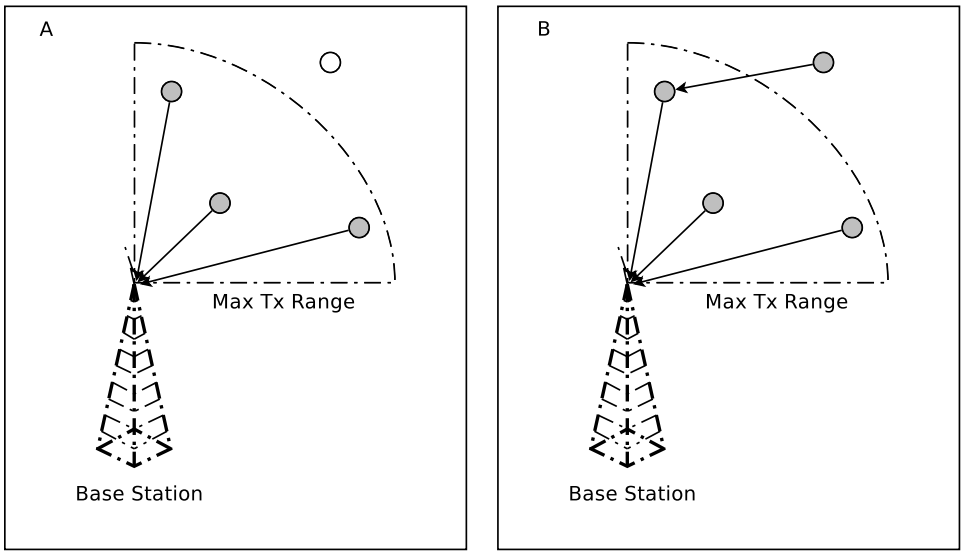


Figure 3.1: (a) Single-hop network: a unit is not-reachable; (b) Multi-hop network: each sensor is connected to the base station.

monitoring applications.

Moreover, we might encounter situations where some nodes are not connected to the bases station (reachability issue); this happens for those cluster heads whose distance from the base station is above the maximum reachable distance R_{MAX}. In turn, this implies that there it might exist few cluster heads not connected to the base station, hence producing not reachable clusters.

Figure 3.1(a) shows an example of an unreachable sub-network (white circle), where in Figure 1(b) we show how the problem can be solved with a multi-hop solution. Unreachability happens in those nodes whose distances from the closest cluster head are larger than R_{MAX} (the radius associated with the maximum power made available to nodes to connect to the cluster head). This phenomenon may arise in LEACH and LLC, as pointed out in [27] and can be solved by modifying the protocol as follows: nodes that after the clustering phase are not reachable have to directly connect with the base station. We should note that the probability that a node is not connected to the network is very low and that, at subsequent rounds, the node will be connected with high probability (buffered mechanisms need to be

envisaged for unreachable nodes to avoid data losses).

As a last note, selection of the algorithm parameters might be critical. In fact, some parameter configurations proposed for LEACH, LC and LLC, such as cluster radius or cluster heads percentage, are not realistic for most applications due to the unknown topology of the deployment and finite bandwidth issues. These aspects will be addressed in the following Section.

3.3 A k-Level Low Energy Localized Clustering

In this section we extend LLC to a k-level hierarchical algorithm (see fig. 3.2) to improve network energy management; selection of LLC derives from the fact that it generally outperforms LEACH and LC in terms of the network lifetime [27] and [30] as we also verified in the experimental section. In addition to the multi-level hierarchical extension, the algorithm we propose also considers node bandwidth limits to bound selection of the network parameters.

3.3.1 Set-up Model

We can safely assume that the antenna of a generic WSN node generates an isotropic spherical electromagnetic field [25]; in this way the network deployment area can be intended as covered by spherical neighbourhoods, each of which ruled by a cluster head. The base station is central to the environment to better exploit communication coverage and, to make the derivation more amenable, we assume that nodes are uniformly distributed within the deployment area with a superficial distribution density δ (as a consequence δ multiplied by a given surface provides the expected number of units deployed within it).

A finite number W of transmission power levels are usable by the nodes (usually $8/10$ levels in commercial units) and fixed by technological constraints. Denote by P_w and R_w the w-th power level and the associated transmission radius of the communication neighbour according to the first order communication

model, respectively; for power level W we have the maximum transmission power $P_W = P_{MAX}$ and $R_W = R_{MAX}$). Since $P_w < P_{w+1}, \forall w \in 1, .., t_p$ we have that $R_w < R_{w+1}$.

The expected number of nodes within the reachable environment is $N_0 = \delta\pi R_{MAX}^2$; for larger networks the above must be intended for each subnetwork communicating each other with a multi-hop communication.

3.3.2 X-LLC: the Proposed Routing Algorithm

We propose an autonomous and adaptive hierarchical routing algorithm, named *X-LLC*, that is a multi-level extension of LLC. The proposed multi-level extension is particularly appealing for reducing the energy consumpion of LLC since it allows a short range transmission activity. Moreover, the proposed algorithm grants adaptability to changing environments (since it automatically provides quick reaction to topological changes in the network by instructing new clustering configurations) and is autonomous in the routing decision (the routing algorithm is not centralized and nodes perform local decisions to select the optimal path route).

Traditional hierarchical routing algorithms consider only one level of cluster heads. Here, we propose to extend the LLC algorithm so has to have $k \geq 1$ intermediate levels of cluster heads by considering k distinct LLC election phases. Each election phase i elects cluster heads of the i-th level. Iteratively, the i-th level cluster heads participate to the election of the $i + 1$-th level cluster heads. The process iterates up to the k-th level. Extension to k levels implies that cluster heads at level $i-1$ send data to cluster heads at level i with a star network topology, cluster heads can possibly -but not necessarily- execute a compression algorithm to further reduce power consumption; level 0 are sensor nodes while cluster heads of the last k level send packet directly to the base station. Cluster heads, in addition to communication and data aggregation, also provide sensorial data acquisition. An example of a hierarchical structure is presented in Figure 3.2 for the case $k = 2$.

The suggested X-LLC algorithm is given in Algorithm 1 and the nomenclature

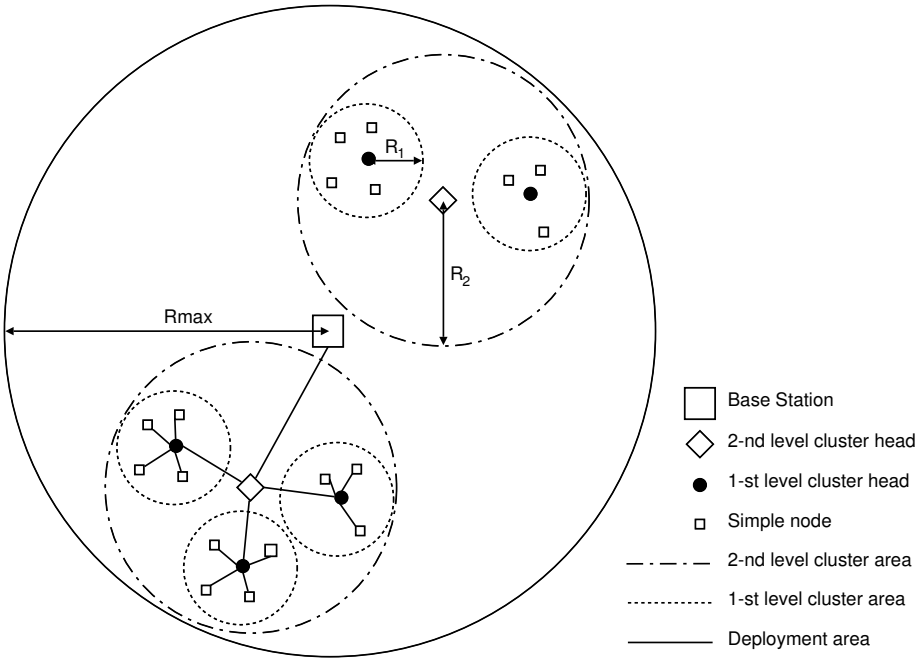


Figure 3.2: X-LLC, k=2.

Symbol	**Explication**
N_0	Total number of nodes
S_i	the set of nodes that may partecipate to elections at the i-th level
n_{jir}	the j-th node $\in S_i$ at election round r
e_{jir}	the residual energy of n_{ijr}
p_{jir}	probability that n_{ijr} partecipate to the election mechanism (computed with Equation 3.2)
R_w	the transmission radius at the w-th transmission power level
T_a, T_b	the node timers
$t_{election}$	the time required for the procedure completion

Table 3.1: Symbols used in Algorithm 1.

Algorithm 1 *electClusterHeads*: i, r, n_{jir}

1: $m = 0$;
2: $u = rand\ \{U(0,1)\}$;
3: **if** $p_{jir} \geq u$ **then**
4: *enableTimer*(T_a);
5: *send* (msg_{ADV}, R_w);
6: **while** *isNotExpired* (T_a) **do**
7: **if** *receiveMessage* (msg_{ADV}) **then**
8: $m = m + 1$
9: **end if**
10: **end while**
11: **else**
12: *sleep* ($t_{election}$)
13: **return**
14: **end if**
15: $T_b =$ *calculatePromotionTimer* (m, e_{ilr})
16: *enableTimer*(T_b)
17: **while** *isNotExpired*(T_b) **do**
18: **if** *receiveMessage* (msg_{ADV}) **then**
19: sleep ($t_{election}$)
20: **return**
21: **end if**
22: **end while**
23: **if** $l < k$ **then**
24: *electClusterHeads* ($i + 1, r, n_{jir}$)
25: **else**
26: *associateNodes*: ($i + 1, r, n_{jir}$)
27: **end if**

of symbols in Table 3.1. Being X-LLC a distributed algorithm, it works locally at a node level (the routing algorithm is executed by each node when participating to the election).

Each node generates a random value u between 0 and 1 (Step 2) that is compared (Step 3) with a threshold p_{jir} defined in (3.2) : if p_{jir} is greater than u, the node becomes a candidate cluster head and participates to the election phase; otherwise, it stays silent (Step 12) until the election process terminates. Each candidate node broadcasts an advertisement message with transmission power P_w and, consequently, covers a spatial neighbourhood of radius R_w. Then each candidate node collects the advertisement messages coming from the other candidate nodes in the neighborhood (Step 7) and sets a promotion timer T_b that is function of the amount of received messages and the node residual energy (Step 19). When T_b expires the candidate node becomes a cluster head at level i and broadcasts an advertisement message with transmission power P_w. If T_b has not yet expired and a candidate node receives an advertisement message (coming from a candidate node that becomes cluster head)(Step 18), it interrupts the promotion time and wait until the election process terminates (Step 19).

Nodes that become cluster heads at the (i)-th level participate to the election of the $(i+1)$-th level cluster heads. Cluster heads not elected at the (i)-th level, simply remain $(i-1)$-th level cluster heads.

The election phase is then iterated up to the k-th level.

To allow transmissions to higher distances, the value of the transmission power P_w have to increase with i. The relationship between the choice of P_w and i, which depends on technological constraints and is application-dependent, can be taken by suitably mapping the number of levels available for the transmission power W to the number of cluster heads levels k.

The association phase, which starts after the completion of the election process, is composed by k specific association sub-phases which are performed in a top-down fashion: starting from the base station to simples nodes. In the first asso-

ciation phase, the k-th level cluster heads associate themselves to the base station which will send back their TDMA schedule. Then, the $(k-1)$-th level cluster heads register themselves to the nearest k-th level cluster head (that answers by providing the TDMA schedule to the associated $(k-1)$-th level cluster heads). This process iterates up to the sensor nodes level.

X-LLC provides a remarkable advantage in terms of transmission energy consumption (once compared with traditional hierarchical algorithms) since it allows to reduce the cluster size in terms of radius and, hence, number of nodes. Moreover,

- each cluster head rules over a small -possibly balanced- number of nodes;
- cluster heads forward collected information to a cluster head at higher abstraction level instead of sending them directly to the base station;
- the transmission range of simple nodes can be reduced w.r.t. the LLC one. As a consequence, the transmission requires less power and the inter-cluster interference decreases.

Determination of the optimal number of levels for a given application depends on the characteristics of the deployment, the nature of nodes, the available bandwidth and energy as we will see in the next Section.

3.3.3 Bandwidth limitations in a generic k-level hierarchy

At the end of the association phase the units of the wireless sensor network are organized as an n-ary balanced tree (see Figure 3.3), where n is defined by the network designer. Obviously, n is a key parameter for all the hierarchical algorithms and our k-level hierarchical routing algorithm does not make exception. Traditionally, the value of n is selected with a trial-and-error approach. Here we investigate the effects of a limited bandwidth on the choice of parameter n hence reducing the feasible values it can assume.

Let B_M and b be the maximum bandwidth (in kbps) for each unit (sensor nodes,

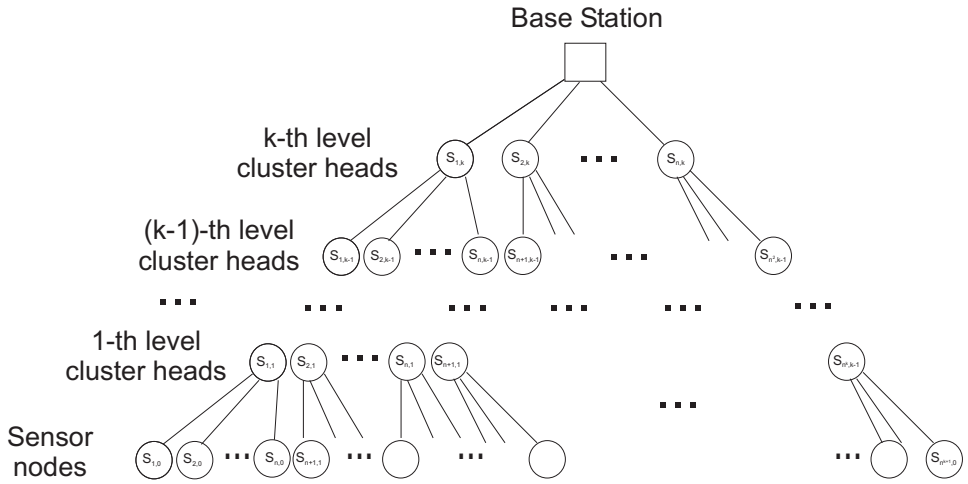


Figure 3.3: Hierarchical structure of the sensor network

cluster heads and base station) and the throughput (in kbps) derived from the sensing activity, respectively.

As shown in Figure 3.3, sensor nodes dispatch the acquired data to the 1-st level cluster heads. Assume that the throughput of each sensor node (which may be considered at level 0 in the hierarchy) is $T_0 = b$.

Then, each 1-st cluster head forwards the received data to the upper level cluster head together with its acquired data: the throughput of each 1-st level cluster head is then $T_1 = nT_0 + b = (n + 1)b$.

This process iterates up to k-level cluster heads that receive data from the $(k - 1)$-level cluster heads and send them to the base station (together with their own acquired data). Hence, the throughput of the k-th level cluster heads is $T_k = nT_{k-1} + b$.

By induction, the throughput of the i-th level cluster heads is $T_i = nT_{i-1} + b$ and, in a closed, not recurrent form we have that:

$$T_i = nb\frac{1 - n^i}{1 - n} + b. \tag{3.3}$$

Due to the hierarchical structure of the sensor network, T_i measures all the traffic passing through each cluster head level i. In other words, T_i represents the

Commercial sensor units	Available bandwidth (kbps)
Mica*z* [16]	250
Tmote-*sky* [31]	250
Mica2 [16]	38.4
Dot [16]	10

Table 3.2: Available bandwidth of commercial sensor units

sum of throughputs of all the sensor units (sensor nodes and cluster heads of lower levels) of a sub-tree rooted in a i-th level cluster head.

Being T_i the throughput of the sensor units at the i-th level, the network has to guarantee a suitable bandwidth to grant correct data transmission to the $(i+1)$-th level. This critical issue has never been addressed by traditional hierarchical routing algorithms (which assume infinite bandwidth). In real applications, the bandwidth available to sensor units is finite and depends on technological constraints (see Table 3.2).

By considering the bandwidth limits a constraint immediately arises: the incoming bandwidth of the sensor units of the i-th level must be larger than the throughput of the sensor units of the $(i-1)$-th level (the bandwidth available to sensor units of the $(i-1)$-th level must satisfy throughput needs). When this constraint is not satisfied the network cannot work properly since packets are lost before reaching the base station.

Assume, without any loss of generality, that the network adopts a TDMA as MAC layer (reasonable assumption as also pointed out in the literature, e.g., see [25]) and let b_0 (kbps) be the overhead (in terms of bandwidth usage) caused by the synchronization message of the TDMA (whose manager is implemented in all cluster heads and in the base station). We have that the incoming bandwidth of the cluster heads and base station is reduced to $B_M - b_0$.

In n-ary balanced trees the number of k-level cluster heads connected to the base station is n and the maximum available outgoing bandwidth for each k-level cluster heads guaranteed by the base station is

$$B_k = \frac{B_M - b_0}{n}. \tag{3.4}$$

Starting from B_k, we can recursively identify the outgoing bandwidth available to cluster heads of level i:

$$B_i = \frac{B_{i+1} - b_0}{n},\ where\ \ 0 \leq i \leq k\ \ and\ \ B_k = \frac{B_M - b_0}{n} \tag{3.5}$$

Which can also be expressed as

$$B_i = \frac{B_M}{n^{k-i+1}} - b_0 \frac{n^{k-i+1} - 1}{n^{k-i+1}(n-1)}. \tag{3.6}$$

As intuitively stated the outgoing bandwidth available to cluster heads increases with i. In other words, k-th level cluster heads have the largest $B = B_k$ bandwidth (from (3.7)), while sensor nodes assume the lowest value of B:

$$B_0 = \frac{B_M}{n^{k+1}} - b_0 \frac{n^{k+1} - 1}{n^{k+1}(n-1)}. \tag{3.7}$$

The amount of available bandwidth w.r.t. the total number of levels (base station, k levels of cluster heads and sensor nodes) for the most important commercial sensor units (see Table 3.2) is presented in Figure 3.5. As expected, the available bandwidth decreases quickly with the increase of the number of levels (in this case $n = 5$ and $b_0 = 1$). Table 3.3, which shows the numerical representation of results, suggests that Micaz and Tmote units are able to support a k-level hierarchical routing mechanism. By considering $b = 1$, which means that the throughput derived from the sensing activity is 1 kbps (typical value for environmental monitoring), Micaz and Tmote support up to $k = 2$ intermediate levels of cluster heads (4 levels). Conversely, Mica2 can supports only one level of cluster heads (3 levels) while Dot can only be used with a star connection to the base station.

Figure 3.5 presents the effect of b over the available bandwidth (in this case $B_M = 250$). The available bandwidth decreases as b_0 increases since the overhead

	Number of levels					
Available bandwidth (kbps)	**1**	**2**	**3**	**4**	**5**	**6**
Mica$z$, Tmote	250	49.8	9.76	1.75	0.15	0
Mica2	38.4	7.4	1.28	0.05	0	0
Dot	10	1.8	0.16	0	0	0

Table 3.3: Available bandwidth of commercial sensor units w.r.t the total number of levels (base station, k levels of cluster heads and sensor nodes). $n = 5$ and b_0=1 kbps.

