IMT Institute for Advanced Studies, Lucca

Lucca, Italy

Novel approaches to in-network processing for the reduction of energy consumption in wireless sensor networks

PhD Program in Computer Science and Engineering

XXI Cycle

Massimo Vecchio

2009

The dissertation of Massimo Vecchio is approved.

Program Coordinator: Prof. Ugo Montanari,
Computer Science Department, University of Pisa

Supervisor: Prof. Beatrice Lazzerini,
Department of Information Engineering, University of Pisa

Supervisor:
Prof. Francesco Marcelloni,
Department of Information Engineering, University of Pisa

Tutor: Prof. Beatrice Lazzerini,
Department of Information Engineering, University of Pisa

The dissertation of Massimo Vecchio has been reviewed by:

Prof. Silvia Giordano,
University of Applied Science - SUPSI - Switzerland

Dr. Aline Carneiro Viana ,
INRIA Saclay - Ile de France sud - France

IMT Institute for Advanced Studies, Lucca

2009

L'ovvio è quel che non si vede mai, finché qualcuno non lo esprime con la massima semplicità.
(K. Gibran)

Contents

List of Figures

List of Tables

Vita

January 23, 1979	Born, Cassino (FR), Italy
March 2005	Laurea Degree in Computer Science Engineering Final mark: 110/110 Magna cum Laude University of Pisa, Italy
During 2005	Collaborator at Department of Information Engineering University of Pisa, Italy
February 2006	PhD Admission at IMT Lucca, Italy
October 2008	5 months internship at INRIA Saclay - Ile de France sud, ASAP Research Group, France
Today	Research Engineer at INRIA Saclay - Ile de France sud, ASAP Research Group, France

Publications

1. M. Cococcioni, P. Ducange, B. Lazzerini, F. Marcelloni, and M. Vecchio, "Identification of Mamdani fuzzy systems based on a multi-objective genetic algorithm," *AI*IA '05: Workshop on Evolutionary Computation*, pp. 1–10, 2005.
2. G. Anastasi, S. Croce, M. Di Francesco, F. Marcelloni, E. Monaldi, and M. Vecchio, "Energy management in sensor networks for environmental monitoring," in *SIRWEC '06: XIII International Road Weather Conference*, 2006.
3. B. Lazzerini, F. Marcelloni, M. Vecchio, S. Croce, and E. Monaldi, "A fuzzy approach to data aggregation to reduce power consumption in wireless sensor networks," *NAFIPS '06: Proceedings of the Annual meeting of the North American Fuzzy Information Processing Society*, pp. 436–441, 2006.
4. S. Croce, F. Marcelloni, and M. Vecchio, "Reducing power consumption in wireless sensor networks using a novel approach to data aggregation," *The Computer Journal*, vol. 51(2), pp. 1227–239, 2008.
5. F. Marcelloni and M. Vecchio, "A simple algorithm for data compression in wireless sensor networks," *IEEE Communications Letters*, vol. 12(6), pp. 411–413, 2008.
6. F. Marcelloni and M. Vecchio, "An efficient entropy data compression algorithm for environmental monitoring wireless sensor networks," *The Computer Journal*, to appear, 2009.

Abstract

Wireless sensor networks (WSNs) are currently an active research area mainly due to the potential of their applications. However, the deployment of a large scale WSN still requires solutions to a number of technical challenges that stem primarily from the features of the sensor nodes such as limited computational power, reduced communication bandwidth and small storage capacity. Further, sensor nodes are typically powered by batteries with limited capacity which do not guarantee an attractive nodes' lifetime unless adequate power saving policies are undertaken.

Since the radio is the main cause of power consumption in a sensor node, most of the energy conservation schemes proposed in the literature have focused on minimizing the energy consumption of the communication unit. To achieve this objective, two main approaches have been followed: power saving through duty cycling and in-network processing. Duty cycling schemes define coordinated sleep/wakeup schedules among nodes in the network. On the other hand, in-network processing consists in reducing the amount of information to be transmitted by means of aggregation and/or compression techniques. This thesis focuses on the latter approach and proposes a novel distributed method to data aggregation and two algorithms to compress data locally on the sensor node.

The distributed data aggregation technique is based on fuzzy numbers and weighted average operators to reduce data communication in WSNs when we are interested in the estimation of an aggregated value such as maximum or minimum temperature measured in the network.

The first compression algorithm is a simple lossless entropy compression algorithm which can be implemented in a few lines of code, requires very low computational power, compresses data on the fly and uses a very small dictionary whose size is determined by the resolution of the analog-to-digital converter.

The second compression algorithm tackles the problem of noisy sampling by performing lossy compression on single node based on a differential pulse code modulation scheme with quantization of the differences between consecutive samples. Since different combinations of the quantization process parameters determine different trade-offs between compression performance and information loss, we exploit a multi-objective evolutionary algorithm to generate a set of combinations of these parameters corresponding to different optimal trade-offs. The user can therefore choose the combination with the most suitable trade-off for the specific application.

Chapter 1

Introduction

Moore's Law states that the number of transistors in an integrated circuit increases exponentially over time, doubling every 18 months. Since its introduction in 1965, Moore's Law has taken on a much broader scope and significance (Moo65). Almost all computing resources increase exponentially in availability over time, including disk capacity, memory, and network bandwidth. The increase in resources is not accompanied by a corresponding increase in size: a 200GB hard drive in 2005 is the same size as a 200MB hard drive in 1998.

On the one hand, Moore's law means that the resources which can fit in a form factor will grow over time. On the other hand, it also means that existing resources can be made smaller. In the first case, the exponential growth of resources has led existing classes of computing devices to be more powerful. In the second case, shrinking size has led to the appearance of new, smaller device classes. Minicomputers emerged in the 1970s. Desktops became feasible in the 1980s. The 1990s saw laptops become commonplace, and in the first few years of the 21st century, cellphones are ubiquitous.

A new class of computing devices is nowadays a reality, though it is in continuous evolution: embedded Wireless Sensor Networks (WSNs), collections of small devices with integrated computing, sensing, and networking (TM03; ASSC02). Their small size allows unobtrusively deploy-

ing them in a wide range of environments, collecting data at a fidelity and scale that was until now impossible.

The ability to embed sensors in large and uncontrolled environments opens up tremendous possibilities for a wide range of disciplines. Biologists at University of California, Berkeley (UCB) have deployed WSNs in redwood trees to gather real-time data so as to measuring how redwood microclimates vary over space and time (Red). Civil engineers have deployed networks on San Francisco's Golden Gate Bridge, to measure how winds and other meteorological conditions affect the bridge (SPC$^+$07). Fire departments are exploring how WSNs can help rescue trapped people or aid in evacuation (Fir).

Furthermore, WSNs have greater potential than simple data collection. When connected to an actuator, a sensor node can actively control its environment. Instead of a single thermostat per floor, homes can have WSNs that monitor and control home climate on a per room basis or smaller granularity. In precision agriculture, sensor nodes can measure soil moisture and control irrigation at the granularity of tens of square meters, rather than hectares (Cam). In fire rescue, a sensor network can detect danger and direct people along safer paths to exits (Fir).

These many possibilities have accompanying challenges. Moore's Law means that wireless sensor nodes can - and will - be tiny, a few millimeters on a side. Their energy sources, however, do not have the same governing principles. Chemical energy density limits storage technologies such as batteries or fuel cells, while ambient energy density limits renewable energy sources such as solar panels. A wireless sensor node's energy source determines its form factor: decrease node's energy requirements, and you can decrease its size.

As deploying sensor networks can be laborious and disruptive, the longer a network lasts, the better it is. This conflict between node form factor and network lifetime makes energy the defining resource of WSNs. It influences software, which must incorporate aggressive energy conservation policies. It also influences hardware, which must trade off increased resources against their accompanying energy costs. While improving technology can reduce the energy consumption of computation

or storage, networking has fundamental energy limits. Ultimately, wireless networking requires emitting energy into the environment. There is a trade-off between the energy costs of transmitting and receiving: a stronger transmit signal can require less processing effort on the receiver side, while additional signal processing on the receiver side can reduce power on the transmitter side.

1.1 Reducing energy consumption: a resume

Datasheets of commercial sensor nodes (Senc), together with several research papers in the field of WSNs (dCdS06; BA06; KWFLS07) show that data communication is very expensive in terms of energy consumption, whereas data processing consumes significantly less: the energy cost of receiving or transmitting a single bit of information is approximately the same as that required by the processing unit for executing a thousand operations . On the other hand, the energy consumption of the sensing unit depends on the specific sensor type. In several cases, however, it is negligible with respect to the energy consumed by the communication unit and sometimes also by the processing unit. Thus, to extend the lifetime of a WSN, most of the energy conservation schemes proposed in the literature aim to minimize the energy consumption of the communication unit. To achieve this objective, two main approaches have been followed: power saving through duty cycling and in-network processing.

Duty cycling schemes define coordinated sleep/wakeup schedules among nodes in the network. A detailed description of these techniques applied to WSNs can be found in (ACDP07).

On the other hand, in-network processing consists in reducing the amount of information to be transmitted by means of aggregation and/or compression techniques.

In particular, aggregation techniques are aimed at reducing data as they flow toward the sink and can be roughly classified into two categories: structure-based techniques, in which data aggregation is performed based on a defined structural organization of the network and structure-free techniques, in which data aggregation is performed with-

out explicit maintenance of a structure.

On the other hand compression techniques reduce the data size by exploiting the structure of the data. Data compression algorithms fall into two broad classes: lossless and lossy algorithms. Lossless algorithms guarantee the integrity of data during the compression/decompression process. On the contrary, lossy algorithms may generate a loss of information, but generally ensure a higher compression ratio.

1.2 Outline and contributions

This thesis is organized in 7 chapters.

Chapter 2 presents WSNs, three key application scenarios and an overview of the requirements for WSNs. It is intended to provide the background necessary for a general understanding of the issues discussed in the remaining chapters.

Chapter 3 provides a survey of works related to this thesis which have focused on in-network processing in WSNs, as well as other activities aimed at reducing their power consumption.

Chapter 4 discusses the critical design issues that must be addressed to build a distributed aggregation framework in WSNs. It underscores the shortcomings of traditional structure-based aggregation techniques and presents an ad-hoc structure-free approach that addresses these issues. The proposed scheme is based on a novel distributed approach relying on fuzzy numbers and weighted average operators. Moreover, it shows how the lifetime of the network can be estimated through the datasheet of the sensor node and the number of received and transmitted messages. Finally, it discusses and evaluates the application of our approach in both simulated and real testbed scenarios.

Chapter 5 presents a simple and effective lossless entropy compression algorithm acting on single node, able to respond to the criticality of some application domains demanding high accuracy for sensor measures. The effectiveness of the proposed algorithm is proven by compressing four environmental datasets collected by real WSNs, in terms of both compression ratios and complexity. Moreover, it shows that the

proposed algorithm can be successfully applied when dealing with non-smooth signals, which theoretically should not favour its performances.

Chapter 6 proposes an approach to perform lossy compression on single node based on a differential pulse code modulation scheme with quantization of the differences between consecutive samples. A multi-objective evolutionary algorithm is exploited in order to let the user choose the most suitable trade-off between reconstruction error and compression ratio for the specific application. Following the previous chapter's wave, our lossy compression approach is tested on three datasets collected by real WSNs, showing that, though very simple, our approach can achieve significant compression ratios despite negligible reconstruction errors.

Chapter 7 summarizes the thesis and concludes with some open issues.

Chapter 2

WSNs: Application, characteristics and metrics

The concept of WSNs is based on a simple equation:
Sensing + Processing + Communications = Thousands of potential applications. As soon as people understand the capabilities of a WSN, hundreds of applications spring to mind. It seems like a straightforward combination of modern technology (Hil03).

However, combining sensors, radios, and CPU's into an effective WSN requires a detailed understanding of both capabilities and limitations of each of the underlying hardware components, as well as a detailed understanding of modern networking technologies and distributed systems theory. Indeed, the limited resources available in a sensor node do not allow using the large amount of in-network algorithms proposed in the last years for completely different applications and different machines (desktop computers, laptops, PDAs, cellular phones), but demand the development of specifically designed/adapted solutions. For example, since sensor nodes are typically equipped with a few kilobytes of memory and a 4-8MHz microprocessor, embedding classical data compression schemes in these tiny nodes is practically unfeasible (KL05; SM06; BA06). Indeed, the proposed algorithms in the last years can achieve very high compression ratios despite non negligible memory occupation and com-

putational effort requirements. For this reason, to make the WSN vision a reality, existent algorithms should be adapted if we want to synthesize the envisioned applications out of the underlying hardware capabilities.

To develop such ad-hoc algorithms, we have to start from understanding the set of target applications, trying to group them in three macrogroups. Then, we give some general characteristics of WSNs and the main differences between them and Mobile Ad-Hoc Networks (MANETs). Finally we briefly review the main system and individual node evaluation metrics to be considered when dealing with WSNs.

2.1 Sensor network application classes

The three application classes we have selected are: environmental data collection, surveillance applications, and sensor node tracking. We believe that the majority of WSN deployments will fall into one of these class templates.

2.1.1 Environmental Data Collection

A typical environmental data collection application is one where a research scientist wants to collect several sensor readings from a set of points in an environment over a period of time in order to detect trends and interdependencies. This scientist would like to collect data from hundreds of points spread throughout the area and then analyze the data offline (MFC$^+$07; MCP$^+$02). Further, the scientist would be interested in collecting data over several months or years in order to look for long-term and seasonal trends. For the data to be meaningful they would have to be collected at regular intervals and the nodes to remain at known locations.

At the network level, the environmental data collection application is characterized by having a large number of nodes continually sensing and transmitting data to a set of base stations that store the data using traditional methods. These networks generally require very low data rates and extremely long lifetimes. In a typical usage scenario, the nodes are distributed over an outdoor environment in a way that the distance be-

tween adjacent nodes is rather short, though the distance across the entire network could be significant. After deployment, the nodes should first discover the topology of the network and then estimate advantageous routing strategies, for example capturing link connectivity statistics so as to achieve reliability (WTC03).

The routing strategy can then be used to route data to a central collection point. In environmental monitoring applications, it is not essential that the nodes develop the routing strategies on their own. Instead, it may be possible to calculate the advantageous routing topology outside of the network and then communicate the necessary information to the nodes as required.

Environmental data collection applications typically use tree-based routing topologies where each routing tree is rooted at high-capability nodes that sink data. Data is periodically transmitted from child node to parent node up the tree-structure until it reaches the sink. With tree-based data collection each node is responsible for forwarding the data of all its descendants. Nodes with a large number of descendants transmit significantly more data than leaf nodes. These nodes can quickly become energy bottlenecks (XHE01; CE02).

Once the network is configured, each node periodically samples its sensors and transmits its data up the routing tree and back to the base station. For many scenarios, the interval between these transmissions can be on the order of minutes. Typical reporting periods are expected to be between 1 and 15 minutes; while it is possible for networks to have significantly higher reporting rates. The typical environment parameters being monitored, such as temperature, light intensity, and humidity, do not change quickly enough to require higher reporting rates. In addition to large sample intervals, environmental monitoring applications do not have strict latency requirements. Data samples can be delayed inside the network for moderate periods of time without significantly affecting application performance. In general the data is collected for future analysis, not for real-time operation.

In order to meet lifetime requirements, each communication event must be precisely scheduled. The sensor nodes will remain dormant most

part of the time; they will only wake up to transmit or receive data. If the precise schedule is not met, the communication events will fail. As the network ages, it is expected that nodes will fail over time. Periodically the network will have to reconfigure itself to handle node/link failure or to redistribute network load. Additionally, as the scientists learn more about the environment they study, they may want to go in and insert additional sensing points. In both cases, the reconfigurations are relatively infrequent and will not represent a significant amount of the overall system energy usage.

The most important characteristics of the environmental monitoring requirements are long lifetime, precise synchronization, low data rates and relatively static topologies. Additionally it is not essential that the data is transmitted in real-time back to the central collection point. The data transmissions can be delayed inside the network as necessary in order to improve network efficiency.

2.1.2 Surveillance applications

The second class of sensor network application is surveillance. Surveillance networks are composed of nodes that are placed at fixed locations throughout an environment and continually monitor one or more sensors to detect an anomaly (NKC09; LHF08) . A key difference between surveillance monitoring and environmental monitoring is that surveillance networks are not actually collecting any data. This has a significant impact on the optimal network architecture. Each node has to frequently check the status of its sensors but it only has to transmit a data report when there is a security violation. The immediate and reliable communication of alarm messages is the primary system requirement. These are "report by exception" networks.

Additionally, it is essential that it is confirmed that each node is still present and functioning. If a node was disabled or failed, it would represent a security violation that should be reported. For surveillance applications, the network must be configured so that nodes are responsible for confirming the status of each other. One approach is to have each node

assigned to peer that will report if a node is not functioning. The optimal topology of a surveillance network will look quite different from that of a data collection network.

In a collection tree, each node must transmit the data of all of its descendants. Thus, a short and wide tree would result to be optimal. On the contrary, with a surveillance network the optimal configuration would be to have a linear topology that forms a Hamiltonian cycle of the network. The power consumption of each node is only proportional to the number of children it has. In a linear network, each node would have only one child. This would evenly distribute the energy consumption of the network.

The majority of the energy consumption in a surveillance network is consumed i) to meet the strict latency requirements associated with signaling the alarm when a security violation occurs, and ii) to maintain the neighboring nodes ready to instantly forward alarm announcements. Indeed, actual data transmission will consume a small fraction of the network energy. Once detected, in fact, a security violation must be communicated to the base station immediately. The latency of the data communication across the network to the base station has a critical impact on application performance. Users demand that alarm situations be reported within seconds of detection. This means that network nodes must be able to respond quickly to requests from their neighbors to forward data.

In these networks, reducing the latency of an alarm transmission is significantly more important than reducing the energy cost of the transmissions. This is because alarm events are expected to be rare. In a fire surveillance system, alarms would almost never be signaled. In the event that one does occur a significant amount of energy could be dedicated to the transmission. Reducing the transmission latency leads to higher energy consumption because routing nodes must monitor the radio channel more frequently.

2.1.3 Node tracking scenarios

A third usage scenario commonly discussed for sensor networks is the tracking of a tagged object through a region of space monitored by a sensor network (TCC07). There are many situations where one would like to track the location of valuable assets or personnel. Current inventory control systems attempt to track objects by recording the last checkpoint that an object passed through. However, with these systems it is not possible to determine the current location of an object. For example, UPS tracks every shipment by scanning it with a barcode whenever it passes through a routing center. The system breaks down when objects do not flow from checkpoint to checkpoint. In typical work environments it is impractical to expect objects to be continually passed through checkpoints.

With WSNs, objects can be tracked by simply tagging them with a small sensor node. The sensor node will be tracked as it moves through a field of sensor nodes that are deployed in the environment at known locations. Instead of sensing environmental data, these nodes will be deployed to sense the RF messages of the nodes attached to various objects. The nodes can be used as active tags that announce the presence of a device. A database can be used to record the location of tracked objects relative to the set of nodes at known locations. With this system, it becomes possible to ask where an object is currently, not simply where it was last scanned.

Unlike sensing or surveillance networks, node tracking applications will continually have topology changes as nodes move through the network. While the connectivity between the nodes at fixed locations will remain relatively stable, the connectivity to mobile nodes will be continually changing. Additionally the set of nodes being tracked will continually change as objects enter and leave the system. It is essential that the network be able to efficiently detect the presence of new nodes that enter the network.

2.1.4 Hybrid networks

In general, complete application scenarios contain aspects of all three categories. For example, in a network designed to track vehicles that pass through it, the network may switch between an alarm monitoring network and a data collection network (MSW07). During the long periods of inactivity when no vehicle is present, the network will simply perform an alarm monitoring function. Each node will monitor its sensors waiting to detect a vehicle. Once an alarm event is detected, all or part of the network, will switch into a data collection network and periodically report sensor readings up to a base station that track the vehicles progress.

2.2 Characteristics of WSNs: A Summary

WSNs are similar to MANETs as both of them are wireless networks and involved in multi-hops wireless communications. However WSNs are very different from traditional data networks including MANETs (Gad06). The characteristics of WSNs are summarized in this section, together with some comparisons with MANETs.

- WSNs are application-driven networks.

  Different WSNs have different task-specific requirements. It is quite different to traditional general-purpose networks. This means that the protocols for WSNs can be totally new. It is not required to tailor the protocol design for WSNs in order to achieve compatibility to existing protocols of traditional general-purpose networks. For example, unlike in MANETs, it is not required to build up an IP address mechanism over WSNs. Also, the design of protocol stack is not confined to traditional layering.

- WSNs are self-organized networks.

  They are usually with large scale and high node density. The number of sensor nodes in a network might be several hundred or even reach over a thousand. Such a large number of sensor nodes are

usually left unattended after they are deployed. This means that a WSN should function in a completely autonomous manner.

- WSNs suffer from limited energy supply.

  Protocols for WSNs should be energy-efficient. Sensor node is battery powered. Recharging a sensor node is expensive, if not impossible, which means that it is impractical to revive a sensor node after its battery energy is drained. Energy efficiency must be a main consideration of protocol design for WSNs.

- WSNs are instable.

  First, low-cost implementation of a sensor node makes it easy to fail. Also, due to limited energy resource, the power of a sensor node is easily drained, which results in permanent disfunction of the sensor node. Second, WSNs are usually working in bad or even hostile environments. The wireless links between sensor nodes are fragile and subject to failures. These two situations make a given wireless communication path between two sensor nodes instable and even easy to be permanently damaged. Therefore, protocols for WSNs should adapt to such network instability.

- The mobility feature of WSNs is different from MANETs.

  In MANETs, nodes are laptop computers and/or PDAs. Nodes are therefore mobile. In WSNs, nodes are sensor devices. Usually they are not mobile after deployment. But the locations of the interesting phenomena might be mobile, e.g., when the network is tracking a motor vehicle. Also, the sink of a WSN may be mobile. A sink may be a hand-held device such as a PDA or a laptop computer. It may be carried to the network area to collect the sensor data hourly, and each time its location may be different.

- The network traffic feature of WSNs is different from MANETs.

  In MANETs, network traffic is like the traditional wired networks. It might usually be unicasting. Every node may require to communicate with all the others ones: traffic is usually in a peer-to-peer

manner. Broadcasting and multicasting traffic are also possible in MANETs. However, in WSNs, usually network traffic is in a many-to-one manner. Many nodes send data packets to a single node, i.e., the sink.

- A global-identity-based addressing mechanism which is required by traditional data networks including MANETs may be unnecessary for WSNs.

  The application of WSNs is mainly on collecting data of some phenomena of interest. Depending on specific task requirement, it may care where a particular phenomenon takes place or whether a particular phenomenon happens, rather than which particular node is currently reporting data.

- In-network data processing may be required in WSNs.

  WSNs are deployed to collect data of some phenomena of interest. Usually, there would be many sensor nodes that collect the data of a particular phenomenon and report the data to the sink. Naturally, these data packets reported by each source sensor nodes could have some redundancy. This redundancy can be further exploited by in-network data processing approaches such as data aggregation or data fusion.

- Finally, WSNs suffer stronger from computational constraints than MANETs.

  A sensor node has by far lower memory capacity than a node of a MANET, and its computational speed is much slower due to its low-cost design, small size and lower battery capacity. One cannot expect to save large volume of data in a sensor node or program it to execute complex algorithms.

  These constraints are the main reasons why many network protocols for traditional data networks including MANETs are not suitable for WSNs. We have to employ another family of network protocols for data communication and network organization in WSNs which take into account the above network characteristics.

2.3 Evaluation Metrics

This chapter ends exploring, more in detail, some key network evaluation metrics. To do this we keep in mind the high-level objectives of the network deployment, the intended usage of the network, and the key advantages of WSNs over existing technologies.

One result is that many of these evaluation metrics are interrelated. Often it may be necessary to decrease performance in one metric, such as sample rate, in order to increase another, such as lifetime. Taken together, this set of metrics form a multidimensional space that can be used to describe the capabilities of a WSN. The capabilities of a platform are represented by a volume in this multidimensional space that contains all of the valid operating points. In turn, a specific application deployment is represented by a single point. A system platform can successfully perform the application if and only if the application requirements point lies inside the capability hyperspace.

One goal of this chapter is to present an understanding of the tradeoffs that link each axis of this space and an understanding of current capabilities.

2.3.1 Lifetime

Critical to any WSN deployment is the expected lifetime. The goal of both the environmental monitoring and surveillance application scenarios is to have nodes placed out in the field, unattended, for months or years (YQ05; NKL+07).

The primary limiting factor for the lifetime of a sensor network is the energy supply. Each node must be designed to manage its local supply of energy in order to maximize total network lifetime. In many deployments it is not the average node lifetime that is important, but rather the minimum node lifetime. In the case of wireless surveillance systems, every node must last for multiple years. A single node failure would create a vulnerability in the surveillance systems.

In some situations it may be possible to exploit external power, perhaps by tapping into building power with some or all nodes. However,

one of the major benefits to wireless systems is the ease of installation. Requiring power to be supplied externally to all nodes considerably reduces this advantage. A compromise is to have a handful of special nodes that are wired into the building's power infrastructure.

In most application scenarios, a majority of the nodes will have to be self-powered. They will either have to contain enough stored energy to last for years, or they will have to be able to scavenge energy from the environment through devices, such as solar cells or piezoelectric generators (KKPG98; RWR03). Both of these options demand that the average energy consumption of the nodes be as low as possible.

The most significant factor in determining lifetime of a given energy supply is radio power consumption. In a wireless sensor node the radio consumes a vast majority of the system energy. This power consumption can be reduced through decreasing the transmission output power, through decreasing the radio duty cycle or through in-network processing. The first two options involve sacrificing other system metrics, while the third one could not affect any other system metric.

2.3.2 Coverage

Next to lifetime, coverage is the primary evaluation metric for a wireless network (CTLW05; HT05). It is always advantageous to have the ability to deploy a network over a larger physical area. This can significantly increase the system's value to the end user. It is important to keep in mind that the coverage of the network is not equal to the range of the wireless communication links being used. Multi-hop communication techniques can extend the coverage of the network well beyond the range of the radio technology alone. In theory they have the ability to extend network range indefinitely. However, for a given transmission range, multihop networking protocols increase the power consumption of the nodes, which may decrease the network lifetime. Additionally, they require a minimal node density, which may increase the deployment cost.

Scalability is a key component of the WSN value proposition. A user can deploy a small trial network at first and then can continually add

sense points to collect more and different information. A user must be confident that the network technology being used is capable of scaling to meet his eventual need. Increasing the number of nodes in the system will impact either the lifetime or the effective sample rate. More sensing points will cause more data to be transmitted which will increase the power consumption of the network. This can be offset by sampling less often.

