POLITECNICO DI MILANO
V Facoltà di Ingegneria
Corso di laurea in Ingegneria Informatica

ADAPTIVE AND QUALITY-DRIVEN APPROACH TO DATA MANAGEMENT FOR WIRELESS SENSOR NETWORKS

Relatore: Prof. Fabio A. SCHREIBER
Correlatore: Ing. Cinzia CAPPIELLO

Tesi di Laurea di:
Federico ROSSINI Matr.740075
Matteo RUGGINENTI Matr.740504

Anno Accademico 2010–2011

Matteo:

Alla mia famiglia, ai miei amici.

Federico:

Alla mia famiglia, ai miei amici, a Rachele.

Abstract

Wireless sensor networks (WSNs) are attracting great interest in a number of application domains concerned with monitoring and control of physical phenomena, as they enable dense and untethered deployments at low cost and with unprecedent flexibility. Nevertheless, due to the sensors size, their associated resources, in terms of memory and energy, are limited. Compression data algorithms address this issue by aggregating similar input values and sending to the base station the value generated by the compression, reducing, in this way, the number of packet transmissions. MAC layer protocols, though, handle the activity period of the devices, affecting the direct proportion between number of sent packets and energy consumption. For this reason, reducing the number of transmissions by itself, results to be insufficient for the energy saving goal. Furthermore, different monitored phenomena call for different network quality requirements. The purpose of this work is, starting from an existing aggregation algorithm, to create a system that, at run-time, adapts a set of parameters, such as MAC layer and aggregation algorithm ones, in order to enhance a set of quality dimensions, according to the network quality requirements.

Acknowledgements

We would like to thank...

Contents

List of Figures

List of Tables

1 Introduction

Wireless sensor networks (WSNs) are attracting great interest in a number of application domains concerned the monitoring and control of physical phenomena, as they enable dense and untethered deployments at low cost and with unprecedented flexibility.

WSNs are distributed systems typically composed of embedded devices, each equipped with a processing unit, a wireless communication interface, as well as sensing and/or actuation equipment [1]. Sensors are equipped with capabilities to store information in memory, process information and communicate with neighbors and with a base station [2].

Such devices cooperatively monitor physical or environmental conditions (such as temperature, pressure or pollution level) of a certain habitat [3].

Initially introduced for military intents (such as military zone surveillance), nowadays WSNs are used in public contexts such as habitat monitoring, health care, automatic systems in houses and offices and traffic control. This has been eased by the rapid advancement of semiconductor manufacturing technology, which enabled electronic devices to get smaller, cheaper, and to require less power for their operation every year.

So far, many applications have been proposed that show the versatility of this technology, and some are already finding their way into the mainstream. Most often, in these scenarios tiny battery-powered devices are used for ease of deployment and increased flexibility [4]. This enables embedding processing and communication within the physical world, providing low-cost, fine-grained interaction with the environment.

However, due to the sensors size, their associated resources are limited. In such a context, the main cause of energy dissipation is the use of the wireless link. Solutions that minimize communication are needed [2]. Aggregation algorithms are approaches that aim to reduce sensors power consumption by aggregating information while they sense, processing the information and eventually sending the processed data [3].

Data compression is one effective method to use limited resources of WSNs. It is assumed that some loss of precision in the compressed version of incoming data can be tolerated if this helps in reducing communication. However, it is important that some data quality requirements are satisfied and the error introduced in the compression is below a specified threshold [2].

For this reason a set of quality metrics is also needed to classify the data and the sending node in order to take future decisions according to quality requirements. In this sense, the system must be quality-aware.

Furthermore, the assessment of the quality of a data source is context dependent, i.e. the notions of "good" or "poor" data cannot be separated from the context in which the data is produced or used [5]. For instance, the data about yearly sales of a product with seasonal variations might be considered quality data by a business analyst assessing the yearly revenue of a product. However, the same data may not be good enough for a warehouse manager who is trying to estimate the orders for next month.

This turns our needed solution into an adaptive system, i.e. a system that adjust its behavior according to the perception of the environment and the system itself. In this way, the system takes decisions accordingly to the context surrounding it.

Our work aims to address the formerly mentioned issues proposing a solution by means of a cross-layer aggregation data algorithm that allows to find a good trade-off among all the data quality metrics (in terms of timeliness, completeness and accuracy) exploiting information given by the context.

The algorithm, working in both the application and the communication layer, guarantees that a satisfying output signal is received by the base station especially thanks to its reasoning capabilities. The base station, in fact, according to its knowledge and the current context is able to take decisions on the current state of the network, such as changing application or communication parameters or turning on/off certain nodes, in order to improve one or more specific dimensions of the data quality.

In Chapter 2 we describe the State of the Art, introducing the basic concepts of WSNs and theirs taxonomy, and compression data algorithms.Then, in Chapter 3, we focus on our aggregation data algorithm, describing its features and functionalities since it will be used on each sensor of the network to process data. Our quality-aware and adaptive approach is fully described in Chapter 4, where we list the quality metrics and the actions that drive the decisions taken by the base station accordingly to the context. In Chapter 5 we first introduce Castalia, the simulator used during the experimentation phase and we describe the environment of our experiments and simulations. Finally, in Chapter 6, we analyze the results of our experiments and we address possible future works.

2
State of the Art

In this chapter we describe the state of art of WSNs. We start by analyzing the sensors and WSNs underlying their limitations in terms of memory and energy consumption. Existing approaches that address these issues, such as compression data algorithms, are then presented and briefly described. We then describe how adopting specific access policies to the communication medium can affect the power consumption too. Data compression, though, leads necessarily to data quality loss, thus a trade-off between energy saving and data quality requirements must be found. For this reason we introduce data quality metrics to assess the data quality according to the requirements. Finally, we present a taxonomy for WSNs to help to classify the different kind of applications and to ease the further analysis of our approach.

2.1 Introduction to Sensors and Wireless Sensor Networks

A Wireless Sensor Network (WSN) is a network where autonomous devices, spatially distributed, use sensors to cooperatively monitor the physical or environmental conditions (such as temperature, pressure or pollution level) of a certain habitat [6].

Each device is equipped with a processing unit, a wireless communication interface, as well as sensors and/or actuators. Sensors are equipped with capabilities to store information in memory, process information and communicate with neighbors and with a base station.

Initially introduced for military intents (such as military zone surveillance), nowadays WSNs are used in public contexts such as habitat monitoring, health care, automatic systems in houses and offices and traffic control. This has been eased by the rapid advancement of semiconductor manufacturing technology, which enabled electronic devices to get smaller, cheaper, requiring less power for their operation every year.

Smaller sizes lead to more limited resources available. In particular, the limitations that have the biggest impact during the design phase are the on-board memory size and the power consumption. Sensors can store only few pieces of data, possibly compressed, for a limited lapse of time. Thus, frequent transfers of data to a device equipped with a larger memory (base station) is needed. On the other hand, battery life-time is limited as well and data transmission is the function requiring the highest energy consumption effort. These two limitations are thus in conflict with each other. It is essential to find a good trade-off between the usage of memory and the amount of consumed energy in order to have a proper amount of free memory for data processing without, on the other hand, causing much power consumption.

Valid solutions to address this problem can be, as described in Section 2.2, data compression algorithms. Furthermore, changing the policies of accessing the communication medium can also affect the power consumption, as shown in Section 2.3

2.2 Data Compression Algorithms

Data compression is one effective method to use the limited resources of a WSN [2]. Assuming that a, possibly small, loss of precision can be tolerated, we can send a compressed version of incoming data, in order to reduce the usage of the communication link. However, compression must be performed wisely, according to some data quality requirements; in this way the error introduced during the compression phase can be below a specified treshold.

Traditionally, data compression approaches focused on saving storage without caring about energy consumption. As a result, the compression ratio is their fundamental metric. However, in the WSNs context, the focus must turn towards energy saving as well. In this way, new design issues arise and have to be addressed [7], calling for new approaches [8].

For this reason, most of the existing data compression algorithms are not suitable in the WSNs field given their size and complexity. However, in literature is possible to find interesting contribution to the data compression studies in the sensor networks context.

For example in [9] the authors address the analysis of high spatial correlation in data from fixed sensors in dense networks. Here, the context is specific and the addressed problems have particular characteristics and criticisms. As described in Chapter 3, the approach we adopted aims at handling data from different type of systems. In other words, we want our approach to be able to adapt and work correctly even in mutable scenarios.

Data compression has been mostly studied to enable in-network processing; with this term we describe those techniques that process data on a node or group of nodes before forwarding it to the user or to the base station. The goal of in-network processing of data streams is to select and give priority to reporting the most relevant data gathered [10].

In this field, two different compression techniques are most widely used: spatial and temporal compression techniques.

Spatial compression deals with data redundancy in a same physical area. Significant contributions in this field propose models that, using specific functions, find similar values in order to aggregate and send them [11]. In [12], similar values compose the base signal used to forecast and evaluate the collected data. In spatial compression analysis, the research contributions on sensors communication paradigms are extremely relevant (e.g., [13]).

Temporal compression is suitable for those contexts where the main goal is to detect data trend changes over time. Considering this scenario, one of the main contribution is a lightweight linear approach [14][15]. The use of linear compression provides a good balance between maximizing compression and minimizing processing complexity for each node. The approach considers the different measure points at different time instants. Each point is compared with the previous one and it is transmitted only if the measure is significantly different. In scenarios where phenomena are quite stable, this approach is very suitable. On the other hand, in case of unstable data, the algorithm would send the measured values one by one, wasting the energy of the sensor.

As we describe in Chapter 3, the data compression algorithm we deploy in each sensor of the network aims at detecting run-time data trend changes as the linear compression algorithms, but it is also able to maximize the compression even when the change in data trend is frequent. This feature makes it suitable for any situation regardless the phenomenon characterization.

Furthermore, the algorithm we adopted is also based on the concept of time series as [16][17][18]. In [16] the authors propose to perform on-line regression analysis over time series on data streams. Autoregressive models built on each sensor are instead used in [17] to forecast time series and approximate the value of sensors readings. Lazaridis and Mehrotra [18] also propose to fit models to time series, but they try to improve system performance, rather than doing regression analysis.

However, data compression is just the first step in the energy saving path, but it is not the only one. Furthermore, data compression leads necessarily to data quality loss. Our approach starts from a data compression algorithm, but adopts other policies and methods to reduce the energy consumption ensuring an acceptable level of data quality. In particular, while all the formerly mentioned approaches are basically sensor-oriented, our proposed approach is network-oriented, since we think data quality depends on the capability of the sensor to provide data with an as little as possible error, as well as on the capability of the network itself to understand the context and to adapt to it. In this way, our algorithm is able to adopt to sudden data trend changes and to unexpected situations, such as faulty sensors, providing a higher level of quality and reliability of data.

2.3 MAC Layer Protocols

The boundaries between programming abstractions and the rest of the software executing on a WSN node is often blurred [1]. The limitations on WSN nodes, in terms of computing and communication resources, along with their application-specific nature, drive necessarily to a *cross-layer* design approach where not only the application layer is involved, but rather the design interweaves the application layer with other layers or system-level services.

An important element of this cross-layer design approach concerns the usage and exploitation of the *Medium Access Control (MAC)* layer. In fact, MAC protocols for WSNs strongly affect the access to the communication media and the energy consumption. Obviously, the goal in the adoption of a MAC protocol rather than another is to guarantee an efficient access to the communication link while carefully managing the energy consumption of the nodes. The former goal is achieved adopting different policies, as latterly described; the latter is typically achieved by switching the communication link (the radio) into a low-power mode according to the transmission schedule.

Most of the existing MAC protocols fall in two categories. *Contention-based* protocols (e.g., [19][20][21]) regulate the access to the physical layer opportunistically, according to the current transmission requests. This class of protocols is surely easier to implement and better tolerates nodes joining or leaving. *Time-slotted* protocols (e.g., [22]), instead, assign the nodes with predefined time-slots to schedule their transmissions over time, providing a higher reliability and greater energy savings.

Typically, in other wireless platforms different from WSNs, MAC functionality is performed at a hardware level. In the WSNs context, instead, MAC protocol is implemented mostly in software, exploiting the low-level language associated with the operating system. For this reason it is important to adopt a cross-layer design approach in order to optimize the overall performance of the algorithm, in terms of energy consumption and radio access.

In our work, we take into consideration this issue and adopt a highly widespread protocol known in literature as the IEEE 802.15.4 standard. Nevertheless, as described in Chapter 4, we made a few changes in the adopted standard in order to better satisfy our quality data needs and requirements giving our application a cross-layer conformation.

2.4 Data quality

As mentioned in the previous section, data compression leads necessarily to data quality loss. Compression, thus, must be performed wisely, according to some data quality requirements.

The algorithm we adopted is able to deal with all types of trends and not only with a limited set as the algorithm proposed by [18]. This feature derives from an adaptive mechanism to change its parameters according to the data stream changes. This adaptation in data stream management is driven by data quality requirements. Several contributions in the literature adopt a similar approach by considering a different set of dimensions; for example, [23] monitors the processing delay to assure data freshness. The total response time is also checked in [24] to optimize the overall QoS performance according to the network condition and work load at run-time.

Furthermore, quality has been often analyzed taking into account costs; tipically, in this field, the most important component of cost is the energy consumed in providing the requested data. In [25] [26] [27], this cost versus quality metrics trade-off has been thoroughly analyzed.

The attitude among the authors of all these mentioned approaches is to discard outliers and try to detect the value trend. In the algorithm we adopted, instead, outliers are important elements of the data stream to consider and store since scientific researchers deem they can be very useful in studying and interpreting natural phenomena.

In addition, we consider other data quality dimensions, such as accuracy and precision, to improve the efficiency of the algorithm. However, these are metrics useful only for the sensor-side quality awareness. We, thus, added other metrics (e.g. completeness) and criteria to satisfy data quality requirements at a network level at run-time, as described in Chapter 4.

2.5 WSN Taxonomy

WSNs are being employed in a variety of scenarios. Such diversity translates into different requirements and, as a result, different programming constructs supporting them. In this section we identify some of the most common and important features of WSN applications that strongly affect the design of programming approaches, introducing the taxonomy suggested in [1]. This description will help to better understand the configuration of our proposed algorithm. Figure 2.1 illustrates the dimensions considered by the authors.