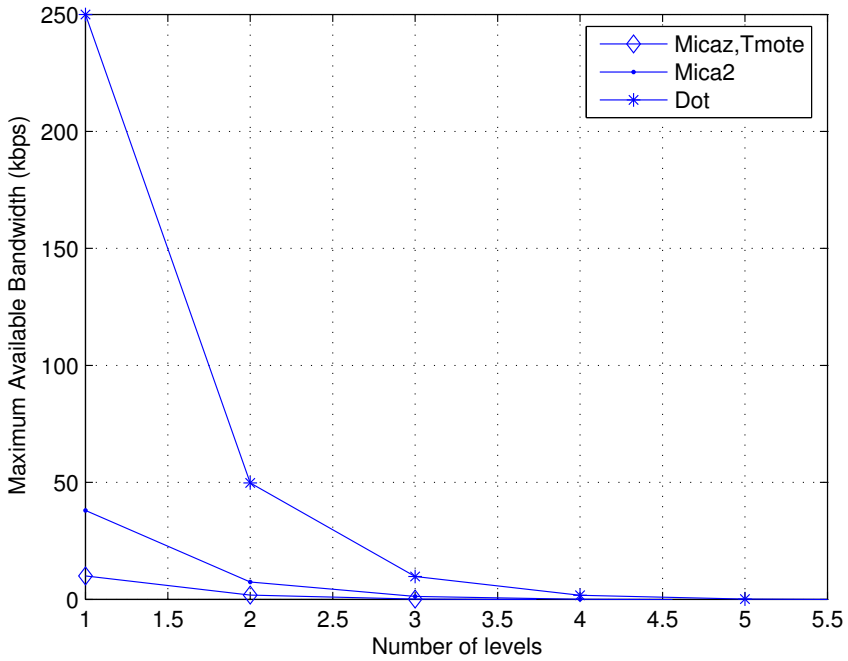


Figure 3.4: Maximum available bandwidth w.r.t. the number of levels for commercial sensor units. n=2 and b=1

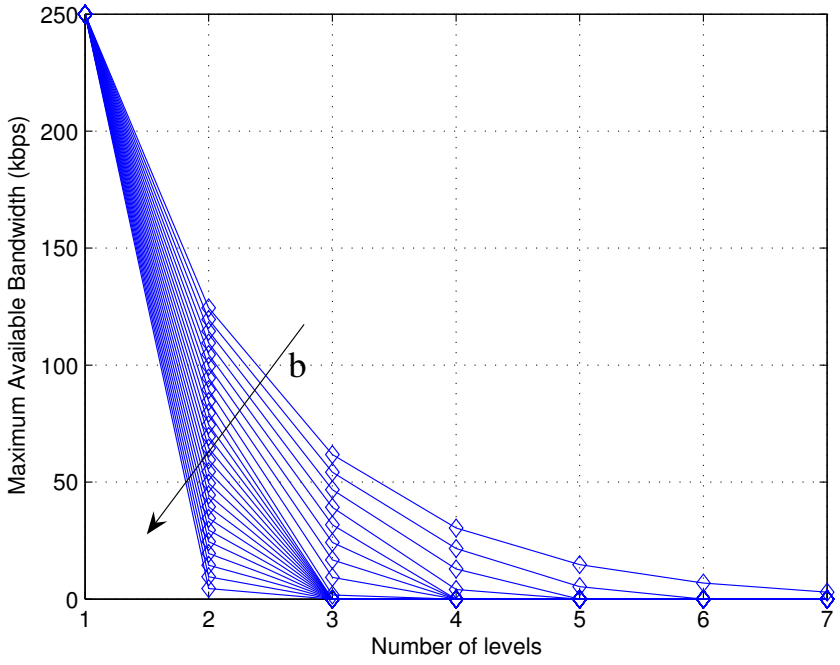


Figure 3.5: Maximum available bandwidth w.r.t. the number of levels for the Micaz (b ranges from 1 kbps to 250 kbps). Legend: n=2 and B_M=250

reduces the bandwidth for the transmission of acquired data.

The effects of n over the available bandwidth is presented in Figure 3.6 (in this case $b = 1$ and $B_M = 250$). As expected from (3.7), the increase of n decreases the available bandwidth. In other words, when n increases, the available bandwidth must be shared by a higher number of units (causing the reduction of the available bandwidth for each unit).

3.3.4 Relationships among design parameters

Let us consider an i-th level cluster composed by n i-th level cluster heads. In the previous Section we derived the relationship between throughput T_i of the i-th level cluster head and the hierarchy level i. We now derive the relationship among n, the transmission radius R_i at level i, and the corresponding transmission power P_i. By assuming a first order radio model, we have that

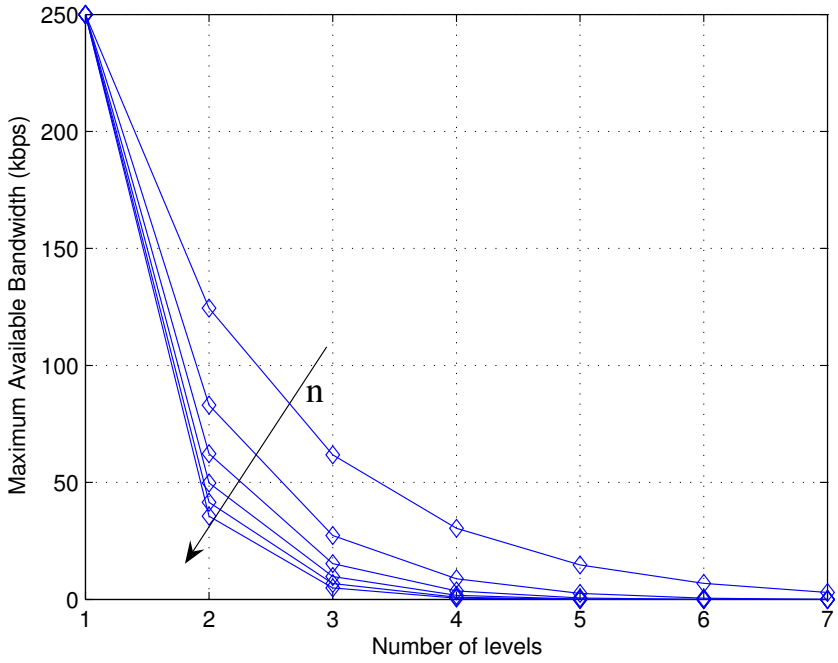


Figure 3.6: Maximum available bandwidth w.r.t. number of levels for the Micaz (n ranges from 2 to 7). Legend: b=1 and B_M=250

$$P_i \propto R_i^2. \tag{3.8}$$

The number n of nodes that compose an i-th level cluster is related to the radius R_i and the node density at, the i-th level:

$$n = \delta_i \pi R_i^2, \tag{3.9}$$

where δ_i is defined as:

$$\delta_i = \frac{n^{k-l}}{\pi R_{MAX}^2}, \forall i \in \{0..k\} \tag{3.10}$$

If we consider the ratio between δ_i, δ_{i+1}, we obtain that:

$$\frac{\delta_i}{\delta_{i+1}} = n \tag{3.11}$$

Equation (3.9) can be used for i and $i+1$:

$$\delta_i \pi R_i^2 = \delta_{i+1} \pi R_{i+1}^2$$

to obtain:

$$R_{i+1} = R_i \cdot \sqrt{\frac{\delta_i}{\delta_{i+1}}} = R_i \cdot \sqrt{n} \tag{3.12}$$

For the k-th hierarchial level $R_k = R_{MAX}$. By using (3.12) it is possible to compute the R_i sequence and consequently, the power needed for transmission.

3.4 Simulation results

To evaluate the different performance in terms of system lifetime, energy consumption and alive nodes distribution on different algorithms, we developed an ad-hoc simulator. We considered a first order radio model (with a r^2 energy loss due to

channel transmission) and the same parameters suggested in [30]. We fixed the packet size for data at $1024bits$; while advertisement and synchronization were $40bits$ each. A generic node was charged with an initial energy $e = 1J$. We uniformly deployed 100, 75 and 50 nodes within a circular environment of radius $R_{MAX} = 30m$ centered in the base station. The node bandwidth is $40kbps$, and a $b = 1kbps$. We also assumed that each node sends a single data packet at each round. Missing parameters (e.g., R_i) have been determined as suggested in Section 3.3.4 by exploiting inter-relationships among the algorithm parameters. For other algorithms such as LLC or LEACH we used the parameters suggested in [30] and [27], respectively. We selected a $k = 2$ level hierarchy of cluster heads for X-LLC.

To compare performance we consider two indexes of metrics:

- the network life activity, defined as the percentage of alive nodes in a given round (with respect to the initial ones);
- the network residual energy, defined as:

$$\frac{\sum_{i=0}^{N_{Tot}(r)} e_i}{N_{Tot}(r)} \tag{3.13}$$

  were N_{Tot} is the number of alive nodes at election round r, and e_i is the residual energy of the i-th node.

Results regarding the network life activity are given in Figures 3.7 ($N_0 = 100$), 3.8 ($N_0 = 75$) and 3.9 ($N_0 = 50$); the *x* axis represents the round (a round is composed by an election procedure and one or more steady-state phases, associated with data delivery (here, we have a single steady-state)). The *y* axis shows the number of alive nodes or the averaged node energy.

We observe that LEACH outperforms, at the beginning, all other algorithms in terms of number of alive nodes. This can be explained by noting that in LEACH the election mechanism simply reduces to one advertisement message to be sent

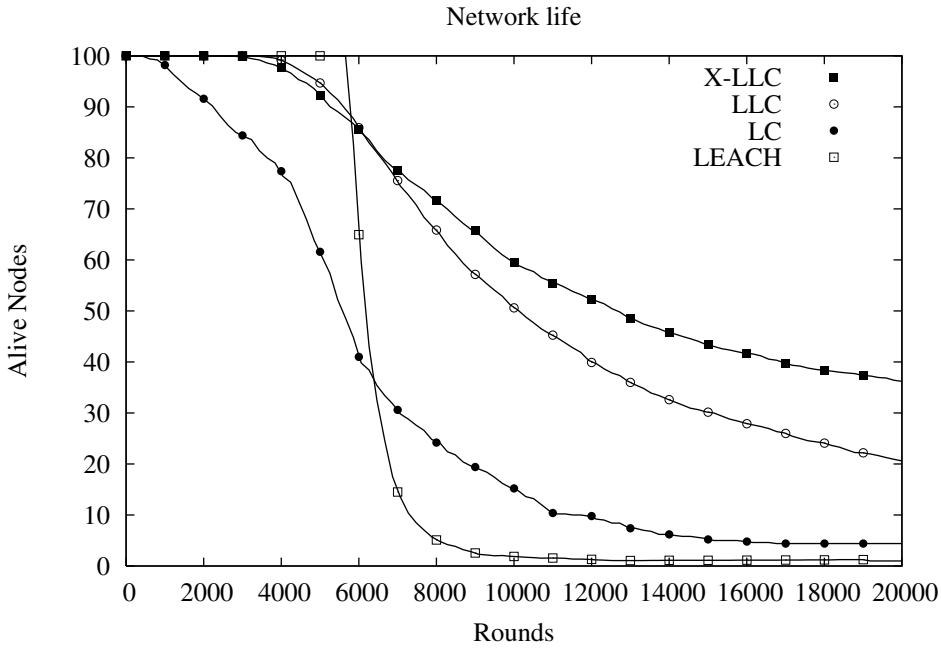


Figure 3.7: Network life activity, with $N_0 = 100$

for each cluster head, while in other algorithms a larger number of messages is required. Nothing is free: overhead introduced is necessary to assure a good cluster quality (a uniform number of nodes in clusters).

However, this initial non-optimality is abundantly compensated both by an improved quality of service (more nodes remain active over time and are characterized by a uniform distribution over the environment) and a longer lifetime. LC, LLC and X-LLC, better exploit available energy by balancing the overhead from the very beginning of the network life; this does not happen with LEACH as can be seen in Figures 3.10, 3.11, and 3.12.

X-LLC outperforms the envisaged algorithms, in particular, at round 20000 (long run), we have that it improves (Figures 3.7, 3.8 and 3.9) over LLC at least for a 35% in terms of the number of alive nodes and a 12% in saved energy (Figures 3.10, 3.11 and 3.12). X-LLC consumes more at the beginning, when many nodes are available but, after about 6000 rounds, outperforms the others. This phenomenon can be explained by considering that, initially, all nodes have the same

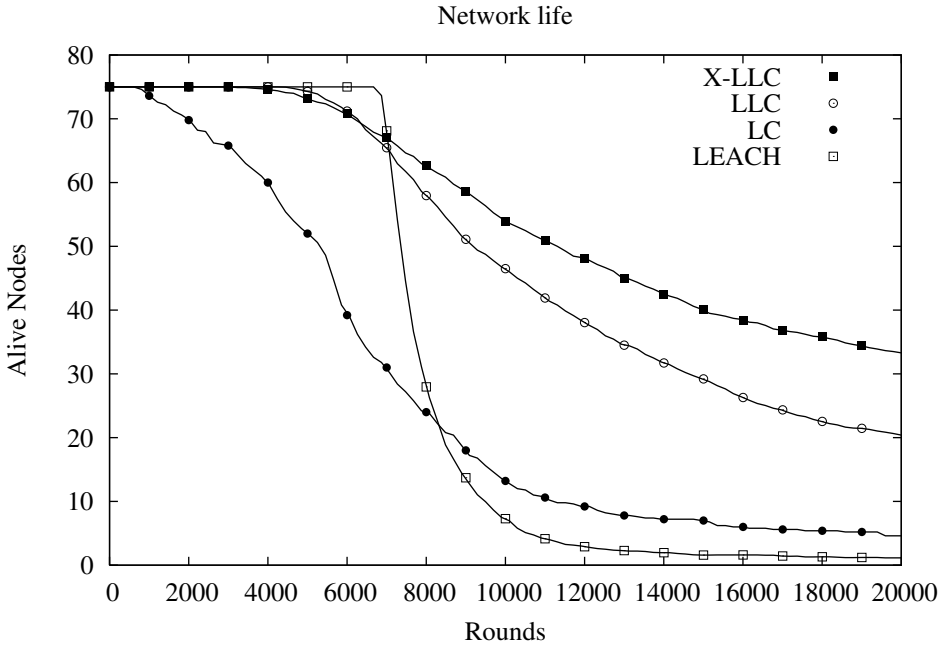


Figure 3.8: Network life activity, with $N_0 = 75$

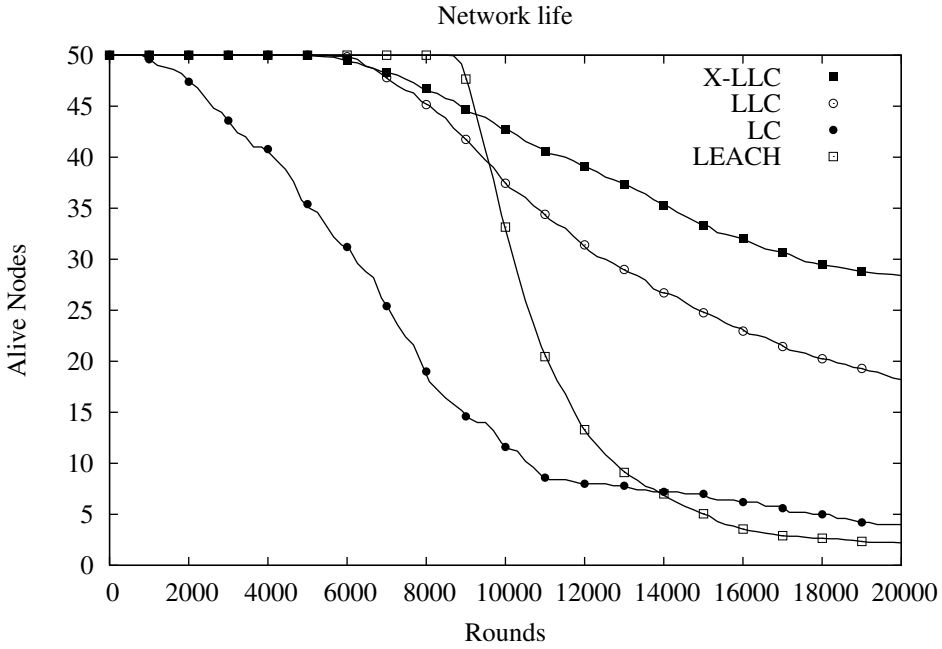


Figure 3.9: Network life activity, with $N_0 = 50$

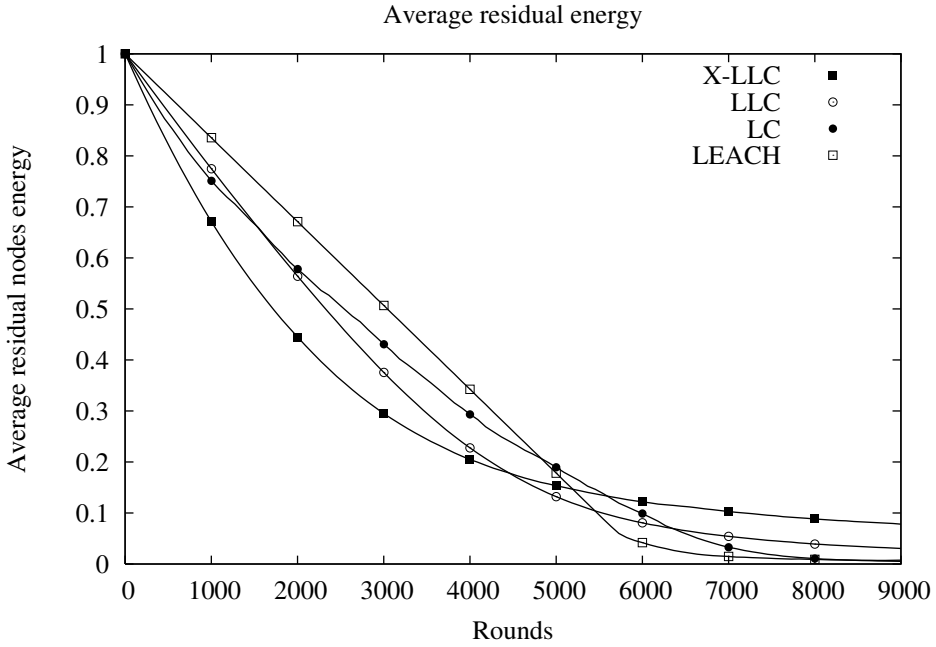


Figure 3.10: Network residual energy, with $N_0 = 100$

energy level e_i (and then, almost the same probability p_i to be selected as clusterheads, according to (3.2)) and, as a consequence, a high number of them go under the election process. This produces an extra overhead due to the quantity of messages sent and received by nodes. In a long term perspective, the reduced transmission range assured by a X-LLC hierarchical structure, is more effective in terms of energy saved than other algorithms.