2.3.3 Cost and ease of deployment

A key advantage of WSNs is their ease of deployment (TM03). Biologists and construction workers, for instance, installing networks cannot be expected to understand the underlying networking and communication mechanisms at work inside the wireless network. For system deployments to be successful, the WSN must auto-configure itself. An unskilled person should place the nodes throughout the environment and the WSN should automatically work.

Ideally, the system should automatically configure itself for any possible physical node placement. However, real systems must place constraints on actual node placements - it is not possible to have nodes with infinite range. The WSN must be capable of providing feedback as to when these constraints are violated. The network should be able to assess quality of the network deployment and indicate any potential problem. This translates to requiring that each device be capable of performing link discovery and determining link quality. In addition to an initial configuration phase, the system must also adapt itself to changing environmental conditions. Throughout the lifetime of a deployment, nodes may be relocated or large physical objects may be placed so that they interfere with the communication between two nodes. The network should be able to automatically reconfigure on demand in order to tolerate these occurrences.

The initial deployment and configuration is only the first step in the network life-cycle. In the long term, the percentage of the maintenance cost will be higher than the percentage of initial deployment cost with re-

spect to the total cost of ownership. The surveillance application scenario in particular requires that the system be extremely robust. In addition to extensive hardware and software testing prior to deployment, the sensor system must be constructed so that it is capable of performing continuous self-maintenance. When necessary, it should also be able to generate requests when external maintenance is required.

In a real deployment, a fraction of the total energy budget must be dedicated to system maintenance and verification. The generation of diagnostic and reconfiguration traffic reduces the network lifetime. It can also decrease the effective sample rate.

2.3.4 Response Time

Particularly in alarm application scenarios, system response time is a critical performance metric (LHF08). An alarm must be signaled immediately when an intrusion is detected. Despite low power operation, nodes must be capable of having immediate, high-priority messages communicated across the network as quickly as possible. While these events will be infrequent, they may occur at any time without notice. Response time is also critical when environmental monitoring is used to control factory machines and equipment. Many users envision WSNs as useful tools for industrial process control. These systems would only be practical if response time guarantees could be met.

The ability to have low response time conflicts with many of the techniques used to increase network lifetime. Network lifetime can be increased by having nodes only operate their radios for brief periods of time. If a node only turns on its radio once per minute to transmit and receive data, it would be impossible to meet the application requirements for response time of a surveillance system.

Response time can be improved by including nodes that are powered all the time. These nodes can listen for the alarm messages and forward them down a routing backbone when necessary. This, however, reduces the ease of deployment for the system.

2.3.5 Temporal Accuracy

In environmental and tracking applications, samples from multiple nodes must be cross-correlated in time in order to determine the nature of phenomenon being measured. The necessary accuracy of this correlation mechanism will depend on the rate of propagation of the phenomenon being measured. In the case of determining the average temperature of a building, samples must only be correlated to within seconds (CMV08). However, to determine how a building reacts to a seismic event, millisecond accuracy is required.

To achieve temporal accuracy, a network must be capable of constructing and maintaining a global time base that can be used to chronologically order samples and events. In a distributed system, energy must be consumed to maintain this distributed clock. Time synchronization information must be continually communicated between nodes. The frequency of the synchronization messages is dependent on the desired accuracy of the time clock.

2.3.6 Security

Despite the seemingly harmless nature of simple temperature and light information from an environmental monitoring application, keeping this information secure can be extremely important. Significant patterns of building use and activity can be easily extracted from a trace of temperature and light activity in an office building. In the wrong hands, this information can be exploited to plan a strategic or physical attack on a company. WSNs must be capable of keeping the information they are collecting private from eavesdropping (MWS08).

As we consider security oriented applications, data security becomes even more significant. Not only must the system maintain privacy, it must also be able to authenticate data communication. It should not be possible to introduce a false alarm message or to replay an old alarm message as a current one. A combination of privacy and authentication is required to address the needs of all three scenarios. Additionally, it should not be possible to prevent proper operation by interfering with transmitted

signals.

Use of encryption and cryptographic authentication costs both power and network bandwidth (PST+02; Riv94). Extra computation must be performed to encrypt and decrypt data and extra authentication bits must be transmitted with each packet. This impacts application performance by decreasing the number of samples than can be extracted from a given network and the expected network lifetime.

2.3.7 Effective Sample Rate

In a data collection network, effective sample rate is a primary application performance metric. We define the effective sample rate as the sample rate that sensor data can be taken at each individual sensor and communicated to a collection point in a data collection network. Fortunately, environmental data collection applications typically only demand sampling rates of 1-2 samples per minute. However, in addition to the sample rate of a single sensor, we must also consider the impact of the multi-hop networking architectures on the node's capability to effectively relay the data of surrounding nodes (MFHH02).

In a data collection tree, a node must handle the data of all of its descendants. If each child transmits a single sensor reading and a node has a total of 60 descendants, then it will be forced to transmit 60 times as much data. Additionally, it must be capable of receiving those 60 readings in a single sample period. This multiplicative increase in data communication has a significant effect on system requirements. Network bit rates combined with maximum network size end up impacting the effective per-node sample rate of the complete system.

One mechanism for increasing the effective sample rate beyond the raw communication capabilities of the network is to exploit in-network processing. Various forms of spatial and temporal compression can be used to reduce the communication bandwidth required while maintaining the same effective sampling rate. Additionally local storage can be used to collect and store data at a high sample rate for short periods of time. In-network data processing can be used to determine when an "in-

teresting" event has occurred and automatically trigger data storage. The data can then be downloaded over the multi-hop network as bandwidth allows.

Triggering is the simplest form of in-network processing. It is commonly used in surveillance systems. Effectively, each individual sensor is sampled continuously, processed, and only when a security breach has occurred data is transmitted to the base station. If there were no local computation, a continuous stream of redundant sensor readings would have to be transmitted.

Chapter 3

Reducing energy consumption in WSNs: State of the Art

In WSNs, the communication cost is often several orders of magnitude higher than the computational cost. Pottie and Kaiser (PK00) reported that the energy consumption for executing 3000 instructions is equivalent to sending a bit in a range of 100 meters by radio. Therefore data aggregation, data compression (from now on we will refer to them as *in-network processing*) and duty cycling schemes are very important to extend the network lifetime. Duty cycling schemes define coordinated sleep/wakeup schedules among nodes in the network. On the other hand, in-network processing consists in reducing the number of information to be transmitted by means of compression or aggregation.

3.1 Duty cycling schemes

With current hardware technology, the most of the energy is spent by the radio transceiver. Besides the energy spent during transmission and reception of messages, also the energy consumption due to listening on the channel waiting for incoming packets is relevant (idle listening). To limit

the energy consumption due to idle listening, the most effective solution is to turn on and off the radio repetitively (duty cycling). This can be done efficiently at the MAC level. Possible approaches are represented by *schedule-based* and *contention-based* protocols. In schedule-based protocols (vHH04; KA06; LL04; vL03; LKR04) there is a common schedule that reserves a period for transmission to each node. Thus, the receiver knows when the radio must be turned on to receive possible incoming data. In contention-based protocols (PHC04; ACMV06; BKK07) the node periodically turns on the radio and checks for activity on the channel. If the channel is found busy, the radio is kept in receive mode and data is received, otherwise the node goes back into the sleep state. Both approaches have pros and cons: schedule-based protocols are more complex to be implemented and may suffer high latency, but at the same time they provide some guarantees to nodes; contention-based protocols are simpler, mainly because of the lack of a shared scheduling policy, and more tolerant to the appearance and disappearance of nodes (e.g. due to mobility), but they may lead to poor performance under high traffic loads (ACMV06). Moreover, in some sensor network applications such as surveillance, fire detection, and object-tracking system, networks are idle most of the time, but a sudden event causes a large amount of packets to yield network congestion. In such circumstances, MAC protocols which have a fixed duty cycle suffer from data loss because they are inappropriate to the heavy traffic load. MAC protocols with a small fixed duty cycle can save energy with no events, but packets can be dropped due to queue overflow whenever an event occurs. To reduce the packet drop, the fixed duty cycle should be increased, but energy is largely wasted for frequent listening whenever no events occur. Hence, it is desirable to increase duty cycle to prevent packet drop under heavy traffic, and to decrease duty cycle for nodes that are unaffected by the traffic or are under the light traffic to save energy (BKK07). For this reason several MAC protocols with adaptive duty cycle have been studied: B-MAC (PHC04) and its improvements (ACMV06; BKK07) are but examples. For a complete review of recent advances in this direction the reader can refer to (ACDP07).

One of the most known and used MAC protocol is the B-MAC (Berkeley MAC) protocol, which has been developed at UCB and now is shipped with the TinyOS operating system (HSW+00). B-MAC uses an asynchronous sleep/wake scheme: nodes wake up periodically to check the channel for activity using a low power listening (LPL). If the channel is not active, they come back to sleep; otherwise, they maintain the radio on to receive the packet. In B-MAC, packets are composed of a long preamble and a payload. The preamble duration has to be longer than the check interval so as to allow a node to detect ongoing transmissions during its check interval. This approach avoids explicit synchronization mechanisms despite a longer packet. Of course, the B-MAC protocol allows saving energy only if the number of packets transmitted by a node is not too high. Thus, in order to achieve a conspicuous power consumption reduction, duty cycling schemes should be accompanied by in-network processing techniques.

3.2 Data Aggregation

In-network aggregation is a well known technique to achieve energy efficiency when propagating data from information sources (e.g., sensors) to sink(s). The main idea behind in-network aggregation is that, rather than sending individual data items from sensors to sinks, multiple data items are aggregated as they are forwarded by the sensor network. Data aggregation is application-dependent, i.e., depending on the target application, the appropriate data aggregation operator, or *aggregator*, will be employed. For example, suppose that in a controlled temperature environment, the average temperature needs to be monitored. As sensors generate temperature readings periodically, internal nodes in the data collection tree (rooted at the information sink and spanning relevant data sources) average data received from downstream nodes and forward the result toward the information sink. The effect is that, by transmitting a lower number of data units, considerable energy savings can be achieved. However, the quantity of saved energy depends on the type of aggregator employed. For instance, in the running average scenario just depicted, a

number of packets containing temperature readings from individual sensors are averaged and result in a single packet of the same size as the ones that carry individual temperature readings. However, if the only possible aggregator is concatenation, i.e., multiple data items are concatenated and transmitted as a single packet, then the sole source of energy savings is a more efficient medium access.

From the information sink's point of view, the benefits of in-network aggregation are that in general i) it yields more manageable data streams avoiding overwhelming sources with massive amounts of information, and ii) performs some filtering and pre-processing on the data, making the task of further processing the data less time and resource consuming.

Because of its well-known power efficiency properties, in-network aggregation has been the focus of several research efforts on sensor networks over the last years. As a result, several data aggregation algorithms targeting different sensor network scenarios have been proposed. Directed diffusion (IGE$^+$03), TAG (MFHH02), eScan (ZGE02), and Sensor Protocols for Information via Negotiation (SPIN) (HKB99) represent milestones in this direction.

Monitoring (including monitoring of continuous environmental conditions like temperature, humidity, seismic activity, etc.) is a good example of such applications. One of the constraints imposed by periodic data generation on aggregation algorithms is timing. In other words, how long should a node wait for receiving data from its children before forwarding data it has already received? Note that the trade-off is between data accuracy and freshness, i.e., the longer a node waits, the more readings it is likely to receive and therefore, the more accurate the information it sends out. On the other hand, waiting too long may result in stale data. Furthermore, if a node waits too long, it may interfere with the next "data wave".

We successfully tackled the issue of enabling aggregation in WSNs developing a novel distributed framework based on fuzzy numbers and weighted average operators to reduce data communication when interested in the estimation of an aggregated value such as maximum or minimum temperature measured in the network. The basic point of our ap-

proach is that each node maintains an estimate of the aggregated value. Based on this estimate, the node decides whether a new value measured by the sensor on board the node or received through a message has to be propagated along the network. We can note that the approach is completely distributed: there is no need to create and maintain a hierarchical structure of the network.

To the best of our knowledge, only a few papers have discussed the topic of structure-free data aggregation in WSNs (KWFLS07; BGS03). Indeed, there are two main challenges in performing structure-free data aggregation. First, as there is no pre-constructed structure, routing decisions for efficient aggregation of packets need to be made on-the-fly. Second, as nodes do not explicitly know their upstream nodes, they can not explicitly wait for data from any particular node before forwarding their own data.

WSNs are typically used to monitor some parameters of a physical process. By its nature, a WSN can only sample these processes and therefore generate an approximation of the parameters under observation.
Thus, all the algorithms in WSN have an inherent extra dimension: *accuracy*. This dimension can be exploited to propose energy-efficient algorithms. Indeed, the less accuracy is required, the less network activity is necessary and the more energy can be saved. Starting from this observation, Boulis et al. have proposed a distributed estimation algorithm that explores the energy/accuracy subspace for the periodic aggregation domain (BGS03). The basic idea is that each node keeps and possibly transmits an estimate (local estimate) of the global aggregate. In the case of max or min aggregation functions, the estimate is represented by the pair of scalars (value, confidence), where value and confidence correspond to, respectively, the mean and the variance of the estimated max or min. Each node accepts estimates from all its neighbors and thus can decide dynamically whether its information is useful to other nodes in determining the global aggregate and therefore whether this information should be broadcast. On the contrary, in periodic aggregation implemented by snapshot aggregation, a node cannot change the rate at which it sends predictions to its parents (MFHH05).

Authors in (KWFLS07) observed that packets need to be aggregated early on their route to the sink for efficiency. Based on this observation they proposed and modeled a MAC layer protocol for spatial convergence called Data-Aware Anycast (DAA). Moreover, they observed that if some nodes wait for other nodes to send data, it can lead to efficient aggregation. With this observations in mind they studied the impact of Randomized Waiting (RW) for improved data aggregation.

In contrast, the on-fashion trend is to enable a structured aggregation to route packets toward the sink. Structured aggregation can be categorized into two families: cluster-based and tree-based (KWFLS07).

3.2.1 Cluster-Based Approaches

In (HCB00), the authors propose the LEACH protocol to cluster sensor nodes and let cluster-heads aggregate data and communicate with the base-station directly using high transmission power. The cluster-heads are randomly elected in each round to distribute energy consumption among all nodes. LEACHC (CSH02) uses the base-station to broadcast cluster-head assignment to further spreading out the cluster-heads throughout the network. Based on LEACH, (ZHL04) refines the cluster-head election algorithm that does not require the participation of the base-station and scatters cluster-heads more evenly across the network. However it requires every node to broadcast at its highest transmission power at the setup stage of each round, which limits its ability to conserve energy.

Lindsey et al. propose PEGASIS (LR02) which organizes all nodes in a chain and lets them play the role of heads in turn to conserve more energy. Since there is only one head node and there are not simultaneous transmissions, latency is an issue in PEGASIS. To address this, two chain-based PEGASIS enhancements are proposed in (LRS01) and (LRS02). In (LRS01) the authors propose a binary hierarchical approach for CDMA-capable sensor nodes, and in (LRS02) they propose a chain-based three level approach that allows simultaneous transmission for non- CDMA-capable sensor nodes. Based on LEACH and PEGASIS, Culpepper et al. propose Hybrid Indirect Transmission (HIT) (CDM04). HIT still uses

LEACH-like clusters, but allows multi-hop routes between cluster-heads and non-head nodes.

LEACH and PEGASIS based protocols assume that the base-station can be reached by any node in one hop, which limits the size of the network for which such protocols are applicable. In addition, in scenarios where the data can not be perfectly aggregated, cluster-based protocols do not necessarily have significant advantage since the cluster-head has to send many packets to the base station using high transmission power.

3.2.2 Tree-Based Approaches

In (IEGH02) the authors propose Greedy Incremental Tree (GIT) to establish an energy-efficient path based on Directed Diffusion (IGE$^+$03). Krishnamachari et al. (KEW02) compare three data-centric routing schemes, Center at Nearest Source (CNS), Shortest Path Tree (SPT) and a variation of (GIT) which establishes the route between the sink and the nearest source to illustrate the advantage of data aggregation. They observe that GIT guarantees the lowest average number of transmissions. In (MFHH02; MRFC02) Madden et al. study the data aggregation issue in implementing a real system and propose the Tiny AGgregation Service (TAG) framework. TAG uses shortest path tree and proposes improvements like snooping-based and hypothesis testing based optimizations, dynamic parent switching, and use of child cache to estimate lost data.

TAG lets parents notify their children about the waiting time to gather all data from children before transmitting, and the sleeping schedule can be adjusted accordingly. Ding et al. use the shortest path tree with parent energy-awareness in (DCX03), where the neighboring node with the shortest distance to the sink among those with high residual energy is chosen as the parent. All the above tree-based data aggregation routing protocols need a lot of message exchanges to construct and maintain the tree. Zhang and Cao propose Dynamic Convoy Tree-Based Collaboration (DCTC) in (ZC04a). In (ZC04b), they further optimize the tree reconfiguration schemes. Essentially, DCTC tries to balance the tree in the monitoring region to reduce the energy consumption. But it assumes the knowl-

edge of the distance to the center of the event at sensor nodes, which may not be feasible to compute with the sensed information in all tracking applications. In addition, DCTC involves heavy message exchanges which are not desired when the data rate is high. Also, the performance of DCTC highly depends on the accuracy of mobility prediction algorithms.

Another class of tree-based data-centric routing protocols considers sensing information entropy in the routing metric (SS02; Sca03; CV03; CBLV04; PKG08). However joint source coding or even its approximation is hard to deploy since the real distribution of collected data is hard to predict; moreover, we will introduce the issues of single node entropy compression in the next subsection. In (CV03; CBLV04), the authors study explicit communication considering joint entropies among nodes into the routing metric, and propose approximation algorithms such as Leaves Deletion approximation and Balanced SPT/TSP tree. But these algorithms are centralized. They assume the global knowledge of the information entropy of each sensor node and the joint entropy of each pair, which makes such approaches non-trivial to implement in practice. Pattern et al. study the impact of spatial correlation on routing for some special cases in (PKG08) and derive the optimal cluster size for these cases. Although authors use cluster structure, the basic tree-based routing is maintained instead of transmitting packets to the base-station in one hop.

3.3 Data Compression

Compression techniques reduce the data size by exploiting the structure of the data, so they can be a valuable tool when we are interested in limiting transmission/reception of data in WSNs. On the other hand, the limited resources available in a sensor node demand, as already highlighted, the development of specifically designed compression algorithms.

Data compression algorithms fall into two broad classes: lossless and lossy algorithms. Lossless algorithms guarantee the integrity of data during the compression/decompression process. On the contrary, lossy algorithms may generate a loss of information, but generally ensure a higher compression ratio.

Obviously, the data compression approach (both lossless and lossy) can be a valuable help in power saving only if the execution of compression algorithms does not require an amount of energy larger than the one saved in reducing transmission. Indeed, after analyzing several families of classic compression algorithms, Barr and Asanović conclude that compression prior to transmission in wireless battery-powered devices may actually cause an overall increase of power consumption, if no energy awareness is introduced (BA06). On the other hand, standard compression algorithms are aimed at saving storage and not energy. Thus, appropriate strategies have to be adopted.

We successfully tackled both the aspects of deploying a lossless and a lossy data compression algorithms.

3.3.1 Lossless Data Compression

The choice of the algorithm type depends on the specific application domain. In WSNs, data are collected by sensors which, due to noise, produce different readings even when they are sampling an unchanging phenomenon. For this reason, sensor manufactures specify not only the sensor operating range but also the sensor accuracy. Datasheets express accuracy by providing a margin of error, but typically do not include a probability distribution for this error. Thus, when a value is measured by a sensor, we are confident that the actual value is within the error margin, but cannot know with what probability that value is some distance from the real value. In this context, lossy compression algorithms may convolute the original error distribution when that distribution is not uniform (SGO$^+$04). On the other hand, the criticality of some application domains demands sensors with high accuracy and cannot tolerate that measures are corrupted by the compression process. In Body Area Networks, for instance, sensor nodes permanently monitor and log vital signs: each small variation of these signs have to be captured because it might provide crucial information to make a diagnosis. Thus, we believe that lossless compression algorithms suitable to WSNs have to be deeply investigated. Since sensor nodes are typically equipped with a few kilobytes of mem-

ory and a 4-8MHz microprocessor, embedding classical data compression schemes in these tiny nodes is practically impossible (KL05; SM06; BA06). Indeed, lossless data compression algorithms proposed in the last years can achieve very high compression ratios despite non negligible memory occupation and computational effort requirements.

Nevertheless, some examples of existing lossless algorithms adapted to sensor nodes exist in the literature and they refer typically to dictionary-based compression. The most famous dictionary-based lossless compression algorithm is the Lempel-Ziv-Welch (LZW) algorithm. Actually, LZW is the result of some modifications made in 1984 by Terry Welch (Wel84) to already existent algorithms developed in 1977 and 1978 by Abraham Lempel and Jacob Ziv, called respectively LZ77 (ZL77) and LZ78 (parts of LZ78 were patent protected in the United States). LZW is surprisingly simple. In a nutshell, it replaces strings of characters with single codes. Since codes are generally smaller than strings, compression can be achieved by replacing strings with codes in the original data. LZW performs no analysis of the incoming text: for each new string, just a new code is created. Codes are of any arbitrary length. When using eight bit-coding for characters, the first 256 codes are by default assigned to the standard character set. The other codes are generated when they are needed. Since both the compressor and the decompressor have the initial dictionary and all new dictionary entries are created based on existing dictionary entries, the receiver can recreate the dictionary on the fly as data are received. An analysis of the original versions of LZ77, LZ78 and LZW (SM06; BA06) highlights that they are far away from being embeddable in sensor nodes, since their processing and memory requirements are greater than the ones available in commercial sensor nodes. Thus, "embedded" versions of these algorithms have been proposed. In (SM06) and (LZO) the authors introduce S-LZW and miniLZO which are purposely adapted versions of LZW and LZ77, respectively. Since S-LZW outperforms miniLZO, as shown in (SM06), we describe only S-LZW in detail and use it as comparison for our algorithm in Chapter 5.

S-LZW splits the uncompressed input bitstream into fixed size blocks and then compresses separately each block. During the block compres-

sion, for each new string, that is, a string which is not already in the dictionary, a new entry is added to the dictionary. For each new block, the dictionary used in the compression is re-initialized by using the 256 codes which represent the standard character set. Due to the poor storage resources of sensor nodes, the size of the dictionary has to be limited. Thus, since each new string in the input bitstream produces a new entry in the dictionary, the dictionary might become full. If this occurs, an appropriate strategy has to be adopted. For instance, the dictionary can be frozen and used as-is to compress the remainder of the data in the block (in the worst case, by using the code of each character), or it can be reset and started from scratch.

Nevertheless, the basic S-LZW algorithm does not take specific advantage of sensor data characteristics. Indeed, sensor data are generally repetitive. Even data that change drastically over time tend to adhere to this pattern since sampling intervals are taken very short so as to model the drastic changes in the data. Thus, to take this repetitive behaviour into account, a mini-cache is added to S-LZW: the mini-cache is a hash-indexed dictionary of size N, where N is a power of 2, that stores recently used and created dictionary entries. Further, the repetitive behaviour can be used to pre-process the raw data so as to build appropriately structured datasets, which can perform better with the compression algorithm. In (SM06), the authors show that the use of structured datasets and the introduction of the mini-cache increase the compression ratios without introducing appreciable computational overhead. It follows that S-LZW has to balance four major inter-related parameters: the block size, the dictionary size, the strategy to follow when the dictionary is full and the mini-cache size. The values of these four parameters deeply influence compression performances, so they have to be fine tuned before deploying S-LZW in sensor nodes.

3.3.2 Lossy Data Compression

When the application does not have to meet high-accuracy measurements requirements cheap sensors should be used. Such sensors, due to noise,

produce different readings even when they are sampling an unchanging phenomenon. Noise increases the entropy of the signal and therefore hinders the lossless compression algorithm to achieve considerable compression ratios. The ideal solution would be to adopt on the sensor node a lossy compression algorithm in which the loss of information would be just the noise. Thus, we could achieve high compression ratios without losing relevant information. To enable lossy compression in WSNs two approaches have been followed in the literature:

- to distribute the computational cost on the overall network (JJRZQG06; GBT+04; GEH03; GDV06; Cia06; TR04; WBD+06; CPOK06; PKR02; XLC04; RM07);
- to enable compression acting at single node independently of the others (SGO+04).

The first approach is natural in cooperative and dense WSNs. Here, nodes can collaborate with each other so as to carry out tasks they could not execute alone. Thanks to the particular topology of these WSNs, data measured by neighboring nodes are correlated both in space and in time.

The simplest distributed approach in WSNs relies on fixing a model, either stochastic or deterministic, of the phenomenon under monitoring and estimating the parameters of that model. An overview of stochastic methods can be found in (JJRZQG06). Here, the phenomenon is modeled as a single scalar or a vector. Nodes quantize their noisy measurements, a central data sink collects the measurements and estimates the scalar or the vector. While the technique is able to accommodate a broad class of noise models, the constant signal model is very restrictive and unable to represent arbitrarily smooth phenomena. A deterministic technique, which allows managing more complicated signal models, has been proposed in (GBT+04). Here, first a set of basis functions is used to locally fit the phenomenon in various regions of the network. Then, the resulting approximations are tied together by kernel regression, implemented using inter-sensor message exchange. Unfortunately, the reliability of the reconstruction depends on the appropriate choice of the basis functions. Since this choice is performed statically and the set of functions does not vary

dynamically, phenomena of arbitrary complexity cannot be adequately modeled. For example, using a set of globally smooth functions, piecewise smooth phenomena separated by a jump discontinuity cannot be reconstructed. Indeed, the basis representation will smooth out the discontinuity, arguably the most interesting feature of the signal (WDB06).

Distributed transform coding bridges this gap by allowing the data itself to guide the representation rather than forcing it to fit a set of prefixed models. This approach is based on theoretical principles of transform analysis performed on the source bits in order to compact the signal energy so as to achieve a better signal-to-noise ratio (SNR). Various kinds of transforms such as Discrete Wavelet Transform (DWT) and Karhunen-Loève Transform (KLT) (Goy01) have been proposed for different types of applications. We recall that transforms are invertible functions which merely change the representation of the signal without altering the information contained in it. The motivating principle of such transform is that a more effective simple coding can be achieved in the transform domain than in the original signal space.

Distributed transform coding, however, requires inter-node communication: nodes need data from neighbors to compute the transform coefficients. Extra power is therefore consumed to transmit data to neighbors and to receive data from neighbors. Further, since multiple nodes may want to communicate to each other at the same time, a channel access method for shared medium has to be adopted (TDMA scheduling or FDMA scheduling, for example). Due to the popularity of dense sensor networks, several works have been proposed for distributed transform (GDV06; GEH03; Cia06; TR04; WBD$^+$06; CPOK06).