Goal. In most WSN applications the goal is to gather environmental data for later, off-line analysis. A network of sensor-equipped nodes funnels their readings, possibly along multiple hops, to a single base station. Typically the base station is much more powerful (in terms of memory and energy) than a sensing node and acts as a data sink, collecting the network data. Along with *sense-only* scenarios, there is another class of applications where WSN nodes are equipped also with actuators;

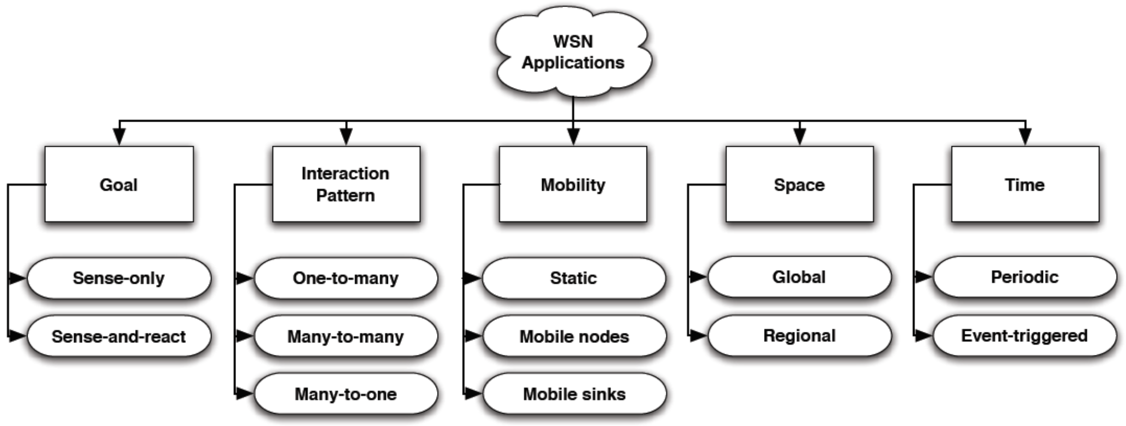


Figure 2.1: *Taxonomy for a WSN application*

in this *sense-and-react* scenario, nodes can react to sensed data, closing the control loop [28].

Interaction pattern. Another key point in WSNs is how the nodes interact with each other within the network, which can be affected also by the application goal they are supposed to to accomplish. In particular, in a sense-only context a *many-to-one* interaction pattern is mostly adopted, where all sensors data is sent to a central collection point. However, *one-to-many* and *many-to-many* interactions can also be found. The former are important when it is necessary to send configuration commands (e.g., a change in the sampling frequency) to the nodes in the network; the latter is typical of scenarios where multiple data sinks are present, as it happens in a sens-and-react context.

Mobility. According to the need of the application, it is possible to deal with a certain degree of dynamism within the network. Mobility may manifest itself in different ways:

– In *static* applications, neither nodes nor sinks are able to move once deployed. This is by far the most common scenario in WSNs.

– Some applications use *mobile nodes* attached to mobile entities such as robots or animals. A typical case is wildlife monitoring where sensors are attached to animals, as in the ZembraNet project [29].

– Some applications, instead, exploit *mobile sinks.* The nodes may or may not move: the key aspect is that data collection is performed opportunistically when the sink moves in proximity of the sensors [30].

Space and time. The phenomena considered in the environment strongly affects either the different portions of the physical *space*, and the instants of time the distributed processing works. Regarding the space dimension we can classify the distributed processing in:

– *Global*, in applications where the processing concerns the entire network, typically when the phenomena of interest span the entire geographical area where the WSN is deployed.

– *Regional*, in applications where the majority of the processing occurs only within some limited area of interest.

Concering the time dimension, instead, the distributed processing can be:

– *Periodic,* in applications that continuously sense and process data.

– *Event-triggered*, in applications mostly characterized by two phases: *i)* during the event detection, the system is mostly quite and the nodes monitor the phenomena with little or no communication involved; *ii)* in case the event condition is satisfied (e.g., a sensed value is above a certain threshold), the distributed processing is triggered.

Introducing the taxonomy helps to describe clearly the basic configuration, widely analyzed in Chapter 4, of our approach, according to the formerly listed dimensions.

In terms of goal, our application is *sense-only.* The base station takes decisions according to the values received and the context affecting only nodes and thus network behaviors. Actuators changing environment conditions are not involved. During the sensing, the nodes *periodically* send their aggregated values to the base station. Thanks to its adaptiveness, the base station is also able to change at run-time a set of parameters (e.g., aggregation parameters) according to the received values or context condition and communicate them to the nodes of the network. In this sense, our approach exploit a combination of *many-to-one* and *one-to-many* interaction patterns. In our algorithm the base station triggers actions according to a set of rules based on events (e.g., a certain data quality metric is not satisfied anymore), giving it an apparent event-triggered configuration. Nevertheless, since the conditions are checked at the base station by periodically collecting data, such application falls in the *periodic* class and not in the event-triggered one. Furthermore, our field of interest is divided into different zones and the base station reasoning is focused differently on each zone. In this sense, in our approach the distributed processing works with *regional* logic. Finally, none of the nodes of the network are able to move. In terms of mobility, thus, our network is *static.*

3 Technologic State of the Art

In our former work (i.e. [3]) we implemented and tested an aggregation data algorithm on a Mica2 mote. The experimentation, though, was limited to a sensing and a processing phase performed by just one sensor, sending periodically aggregated values to the base station.

This simplistic network configuration was due to its particular goal. Our purpose then, in fact, merely concerned the analysis of the efficiency of the proposed approaches, in terms of data quality and energy savings on the single node.

Even though its simplicity helped to fully analyze the algorithm behaviors in different case studies, it does not represent a realistic network scenario. For this reason, the intent of the present work is to expand the approach of [3] to a more realistic network scenario, adding several nodes to the network, all equipped with our aggregation data algorithm. This led us to adopt a MAC layer protocol to improve communications between nodes, and to add also new functionalities to ensure network stability and a high data quality level as described in Chapter 4.

In this chapter we first introduce the reader to the basic features of our starting aggregation data algorithm. We, then, describe the basic features of the MAC layer protocol we adopted, namely IEEE 802.15.4 standard.

3.1 System Quality Requirements

In a wireless sensor network, a set of queries Q is submitted to the base station to get the desired data. The base station then, forwards the queries to the WSN in order to gather sensors data. The quality of the data provided by a sensing application is a combination of different quality metrics. Quality requirements are typically used by the base station to collect only the most relevant data.

In our aggregation data algorithm, the behavior of each sensor is influenced by the following requirements: each sensor node just collects and sends data in order to satisfy all the quality requests [31]. Quality requirements specify constraints in terms of *accuracy* and *precision*. The former is the main variable that identifies the measurement quality and enable the system to detect possible errors or changes in the input data stream trend. Typically, it is defined as the degree of conformity between a measured or processed quantity with its corresponding actual value [2]. Since in a sensing scenario it is impossible to know a priori the real value, our algorithm compares the sensed value with a value of reference to calculate its accuracy. This value is set at design time and represents the expected value to be measured. Obviously, the phenomena of interest may change in time; for this reason, the algorithm changes the value of reference at run-time in order to adapt to the context under exam. Accuracy is thus measured using the sensed value and this value of reference as follows:

$$Accuracy = abs(sensedValue - vRef)$$

The algorithm takes decisions according to the value of accuracy. A value is considered accurate if it respects a accuracy threshold called aTolerance as shown in Program 3.1.

The latter is the degree to which further measurement show the same or similar results [32]. How these quality dimensions impact on the data stream management is

Program 3.1 Accuracy assessment

```
int checkAccuracy(double val)
{
     if(fabs(sensedValue - Vref) < aTolerance)
          return 1;
     else
          return 0;
}
```

discussed in the next sections of this chapter.

3.2 System Architecture and Data Structure

The input data stream can be seen as a continuous flow of real-time data tuples of the form $< sensor\text{-}id, timestamp, value >$ coming to the sensor's input buffer. As in many real-time systems, we can imagine the Input Buffer (IB) split into two different storage areas (i.e., IB_1 and IB_2). Initially, data are stored into IB_1 until it is full, then input data are fed to IB_2. Meanwhile data in the first buffer are transferred to the compression engine to be sent to the output buffer as shown in Figure 3.1. Analogously, when the compression is performed, data stored in IB_2 are transferred to the compression engine, and incoming data are fed to IB_1.

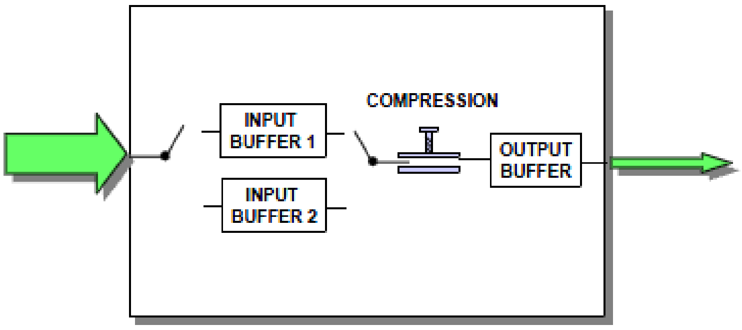


Figure 3.1: *System architecture*

In this sense, the algorithm uses a *windowing* approach. In fact, the potentially infinite data stream is reduced to a sequence of finite time-ordered data sets on each of which the compression algorithm can work [2]. Furthermore, the main window can be partitioned itself into smaller sub-windows (see Figure 3.2) ; values in this kind of sub-windows are considered so *similar* that can be aggregated by computing their average.

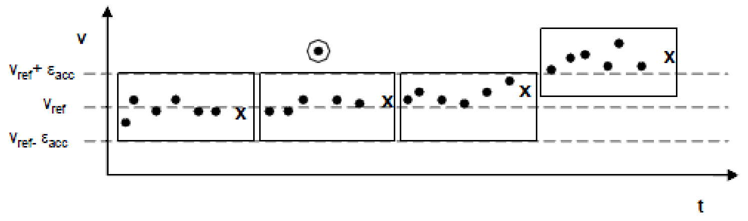


Figure 3.2: *Windowing on data stream*

The algorithm reduces the communications by sending the base station only the average values and the *outliers*, i.e., those values that depart from the expected trend and could stand for either a measurement error or a sudden change in the measured phenomena. . The former are sent together with the timestamps of the first and the last value which took part to the aggregation; the latter are associated with the actual timestamp in which they are sensed by the node. Dealing with the timestamps is a fundamental aspect of this approach since it allows to re-build the incoming signal once the base station received the aggregated data.

The sub-windows shown in Figure 3.2 may change in terms of size according to the *similarity* of the values involved in the data stream. The similarity concept is defined by considering the two formerly mentioned data quality dimensions of *accuracy* and *precision.* Accuracy can be identified with the window's height; in particular, the window height influences the accuracy of the measure and the robustness in finding *outlier values.* To understand whether the value falls into one case or the other we need to measure its precision. Indeed, if the values are not accurate (outside the window height), but precise, it means that they are not similar to the reference value, but they are characterized by a small standard deviation. This means that a change in the system under control has occurred.

Another possible scenario occurs when data values are not accurate nor precise; this happens when the trend in the measured phenomena is very irregular. On the other hand, a measurement error is an occasional and isolated event and all the other values are still considered accurate and precise. The different cases are all formally described in Section 3.3 .

As the sub-window height do, also the sub-window width plays an important role in this approach. It, indeed, affects the compression amount: the larger the number of points in the window, the larger the compression we get. On the other hand, a window with a single point corresponds to no compression whatsoever. It is important to underly, though, that in case the sampling interval is very low (high data rate), enlarging the sub-window width can decrease the overall network traffic. In fact, if the phenomena under exam is stable enough, the larger the window, the the more aggregation and thus the less packet sent. As illustrated in Chapter 4 this factor results to be very important to control the network traffic and preserve the network quality requirements.

3.3 The Model

In this section, we focus the description on the interaction between a single data producer (sensor) and a data archiver (database located in the base station). Our aim is to describe the sensor's behavior rather than the data processing and reasoning performed by the base station, which are, instead, described in Chapter 4.

3.3.1 Preliminaries

Each sensor has a sampling frequency that affects the time instant t_i in which data are acquired. Considering these time instants, we define a value series

$$V =< v[1], v[2], \ldots, v[n] >$$

as a set of values observed in subsequent n time instants. The maximum number of measure points N represents the cardinality of data in the input buffer (window width). Sub-windows are characterized by their width W and height H. The former coincides with the number of points in a sub-window and affects the compression factor according to the data trend variability. The latter represents the biggest difference between two measure points in the same window and sets the accuracy tolerance affecting the robustness in finding outlier values.

The quality-oriented approximation starts from the definition of the requirements about accuracy and precision, which can be set by the user. In continuous value monitoring, accuracy and precision can be used to detect errors or changes in the monitored phenomena. Precision is typically assessed by mean of the standard deviation of the measured values. The smaller the standard deviation, the higher the precision. On the other hand, accuracy can be defined as the error expressed by the difference between the measured value and a reference value (i.e., v_{ref} in Figure 3.2).

As mentioned in Section 3.2, different combination of accuracy and precision distinguish different scenarios in the input data trend, as shown in Figure 3.3:

1. ***Expected trend***: values are either precise and accurate; as a consequence the trend is regular;
2. ***Slow Change***: the trend is characterized by an unexpected, but not permanent and small variation. Values are still precise but not accurate anymore;
3. ***Fast Change***: similar to the former; in this case,though, the trend is characterized by an unexpected, but lasting and large variation in a way that precision requirements are satisfied again sometimes after the instant at which the variation occurred;
4. ***Irregular/Bursty trend***: the trend is characterized by unexpected and discontinuous variations. Values are not precise but they can be both accurate or inaccurate.

Note that any data stream can be represented as the combination of the described trends with possible outlier values. Outliers are identified in all cases in which both accuracy and precision requirements are not satisfied, but the overall trend has not changed (in Figure 3.2, the dot inside the circle). Outliers might hide significant information and should not be discarded. In fact, analyzing historical series, outliers sometimes reveal periodical, and thus systematic, irregularities that can not be inferred with a local and limited analysis.