We finally present the distribution of alive nodes in the deployment area at different rounds: $r = 1$, $r = 3000$, $r = 6000$, and $r = 9000$ (we selected the case where the number of nodes is $N_0 = 100$). Nodes are uniformly distributed in the deployment area (Figures 3.13, 3.14, 3.15 and 3.16, $r = 1$).

A uniform distribution for the alive nodes over rounds (or, again, the absence of areas not covered by nodes) is an amenable property we would expect from an effective monitoring network. In fact, only in this way we can grant a quality of service over time, i.e., a proper monitoring of the environment.

In Figures 3.13 (X-LLC), 3.14(LLC), 3.15(LC) and 3.16(LEACH) we present

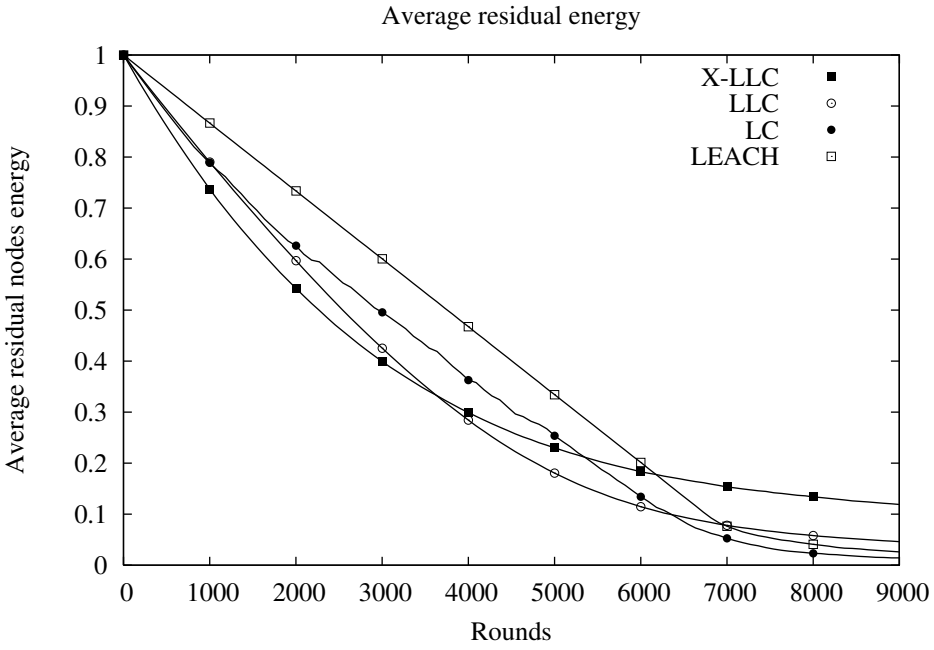


Figure 3.11: Network residual energy, with $N_0 = 75$

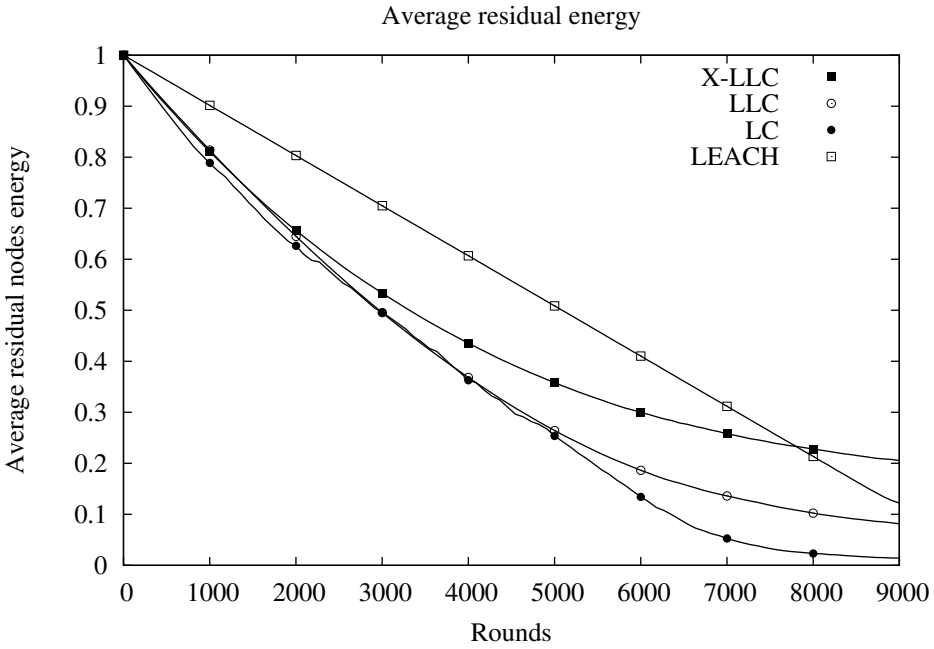


Figure 3.12: Network residual energy, with $N_0 = 50$

the distribution of alive nodes in cases $r = 1$, $r = 3000$, $r = 6000$, and $r = 9000$, respectively. Algorithms experience a degradation in performance as function of r (the number of alive nodes decreases). In particular, we see that LLC, LC and LEACH introduced a non monitored area in proximity of the base station. This phenomenon arises for LC at $r = 3000$ (Figure 3.15); at $r = 6000$ for LLC (Figure 3.14) and for LEACH. We believe that the non uniform distribution in the long run is associated with the excessive usage of nodes near to the base station which became cluster heads with high probability (due to (3.2)). Hence, these nodes will consume the available energy before the others causing a non-uniform distribution of the alive nodes.

Differently, X-LLC grants a uniform distribution of alive nodes over rounds (see Figure 3.13). This amenable behavior guarantees the proper functioning of the WSN: alive nodes acquire data from the area even if with a coarser spatial resolution. The reason of this behavior resides in the ability to reduce side effects caused by the election mechanism (the possibility that a node is elected depends on the amount of nodes in its neighborhood): X-LLC, by creating smaller clusters, supports the election of cluster heads everywhere in the environment, leaving to a uniform distribution of alive nodes. Differently, in LLC, LC, LEACH the averaged cluster size is larger than the X-LLC one and eligible nodes close to the boundary heavily suffer from bound effects (i.e., are elected as cluster nodes with lower probability).

3.5 Final remarks

This chapter presents an extension of the LLC routing algorithm, a particularly interesting algorithm in distributed wireless measurement systems, by introducing a hierarchical structure in the network management (network nodes are clustered with a hierarchical approach) which, by exploiting the nature of the topology, allows us for improving adaptability, network lifetime and overhead load balance

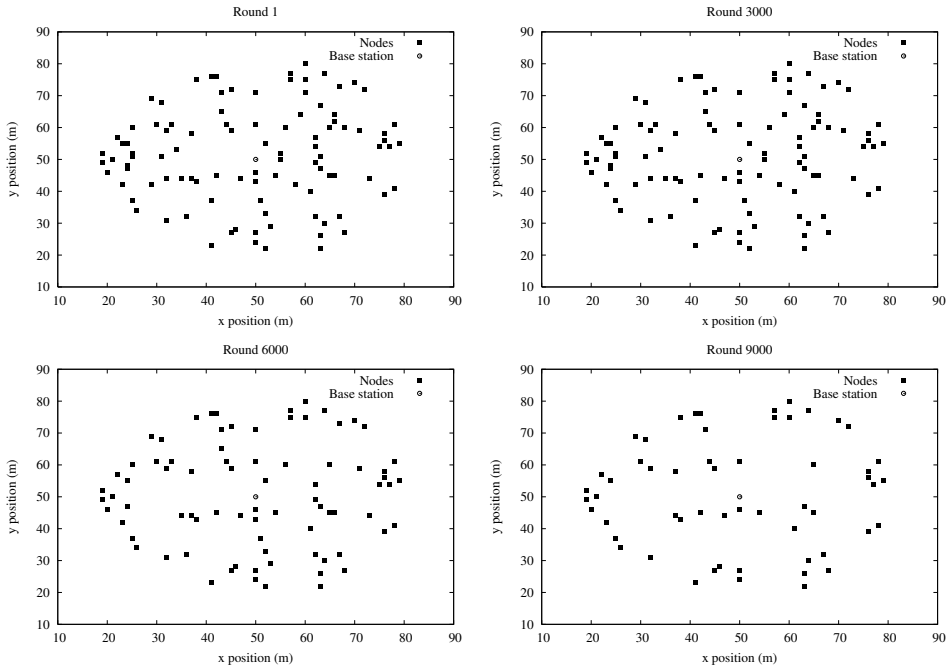


Figure 3.13: XLLC

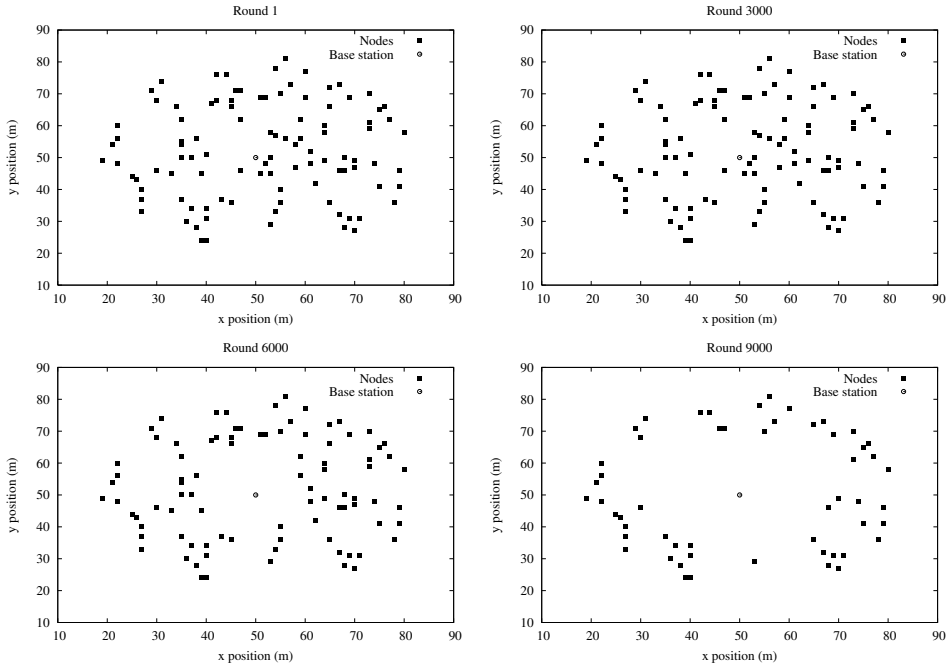


Figure 3.14: LLC

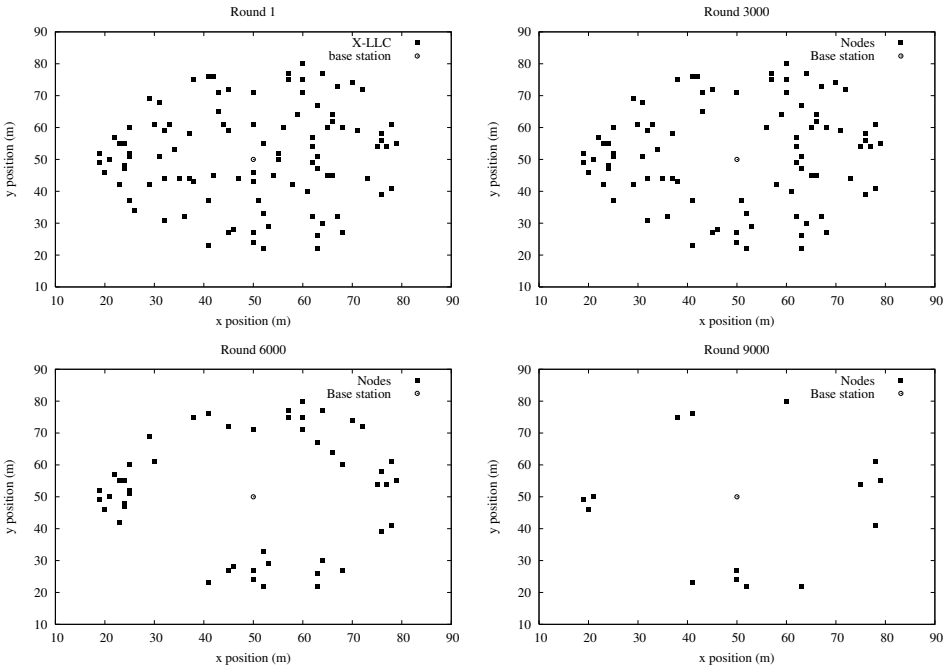


Figure 3.15: LC

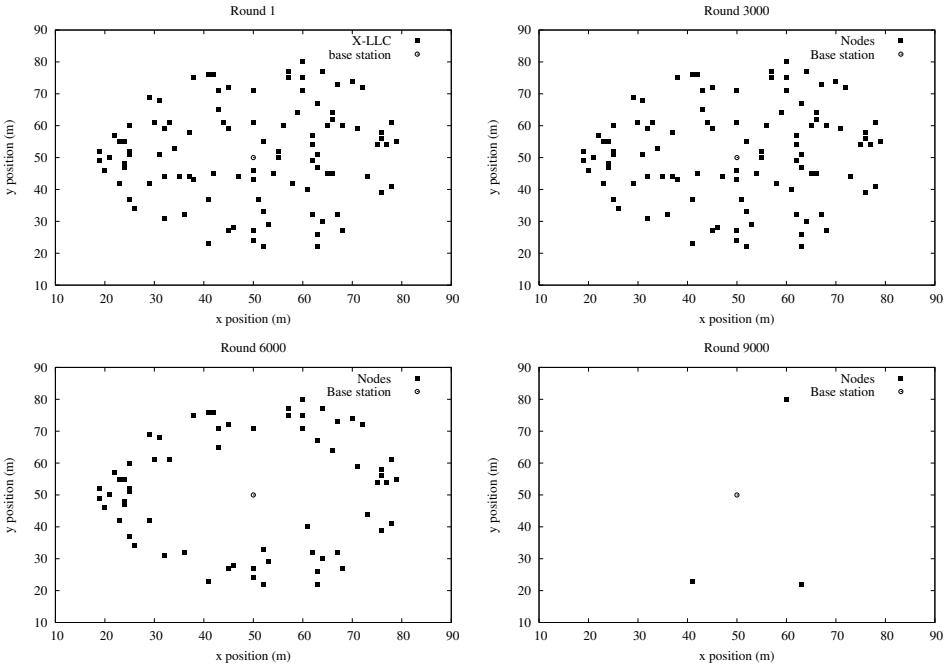


Figure 3.16: LEACH

among clusters. The novelty of the proposed approach resides in a k level hierarchical structure of cluster heads (the literature only considers one level of cluster heads) and considering the realistic case of finite bandwidth for nodes. This latter point is crucial in WSN nodes since available routing algorithms may become unfeasible in a large class of applications for technological constraints associated with a reduced bandwidth.

Moreover, the proposed algorithm is particularly appealing for its ability to maintain a uniform distribution of alive nodes in the deployment area, feature associated with the monitoring QoS, where other routing algorithms introduce vast areas not covered by sensor nodes hence impairing the effectiveness of the distributed measurement system.

Chapter 4

Node level mechanism: start-restart framework

In the last two chapters, a MAC layer and a routing layer to deal with the insertion/removal of nodes (due to permanent or transient node and communication faults, energy availability for the nodes) have been presented. In this chapter, the thesis focuses the development of node-level mechanisms allowing nodes to suspend/resume its activity.

This goal has been achieved with following two steps:

- development of a framework for the start and restart of the nodes that use both detailed information about residual energy;
- a major change in core part of the nodes OS to outperform the standard power management policy, hence, to increase the nod life.

The first step has been addressed with the integration, at software level, of specialized hardware designed for real applications (see chapter 6). In fact, this specialized hardware (i.e., MPPT [13]) provides both adaptive harvesting mechanisms and the hardware interface for retrieving information about the current energy state of the node (e.g., incoming solar power, battery voltage, absorbed current). However, by considering traditional batteries, MPPT is unable to provide an accurate

value of the unit residual energy. In the practice, traditional batteries have a non-linear discharge; besides, the behavior changes over time due to both aging effect and operational conditions (e.g., temperature).

Conversely, supercapacitors overcome these limitations [32]. Supercapacitors have the same electrostatic mechanism of a traditional capacitors but are based on a different technology that allows capacity up to $1000F$. Hence, the energy E stored inside a supercapacitor is computable as $\frac{1}{2}CV^2$, were C is the capacity of the supercapacitor and V is the voltage. It implies that a joint use of MPPT and supercapacitors technology allows to know the residual energy of the node.

The second step is achieved by using the information about the residual energy of the node at the OS level to exploit an effective power management. In this thesis the considered operating system is TinyOS that is an operating system developed by the academic community, which has been specifically designed for WSN nodes hardware (in particular for *mica* and *telos* families). All the OS software mechanisms for the proposed framework have been developed for such an operating system.

The suggested steps, together with ad-hoc hardware, allow the development of a novel generation of power manager systems for WSN units. The knowledge about the residual energy of nodes is used to trigger the power manager. Instead of a complete node shutdown, the proposed power manager can turn-off system modules (e.g., radio). However, if the energy level is critically low, the power manager hibernates the node by saving its internal state (e.g., tasks, memory). The specialized hardware wakes up the node when the energetic level in the supercapacitors allows a normal operational life of the node. Then, the node restores its previous internal state. To achieve this goal essential parts of the node OS (i.e., scheduler, power manager) have been modified and the support for the MPPT hardware has been integrated.

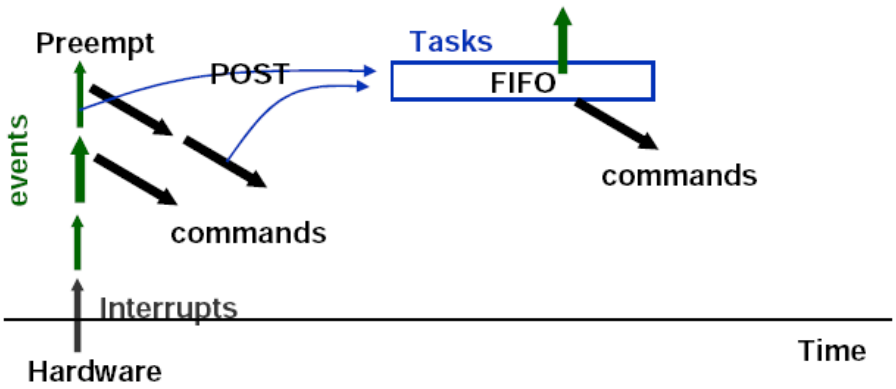


Figure 4.1: Default scheduler in TinyOS

4.1 TinyOS overview

The main design of TinyOS goals are the lightweight of the kernel and the small memory footprint. To address these goals the OS has a design very simple and the application is compiled together with the kernel modules and the hardware driver needed by application itself. The drawback is that only an application can be executed by tinyOS; moreover, to change application it is necessary reprogram offline the node.