Finally, to overcome the problem of inter-node communication, Distributed Source Coding (DSC) (PKR02; XLC04), also known as Slepian-Wolf coding (in the lossless case) (SW73) and Wyner-Ziv coding (in the lossy case) (WZ76), has been proposed. Here, the assumption is that sensor nodes are densely deployed. Thus, the readings of one sensor are highly correlated with those of its neighbors. DSC refers to the compression of multiple correlated sensor outputs that do not communicate with each other. These sensors send their compressed outputs to a cen-

tral point, e.g., the base station, for joint decoding the encoded streams. DSC is the on-fashion trend in distributed data compression and a large number of papers focus on this scheme: for the sake of brevity we invite interested readers to refer to (RM07). We observe that DSC cannot be used without proper synchronization among the nodes of the WSN, i.e., assumptions have to be made for the routing and scheduling algorithms and their connection to the DSC scheme (RM07).

To the best of our knowledge, only one paper has discussed the second approach: the Lightweight Temporal Compression (LTC) algorithm (SGO$^+$04). It is an efficient and simple lossy compression technique for the context of habitat monitoring. LTC introduces a small amount of error into each reading bounded by a control knob: the larger the bound on this error, the greater the saving by compression. Basically LTC is similar to Run Length Encoding (RLE) in the sense that it attempts to represent a long sequence of similar data with a single symbol (Sal07). The difference with RLE is that while the latter searches for strings of a repeated symbol, LTC search for linear trends.

Actually, to compress data locally and independently of the other nodes has a non-negligible advantage with respect to the distributed transform approach: nodes do not need to communicate with each other. Thus, nodes save energy and do not compete for the shared medium. On the other hand, in an attempt to decorrelate the information, each sensor can use only its local information, thus vanishing the promising results obtained by DSC. This can be problematic whether either the compression ratios achieved by the compression algorithm executed independently on the sensor node are too low (the amount of compressed data is still large), or, though the compression ratios are high, these have been obtained by executing a high number of instructions, with consequent large power consumption.

Chapter 4

A Distributed Fuzzy Aggregation Approach for WSNs

In order to develop our distributed aggregation framework we were inspired by some good characteristics of the algorithm proposed by Boulis et al. (BGS03). Here, whenever a node gets an estimate from a neighboring node or acquires a local measurement from the sensor, it aggregates the estimate or the local measurement with its local estimate. The first aggregation (global fusion) is performed by using the covariance intersection algorithm (CAM02). As regards the second aggregation (local fusion), local measurements are represented by a pair of scalars (value, confidence), where value and confidence are, respectively, the measured value and the variance of the measurement noise. The aggregation is computed as a weighted sum where weights depend on the reciprocal positions of the measured value and of the value of the local estimate, and on the two corresponding confidences. To decide whether the results of the global and local fusions have to be broadcast, each node consults a local table, where the most recently received estimates from the neighborhood are maintained: if the difference between the new generated estimate and at least one of the estimates in the table are higher than

a pre-fixed threshold, then the new estimate is broadcast. In contrast to tree-based approaches that obtain the aggregates at a single (or a few) sink node, the algorithm maintains the aggregate at each node and therefore each node can interpret the role of sink. Further, the algorithm takes advantage of the broadcast nature of wireless transmission: all nodes within the radio range can hear and receive the signal of a wireless transmission. Finally, there is no need to create and maintain a hierarchical structure of the network. In our approach we exploited these good features, but adopting different techniques in managing measurement error, aggregation, decision and routing.

We will show how network lifetime can be computed. Then we will discuss and evaluate the application of the algorithm to the monitoring of the maximum temperature in a 100-node simulated WSN and a 12-node real WSN. For the real WSN, we will use Tmote Sky motes from Sentilla as nodes (Senc). We will discuss how our distributed algorithm promptly reacts to changes of temperature, maintaining the total number of messages exchanged among the nodes very low. We will show that, in the worst case, the WSN lifetime is approximately 418 days with a battery capacity of 2500mAh. This result is extremely interesting. Indeed, we have to consider that applications which do not switch off the radio when no message is sent or received, and do not try to reduce the number of messages on the network, are characterized by a lifetime of a few days. Finally, we will show that our algorithm performs favorably with respect to the approach proposed in (BGS03).

4.1 Our Approach

4.1.1 Overview

The basic idea of our approach is that each node maintains an estimate E of the global aggregated function. Thus, when the node receives a new value from a neighbor or measures a new value from the sensor, it can decide whether the new value has to be forwarded or not. To take the measurement error into account, we represent the value measured by the

sensor as a symmetric triangular fuzzy number $V = (v, vs)$, where v is the central value and vs is the spread of the triangle. The spread is proportional to the measurement error. It follows that estimate E is a fuzzy number in turn and is represented as $E = (e, es)$. For the sake of simplicity, we chose to adopt triangular fuzzy numbers. Anyway, our approach can be easily adapted to other types of membership functions.

Figure 1 shows the modules deployed on each generic node n_k. The module Fuzzifier is activated by each new value v_k measured by the sensor on board the node. The output of the module is the fuzzification of v_k as the symmetric triangular fuzzy number $V_k = (v_k, vs_k)$. The module Message Analyzer is activated whenever node n_k receives a message from a neighboring node, say n_i. The Message Analyzer extracts the information from the message and decides whether this information has to be sent either to the Table of Estimates, or to the Aggregation and Decision modules. The information is the estimate E_i of the global aggregated function maintained in node n_i. The Table of Estimates has an entry i for each neighboring node n_i: the entry contains the last value received from node n_i. The Aggregation module aggregates either V_k or E_i with E_k. The output of the module is a new estimate nE_k, which replaces the old estimate E_k and is input to the Decision module. Based on the Table of Estimates, the Decision module decides whether nE_k is relevant and therefore has to be sent to the neighbors or not. In the case of maximum (minimum) estimation, if the new estimate is larger (lower) than at least one entry in the table and the degree of uncertainty of this entry is higher than the one of the new estimate, then the new estimate is sent in broadcast; otherwise no communication occurs. If the variable we are monitoring varies slowly, the number of exchanged messages is very low. In the following, we analyze the single modules in detail.

4.1.2 The Aggregation Module

The Aggregation Module compares either $E_i(e_i, es_i)$ or $V_k(v_k, vs_k)$ to $E_k(e_k, es_k)$ and produces a new estimate $nE_k(ne_k, nes_k)$. In the following, we denote $E_i(e_i, es_i)$ and $V_k(v_k, vs_k)$ as $I(i, is)$. The Aggregation

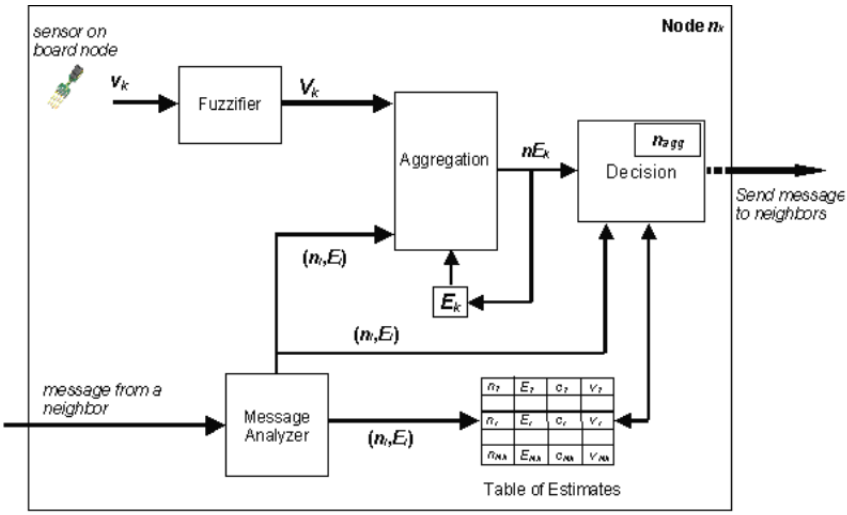

Figure 1: Block diagram of the application deployed on each node

Module behaves differently with respect to the following different cases:

1. $I \cap E_k = 0$. The new value nE_k depends on the type of aggregate we are monitoring in the network:
   (a) if the aggregate is the minimum, then $nE_k = min(I, E_k)$;
   (b) if the aggregate is the maximum, then $nE_k = max(I, E_k)$;
2. $I \cap E_k \neq 0$. The new value nE_k is computed as follows:
   $ne_k = e_k \cdot w_{E_i} + i \cdot w_I$
   $nes_k = es_k \cdot w_{E_i} + is \cdot w_I = \frac{2 \cdot es_k \cdot is}{es_k + is}$,

where $w_{E_i} = \frac{is}{es_k + is}$ and $w_i = \frac{es_k}{es_k + is}$. Actually, nes_k is the harmonic mean of es_k and is.

The aggregation is therefore performed only if the two input estimates are comparable; otherwise the new value of the estimate depends on the type of aggregate we are monitoring. The aggregation is implemented as a weighted sum, where weights are inversely proportional to the spread of the two fuzzy numbers. The rationale for this choice is that the less

uncertain fuzzy number has to be more relevant in the aggregation computation.

4.1.3 The Table of Estimates

The Table of Estimates is shown in Figure 2. Each entry of the table collects information about a neighboring node n_i. The first column contains the identifier of the neighboring node. The second column stores the last estimate E_i which node n_k has received from node n_i. The third column contains a timeout counter c_i which is increased whenever node n_k sends a message. The counter is set to 0 whenever a new estimate E_i is received from node n_i. The fourth column stores bit b_i which indicates if the entry is valid or not. The bit is initialized to 1. When counter c_i exceeds a value α fixed by the user, bit b_i is set to 0 (invalidation). The value of the counter can exceed α when:

- node n_i is dead (for instance, because battery is exhausted);
- node n_i cannot (temporally or definitively) communicate directly with node n_k. The reason can be, for instance, the movement of node n_i which brings n_k out of the radio range of n_i.

The invalidation of an entry of the table is very important for allowing, as explained in the next section, the Decision module to decide whether the new estimate has to be sent or not.

The Table of Estimates is generated dynamically: at each message received from a neighboring node n_i, if identifier n_i is not in the table, a new entry is created. In the new entry, the Message Analyzer module copies the identifier n_i and the estimate E_i. Further, it sets counter c_i to 0 and validity v_i to 1.

4.1.4 The Decision Module

The role of the Decision module is to decide whether the new value nE_k computed by the Aggregation module has to be sent or not. To take this decision, the Decision module uses the Table of Estimates. The behavior

Node	Estimate	Counter	Validity
n_1	(e_1, es_1)	c_1	b_1
...	...	...	...
n_i	(e_i, es_i)	c_i	b_i
...	...	...	...
n_{Nk}	(e_{Nk}, es_{Nk})	c_{Nk}	b_{Nk}

Figure 2: Table of Estimates

of the Decision module depends on the type of aggregate, minimum or maximum, we are interested in. In the following, we consider the two cases separately.

- Minimum. The message is sent if, in the Table of Estimates, $\exists j : (e_j - ne_k) > \sigma_1$ and $(nes_k - es_j) \leq \sigma_2$;

- Maximum. The message is sent if, in the Table of Estimates, $\exists j : (ne_k - e_j) > \sigma_1$ and $(nes_k - es_j) \leq \sigma_2$,

where σ_1 and σ_2 are two real positive parameters chosen by the user, and σ_2 is smaller than or equal to the measurement error.

In other words, the message is sent only if the Decision module, based on the Table of Estimates, considers that there exists some neighboring node n_j with an estimate that is not updated with respect to new estimate nE_k and if the spread of nE_k is smaller than es_j, or, in the case of the spread of nE_k is larger than es_j, the difference between the two spreads is at the most equal to the measurement error. In other words, we forward nE_k only if its degree of uncertainty is comparable to or smaller than the one of es_j. This approach avoids circulating within the network estimates characterised by a high degree of uncertainty. The proper choice of σ_1 is very important because σ_1 determines the level of accuracy on the estimate of the maximum or of the minimum. The lower σ_1 is, the higher the level of accuracy is. On the other hand, the higher the level of accuracy is, the higher the probability of sending messages is. Thus, the value of has to be fixed by determining the right trade-off between accuracy and energy consumption. Typically, this trade-off depends on the application.

If the value nE_k has been computed as a consequence of the reception of a message sent by node n_i, then the fuzzy value E_i contained in the message is used to update the corresponding entry in the Table of Estimates. Note that the table is updated after taking the decision on sending the message. This allows to send a message even if all the entries, except for n_i, are updated. The message is important because it serves as acknowledgement for node n_i.

The algorithm used by the Decision module causes the following problem. Let us assume that node n_k has N_k neighboring nodes and nE_k results to be the most updated among all the values in the Table of Estimates. Then, the module decides to send a message to update the neighbors. When one of the neighbors replies, the received message triggers the activation of the Aggregation module with the consequent generation of a new estimate nE_k, which is analyzed by the Decision module. Since the entries in the Table of Estimates are not updated, the Decision module decides to send another message with the same information as the previous sent message. This behavior would produce at least N sent messages. The problem is avoided by the Message Analyzer module. Before describing this module, let us examine the structure of a generic message sent by node n_k. Figure 3 shows this structure. The first field is the identifier of the sender (actually, this field would not be necessary because the sender is inserted automatically into the message at MAC level). The second and third fields contain the central value and the spread of new estimate nE_k. The values of the fields id_reply_k and id_agg_k depend on the event that has activated the Aggregation module and therefore produced the new estimate nE_k.

1. If the activation of the Aggregation module has been caused by the reception of a message from a neighboring node, say n_i, then the values depend on the type of aggregate, minimum or maximum, we are interested in:

   - Minimum. If $E_i < E_k$, then $id\_reply_k = n_i$, $id\_agg_k = id\_agg_i$ and $n_{agg} = id\_agg_i$; otherwise, $id\_reply_k = n_k$ and $id\_agg_k = n_{agg}$; we consider $E_i > E_k$ if $e_i > e_k$.

- Maximum. If $E_i > E_k$, then $id\_reply_k = ni$, $id\_agg_k = id\_agg_i$ and $n_{agg} = id\_agg_i$; otherwise, $id\_reply_k = n_k$ and $id\_agg_k = n_{agg}$; we consider $E_i < E_k$ if $e_i < e_k$.

2. If the activation of the Aggregation module has been caused by measuring a new value v_k from the sensor on board the node, then $id\_reply_k = -1$ by default and $id\_agg_k = n_{agg} = n_k$.

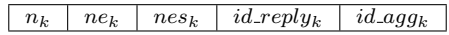

Figure 3: Structure of a generic message

The role of field id_reply_k will be clear in the next section. Field id_agg_k allows propagating the identifier of the node which has measured the maximum (or minimum) value in the network. Since each node stores the current value of id_agg_k, each node knows where maximum (or minimum) value has been measured. This allows satisfying our requirement that each node can interpret the role of sink and, therefore, can provide both the maximum (minimum) value and the identifier of the node where this value has been measured. The knowledge of the node identifier can be very important in, for instance, alarm detection applications, because it allows localizing the exact area where the dangerous event occurred, thus speeding up the problem solution.

4.1.5 The Message Analyzer Module

The Message Analyzer Module analyzes the message received by the node and decides whether the information contained in the message has to be forwarded to the Aggregation and Decision modules, or simply stored in the Table of Estimates. Obviously, if the Aggregation and Decision modules do not receive the information as input, they are not activated and no transmission occurs. The decision is based on the analysis of the field id_reply. Let us assume that the message has been received from node n_i.

- If $id\_reply_i = n_k$, then the module updates the Table of Estimates (we recall that the first field of the message contains the identifier of the sender) without forwarding the value E_i to the Aggregation module; in this case, node n_k is receiving a message which is an acknowledgment to a previous message sent by node n_k itself. Thus, no further estimate, which is not already known by the neighboring nodes, has to be propagated.

- If $id\_reply_i \neq n_k$, then the module sends the value E_i to the Aggregation module; in this case, the message is not a reply to a message sent by node n_k, but rather the communication that a new value is being propagated through the network.

4.1.6 Handling Estimate Staleness

As described in Section 4.1.2, estimate E_k in a generic node n_k is updated when value V_k measured by the sensor on board n_k or estimate E_i received from a neighboring node n_i intersect E_k or, otherwise, have the central values larger (lower) than their central value of E_k if the estimated aggregate is the maximum (minimum). If V_k and E_i do not intersect E_k and the central values are lower (larger) than the central value of E_k, then E_k is not updated and becomes stale with the passing of time. This occurs, for instance, when we adopt a WSN for estimating the trend of the maximum temperature during a day (0-24). The temperature will increase during the first hours of the day until to achieve the highest value in the early afternoon. When the temperature will start to decrease, the value of E_k will not be updated anymore, making this value staler and staler with the passing of time and, therefore, more and more unreliable with respect to the actual maximum value currently measured in the network. To model staleness, we increase the uncertainty of fuzzy number E_k with the passing of time. In the case of maximum (minimum) estimation, if the fuzzification V_k of the value v_k measured by the sensor does not intersect E_k, and V_k is lower (higher) than E_k, then the central value e_k of E_k is preserved, but the spread es_k is increased by a value δ fixed by the user. In this way, the spread of the symmetric triangular fuzzy number

is increased at prefixed instants. By increasing the spread, we also enhance the probability that E_k intersects V_k (or E_i), thus performing the aggregation and certainly generating a value nE_k lower (higher) than E_k. This allows following the possible decreasing (increasing) of the maximum (minimum) in the network. To the aim of allowing the transmission of the new estimate, we extend the Decision module. Besides the messages sent by using the algorithm described in Section 4.1.4, the Decision module also will send a new estimate nE_k if:

- in the case of Minimum estimation, $\exists j : (ne_k - e_j) > \chi$ in the Table of Estimates,
- in the case of Maximum estimation, $\exists j : (e_j - ne_k) > \chi$ in the Table of Estimates.

Typically, values δ and χ are chosen so as to follow the decreasing (increasing) of the maximum (minimum) temperature more slowly than the increasing (decreasing). We observe that δ is the unique parameter affecting the fuzzy numbers which the user can control. Indeed, the spread of the triangular membership functions is fixed by the measurement error which depends on the type of sensors. The choice of δ is application-dependent: the largest the value of δ, the faster the adapting of the network to decreasing (increasing) variations of the maximum (minimum).

4.2 Estimation of the node lifetime

In this section, we show how the WSN lifetime can be estimated. Our approach relies on the B-MAC protocol with low power listening (LPL) at the data link layer. Thus, we can adopt the model proposed by Polastre et al. (PHC04). This model requires the knowledge of some values of current consumption, which can be extracted from the datasheet of the Tmote Sky. We have rearranged some equations to fit them to our application.

Table 1 shows times and consumptions for completing primitive operations of a Tmote Sky.

Table 1: Time and Current Consumption in Tmote Sky

Operation		Time (ms)		I (mA)
Initialize radio (b)	t_{INIT}	0.47	c_{INIT}	14
Turn on radio (c)	t_{ON}	1.42	c_{ON}	1
Switch to RX/TX (d)	t_{SW}	0.212	c_{SW}	14
Time to sample radio (e)	t_{SR}	0.288	c_{SR}	21
Evaluate radio sample (f)	t_{EV}	0.197	c_{EV}	14
Receive 1 byte	t_{RXB}	0.032	c_{RXB}	19.7
Transmit 1 byte	t_{TXB}	0.032	c_{TXB}	17.4

The total energy E consumed by a node can be expressed as:

$$E = E_S + E_{RX} + E_{TX} + E_{LISTEN} + E_{SLEEP} \tag{4.1}$$

where E_S, E_{RX}, E_{TX}, E_{LISTEN} and E_{SLEEP} are the amount of energy consumed for sampling sensor, receiving and transmitting messages, listening the radio channel and sleeping. In the following, we consider the computation of E for a time interval $T = 24$ *hours* $= 86400$ *seconds*.

Energy E_S is calculated as

$$V \cdot c_{DATA} \cdot t_D \tag{4.2}$$

where $V = 3$ *Volt* is the battery voltage, $c_{DATA} = 2.95$ *mA* is the current consumption to sample the temperature sensor and t_D is the total time spent to sample the sensor during 24 hours test. If we assume that the sensor is sampled every 15 seconds, $t_D = \frac{T}{15} \cdot t_{DATA} = \frac{86400}{15} \cdot t_{DATA} = 5760 \cdot t_{DATA} = 1152$ *s*, where $t_{DATA} = 200$ *ms* is the time needed for each sampling. It follows that $E_S = 10195.2$ *mJ*.

Energy E_{RX} is expressed as:

$$E_{RX} = V \cdot c_{RXB} \cdot t_{RX} \tag{4.3}$$

where $t_{RX} = n_{RMSG} \cdot (L_{PACKET} + L_{PREAMBLE}) \cdot t_{RXB}$ is the total time spent to receive n_{RMSG} messages arrived during T, $L_{PACKET} + L_{PREAMBLE}$ is the length of the message (length of the packet plus length of the preamble) expressed in bytes and t_{RXB} is the time spent to receive one byte. Since we adopt the send and receive primitives of TinyOS,

$L_{PACKET} = 38$ *bytes*. The value of $L_{PREAMBLE}$ depends on the B-MAC protocol. The B-MAC protocol, as explained in Section 3.1, periodically samples the channel for activity. To avoid losing messages, the sample period t_I has to be smaller than the duration of the preamble (PHC04). Thus, $L_{PREAMBLE} \geq \left\lceil \frac{t_I}{t_{RXB}} \right\rceil$.

The sampling of the channel adopts a low power listening (LPL) approach. Figure 4 shows the sequence of operations required to perform a LPL using the CC2420 radio, which is the radio on board the Tmote Sky nodes. Though the current profiles are slightly different from those of the CC1000 radio used in (PHC04), we assume the same power consumption $E_{LPL} = 17.3$ *J* for each single LPL.

During interval T, the number n_{LISTEN} of checks for channel activity is $n_{LISTEN} = \left\lceil \frac{T}{t_I} \right\rceil$. It follows that energy $E_{LISTEN} \leq E_{LPL} \cdot n_{LISTEN}$.

B-MAC protocol provides 8 different modes corresponding to 10, 20, 50, 100, 200, 400, 800 and 1600 *ms* for the check interval t_I. We chose $t_I = 100$ *ms* in our experiments. Thus, $E_{LISTEN} \leq 14947.2$ *mJ* and $L_{PREAMBLE} \leq 3126$.

Similar to E_{RX}, energy E_{TX} is computed as

$$E_{TX} = V \cdot c_{TXB} \cdot t_{TX} \tag{4.4}$$

where $t_{TX} = n_{SMSG} \cdot (L_{PREAMBLE} + L_{PACKET}) \cdot t_{TXB}$ is the total time spent to transmit n_{SMSG} messages during T.

When the node is not busy in sampling sensor, receiving and transmitting messages, listening the radio channel, it can enter the sleep mode. The time spent sleeping is $t_{SLEEP} = T - t_D - t_{RX} - t_{TX} - t_{LISTEN}$. We can deduce from Figure 4 that the time t_{LISTEN} spent for performing one LPL of the channel is

$$t_{LISTEN} = t_{INIT} + t_{ON} + t_{SW} + t_{SR} + t_{EV} = 2.587ms. \tag{4.5}$$

Energy $E_{SLEEP} = V \cdot c_{SLEEP} \cdot t_{SLEEP}$, with $c_{SLEEP} = 0.021$ *mA* (from the Tmote Sky datasheet). Lifetime t_l of the node can be computed as:

$$t_l = \frac{C_{BATT} \cdot V}{E} \tag{4.6}$$

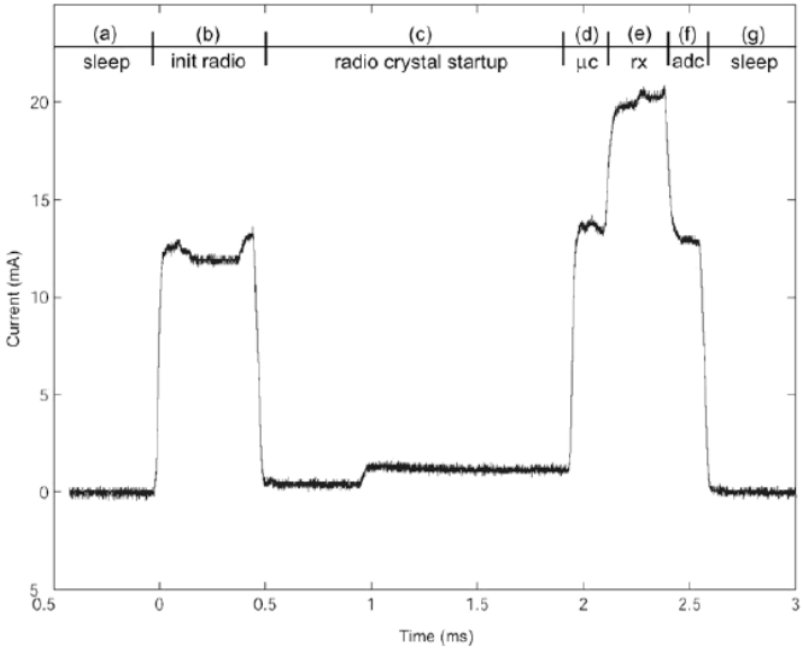

Figure 4: A trace of the power consumption while sampling the channel on a Tmote Sky

where C_{BATT} is the battery capacity. To compute total energy E, and therefore the network lifetime, we need to know the numbers n_{RMSG} and n_{SMSG} of, respectively, received and sent messages. Both these numbers are application-dependent. In the next sections, we will show two examples (one simulated and the other real) of WSNs and will compute an estimate of the lifetime for both. In the examples, we set the sensor sampling period to *15s*. To take the measurement error into consideration we fixed the spread of the triangular fuzzy number to 0.3. The two parameters σ_1 and σ_2 used in the Decision Module to decide whether an estimate has to be propagated were set to 1^oC and 0.3^oC, respectively. The threshold α used to invalidate an entry in the Table of Estimates was fixed to 5. The step δ used to increase the spread of the estimate of the global ag-

gregate was fixed to 0.01. The parameter χ used in the Decision Module for deciding whether an estimate has to be propagated during handling estimate staleness was set to 1^oC.

4.3 Power Consumption: a Simulation

As first example, we simulated a 100-node WSN on TOSSIM. TOSSIM is a discrete event simulator for TinyOS sensor networks (Tos). The topology of the network is shown in Figure 5. Each node is equipped with a temperature sensor.