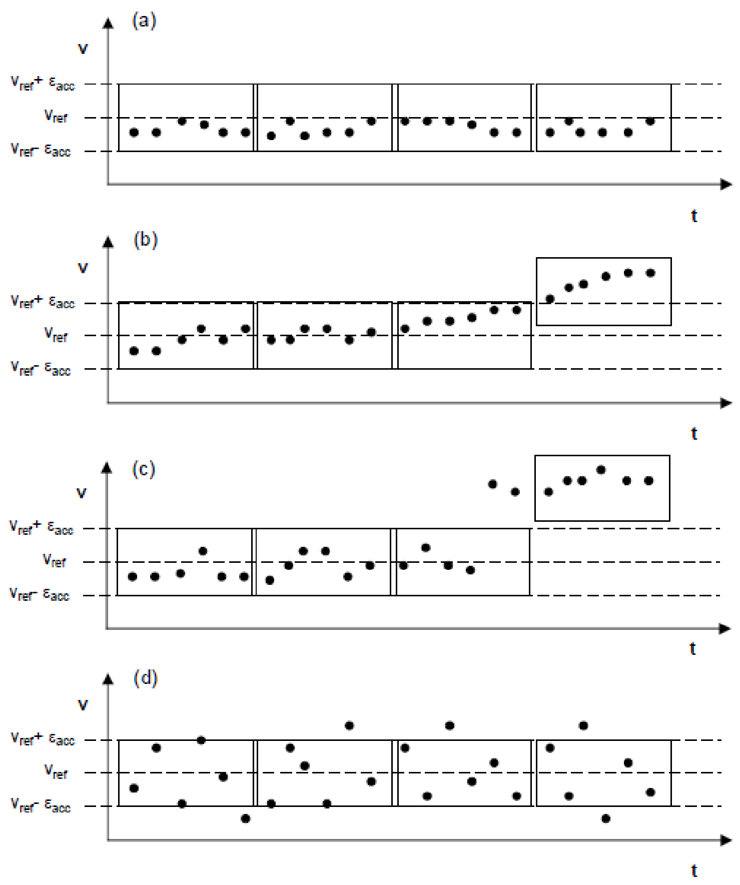


Figure 3.3: *a) Expected trend, b) Slow Change, c) Fast Change and d) Irregular trend*

The sensor has a finite energy supply which is consumed during normal sensing operation at some rate. Additional energy drains are caused when using any of the sensor equipment, namely (i) powering its memory, (ii) using its CPU, (iii) sending/receiving data. The rates of energy consumption for these operations are sensor-specific. Typically, communication is the major cause of energy drain in sensors and hence, since our interest is to extend sensor's life, communication must be limited. For this reason, each sensor is characterized by:

$\mathbf{e}_t$: energy consumption for the transmission of one byte;

$\mathbf{e}_e$: energy consumption for processing one instruction;

$\mathbf{E}_t$: energy consumption for data transmission to the base station;

$\mathbf{E}_e$: energy consumption for processing analysis and aggregation algorithms;

$\mathbf{E}_{tot}$: total energy consumption of the sensor, calculated as $\mathrm{E}_t + \mathrm{E}_e$.

Our approach aims at minimizing the total energy consumption assuming that $\mathrm{e}_t \gg \mathrm{e}_e$. In fact, the algorithm processes the input data stream in order to transmit to the base station only the aggregated values and the possible outliers. Assuming Z to be the set of aggregated values, J the set of the outlier values and N the total number of values in the input data stream; the algorithm is considered efficient if the output is composed by (Z+J) values instead of N where $(\mathrm{Z}+\mathrm{J}) \ll \mathrm{N}$.

3.3.2 The Algorithm

In this section we detailedly describe our approach, showing its feasibility and adaptivity.

Setting the input parameters

The main input to be fed to the algorithm concerns the quality requirements. In our approach, as stated in Section 3.2, quality requirements are represented by the value accuracy and precision. Formally, we define accuracy as the difference between the measured value v_n and the reference value v_{ref}, which represents the bias [32]. Assuming

ε_{acc} to be the tolerable error in a measured value, the measure is considered accurate if:

$$|v_n - v_{ref}| < \varepsilon_{acc}$$

Note that defining ε_{acc} corresponds to defining the height H of the sub-window since $H = 2 \cdot \varepsilon_{acc}$.

On the other hand, precision can be defined as the reciprocal of variance. In this way, a measure is considered precise if:

$$1/N \cdot \sum_{n=1}^{N} (v_n - \mu)^2 < \varepsilon_{prec}$$

where μ is the average calculated among all the v_n values.

Other parameters needing to be set are the maximum number of points to consider in a sub-window W and the minimum number of outliers contained in a window L. The latter defines the threshold number of outliers in a single window after which the trend is considered to be irregular. In this case, the algorithm halts the aggregation function and sends the values one by one. Finally, it is necessary to specify the number of observation C needed to evaluate if the irregularities in a trend are only isolated events or are part of a permanent change in the trend.

Algorithm body

The input parameters and the outputs of the algorithm are listed in Table 3.1.

As mentioned in Section 3.2, the algorithm is based on the evaluation of the similarity of the incoming data with the previous data values acquired in the sensing activity. Similarity degree depends on the precision and accuracy assessment as described in Table 3.2.

The similarity assessment enables the algorithm to classify the current trend characterizing the input data and, thus, take the corresponding sequence of actions.

	Name	**Symbol**
INPUT	Time Series	V=$< v_1, v_2, \ldots, v_n >$
	Expected Value	v_{ref}
	Accuracy Tolerance	ε_{acc}
	Precision Tolerance	ε_{prec}
	Window Width	W
	Continuity Interval	C
	Minimum Outlier Number	L
OUTPUT	Aggregate values	T=$< t_1, t_2, \ldots, t_z >$
	Outliers	O=$< o_1, o_2, \ldots, o_j >$

Table 3.1: *Algorithm input and output*

The aggregation algorithm can be divide into three main phases:

1. Check possible values from previous windows;
2. Processing all the data in the buffer one by one;
3. Post-processing checks.

The first phase allows the system to correctly handle possible trend changes occurring among two different windows. The second phase can be considered the actual core of the algorithm. It, indeed, identifies and classifies the values in the current window according to their similarity value.

Accuracy	**Precision**	**Similarity**	**Trend**
Accurate	Precise	High	Expected Trend
Not Accurate	Precise	Medium	Slow Change
Accurate	Not Precise	Low	Irregular Trend
Not Accurate	Not Precise	None	Irregular Trend

Table 3.2: *Similarity Assessment*

We now illustrates the main algorithm steps in pseudo-code:

```
(1) Indata[w] = $v_w$
```

Check accuracy and precision and evaluate the different cases. The algorithm analyses the different data streams trends described in Figure 3.3.

```
(2) SWITCH($\varepsilon_{acc}$, $\varepsilon_{prec}$)
```

CASE 1: Data follow the expected trend. If we have analysed all the measure points in the buffer, the average of the acceptable values is calculated. The acceptable values are defined as all the stored values except for outliers. If the number of outliers is more than the threshold L then all the values will be transmitted without any further analysis.

```
(3)  CASE ($< \varepsilon_{acc}$, $< \varepsilon_{prec}$)
(4)      IF number of analysed values=W AND Number of outliers $< L$ THEN t[z]=AVG(Acceptable
         values); Increment z
(5)      ELSE IF Number of outliers $> L$
(6)          THEN T= $\langle t_1, t_2, ...t_z \rangle$= V= $\langle v_1, v_2, ...v_w \rangle$
(7)          ELSE Analyse new value and GO TO (1)
```

CASE 2: Data undergo a slow change. In this case when the algorithm detects an outlier, it controls if it is associated with a permanent or transient data trend change. In the former case, it calculates the average of the values stored before the exception and recalculates the expected value v_{ref} along the last inaccurate values. In the latter case, inaccurate values are transmitted to the base station as outliers.

```
(8)  CASE ($> \varepsilon_{acc}$, $< \varepsilon_{prec}$)
(9)  Variable initializations: the number of unexpected values, the time instant in which the
     exception occurs (T$_e$)
(10) O[j]=$v_w$
(11) Indata[w]= $v_{w+1}$
(12) WHILE accuracy $> \varepsilon_{acc}$ AND precision $< \varepsilon_{prec}$
```

```
(13)    INCREMENT the number of unexpected value and the number of outliers
(14)    O[j]=$v_w$
(15)    IF number of analysed values=W AND number of subsequent unexpected values = C
(16)   THEN t[z]=AVG(Acceptable values arrived before T$_e$); Increment z ;
(17)      t[z]=AVG(Acceptable values arrived after T$_e$); Increment z;
(18)       $v_{ref}$=AVG(Acceptable values arrived after T$_e$)
(19)   ELSE IF number of analysed values=W AND number of subsequent unexpected values<C
(20)      THEN t[z]=AVG(Acceptable values); Increment z;
(21)      ELSE IF number of subsequent unexpected values = C
(22)         THEN t[z]=AVG(Acceptable values arrived before T$_e$); Increment z;
(23)         $v_{ref}$=AVG(Acceptable values arrived after T$_e$)
(24)          Delete outliers from O[j] to O[j-C]
(25)          GO TO 1
(26)         else Indata[w]=$v_{w+1}$ and GO TO 1
```

CASE 3: Data are characterized by an oscillatory trend or bursts. The analysed value is classified as an outlier.

```
(27) CASE ($< \varepsilon_{acc}$, $> \varepsilon_{prec}$) OR ($> \varepsilon_{acc}$, $> \varepsilon_{prec}$)
(28) O[j]=v$_w$
(28)    IF number of analysed values=N AND Number of outliers $< L$ THEN t[z]=AVG(Acceptable
        values); Increment z;
(29)    ELSE IF Number of outliers $> L$
(30)       THEN T= $\langle t_1, t_2, ...t_z\rangle$= V= $\langle v_1, v_2, ...v_w\rangle$
(31)       ELSE Analyse new value and GO TO (1)
```

Energy Consumption

Given the nomenclature described in Section 3.3.1, the proposed algorithm can be considered efficient if

$$e_t \cdot N > E_c + e_t \cdot Z + e_t \cdot J$$

Note that, if the trend is particularly irregular, most of the values will be classified as outliers; therefore, if the number of outliers exceeds the threshold L, the algorithm is bypassed and all the values are transmitted one by one, saving, in this way, the additional computation energy consumption due to the aggregation process.

Cost-quality trade-offs

The outputs of the algorithm, in terms of aggregated data transmitted to the base station, and its relative energy consumption are mainly affected by the input parameter settings.

Changing in the precision, accuracy or continuity interval values strongly influences the output quality. In particular, high precision and accuracy requirements lead to higher number of values transmitted to the base station. Therefore, the higher the quality level required the higher the energy consumed. For this reason, quality requirements have to be properly designed according to the analyzed context. For instance, stable and not critical phenomena do not need high quality level, increasing energy saving benefits. On the other hand, irregular and critical contexts require high quality results since even the smallest system change should be detected and transmitted.

For this reason, the choice of the parameters values must be made wisely, on the basis of a previous knowledge of the application and its environmental deployment.

3.4 IEEE 802.15.4 MAC Protocol: a General Description

Providing *quality of service* (QoS) support is a challenging issue due to highly resource constrained nature of sensor nodes, unreliable wireless links and harsh operation environments.

Our previous work [3] intent concerned merely the analysis of the efficiency of the proposed approach, in terms of data quality and energy savings, only on a single node. In the present work, instead, we treat a more realistic scenario, focusing our attention at the whole network level. For this reason, it is necessary the adoption of different strategies and policies. In particular, we now focus on the QoS support at the MAC layer which forms the basis of the communication stack and has the ability to tune key QoS-specific parameters, such as the duty cycle of sensor devices.

MAC layer possesses a particular importance among them since it rules the sharing of the medium and all other upper layer protocols are bound to that. QoS support in the network or transport layers cannot be provided without the assumption of a MAC protocol which solves the problems of medium sharing and supports reliable communication [33].

The MAC layer protocol adopted by our approach is the one defined by the IEEE 802.15.4 standard. The standard, which is used as a basis for the ZigBee, WirelessHART, and MiWi specifications, has been originally designed for low-data-rate WPANs. IEEE Std 802.15.4-2003, indeed, defined the protocol and compatible interconnection for data communication devices using low-data-rate, low-power, and low-complexity short-range radio frequency (RF) transmissions in a wireless personal area network (WPAN) [34].

As stated in Section 2.3, our algorithm adopts a cross-layer approach, dealing with both the application and MAC layer. For this reason, we now illustrate the main features of the standard, in order to provide the reader with some basic knowledge necessary to understand how the MAC layer behaves. In Section 4.3, instead, we show how changes in the protocol parameters can affect the overall data quality, in terms of the formerly described metrics.

3.4.1 Introduction

An LR-WPAN is a simple, low-cost communication network that allows wireless connectivity in applications with limited power and relaxed throughput requirements. The main objectives of an LR-WPA are ease of installation, reliable data transfer, short-range operation, extremely low cost, and a reasonable battery life, while maintaining a simple and flexible protocol.

Some of the characteristics of an LR-WPAN are as follows:

– Over-the-air data rates of 250 kb/s, 100kb/s, 40 kb/s, and 20 kb/s

– Star or peer-to-peer operation

– Allocated 16-bit short or 64-bit extended addresses

– Optional allocation of guaranteed time slots (GTSs)

– Carrier sense multiple access with collision avoidance (CSMA-CA) channel access

– Fully acknowledged protocol for transfer reliability

– Low power consumption

3.4.2 Network Topologies

Depending on the application requirements, an IEEE 802.15.4 LR-WPAN may operate in either of two topologies: the star topology or the peer-to-peer topology (see Figure 3.4). In the star topology (he one adopted by our approach), the communication is established between devices and a single central controller, called the PAN coordinator. A PAN coordinator may also run a specific application, but it can be used to initiate, terminate, or route communication around the network. The peer-to-peer topology also has a PAN coordinator; however, it differs from the star topology in that any device may communicate with any other device as long as they are in range of one another. Peer-to-peer topology allows more complex network conformations to be implemented, such as mesh networking topology.

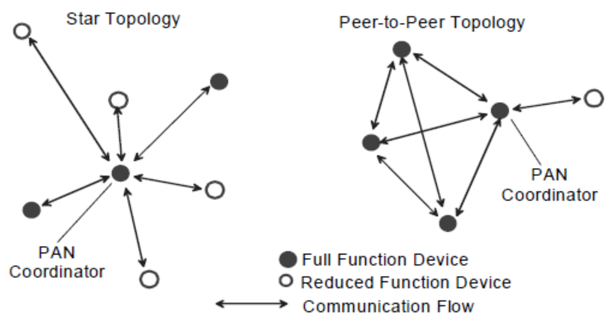


Figure 3.4: *Star and peer-to-peer topology examples*

3.4.3 Functional Overview

A brief overview of the general functions enabled by the usage of the standard is given in the next sections to make the reader confident with the different parameters we take into consideration in Section 4.3.

Superframe structure

This standard allows the optional use of a superframe structure. The format of the superframe is defined by the coordinator. The superframe is bounded by network beacons sent by the coordinator (see Figure 3.5a) and is divided into 16 equally sized slots. Optionally, the superframe can have an active and an inactive portion (see Figure 3.5b). During the inactive portion, the coordinator may enter a low-power mode. The beacon frame is transmitted in the first slot of each superframe. The beacons are used to synchronize the attached devices, to identify the PAN, and to describe the structure of the superframes. Any device wishing to communicate during the contention access period (CAP) between two beacons competes with other devices using a slotted CSMA-CA mechanism. All transactions are completed by the time of the next network beacon.