The TinyOS has a component-based design and an application is, in fact, a graph of system and application modules. Each module expose an interface and may require other functionalities. Hence, an application is build connetting modules together. TinyOS has an event-based execution model to provide a efficient interface for both the sensors data gathering and the radio. Modules "post" computational tasks in a FIFO queue (the *scheduler* modules) and executed subsequently: this mechanism, named "differed procedure call" (DPC). Each posted task have the same priority (see figure 4.1), hence an intra-task preemption is not possible with the default scheduler. Conversely, tasks can be only preempted by events generated by modules or hardware interrupts

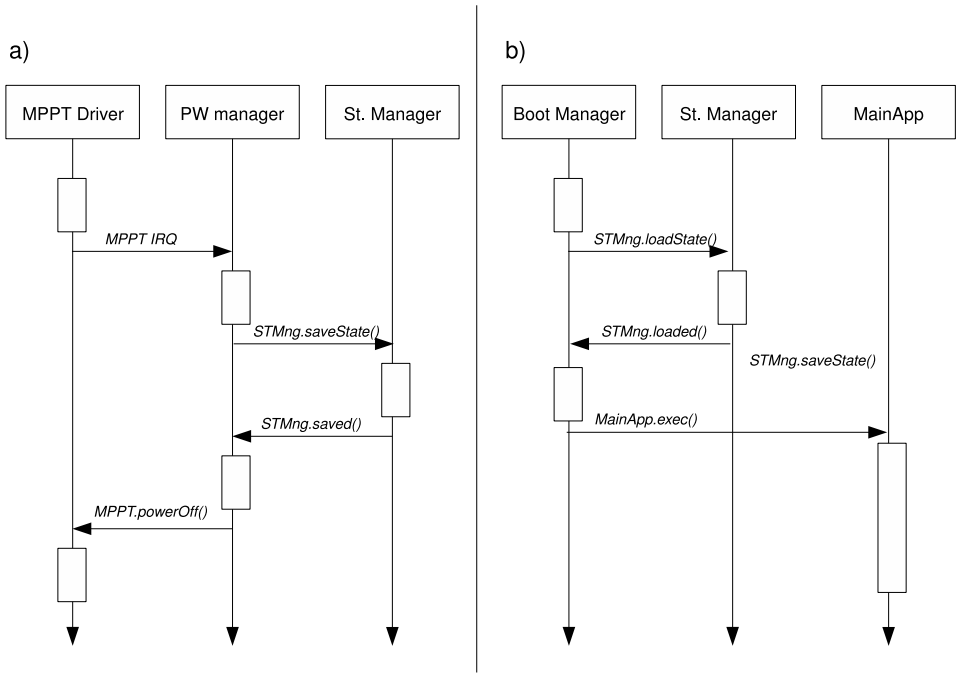


Figure 4.2: Restart mechanism and main system components

With the envisaged OS architecture the proposed mechanism for the controlled shutdown and restore is not feasible. This issue can be addressed only with a radical change in the core modules.

4.2 Shutdown/Restore mechanism: overview

In Figure 4.2 the mechanisms to save (fig. 4.2a) an restore (fig. 4.2b) are shown, respectively. In the figures only the main modules and the high level functions are reported.

Figure 4.2a shows the typical case of controlled node shutdown. An interrupt (*MPPT IRQ*) coming from MPPT (in particular, from its driver) is handled by the Power Manager (PM).

Then, the PM preempts the normal system tasks with a high priority procedure (*STMng.saveState()*), which is managed by the Store Manager (SM).

With this procedure, the state of the node is dumped into the flash memory. In particular, what is dumped is the value of variables and the tasks previously preempted by the procedure itself.

The SM ends this procedure with a signal (*STMng.saved()*) handled by PM; hence, it calls the shutdown procedure (*MPPT.powerOff()*) provided by MPPT driver.

Finally, the MPPT hardware turns off the node.

Conversely, when the MPPT turns on the node, its Boot Loader initially calls the state restore procedure (*STMng.loadState()*) handled by the SM that ends the procedure with a signal (*STMng.loaded()*).

The boot loader starts the main application resuming the first task previously preempted (*STMng.loaded(MainApp.exec())*).

The mechanisms here presented are based on custom modules that have been added to TynOS.

4.3 Proposed custom scheduler

To support the controlled shutdown mechanism is also necessary to modify the basic scheduler. The main change consists in the introduction of new levels of priority in tasks that allow, for example, to execute immediately the power manager tasks. The ability of cancel the execution of tasks is another feature, which has been added to the OS, to allows the complete node shutdown. Hence, the developed scheduler supports additional two types of task:

- *high priority tasks* are inserted in the head of the queue to be processed immediately;
- *application tasks* have the same priority of basic ones, but can be removed from the queue.

In figure 4.3 is shown an example of execution: in the queue there are four basic tasks *B1*, *B2*, *B3*, *B4*. Then an application task *A* is posted. This task get the

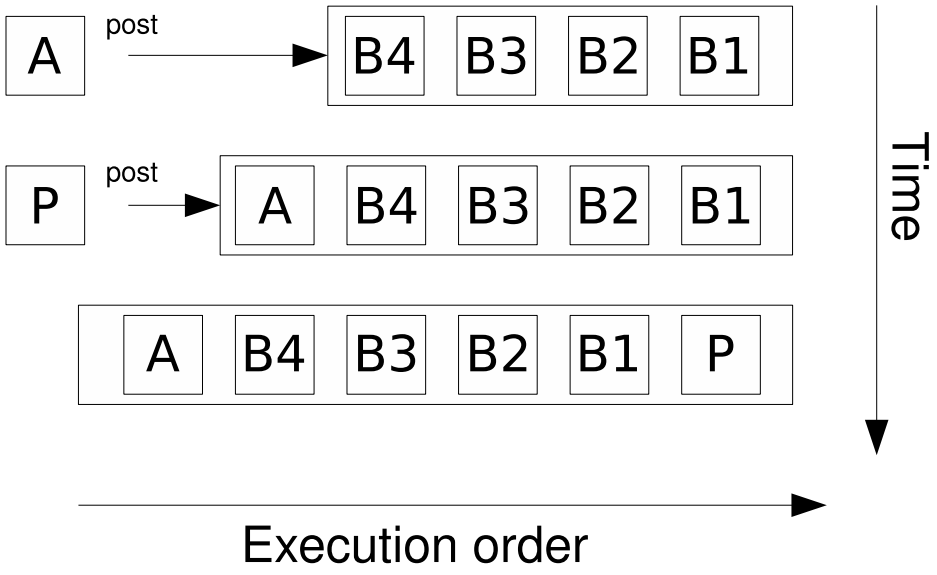


Figure 4.3: Custom scheduler examples: posting tasks with different priorities

last position in the queue. Conversely, is a high priority task *P* would have been posted, it would be placed in the first position.

In figure 4.4 another example is presented: in the queue the basic tasks *B1*, *B2*, *B3*, *B4* and the application tasks *A1* and *A1* are present. The *flush()* operation removes only *A1*, and *A2* from the queue.

4.4 State save-restore framework

In TinyOS a low level driver to save in the node flash memory exists and provides low level mechanisms to read and write. By using only these low-level functionalities, it is possible *at application level* to provide a save-restart mechanism.

However, the aim of the proposed framework consists of the integration of this mechanism *at system level*. Thus, for the application (and, consequently, for the programmer) it must be transparent: the programmer has to write the application layer only. The complexity of the access to the flash (read and write operations, I/O errors handling) is managed automatically by the system. The framework is

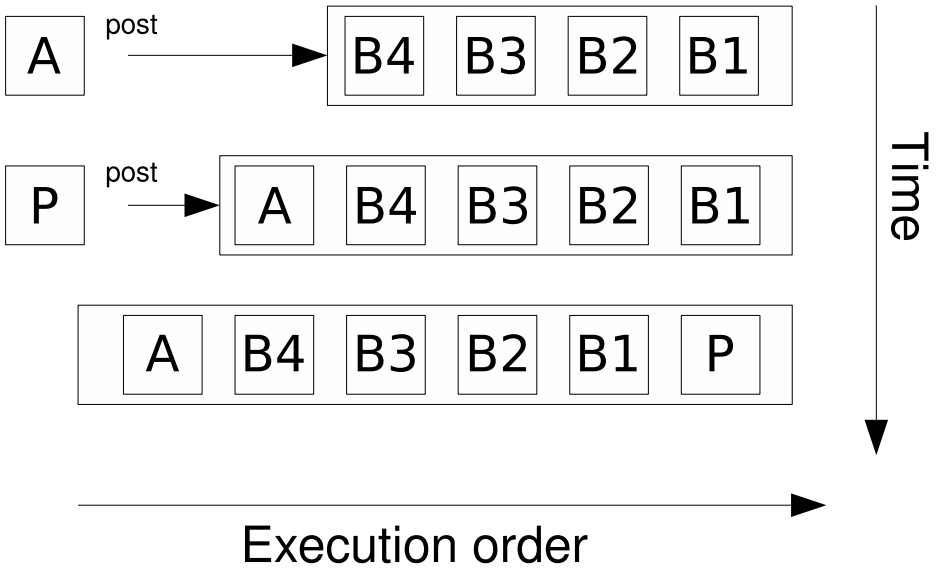


Figure 4.4: Custom scheduler examples: flush application tasks

thus independent from the application and can be easily reused in other contexts.

The save-restore framework is based on a custom components: the Store Manager. It manages transparently the node flash memory. In particular, the information required to restore the node state are the state of the variables and the list of the tasks flushed in the last shutdown. These two information are enough to restore the node state.

Chapter 5

Application Level mechanism: Informative system integration

In this Chapter, a language for managing data in highly pervasive systems made of very different devices as to their technology and functional capabilities is presented . Functional and non-functional requirements are dealt with in a transparent mode by a SQL like interface. The most relevant features of the language, the related data structures and some query examples are here introduced.

5.1 Introduction

In a very short time the complexity of the envisaged WSN-based systems grew from a handful of homogeneous sensors to hundreds or thousands of devices differing as to their capabilities, architectures, and languages.

The result of this inflationary expansion is the difficulty an application programmer encounters in dealing with the languages and protocols, which characterize different portions of the system, and in optimizing sampling, storage, and transmission strategies in order to save as much as possible of power in the batteries, which are the most wearable components in a system of mostly unattended small devices.

As a motivating example of such systems, we refer to one of the case studies in the ART DECO project [33], a large project funded by the Italian University Ministry: the automation of a large wine production farm *from the vineyard to the table*. Table 5.1 schematically shows information items and devices used for supporting the different phases of the production and delivery processes.

Table 5.1: **Devices usage and types**

WHERE	WHAT	HOW
vineyard	humidity, temperature, chemicals	sensors
cellar	humidity, temperature	sensors
bottle	tracking information	RFID tag
pallet	tracking information, temperature	RFID tag, sensors
truck	position information	GPS
workers	information system	PDA

A number of efforts have been made to define and implement a High Level Language for managing data in WSN applications, some of the most noticeable among them are TinyDB, GSN, and DSN.

TinyDB [34] [35] is one of the first efforts in querying WSN data with a SQL like interface; multiple persistent queries with different sampling time are issued from a pc, data are collected from Motes sensors in the environment, filtered, possibly aggregated, and routed out to a base station. It is defined over TinyOS and it exploits power-efficient in-network processing algorithms. Its portability is bound to that of TinyOS.

GSN [36] [37] is a scalable, lightweight system which can be easily adapted, even at run-time, to new types of sensors, thus allowing a dynamic reconfiguration of the system. XML is used as the network and data specification language, while SQL is used as the data manipulation language.

DSN [38] [39] uses a completely different approach. The whole system is built in the declarative language Snlog - a dialect of Datalog - both for data acquisition and for network and transmission management.

The large heterogeneity of the pervasive systems we are dealing with, comprising devices ranging from passive RFID tags to application servers, requires a very high level of transparency so that as much as possible of the technical problems be handled by the system, while the application programmer only writes high level code. This observation led us to the approach *"single system - single language"*, but, unlike in DSN, we stay as near as possible to SQL, since it is still the most widely known and used data language.

At the outset, we only thought to implement a multi-platform version of TinyDB, which could be deployed on different sensor families with little effort; however, during the analysis phase, we extended our goal in three successive steps:

- *run time* support of heterogeneity;
- support of *non intelligent* device classes, such as RFID tags;
- extend the target to *generic pervasive systems*.

While the impact of the first step is limited to the middleware, which must provide a set of APIs to manage the different platforms in a uniform way, the other two steps require a rethinking of the functional features of the language. Tags are not equipped with sensors and cannot perform data manipulation or transmission, but their interaction with the external world is mediated by the RFID reader while the exchanged information is the tag presence itself, and not some sensed value. Therefore, we must provide an abstraction for the RFID behavior and this can be done in two ways: the first considers an *RFID tag as it were a sensor* whose "sampled data" is the *identifier of the last reader* which sensed the tag. The second one considers the *RFID reader as it were a sensor* whose "sampled data" is the *identifier of the last tag* sensed by the reader. Since communications with the network are always handled by the reader, the first solution provides a higher level of abstraction and requires a more sophisticated middleware. Moreover, the presence of RFIDs claims for the introduction of an *event based semantics* at the language

level, since we cannot rely on a fixed sampling interval, but actually "sampling" is performed by the reader at the moment of the tag passage.

The third step has been made possible thanks to the availability of a logical and distribution independent software platform for large heterogeneous networks, described in Section 5.2, provided by another workgroup of the ART DECO project [40]. This allowed our work to focus on the high level declarative language definition, being freed from low level programming issues, and processing the queries in terms of logical objects abstraction.

The language has been defined in order to manage both *functional features*, comprising the definition of operations which manipulate raw data to generate the query output and statements for the setting of sampling parameters, and *non-functional features*, which account for constraints on the offered functionalities and on the Quality of Service (QoS); in WSNs the QoS is mainly related to power management, however node latency and sensors availability are considered as well. Non-functional features are dealt with through a "policies" mechanism, which will be explained in Section 5.3.

The heterogeneity of the considered devices classes led us to identify two levels of abstraction represented by two levels in the language:

- a *Low Level Language* whose goal is to manage the sampling operations performing only data manipulations on a single sensor;
- a *High Level Language* whose role is that of manipulating sampled data in order to produce queries results.

Both languages have a SQL-like syntax; however, the semantics of the Low Level Language differs, since it can be thought as a mechanism to generate the data streams as in figure 5.1, and to determine when sampling should be performed on which nodes.

The rest of the chapter is organized as follows: Section 5.2 presents the architecture of the system and its middleware components; Section 5.3 introduces the

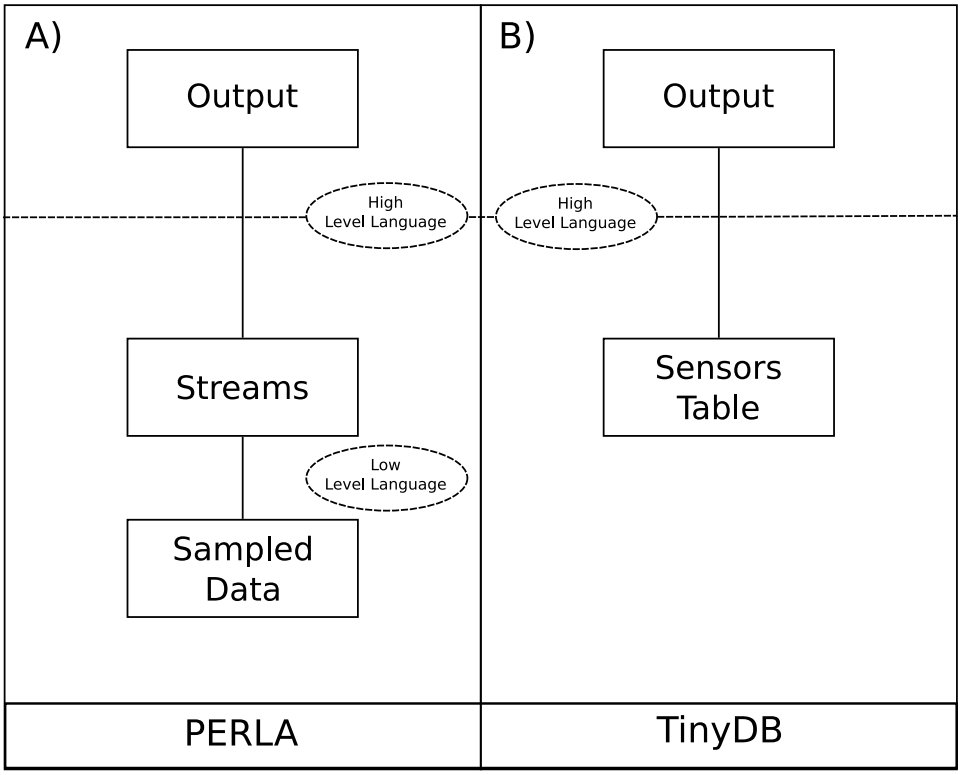


Figure 5.1: Comparison between PERLA and TinyDB.

PERLA language functional and non-functional features; finally, in Section 5.4 the query processing mechanism is explained together with examples - drawn from the above mentioned case study - which show the specificities of our approach.

5.2 System architecture and middleware

In this section the architecture for pervasive systems, over which we based our language, is briefly explained. The abstraction levels provided by the architecture and the interface supporting the two levels of the language are presented.

Figure 5.2 shows the system architecture:

- *Application Layer*: is the front-end used by applications in order to access data coming from the physical devices;

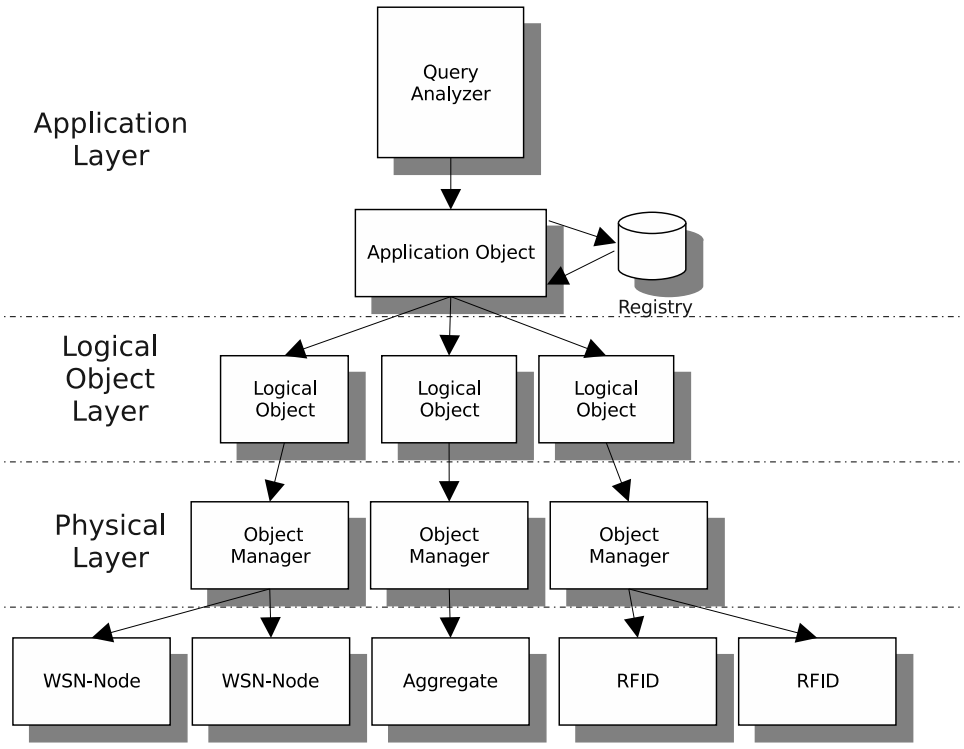


Figure 5.2: Middleware architecture.