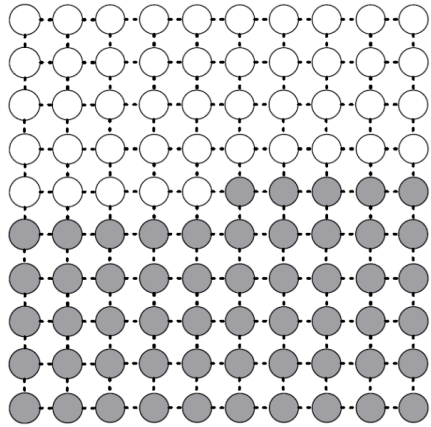

Figure 5: Sensor network topology

We aim to measure the maximum temperature in the network. Monitoring maximum temperature can be useful, for instance, to detect possible fires. In the simulation, we used the temperature profile measured by a weather station, located in Siena (Italy), during a 24 hours interval between 14 and 15 May 2006. This profile is shown in Figure 6. To make the simulation as close as possible to the reality, we considered that a number of sensor nodes were placed in a shadowy zone, whereas the others were placed in a sunny zone. Thus, we associated the original temperature profile with the sensor nodes in the sunny zone (white circles in

Figure 5) and the original temperature profile decreased by 2^oC with the sensor nodes in the shadowy zone. Obviously, a scenario with two different temperature profiles increases the probability of message exchange, especially in the shadowy zone which has to be reached by the maximum temperature estimate. Thus, this scenario penalizes our approach with respect to a scenario with all the sensor nodes with the same temperature profiles. Actually, the latter can be considered as the best scenario for our approach.

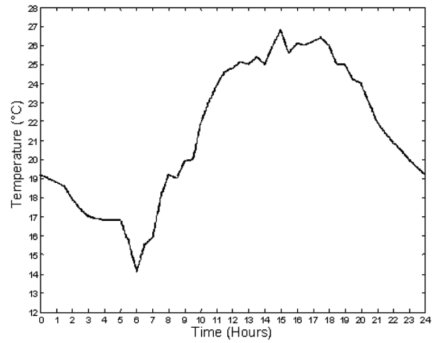

Figure 6: Temperature profile used in the simulations

We executed two different simulations, using respectively the fixed and empirical radio models implemented in TOSSIM. In the fixed radio model, a message sent by a node, say node A, is received only by the four nodes closest to A with probability 1. In the empirical radio model, a message sent by node A is received by its neighboring nodes with different probabilities. Figure 7 shows the fixed and empirical radio models adopted in the simulations. Here, the numbers close to the nodes represent the probabilities of the nodes to receive a message sent by node A. Table 2 shows the number of sent and received messages in average for all the nodes and the number of sent and received messages for the worst node during the time duration $T = 24h = 86400s$ (corresponding to the overall temperature profile shown in Figure 6). By setting n_{RMSG} and n_{SMSG} to the values in Table 2 for the formulas introduced in Section 4.2, we obtained $E \leq 40906.2$ and $E \leq 51967.6$ for, respectively, the fixed and

empirical radio models. Assuming that $C_{BATT} = 2500\ mAh$, then the lifetimes of the worst sensor node for the two models are, respectively, $t_l \geq 660$ and $t_l \geq 519$ days. For the sake of simplicity, we assume that the lifetime of the worst sensor determines the lifetime of the WSN. Obviously, this is not true in the presence of redundant nodes. Here, if a node dies, the function of the node in the multi-hop routing model can be replaced by other nodes. This is likely the case for the empirical model.

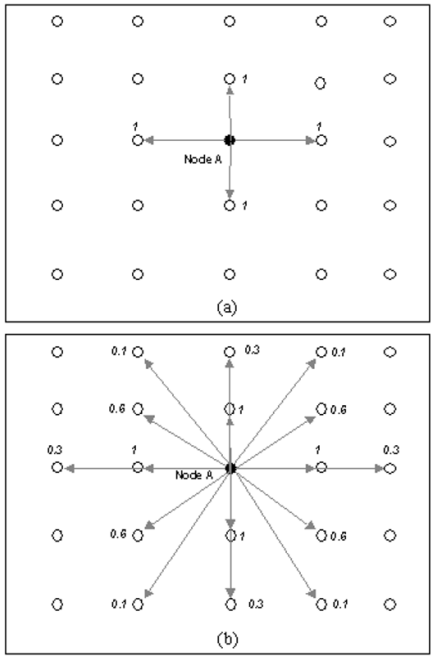

Figure 7: Radio models: (a) Fixed and (b) empirical

As expected, the WSN lifetime using the empirical radio model is shorter than using the fixed radio model. This is due to two factors: the number of neighboring nodes, which is higher in the empirical radio model than in the fixed radio model, and the variability of the number of neighboring nodes. This variability is a consequence of the different probabilities of message reception. The variability increases the number

of messages. Indeed, when a node, say node A, disappears in the neighboring of another node, say node B, B continues to send messages until it does not receive a reply from A or the number of messages sent by B becomes higher than threshold α.

Table 2: Sent and received messages during the 24 hours simulations

		Fixed radio model	Empirical radio model
In average	Sent messages	330	559
	Received messages	749	1359
Worst sensor node	Sent messages	586	1146
	Received messages	1243	2599

To highlight how in our approach the maximum temperature is propagated along the network, we randomly selected a node in the sunny zone and a node in the shadowy zone. Figure 8 and Figure 9 show the estimates of the maximum temperature stored in the two nodes when adopting the empirical radio model. The estimate (continuous line) is compared with the temperature profile (dashed line) used as input to the sensors. We can observe that the estimate in Figure 9 is less precise than the one in Figure 8. Obviously, the estimate in the sensor node placed in the shadowy zone is affected both by the local measurement (2 Celsius degrees less than the maximum temperature) and by the latency determined by the Decision module. Indeed, the Decision Module transmits increases in temperature only if formula (4) holds. Since we set σ_1 to 1^oC, we can have differences in the range $[0^oC, 1^oC]$ between the maximum temperature and the estimate in the nodes in the shadowy zone. The maximum difference is however achieved only in very brief intervals (see spikes in Figure 9). If we consider the energy saving obtained by using our approach, we can conclude that the difference between the estimated and real maximum temperature is more than acceptable.

To assess the goodness of the results achieved by our approach, we executed the same simulations using the method proposed by Boulis et

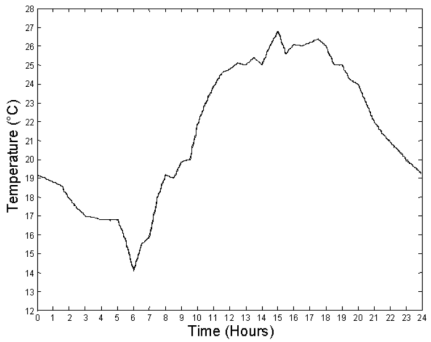

Figure 8: Trend of the maximum temperature estimate in a sensor node placed in the sunny zone

al. in (BGS03) and already discussed in the beginning of this chapter. We set the parameters of the method so as to have comparable accuracy with our approach. In particular, the method takes the decision to propagate a new estimate based on the difference between the new generated estimate and the old estimate. This difference is computed both for the means and for the variances: if either the difference between the means is higher than a pre-fixed threshold m, or the difference between the variances is higher than a pre-fixed threshold v, the new estimate is broadcast. We fixed thresholds m and v to 1^oC and 0.3^oC, respectively. These values are equal to the ones used in our method for the two parameters σ_1 and σ_2, which actually have approximately the same function. Table 3 shows the number of sent and received messages in average for all the nodes and the number of sent and received messages for the worst node during the time duration $T = 24h = 86400s$.

By setting n_{RMSG} and n_{SMSG} to the values in Table 3 for the formulas introduced in Section 4.2, we obtained, $E \leq 41448.3$ and $E \leq 63795.6$ for, respectively, the fixed and empirical radio models. Assuming that $C_{BATT} = 2500$ mAh, then the lifetimes of the worst sensor node for the two models are $t_l \geq 651$ and $t_l \geq 423$ days, respectively. The Boulis et al. approach is comparable to our approach for the fixed radio model, whereas it is considerably worse for the empirical radio model. This be-

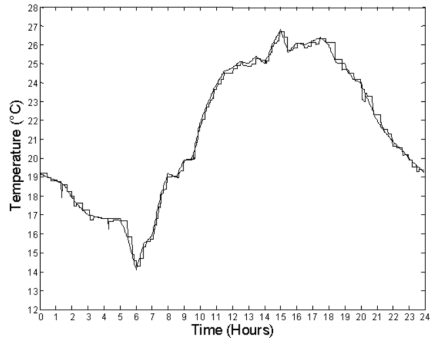

Figure 9: Trend of the maximum temperature estimate in a sensor node placed in the shadowy zone

Table 3: Sent and received messages during the 24 hours simulations (Boulis et Al. approach)

		Fixed radio model	Empirical radio model
In average	Sent messages	490	793
	Received messages	773	1452
Worst sensor node	Sent messages	734	1490
	Received messages	1203	4274

haviour is due to the different management of the overhead of passing on redundant information. In our approach, we structured the message so as to reduce the redundant information on the network. In their paper, Boulis et al. discuss this problem and suggest to maintain information about two-hop neighbors at every node as a possible solution. To this aim, they recommend to adopt the protocol proposed in (MB01). The protocol is executed at network boot-up phase and therefore assumes that the network cannot be modified. Further, maintaining information about two-hop neighbors can be useful only if the nodes involved in the communication are boundary nodes, that is, nodes that have not to forward data to other nodes.

4.4 Power Consumption: a Real Example

In order to apply the algorithm to a real example, we loaded the nesC implementations of the modules described in the previous sections on a network of twelve Tmote Sky nodes to monitor the maximum temperature in a $200m^2$ flat. Figure 10 shows the plant of the flat and the position (black circles) of the nodes in the rooms. The nodes, which are connected to one node through arrows, represent the neighborhood of this node. Neighborhood is determined by the radio range. In our experiments, we set a low radio power so as to allow multi-hop communication ($-25\ dBm$).

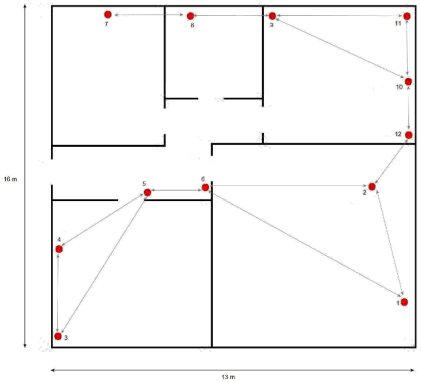

Figure 10: The plant of the example flat

Figure 11 shows the trend of the maximum temperature measured in the network during a 24 hours test during winter. We observe that the maximum temperature decreases in the time interval $[11:00-19:30]$ because the heating system is turned off. At $19:30$ the heating system is turned on and the maximum temperature increases from 17^oC to 20^oC. At $21:00$, the heating system is turned off and the maximum temperature decreases till 18.5^oC. Suddenly, at $23:00$ the temperature increases till 21^oC. This increase is caused by one of the inhabitants of the flat who had a shower, so increasing the temperature in the bathroom.

Table 4 shows the number of sent and received messages for each node

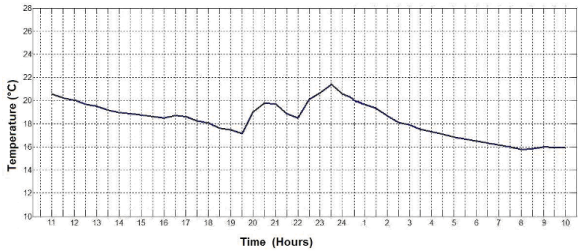

Figure 11: 24 hours test

in the network during the time duration $T = 24h = 86400s$ of the test. We can observe that the nodes (9, 10, 11 and 12), which have a wider neighborhood, exchange a higher number of messages. This is a direct consequence of our algorithm. Indeed, when a node sends a message, it receives an acknowledgement from each neighbor. If the number of neighbors is high, the number of received messages is high too. As an example, we compute the lifetime t_l of node 10. Note that this is the node with the highest power consumption. By setting n_{RMSG} and n_{SMSG} to the values in Table 4 for the formulas introduced in Section 4.2, we obtain that $E_{RX} = 24282.1$ *mJ* and $E_{TX} = 9867.4$ *mJ* for, respectively, 4058 received messages and 1867 sent messages. It follows that $E \leq 64494.6$. If $C_{BATT} = 2500$ *mAh*, then $t_l \geq 418$ days. Considering that, without using the B-MAC protocol and our approach to data aggregation, the lifetime would have been a few days, the value of t_l appears very satisfactory for the type of application.

Table 4: Sent and received messages during the 24 hours test

node-ID	**received messages**	**sent messages**
1	1180	484
2(sink)	2059	1129
3	1150	503
4	1583	674
5	2078	954
6	2041	902
7	1090	401
8	1260	611
9	3672	1650
10	4058	1867
11	3181	1493
12	3057	1359

Chapter 5

A Lossless Compression Algorithm for WSNs

In order to enable lossless compression in WSNs we propose LEC (Lossless Entropy Compression), which exploits the natural correlation that exists in data typically collected by WSN and the principles of entropy compression. Its low complexity and the small amount of memory required for its execution make the algorithm particularly suited to be used on available commercial sensor nodes. Further, the algorithm is able to compute a compressed version of each value on the fly, thus reducing storage occupation. Finally LEC exploits a very short fixed dictionary, whose size depends on the precision of the analog-to-digital converter (ADC). Thus, being the dictionary size fixed a-priori, LEC does not suffer from the growing dictionary problem, which affects other approaches specifically proposed in the literature for WSNs.

We will first test the LEC algorithm on four temperature and relative humidity datasets collected by real WSNs and show that the LEC algorithm, though very simple, can achieve compression ratios up to 70.81% for temperature datasets and up to 62.14% for relative humidity datasets.

Second, we will compare our algorithm with a lossless compression algorithm, namely S-LZW (SM06), and a lossy compression algorithm, namely LTC (SGO+04), specifically designed to be embedded in sensor

nodes. We will show that on average LEC achieves compression ratios about 30% higher than S-LZW using about 11 against 69.185 instructions per saved bit. Further, LTC obtains the same compression ratios as LEC only introducing RMSEs of the order of about 0.1 and 0.5 for temperature and relative humidity, respectively, and using 48.237 instructions per saved bit.

Third, we will test the LEC algorithm on solar radiation, seismic and ECG datasets. These datasets have been chosen because they represent non-smooth signals, which theoretically should not favour the performance of the LEC algorithm. Nevertheless, we will observe that the LEC algorithm achieves compression ratios of the order of 70%.

Finally, we will compare the LEC algorithm with five well-known compression algorithms, namely, gzip, bzip2, rar, classical Huffman encoding and classical arithmetic encoding, though as already shown in (KL05; SM06; BA06), due to their hardware requirements and computational complexity, these five algorithms cannot be used in sensor nodes. In the comparison, since we adopt a differential compression scheme, we will use both the original signals and the differentiated signals as inputs to the five compression algorithms. We will observe that the LEC algorithm achieves compression ratios about 10% higher than the five algorithms when the inputs are the original signals, whereas the five algorithms obtain compression ratios about 15% higher than the LEC algorithm when the inputs are the differentiated signals. On the one hand, these results prove that the choice of adopting a differential-based compression scheme is appropriate. On the other hand, since compression ratios achieved by these algorithms can be considered as a sort of benchmark, a difference of about 15% when the inputs to the five algorithms are the differentiated signals is certainly satisfactory, just considering the simplicity of LEC and the practical impossibility to embed the five algorithms in sensor nodes.

5.1 The LEC Algorithm

The LEC algorithm follows a scheme similar to the one used in the baseline JPEG algorithm for compressing the so-called DC coefficients of a digital image (PM92). Such coefficients are characterized by a high correlation, very similar to that characterizing data collected by WSNs. LEC exploits a modified version of the Exponential-Golomb code (Exp-Golomb) of order 0 (Teu78), which is a type of universal code. The basic idea is to divide the alphabet of numbers into groups whose sizes increase exponentially. Like in Golomb coding (Gol66) and Elias coding (Eli75), a codeword is a hybrid of unary and binary codes: in particular, the unary code (a variable-length code) specifies the group, while the binary code (a fixed-length code) represents the index within the group.

In LEC, we maintain the approach of dividing the alphabet of numbers into groups whose sizes increase exponentially, but groups are entropy coded, rather than unary coded. This modification introduces the possibility of specifying prefix-free codes for the groups: this is a non-negligible advantage since these codes can be re-calculated to best fit the particular probability distribution of the inputs being compressed. Moreover, since the original version of Exp-Golomb manages only nonnegative integers, a minor modification to the original scheme has been the introduction of a bijection to map the actual values onto a nonnegative domain.

In the sensing unit of a sensor node, each measure m_i acquired by a sensor is converted by an ADC to a binary representation r_i on R bits, where R is the resolution of the ADC, that is, the number (2^R) of discrete values the ADC can produce over the range of analog values. Figure 12 shows the block scheme of the LEC approach. For each new acquisition m_i, LEC computes the difference $d_i = r_i - r_{i-1}$, which is input to an entropy encoder (in order to compute d_0 we assume that r_{-1} is equal to the central value among the 2^R possible discrete values). The entropy encoder performs compression losslessly by encoding differences d_i more compactly based on their statistical characteristics. Each nonzero d_i value is represented as a bit sequence bs_i composed of two parts $s_i|a_i$, where s_i

codifies the number n_i of bits needed to represent d_i (that is, the group to which d_i belongs) and a_i is the representation of d_i (that is, the index position in the group). When d_i is equal to 0, the corresponding group has size equal to 1 and therefore there is no need to codify the index position in the group: it follows that a_i is not represented.

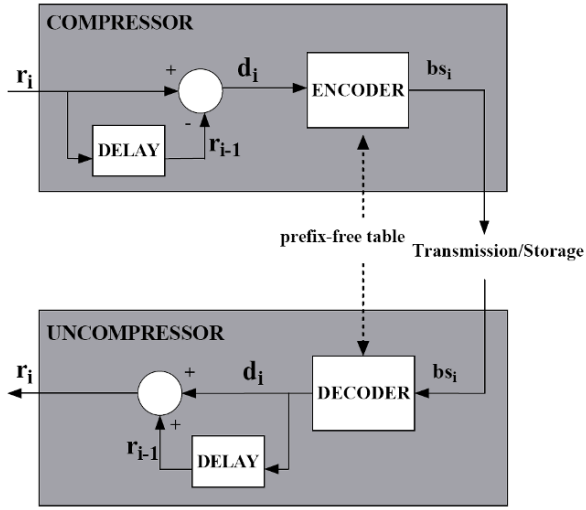

Figure 12: Block diagram of the encoding/decoding schemes

For any nonzero d_i, n_i is trivially computed as $\lceil log_2(|d_i|) \rceil$: at most n_i is equal to R. Thus, in order to encode n_i a prefix-free table of $R + 1$ entries has to be specified. This table depends on the distribution of the differences d_i: more frequent differences have to be associated with shorter codes. In typical data collected by WSNs, we verified that the most frequent differences are those close to 0. Thus, in order to avoid the cost of computing frequencies on sensor nodes, we exploited the amount of work already carried out on JPEG algorithm. We adopted Table 5 in which the first 11 lines coincide with the table used in the baseline JPEG algorithm for compressing the DC coefficients (PM92). On the other hand, these coefficients have statistical characteristics similar to the measures

acquired by the sensing unit. Of course, whether the resolution of the ADC is larger than 14 bits, the table has to be appropriately extended. On the other hand, this extension will produce a higher memory occupation, but will not affect perceptibly the compression ratios. Indeed, differences corresponding to longer codes have a probability very close to 0, thus letting practically unchanged the compression ratios obtainable using our approach.

Table 5: The Huffman variable length codes used in the experiments

n_i	s_i	d_i
0	00	0
1	010	-1,+1
2	011	-3,-2,+2,+3
3	100	-7,...,-4,+4,...,+7
4	101	-15,...,-8,+8,...,+15
5	110	-31,...,-16,+16,...,+31
6	1110	-63,...,-32,+32,...,+63
7	11110	-127,...,-64,+64,...,+127
8	111110	-255,...,-128,+128,...,+255
9	1111110	-511,...,-256,+256,...,+511
10	11111110	-1023,...,-512,+512,...,+1023
11	111111110	-2047,...,-1024,+1024,...,+2047
12	1111111110	-4095,...,-2048,+2048,...,+4095
13	11111111110	-8191,...,-4096,+4096,...,+8191
14	111111111110	-16383,...,-8192,+8192,...,+16383

In order to manage negative d_i, LEC maps the input differences onto nonnegative indexes, using the following bijection:

$$index = \begin{cases} d_i & d_i \geq 0 \\ 2^{n_i} - 1 - |d_i| & d_i < 0 \end{cases} \tag{5.1}$$

Finally, s_i is equal to the value at entry n_i in the prefix-free table and a_i is the binary representation of index over n_i bits. Since d_i is typically represented in two's complement notation, when $d_i < 0$, a_i is equal to the n_i low-order bits of $d_i - 1$. The procedure used to generate a_i guarantees that all possible values have different codes. Using Table 5, we have, for instance, that $d_i = 0$, $d_i = +1$, $d_i = -1$, $d_i = +255$ and $d_i = -255$ are encoded as 00, 010|1, 010|0, 111110|11111111 and 111110|00000000,

respectively. Once bs_i is generated, it is appended to the bitstream which forms the compressed version of the sequence of measures m_i.

```
compress(ri, ri_1, stream)
  // compute difference di
  SET di TO ri - ri_1
  // encode difference di
  CALL encode() with di RETURNING bsi
  // append bsi to stream
  SET stream TO <<stream,bsi>>
  RETURN stream
```

Figure 13: Pseudo-code of the *encode* algorithm

Figure 13 shows the pseudo-code of the *compress()* function based on the LEC algorithm. We denote the concatenation of the bit sequences a and b as $<< a, b >>$. After computing the difference d_i between the binary representations r_i and r_{i-1} of the current and previous measurements, respectively, the *compress()* function calls the *encode()* function passing d_i as parameter. The *compress()* function returns the bit sequence bs_i corresponding to the difference d_i. The sequence bs_i is, therefore, concatenated to the bit sequence stream generated so far.

Figure 14 shows the pseudo-code of the *encode()* function. Here, first the number n_i of bits needed to encode the value of d_i is computed. Then, the part s_i of the bit sequence bs_i is generated by using the table which contains the dictionary adopted by the entropy compressor. Finally, the part a_i of the bit sequence bs_i is generated. In the pseudo-code, $v|_{n_i}$ denotes the n_i low-order bits of v.

Binary logarithm is not in general provided in the instruction sets of the microprocessors on-board commercial sensor nodes and, therefore, $\lceil log_2 (|d_i|) \rceil$ cannot be executed directly. To overcome this problem, we can use function *computeBinaryLog()* shown in Figure 15. The function returns the number n_i of bits needed to encode the value of d_i using only integer divisions.

We observe that the compression algorithm described above is very simple (it can be implemented in a few lines of code) and requires only to maintain the first two columns of Table 5 in memory. An evaluation

```
encode(di, Table)
  IF di = 0 THEN
    SET ni TO 0
  ELSE
    SET ni TO ⌈log2(|di|)⌉    //compute category
  ENDIF
  SET si TO Table[ni]         //extract si from Table
  IF ni = 0 THEN              //build bsi
    SET bsi TO si             //ai is not needed
  ELSE
    IF di > 0 THEN            //build ai
      SET ai TO (di)|ni
    ELSE
      SET ai TO (di − 1)|ni
    ENDIF
    SET bsi TO ≪ si, ai ≫     // build bsi
  ENDIF
RETURN bsi
```

Figure 14: Pseudo-code of the *encode* algorithm

```
computeBinaryLog(di)
  //⌈log2 (|di|)⌉
  SET ni TO 0
  WHILE di > 0
    SET di TO di/2
    SET ni TO ni + 1
  ENDWHILE
  RETURN ni
```

Figure 15: Pseudo-code of the *computeBinaryLog()* function

of its complexity will be performed in the next section. Thus, it results to be particularly attractive to reduce the amount of data generated by the sensing units on-board sensor nodes and consequently to reduce the number of packets to be transmitted, thus saving energy.

5.2 Performance assessment results

In order to show the effectiveness and validity of our compression algorithm, we tested it against various real-world datasets. First, we considered smooth signals like temperature and relative humidity which theoretically are particularly suitable to our algorithm, as explained in the previous sections. Then, we assessed the performance of our algorithm against real non-smooth signals, like solar radiation, seismic and ECG signals.

5.2.1 Smooth signals

We used temperature and relative humidity measurements from four SensorScope deployments (Senb): *HES-SO FishNet Deployment*, *LUCE Deployment*, *Grand-St-Bernard Deployment* and *Le Génépi Deployment*. We chose to adopt public domain datasets rather than to generate data by ourselves to make the comparison as fair as possible. The WSNs adopted in the deployments employ a TinyNode node type (Tin), which uses a TI MSP430 microcontroller, a Xemics XE1205 radio and a Sensirion SHT75 sensor module (Sena). This module is a single chip which includes a capacitive polymer sensing element for relative humidity and a bandgap temperature sensor. Both the sensors are seamlessly coupled to a 14-bit ADC and a serial interface circuit on the same chip. The Sensirion SHT75 can sense air temperature in the $[-20^oC, 60^oC]$ range and relative humidity in the $[0\%, 100\%]$ range. The outputs raw_t and raw_h of the ADC for temperature and relative humidity are represented with resolutions of 14 and 12 bits, respectively. The outputs raw_t and raw_h are converted into measures t and h expressed, respectively, in Celsius degrees (C) and percentage ($\%$) as described in (Sena). The datasets corresponding to the four

deployments store measures t and h. On the other hand, the LEC algorithm works on raw_t and raw_h. Thus, before applying the algorithm, we extracted raw_t and raw_h from t and h, respectively, by using the inverted versions of the conversion functions in (Sena).

We built our datasets by extracting from the four SensorScope deployment data bases the temperature and relative humidity measurements for a randomly extracted sensor node within a specific time interval. Table 6 summarises the main characteristics of the datasets. In the following we will refer the mentioned datasets by using their symbolic names. Table 7 and Table 8 show some statistical characteristics of these datasets. In particular, we have computed the mean $\bar{s}$ and the standard deviation $\sigma_{\bar{s}}$ of the samples, the mean $\bar{d}$ and the standard deviation $\sigma_{\bar{d}}$ of the differences between consecutive samples, the information entropy $H = -\sum_{i=1}^{N} p(x_i) \cdot log_2(p(x_i))$ of the original signal, where N is the number of possible values x_i (the output of the ADC) and $p(x_i)$ is the probability mass function of x_i, and the information entropy $H_d = -\sum_{i=1}^{N} p(d_i) \cdot log_2(p(d_i))$ of the differentiated signal. We can observe that the first two datasets are characterised by lower entropy values than the other two datasets. We expect that performance of entropy compression algorithms on the first two datasets are higher than on the other two datasets.