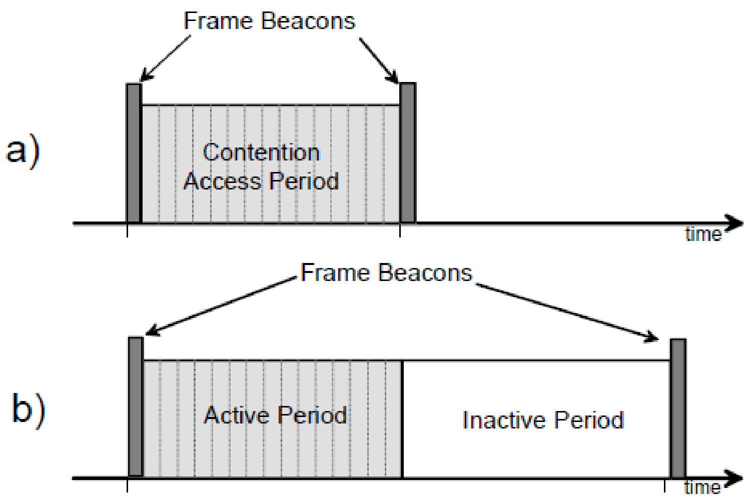


Figure 3.5: *Superframe structure without guaranteed time slots (GTSs)*

Specifically, the structure of this superframe is described by the values of *macBeaconOrder* and *macSuperframeOrder*. The MAC attribute *macBeaconOrder*, describes the interval at which the coordinator shall transmit its beacon frames. The value of *macBeaconOrder*, *BO*, and the beacon interval, *BI*, are related as follows: for $0 \leq BO \leq 14$, $BI = aBaseSuperframeDuration \cdot 2^{BO}$ symbols.

The MAC attribute *macSuperframeOrder* describes the length of the active portion of the superframe, which includes the beacon frame. The value of *macSuperframeOrder*, *SO* (in our work also referred to as *FO*), and the superframe duration, *SD*, are related as follows: for $0 \leq SO \leq BO \leq 14$, $SD = aBaseSuperframeDuration \cdot 2^{SO}$ symbols.

The active portion of each superframe shall be divided into *aNumSuperframeSlots* equally spaced slots of duration $2^{SO} \cdot aBaseSlotDuration$ and is composed of three parts: a beacon, a CAP and a contention-free period (CFP). The beacon shall be transmitted, without the use of CSMA, at the start of slot 0, and the CAP shall commence immediately following the beacon. The start of slot 0 is defined as the point at which the first symbol of the beacon is transmitted. The CFP, if present, follows immediately after the

CAP and extends to the end of the active portion of the superframe. Any allocated guaranteed time slot (GTS) shall be located within the CFP.

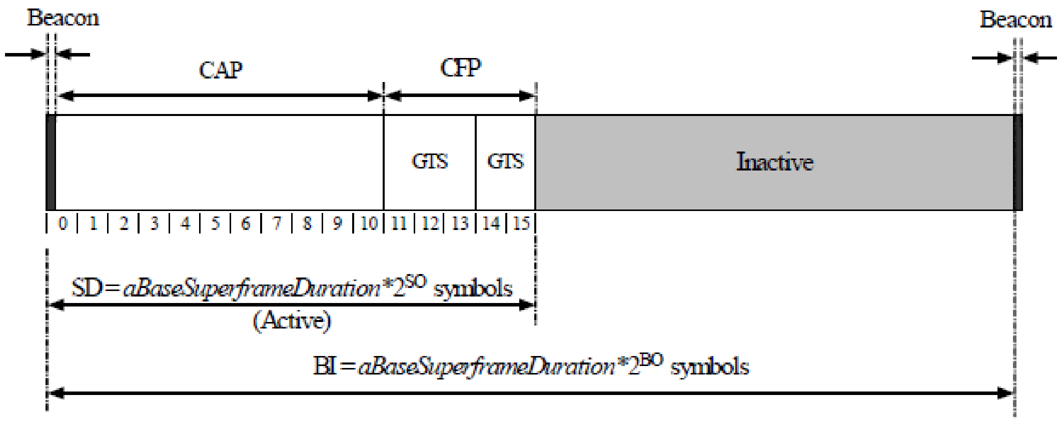

Beacon
Beacon
CAP
CFP
GTS
GTS
Inactive
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
SD=aBaseSuperframeDuration*2^SO symbols
(Active)
BI=aBaseSuperframeDuration*2^BO symbols

Figure 3.6: *An example of the superframe structure*

Figure 3.6 shows an example of a superframe structure. In this case, the beacon interval, BI, is twice as long as the active superframe duration, SD, and the CFP contains two GTSs.

Furthermore, we made a change in the MAC protocol behavior: when GTSs are used, if the MAC layer output buffer is filled for less than half of its size, the contention access period (CAP) will be disabled. In this way, the active period is formed by the only contention-free period (CFP); in other words, the sensor results to be active only during its time slot instead of the entire active period, resulting in a remarkable energy saving (see Section 4.3.2).

Power consumption considerations

As previously mentioned in Section 2.3, MAC protocols, affecting the access to the communication media have considerable effects on the energy consumption of each sensor.

In our previous work ([3]), we could assume that the power consumption, mostly caused by the transmission phase, was directly proportional to the amount of data transmitted. This happened because in our first approach we did not adopt any kind of MAC layer protocol, turning on and off the radio whenever a piece of data was ready to be sent.

With the current approach, instead, the active period of the radio is dictated by the parameters of the MAC layer. In this way, the energy consumption is mainly affected by the duration of the active period rather than the number of sent packets (as shown in Figure 4.7 in Section 4.3.2).

The present work results to be a cross-layer application because, given the effects of the MAC layer on the energy consumption (and on the other data quality metrics, as shown in Section 4.3), we make the base station able to change MAC layer parameters at run-time in order to adapt to the context.

4 Methodology: Quality- and Context-aware algorithm

In this chapter we describe in detail the functionalities of our approach. We start by listing our data quality requirements defining quality metrics for the output. Furthermore, we illustrate some additional functionalities the network can adopt to enhance the data quality and the overall network reliability. Finally we analyze the adaptiveness of our approach. For this purpose, we present a study that enables us to understand how changing MAC and algorithm parameters affects the overall data quality in terms of quality metrics. From the results of this study, we introduce the actions the base stations adopt at run-time to autonomously adapt to the context.

4.1 Network Quality Requirements

As formerly stated, the main aim of our algorithm is to reduce the energy consumption preserving, at the same time, outgoing data quality in terms of pre-established quality metrics. Since our approach is adaptive, it is fundamental to control these metrics at run-time, in order to take decisions in case values do not conform certain threshold.

Quality is assessed in terms of accuracy and precision dimensions in the aggregation data algorithm. Nevertheless, in this work we aim to provides a high quality data level to the entire network. For this reason, we refer to the following quality metrics: *timeliness*, *completeness* and *energy consumption*. It is necessary, thus, to find a good trade-off among these dimensions according to the power consumption they imply.

Timeliness is the property of information to arrive early or at the right time. Typically it measured as a function of two elementary variables: currency and volatility [2]:

$$Timeliness = max(1 - Currency/Volatility; 0)^s$$

where the exponent s is a parameter necessary to control the sensitivity of timeliness to the currency-volatility ratio. In our context, *currency* can be defined as the interval from the time the value was sampled to the time instant at which the base station receives the data. On the other hand, *volatility* is a static information that indicates the amount of time units (e.g., seconds) during which data is considered valid. Volatility is usually associated with the type of phenomena that the system has to monitor and depends on the change frequency. Timeliness constraints are one of the main drivers for data processing and transmission.

Completeness represents the amount of packets received in a defined window over a predefined window size (expressed in number of packets) [35].

$$Completeness(s(t)) = \frac{s_W(t)}{W}$$

where $s_W(t)$ is the amount of data received in the window W at time t, and W the window size, in terms of packets. In this sense, the completeness of the a window is 1 if no packet is lost and is 0 if all the packets of the window are lost.

Energy consumption is a fundamental dimension in the network quality assessment. Theoretically, the base station cannot know the amount of energy consumed by each sensor at run-time, unless it is not explicitly communicated by the sensor. For the sake of simplicity, we made the base station able to know the energy level of each sensor without communicating it, exploiting the functionalities of the simulator we adopted (widely described in Section 5). In this way, the base station is aware of the overall power consumption and can take decision accordingly.

The adaption of different strategies can enhance one dimension or another. The choice of a MAC protocol rather than another, for instance, is a strategy itself and strongly affects all the former quality dimensions. In particular, in Section 3.4 we introduce the MAC protocol we adopted for our approach and in Section 4.3 we illustrate how changes in its parameters affect each dimension.

Besides the adoption of a MAC protocol, other functionalities can also affect the quality of the network in terms of quality metrics and reliability; these functionalities, such as data acknowledgment and faulty sensor detection, are fully described in Section 4.2.

4.2 Network Data Quality and Reliability

WSNs are being employed in a variety of contexts. Typically, such diversity translates into different requirements and, in turn, different programming constructs supporting them. The intent of our work is to give an approach able either to adapt to the current context and to offer functionalities that the user can select at design time according to the system requirements. These latter functionalities can also be introduced at run-time in case the base station reckons one of them is needed to improve the network reliability or quality, in terms of quality metrics.

4.2.1 Fault Detection: Redundant information

Data quality dimensions are not enough to assess the overall quality of a network by themselves. For instance, let us assume a field divided into different zones. If monitoring the temperature of a zone the values of a certain sensor start plausibly drift to a different value of reference the base station would still consider them as good values. In this case, though, if the drift was driven by some kind of event affecting only that sensor, the information about that zone would be corrupted and thus wrong.

Thus, the objective of the context system is to classify each sensor to the corresponding level of reliability, comparing, when needed, their values to values produced by other sensors in the same zone.

The Model

Assuming the monitored field divided into i zones, we can see each zone as a set of sensors $Z_i =< s_{i,1}, s_{i,2}, \ldots, s_{i,n} >$, where n is the total number of sensors deployed in zone i. Figure 4.1 illustrates an example with a field divided into 4 zones with 3 sensors in each zone. During the start up, the base station chooses a leader for each zone (green sensor in the figure); the zone leaders are the sensors entitled to sense, aggregate and send data to the base station.

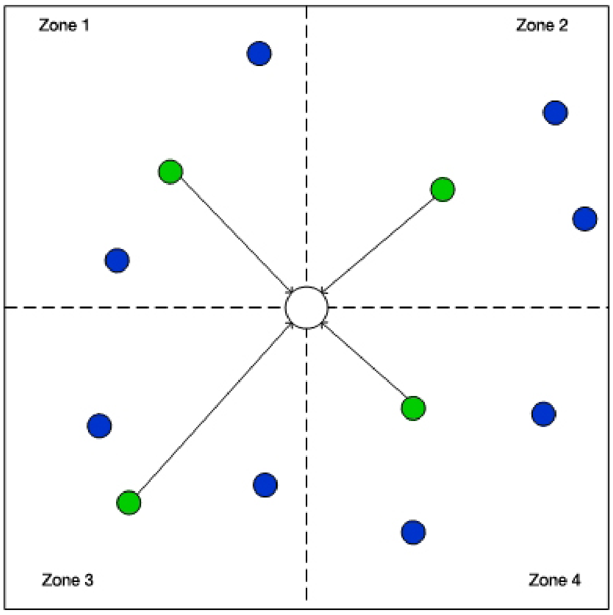


Figure 4.1: *An example of network configuration*

Furthermore, the zone leaders send the aggregate data to the other sensors in the same zone (blue sensor in the figure). When the other sensors detect a change in the trend, a *redundant information* approach is used: the two sensors start sending their own values to the base station to compare their values to the leader's ones for a previously established elapse of time in order to make the base station able to take a decision. The base station takes decision also according to the current *status* of the sensors. We defined different possible status as shown in Table 4.1

STATUS	**Meaning**
LEADER	The node is sensing and is the leader of the zone.
ON	The node is entitled to sense and send data to the base station to control the leader's values.
OFF	The node is not sensing nor sending data to the base station.
UNRELIABLE	The node has been labeled as unreliable, its values may be wrong.
FAULTY	The node is broken and must be repaired or substituted.

Table 4.1: *Status each sensor can assume*

We now illustrates the main steps in pseudo-code:

a non-leader sensor (status ON) receives a value from leader sensor j in zone i. (1) sensorBuffer[k] $= v_{s_{i,j}}$

The sensor then compares the leader's value to its own.

```
(2)    IF abs(sensorBuffer[k] − sensedValue)< Control_Tolerance)
(3)    THEN GO TO (1)
(4)    ELSE
(5)        send(sensedValue, BaseStation)
(6)        GO TO (1)
```

On the other hand, when the base station receives a packet from a

the base station receives a value from sensor j (with status ON) and one from sensor k (status LEADER) in zone i.

The base station compare the sensor values to the zone leader ones.

```
(1)   baseStationBuffer[k,i] $= v_{s_{i,k}}$
(2)   baseStationBuffer[j,i] $= v_{s_{i,j}}$
(3)   IF abs (baseStationBuffer[k,i] $-$ baseStationBuffer[j,i])$<$ Control_Tolerance)
(4)   THEN GO TO (3)
(5)   ELSE
(6)       start_comparison()
```

At this point the algorithm checks the similarity among sensors values two by two increasing a counter each time the sensors values are similar.

```
(7)       IF sensor leader has maximum similarity
(8)       THEN GO TO (1)
(9)       ELSE IF sensors ON have maximum similarity
```

The base station has to change the status of $s_{i,j}$ into UNRELIABLE and promote to leader one of the two sensors ON.

```
(10)          status($Z_{i,j}$, UNRELIABLE)
(11)          status($Z_{i,k}$, LEADER)
```

Obviously, this approach needs a configuration with at least three sensors per zone.

An Example

Let us consider a single zone with four sensors: Sensor 1, Sensor 2, Sensor 3 and Sensor 13. During the start-up phase, the base station randomly sets Sensor 1 as zone leader. Figure 4.2a shows zone 1 configuration. Sensor 1 is the leader (green), Sensor 2 and Sensor 3 are simply ON (blue) while Sensor 13 is OFF (grey). As shown in Figure 4.3, the zone leader starts sensing plausible values, according to the application parameters. At a specified time the simulator forces the data trend to drift from the expected value.