- *Logical Objects Layer*: provides an abstraction for physical devices;
- *Device Access Layer*: provides the underlying infrastructure for accessing devices, abstracting from the required software distribution.

In the application layer, the *query analyzer* handles user-submitted queries and, using a dedicated component (i.e., *the registry*), it retrieves information about logical objects composing the system. Thus, the query analyzer selects logical objects relying on this information to execute the query.

In the logical layer, each component (i.e., logical object) wraps single or homogeneous groups of devices. In the first case, a logical object hides the complexity for accessing devices (in this way, a change in lower layers is transparent to the higher ones). In the second case, besides hiding the physical devices composing the aggregate, the operation manager directly provides aggregated results to the logical object.

Logical objects expose a standard interface that unifies and simplifies the access to higher layers. In particular, they provide three categories of attributes:

- *Static attributes* represent constant values describing a characteristic of the node (i.e., node type, maximum sampling rate, etc.);
- *Dynamic probing attributes* are the variables that a logical object must read from physical devices (i.e., a sensor measurement);
- *Dynamic non probing attributes* are those for which a logical object can return a local cached value without actually dealing with the physical device (i.e., current base station).

Note that different logical objects can expose different sets of attributes and that the same attribute can be dynamic for some devices and static for other ones (e.g. location). Logical objects interface can also provide events used to signal changes in the physical devices. Non probing attributes often have an associated event that notifies a change of their values (e.g. last sensed RFID reader changed).

The implementation of logical objects will be deployed as much as possible on the lowest layers of the architecture: if a device is too tiny and has very poor computational resources, the logical object wrapping it will be deployed at a higher layer.

The device access layer provides the access point to devices allowing to abstract the software infrastructure required by a specific device technology. For instance, in RFID-based systems, the operation manager is the middleware used to drive the reader.

5.3 The language features

We classified the features that should be exploited by the language in four categories:

- *Data representation*. The language should be able to hide physical devices as much as possible and to provide a database view of the whole pervasive system: statements written by users are queries on this database. Differently from traditional databases, every query must also specify how, when and where the sampling operations should be executed.

- *Physical devices management*. The main issues in providing physical devices abstraction are the definition of the sampling semantics for each class of devices and the introduction of a temporal semantics taking into account the existence of losses and delays in physical communications. With reference to the architecture presented in Section 5.2, each device is abstracted by a logical object and each sampling operation on the device is abstracted by the reading of a logical object attribute. Two types of sampling are supported: periodic or event based; the latter is activated when a logical object event is fired (typically used in RFID devices).

- *Functional characteristics*. Language statements should allow the user to

specify sampling parameters (time, mode, etc.) and the set of operations to manipulate raw data in order to generate a query output.

- *Non functional characteristics*. In WSNs the most important non functional parameters are related to power management. However, due to the node heterogeneity we are considering, a generic approach must be introduced: a small number of clauses should be provided by the language to support all the non functional characteristics that can be considered now or discovered in the future. From the user point of view there are no differences between logical objects attributes that retrieve data of interest and logical object attributes that return non functional parameters. However, an important difference exists from the system point of view: non functional fields exposed by logical objects are expressed in an abstract way. Then, they are internally translated into concrete values that can be handled by physical devices. For instance, a percentage power level attribute can be obtained from the voltage value for a certain class of devices, predicted from the number of executed operations for other devices, or just set to 100% for AC powered devices. Non functional attributes can be used to decide if a node should participate to a query, to set the current sampling rate, to retrieve information about network nodes, etc.

We now briefly outline the functionalities that should be supported by logical object interfaces of physical devices involved in query execution, so that the language semantics can be then explained in terms of the interaction between the query analyzer and the logical objects. The interface must provide three kinds of functionalities:

- *Retrieving attributes values:* attributes can be both data of interest and policy values. A special ID attribute must be supported by all the logical objects to allow the query analyzer to univocally identify them.

- *Firing notification events*: events can be used to perform event based sampling or to activate query selections.

- *Getting the list of supported attributes*: a set of methods should be provided to allow the query analyzer to know the set of attributes (and their data types) a logical object exposes.

In the following the main language issues are outlined.

5.3.1 Data structures

The data structures which support our language are the *stream tables* (or *streams*) and the *snapshot tables* (or *snapshots*). Streams are unbounded lists of records produced by queries. Each record has a set of user defined fields and a native timestamp field. It supports the *insertion* and the *reading* operations. In *insertion*, new records can be inserted into the stream by a running sub-query. The execution of this operation generates an *insertion event* that can be detected and used by other sub-queries. In *reading*, a data window can be extracted from the stream by a running query. This window is defined by a timestamp value and by a size (i.e., number of records).

The snapshot table is a set of records produced by a query in a given period. During this period, all the records generated are stored in a buffer. At the end of the period the snapshot table is filled with records from the buffer. When the snapshot is read, the current content is returned.

5.3.2 High and Low Level Queries

As pointed before, we defined two SQL like languages: the Low and High Level Language. A user submitted query is composed of some Low Level and some High Level Queries, each of them having the role of retrieving and manipulating data and inserting the produced results in a data structure. A query written with the Low Level Language is used to define the behavior of a single device (or a

group of devices abstracted by a single logical object). The main role of Low Level statements is allowing a precise definition of the sampling operations, but also allowing the application of some SQL operators to sampled data. An object that is executing a Low Level Query should maintain a local buffer and perform the following activities:

- to sample data by reading some logical object attributes and inserting the read values into the local buffer.
- to perform SQL operations (selection, aggregation, filtering, grouping, etc.) on the current content of the local buffer and insert the obtained records into the output data structure.

The query can request the execution of both the previous activities, periodically or when an event happens. The local buffer is conceptually unbounded and its size increases indefinitely. Practically, the executor should be able to do a garbage collection of records which are no longer required by SQL operations. It is worth to note that all the processing executed at low level is relative to data extracted from a single logical object and has the goals of discarding bad values and optionally aggregating a group of sampled values before sending them to High Level Queries. If a node cannot process data or maintain a buffer (e.g. an RFID tag) the query will be executed on another physical device (e.g. the RFID reader), but this distribution is hidden to the language. A Low Level statement can contain also an `Execute If` clause allowing the definition of the set of logical objects that will execute the query, in terms of conditions on logical object attributes. High Level Queries take one or more *streams* (generated by Low Level Queries or other High Level Queries) as input, perform SQL operations on windows extracted from *input streams* and insert the generated records in an output data structure. The activation of a High Level Query can be specified either in terms of a time period or in terms of an event (insertion of a record into a stream). Note that the High Level Language has similar functions as the TinyDB language, because both are used

to manipulate *data streams* coming from sensors. The Low Level Language has not a counterpart in TinyDB and can be thought as a mechanism to *generate data streams* corresponding to the TinyDB sensors table (see figure 5.1).

5.3.3 Pilot join operation

Analyzing some case studies, we realized that in many real situations *a sampling on a node should be started if and only if a certain value has been retrieved from a sampling done on another node*. For example, suppose that a user requires a temperature monitoring of all the wine pallets placed in the trucks that are currently in a given parking area. If temperature sensors are mounted on the pallets, the sampling operation on a node should be activated only when its truck is in the parking area. Truck locations are detected by position sensors, that are nodes physically different from the temperature nodes and not directly connected to them.

The above consideration suggested us that a specific operation should be supported to allow sampling activation depending on data sampled from other nodes. We called this new operation *Pilot Join*, because it is conceptually similar to the SQL join operation, but it is used to activate the execution of a Low Level Query on logical objects.

Two kind of Pilot Join are possible:

- *Event based Pilot Join*. When an event happens (i.e. a record is inserted into a stream) a given set of nodes should start sampling (typically for a fixed period). Suppose that, in the previous example, the temperature must be sensed once every time a truck enters the parking area; in this case an event based pilot join is required.

- *Condition based Pilot Join*. A continuous sampling (with frequency f) should be done on all the nodes that are connected to a base station that is in a list of base stations satisfying given criteria. This list is periodically updated with a frequency lower than f: this behavior is obtained using a snapshot data struc-

ture. Consider again the previous example. Suppose that a running query is sensing (with low frequency) the position of all the trucks and inserting them into a stream. Suppose also that the required behavior of the system is the continuous sampling of temperature sensors mounted on pallets whose last monitored position is in the parking area. In this case a condition based pilot join must be used.

5.4 Query processing and examples

A user-defined query can be expressed as a graph: nodes are either data structures or queries, while edges represent information flows and *pilot join operations*.

A real example of a query graph is reported in figure 5.3: it requires a temperature sampling only on the pallets placed in the nearest truck to a given point. The query LLQ1 represents the query that periodically (with period T) samples trucks positions through GPS. HLQ1 is activated every T instants to find the truck nearest to the point P. The ID of the base station mounted on that truck is then inserted in the snapshot *NearestTruck*. The query LLQ2 fills the output stream *Temperatures* and is activated on all the logical objects abstracting temperature sensors and currently connected to the base station with the ID contained in *NearestTruck*.

Figure 5.4 shows the execution of a query in the system, providing an example of the process of query decomposition and distribution. When a query is sent to the system (Q1), the query analyzer parses it and extracts the definition of the nodes composing the query graph. Needed data structures are instantiated and the execution of High Level Queries is started (Q2). Then, the set of logical objects that will take part in each Low Level Query is identified using a registry. Finally, the system starts sending logical objects the commands needed to execute Low Level Queries (Q3, Q4), that are expressed at a higher abstraction level with respect to the set of commands directly intelligible by physical devices. Abstract requests (Q3, Q4) are translated by logical objects into concrete ones (Q5, Q6, Q7) in order

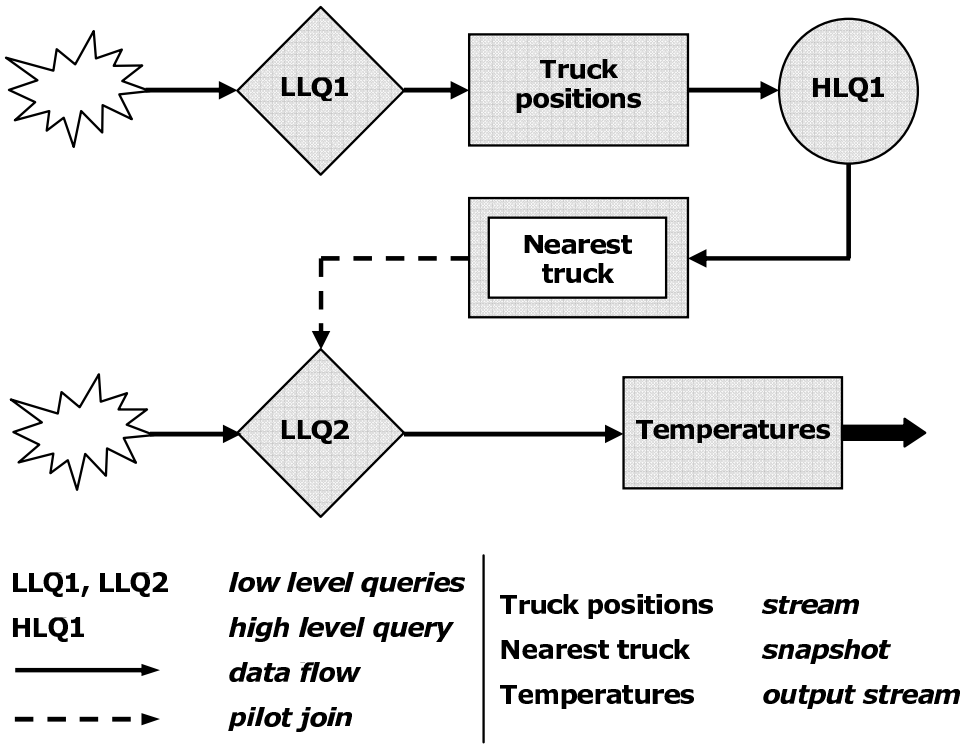


Figure 5.3: Query graph example.

to allow physical devices to execute them. The set of rules used to perform this operation are represented in the figure by the policy container. It is to be noticed that the same abstract command can be translated into different concrete commands if it should be executed on different devices, recognizing different concrete policies constraints. For example, in the figure, Q3 has been translated to Q5 for a class of devices and to Q6 for another class of devices. When all the query components are started, produced data flows go from logical objects to the output data structure, following the path shown in figure 5.3. If required by the user (with a specific language clause) the set of logical objects participating to the query is updated periodically or when some events happen.

To better define a temporal semantics for the Pilot Join operation, a native timestamp concept was introduced: it is a timestamp field attached to each record produced in the system. Low Level Queries are the first elements of the query graph

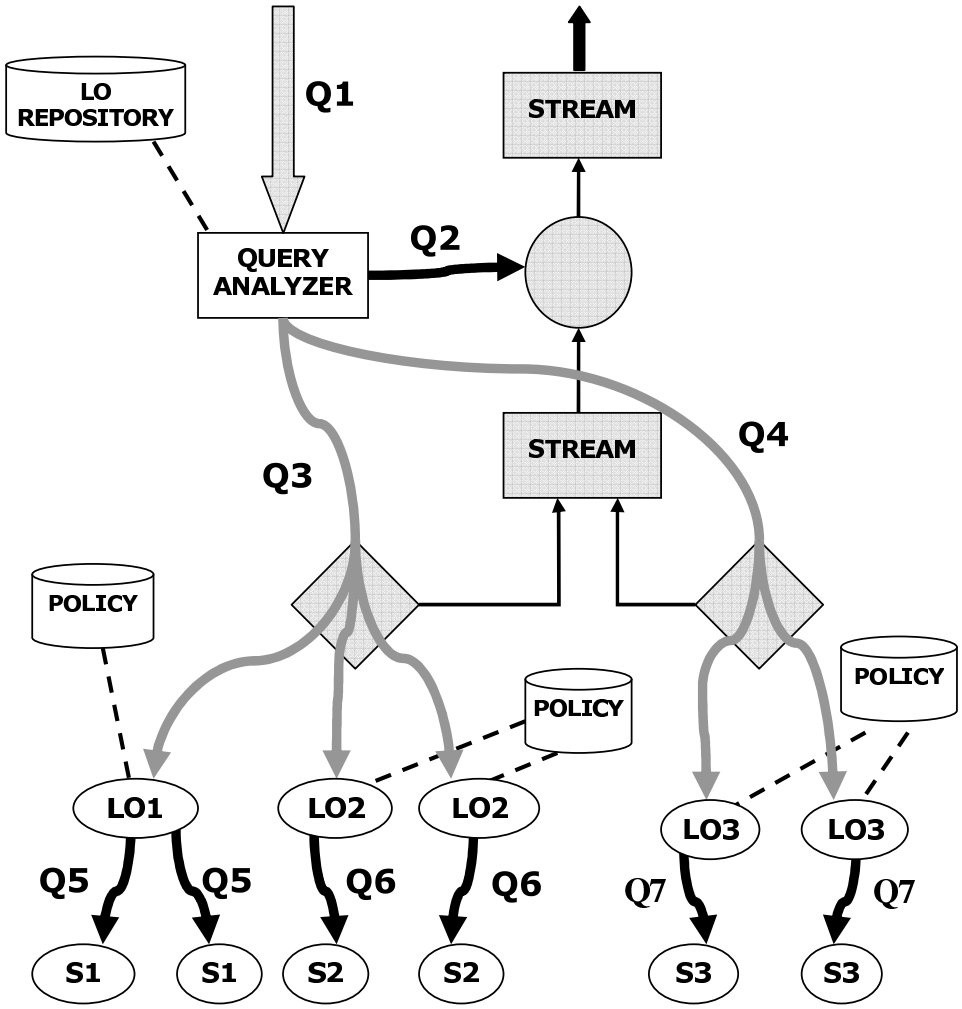


Figure 5.4: Query decomposition and distribution process.

that generate native timestamp values. As said before, when they are activated, these queries compute some SQL operations on data contained in their local buffer and insert the obtained records in an output data structure. They set the native timestamp field of these records equal to the current timestamp, (i.e. the timestamp at which the query was activated). A synchronization algorithm is used to share the current timestamp among all the system nodes.

Each record generated by High Level Queries is timestamped with the native timestamp of the record that caused the event, if the query is event based, or with the activation timestamp, if the query is activated periodically.

In the following some examples - drawn from the ART DECO case study related to wine production and transport processes - are provided to show the peculiarities of our language with respect to TinyDB and GSN. Suppose that the vineyard is equipped with a set of wireless nodes, having on board a temperature and a humidity sensor. The first query we consider has the role of monitoring environment parameters of the area in which wine is cultivated. More specifically we want to sample temperature and humidity every 30 minutes, returning these values, together with the location of the sensor, *only if the sensed temperature is in a critical range*. In order to provide more accuracy in the results, each node is required to sample its sensors every ten minutes and to consider the average of the three last values. In this situation, the user submitted query is only composed of a Low Level statement:

```
CREATE OUTPUT STREAM EnvironmentParameters
(sensorID ID, temp FLOAT, humidity FLOAT, locationX FLOAT, locationY FLOAT)
AS LOW:
    EVERY 30 m
    SELECT ID, AVG (temp, 30 m), AVG (humidity, 30 m), locationX, locationY
    SAMPLING
        EVERY 10 m
    EXECUTE IF EXISTS (temp) AND is_in_Vineyard(locationX, locationY)
        REFRESH EVERY 5 m
```

Unlike TinyDB, our language can run a unique query simultaneously on dif-

ferent kinds of physical devices, thus supporting runtime heterogeneity. In fact, note that the `Execute If` clause requires that the query be executed on all the nodes currently placed in the yard and having on board a temperature sensor and it does not restrict the query execution to wireless nodes only. Therefore suppose that a worker is in the yard with a PDA which can sense the current temperature: this PDA will execute the query exactly as the wireless nodes (if the PDA has not a humidity sensor on board, null values will be produced). In this example, it should be noticed that the location attribute is static for wireless nodes (and it is configured during the network deployment phase), while it is dynamic for PDAs: if a worker enters the yard his PDA will start executing the query as soon as the *Execute If* clause will be reevaluated, possibly recovering data which should be provided by a "dead" sensor.