Table 6: Main characteristics of the four datasets

Deployment Name	NODE ID	Symbolic Name	Number of samples	Time interval	
				From day	To day
HES-SO FishNet	101	FN_ID101	12652	09/08/2007	31/08/2007
LUCE	84	LU_ID84	64913	23/11/2006	17/12/2006
Grand-St-Bernard	10	GSB_ID10	23813	15/09/2007	18/10/2007
Le Génépi	20	LG_ID20	21523	04/09/2007	03/10/2007

Table 7: Statistical characteristics of the four temperature datasets

Dataset	Temperature			
	$\bar{s} \pm \sigma_{\bar{s}}$	$\bar{d} \pm \sigma_{\bar{d}}$	H	H_d
FN_ID101	14.92 ± 3.88	$3.02 \cdot 10^{-4} \pm 0.26$	10.26	5.10
LU_ID84	7.21 ± 3.16	$-2.87 \cdot 10^{-5} \pm 0.05$	10.07	4.05
GSB_ID10	4.34 ± 3.95	$-4.28 \cdot 10^{-4} \pm 0.21$	10.29	6.15
LG_ID20	4.09 ± 4.05	$1.38 \cdot 10^{-4} \pm 0.33$	10.25	6.82

Table 8: Statistical characteristics of the four relative humidity datasets

Dataset	Relative Humidity			
	$\bar{s} \pm \sigma_{\bar{s}}$	$\bar{d} \pm \sigma_{\bar{d}}$	H	H_d
FN_ID101	83.94 ± 9.79	$-1.60 \cdot 10^{-3} \pm 1.26$	9.75	5.84
LU_ID84	87.04 ± 8.04	$1.12 \cdot 10^{-4} \pm 0.55$	10.08	5.85
GSB_ID10	68.17 ± 18.18	$-3.28 \cdot 10^{-4} \pm 1.68$	10.78	7.14
LG_ID20	69.71 ± 19.43	$-1.70 \cdot 10^{-4} \pm 2.77$	10.84	7.67

Compression performance

The performance of a compression algorithm is usually computed by using the compression ratio (CR) defined as:

$$CR = 100 \cdot \left(1 - \frac{encSize}{origSize}\right), \tag{5.2}$$

where $encSize$ and $origSize$ are, respectively, the sizes in bits of the compressed and the uncompressed bitstreams. Considering that uncompressed samples are normally byte-aligned, both temperature and relative humidity samples are represented by 16-bit unsigned integers. Thus, from Table 6, it is easy to compute $origSize$ for the given datasets. Table 9 shows the results obtained for the four datasets. For each dataset, we show the sizes of the uncompressed and compressed bitstreams and the corresponding *CRs*. As expected, the LEC algorithm achieves higher compression ratios on datasets characterised by a low entropy and, in general, a low variability between consecutive samples (that is, low values of the mean and standard deviation of the differences between consecutive samples). Indeed, by comparing the results in Table 9 with the statisti-

cal descriptions of the datasets shown in Table 7 and Table 8, we observe that the LU_ID84 temperature dataset, being characterised by the lowest entropies, the lowest mean and standard deviation of the differences between consecutive samples, achieves the highest compression ratio. On the other hand, we have to point out that samples in LU_ID84 datasets are obtained by measuring temperature and relative humidity at intervals of 30 seconds unlike intervals of 2 minutes used in the other datasets.

Table 9: *CRs* obtained by the LEC algorithm on the four datasets

Dataset	*origSize*	Temperature		Relative Humidity	
		encSize	*CR*	*encSize*	*CR*
FN_ID101	202432	70069	65.39	76635	62.14
LU_ID84	1038608	303194	70.81	396442	61.83
GSB_ID10	381008	156099	59.03	180192	52.71
LG_ID20	344368	158994	53.83	178725	48.10

Let us assume that all samples have to be transmitted to the sink by using the lowest number of messages so as to have power saving (MCP+02). Supposing that each packet can contain at most 29 bytes of payload (MV08), we can count the number of packets necessary to deliver the uncompressed ($origPkt$) and the compressed ($encPkt$) bitstreams. Table 10 summarizes the results obtained. In the table, PCR denotes the packet compression ratio and is defined as:

$$PCR = 100 \cdot \left(1 - \frac{encPkt}{origPkt}\right), \tag{5.3}$$

Table 10: Number of packets needed to deliver the uncompressed and compressed versions of the datasets

Dataset	*origPkt*	Temperature		Relative Humidity	
		encPkt	*PCR*	*encPkt*	*PCR*
FN_ID101	873	302	65.41	331	62.08
LU_ID84	4477	1307	70.81	1709	61.83
GSB_ID10	1643	673	59.04	777	52.71
LG_ID20	1485	686	53.80	771	48.08

We can observe that compression allows reducing considerably the number of transmitted packets, thus saving a lot of power. Since we adopt the send and receive primitives of TinyOS, we transmit or receive fixed-size packets with 29 bytes of payload. Thus, these compression ratios can be achieved only when the payload of the packet is completely full. This implies that in order to use LEC favorably we have to introduce a high data latency. It follows that LEC cannot be used in applications which require data in real time.

With the aim of maintaining our approach as simple as possible, we have adopted the same Huffman table for all datasets. Actually, we would improve our performance by adopting an appropriate Huffman table for each different dataset, and possibly for each sampling rate. The table would have to be derived by occurrence frequencies of the differences between consecutive samples. This can be obtained by using two different approaches. The first approach collects a set of samples sufficient to perform reliable statistics and then compute the occurrence frequencies: this approach is known in the literature as semi-adaptive Huffman coding (Sal07). The second approach generates the Huffman table on the fly by considering at each sample how the occurrence frequencies vary. This approach is called adaptive Huffman coding. The approach was originally developed by Faller (Fal73) and Gallager (Gal78) with substantial improvements by Knuth (Knu85). Obviously, the use of adaptive Huffman coding provides a lot of flexibility despite an increase in complexity. Table 11 shows the compression ratios obtained by the three approaches. As expected, the semi-adaptive Huffman coding performs better than the other two approaches. On the other hand, semi-adaptive Huffman coding has been generated by assuming to exactly know the occurrence frequencies of the differences between consecutive samples. This assumption is difficultly verified in real applications. We can note, however, that the compression ratios obtained by using the same table for all datasets are quite close to those obtained by the other two approaches. Thus, we can avoid the increase of complexity introduced by the adaptive Huffman coding without affecting the compression ratios. On the other hand, the semi-adaptive Huffman coding does not increase complexity, but re-

quires an analysis of occurrence frequency of the differences, which can be performed only after collecting an appropriate number of samples.

Table 11: Comparison between different approaches to Huffman table generation

Dataset	Temperature			Relative Humidity		
	fixed	**semi adaptive**	**adaptive**	**fixed**	**semi adaptive**	**adaptive**
FN_ID101	65.39	67.18	67.13	62.14	62.29	62.18
LU_ID84	70.81	73.54	73.49	61.83	62.51	62.45
GSB_ID10	59.03	60.44	60.35	52.71	54.40	54.35
LG_ID20	53.83	56.35	56.29	48.10	51.08	51

To assess the performance achieved by our approach, we first compare the LEC algorithm to S-LZW and then to LTC. As already discussed in Section 3.3.1 , S-LZW requires to set some parameters: the size of the data block (`BLOCK_SIZE`), the maximum number of dictionary entries (`MAX_DICT_ENTRIES`), the strategy to follow when the dictionary is full (`DICTIONARY_STRATEGY`) and the number of mini-cache entries (`MINI-CACHE_ENTRIES`). Table 12 shows the parameters used in the experiments. We adopted the values suggested in (SM06). Table 13 summarises the results obtained by the S-LZW algorithm in terms of CR on the four datasets. We observe that the LEC algorithm outperforms considerably the CRs obtained by the S-LZW algorithm.

Table 12: S-LZW parameters

`BLOCK_SIZE`	528 Bytes
`MAX_DICT_ENTRIES`	512
`DICTIONARY_STRATEGY`	Frozen
`MINI-CACHE_ENTRIES`	32

As second comparison, we adopt the LTC algorithm. Since it is a lossy compression algorithm, we have to introduce a measure to assess how much the reconstructed data differ from the original data. We use the

Table 13: Compression ratios obtained by the S-LZW algorithm on the four datasets

Dataset	Temperature	Relative Humidity
	CR	***CR***
FN_ID101	30.35	36.27
LU_ID84	48.99	31.24
GSB_ID10	26.86	24.92
LG_ID20	22.02	21.93

root mean squared error ($RMSE$), defined as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{5.4}$$

where y_i is the original sample, $\hat{y}_i$ is the reconstructed sample and N is the number of samples of the signal. Obviously a lossless compression algorithm will result in an $RMSE = 0$, since there is no difference between y_i and $\hat{y}_i$ (the reconstructed signal is exactly equal to the original). The LTC algorithm generates a set of line segments which form a piecewise continuous function. This function approximates the original dataset in such a way that no original sample is farther than a fixed error e from the closest line segment. Thus, before executing the LTC algorithm, we have to set error e. We choose e as a percentage of the Sensor Manufactured Error (SME). From the Sensirion SHT75 sensor data sheet (Sena), we have $SME = \pm 0.3^oC$ and $SME = \pm 1.8\%$ for temperature and relative humidity, respectively. To analyse the trend of the CRs with respect to increasing values of e, we varied e from 0% to 200% of the SME with step 10%.

Figure 16 and Figure 17 show, for all the four datasets, the resulting trends of the CRs for temperature and relative humidity, respectively. Due to the particular approach based on linear approximation used in LTC, we observe that the lossless version of LTC (corresponding to $e = 0$) generates a negative CR, that is, the size of the compressed data is larger than the original data. Further, we note that interesting CRs are obtained only with relevant compression errors. Figure 18 and Figure 19 show,

for all the four datasets, the trends of the $RMSE$ for temperature and relative humidity, respectively. As expected, the $RMSE$ varies almost linearly with error e.

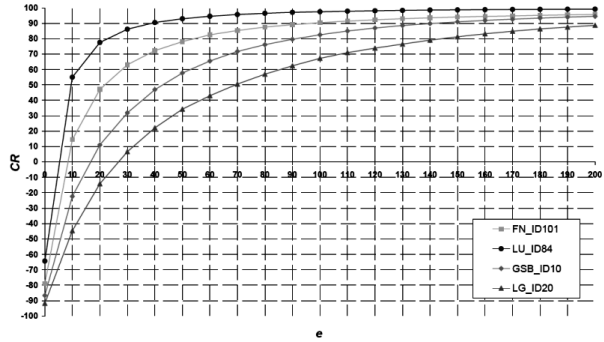

Figure 16: Compression ratios obtained by the LTC algorithm for different error values on the four temperature datasets

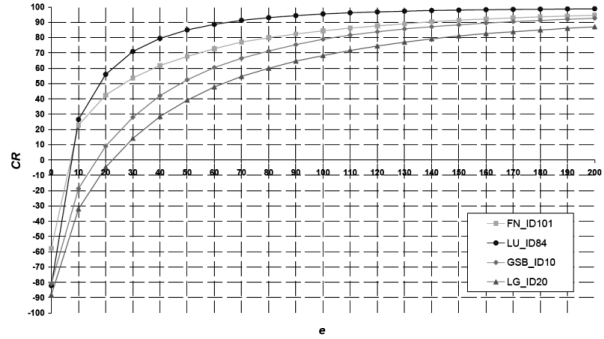

Figure 17: Compression ratios obtained by the LTC algorithm for different error values on the four relative humidity datasets

By comparing Table 9, where we have shown the CRs obtained by the LEC algorithm, with Figure 16-Figure 19, we can observe that the LTC algorithm can achieve the same CRs as the LEC algorithm, but despite a quite high $RMSE$. For instance, the LEC algorithm achieves a CR equal to 65.39 for the FN_ID101 temperature dataset. From Figure 16, we can

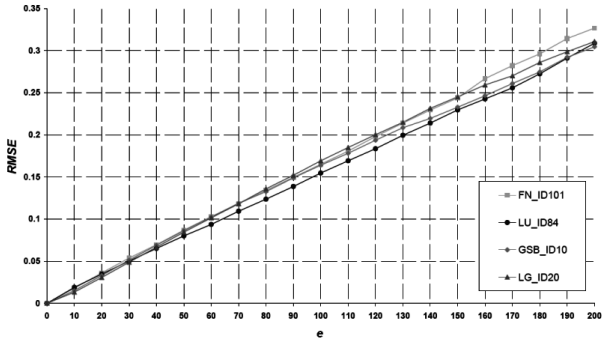

Figure 18: Root mean squared errors obtained by the LTC algorithm for different error values on the four temperature datasets

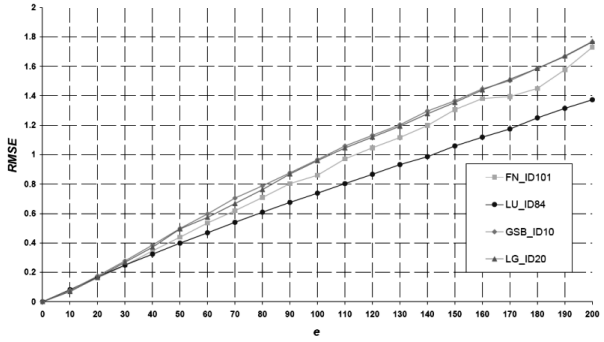

Figure 19: Root mean squared errors obtained by the LTC algorithm for different error values on the four relative humidity datasets

observe that, in order to achieve similar CRs, the LTC algorithm should be executed with a value of e between 30% and 40% of the temperature SME. From Figure 18, we can deduce that these values of e lead to have an $RMSE$ between 0.0538 and 0.0696. Table 14 and Table 15 are generated from similar considerations.

Table 14: Correspondences between LEC compression ratios and LTC compression ratios and root mean squared errors for the four temperature datasets

Dataset (Temperature)	**LEC *CR***	***e* [min, max]**	**LTC *CR* [min, max]**	**LTC *RMSE* [min, max]**
FN_ID101	65.39	[30, 40]	[63.09, 72.26]	[0.0538, 0.0696]
LU_ID84	70.81	[10, 20]	[55.00, 77.53]	[0.0190, 0.0348]
GSB_ID10	59.03	[50, 60]	[57.75, 65.54]	[0.0863, 0.1025]
LG_ID20	53.83	[70, 80]	[50.54, 56.97]	[0.1182, 0.1359]

Table 15: Correspondences between LEC compression ratios and LTC compression ratios and root mean squared errors for the four relative humidity datasets

Dataset (Relative Humidity)	**LEC *CR***	***e* [min, max]**	**LTC *CR* [min, max]**	**LTC *RMSE* [min, max]**
FN_ID101	62.14	[40, 50]	[61.81, 67.67]	[0.3461, 0.4390]
LU_ID84	61.83	[20, 30]	[55.97, 70.99]	[0.1681, 0.2496]
GSB_ID10	52.71	[50, 60]	[52.44, 60.49]	[0.4979, 0.5971]
LG_ID20	48.10	[60, 70]	[47.79, 54.69]	[0.5733, 0.6674]

Note that e equal to 30%, 10%, 50% and 70% (the left extremes of the e intervals in Table 14) correspond to tolerate a temperature error of 0.09^oC, 0.03^oC, 0.15^oC, 0.21^oC, respectively, for each reconstructed sample. Similarly, e equal to 40%, 20%, 50% and 60% (the left extremes of the e intervals in Table 15) correspond to tolerate a relative humidity error of 0.72%, 0.36%, 0.9%, and 1.08% respectively, for each reconstructed sample. Just to give an idea of the effects of the errors on the original data, we compare the first block (264 bytes) of the original samples to the corresponding reconstructed samples for the all datasets. We executed the LTC algorithm with values of error e equal to the left extremes of the e intervals shown

in Table 14 and Table 15. Figure 20-Figure 23 show the results of the comparison for temperature and for values of e equal to 30%, 10%, 50% and 70%, respectively. Figure 24-Figure 27 show the results of the comparison for relative humidity and for values of e equal to 40%, 20%, 50% and 60%, respectively. We observe that there exists a considerable difference between the original and reconstructed samples. Thus, to achieve the CRs of the LEC algorithm, the LTC algorithm can only approximate the original samples. Further, we have to consider that using the extreme values of the e interval corresponds to execute the LTC algorithm in the most favourable condition with respect to the LEC algorithm.

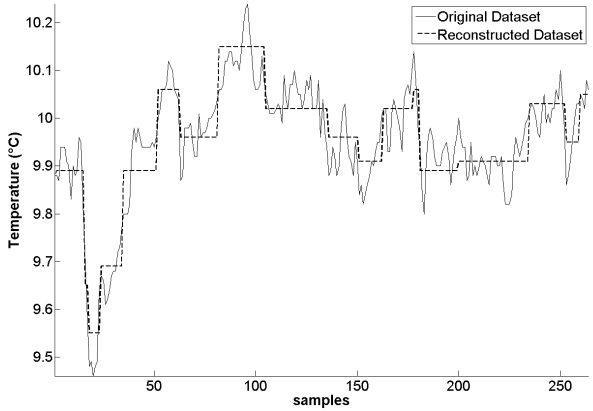

Figure 20: Comparison between the original and the reconstructed samples for the first block of the FN_ID101 temperature dataset: compression performed by the LTC algorithm with $e = 30\%$

Complexity

Compression ratio is only one of the factors which determine the choice of a compression algorithm suited to WSNs. Another fundamental factor is complexity. To assess the complexity of our algorithm and of the algorithms considered for comparison, we have performed a comparative analysis on the number of instructions required by each algorithm to compress data. To this aim, we have adopted the Sim-It Arm simula-

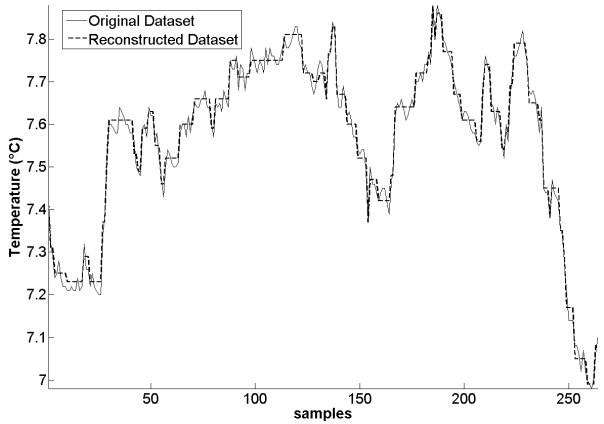

Figure 21: Comparison between the original and the reconstructed samples for the first block of the LU_ID84 temperature dataset: compression performed by the LTC algorithm with $e = 10\%$

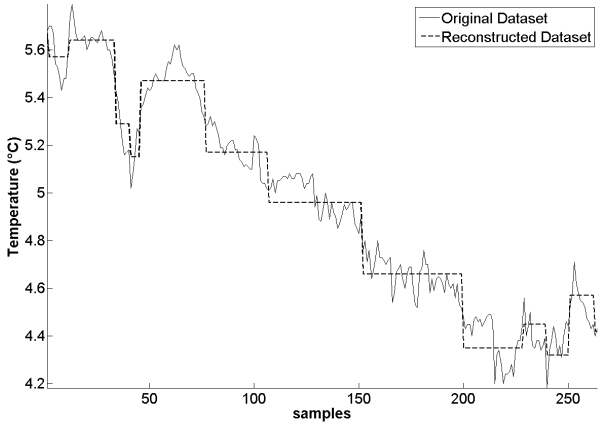

Figure 22: Comparison between the original and the reconstructed samples for the first block of the GSB_ID10 temperature dataset: compression performed by the LTC algorithm with $e = 50\%$

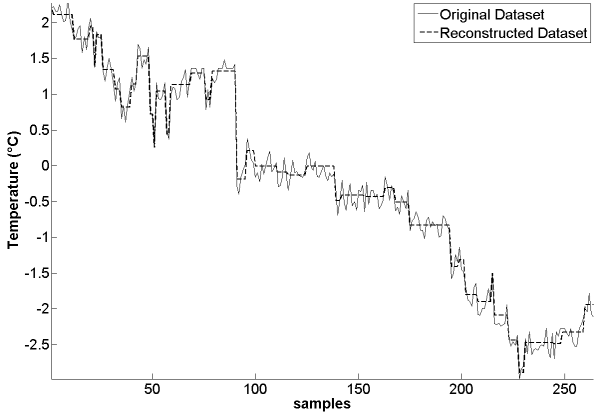

Figure 23: Comparison between the original and the reconstructed samples for the first block of the LG_ID20 temperature dataset: compression performed by the LTC algorithm with $e = 70\%$

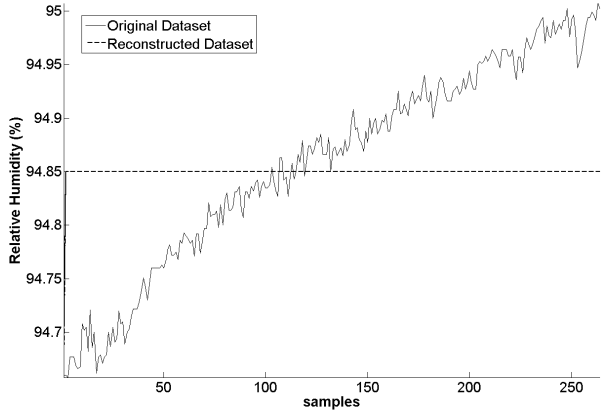

Figure 24: Comparison between the original and the reconstructed samples for the first block of the FN_ID101 relative humidity dataset: compression performed by the LTC algorithm with $e = 40\%$

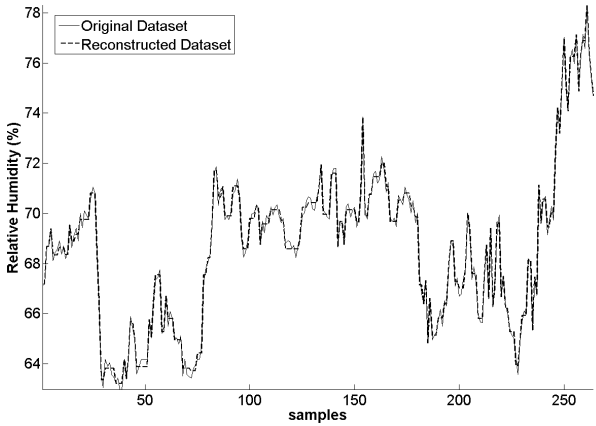

Figure 25: Comparison between the original and the reconstructed samples for the first block of the LU_ID84 relative humidity dataset: compression performed by the LTC algorithm with $e = 20\%$

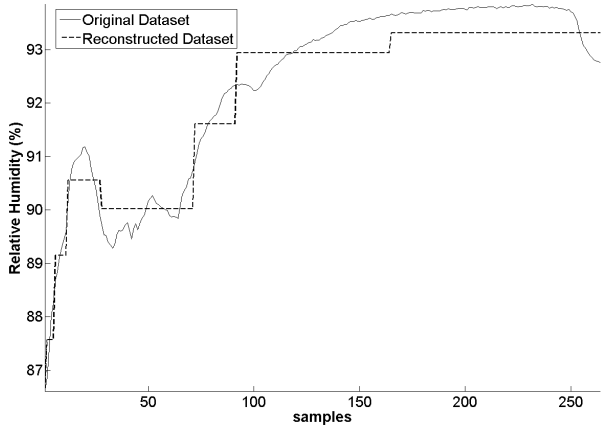

Figure 26: Comparison between the original and the reconstructed samples for the first block of the GSB_ID10 relative humidity dataset: compression performed by the LTC algorithm with $e = 50\%$

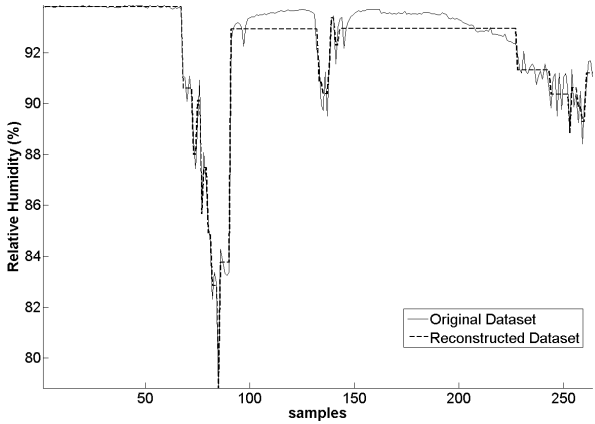

Figure 27: Comparison between the original and the reconstructed samples for the first block of the LG_ID20 relative humidity dataset: compression performed by the LTC algorithm with $e = 60\%$

tor (Sim), since there already existed a free available version of S-LZW implemented for this simulator by the same authors of this compression algorithm. Sim-It Arm is an instruction-set simulator that runs both system-level and user-level ARM programs. Since S-LZW compresses each dataset block by block, we executed the three algorithms on Sim-It Arm simulator to compress the first block of each dataset. A block consists of 528 bytes (corresponding to 264 samples of 16 bits). S-LZW has been executed by using the parameters in Table 12. For LTC, we have set e to the left extremes of the e intervals in Table 14 and Table 15. Table 16 shows the average numbers of instructions required for compressing a block, the average numbers of saved bits and the average numbers of instructions per saved bit for temperature and relative humidity datasets, respectively.

We note that, though our algorithm achieves the highest compression ratios among the considered algorithms, it requires the lowest number of instructions. We observe that, in average, the LEC algorithm executes, for instance, 11.35 instructions for each saved bit against 68.65 and 48.16 executed by S-LZW and LTC for compressing the first block of each tem-

Table 16: Complexity of the three compression algorithms

	LEC		S-LZW		LTC	
	Temp.	**Rel.Hum.**	**Temp.**	**Rel.Hum.**	**Temp.**	**Rel.Hum.**
Avg. number of instructions	30549.25	26610.25	63207.00	63207.00	144149.75	150005.50
Avg. number of saved bits	2762.75	2844.00	1598.00	1880.00	3088.00	2919.00
Avg. number of instructions per saved bit	11.35	10.58	68.65	184.55	48.16	237.38

perature dataset.

The problem of the first sample

LEC, as all the differential compression algorithms, suffer from the following problem. In order to reconstruct the original samples, the decoder must know the value of the first sample: if the first sample has been lost or corrupted, all the other samples are not correctly decoded. In our case, the compressed bitstream is sent by a wireless communication to the collector, which takes the decompression process in charge. Since the transmission can be non-reliable, the first packet could be lost and thus also the first value, making correct reconstruction of samples impossible.

To make a communication reliable a number of solutions have been proposed. In general, these solutions involve protocols based on acknowledgements which act at Transport layer. Obviously, these protocols require a higher number of message exchanges between nodes and this increases the power consumption. A review of these algorithms is out of the scope of this thesis. Anyway, a solution to this problem can be also provided at the application layer without modifying the protocols of the underlying layers: when we insert the first sample into the payload of a new packet, we do not insert the difference between the current and the previous sample, but rather the difference between the current sample and a reference value known to the decoder (for instance, the central value of the ADC). Thus, the decoding of each packet is independent of the re-

ception of the previous packets. In the following the packet compression ratios obtained by using this expedient will be denoted by PCR^*. Table 17 shows a comparison between $PCRs$ and PCR^*: we can note that the decrease of PCR is not high. Further, the PCR^* are still higher than those achieved by S-LZW. Thus, we can conclude that the LEC scheme can be made more robust without significantly affecting its performance.