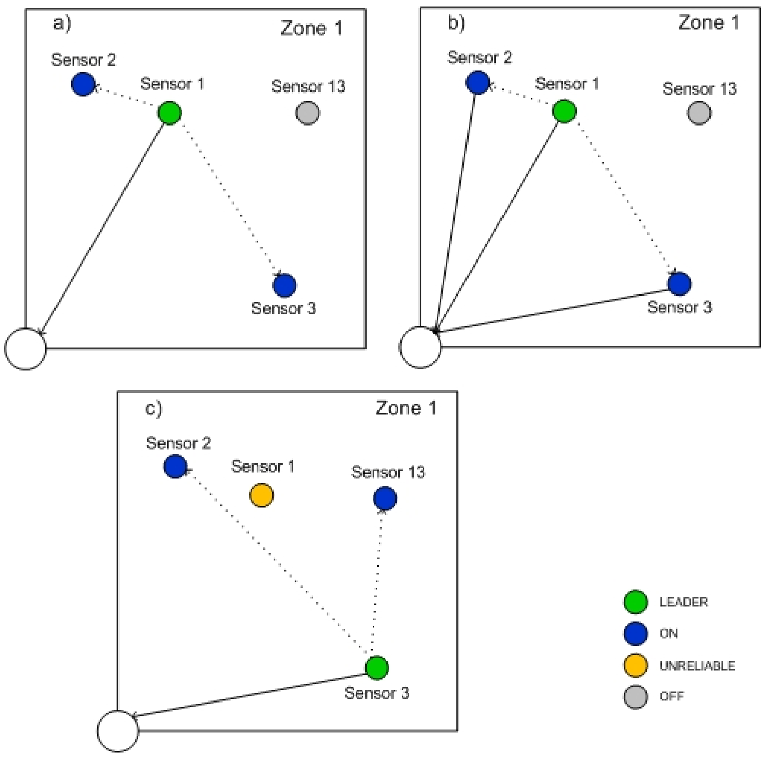


Figure 4.2: *An example of the redundant information approach: changes in network configuration*

When the ON sensors detects the drift, they start sending to the base station their values (see Figure 4.2b) to start the comparison. The comparison takes into consideration a number of samples set at design time; after this phase the base station notices that

Sensor 1 drift was not justified by the context and decides to promote Sensor 3 as the new zone leader, and sets Sensor 1 status as UNRELIABLE (Figure 4.2c) and set Sensor 13 status as ON for further comparisons.

In this way, we use a context information to categorize and increase the quality of measured data in our WSN. The context-aware controller, thus, is able to both increase the performances of the network classifying faulted data, enhancing the reliability of the sensor fusion.

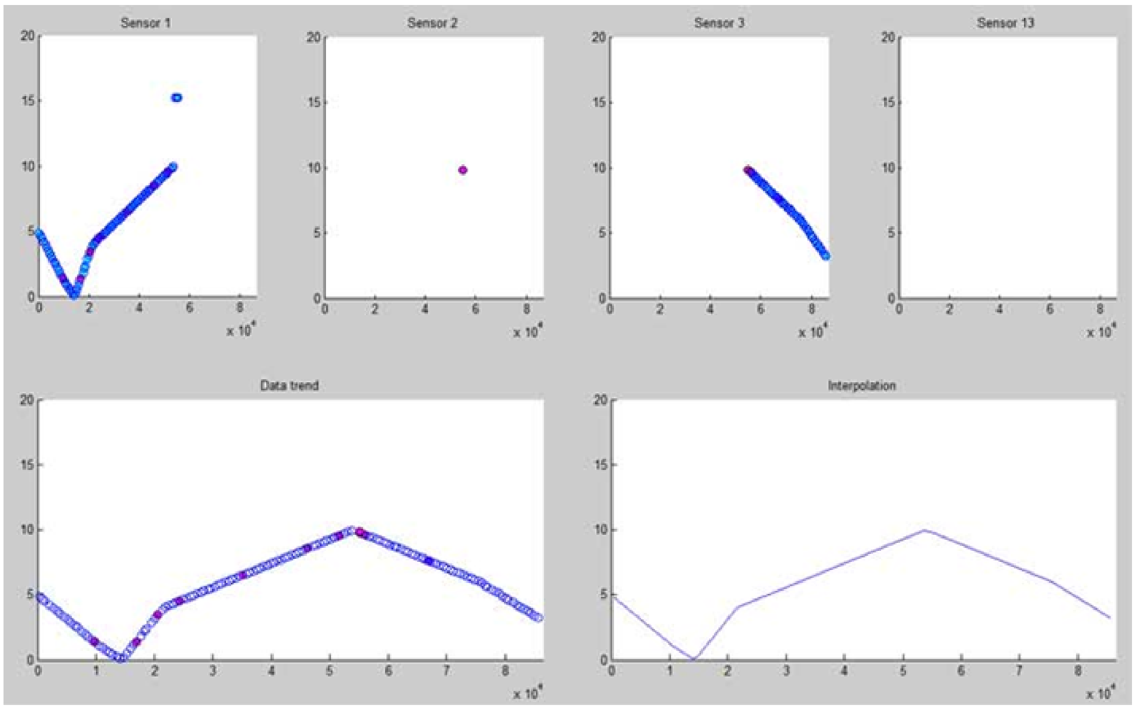


Figure 4.3: *An example of the redundant information approach: base station data management*

Figure 4.3 shows also how the base station, thanks to the information retrieved during the monitoring, is able to reconstruct the trend of the data, in the example assumed to be a day-long temperature monitoring, by means of only the values considered as reliable.

As just described, this function is called when the values of the leader are drifting, but they are still plausible. On the other hand, if the base station detects an irregular drift, for instance, measuring the environment temperature, a sensor reports 70^{o}C (see Figure 4.4a) , without waiting for other ON sensors values for the comparison, it sets the leader zone as FAULTY, promotes one ON sensor to LEADER and sets one OFF sensor to ON (see Figure 4.4b).

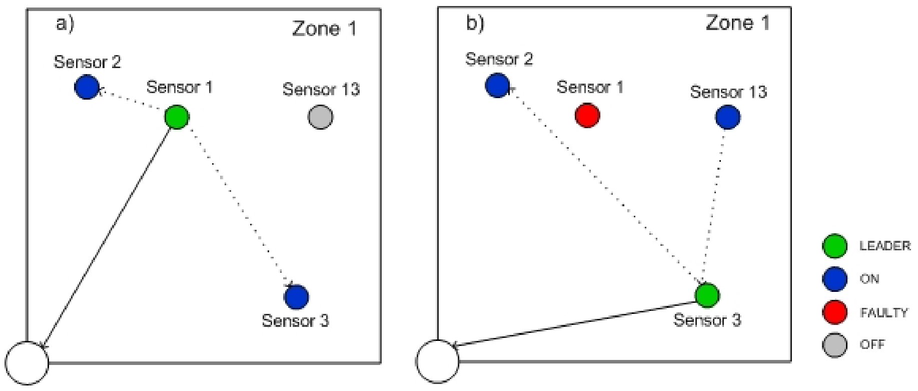


Figure 4.4: *An example of fault detection*

4.2.2 Packet Retrieval

Considering a realistic scenario we cannot fail in taking into consideration also a possible packets loss. Indeed, in real networks several factors can affect the loss of packets. The wireless channel, described in Chapter 5, is one of the main factors affecting the correct transmission between nodes.

On the other hand a realistic design has to consider also other possible causes of packets loss such as output buffer overflow, collisions and interference.

Buffer overflow occurs when the output buffer is fed with data at a rate higher than the sending (and thus, emptying) rate. In this case, the system will discard the extra packets, causing a packets loss.

Collisions occur when two or more sensors try to access to the communication link at the same time. In this case the transmissions fail and have to be re-scheduled.

Interference is the disturbance that affects an electrical circuit due to either electromagnetic induction or electromagnetic radiation emitted from an external source. The disturbance may interrupt, obstruct, or otherwise degrade or limit the effective performance of the transmission. These effects can range from a simple degradation of data to a total loss of data.

Adopting a MAC layer protocol is a first step in trying to address these issues. In fact, increasing the active period (see Section 3.4.3), for instance, allows the sensor to send more data, lowering the possibility of buffer overflows to occur. Furthermore, most of the MAC layer protocols, including the one adopted in our approach, provide the system with a set of functionalities that help to detect or avoid collisions. In particular, the IEEE 802.15.4 protocol uses a slotted carrier sense multiple access with collision avoidance (CSMA-CA) mechanism [34]. In Section 4.3 we illustrate in detail how changes in MAC layer protocol parameters can affect the performance of the network underlying their effects on the quality metrics under exam.

Nevertheless, the MAC layer protocol could be not sufficient; in fact, the protocol can help in increasing the overall network performance, but in case the recipient (i.e., the base station) does not receive a packet the MAC layer cannot trace the loss. In fact, the main MAC layer concern is to correctly access the communication medium and not to guarantee the effective reception at the base station.

In this sense, each time a packet is not received the application is not notified and cannot detect the packet loss. For this reason, we introduced a *packet retrieval* functionality at the application layer to overcome this issue. The base station detects a packet is missing and asks the sensor to re-send it; this function is particularly suitable in context in which completeness has priority over the other quality dimensions. In fact, as illustrated in Section 4.3, our approach receives as input the list of quality dimensions sorted by priority level in order to change and adapt its parameters to the context, according to the network quality requirements.

We now illustrate the packet retrieval function.

The Model

The packet retrieval is based on the sequence number of the different packets. Let us assume the packet defined by the following tuple:

$$P = < V, ts_1, ts_2, w, s, T >$$

here V is the aggregated value and ts_1 and ts_2 are respectively the timestamps of the first and the last value participating to the aggregation; w is the window the value belongs to, s is the sequence number of the value in the window and T is the total number of aggregate values in window w. Note that in case the value is an outlier ts_2 is 0.

Every time the base station receives a packet it checks in memory whether all the previous packets are present. It then requests all the missing values to the sensor. Obviously, this approach requires the sensor to store sent packets in memory.

We now illustrates the main steps in pseudo-code:

```
(1) BS stores packet P in memory
```

lastWin is the number of the window the last packet received belonged to.

```
(2)     IF lastWin == w
(3)         THEN FOR all $i < s$
(4)             IF P(i,w) NOT found THEN requestData(P(i,w))
(5)     ELSE IF lastWin < w
(6)             THEN FOR all $i$ from lastWin to w-1
(7)                 requestWholeWindow(i)
(8)     lastWin = w
```

4.3 Adaptation and Context-Awareness

Adaptive systems adjust their behavior according to the perception of the environment and of the system state itself [35]. Context-awareness is a fundamental characteristic of such systems and can be obtained from the analysis of the physical environment in which users and devices interacts. A large amount of definitions about context have been proposed; the one that best fits our approach perception of context is Dey's definition [36]: *Context is any information that can be used to characterize the situation of an entity.*

In our approach this context information derives from different sources:

Detecting a change in the input data trend is the first and most common example of context information. In this case, as illustrated in Chapter 3, the algorithm changes its parameter (namely the value of reference) to adapt to the incoming input data.

Another kind of context information derives from the detection of faulty or unreliable sensors; when this occurs, the base station is able to switch on and off the nodes adapting, in this way the network configuration to the sensors reliability (see Section 4.2.1).

Finally, the best source of information is dictated by the monitored phenomena itself. In fact, different phenomena call for different approaches, in terms of quality requirements and priorities. For instance, monitoring the temperature of an office does not call for a system able to provide high performances in terms of timeliness and completeness, but it would rather focus the priority on the energy saving. On the other hand, for example, an intrusion detection system calls for strict completeness and timeliness requirements to best accomplish its function.

These quality requirements and priority are well known a priori; thus, the user can set, at design time, an order of priorities among the quality dimensions, according to the monitored phenomena.

This is a fundamental step in our *action-based* approach; the base station, in fact, chooses the action to apply among a set of actions in order to satisfy the quality dimensions following their order of priority. Adopting this approach, our algorithm results to be suitable for different kind of applications.

To summarize, we can regard our approach's adaptiveness at two different levels:

design adaptiveness is based on the monitored phenomena. The user feeds the system with the network quality dimensions priorities; the base station will apply actions at run-time according to the order of priority.

run-time adaptiveness is strongly context-driven. According to the context, in fact, the base station adapts the network configuration, in terms of switched-on sensors (see Section 4.2.1) and changing MAC layer protocol parameters.

4.3.1 Action-based approach

The actions

So far, we illustrated the network quality requirements, in terms of quality dimensions, and a set of factors that mainly affect these quality dimensions. In particular, in Section 3.4, illustrating the MAC layer features, we focused the attention on those parameters that strongly impact the metrics under exam: the *macBeaconOrder* (BO), the *macSuperframeOrder* (SO or FO) and the number of *guaranteed time slots* (GTSs) assigned. Each parameter may influence the quality metrics differently, improving one dimension rather than another. These parameters must be tuned at run-time to better satisfy the quality requirements, adapting, in this way, the communication configuration to any possible variation in the context. To do so, we introduced the concept of action: according to what the base station detects, it decides to change the value of a parameter rather than another. At the MAC layer level, the actions are, thus, increasing or decreasing BO and FO values, and introduce ore increment the number of GTSs. The effects of these actions are clearly shown in Section 4.3.2.

Adaptive actions
Increase beacon order (BO)
Decrease beacon order (BO)
Increase frame order (FO)
Decrease frame order (FO)
Introduce/increase GTSs
Decrease/Remove GTSs
Activate Packet Retreival functionality
Deactivate Packet Retreival functionality
Enlarge window width
Decrease window width

Table 4.2: *Adaptation actions*

Furthermore, another factor that may affect the quality dimensions is the usage of the packet retrieval functionality introduced in Section 4.2.2. In fact, it is designed to improve the completeness dimension, but, on the other hand, retrieved packets can be received after a longer time, worsening in this way the timeliness dimension. This leads us to consider another action the base station can perform: if it detects an unsatisfactory level of completeness, it can enable the retrieval packet function or, on the other hand, disable it if the timeliness value is below a certain threshold. As for the MAC parameters, the effects of the usage of this functionality are illustrated in Section 4.3.2.

Finally, as mentioned in Section 3.2, another important parameter to take into consideration at run time is our aglorithm's *window width* (see Chapter 3). In fact, when the sampling interval is too short, the network traffic remarkably increases, causing to the network several issues such as buffer overflows and interferences. If the input signal is not too irregular, increasing this fundamental parameter results in a decrease of the network traffic. In this way, the compression degree may be higher but it would still satisfy the quality requirements because controlled by the *accuracy* and *precision* parameters of the algorithm. Thus, when the traffic is very high, it may be opportune to enlarge the window size, reducing it when other quality metrics are negatively affected by the change of it.

Table 4.2 summarizes all the actions considered by our approach.

The Model

The action-based system needs the user to set an order of priority among the quality metrics and a threshold of tolerance for each priority.

Let us assume to have a set of quality dimensions $Q =< m_1, m_2, \ldots, m_n >$. For each m_i we define a set of actions which enhance the network quality in terms of metrics m_i: $A_{m_i} =< a_{m_i,1}, a_{m_i,2}, \ldots, a_{m_i,k} >$. Note that the order of the actions for each metric is assigned in the configuration file and can be modified by the user according to his requirements.