Another important peculiarity of our language is the ability of querying the network state as "normal" data. Suppose that, in the previous example, we want to monitor the power state of the wireless nodes in order to detect low powered devices. The following query can be written:

CREATE OUTPUT STREAM LowPoweredDevices **(**sensorID **ID)**
AS LOW:
EVERY ONE
SELECT ID
SAMPLING EVERY 24 **h**
WHERE powerLevel **=** *low*
EXECUTE IF deviceType **=** "WirelessNode"

The previous queries don't make use of the High Level Language which, for instance, is often used when spatial aggregations have to be performed on sampled data. As an example, consider a query that returns the number of low powered wireless nodes: a High Level Query performing the count aggregation can be written on the stream generated by the previous Low Level Query:

CREATE OUTPUT STREAM NumberOfLowPoweredDevices (counter **INTEGER)**
AS HIGH:
EVERY 24 **h**

```
SELECT COUNT(*)
FROM LowPoweredDevices(24 h)
```

While the GSN concept of wrapper is quite similar to our Low Level Queries and that of virtual sensor is quite similar to our High Level Queries, the next example shows the pilot join operation that is the feature really improving our language potential with respect to GSN.

We consider the monitoring of wine during the transport. In particular, suppose that every truck is equipped with a GPS and a base station. Suppose also that each pallet has a temperature sensor used to sense the temperature of the contained bottles. The query requires to produce as output the list of pallets whose temperature exceeded a certain threshold while the truck was traveling through a given zone, which is considered particularly critical:

```
CREATE SNAPSHOT TrucksPositions (linkedBaseStationID ID)
WITH DURATION 1 h
AS LOW:
    SELECT linkedBaseStationID
    SAMPLING
        EVERY 1 h
        WHERE is_in_CriticalZone(locationX, locationY)
    EXECUTE IF deviceType = "GPS"

CREATE OUTPUT STREAM OutOfTemperatureRangePallets (palletID ID)
AS LOW:
    EVERY 10 m
    SELECT ID
    SAMPLING EVERY 10 m
        WHERE temp > [threshold]
    PILOT JOIN TrucksPositions
        ON baseStationID = TrucksPositions.linkedBaseStationID
```

In this query the pilot join operation is used to activate the temperature sampling only on the pallets contained in the trucks that are driving into the critical zone. Table 5.2 shows the attributes of the logical objects involved in the execution of the query: GPS mounted on trucks and sensors attached to pallets. Note that the

Table 5.2: **Logical objects used in the transport monitoring query**

GPS - Logical object wrapping a GPS device			
Field Name	**Data Type**	**Field Type**	**Description**
ID	ID	ID	Logical object identifier
linked BaseStationID	ID	Static	ID of the base station mounted over the truck
locationX	FLOAT	Dyn. prob.	Sensor location X coordinate
locationY	FLOAT	Dyn. prob.	Sensor location Y coordinate
deviceType	STRING	Static	Type of device

WSN node - Logical object wrapping a single WSN node			
Field Name	**Data Type**	**Field Type**	**Description**
ID	ID	ID	Logical object identifier
baseStationID	ID	Dyn. non prob.	ID of the base station the WSN node is currently connected to
temp	FLOAT	Dyn. prob.	Sampled temperature

interface of logical objects wrapping GPS devices has an attribute “baseStationId” that retrieves the id of the base station mounted on the same truck (this is a static attribute, whose value is defined during the network deployment time).

Chapter 6

Experimental phase

To validate the effectiveness of the suggested approach, we considered two real monitoring applications:

- a real monitoring application envisaging a prototypal deployment of the WSN designed for monitoring the Australian Coral Reef (Section 6.1). The prototype application aims at monitoring temperature and lightness in the water at different depths in Moreton Bay (Brisbane, Australia).
- a real civil buildings and unstable cliffs monitoring application for the evaluation of the geological risk in a rock faces collapse scenario (Section 6.2).

6.1 Applicative case: coral reef monitoring

The proposed framework has been applied to a real monitoring application envisaging a prototypal deployment of the WSN designed for monitoring the Australian Coral Reef. The prototype application aims at monitoring temperature and lightness in the water at different depths in a circular 2800 m^2 sea area in Moreton Bay (Brisbane).

The developed proposed system encompasses the sensing activity, a local transmission from sensor nodes to the gateway, a remote data transmission from gate-

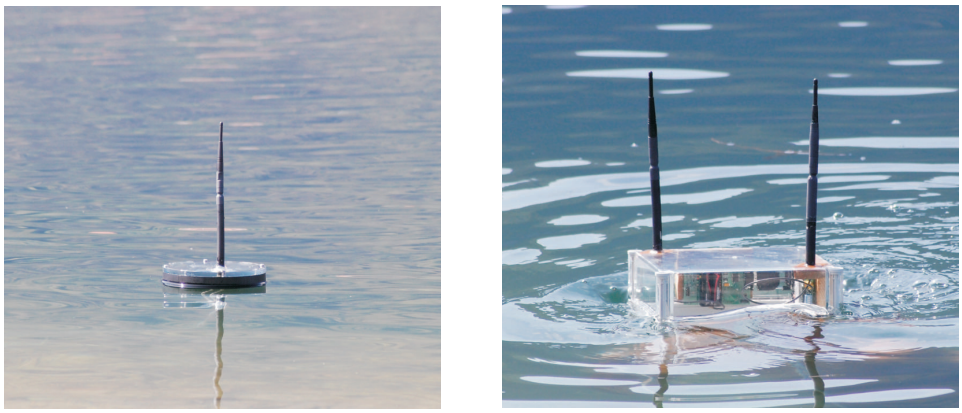
Figure 6.1: Final deployment in Moreton Bay

ways to the control center and subsequent data storage in a DB and visualisation.

The developed WSN is composed of 9 sensor nodes (see Figure 6.2.a) and a gateway (see Figure 6.2.b) that are inserted in buoys anchored to the coral reef and organized in a star topology.

In Figure 6.1 is shown the final deploymet. The sampling frequency of both the temperature and the lightness sensor is $1Hz$; the acquired measurements are then averaged (to reduce acquisition noise) and sent to the gateway every $30s$. Each DATA message is composed of 24 bytes with 14 bytes of payload. Table 6.1 shows the transmission parameters of the adaptive power-aware TDMA (presented in Chapter 2) that have been experimentally identified for this application.

The technological infrastructure for the processing and the transmission unit in the proposed framework relies on the Crossbow MICAz [16]. This sensor node provides a 7,37 MHz ATMega 128L processing unit (with 4KByte for the RAM and 138 KByte for the program memory) and a CC2420 transceiver radio (single-chip 2.4 GHz IEEE 802.15.4 compliant RF transceiver with a 250 kbps effective



a) b)

Figure 6.2: a) A sensor node; b) the gateway

data rate).

The technological infrastructure we rely on for the radio link is the MaxStream 2.4GHz XStream Radio Modem [41]. Thus, the gateway uses the CC2420 radio to communicate with the sensor nodes (local transmission) and the XStream Radio Modem to transmit remotely the data - the measurements received from the sensor nodes together with the gateway measurement - to the control center (remote transmission).

Figure 6.3 presents the temperature, the brightness and the current generated by solar panels for a node acquired in 2 days. Focusing on the brightness and current subplots, it is possible to appreciate the night-day periodicity. Is is worth noting that temperature is measured at the sea bottom. This ise the reason for the peculiar behavior of the temperature in the temperature subplot: the temperature increase with a delay with respect to the brightness due the thermal inertia of water.

Figure 6.4 shows the measurement of the two batteries packs and the current generated by solar panels for three nodes: *gateway*, *node 1*, *node 2*.

Each node presents a different energetical situation and it is possible to identify three different behavior of MPPT.

In figure 6.4, the x axis (the same for the three subplots) is the sequence of sampling over time. The samples are acquired every 30 second. Hence, the equivalent

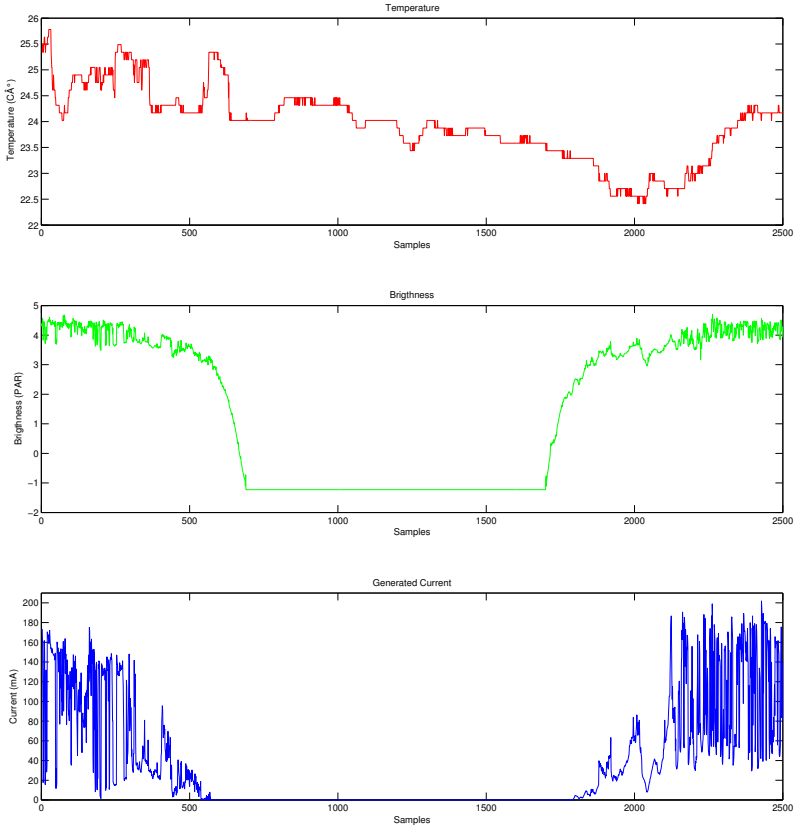


Figure 6.3: Sensors data

Protocol Parameter	Aim	Value
PERIOD	Time of the duty cycle	30s
DELTA	Guard time	1s
TX_TIME	Time of the TDMA time slot	0.142ms
TAB_TIMEOUT	Time within the gateway sends the TDMA TABLE	2s
ACK_TIMEOUT	Time within the gateway sends the ACK message	1s
SYNC_TIMEOUT	Time within the gateway sends the SYNC message	4s
LOST_CYCLE_TIMEOUT	Time between the SYNC_TIMEOUT and the reception of the next SYNC message	25s
RETRY_TIMEOUT	Random time the sensor node waits to re-send the SUBSCRIBE message. RETRY_TIMEOUT = PERIOD/2 + U(0,10)	Random

Table 6.1: Power-aware TDMA protocol parameters.

time span is about 24 hours.

Left y axes of subplots represents the span of the batteries voltage (V), while the right y axes represents the span of the current generated by solar panels (expressed in mA). In the first subplot it is possible to observe a typical battery switch, that is performed by MPPT when a battery pack is discharged. At $x \cong 1150$ the battery pack 1 (dashed line) is dramatically discharged. The system starts to use the other pack. When the current generated by solar panel is available, pack 1 is fully recharged.

In the second subplot, the battery pack 1 (dashed line) is used up to $x \cong 630$; in this instant of time, the MPPT switches to battery pack 2 (used up to $x \cong 1880$). At $x \cong 1800$ solar panels starts to recharge the battery pack 1. However, this pack is not fully recharged when MPPT switches again the battery packs. The side effect is the quick discharge of the pack 1. The next switch will happen when pack 2 is fully recharged.

In third subplot the state of the gateway batteries is presented. Notice that both battery packs have enough energy; hence, the MPPT hardware does not provide

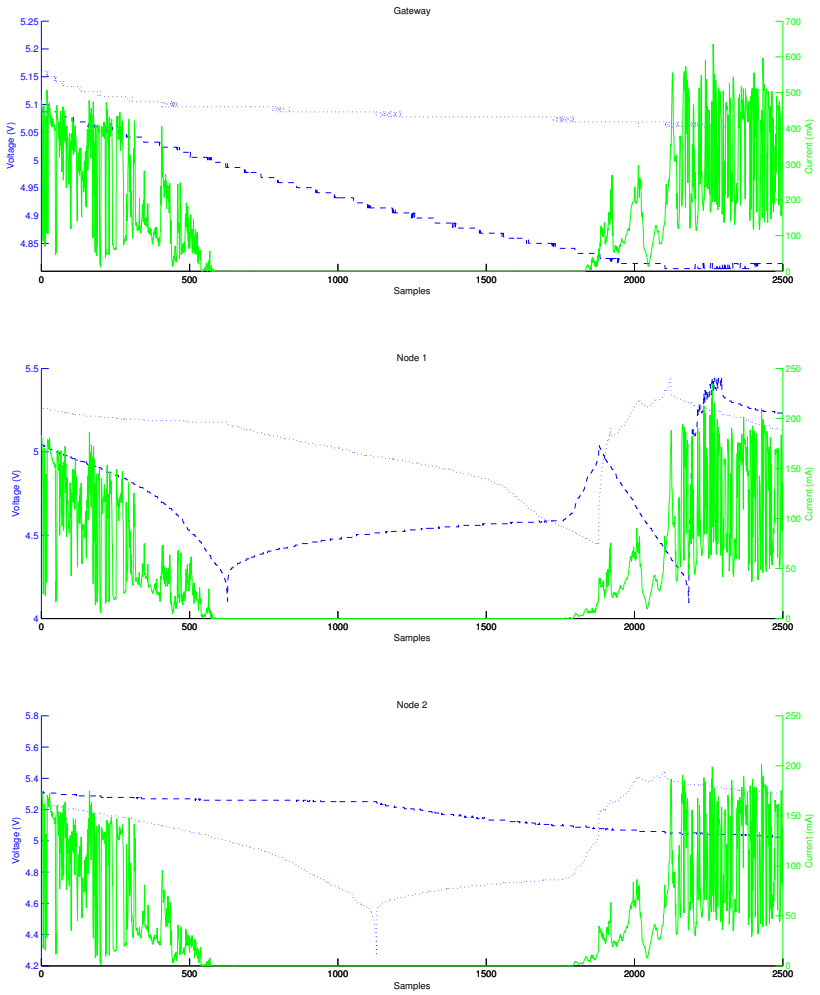


Figure 6.4: figura batterie

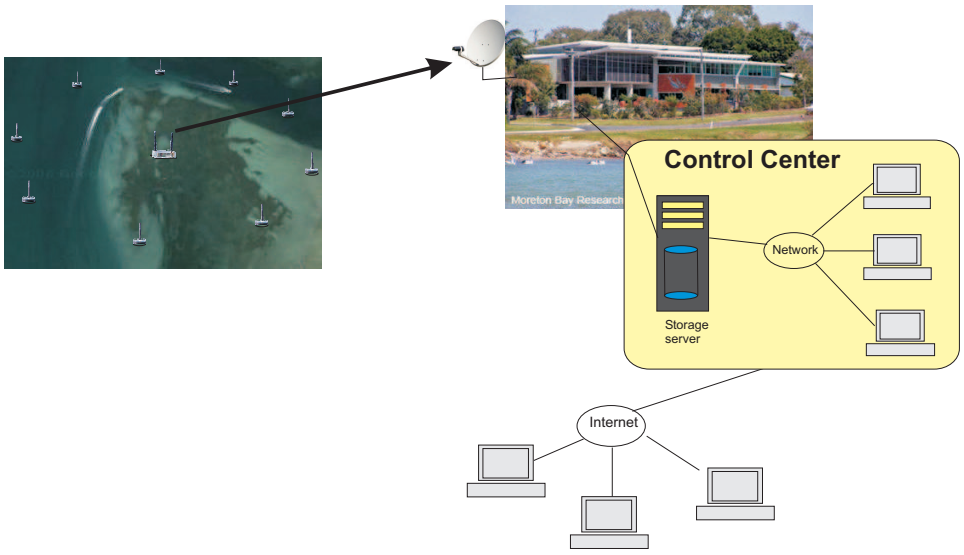


Figure 6.5: The remote transmission and the control center.

current to their recharge.

6.1.1 Remote transmission, data storage and visualization

The power-aware TDMA protocol presented in the Chapter 2 is used in the proposed framework to allow the gateway for collecting the sensor node measurements. Once these data are available at the gateway (i.e., after the DATA_READY state), it can forward them (together with its own measurement) to the remote center by the radio link. The radio link is thus activated only once in the TDMA cycle and remains off for the rest of the time.

At the remote control center the counterpart radio link listens to the channel for the gateway transmission. Once the message arrives, a software routine is activated to insert the data contained in the payload in the database server (see Figure 6.5).

Human operators at the control center (or outside the control center through internet) access the database server with a SW application that allows for

- accessing the current and the historical data of the network,

- controlling the status of the sensor nodes in the network,
- controlling the status of the network.

6.2 Applicative case: civil protection and homeland security

The proposed framework has been also considered in a strategic research project of Politecnico di Milano, PROMETEO, focusing on civil protection and homeland security. In particular, one of the main goal is the evaluation of the geological residual risk in first emergency actions in a rock faces collapse scenario. The micro acoustic signal detection and compression algorithm, which is here presented, perfectly fits in this application. The first tests in laboratory and on the field provide successful results. The developed system will also involve aggregation, processing, storage and publication of acquired data. In the next subsections we discuss about a credible node level signal processing for the microacustic bursts.

6.2.1 Node-level signal processing algorithms for microacoustic bursts

Collapse of rock faces and stability assessment of buildings is related to the geological substrate or the nature of the concrete and the presence of fractures which influence the stability properties of the envisaged material. A wireless sensor network-based approach has been suggested to monitor crack formation which, due to the high sampling rate of sensors, requires both online temporal identification of the event and compression of the same before transmittal to a remote control room (the action containing the energy consumption of the unit by triggering and reducing the radio activity). The Section proposes a simple -yet effective- algorithm for detecting microacoustic emissions generated by formation/organization of cracks and identifies a suitable lossless technique for microacoustic event compression. Finally, to satisfy real time performance, the optimal processing chain has been implemented on a FPGA chip to be mounted onto the WSN units.

6.2.2 Introduction

Monitoring civil buildings, unstable cliffs and slopes and rock faces is crucial to evaluate the dynamic of structures as well as provide data for stability and structural properties assessment (for a possible subsequent infrastructure collapse forecast). These aspects have been addressed in the related literature with important monitoring techniques mainly based on wired solutions (e.g., see [42] [43] [44]). However, the need of wireless solutions in several applications (basically those for which energy supply or cabled systems do not constitute a feasible framework) are pushing research towards wireless sensor networks (despite the fact that, up to now, several still-open issues need to be addressed to make the technology credible, e.g., clock synchronization, effective energy harvesting mechanisms, power-aware node and network management, real time execution). In this direction, we face the energy management aspect in a specific application of WSNs by reducing the events to be transmitted at the base station through an event selection and compression algorithm. The event to be looked for refers to local microacoustic/seismic activity, a phenomenon associated with the hierarchical organization of fractures inside the rock [45] or concrete-based structures. Microacoustic bursts are generated in correspondence with the fracture evolution and cover a bandwidth up to 10kHz. Such a high bandwidth requires, in turn, a high acquisition rate, hence introducing communication and data storage problems in WSN units (problems introduced by their limited resources).