Table 17: Comparison between the standard packet compression ratios ($PCRs$) and the ones obtained by transmitting the first value in each packet ($PCRs^*$)

Dataset	Temperature		Relative Humidity	
	PCR	PCR^*	PCR	PCR^*
FN_ID101	65.39	62.41	62.14	58.63
LU_ID84	70.81	68.19	61.83	58.21
GSB_ID10	59.03	55.49	52.71	48.43
LG_ID20	53.83	49.88	48.10	43.48

5.2.2 A comparison with standard compression algorithms

In this section, we compare our approach with well-known compression methods. We show the compression ratios obtained on both the original signals and the differentiated signals, that is, the signals obtained by differentiating pairs of consecutive samples. We also use the differentiated signals to understand whether the choice of using a differential-based compression scheme is correct. We consider five compression methods: gzip, bzip2, rar, classical Huffman encoding and classical arithmetic encoding. Gzip, bzip2 and rar have a parameter which allows setting the compression level. This parameter is between 1 and 9 (default 6) for gzip and bzip2, and between 1 and 5 (default 3) for rar. We fixed this paramater to the maximum possible compression (9 for gzip and bzip2 and 5 for rar).

We point out that the algorithms discussed in this section, as already shown in (KL05; SM06; BA06), cannot be executed in a sensor node, due to memory requirements and computational power needed for their execution. Indeed, the executable codes are too large to be embedded in tiny

sensor nodes. Further, the compression ratios are obtained after collecting all the samples and therefore all the samples have to be stored in memory. This implies that large datasets cannot be managed. In addition, the compression cannot be performed on the fly. Finally, during their execution, these algorithms require a large memory to manage some step of the execution. Thus, we used these algorithms only as benchmarks to validate the compression ratios obtained by applying LEC. Table 18 shows the results obtained by these algorithms on the four temperature and relative humidity datasets.

Table 18: Compression ratios obtained by five classical compression algorithms on the four datasets

Dataset	Algorithm	**Temperature**		**Relative Humidity**	
		ORIG	DIFF	ORIG	DIFF
FN_ID101	Gzip	34.76	74.48	41.29	70.96
	Bzip2	55.20	78.16	56.22	74.99
	Rar	63.59	79.75	59.12	68.66
	Huffman	21.59	75.56	23.19	71.69
	Arithmetic	22.06	68.72	23.34	67.39
LU_ID84	Gzip	48.87	80.23	37.86	70.94
	Bzip2	69.24	83.22	57.82	75.32
	Rar	69.16	85.03	59.03	77.46
	Huffman	23.98	79.64	23.79	72.61
	Arithmetic	26.62	72.99	24.06	66.56
GSB_ID10	Gzip	34.35	69.78	31.02	65.85
	Bzip2	52.12	74.18	45.73	70.40
	Rar	56.72	75.95	49.70	72.44
	Huffman	22.32	72.12	18.97	68.45
	Arithmetic	22.75	66.35	19.37	64.61
LG_ID20	Gzip	31.38	65.95	27.61	63.46
	Bzip2	46.84	70.99	42.56	67.72
	Rar	51.56	72.79	45.26	69.83
	Huffman	22.34	69.30	18.85	66.78
	Arithmetic	22.65	64.19	19.30	63.65

We observe that our algorithm outperforms all the algorithms when the input is the original signal. The other algorithms, however, outperform our algorithm when the input is the differentiated signal. On the one hand, these results prove that the choice of adopting a differential-based compression scheme is winning for this type of data. On the other hand,

the compression ratios shown in Table 18 are not so far from the ones achieved by our approach. Further, we have to consider that these compression ratios are obtained by applying the algorithms after collecting all the samples, thus allowing the selection of the optimal codes. In addition, these algorithms require the execution of a large number of instructions: also if it was possible to embed these algorithms in a tiny sensor node, the power consumption would result to be considerably larger than the energy saved by reducing the number of bits. All in all, we think that, considering the limited resource requirements of LEC, the compression ratios achieved by LEC can be considered very significant with respect to both the results shown in Table 18 and the compression ratios obtained by the two algorithms purposely adapted to sensor networks described in the previous sections.

5.2.3 Non-smooth signals

The datasets described in Section 5.2.1 are smooth over time and therefore are characterised by a strong correlation between consecutive samples. These signals are quite typical in WSNs because they are produced by sensors dedicated to measure atmospheric (and therefore non rapidly changing) phenomena.

To test our approach against non-smooth (and possibly high-rate) signals, we consider three further datasets:

1. the solar radiation dataset in FN_ID101 deployment. Here, 12652 samples have been collected by a Davis Solar Radiation sensor (Dav). Figure 28 shows a part of the dataset.

2. a seismic dataset consisting of a collection of 36000 seismic data samples collected by the OhioSeis Digital Seismographic Station located in Bowling Green, Ohio, covering measurements from 2:00 PM to 3:00 PM on September 21, 1999 (UT). Each measurement in this dataset represents the vertical displacement of the spring in the seismometer, measured in micrometers, acquired with a frequency of 10 Hz (Sei). Figure 29 shows a part of the dataset.

3. a single-channel ambulatory ECG recording extracted from the 100.dat file in the MIT-BIH Arrhythmia Database (ECG). Each recording in this dataset represents the digitized measurement of the first-channel performed with a frequency of 360 Hz (360 samples per second) with 11-bit resolution over a 10 mV range. The recording consists of a collection of 30000 data samples. Figure 30 shows a part of the dataset.

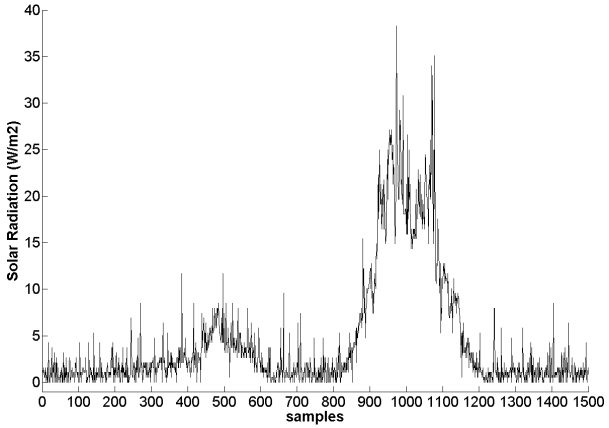

Figure 28: First 1500 samples of the solar radiation dataset

We point out that only the first dataset has been collected by using the sensor nodes considered in this thesis. The other datasets have been employed because they are often used to compare different compression techniques in the literature. Anyway, these datasets can be collected by appropriately adapted sensor nodes.

Table 19 shows some statistical characteristics of these datasets. The results obtained by LEC are summarized in Table 20, where comparisons with S-LZW, gzip, bzip2, rar, classical Huffman encoding and classical arithmetic encoding are also shown. We can observe that our algorithm achieves considerable compression ratios also when applied to non-smooth (and possibly high-rate) signals, thus confirming its validity and reliability. Actually, we note that LEC achieves on average higher compression

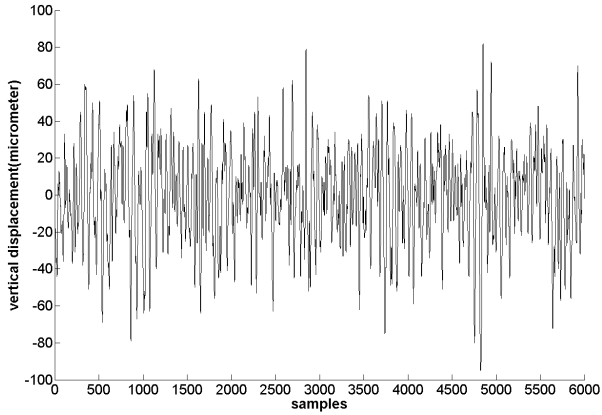

Figure 29: First 6000 samples of the seismic dataset

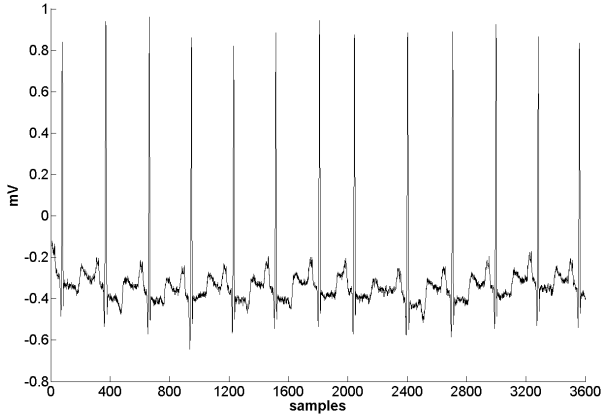

Figure 30: First 3600 samples of the ECG dataset

ratios on the non-smooth signals than on the smooth signals. On the other hand, by comparing Table 7 and Table 8 with Table 19 we can observe that temperature and relative humidity datasets are characterized by higher entropies than solar radiation, seismic and ECG datasets. Since LEC is an entropy compression algorithm, independently of the smoothness of the signals, its performance increases when the entropy decreases. This difference between the entropies of smooth and non-smooth datasets can be justified as follows: though temperature and relative humidity vary slowly and smoothly, the number of possible values of differences between consecutive samples is on average larger than the number of possible values of differences between consecutive samples of solar radiation, seismic and ECG signals. In other words, these signals are characterized by possibly large differences between consecutive samples, but these differences are quite repetitive. It follows that a larger number of symbols is needed for encoding temperature and relative humidity signals than for solar radiation, seismic and ECG signals.

Table 19: Statistical characteristics of the three non-smooth datasets

Dataset	$\bar{s} \pm \sigma_{\bar{s}}$	$d \pm \sigma_{\bar{d}}$	H	H_d
Solar Radiation	10.36 ± 15.35	$1.260 \cdot 10^{-4} \pm 7.83$	5.34	4.24
Seismic	-0.36 ± 26.96	$-7.78 \cdot 10^{-4} \pm 3.65$	6.79	3.91
ECG	-0.33 ± 0.18	$-4.83 \cdot 10^{-6} \pm 0.05$	6.23	3.78

Table 20: Compression ratios obtained by LEC, S-LZW and five classical compression algorithms on three non-smooth datasets

Dataset	LEC	S-LZW	Algorithm	ORIG	DIFF
Solar Radiation	70.31	58.25	Gzip	60.53	78.41
			Bzip2	71.34	80.82
			Rar	68.38	82.73
			Huffman	56.49	78.20
			Arithmetic	51.15	71.45
Sismeic	69.72	43.33	Gzip	62.72	86.66
			Bzip2	80.50	90.15
			Rar	83.25	87.86
			Huffman	44.50	79.39
			Arithmetic	40.15	71.63
ECG	71.53	54.59	Gzip	72.02	81.88
			Bzip2	74.17	86.08
			Rar	72.02	81.39
			Huffman	47.62	79.84
			Arithmetic	43.33	73.44

Chapter 6

A loss-aware Compression Algorithm for WSNs

In Chapter 5, we have tackled the problem to develop a lossless compression scheme suitable for sensor networks proposing a simple algorithm based on a modified version of the exponential Golomb code. Moreover in Section 3.3.1, we have introduced some concepts to justify the need of enabling lossless data compression in sensor nodes. In few words, the criticality of some application domains demand sensors with high accuracy, since these applications cannot tolerate that measures are corrupted by the compression process.

On the other hand, when the application does not impose strong requirements on accuracy (i.e., environmental monitoring), cheap and unreliable sensors should be employed in large scale. Due to noise, such sensors will produce different readings even when they are sampling an unchanging phenomenon. For this reason, sensor manufactures specify not only the sensor operating range but also the sensor accuracy. Datasheets express accuracy by providing a margin of error, but typically do not include a probability distribution for this error. Thus, when a value is measured by a sensor, we are confident that the actual value is within the error margin, but cannot know with what probability that value is some distance from the real value (SGO+04). Signals produced by these sensors

are generally "cleaned" by employing some de-noising technique.

A de-noising technique can be applied directly on the sensor node or offline on the base station. In the first case, an energy-aware and low-complexity de-noising technique is required (WDB06). Further, before executing the de-noising process a quite large number of samples has to be stored in the data memory, increasing the memory requirements of the sensor node. In the second case, the sensor node simply acquires, compresses and stores noisy samples; when a packet payload is filled, the sensor node sends the compressed packet to the base station, where data are uncompressed and later de-noised. Obviously, this approach can be adopted only if data have not to be transferred in real-time. In applications of environmental monitoring, for instance, this is the typical case. In this scenario the use of a lossless compression algorithm lets us simply post-pone the de-noising process so as to perform it on a machine, with generally no hard constraint on energy, computational power and memory. We have to consider, however, that noise increases the entropy of the signal and therefore hinders the lossless compression algorithm to achieve considerable compression ratios. Indeed data which will be discharged by the de-noising process at the base station are equally transmitted by the sensor node affecting power consumption and consequently the sensor node lifetime.

In this scenario, the ideal solution would be to adopt on the sensor node a lossy compression algorithm in which the loss of information would be just the noise. Thus, we could achieve high compression ratios without losing relevant information.

To this aim, we exploit the observation that data typically collected by WSNs are strongly correlated. Thus, differences between consecutive samples should be regular and generally very small. If this does not occur, it is likely that samples are affected by noise. To de-noise and simultaneously compress the samples, we quantize the differences between consecutive samples. Further, to reduce the number of bits required to code these differences, we adopt a Differential Pulse Code Modulation (DPCM) scheme (Cut52). Of course, different combinations of the quantization process parameters determine different trade-offs between com-

pression performance and information loss. To generate a set of optimal combinations of the quantization process parameters, we adopt one of the most popular Multi-Objective Evolutionary Algorithms (MOEAs), namely NSGA-II (DPAM02).

MOEAs generate a family of equally valid solutions, where each solution tends to satisfy a criterion to a higher extent than another. Different solutions are compared with each other by using the notion of Pareto dominance. A solution x associated with a performance vector $\mathbf{u}$ dominates a solution y associated with a performance vector $\mathbf{v}$ if and only if, $\forall i \in \{1, ..., I\}$, with I the number of criteria, u_i performs better than v_i, or equal to, v_i and $\exists i \in \{1, ..., I\}$ such that u_i performs better than v_i, where u_i and v_i are the i-th elements of vectors $\mathbf{u}$ and $\mathbf{v}$, respectively. A solution is said to be Pareto optimal if it is not dominated by any other possible solution. The set of Pareto-optimal solutions is denoted as Pareto front. Thus, the aim of an MOEA is to discover a family of solutions that are a good approximation of the Pareto front.

To execute NSGA-II, we first collect a short sequence of samples from the sensor node. Then, we apply a popular de-noising technique to this sequence so as to obtain a sequence of de-noised samples. Each solution generated by NSGA-II is evaluated by quantizing the original samples and computing the information entropy of the quantized sequence as first objective, the number of levels used in the quantization process as second objective and the mean square error (MSE) between the quantized samples and the de-noised samples as third objective. The entropy and the number of levels provide an indirect measure of the possible obtainable compression ratios. The MSE quantifies the loss of information with respect to the ideal (not affected by noise) signal. Each solution in the Pareto front represents, therefore, a quantizer with an associated trade-off among information entropy, number of quantization levels and MSE between the original de-noised and the quantized sequences of samples. We show that the lossy compression scheme obtained by using the quantizers generated by the MOEAs in the DPCM framework is characterized by low complexity and memory requirements for its execution. Further, it is able to compute a compressed version of each value on the fly, thus

reducing storage occupation.

We will test our lossy compression approach on three datasets collected by real WSNs. We will show that, though very simple, our approach can achieve significant compression ratios despite negligible reconstruction errors, that is, MSEs between the original de-noised and the reconstructed signals. We will compare our approach again with LTC (SGO$^+$04), remembering that it was specifically designed to be embedded in sensor nodes. We will show that our approach outperforms LTC in terms of compression ratios (and consequently number of messages sent by a generic sensor node to transmit measures to the sink), complexity (average number of instructions required to compress a sample) and reconstruction errors, thus representing a very interesting state-of-art solution to the problem of compressing noisy data in WSNs.

6.1 DPCM and Quantization Principles

Our compression scheme is a purposely adapted version of the DPCM scheme often used for digital audio signal compression. DPCM is a member of the family of differential compression methods. These methods exploit the high correlation that typically exists between neighboring samples of smooth digitized signals, achieving compression by appropriately encoding differences between these samples.

The simplest differential encoder calculates and encodes the differences $d_i = s_i - s_{i-1}$, between consecutive samples s_i. The first data sample, s_0, is either encoded separately or is written on the compressed stream in raw format; in both the cases the decoder can reconstruct s_0 exactly. We will adopt the second solution. The decoder simply reverses the encoding tasks, in a symmetric manner, that is, it decodes the differences d_i and uses the decoded values to generate the reconstructed samples s_i ($s_i = s_{i-1} + d_i$).

In principle, any suitable method, lossy or lossless, can be used to encode the differences. In practice, scalar quantization is often used, resulting in lossy compression. The quantity encoded is therefore not the difference d_i but its quantized version, denoted by $\hat{d}_i$. The difference

q_i between d_i and $\hat{d}_i$ is denoted as *quantization error*. The design of the quantizer is a crucial point in the development of a DPCM compression scheme.

Let $S = \{S_1, ..., S_L\}$ be a set of *cells* S_l, with $l \in [1..L]$, which form a disjoint and exhaustive partition of the input domain D (difference domain in our case). Let $C = \{y_1, \ldots, y_L\}$ be a set of levels $y_l \in S_l$, with $l \in [1..L]$. The quantization process is performed by a quantization operator $Q : D \rightarrow C$ such that $Q(d_i) = y_l \Leftrightarrow d_i \in S_l$. Cells S_l are often expressed in the form of intervals $S_l = (a_{l-1}, a_l]$, where the bounds a_l are called *thresholds*. The width of a cell is expressed by $|a_l - a_{l-1}|$ (GN98). The quantization rule can be expressed as $Q(d_i) = g(\lfloor f(d_i) \rfloor)$, where $\lfloor f(\cdot) \rfloor$ returns the index l of the cell S_l, which the difference d_i belongs to, and $g(\cdot)$ returns the quantized output $\hat{d}_i = y_l$.

A quantizer is said to be uniform if the levels y_i are equispaced and the thresholds are midway between adjacent levels. If an infinite number of levels are allowed, then all cells S_i will have equal width Δ. If only a finite number of levels are allowed, then all but two cells will have width Δ and the outermost cells will be semi-infinite. Given a uniform quantizer with cell width Δ, the region of the input space within $\Delta/2$ of some quantizer level is called the *granular region* or simply the *support* and that outside (where the quantizer error is unbounded) is called the *overload* or *saturation region* (GN98).

When a good rate-distortion performance is requested to a quantizer, the zero-cell width is usually treated individually, even if the quantizer is uniform. Since each input within the zero-cell is quantized to 0, this cell is often called *dead zone*: uniform quantizers with dead zone have been successfully employed, for example, in many international standards for image and video coding, such as JPEG and MPEG (T.01).

Unfortunately, the introduction of the quantization block in a DPCM scheme introduces a new problem, namely, the *accumulation of errors* problem (Sal07). This problem is easy to understand by analyzing separately the operations performed by the encoder and the decoder. The encoder generates the exact differences d_i from the original data samples s_i and s_{i-1}, while the decoder generates the reconstructed sample $\hat{s}_i$ using only

the quantized differences $\hat{d}_i$. The decoder decodes the $\hat{d}_i$'s and uses them to generate reconstructed samples $\hat{s}_i$ ($\hat{s}_i = \hat{s}_{i-1} + \hat{d}_i$) rather than the original samples s_i's. The generic *n-th* reconstructed sample $\hat{s}_n$ at decoder will contain the sum of the quantization errors accumulated during the reconstruction of the previous $n-1$ samples plus the quantization error of the current sample:

$$\hat{s}_n = s_n + \sum_{i=1}^{n} q_i \tag{6.1}$$

To overcome this problem, the encoder is modified so as to compute the differences $d_i = s_i - \hat{s}_{i-1}$, that is, to calculate difference d_i by subtracting the most recent reconstructed value $\hat{s}_{i-1}$ (which both encoder and decoder have) from the current original sample s_i. Thus, the decoder first decodifies s_0. Then, when it receives the first quantized difference $\hat{d}_1$, it computes $\hat{s}_1 = s_0 + \hat{d}_1 = s_0 + d_1 + q_1 = s_1 + q_1$. When it receives the second quantized difference $\hat{d}_2$, it computes $\hat{s}_2 = \hat{s}_1 + \hat{d}_2 = \hat{s}_1 + d_2 + q_2 = \hat{s}_1 + s_2 - \hat{s}_1 + q_2 = s_2 + q_2$. The decoded value $\hat{s}_2$ contains just the single quantization error q_2. In general, the decoded value $\hat{s}_i$ is equal to $s_i + q_i$, thus it contains just the quantization error q_i.

Typically, the DPCM scheme takes also advantage of the fact that the current sample depends on several and not only one of its near neighbors. Thus, to improve prediction, we can use K of the previously seen neighbors to encode the current sample s_i by using a prediction function in the form $\Phi(\hat{s}_{i-1}, \ldots, \hat{s}_{i-K})$. Methods which use such a predictor are called *differential pulse code modulations*.

Usually the $\hat{d}_i$ data sequence is further compressed by using any lossless compression algorithm which makes $\hat{d}_i$ more compactly represented by removing some redundancies. Typically run length encoding schemes, entropy schemes and arithmetic schemes are adopted to achieve this data compaction (Sal07).

6.2 Our lossy compression scheme

Figures 31 and 32 show the block diagrams of our purposely adapted compressor and uncompressor, respectively. As regards the compressor,

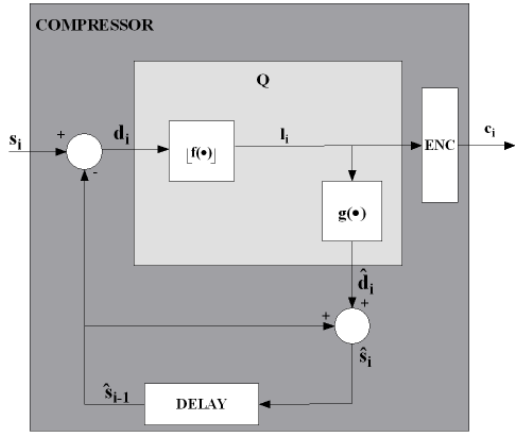

Figure 31: Block diagram of the compressor

the generic difference d_i is calculated by subtracting only the most recent reconstructed value $\hat{s}_{i-1}$, that is, there is only a delay block rather than a prediction block. The introduction of a prediction block, in fact, would have caused an overall increase in the complexity of the compression algorithm, without a tangible increase of the compression performance (at least for the type of data typically collected by WSNs). The $\lfloor f(\cdot) \rfloor$ block returns the index l_i of the cell S_{l_i} which d_i belongs to. The index l_i is input to the $g(\cdot)$ block, which computes the quantized difference $\hat{d}_i$, and to the encoding block *ENC*, which generates the codeword c_i (O'N76).

In the uncompressor, the codeword c_i is analyzed by the decoding block *DEC* which outputs the index l_i. This index is elaborated by the block $g(\cdot)$ to produce $\hat{d}_i$, which is added to $\hat{s}_{i-1}$ to output $\hat{s}_i$.

As regards the block *ENC*, it is well-known in information theory that, when the quantization indexes have unequal probabilities, assigning equal numbers of bits to all quantization indexes is wasteful (Sha48). Indeed, the number of bits produced by the quantizer will be reduced if shorter binary codewords are assigned to more probable indexes. This observation is used in the entropy encoders, which encode more probable indexes with lower number of bits. Further, the set of binary codewords

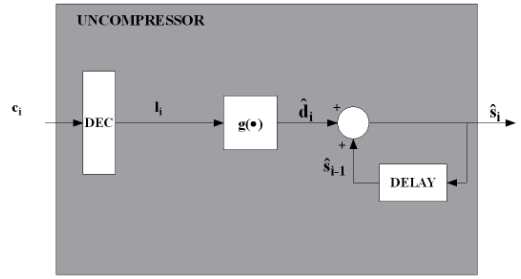

Figure 32: Block diagram of the uncompressor

satisfies the prefix condition, that is, no member is a prefix of another member, in order to ensure unambiguous decodability.

In our case, inputs d_i to the quantizer represent the general differences between consecutive digitized environmental data samples s_i. In general, environmental signals are quite smooth and therefore small differences are more probable than large. Thus, we can use an entropy encoder in order to further compress the integer-valued quantization indexes.

Any scalar quantization operator Q can be adopted. In deciding the type of operator, we have to consider that a coarse quantization generates a high data reduction, but also a high reconstruction error at the decoder. Further, the operator cannot be computationally heavy, since it is executed on a battery powered tiny device. Thus, a right trade-off among compression, reconstruction error and complexity has to be found.

In the next section, we propose to use an MOEA to determine a set of optimal operators with different trade-offs among information entropy H, quantization complexity C and mean square error MSE_d between the quantized and the de-noised samples.

Information entropy H provides an indirect measure of the possible obtainable compression ratios and is defined as:

$$H = -\sum_{l=1}^{L} p_l \cdot \log_2(p_l) \tag{6.2}$$

where p_l is the probability mass function of quantization index l.

Quantization complexity C is computed as the number L of distinct quantization levels (and consequently indexes) used in the quantizer:

$$C = L. \tag{6.3}$$

A lower value of C implies a lower number of symbols needed to encode the quantization indexes and therefore a lower number of bits in the binary codewords.

The MSE_d quantifies the loss of information with respect to the ideal (not affected by noise) signal and is defined as

$$MSE_d = \frac{1}{N} \sum_{i=1}^{N} (s_i - s_i^*)^2 \tag{6.4}$$

where N is the number of samples, and s_i and s_i^* are, respectively, the original and the de-noised samples. To de-noise the samples, we have adopted the wavelet shrinkage and thresholding method proposed in (DJ94). We have used the Symmlet 8 wavelet, a level of decomposition equal to 5 and the soft universal thresholding rule for thresholding the detail coefficients at each level. The de-noising process has been performed by using standard Matlab built-in functions. To provide a glimpse of the effectiveness of the de-noising approach, Fig. 33 shows a real temperature dataset collected by a sensor on board a node of a WSN before and after applying the de-noising process. We can observe how the noise is practically completely removed.