The user's set of priority is described by $P =< p_1, p_2, \ldots, p_m >$. Each priority p_i is represented by a tuple in the form $< m, t, tol >$, where m is the metric the priority p_i refers to, t is the threshold the network must satisfy for metric m, and tol is the percentage of tolerance of satisfaction.

Each time the base station receives a packet from a zone leader sensor it calculates the value of the quality metrics (timeliness, completeness and delta energy consumption) and stores them in memory. When it collects a sufficient number of values (set at design time) for each metric, the base station computes the mean among the values for each metrics. It then analyzes the metrics values according to the quality dimensions order of priority. When a value is below the established threshold for a certain quality metric the base station applies the first action in the set of actions related to that metric.

Each time an action is performed, the base station waits for a defined amount of time to let the network find an equilibrium and then it starts again calculating the value of the quality metrics as illustrated before. From these new values the base station evaluates whether the action it just performed enhanced the overall network quality analyzing all the metrics in order of priority.

In case the performed action worsened the quality, the base station goes back on its steps, unapplying the action, and performs the next action for that metric, if existing. Otherwise, the base station goes on analyzing the other priorities.

This process continues until all the priorities are satisfied or until there are no more applicable action for that metric. The former case refers to the process we just illustrated. In the latter case, instead, the base station stops trying to enhance that specific quality metric or, if specified at design time, it suspends the attempts just for a defined elapse of time.

Once the time interval is expired the base station starts to consider again the former metric in case something in the context changed enabling it to enhance also the quality in respect of that metric.

Here we illustrate our model in pseudocode.

BS receives a packet p

The base station calculates the values of each metric V_{m_i} for the packet p and checks whether it has a sufficient number of values (for each zone leader sensor) to start the action-based reasoning. The following pseudo-code refers to the positive case. For the sake of simplicity, we refers to i as the metric under exam in the current state and to j as the metric under exam in the previous state. Note that i may be equal to j.

```
(1) FOR each p in P
(2)     IF i ≥ j
(3)         IF $V_{m_i} > p_i.t \pm p_i.tol$ AND $V_{m_i} < V_{m_j}$
(4)         THEN applyFirstAvailableAction in $R_{m_i}$
(5)                 EXIT FOR
(6)     ELSE IF i < j
(7)         IF $V_{m_i} > p_i.t \pm p_i.tol$
```

```
(8)         THEN unapplyLastActionApplied in $R_{m_j}$ AND applyFirstAvailableAction in
    $R_{m_i}$
(9)             EXIT FOR
(10) END FOR
```

Note that the base station keeps trace of each applied action, and thus of each state the network assumes. The function *applyFirstAvailableAction* decides which action to perform. It, obviously, applies the first action ($a_{1,1}$) of the set for that metric as first try. Then, if the threshold is still not respected but $a_{1,1}$ enhanced the value (V_{m_i}) relative to m_1, the base station applies again the $a_{1,1}$ otherwise, if it did not enhanced the value applies $a_{1,2}$ and so on, until it tries all the actions.

On the other hand, if performing an action $a_{m,i}$ results in decreasing any other previous quality dimension, the base station detects it and unapply the action (*unapplyLastActionApplied*) going back to the previous state, and, if possible, tries to apply $a_{m,i+1}$. To clarify the stream of actions, the example in Figure 4.5 illustrates, state by state, the decisions, and thus the actions, the base station takes in order to enhance the overall quality of a network.

In the example, each state is represented by a circle with a number inside; the three squares below the state represent the metrics (in this case three) under exam. Furthermore, the background color of the metric stands for its condition: green means the value of the metric satisfies the threshold; red means the value of the metric does not satisfy the threshold; yellow means the value of the metric is better than its value in the previous state, but it is still under the threshold; grey metrics are those metrics not checked yet due to the order of priority. The number in brackets on the arrow represents the number of move. Notice that two arrows having the same number in brackets are applied consequently, without calculating the value of the metrics again; otherwise, the base station waits until it has a sufficient number of values for each metric before performing any action. Finally, $A_{i,j}$ means the base station performed the action number

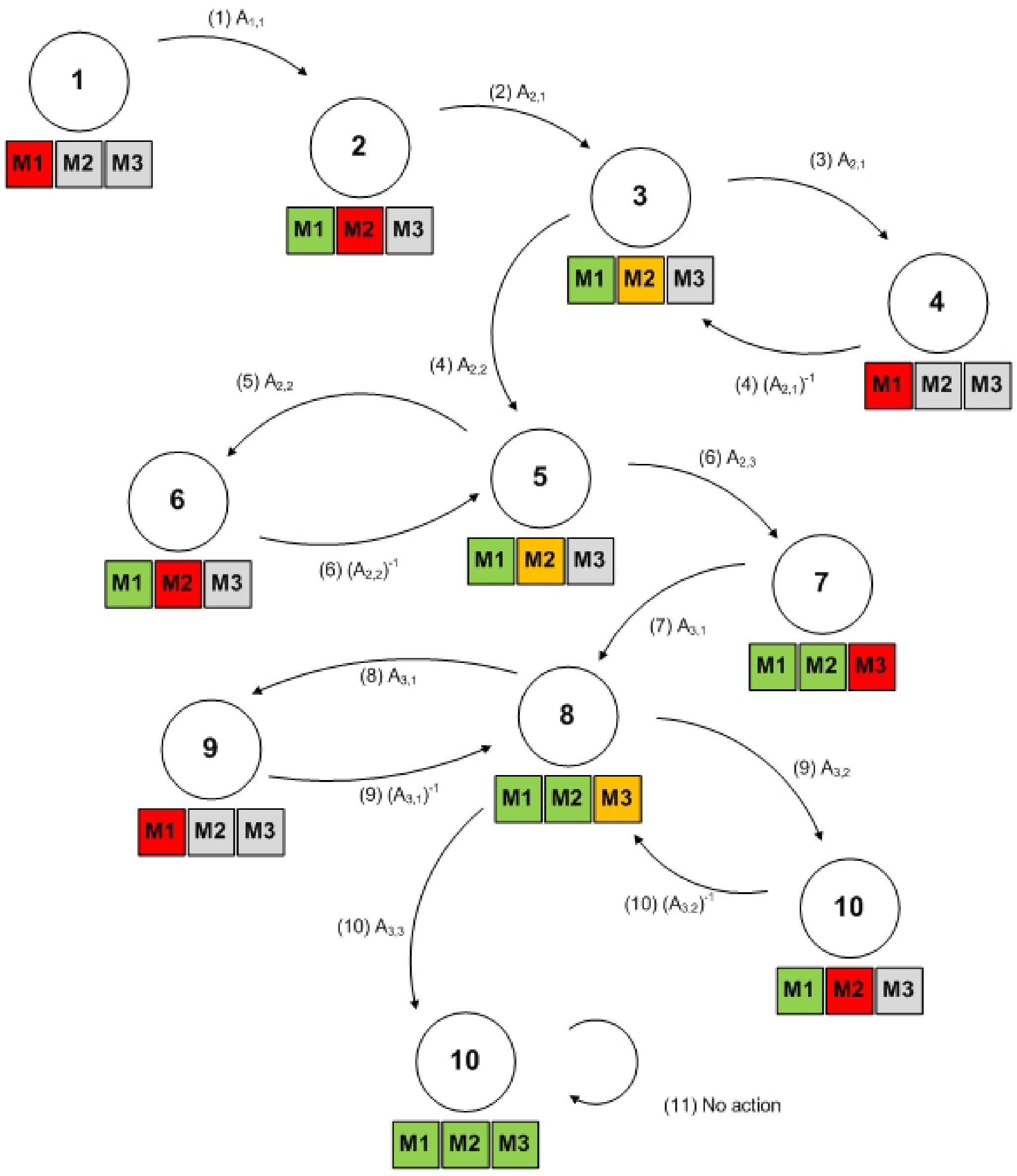


Figure 4.5: *Example of state flow of the action-based approach.*

j for metric i. The negative exponent means the action has been unapplied.

(1) The base station detects the first metric is not satisfied. Thus, it performs $A_{1,1}$

(2) Applying $A_{1,1}$ satisfies the threshold for M_1 but now the second priority is not satisfied. It, then, applies $A_{2,1}$

(3) $A_{2,1}$ enhances the quality in terms of M_2 but not sufficiently enough to surpass the threshold: it is performed again.

(4) Applying twice $A_{2,1}$ leads to a configuration in which M_1 is not satisfied anymore: the base station goes back on its steps and unapplies $A_{2,1}$ and performs, instead, $A_{2,2}$.

(5) In state 5, M_1 is still satisfied and the performed action enhances M_2, without, though, surpassing the threshold: $A_{2,2}$ is applied again.

(6) Performing twice $A_{2,2}$ instead of enhancing, worsens the configuration. $A_{2,2}$ is unapplied and the base station tries to performs $A_{2,3}$

(7) The last performed action enhances M_2. The base station detects, though, that M_3 is not satisfied. It performs $A_{3,1}$

(8) $A_{3,1}$ negatively affects M_1 which has the highest priority. $A_{3,1}$ is undone and $A_{3,2}$ is performed instead.

(9) $A_{3,2}$ does not affect M_1 but affects M_2. It also is undone and $A_{3,3}$ is applied.

(10) In state 10 all the metrics are satisfied. No more actions will be performed.

Notice that, if the first metric has a too strict threshold, the system might never be able to satisfy it. When this occurs, the system keeps changing parameters trying to improve that quality dimension without caring about the other dimensions, resulting in a bad overall configuration of the network. For this reason, the first set of priority must reflect a default configuration where the threshold are lower and easily satisfiable. In this way, we ensure that a minimum threshold for each dimension is respected. After the first set of priority is decided, it is possible to add other more constricting thresholds.

4.3.2 Changes in the configuration parameters: effects on the quality metrics

In this section we experimentally analyze how different configurations of the network, in terms of MAC layer parameters and activated functionalities, may affect the different network quality dimensions. In particular, we focus on the MAC layer parameters we discussed in Section 3.4.3 and the functionalities introduced in Section 4.2. To clearly show the effects, we present our tests grouped by affected data quality dimension (listed in Section 4.1).

The simulations present a star topology where four sensors sense and communicate to the base station. All the following simulations last for 1000 seconds and have been run with $sampleInterval$ = 100ms. The shown results represent the mean over 20 repetitions of the same simulation. The simulator used in our approach will be fully described in Chapter 5.

Energy

In our approach, the sensor energy consumption is mainly affected by the active period, i.e., the elapse of time during which the sensor radio is turned on for either transmission and reception activities. The active period is dictated by the MAC layer parameters of *macBeaconOrder* (BO) and *macSuperframeOrder* (FO). Thus, we calculated the energy consumption varying the values of BO and FO.

Figure 4.6 illustrates how these parameters impact on the energy consumption. The values in the graph represent the mean of the consumed energy among the four sensors of the network and are expressed in Joule. The figure clearly shows how, considering a fixed value of BO, the energy consumption increases as the FO parameter increases. On the other hand, considering a value of Frame Order, we obtain a greater energy consumption

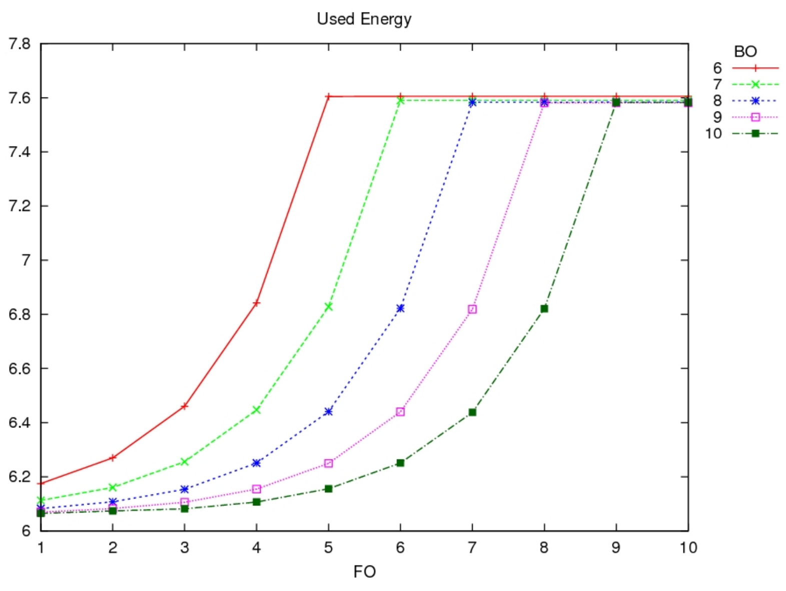


Figure 4.6: *Consumed Energy as FO and BO change.*

as the Beacon Order decreases. These results reflect what we formerly explained; in fact, decreasing the Frame Order length means decreasing the active period of the sensor: the less the sensor is active, the less it consumes. On the other hand, decreasing the Beacon Order value means decreasing the distance between a beacon and the next one. In this way, given a fixed time of simulation, the less the distance between beacons, the more beacons sent per simulation, which leads to a higher energy consumption.

Note that, as stated in Section 3.4.3, the following constraint must be respected: $0 \leq SO \leq BO \leq 14$. For this reason, we forced the simulator to change BO and FO according to this limitation. In particular, if $FO = BO$ the simulator put $FO = BO - 1$. This means that the horizontal lines in Figure 4.6 are meaningless and can be ignored.

The strict correlation between active period and energy consumption is clearly shown in Figure 4.7, where varying the sampling interval and thus the packets sent and received the consumed energy does not change at all.

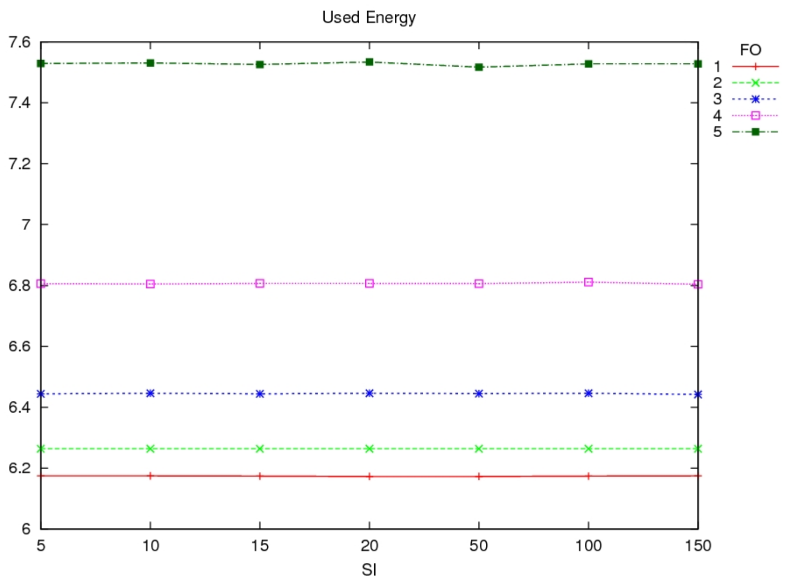


Figure 4.7: *Consumed Energy as the sampling interval changes*

Obviously, while the energy does not change as the sampling interval increases, the other dimensions may be affected as shown in the completeness section.