By triggering the acquired data in search for events we largely reduce data to be processed and transmitted, operation resulting in a net gain in energy savings (and making feasible the monitoring application here presented). A typical scenario presenting the technique applied to rock faces monitoring is depicted in figure 6.6 while the technique applied to civil structures monitoring is given in figure 6.7.

Microacoustic signals can be acquired through geophones, piezoelectric or MEMS accelerometers (all solutions are feasible in a wireless sensor network use). More in detail, geophones are surely the most sensitive sensors but, relying on a

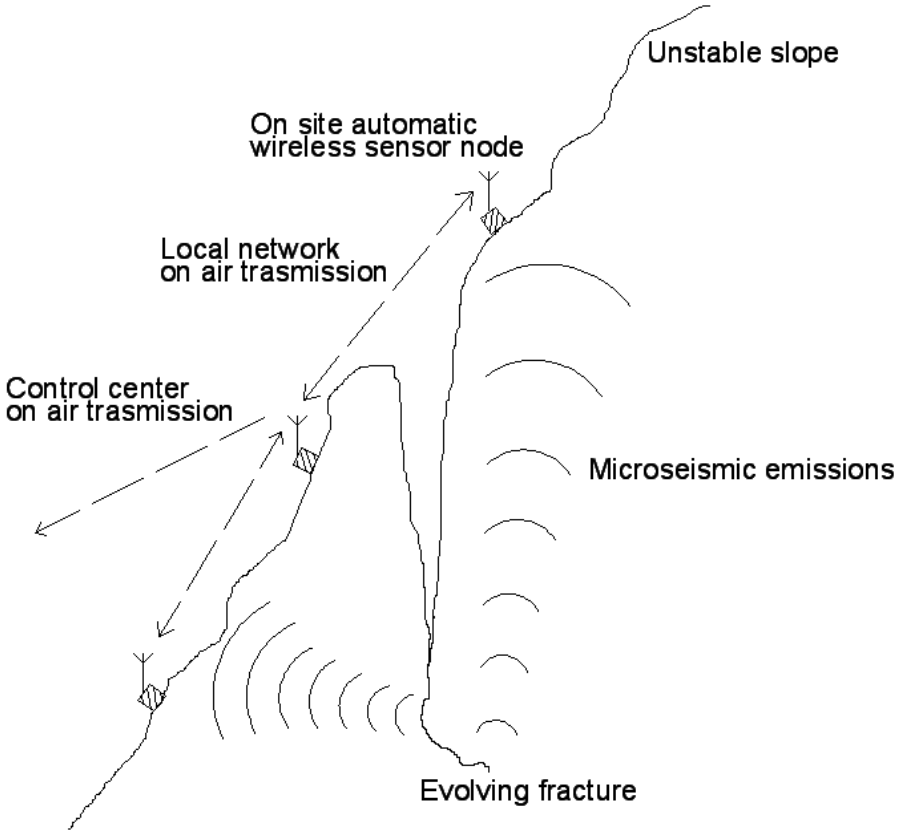


Figure 6.6: Application scenario: the case of rock faces monitoring.

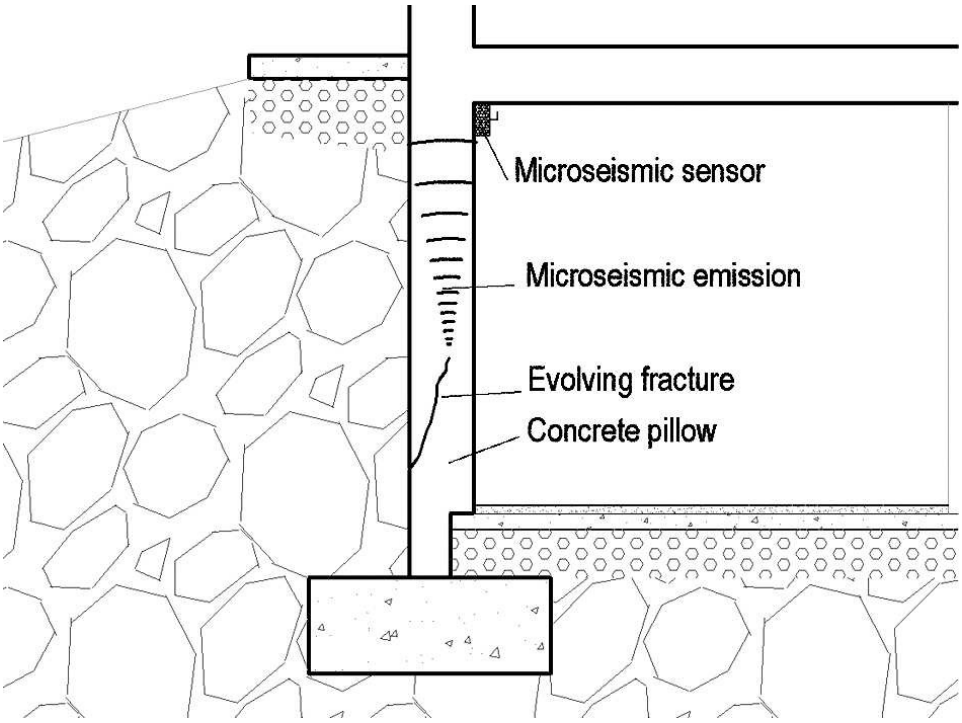


Figure 6.7: Application scenario: the case of civil buildings monitoring.

seismic mass displacement reading mechanism, are of large dimensions; on the other hand, they are passive with energy consumption associated solely with the signal conditioning and the ADC conversion. Piezoelectric sensors are sensitive, provide a good noise immunity and a wide bandwidth (more than 10kHz, depending on the particular model) but their cost is quite high.

Of particular interest are MEMS micro fabricated mechanical accelerometers that grant noise immunity in the order of the piezoelectric one and bandwidth up to 1.5-2kHz [46]. Surely, MEMS technology is appealing in the envisaged application due to its price and size; moreover, the technology grants that the metrological characteristics of MEMS are in line with the application requirements and their power consumption is becoming very small (below 10mW per axis along with the signal conditioning electronics).

The work focuses on MEMS-acquired signals; however the validity of what here suggested is independent from the envisaged sensor for microacoustic event

detection.

The first step to decrease the bandwidth required for a raw transmission of data requires development of a HW trigger based on the averaged energy of the input signal. The second step requires extraction of the burst from the signal and its compression to reduce the number of bits to be transmitted to the remote station. As a consequence, data stream compression techniques allow us also for reducing communication power, network traffic load and data storage space.

In general, data compression is achieved through exploitation of spatial and temporal correlations present in data streams but application specific properties of the data can be used to improve the compression rate. In this section we present and discuss a set of lossless compression algorithms specifically designed for being effective in data streams containing asynchronous events and suitable for execution in wireless sensor network units.

The structure of the section is as follows. Section II introduces the event detection and extraction algorithm and investigates and compares the performances of 6 different lossless compression techniques. Section III overviews the overall system design to be placed on the node unit.

6.2.3 Microacustic signals: event extraction and compression

As presented in the introduction, the processing chain requires event identification and extraction followed by event compression.

The first step relies on a simple algorithm able to extract bursts in noisy signals. Briefly, the algorithm operates by comparing two moving averages of the squared signals computed on a different number of samples n_1 (e.g., 60 samples) and n_2 (e.g., 500), with $n_2 \gg n_1$. Intuitively, the first window behaves as an event detector being sensitive to structured variations in the signal energy; conversely, the second one, represents a reference baseline. In case of a burst the two averages significantly differ and the event can be promptly detected through a suitable threshold identified on the basis of the event characteristics. This algorithm has shown to be

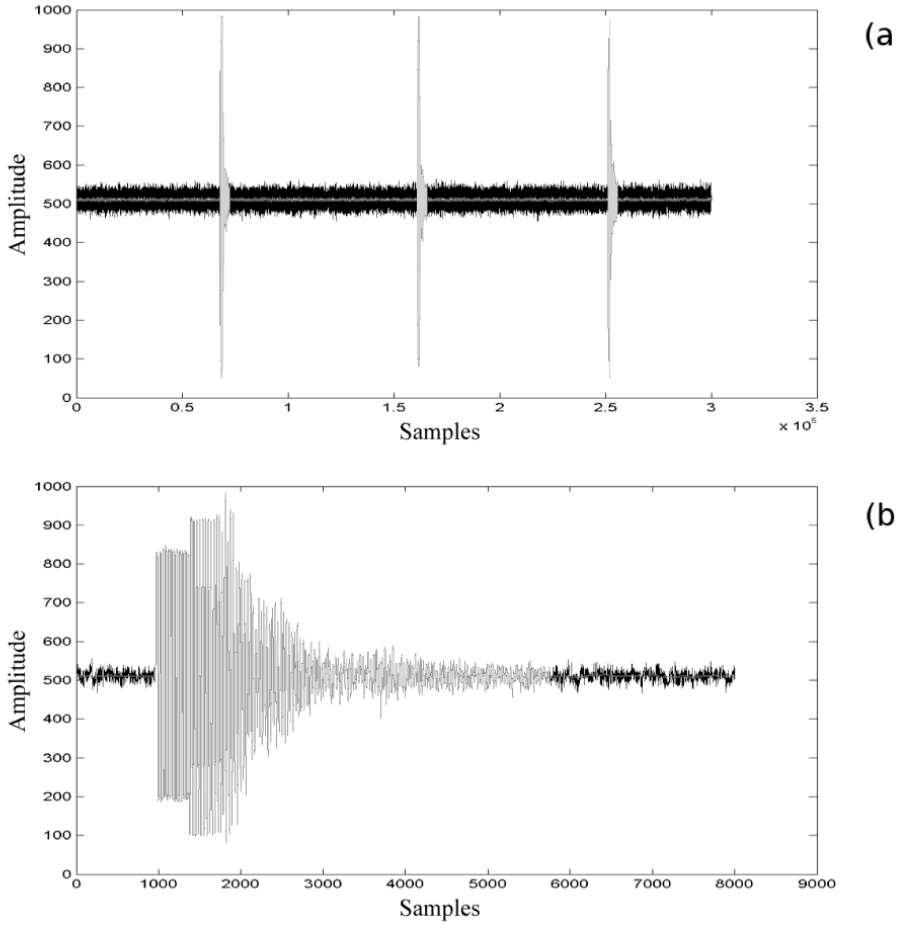


Figure 6.8: The event detection algorithm applied to a real burst signal.

simple enough to be placed on a low power FPGA yet proven to be effective.

Figure 6.8a shows a typical event associated with generation of microacoustic bursts induced by cracks formation; a detail of a burst is given in figure 6.8b. Signals have been acquired with an experimental campaign, where 6 different types of concrete rafters (differing in mix composition) have been subject to three points bending stress in a testing press. The press applies a controlled mechanical load in the center position of the rafter through a knife, which is counterbalanced by other two knives posed beneath the rafter at its sides [47]. The MEMS sensor has been fixed on the rafter by means of an appropriate glue not to introduce additional

$Signal\,no.$	$Orig.\,size$ $(samples)$	$Final\,size$ $(samples)$	$Events$	$Compr.\,rate$ $(\%)$
1	5403201	160450	40	97.1
2	5190577	107375	22	98.0
3	4469756	292500	76	93.5
4	5092080	73425	30	98.6
5	5192609	359250	88	93.1
6	5170796	133050	44	97.1

Table 6.2: Data size reduction percentage of the extraction algorithm on 6 different experiments.

vibrational modes.

The lighter part of the signal in the figure is considered for compression; the black rejected being background noise. As we can expect this operation is very effective in reducing data rates since the event appears with low probability. Data rate reduction results are summarised in table 6.2 for the 6 experiments.

The second part of the algorithm refers to event compression. Although lossy compression techniques grant better compression performance, their use in our target application is not adequate. In fact, for this application, it is requested that the entire event is transmitted to the base station for further processing and data analysis; waveforms distortion brought by lossy compression algorithms are thus not acceptable (at least until the phenomenon has not been fully studied and observed). Despite the fact that a vast literature exists on lossless compression [48], only few algorithms show to be feasible in WSNs due the strong HW constraints posed by commercial wireless sensor units [49] (e.g. MoteIV TmoteSky and XBow MicaZ).

Here, we study and contrast performances of Plain Huffman, Delta Coding + Plain Huffman, Adaptive Huffman, Delta Coding + Adaptive Huffman, Lempel-Ziv SS type, Delta Coding + Lempel-Ziv SS type compression techniques.

Delta coding is commonly used in acoustic signal analysis [50]. There, each encoded symbol is the difference between the current sample and the previous one. This coding technique aims at reducing the dynamics of the signal and, hence, the number of symbols to be considered; delta coding is particularly effective when

Algorithm	Comp compl.	Memory compl.
Delta	$O(n)$	$O(1)$
Huffman	$O(lm) + O(m^2)$	$O(l)$
Adapt. Huffman	$O(nk)$	$O(k)$
LZSS	$O(ln)$	$O(n)$

Table 6.3: Computational and memory complexity comparison

Parameter	Explication
n	number of processed signal samples
m	number of different symbols
l	processing windows size
k	dictionary size $\left(k = \frac{m+1}{2}\right)$

Table 6.4: Algorithms parameters explication

combined with other algorithms.

The Huffman coding [51] is a transformation from samples space to symbol space. Coded symbols have different lengths that are inversely proportional to the samples occurrence in input streams. Encoder and decoder use the same "dictionary" to transform one space into the coded one and vice versa. Adaptive Huffman coding [52] is an extension of the Huffman coding allowing the stream encoding and the decoding with an adaptive size sample window where Huffman does not. LZSS coding [53] is based on the substitution of stream symbols (or group of them) with a reference to the first occurrence in a sliding data window.

The computational and memory complexity for different compression algorithms is given in table 6.3 (with parameters' means is reported in table 6.4).

Signals are encoded using a 10-bit ADC. The compression percentage provided by the envisaged compression algorithms averaged over more than 250 acoustic bursts is given in figure 6.9 for the 10 and 7 bits.

Figure 6.10 show the compression percentage of Huffman, Ad. Huffman and LZSS algorithms for different value of operating parameter: in particular the block data size for the hufman and LZSS and the vocabluary size for the adaptive one.

A 7 bits code, which implies an information loss, shows to be particularly effective in compression once compared with the 10bits one both for the Huffman-based

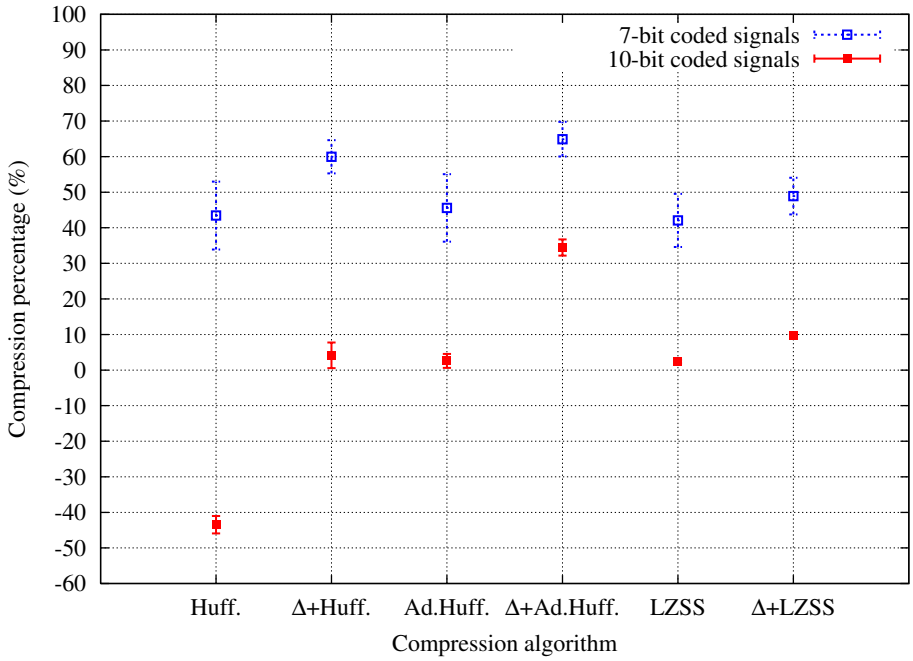


Figure 6.9: Compression percentage of the six lossless algorithms on the available bursts

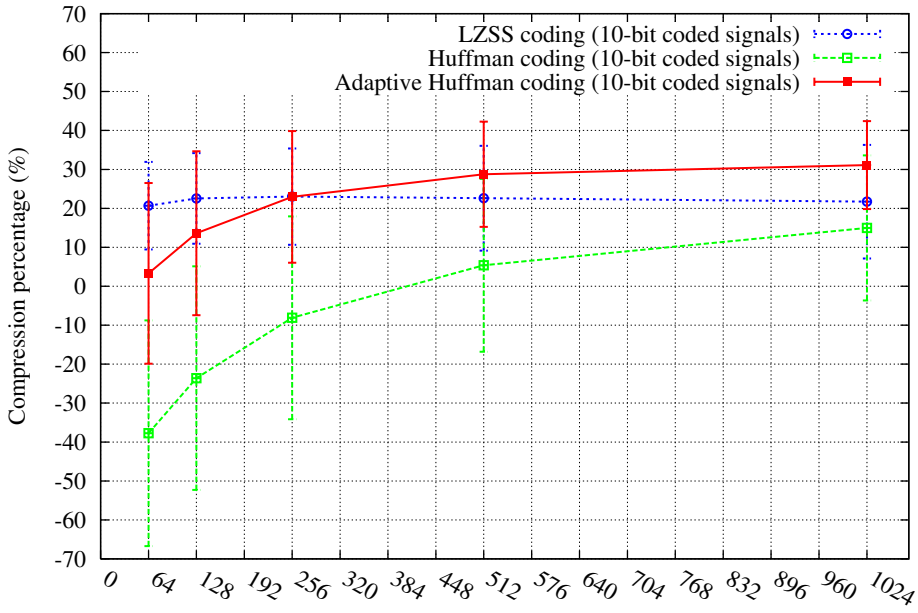


Figure 6.10: Compression percentage of the lossless algorithms for different operating parameter

and the LZSS codes. In particular, Huffman compression gain derives from the fact that both the symbol vocabulary and the coded wordlength are monotonic with the number of bits. Differently, in LZSS, a bit reduction increases the probability that two consecutive words are identical (we have a collapse of classes).

In Huffman we have to transmit the symbol dictionary in correspondence with each processed block; this results in a significant overhead when the vocabulary is particularly reach, as it happens in the figure in correspondence with a 10 bits coding (negative value in the plot which implies an increment of the number of bits to be transmitted). As a conclusive note, the Delta Coding with the Adaptive Huffman algorithm is the most effective compression algorithm, with an average compression rate of $\sim 65\%$ and contained required computational load.

Event detection followed by Delta Coding and Adaptive Huffman algorithms have been implemented on a FPGA chip to satisfy real time processing requirements. The final algorithm to be implemented in FPGA is as follows.