6.3 The Optimization Framework

MOEAs have been investigated by several authors in recent years (ZDT00). Some of the most popular among these algorithms are the Strength Pareto Evolutionary Algorithm (SPEA) (ZT99) and its evolution (SPEA2) (ZLT02), the Niched Pareto Genetic Algorithm (NPGA) (HNG94), the different versions of the Pareto Archived Evolution Strategy (KC00), and the Non-dominated Sorting Genetic Algorithm (NSGA) (SD94) and its evolution (NSGA-II) (DPAM02). Since NSGA-II is considered as one of the most effective MOEAs, we used NSGA-II in the experiments. On the other hand,

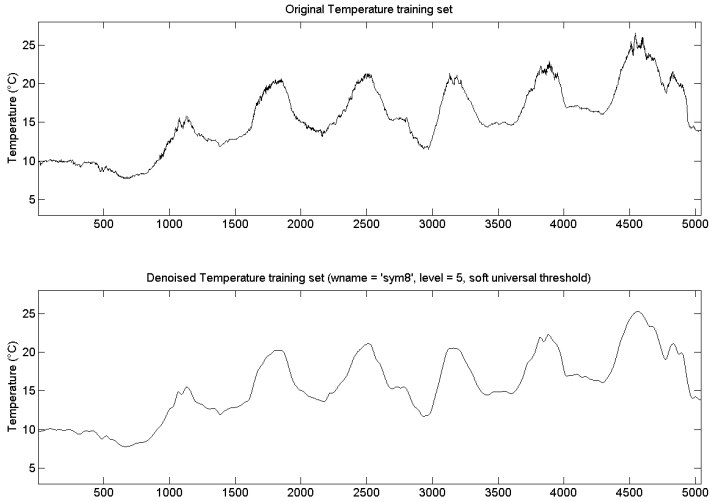

Figure 33: Portions of the original and de-noised signals

to compare the performance of different MOEAs is out of the scope of this thesis: we only aim to show the effectiveness of an MOEA approach in determining a set of quantizers which allow achieving different trade-offs among information entropy, quantization complexity and MSE_d. We use the jMetal (DNL$^+$06) implementation of NSGA-II for our optimization.

The choice of the solution with the best trade-off among H, C and MSE_d for the specific application can be made on the basis of the constraints which have to be satisfied at the moment.

In the following subsections, we describe the chromosome coding, the genetic operators and the NSGA-II algorithm.

6.3.1 The chromosome coding

Each chromosome codifies a different quantizer. The choice of the parameters which identify the quantizers is based on the following considerations. The signals collected by sensors on board nodes are affected by

noise: these sensors produce different readings even when they are sampling an unchanging phenomenon. To reduce this problem, the quantizer has to be characterized by a dead zone. Further, to guarantee a higher flexibility than a uniform quantizer, but without complicating too much the quantization rule, we split the granular region into two subregions. Then, we partition both the subregions uniformly with appropriate different cell widths. It follows that each quantizer is determined by the following five parameters:

1. width of the dead zone (DW)
2. width of the cell in the first granular subregion (FW)
3. number of cells in the first granular subregion (FN)
4. width of the cell in the second granular subregion (SW)
5. number of cells in the second granular subregion (SN)

Every general difference within the interval $(-DW,DW)$ is quantized to zero.
The first granular subregions $(-DW-FN\cdot FW,-DW]$ and $[DW,DW+FN\cdot FW)$ are uniformly partitioned by FN cells of FW width.
The second granular subregions $(-DW-FN\cdot FW-SF\cdot SW,-DW-FN\cdot FW]$ and $[DW+FN\cdot FW,DW+FN\cdot FW+SF\cdot SW)$ are uniformly partitioned by SN cells of SW width.
The differences which fall in the two semi-infinite saturation regions $(-\infty,-DW-FN\cdot FW-SF\cdot SW]$ and $[DW+FN\cdot FW+SF\cdot SW,+\infty)$ are quantized to the midway of the adjacent cells of the second granular subregions. It follows that equation (6.3) can be rewritten as:

$$C = 2\cdot(FN+SN)+1 \tag{6.5}$$

The choice of these parameters originates several different types of quantizers. In the chromosome, each parameter is expressed as a positive integer in the range $[1, MAX]$ and is codified by a Gray binary code. The value of MAX depends on the resolution of the ADC on board the sensor node. On the other hand, to constrain the upper bound of the range

reduces the search space and allows a better exploration. In our experiments we set $MAX = 64$: it follows that each chromosome is represented by a string of 30 bits.

Each chromosome is associated with a vector of three elements, where each element expresses the fulfillment degree of the three objectives (Equations (6.2), (6.5), (6.4)). To compute H, C and MSE_d, we use a small set (*training set*) of samples collected by the sensor on board the node.

6.3.2 Genetic operators

We apply classical one-point crossover operator and a gene mutation operator (Mic94). The one-point crossover operator cuts two chromosomes at some chosen common point and swaps the resulting sub-chromosomes. The common point is chosen by extracting randomly a number in $(1, 30)$.

In the mutation operator a point is randomly selected and its value is swapped (0 becomes 1 and vice versa). The crossover operator is applied with probability P_X; the mutation operator is applied with probability P_M. In the experiments, we adopted $P_X = 0.9$ and $P_M = 0.02$.

In order to select the mating operators and probability values, we performed several experiments by comparing the different Pareto fronts obtained by applying NSGA-II with different crossover and mutation operators, and different probabilities. We verified that the selected mating operators and probability values allow obtaining the widest Pareto fronts and the best trade-offs among H, C and MSE_d.

6.3.3 NSGA-II

The NSGA-II algorithm was introduced in (DPAM02) as an improved version of the Non-dominated Sorting Genetic Algorithm (SD94). It is a population-based genetic algorithm, which uses an ad-hoc density-estimation metric and a non-dominance rank assignment. NSGA-II starts from an initial random population P_0 of N_{pop} individuals (100 in our experiments) sorted based on the non-dominance. Each individual is associated with a rank equal to its non-dominance level (1 for the best level, 2 for the next-best level, and so on). To determine the non-dominance level (and

consequently the rank), two entities are computed for each individual p: i) the number n_p of individuals that dominate p and ii) the set S_p of individuals dominated by p. All individuals with $n_p = 0$ belong to the best non-dominance level associated with rank 1. Indeed, these individuals are dominated by no other individual. To determine the individuals associated with rank 2, for each solution p with rank 1, we visit each member q of the set S_p and decrease n_q by one. If n_q becomes zero, then q belongs to the non-dominance level associated with rank 2. The procedure is repeated for each solution with rank 2, rank 3 and so on until all fronts are identified. At each iteration t, $t = 0, \ldots, T_{max}$, an offspring population Q_t of size N_{pop} is generated by selecting mating individuals through the binary tournament selection, and by applying the crossover and mutation operators. Parent population P_t and offspring population Q_t are combined so as to generate a new population $P_{ext} = P_t \cup Q_t$. Then, a rank is assigned to each individual in P_{ext} as explained above. Based on these ranks, P_{ext} is split into different non-dominated fronts, one for each different rank. Within each front, a specific crowding measure, which represents the sum of the distances to the closest individual along each objective, is used to define an ordering among individuals: in order to cover the overall objective space, individuals with large crowding distance are preferred to individuals with small crowding distance. The new parent population P_{t+1} is generated by selecting the best N_{pop} individuals (considering first the ordering among the fronts and then among the individuals) from P_{ext}. The algorithm terminates when the number of iterations achieves T_{max} (10000 in our experiments).

6.4 Performance assessment results

In order to show the effectiveness and validity of our lossy compression approach, we tested it against some real-world temperature datasets. In particular, we used temperature measurements from three SensorScope deployments (Senb), already introduced in Section 5.2.1: *HES-SO FishNet Deployment*, *Grand-St-Bernard Deployment* and *Le Génépi Deployment*. For a detailed description of these datasets the reader can refer to Section 5.2.1.

6.4.1 The optimization process

We used the first $N = 5040$ samples from FN_ID101 temperature dataset as training set. Since this dataset is a collection of temperature samples collected with a frequency of 1 sample each 2 minutes, it is equivalent to consider a 7-days training set. The extracted portion of the original signal is first de-noised and then converted back to raw data (we recall that our compression scheme works on raw data). Fig. 33 shows the portion of the original signal used to build the training set and its de-noised version.

We applied NSGA-II to the training set. At the end of the optimization process we obtained an archive of non-dominated solutions with respect to the three objectives. In particular, the objective C (quantization complexity) has allowed us to steer the search of the best solutions towards the ones with the minimum number of cells. Thus, during the evolution, the archive is populated preferably by quantizers which, having equal entropy and MSE_d, are characterized by a lower number of cells, thus avoiding to consider quantizers with a high number of unused cells. This allows simplifying the implementation of the quantizer and consequently of the encoder. Indeed, if the number of indexes is low, the encoder can use a small dictionary to encode the quantization indexes. This dictionary can be generated by using the Huffman's algorithm (Huf52) which provides a systematic method of designing binary codes with the smallest possible average length for a given set of symbol occurrence frequencies. Once the binary codeword representation of each quantization index has been computed and stored in the sensor node, the encoding phase reduces to a look-up table consultation.

The only critic point of this approach is that the Huffman's algorithm requires to know the probability with which the source produces each symbol in its alphabet. To determine an approximation of these probabilities, we can exploit again the training set: for the specific quantizer, we compute the probability with which each quantization index occurs when quantizing the differences between consecutive samples of the training set and build the optimal dictionary for that data source by applying the Huffman's algorithm.

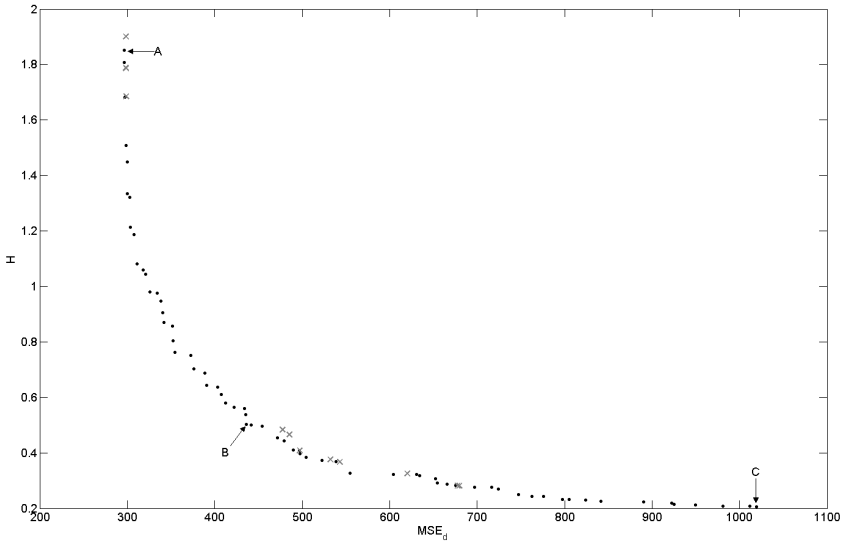

Figure 34: Projection of the Pareto front approximation on the $H - MSE_d$ plane

If we project the final archive on the $MSE_d - H$ plane (see Fig. 34), we realize that actually almost all solutions maintain the non-dominance property with respect to the H and MSE_d objectives: only 12 out of 100 solutions result to be dominated by one or more solutions in the archive. In the figure, dots and crosses represent, respectively, non-dominated and dominated solutions with respect to the H and MSE_d objectives. Non-dominated solutions in the $MSE_d - H$ plane are actually the solutions of interest. We can observe that the front is wide and the solutions are characterized by a good trade-off between H and MSE_d.

6.4.2 Selected Solutions and their validation

To perform an accurate analysis of some solution, we selected from the front in Fig. 34 three significant quantizers: solutions (A) and (C) characterized by, respectively, the highest H and MSE_d, and solution (B) char-

Solution	DZ	FW	FN	SW	SN
A	8	1	3	2	1
B	32	15	1	5	1
C	63	61	1	44	1

Table 21: Parameters of solutions (A), (B) and (C)

index	Probability $[A, B, C]$	Codeword $[A, B, C]$
-4	$[0.1198, -, -]$	$[101, -, -]$
-3	$[0.0163, -, -]$	$[100011, -, -]$
-2	$[0.0200, 0.0052, 0]$	$[100001, 1011, 1111]$
-1	$[0.0252, 0.0311, 0.0131]$	$[10010, 11, 110]$
0	$[0.6309, 0.9258, 0.9730]$	$[0, 0, 0]$
1	$[0.0023, 0.0311, 0.0139]$	$[10011, 100, 10]$
2	$[0.0186, 0.0069, 0]$	$[100010, 1010, 1110]$
3	$[0.0206, -, -]$	$[100000, -, -]$
4	$[0.1258, -, -]$	$[11, -, -]$

Table 22: Codewords used in solutions (A), (B) and (C)

acterized by a good trade-off between H and MSE_d. Table 21 shows the values of the five parameters which characterize the three selected quantizers.

Solutions (A), (B) and (C) correspond, respectively, to the quantization rules represented in Fig. 35, Fig. 36 and Fig. 37, where the black dots and the circles represent the differences d_i and quantized differences $\hat{d}_i$, respectively. Table 22 shows the quantization indexes, their probabilities and the codewords assigned by the Huffman's algorithm when the selected quantizers are used in the proposed scheme for compressing the training set.

To assess the performances of the three compression algorithms generated by, respectively, the quantizers corresponding to (A), (B) and (C), we use the compression ratio (CR) and the two different packet compression ratios (PCR and PCR^*) defined in Chapter 5.

Table 23 shows the CR, PCR and PCR^* obtained for the three temperature datasets and the three selected quantizers. Further, the table reports the MSE between the original noisy and the reconstructed sam-

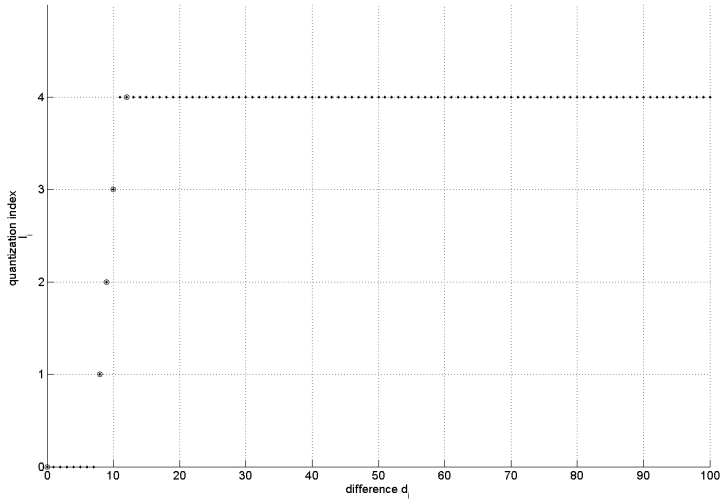

Figure 35: Quantization rule for solution (A)

Dataset	Solution	CR	PCR	PCR^*	MSE_n	MSE_d
TRAINING SET	A	87.8919	87.6437	87.0690	0.0120	0.0036
	B	92.9253	92.8161	92.2414	0.0206	0.0076
	C	93.4834	93.3908	93.1034	0.0785	0.0351
FN_ID101	A	87.4679	87.3998	86.8270	0.0676	0.0431
	B	92.8128	92.7835	92.3253	0.0305	0.0097
	C	93.4264	93.3565	93.0126	0.0905	0.0384
GSB_ID10	A	86.1882	86.1838	85.4534	0.0436	0.0071
	B	91.7356	91.7225	91.2355	0.0250	0.0043
	C	93.1495	93.1223	92.7572	0.0887	0.0275
LG_ID20	A	85.2867	85.2525	84.4444	0.1200	0.0241
	B	89.7784	89.7643	89.2256	0.0422	0.0032
	C	92.4888	92.4579	92.0539	0.0882	0.0173

Table 23: Results obtained by solutions (A), (B) and (C) on the three datasets

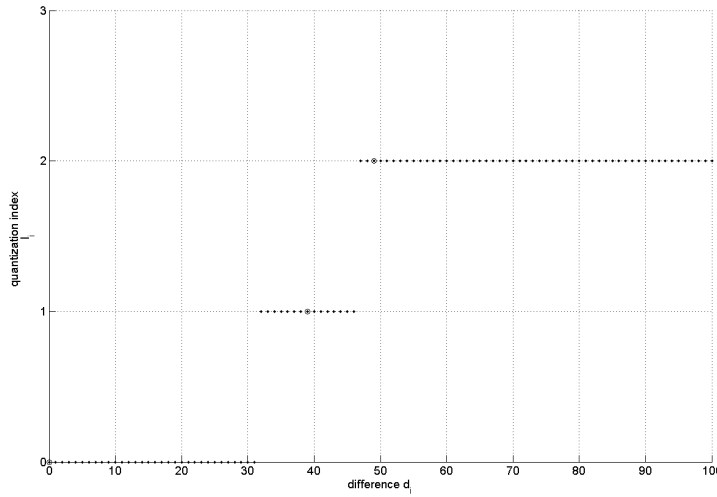

Figure 36: Quantization rule for solution (B)

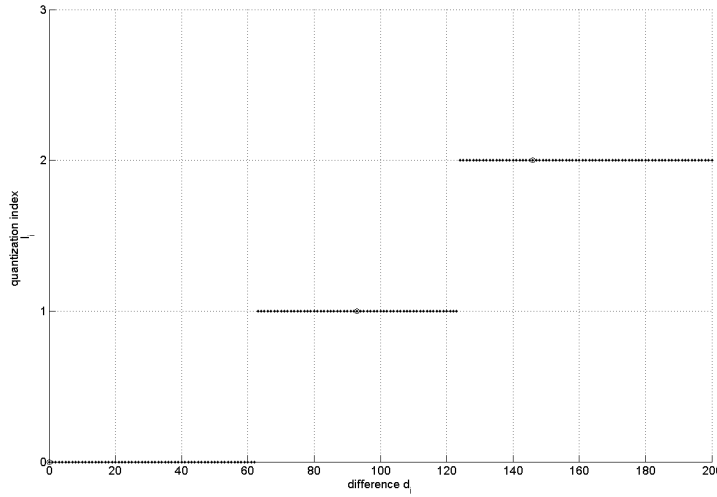

Figure 37: Quantization rule for solution (C)

ples, denoted as MSE_n, and MSE_d. We can observe that all solutions achieve good trade-offs between compression ratios and MSE. Further, there do not exist considerably differences between the results obtained in the training set and the ones achieved in the overall FN_ID101 dataset and the other two datasets. This result could be considered enough surprising. Indeed, we highlight that both the optimization and the Huffman's algorithm were executed using only a portion of the FN_ID101 dataset. Thus, GSB_ID10 and LG_ID20 datasets are completely unknown to the compression scheme. We are conscious that the procedure adopted for FN_ID101 could have been exhaustively applied also to the other datasets, in order to find ad-hoc solutions for the particular deployment. On the other hand, the three temperature datasets were collected, though in different places and times, by the same sensor nodes with the same type of temperature sensor and the same sampling frequency: for this reason it could be unnecessary to perform the optimization on each dataset. To validate this assumption for each selected solution, we executed the Huffman's algorithm on a portion of $N = 5040$ samples extracted respectively from GSB_ID10 and LG_ID20 datasets and used the resulting dictionaries to compress the corresponding datasets. Table 24 shows the CR, PCR and PCR^* obtained in this case. If we compare the results in Table 24 with those in Table 23, we can observe that the increases in CR, PCR and PCR^* are very small and almost negligible, thus confirming the possibility of adopting the same encoding for similar applications of the same sensor. Similar considerations (for the sake of brevity these results are not shown) can be also made for the genetic optimization.

Solution (B) which was chosen on the knee of the Pareto front is characterized by compression ratios comparable to those achieved by solution (C) and by MSE_d comparable to those obtained by solution (A). This solution therefore represents a good trade-off between compression ratios and MSE_d. For this reason, we chose this solution to perform the comparisons discussed in the following subsections.

Dataset	Solution	CR	PCR	PCR^*
GSB_ID10	A	86.1882	86.1838	85.4534
	B	91.7797	91.7833	91.2355
	C	93.1532	93.1223	92.7572
LG_ID20	A	85.5332	85.5219	84.5792
	B	90.2866	90.2357	89.697
	C	92.4888	92.4579	92.0539

Table 24: Compression ratios and packet compression ratios achieved by the three solutions (A), (B) and (C) on GSB_ID10 and LG_ID20 datasets when applying the Huffman's algorithm to training sets extracted from the two datasets

6.4.3 Comparison with LTC

To assess the effectiveness of our approach, we adopt the LTC algorithm proposed in (SGO+04), and already used as comparison in Chapter 5. Again, to analyze the trend of LTC's CRs with respect to increasing values of e, we varied e from 0% to 200% of the SME with step 10% (e and SME are set as in Chapter 5).

6.4.4 Compression ratio and distortion

Fig. 38 shows, for all the three datasets, the resulting trends of the CRs obtained by the LTC algorithm for different error values. Due to the particular approach based on linear approximation used in LTC, we observe that the lossless version of LTC (corresponding to $e = 0$) generates a negative CR, that is, the size of the compressed data is larger than the original data. Further, we note that interesting CRs are obtained only with relevant compression errors. Fig. 39 shows, for all the three datasets, the trends of the MSE_n. By comparing Table 23, where we have shown the CRs obtained by the our algorithm, with Figs. 38 and 39, we can observe that the LTC algorithm can achieve the same CRs as our algorithm, but despite a quite high MSE_n. For instance, our algorithm achieves a CR equal to 92.81 for the FN_ID101 temperature dataset. From Fig. 38, we can observe that, in order to achieve similar CRs, the LTC algorithm should be executed with a value of e between 120% and 130% of the SME.

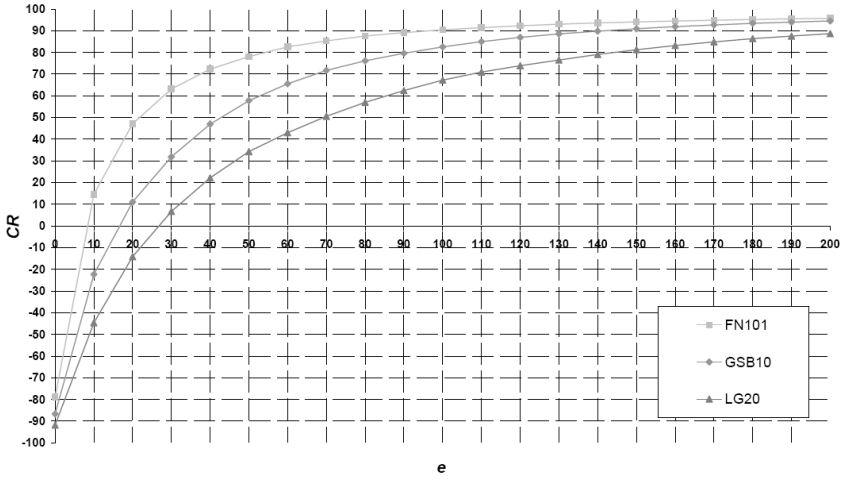

Figure 38: Compression ratios obtained by the LTC algorithm for different error values on the three datasets

From Fig. 39, we can deduce that these values of e lead to have an MSE_n between 0.039 and 0.045 against an $MSE_n = 0.030$ for our algorithm. Table 25 shows the correspondences between the CR achieved by our algorithm, and the CR, MSE_n and MSE_d obtained by LTC on the three datasets. By comparing Table 25 with Table 23, we can observe that our algorithm obtains lower MSEs than LTC in correspondence to equal CRs. For example, for the FN_ID101 dataset and CR equal to 92.81, MSE_d is between 0.0972 and 0.0121 for LTC, whereas $MSE_d = 0.0097$ for our algorithm.

6.4.5 Complexity

Again, to assess the complexity of our algorithm and of LTC, we have performed a comparative analysis on the number of instructions required by each algorithm to compress data. To this aim, we have adopted the Sim-It Arm simulator (Sim). For LTC, we have set e to the left extremes of the e intervals in Table 25 (we recall that the left extremes are the

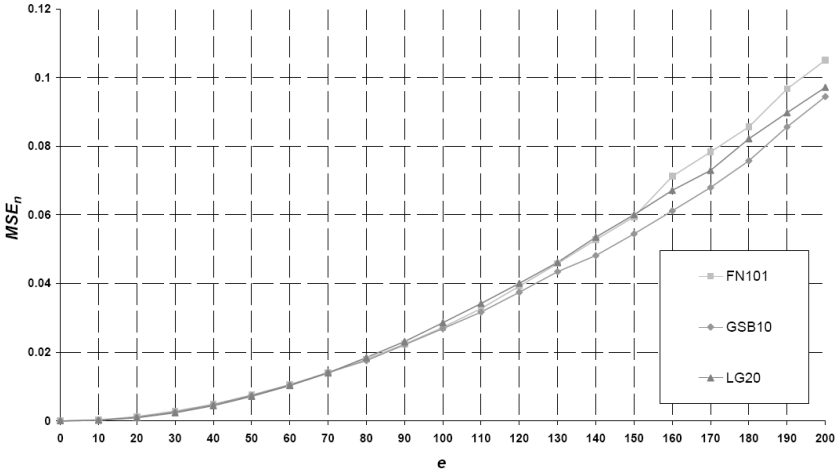

Figure 39: Mean squared errors obtained by the LTC algorithm for different error values on the three datasets.

Dataset	CR	e $[min, max]$	CR $[min, max]$	MSE_n $[min, max]$	MSE_d $[min, max]$
FN_ID101	92.8128	$[120, 130]$	$[92.23, 92.98]$	$[0.0390, 0.0450]$	$[0.0972, 0.0121]$
GSB_ID10	91.7356	$[150, 160]$	$[90.90, 91.90]$	$[0.0540, 0.0600]$	$[0.0140, 0.0172]$
LG_ID20	89.7784	$[200, 210]$	$[88.65, 89.51]$	$[0.0960, 0.1041]$	$[0.0203, 0.0229]$

Table 25: Correspondences between compression ratios achieved by our algorithm, and compression ratios and mean squared errors achieved by LTC on the three datasets

Dataset	instr.		saved bits		instr/saved bits	
	our	LTC	our	LTC	our	LTC
FN_ID101	1065656	7440517	187868	186768	5.67	39.83
GSB_ID10	2050525	13951039	349520	346442	5.86	40.26
LG_ID20	1919529	12574212	309168	305296	6.20	41.18
average	**1678570**	**11321923**	**282185**	**279502**	**5.92**	**40.43**

Table 26: Complexity of our algorithm and LTC

most favourable cases for the LTC algorithm). Table 26 shows the numbers of instructions required for compressing each dataset, the number of saved bits, the numbers of instructions per saved bit for each temperature datasets and their average values. We note that, though our algorithm achieves lower MSEs at the same bitrate as LTC, it requires lower number of instructions. We observe that, on average, our algorithm executes 5.93 instructions for each saved bit against 40.43 executed by LTC.

Chapter 7

Conclusions and open issues

The difficulties of in-processing strategies in WSNs mainly stem from the constraints imposed by the simplicity of sensor devices: limited power, limited communication bandwidth and processing capabilities, and small storage capacity.