In the previous experiment, we did not use guaranteed time slots (GTSs) during the active period. Nevertheless, assigning GTSs to the sensors during the active period has effects on the consumed energy too.

Figure 4.8 shows the impact of the of the usage of GTSs on the energy consumption. The example takes into consideration different configurations, in terms of BO and FO values and number of GTSs used per sensor. For the sake of simplicity in the figure we reported only configurations with BO = 6 and BO = 8, since the pattern did not change even with other values of Beacon Order.

Figure 4.8 shows that using GTSs remarkably decreases the energy consumption. This happens because, when GTSs are assigned, the active period is divided into con-

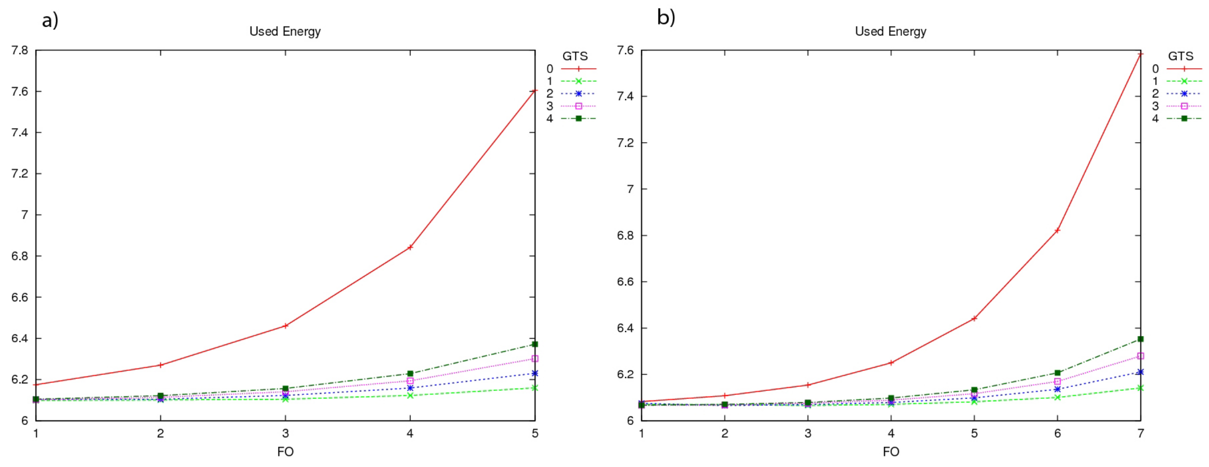


Figure 4.8: *Consumed Energy as the number of GTSs changes. a) Beacon Order = 6. b) Beacon Order = 8*

tention active period (CAP) and contention free period (CFP). During the CAP phase all the node are free to communicate, whilst during the CFP they can communicate only during their guaranteed time slots (GTS), resulting in a decrease of the overall active period and, thus, of the consumed energy.

Furthermore, if the MAC layer output buffer is filled for less than its half size (low packet rate), the CAP phase is disabled and sensors communicate only during their GTSs (see Section 3.4.

Note that GTS = 0 means no use of GTSs at all, which is the case of the first experiment in Figure 4.6.

Timeliness

Given the definition of *timeliness* in Section 4.1, it is highly expectable to detect changes in the timeliness dimension as the Beacon Order and the Frame Order parameters change. In fact, if the MAC layer does not manage to send all the packets in its buffer during its active period, it has to wait the next beacon to send them. For this reason, the higher the beacon order, the longer delay in packets delivery resulting in a lower timeliness value. On the other hand, if the active period (dictated by the FO parameter) is long enough there will not be packets postponed to the next beacon, resulting in a higher timeliness value. Figure 4.9 show the trend of the timeliness as the BO and FO parameters change. The value of timeliness under exam represents the average timeliness among all the four sensors, calculated as the mean of the timeliness of each packet received by the base station from each sensor.

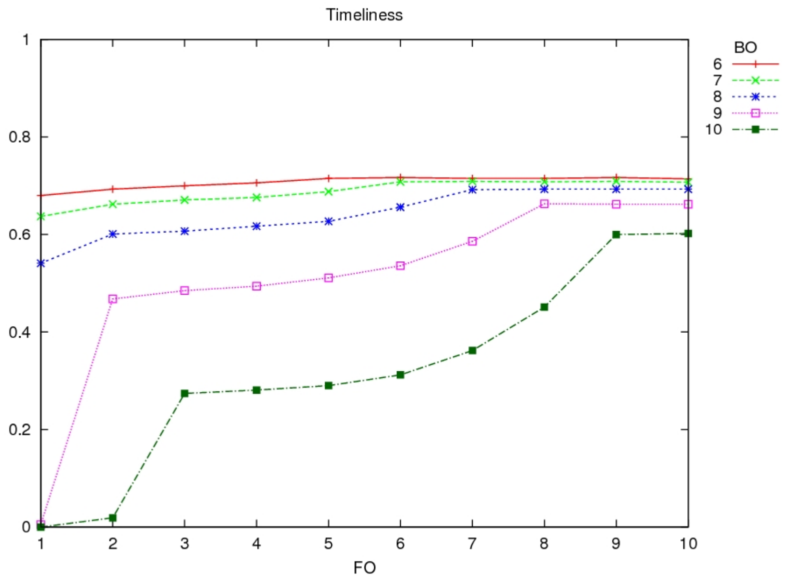


Figure 4.9: *Timeliness as as FO and BO change.*

In the experiment, to calculate the timeliness, we used a value of volatility equal to twice the window size in time unit: $volatility = samplingInterval * windowSize * 2$. In this way, packets received with more than two windows of delay will have a timeliness value equal to zero.

As we can see, for BO=10 when FO is equal either to 1 or 2, almost no packet is received within the limit of two windows, resulting in an overall timeliness equal to zero; increasing the frame order parameter, though, the value of timeliness increase. On the other hand, when BO=6, the frame order value barely affects the timeliness value.

This means that, no matter how short the active period is, the distance between two beacons, and thus between two active period, is small enough to allow the sensor to send the packets within the limit of two windows.

Obviously, timeliness is strongly related to the value of volatility. Considering the configurations in Figure 4.9, if the volatility was higher (for instance, three or four windows) the overall timeliness would have been higher and the initial step in BO=10 and BO=9 configuration would have been a smoother curve.

Completeness

As stated in Section 4.2.2, in a real scenario many factors may affect the packet loss. Packet loss due to buffer overflow occurs when the data rate is particularly elevated. In this case, the system loses packets because the buffer cannot be emptied in time. Figure 4.10 shows how varying the beacon order and frame order size can affect the overall completeness of the network.

As expected, increasing the frame order parameter enhances the network performance in terms of completeness. In fact, a longer active period enables each sensor to send packets before the buffer is full. Furthermore, the figure shows that the completeness increases as the beacon order decreases. Indeed, the shorter the inactive period the more packets the sensor is able to send.

During this simulation, we had to increase the network traffic in order to make the buffer overflow occur. For this purpose, we decreased the value of sampleInterval; in particular, for the experiment we used $sampleInterval = 10$. Figure 4.11 illustrates how the number of lost packets due to buffer overflow changes according to different beaconOrder and frameOrder configurations.

Nevertheless, buffer overflow is not the only cause of data loss and often, changing accordingly the frame order and the beacon order parameters, does not enhance the completeness. Indeed, observing Figure 4.10 is clear that, even in the best case, the

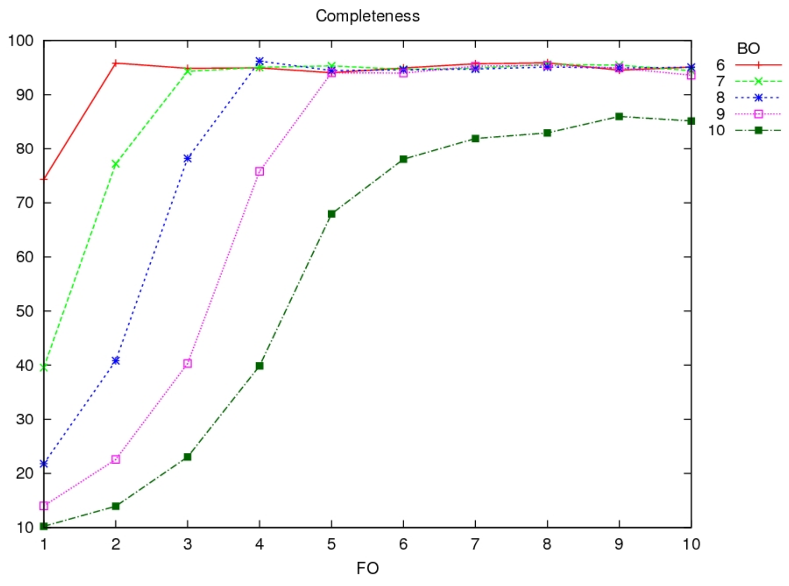


Figure 4.10: *Completeness as FO and BO change.*

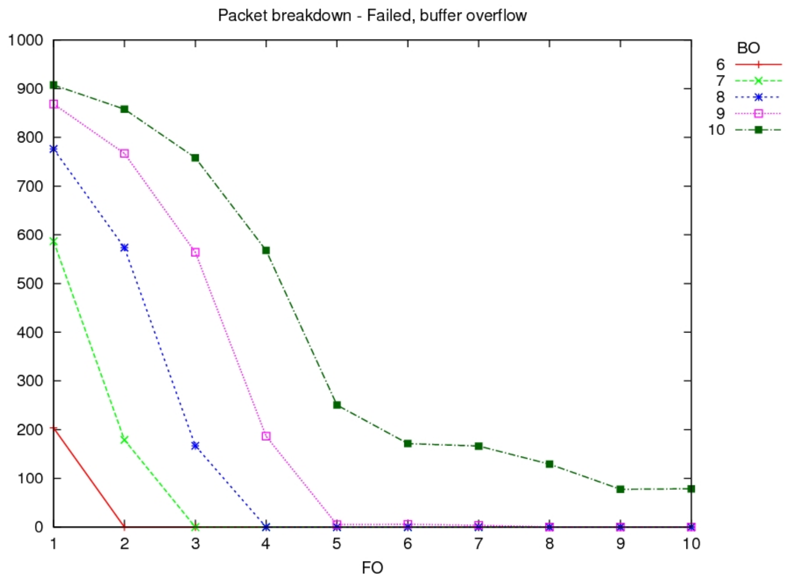


Figure 4.11: *Lost packets due to buffer overflow.*

completeness never reaches 100%, even in those configurations the number of lost packets due to buffer overflow is zero. This loss may be due to possible interference as shown in Figure 4.12

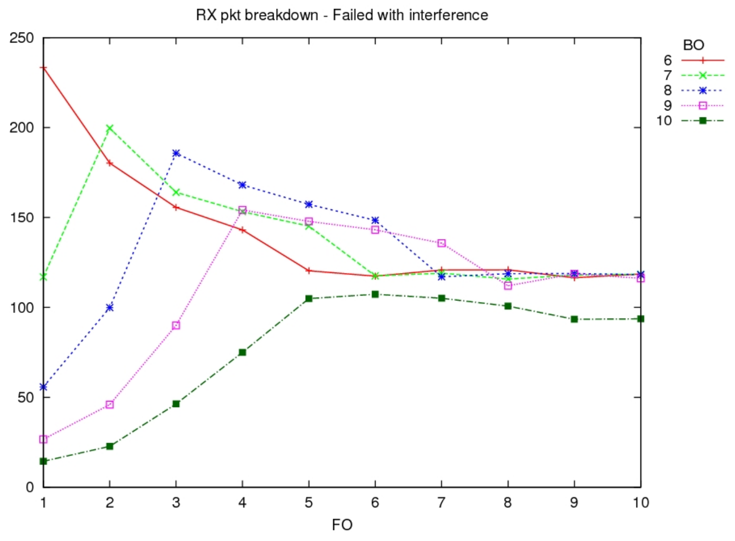


Figure 4.12: *Lost packets due to interference.*

In a scenario where even the smallest data loss is intolerable, though, other approaches must be adopted. A possible solution, in this case, would be the packet retrieval system we described in Section 4.2.2. Using this functionality, the base station is able to detect packet losses and then request to the sensor any missing packet. As long as the requested packet is still stored in the sensor memory, it can be re-sent to the base station. Figure 4.13 shows the effects of the packet retrieval functionality on the completeness compared to a general configuration not adopting this functionality. For the sake of simplicity, in the figure we consider a smaller set of configurations; as the previous experiment, $SampleInterval = 10$ and the simulation lasts for 100 seconds.

From the figure we notice that in most of the cases the packet retrieval function enhance the overall completeness of the network. Nevertheless, in some configurations it even worsens the level of completeness. This happens when the ratio between frameOrder

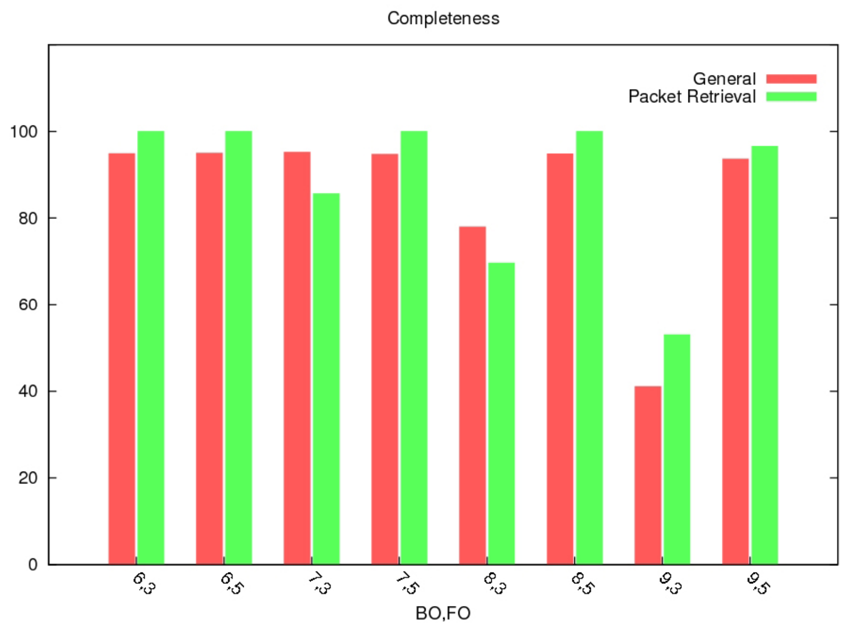


Figure 4.13: *Completeness in General and Packet Retrieval configurations with very high packet rate*

and beaconOrder is low or, in other words, when the active period is too short compared to the beacon period. In this way, since the packet rate is very high, the active period is too short to send the packets in time and during the inactive period the node add to the output buffer too many other packets. Activating the packet retrieval in such configurations, thus, may not be opportune because the policy behind that function is to request missing packets, resulting in an additional increase of the network traffic.