The event detection algorithm operates by comparing the means computed on different-sized sliding windows and evaluating their discrepancy with a threshold function of the signal variance. More in detail, let $\{x_i, \ldots, x_{k+i-1}\}$ a k sample window opened on the incoming data at time i. Evaluate the averages over $k = k_1 = 64$ and $k = k_2 = 512$ samples and the variance of k_2. Since averages and variances need to be evaluated in correspondence with each incoming sample recurrent formulas have been considered:

$$\mu_{k,i+1} - \mu_{k,i} = \frac{1}{k} \sum_{j=i+1}^{k+i} x_j - \frac{1}{k} \sum_{j=i}^{k+i-1} x_j = \frac{x_{k+i} - x_i}{k} \tag{6.1}$$

i.e.,

$$\mu_{k,i+1} = \mu_{k,i} - \delta_{k,i} \tag{6.2}$$

where $\delta_{k,i} = \dfrac{x_i - x_{k+i}}{k}$. For the variance

$$\sigma^2_{k,i} = \frac{1}{k} \sum_{j=i}^{k+i-1} (x_j - \mu_{k,i})^2 \tag{6.3}$$

we have that

$$\sigma^2_{k,i+1} = \sigma^2_{k,i} + \delta_{k,i}\Big(2\mu_{k,i} + x_{k+i}\frac{1-k}{k} - x_i\frac{1+k}{k}\Big) \tag{6.4}$$

An event is detected when $\mu_{k_1,i+1} - \mu_{k_2,i} > \lambda\sigma^2_{k_2,i}$. The λ value has been determined through experiments.

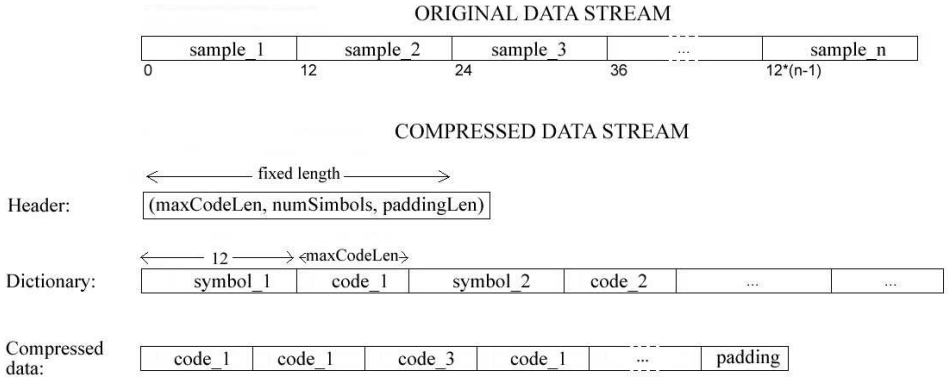


Figure 6.11: Huffman Algorithm.

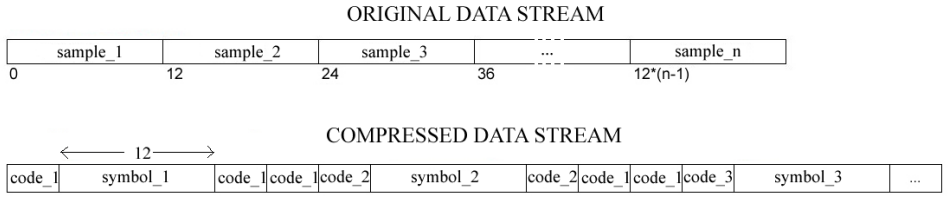


Figure 6.12: Adaptive Huffman Algorithm.

Finally, we have the Huffman coding. At first the incoming delta-transformed data are buffered to create a data block of size $l = 512$ samples. Huffman's algorithm builds a symbol-code vocabulary where the bitlength of a code word is associated with its appearance probability in a given data block (i.e., code lengths are inversely proportional to the symbol frequency in a data block). The vocabu-

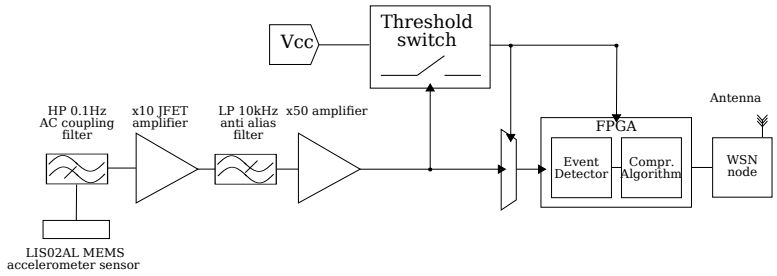


Figure 6.13: System architecture.

lary, which depends on the symbols within the data block, is sent along with the compressed data stream for subsequent decompression. The high level description of the algorithm is given in figure 6.11 where, starting from the raw data, the Huffman code is given (i.e., an header containing the length of the longest coded symbol $maxCodeLen$, the number of symbols composing the dictionary, the number of bits added to create a power of two data stream, the dictionary composed of couples $(symbol, code)$, the current extracted codes and, finally the padding bits).

Differently, Adaptive Huffman does not require a vocabulary retransmission for each block (i.e., stream-oriented). If a symbol occurrence changes over time, only the modification in the vocabulary is transmitted (see figure 6.12).

Finally, the output is then packed and sent to the remote station.

6.2.4 The ad-hoc Hardware

The whole processing system is depicted in figure 6.13. At high level it comprises:

- *An analog conditioning stage* which amplifies the signal coming from the microacoustic sensor. If a MEMS sensors is employed, the conditioning stage must be able to reject the DC part of the signal (originated by the gravity).

- *An analog to digital converter*, which digitizes the conditioned signal providing a digital data stream to the subsequent acquisition chain.

- *A FPGA* implementing the digital processing encompassing event recognition and compression. The FPGA provides a suitably processed data stream to the wireless unit.

- *A wireless unit* hosting the FPGA and providing data transmission towards the base station.

The analog conditioning stage is mainly composed of a chain of low power and low noise amplifiers to enhance very weak signals generated by microcracks and detected by the sensors. Usually amplifying gains ranging from +30 to +60 dB are employed (depending of the sensor type). A subsequent frequency shaping has been implemented to reduce noise and to provide anti-aliasing and DC filtering [47].

The ADC digitizes the signal for successive processing and digital radio transmission. This part of the system must be carefully designed (in terms of bit accuracy, converter architecture and sampling frequency). In fact, since the ADC is continuously running at the sampling frequency, it is a constant load for the power supply unit. Thus, bit per sample, sampling speed and architecture (in terms of cycles per sample) must be kept at their minimum value. The analog threshold (here depicted but not yet implemented) has to enable the FPGA only when a possible event is detected (de facto it behaves as a first level trigger). Such a trigger keeps the ADC and the FPGA switched off -when not required- to reduce power consumption. The resulting core requires about 20.000 slices (by using Xilinx technology) and consumes 100mW (I/O towards the WSN units included).

6.2.5 Final remarks

The Section presented an algorithm for microacoustic event detection and compression and its hardware implementation on a FPGA. Effectiveness of the algorithm shows that an ad-hoc HW implementation is feasible and provides transmission reduction up to 95%.

Chapter 7

Conclusions and future work

The thesis goal was to develop a methodology for the design of credible WSNs. The credibility of WSN deployments requires the WSN to be energy-aware, robust with respect to perturbations affecting the normal operational life that adapts itself to face a change in the network topology.

In particular, at the network level, we focused on MAC and routing layer suggesting both an adaptive and power-aware TDMA-base MAC layer (which proves to be very satisfactory for its energy-aware capabilities and its robustness to network topological changes - see Section 6.1) and an adaptive self-organizing LLC-base routing algorithm (which, for the first time in the literature, takes into account the effects introduced by finite unit bandwidth and provides a uniform distribution of alive units in the deployment area - feature associated with the QoS of the network).

At the *node level*, the thesis presented both a start/restart framework providing the node-level mechanisms to suspend/resume unit activity and a major change in the core part of the unit operating system to outperform the standard power management policy (increasing the unit lifetime). With these improvements, the information about the residual energy of nodes (information available with specialized hardware and super-capacitors) modules, when necessary, the node power manager for turning on/off power-eager systems parts (e.g., radio) or to save its

internal state (in case of critically low levels of energy). When the energy of the unit returns to an acceptable level (e.g., thanks to energy harvesting mechanisms), the node restores its previous internal state.

At the *application level*, this thesis suggested a language and a middleware for managing data in highly adaptive and pervasive systems (e.g, WSNs) composed by different devices (in terms of technology and functional capabilities). This novel approach provides mechanisms to manage both functional (e.g., definition of operations, parameter settings) and nonfunctional features (e.g., constraints on the offered functionalities and on the Quality of Service) of the envisaged pervasive system.

Moreover, at the application level, this thesis focused on a specific and challenging research issue: detection and compression techniques for micro acoustic signals in monitoring civil buildings and unstable cliffs. The thesis suggested an algorithm for microacoustic event detection and compression and its hardware implementation on FPGA. Effectiveness of the proposed algorithm and ad-hoc hardware implementation is shown.

As presented in Chapter 6, the methodology for the design of credible WSNs presented in this thesis results to be very satisfactory in the considered two real monitoring applications: an environmental monitoring application of the Australian Coral Reef and a homeland security application of unstable rock cliffs monitoring.

A very interesting research field can be the development of a stronger integration between software and hardware mechanisms to suspend/resume the unit activity. The possibility to include the software mechanisms in the hardware platform would allow the definition of smarter, adaptive and more efficient power manager systems.

Another interesting research activity concerns the introduction of sensor fusion techniques to reduce the amount of data to be transmitted between node units and remote center. Moreover, this novel approach would allow the development of

intelligent and distributed measurement systems (where the computation and the intelligence of the application are brought as much as possible towards the nodes) which will become the next-generation measurement systems.

Presently, the proposed methodology focuses on lossless compression techniques (as presented in Section 6.2). A very interesting research field could be the introduction of lossy compression techniques that, by accepting a relative loss in performance, would further on reduce data to be transmitted (and, hence, the power consumption).

Moreover, a large amount of work can be focalized on the development of ”smart” mechanisms to allow the preemption in power-aware WSN operating systems. This step would allow the development of even more efficient power manager systems.

Bibliography

[1] C. Alippi, R. Camplani, C. Galperti, M. Roveri, and L. Sportiello, "Towards a credible wsns deployment: a monitoring framework based on an adaptive communication protocol and energy-harvesting availability," *Accepted at IEEE IMTC 2008. Under publications.*, 2008.

[2] C. Alippi, R. Camplani, and M. Roveri, "A hierarchical llc routing algorithm for wsns," *IEEE International Workshop on RObotic and Sensors Environments, ROSE 2007, Ontario, Canada, 12-13 Ottobre 2007.*, pp. 1–6, 2007.

[3] ——, "An adaptive, llc-based and hierarchical power-aware routing algorithm," *Submitted at IEEE Transactions on Instrumentation and Measurement*, 2007.

[4] R. Camplani, "Titolo paper," *Accepted at IEEE PerCom 2008. Under publications.*, 2008.

[5] C. Alippi, R. Camplani, and G. Galperti, "Lossless compression techniques in wireless sensor networks: Monitoring microacoustic emissions," *IEEE International Workshop on RObotic and Sensors Environments, ROSE 2007, Ontario, Canada, 12-13 Ottobre 2007.*, pp. 1–5, 2007.

[6] D. E. et al., "Instrumenting the world with wireless sensor networks," *Proc. of the IEEE Acoustic, Speech, and Signal Proc. Conf.*, vol. 4, pp. 2033–2036, 2001.

[7] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," *Proc. ACM international workshop on Wireless sensor networks and applications*, pp. 88–97, 2002.

[8] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh, "Monitoring volcanic eruptions with a wireless sensor network," *Wireless Sensor Networks, 2005. Proceeedings*, pp. 108–120.

[9] C. Hartung, R. Han, C. Seielstad, and S. Holbrook, "FireWxNet: a multitiered portable wireless system for monitoring weather conditions in wildland fire environments," *Proc. International conference on Mobile systems, applications and services*, pp. 28–41, 2006.

[10] J. H. J. F. M. V.Raghunathan, A Kansal, "Design considerations for solar energy harvesting wireless embedded systems," *Proc.IEEE/ACM IPSN*, 2005.

[11] "Ieee pervasive computing," January-March 2005.

[12] A. Kansal and M. Srivastava, "An environmental energy harvesting framework for sensor networks," *Proc. IEEE-ISLPED*, 2003.

[13] C. Alippi and C. Galperti, "An adaptive maximum power point tracker for maximising solar cell efficiency in wireless sensor nodes," *Proceedings of the IEEE International Symposium on Circuits and Systems, ISCAS 2006*, pp. 1–4, 2006.

[14] K. A. et al., "Models and solution techniques for frequency assignment problems," *Ann. Oper. Res.*, vol. 153, p. 79129, 2007.

[15] K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie, "Protocols for self-organization of a wireless sensor network," *Personal Communications, IEEE*, vol. 7, no. 5, pp. 16–27, 2000.

[16] XBow, "home page [online] http://www.xbow.com/." [Online]. Available: http://www.xbow.com/

[17] A. Goldsmith, “Wireless Communications,” 2005.

[18] Zigbee, “[on line].” [Online]. Available: http://www.zigbee.com/

[19] C. Perkins and E. Royer, “Ad-hoc on-demand distance vector routing,” in *Mobile Computing Systems and Applications, 1999. Proceedings.*, 25-26 Feb. 1999, pp. 90–100.

[20] B. Crow, I. Widjaja, L. Kim, and P. Sakai, “Ieee 802.11 wireless local area networks,” *Communications Magazine, IEEE*, vol. 35, no. 9, pp. 116–126, Sept. 1997.

[21] J. Haartsen, “The bluetooth radio system,” *Personal Communications, IEEE*, vol. 7, no. 1, pp. 28–36, Feb. 2000.

[22] E. Elnahrawy, X. Li, and R. Martin, “The limits of localization using signal strength: a comparative study,” in *Sensor and Ad Hoc Communications and Networks, 2004.*, 4-7 Oct. 2004, pp. 406–414.

[23] B. Parkinson and J. Spilker, *The global positioning system: theory and applications*, A. I. of Aeronautics and Astronautics, Eds., 1996, vol. Progress in astronautics and aeronautics.

[24] K. Sha and W. Shi., “Modeling the lifetime of wireless sensor networks.” *Sensor Letters*, vol. 3(2), p. 126135, June 2005.

[25] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocol for wireless microsensor networks,” in *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, Jan 4-7 2000, p. 10pp.vol.2.

[26] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, “Next century challenges: scalable coordination in sensor networks,” in *MobiCom ’99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile*

computing and networking. New York, NY, USA: ACM Press, 1999, pp. 263–270.

[27] M. Khan, B. Bhargava, and L. Lilien, "Self-configuring clusters, data aggregation, and authentication in microsensor networks," Department of Computer Science, Purdue University, Techical Report CSD TR 03-003, March 2003.

[28] O. Younis and S. Fahmy, "Heed: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *Mobile Computing, IEEE Transactions on*, vol. 3, no. 4, pp. 366–379, Oct.-Dec. 2004.

[29] A. Amis, R. Prakash, T. Vuong, and D. Huynh, "Max-min d-cluster formation in wireless ad hoc networks," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, 26-30 March 2000, pp. 32–41vol.1.

[30] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, pp. 393–422, 2002.

[31] MoteIV, "home page [online] http://www.moteiv.com/." [Online]. Available: http://www.moteiv.com/

[32] B. Conway, *Electrochemical Supercapacitors: Scientific Fundamentals and Technological Applications*. Kluwer Academic/Plenum Publishers, 1999.

[33] "http://artdeco.elet.polimi.it/."

[34] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tinydb: an acquisitional query processing system for sensor networks," *ACM Trans. Database Syst.*, vol. 30, no. 1, pp. 122–173, 2005.

[35] ——, "TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks," *Proceedings of the ACM Symposium on Operating System Design and Implementation (OSDI)*, 2002.

[36] K. Aberer, M. Hauswirth, and A. Salehi, "A middleware for fast and flexible sensor network deployment," *Proceedings of the 32nd international conference on Very large data bases*, pp. 1199–1202, 2006.

[37] ——, "The Global Sensor Networks middleware for efficient and flexible deployment and interconnection of sensor networks," *TR LSIR-REPORT-2006-006*, pp. 1–21, 2006.

[38] D. Chu, A. Tavakoli, L. Popa, and J. Hellerstein, "Entirely declarative sensor network systems," *Proc. VLDB'06*, pp. 1203–1206, 2006.

[39] D. Chu, L. Popa, A. Tavakoli, J. Hellerstein, P. Levis, S. Shenker, and I. Stoica, "The design and implementation of a declarative sensor network systems," *T.R. UCB/EECS-2006-132*, pp. 1–14, 2006.

[40] L. Baresi, D. Braga, M. Comuzzi, F. Pacifici, and P. Plebani, "A service-based infrastructure for advanced logistics," in *IW-SOSWE '07: 2nd international workshop on Service oriented software engineering*. New York, NY, USA: ACM Press, 2007, pp. 47–53.

[41] MaxStream, "[on line]." [Online]. Available: http://www.maxstream.com/

[42] I. S. of Rock Mechanics, "Suggested methods for monitoring rock movements using inclinometers and tiltmeters," *Rock Mechanics*, vol. 10, pp. pagg. 81–106.

[43] J. P. Lynch, "An overview of wireless structural health monitoring for civil structures," *Philosophical Transactions of the Royal Society of London: Series A, Mathematical and Physical Sciences, The Royal Society, London*, 2005.

[44] H. e. a. Li, "Recent applications of fiber optic sensors to health monitoring in civil engineering," *Engineering Structures*, vol. 26,11, pp. 1647–1657, 2004.

[45] L. G. Green, H. Maürer, T. Spillmann, B. Heincke, and H. Willenberg, "High-resolution geophysical techniques for improving hazard assessments of unstable rock slopes," *Swiss Federal Institute of Technology, , Zürich.*

[46] C. STMicroelectronics, "Lis3l02al mems inertial sensor: 3-axis - +/-2g ultra-compact linear accelerometer," vol. Datasheet number CD00068496.

[47] C. Alippi, C. Galperti, and M. Zanchetta, "Micro acoustic monitoring with mems accelerometers: Towards a wsn implementation," *Sensors, 6th IEEE Conference on*, 2007.

[48] Sayood, *Introduction to Data Compression*. New York: Morgan Kaufmann, 1996, ch. 13.

[49] M. Chen and M. L. Fowler, "Data compression trade-offs in sensor networks," *Proceedings of SPIE*, 2004.

[50] N. M. B Gold, *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*, N. U. John Wiley & Sons, Inc. New York, Ed., 1999.

[51] D. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, pp. 1098–1101, 1952.

[52] J. S. Vitter, "Design and analysis of dynamic huffman codes," *J. ACM*, vol. 34, no. 4, pp. 825–845, 1987.

[53] J. A. Storer and T. G. Szymanski, "Data compression via textual substitution," *J. ACM*, vol. 29, no. 4, pp. 928–951, 1982.