In this thesis, we first have focused on the study and design of a distributed aggregation framework: we have proposed a novel distributed approach based on fuzzy numbers and weighted average operators to perform energy efficient aggregation. We have shown that the algorithm is able to reduce the number of received and sent messages without affecting the quality of the aggregate estimation. We have applied our algorithm to the monitoring of the maximum temperature in a 100-node simulated WSN and a 12-node real WSN. In the hypothesis of using the B-MAC protocol with LPL, we have computed the lifetimes of the two WSNs for the simulated and real applications (exploiting the mathematical model proposed by B-MAC's authors, adapted to our application). In the worst case, the WSN lifetime is approximately 418 days with a battery capacity of 2500mAh. If we had maintained the radio on all the time, the WSN lifetime would have been a few days. Thus, we have concluded that the combination B-MAC protocol and proposed aggregation allows

considerably prolonging the WSN lifetime.

Then, we have concentrated our effort on the design of compression algorithms for single node compression. In this context, we have introduced LEC, a simple lossless compression algorithm particularly suited to the reduced storage and computational resources of a WSN node. The compression scheme exploits the high correlation that typically exists between consecutive samples collected by a sensor on-board a node. Thanks to this characteristic and following the principles of entropy compression, the algorithm is able to achieve considerable compression ratios and to compute a compressed version of each value acquired from a sensor on the fly. We have evaluated the LEC algorithm by compressing four temperature and relative humidity datasets collected from SensorScope deployments. We have obtained compression ratios up to 70.81% for temperature datasets and up to 62.08% for relative humidity datasets, without introducing any error in the reconstructed signals. Then, we have compared LEC with S-LZW, a lossless compression algorithm specifically designed for WSNs. We have shown that the LEC algorithm can achieve higher compression ratios than S-LZW. Finally, we have discussed how LTC, a lossy compression algorithm proposed for WSNs, has to tolerate a considerable error in the reconstructed samples to achieve compression ratios comparable to the LEC algorithm.

Finally, although, as already discussed in the thesis we believe that lossless compression algorithms play an important role in WSNs, we are conscious that, in general, WSNs are deployed with cheap sensors. In this context, it is necessary to introduce lossy compression algorithm trying to reduce the effect of noisy measurements as much as possible, so as to affect as less as possible the quality of the reconstructed samples. To this aim, we have introduced in the original LEC scheme, a quantization process before encoding, following the well-known scheme of the Differential Pulse Code Modulation scheme. Here the general differences between consecutive samples are quantized before being encoded. The quantization process affects both the compression ratio and the information loss. To generate different combinations of the quantization process parameters corresponding to different optimal trade-offs between com-

pression performance and information loss, we have applied NSGA-II, a popular multi-objective evolutionary algorithm, on a subset of samples collected by the sensor. The user can therefore choose the combination with the most suitable trade-off for the specific application. We tested our lossy compression approach on three datasets collected by real WSNs, obtaining compression ratios up to 93.48% with very low reconstruction errors. In the comparative analisys, we have shown how our approach outperforms LTC, in terms of both compression ratio and complexity.

7.1 Open Issues

7.1.1 Delay

The goal of the in-network algorithms proposed in this thesis was to maximize the node lifetime in habitat applications, by reducing the communication data flow toward the sink by means of aggregation and compression techniques. As highlighted, in-networks approaches can be adopted only if data have not to be transferred in real-time. Some applications (like monitoring of temperature, humidity, seismic activity, etc.) were good scenarios for enabling these techniques. However, as we have discussed in Chapter 2, some critic real applications have a maximum tolerable delay, above which data becomes useless. In these critic scenarios, a performance measure that combines delay and throughput is needed.

open issue: *How can we analyse the trade-off between the benefit of delaying data communication and the threat of having stale information at sink?*

7.1.2 Data correlation

Dense networks provide an enormous quantity of information about the physical process being measured. This information is typically highly correlated in time and space among sensors. This correlation can be exploited to overcome sensor network limitations by designing distributed coding schemes that trade this redundancy. Actually this is the on-fashion trend in the literature.

open issue: *How could we enable a distributed framework in which the computational overload can be leveraged among the nodes and a joint decoding is performed using more expensive (in terms of computational requirements) and more reliable lossless compression together with low-power lossy compression to reconstruct the phenomenon under monitoring?*

7.1.3 Data gathering protocols

In the proposed compression algorithms each node simply samples, compresses and stores measured samples. When a packet payload is full the node forwards it to a data collection center, using generally node to node - multi-hop data propagation. However, an emergent trend in the literature of habitat monitoring with WSNs is the *data mule* architecture. Here a WSN consists of a number of static sensor nodes, placed in several different pre-fixed points in an area under monitoring, and one or more data collectors (data mules) come into contact with the static sensors at approximately regular intervals to collect values measured by them. This model is characterised by a number of advantages in comparison with the traditional approach based on multi-hop communication: the network lifetime is longer (the number of exchanged packets decreases), the packet loss probability decreases (the number of hops decreases), the network capacity increases and node synchronization error decreases (the number of hops is smaller than in the multi-hop approach). On the other hand, the data latency and the costs of the network infrastructure might increase (SKJ+06). Since the model with data mules is typically applied in environmental monitoring applications, latency is not generally an issue. Further, the additional cost for data mules can be maintained very low if we exploit the mobility of external agents available in the environment. In this framework, each static node could trade the power consumption, the compression performances and the introduced error off by adapting its compression parameters during the coding phase and by tuning its radio duty-cycle.

open issue: *How does a data mule architecture influence the compression task in terms of compression performance, reconstructed error and latency?*

References

[ACDP07] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella. How to prolong the lifetime of wireless sensor networks. In M. Denko and L. Yang, editors, *Mobile Ad Hoc and Pervasive Communications*. American Scientific Publishers, Valencia, CA, USA, 2007. to be published. 3, 23

[ACMV06] M. Awenuti, P. Corsini, P. Masci, and A. Vecchio. Increasing the efficiency of preamble sampling protocols for wireless sensor networks. In *MCWC '06: Proceedings of the First International Conference on Mobile Computing and Wireless Communication*, pages 117–122, 2006. 23

[ASSC02] I.F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *Communications Magazine, IEEE*, 40(8):102–114, August 2002. 1

[BA06] K. C. Barr and K. Asanović. Energy-aware lossless data compression. *ACM Trans. Comput. Syst.*, 24(3):250–291, August 2006. 3, 6, 30, 31, 59, 81

[BGS03] A. Boulis, S. Ganeriwal, and M.B. Srivastava. Aggregation in sensor networks: an energy-accuracy trade-off. *Ad Hoc Networks*, 1(2-3):317–331, 2003. 26, 36, 37, 53

[BKK07] S. Bac, D. Kwak, and C. Kim. Traffic-aware MAC protocol using adaptive duty cycle for wireless sensor networks. In *ICOIN '07: International Conference on Information Networking. Towards Ubiquitous Networking and Services*, pages 142–150, 2007. 23

[Cam] Camalie Networks Wireless Sensing for Viticulture. `http://camalienetworks.com/index.htm`. 2

[CAM02] L. Chen, P.O. Arambel, and R.K. Mehra. Estimation under unknown correlation: Covariance intersection revisited. *IEEE Trans. Autom. Control*, pages 1879–1882, 2002. 36

[CBLV04] R. Cristescu, B. Beferull-Lozano, and M. Vetterli. On network correlated data gathering. In *INFOCOM '04: Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 2571–2582, 2004. 29

[CDM04] B. J. Culpepper, L. Dung, and M. Moh. Design and analysis of hybrid indirect transmissions (hit) for data gathering in wireless micro sensor networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 8(1):61–83, 2004. 27

[CE02] J. Chhabra and B. Elliott. Real-world experiences with an interactive ad hoc sensor network. In *Proceedings of the International Conference on Parallel Processing Workshops*, page 143, 2002. 8

[Cia06] A. Ciancio. *Distributed wavelet compression algorithms for wireless sensor networks*. PhD thesis, University of Southern California, 2006. 33, 34

[CMV08] S. Croce, F. Marcelloni, and M. Vecchio. Reducing power consumption in wireless sensor networks using a novel approach to data aggregation. *The Computer Journal*, 51(2):227–239, 2008. 19

[CPOK06] A. Ciancio, S. Pattem, A. Ortega, and B. Krishnamachari. Energy-efficient data representation and routing for wireless sensor networks based on a distributed wavelet compression algorithm. In *IPSN '06: Proceedings of the 5th International Conference on Information Processing in Sensor Networks*, pages 309–316, April 2006. 33, 34

[CSH02] A.P. Chandrakasan, A.C. Smith, and W.B. Heinzelman. An application-specific protocol architecture for wireless microsensor networks. *IEEE Trans. Wireless Commun.*, 1:660–670, 2002. 27

[CTLW05] M. Cardei, M.T. Thai, Yingshu Li, and Weili Wu. Energy-efficient target coverage in wireless sensor networks. In *INFOCOM '05: Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1976–1984, March 2005. 16

[Cut52] C. C. Cutler. Differential quantization of communication signals. Patent, July 1952. 2 605 361. 89

[CV03] R. Cristescu and M. Vetterli. Power efficient gathering of correlated data: optimization, NP-completeness and heuristics. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):31–32, 2003. 29

[Dav] Davis Solar Radiation sensor Homepage. `http://www.davisnet.com/weather/products/weather_product.asp?pnum=06450`. 83

[dCdS06] A. B. da Cunha and D.C. Jr. da Silva. An approach for the reduction of power consumption in sensor nodes of wireless sensor networks: Case analysis of Mica2. In *SAMOS*, pages 132–141, 2006. 3

[DCX03] M. Ding, X. Cheng, and G. Xue. Aggregation tree construction in sensor networks. In *VTC '03: IEEE 58th Vehicular Technology Conference*, volume 4, pages 2168–2172, October 2003. 28

[DJ94] D. L. Donoho and I. M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994. 96

[DNL+06] J. J. Durillo, A. J. Nebro, F. Luna, B. Dorronsoro, and E. Alba. jMetal: A java framework for developing multi-objective optimization metaheuristics. Technical Report ITI-2006-10, E.T.S.I. Informática, Campus de Teatinos, 2006. 97

[DPAM02] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.*, 6(2):182–197, April 2002. 90, 96, 99

[ECG] ECG dataset. http://www.physionet.org/physiobank/database/mitdb/. 84

[Eli75] P. Elias. Universal codeword sets and representations of the integers. *IEEE Trans. Inf. Theory*, 21(2):194–203, 1975. 60

[Fal73] N. Faller. An adaptive system for data compression. In *Proceedings of the 7th Asilomar Conference on Circuits, Systems, and Computers*, pages 593–597, 1973. 69

[Fir] Fire: Fire information and rescue equipment. `http://fire.me.berkeley.edu/`. 2

[Gad06] Y. Gadallah. A comparative study of routing strategies for wireless sensor networks: Are MANET protocols good fit? In *ADHOC-NOW '06: Proceedings of the 5th International Conference on Ad-Hoc, Mobile, and Wireless Networks*, pages 5–18, 2006. 12

[Gal78] R. G. Gallager. Variations on a theme by Huffman. *IEEE Trans. Inf. Theory*, 24(6):668–674, 1978. 69

[GBT+04] C. Guestrin, P. Bodik, R Thibaux, M. Paskin, and S. Madden. Distributed regression: an efficient framework for modeling sensor network data. In *IPSN '04: Proceedings of the Third International Symposium on Information processing in sensor networks*, pages 1–10, 2004. 33

[GDV06] M. Gastpar, P. L. Dragotti, and M. Vetterli. The distributed Karhunen-Loève Transform. *IEEE Trans. Inf. Theory*, 52(12):5177–5196, December 2006. 33, 34

[GEH03] D. Ganesan, D. Estrin, and J. Heidemann. Dimensions: why do we need a new data handling architecture for sensor networks? *SIGCOMM Comput. Commun. Rev.*, 33(1):143–148, 2003. 33, 34

[GN98] R.M. Gray and D.L. Neuhoff. Quantization. *IEEE Trans. Inf. Theory*, 44(6):2325–2383, October 1998. 92

[Gol66] S. W. Golomb. Run-length encodings. *IEEE Trans. Inf. Theory*, 12(3):399–401, 1966. 60

[Goy01] V. Goyal. Theoretical foundations of transform coding. *IEEE Signal Process. Mag.*, 18(5):9–21, September 2001. 34

[HCB00] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8*, page 8020, 2000. 27

[Hil03] J. L. Hill. *System Architecture for Wireless Sensor Networks*. PhD thesis, University of California, Berkeley, 2003. 6

[HKB99] W.R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 174–185, 1999. 25

[HNG94] J. Horn, N. Nafpliotis, and D. E. Goldberg. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, volume 1, pages 82–87, 1994. 96

[HSW+00] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. *SIGPLAN Not.*, 35(11):93–104, 2000. 24

[HT05] C.F. Huang and Y.C. Tseng. The coverage problem in a wireless sensor network. *Mob. Netw. Appl.*, 10(4):519–528, 2005. 16

[Huf52] D.A. Huffman. A method for the construction of minimum-redundancy codes. *in: Proceedings of the IRE*, 40(9):1098–1101, September 1952. 101

[IEGH02] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann. Impact of network density on data aggregation in wireless sensor networks. In *ICDCS '02: Proceedings of the 22nd International Conference on Distributed Computing Systems*, page 457, 2002. 28

[IGE+03] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.*, 11(1):2–16, 2003. 25, 28

[JJRZQG06] X. Jin-Jun, A. Ribeiro, L. Zhi-Quan, and G.B. Giannakis. Distributed compression-estimation using wireless sensor networks. *IEEE Signal Process. Mag.*, 23(4):27–41, July 2006. 33

[KA06] S. Kulkarni and M. Arumugam. SS-TDMA: A self-stabilizing MAC for sensor networks. In S. Phoha and T.C. La Porta, Griffin, editors, *Sensor Network Operations*. Wiley-IEEE Press, 2006. 23

[KC00] J.D. Knowles and D.W. Corne. Approximating the nondominated front using the Pareto Archived Evolution Strategy. *IEEE Trans. Evol. Comput.*, 8(2):149–172, 2000. 96

[KEW02] B. Krishnamachari, D. Estrin, and S. B. Wicker. The impact of data aggregation in wireless sensor networks. In *ICDCSW '02: Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 575–578, 2002. 28

[KKPG98] J. Kymissis, C. Kendall, J. Paradiso, and N. Gershenfeld. Parasitic power harvesting in shoes. In *Second International Symposium on Wearable Computers*, pages 132–139, October 1998. 16

[KL05] N. Kimura and S. Latifi. A survey on data compression in wireless sensor networks. In *ITCC '05: International Conference on Information Technology: Coding and Computing*, volume 2, pages 8–13, April 2005. 6, 31, 59, 81

[Knu85] D. E. Knuth. Dynamic Huffman coding. *Journal of Algorithms*, 6:163–180, 1985. 69

[KWFLS07] K.W. Kai-Wei Fan, S. Liu, and P. Sinha. Structure-free data aggregation in sensor networks. *IEEE Trans. Mobile Comput.*, 6(8):929–942, August 2007. 3, 26, 27

[LHF08] G. Li, J. He, and Y. Fu. A group-based intrusion detection scheme in wireless sensor networks. In *GPC-WORKSHOPS '08: Proceedings of the 2008 The 3rd International Conference on Grid and Pervasive Computing - Workshops*, pages 286–291, 2008. 9, 18

[LKR04] G. Lu, B. Krishnamachari, and C.S. Raghavendra. An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks. In *WMAN '04:Proceedings of the 18th International Workshop on Algorithms for wireless, mobile, ad hoc and sensor networks*, 2004. 23

[LL04] J Li and G.Y. Lazarou. A bit-map-assisted energy-efficient MAC scheme for wireless sensor networks. In *IPSN '04: Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 55–60, 2004. 23

[LR02] S. Lindsey and C.S. Raghavendra. PEGASIS: Power-efficient gathering in sensor information systems. In *Proceedings of IEEE Aerospace Conference*, volume 3, pages 1125–1130, 2002. 27

[LRS01] S. Lindsey, C. Raghavendra, and K.M. Sivalingam. Data gathering in sensor networks using the energy*delay metric. In *IPDPS '01: Proceedings of the 15th International Parallel & Distributed Processing Symposium*, pages 924–935, 2001. 27

[LRS02] S. Lindsey, C. Raghavendra, and K.M. Sivalingam. Data gathering algorithms in sensor networks using energy metrics. *IEEE Trans. Parallel Distrib. Syst.*, 13(9):924–935, September 2002. 27

[LZO] LZO Homepage. `http://www.oberhumer.com/opensource/lzo/`. 31

[MB01] M. J. McGlynn and S.A. Borbash. Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 137–145, 2001. 54

[MCP+02] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97, 2002. 7, 68

[MFC+07] G. Manes, R. Fantacci, F. Chiti, M. Ciabatti, G. Collodi, D. Palma, and A. Manes. Enhanced system design solutions for wireless sensor networks applied to distributed environmental monitoring. In *LCN '07: Proceedings of the 32nd IEEE Conference on Local Computer Networks*, pages 807–814, 2007. 7

[MFHH02] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: a Tiny AGgregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):131–146, 2002. 20, 25, 28

[MFHH05] 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.*, 30(1):122–173, 2005. 26

[Mic94] Z. Michalewicz. *Genetic algorithms + data structures = evolution programs*. Springer-Verlag New York, Inc., New York, NY, USA, second edition, 1994. 99

[Moo65] G. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8):114–117, 1965. 1

[MRFC02] S. Madden, Szewczyk R., M. J. Franklin, and D. Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In *WMCSA '02: Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*, page 49, 2002. 28

[MSW07] J. Magott, P. Skrobanek, and M. Woda. Analysis of timing requirements for intrusion detection system. In *DEPCOS-RELCOMEX '07: Proceedings of the 2nd International Conference on Dependability of Computer Systems*, pages 278–285, 2007. 12

[MV08] F. Marcelloni and M. Vecchio. A simple algorithm for data compression in wireless sensor networks. *IEEE Commun. Lett.*, 12(6):411–413, June 2008. 68

[MWS08] D.J. Malan, M. Welsh, and M.D. Smith. Implementing public-key infrastructure for sensor networks. *ACM Trans. Sen. Netw.*, 4(4):1–23, 2008. 19

[NKC09] K. Na, Y. Kim, and H. Cha. Acoustic sensor network-based parking lot surveillance system. In *EWSN '09: Proceedings of the 6th European Conference on Wireless Sensor Networks*, pages 247–262, 2009. 9

[NKL+07] Y. Nam, T. Kwon, H. Lee, H. Jung, and Y. Choi. Guaranteeing the network lifetime in wireless sensor networks: A mac layer approach. *Comput. Commun.*, 30(13):2532–2545, 2007. 15

[O'N76] J. O'Neal. Differential pulse-code modulation (PCM) with entropy coding. *IEEE Trans. Inf. Theory*, 22(2):169–174, March 1976. 94

[PHC04] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107, 2004. 23, 45, 47

[PK00] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Commun. ACM*, 43(5):51–58, 2000. 22

[PKG08] S. Pattem, B. Krishnamachari, and R. Govindan. The impact of spatial correlation on routing with compression in wireless sensor networks. *ACM Trans. Sen. Netw.*, 4(4):1–33, 2008. 29

[PKR02] S.S. Pradhan, J. Kusuma, and K. Ramchandran. Distributed compression in a dense microsensor network. *IEEE Signal Process. Mag.*, 19(2):51–60, March 2002. 33, 34

[PM92] W.B. Pennebaker and J. L. Mitchell. *JPEG Still Image Data Compression Standard*. Kluwer Academic Publishers, Norwell, MA, USA, second edition, 1992. 60, 61

[PST+02] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D.E. Culler. Spins: security protocols for sensor networks. *Wirel. Netw.*, 8(5):521–534, 2002. 20

[Red] CNN. Redwoods now part of wireless network. `http://www.cnn.com/2003/TECH/science/08/15/coolsc.redwoods/`. 2

[Riv94] R. L. Rivest. The RC5 encryption algorithm. In *Proceedings of the Second International Workshop on Fast Software Encryption*, pages 86–96. Springer-Verlag, 1994. 20

[RM07] D. Rebollo-Monedero. *Quantization and transforms for distributed source coding*. PhD thesis, Stanford University, 2007. 33, 35

[RWR03] S. Roundy, P.K. Wright, and J. Rabaey. A study of low level vibrations as a power source for wireless sensor nodes. *Computer Communications*, 26:1131–1144, 2003. 16

[Sal07] D. Salomon. *Data Compression: The Complete Reference*. Springer-Verlag, London, UK, fourth edition, 2007. 35, 69, 92, 93

[Sca03] A. Scaglione. Routing and data compression in sensor networks: stochastic models for sensor data that guarantee scalability. In *Proceedings of the IEEE International Symposium on Information Theory*, page 174, July 2003. 29

[SD94] N. Srinivas and K. Deb. Multiobjective optimization using Nondominated Sorting in Genetic Algorithms. *IEEE Trans. Evol. Comput.*, 2(3):221–248, 1994. 96, 99

[Sei] Seismic dataset. `http://www-math.bgsu.edu/~zirbel/wavelets/`. 83

[Sena] Sensirion Homepage . `www.sensirion.com`. 65, 66, 71

[Senb] SensorScope deployments Homepage. `http://sensorscope.epfl.ch`. 65, 100

[Senc] Sentilla Homepage. `http://www.sentilla.com/`. 3, 37

[SGO+04] T. Schoellhammer, B. Greenstein, E. Osterweil, M. Wimbrow, and D. Estrin. Lightweight Temporal Compression of microclimate datasets. In *29th Annual IEEE International Conference on Local Computer Networks*, pages 516–524, November 2004. 30, 33, 35, 58, 88, 91, 107

[Sha48] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423/623–656, July/October 1948. 94

[Sim] SimIt-ARM Homepage. `http://simit-arm.sourceforge.net/`. 79, 108

[SKJ+06] A. A. Somasundara, A. Kansal, D. D. Jea, D. Estrin, and M. B. Srivastava. Controllably mobile infrastructure for low energy embedded networks. *IEEE Trans. Mobile Comput.*, 5(8):958–973, August 2006. 114

[SM06] C. M. Sadler and M. Martonosi. Data compression algorithms for energy-constrained devices in delay tolerant networks. In *SenSys '06: Proceedings of the 4th International Conference on Embedded networked sensor systems*, pages 265–278, 2006. 6, 31, 32, 58, 59, 70, 81

[SPC+07] K. Sukun, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon. Health monitoring of civil infrastructures using wireless sensor networks. In *IPSN '07: 6th International Symposium on Information Processing in Sensor Networks*, pages 254–263, April 2007. 2

[SS02] A. Scaglione and S. D. Servetto. On the interdependence of routing and data compression in multi-hop sensor networks. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 140–147, 2002. 29

[SW73] D. Slepian and J. Wolf. Noiseless coding of correlated information sources. *IEEE Trans. Inf. Theory*, 19(4):471–480, July 1973. 34

[T.01] Bo T. On optimal entropy-constrained deadzone quantization. *IEEE Trans. Circuits Syst. Video Technol.*, 11(4):560–563, April 2001. 92

[TCC07] H.W. Tsai, C.P. Chu, and T.S. Chen. Mobile object tracking in wireless sensor networks. *Comput. Commun.*, 30(8):1811–1825, 2007. 11

[Teu78] J. Teuhola. A compression method for clustered bit-vectors. *Inf. Process. Lett.*, 7(6):308–311, 1978. 60

[Tin] TinyNode Homepage. http://www.tinynode.com. 65

[TM03] M. Tubaishat and S. Madria. Sensor networks: an overview. *Potentials, IEEE*, 22(2):20–23, April-May 2003. 1, 17

[Tos] TOSSIM Homepage. `http://www.cs.berkeley.edu/~pal/research/tossim.html`. 49

[TR04] C. Tang and C. S. Raghavendra. Compression techniques for wireless sensor networks. In *Wireless sensor networks*, pages 207–231. Kluwer Academic Publishers, Norwell, MA, USA, 2004. 33, 34

[vHH04] L. van Hoesel and P. Havinga. A lightweight medium access protocol (LMAC) for wireless sensor networks. In *INSS '04: First International Workshop on Networked Sensing Systems*, 2004. 23

[vL03] T. van Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 171–180, 2003. 23

[WBD+06] R. S. Wagner, R. G. Baraniuk, S. Du, D. B. Johnson, and A. Cohen. An architecture for distributed wavelet analysis and processing in sensor networks. In *IPSN '06: Proceedings of the 5th international conference on Information processing in sensor networks*, pages 243–250, 2006. 33, 34

[WDB06] R. Wagner, V. Delouille, and R. Baraniuk. Distributed wavelet denoising for sensor networks. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 373–379, December 2006. 34, 89

[Wel84] T.A. Welch. A technique for high-performance data compression. *IEEE Trans. Comput.*, 17(6):8–19, June 1984. 31

[WTC03] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 14–27, 2003. 8

[WZ76] A. Wyner and J. Ziv. The rate-distortion function for source coding with side information at the decoder. *IEEE Trans. Inf. Theory*, 22(1):1–10, January 1976. 34

[XHE01] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 70–84, 2001. 8

[XLC04] Z. Xiong, A.D. Liveris, and S. Cheng. Distributed source coding for sensor networks. *IEEE Signal Process. Mag.*, 21(5):80–94, September 2004. 33, 34

[YQ05] C. Yunxia and Z. Qing. On the lifetime of wireless sensor networks. *IEEE Communications Letters*, 9(11):976–978, Nov. 2005. 15

[ZC04a] W. Zhang and G. Cao. DCTC: dynamic convoy tree-based collaboration for target tracking in sensor networks. *IEEE Trans. Wireless Commun.*, 3(5):1689–1701, September 2004. 28

[ZC04b] W. Zhang and G. Cao. Optimizing tree reconfiguration for mobile target tracking in sensor networks. In *INFOCOM '04: 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2434–2445, March 2004. 28

[ZDT00] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *IEEE Trans. Evol. Comput.*, 8(2):173–195, 2000. 96

[ZGE02] J. Zhao, R. Govindan, and D. Estrin. Residual energy scans for monitoring wireless sensor networks. In *WCNC '02: IEEE Wireless Communications and Networking Conference*, pages 17–21, 2002. 25

[ZHL04] L. Zhao, X. Hong, and Q. Liang. Energy-efficient self-organization for wireless sensor networks: a fully distributed approach. In *GLOBECOM '04: IEEE Global Telecommunications Conference*, volume 5, pages 2728–2732, December 2004. 27

[ZL77] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Trans. Inf. Theory*, 23:337–343, 1977. 31

[ZLT02] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In K.C. Giannakoglou et al., editors, *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems*, pages 95–100. International Center for Numerical Methods in Engineering (CIMNE), Barcelona, Spain, 2002. 96

[ZT99] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans. Evol. Comput.*, 3(4):257–271, November 1999. 96