On the other hand, when the packet rate is not too elevated the latter issue does not occur, and introducing the packet retrieval function carries out good results, as shown in Figure 4.14; in this example the $SampleInterval = 100$ and the simulation lasts for 1000 seconds. Notice that, even in cases where the ratio between frameOrder and beaconOrder is low, the level of completeness using the packet retrieval is always 100%.

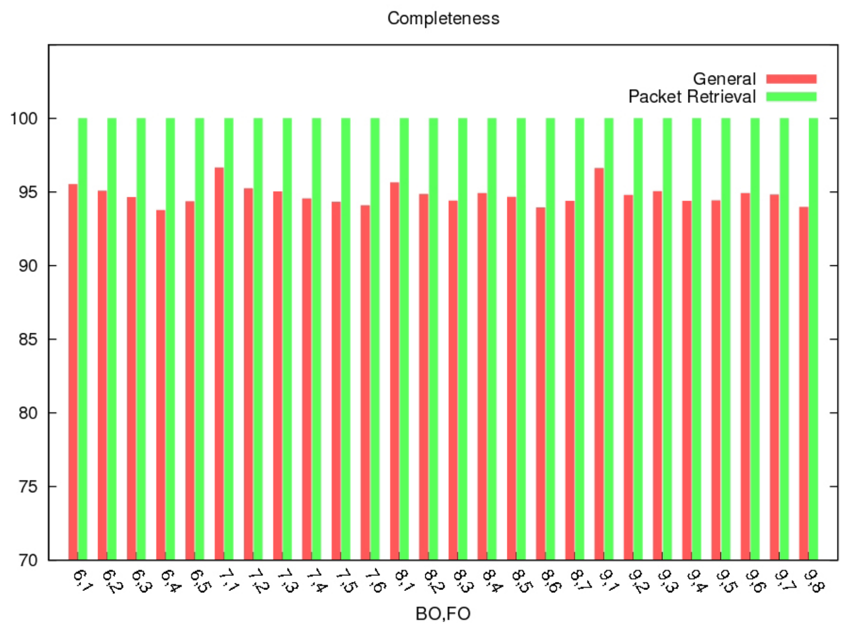


Figure 4.14: *Completeness in General and Packet Retrieval configurations with lower packet rate*

Note that packets received thanks to the retrieval functionality will have necessary a higher delay, resulting in a greater overall timeliness value, as shown in Figure 4.15. The timeliness values refer to the previous simulation.

Finally, in case where the data rate is elevated and timeliness has the priority among the quality metrics of the network, instead of adopting the packet retrieval system, a solution is assigning to the sensor guaranteed time slots (GTSs). Indeed, GTSs allows the sensors to transmit in different time slots avoiding, in this way, collisions during the packet transmissions. The precise number of GTSs assigned to each sensors strongly affect the completeness too. Figure 4.16 illustrates how the completeness changes as the number of assigned GTSs changes.

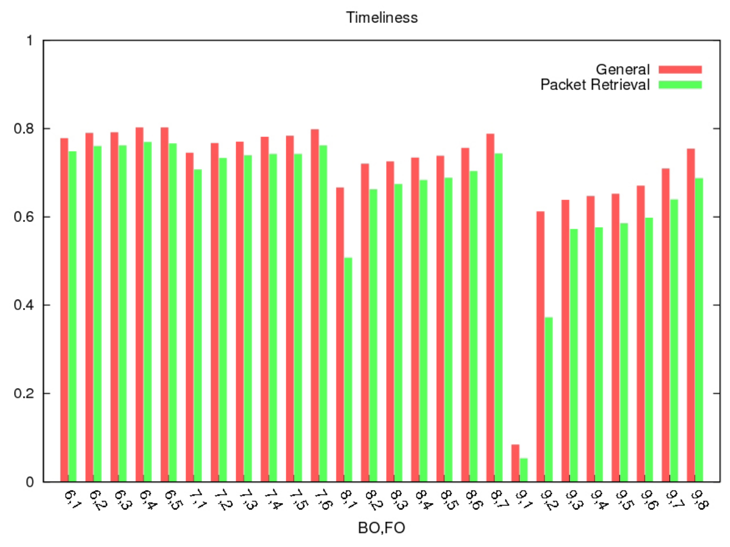


Figure 4.15: *Timeliness in General and Packet Retrieval configuration*

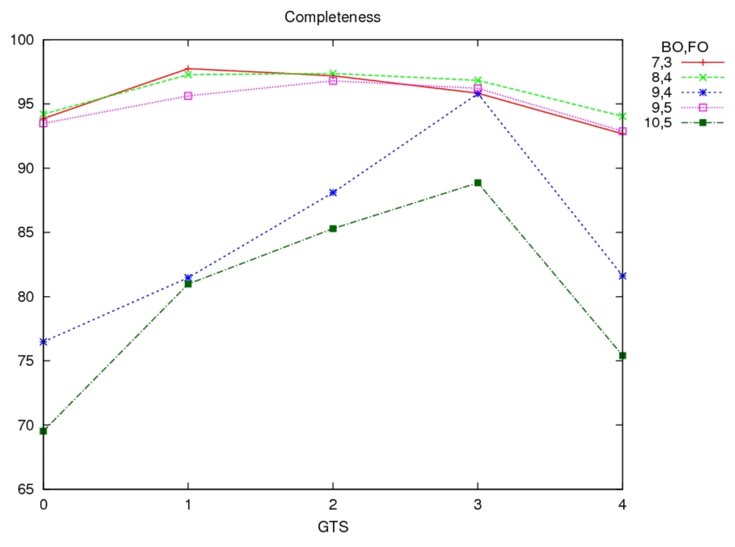


Figure 4.16: *Completeness and different GTSs*

From the figure we evince that the introduction of guaranteed time slots (from

0 to 1) always enhance the overall completeness of the network. Furthermore, it is clear that a large number of GTSs (in our example: 4) may decrease it This occurs according to the number of sensors involved in the simulation. In fact the number of GTSs represents the maximum number of GTSs the base station *can* assign to a sensor. The configuration of the network in our simulation presented four sensors sending data to the base station. With four GTSs some sensors might have more GTSs assigned rather than others, resulting in a possible data loss for the latter sensors. Since the performance of this policy is highly related to the number of sensors in the network, the user is able to set the maximum number of GTSs assignable at design time.

4.3.3 Implemented Adaptation Actions

According to the results previously presented we defined a set of actions to be applied at run-time to improve one quality dimension or another. The actions are listed in Table 4.3, grouped by quality dimension.

Quality dimension	**Action**
ENERGY	Introduce GTSs/increase GTSs
	Increase beacon order (BO)
	Decrease frame order (FO)
	Enlarge window width
TIMELINESS	Decrease beacon order (BO)
	Increase frame order (FO)
	Reduce window width
COMPLETENESS	Activate Packet Retreival functionality
	Deactivate Packet Retreival functionality
	Increase frame order (FO)
	Decrease beacon order (BO)
	Introduce GTSs/increase GTSs

Table 4.3: *Set of rules, grouped by quality metric*

In each action, increasing and decreasing the parameter imply the variation of one unit each time the action is performed. Activation and deactivation of the packet retrieval function are applied according to the conditions illustrated in Section 4.3.2.

Furthermore, the order they are applied in can be set at design time by changing the configuration file according to the network requirements.

It is important to underlying that changing MAC layer parameters at runtime gives our algorithm a cross-layers configuration.

5 Experimentation

In this chapter we present a set of tests and results performed thanks to the simulator Castalia. For this reason, in the first section we introduce the reader to this useful tool, illustrating its most important features and potentialities. We then describe the different scenarios of each simulation and their relative network configuration reporting the results we obtained by mean of Castalia.

5.1 Castalia: the Simulator

5.1.1 General overview

Castalia is a simulator for Wireless Sensor Networks (WSN), Body Area Networks (BAN) and generally networks of low-power embedded devices [37]. It is based on the OMNeT++ platform, an excellent framework to build event-driven simulators, and can be used by researchers and developers who want to test their distributed algorithms and/or protocols in realistic wireless channel and radio models, with a realistic node behavior especially relating to access to the radio.

The main features of Castalia are:

- Advanced **channel model** based on empirically measured data.

  - Model defines a map of path loss, not simply connections between nodes
  - Complex model for temporal variation of path loss
  - Fully supports mobility of the nodes
  - Interference is handled as received signal strength, not as separate feature
- Advanced **radio model** based on real radios for low-power communication.
  - Probability of reception based on SINR, packet size, modulation type. PSK FSK supported, custom modulation allowed by defining SNR-BER curve.
  - Multiple TX power levels with individual node variations allowed
  - States with different power consumption and delays switching between them
  - Realistic modeling of RSSI and carrier sensing
- Extended **sensing** modeling provisions
  - Highly flexible physical process model.
  - Sensing device noise, bias, and power consumption.
- Node **clock drift**
- MAC and routing protocols available.
- Designed for **adaptation** and **expansion**.

Proper modularization and a configurable, automated build procedure help the users to implement/import their algorithms and protocols into Castalia while making use of the features the simulator is providing. The modularity, reliability, and speed of Castalia is partly enabled by OMNeT++. OMNeT's basic concepts are modules and messages. A simple module is the basic unit of execution. It accepts messages from other modules or itself, and according to the message, it executes a piece of code.

5.1.2 Structure

Figure 5.1 shows Castalia's basic module structure.

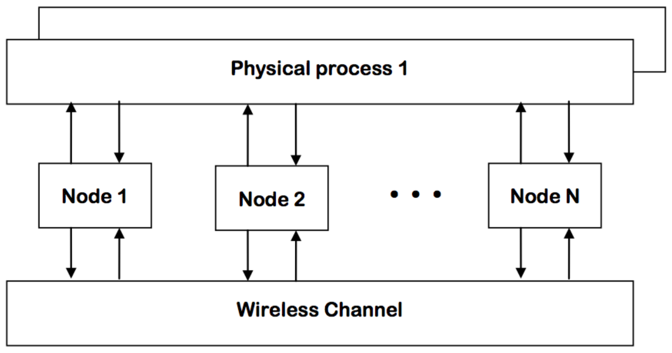


Figure 5.1: *The modules and their connections in Castalia*

Notice that the nodes do not connect to each other directly, but through the wireless channel module. The arrows signify message passing from one module to another. When a node has a packet to send this goes to the wireless channel which then decides which nodes should receive the packet. The nodes are also linked through the physical processes that they monitor. For every physical process there is one module which holds the "truth" on the quantity the physical process is representing. The nodes sample the physical process in space and time (by sending a message to the corresponding module) to get their sensor readings.

The node module is a composite one. Figure 5.2 shows the internal structure of the node composite module. The solid arrows signify message passing and the dashed arrows signify simple function calling. For instance, most of the modules call a function of the resource manager to signal that energy has been consumed.

This structure depicted in the figure above is implemented in Castalia with the use of the OMNeT++ NED language. This language allows to easily define modules, i.e., define a module name, module parameters, and module interface (gates in and gates out) and possible submodule structure (if this is a composite module).

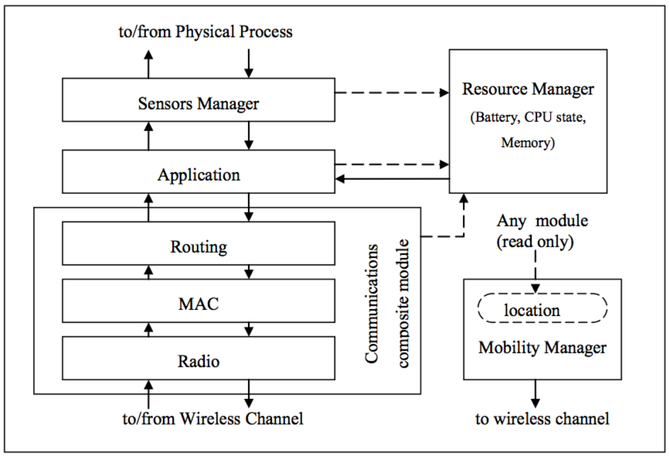


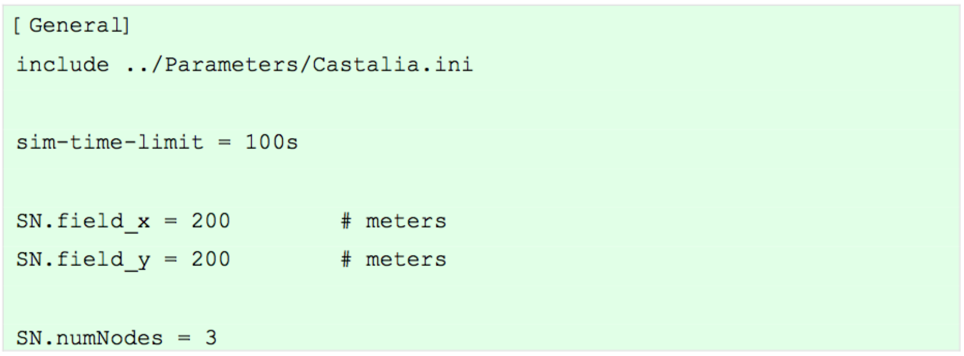
Figure 5.2: *The node composite module*

Notice that these files are dynamically loaded and processed (using a feature of OMNeT) so that any change does not require the recompilation of Castalia (unless of course new simple modules with new functionality appear).

5.1.3 The Configuration File

As already mentioned, the NED file allows to define the module. In the NED file, though, we can also define default values for the parameters. A configuration file (usually named omnetpp.ini and residing in the Simulations dir tree) assigns values to parameters, or just reassigns them to a different value from their default one. This configuration file allows the user to build a great variety of simulation scenarios.

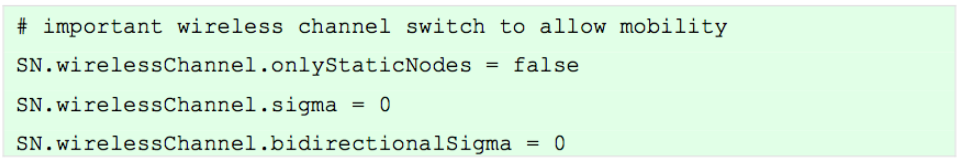
The configuration file starts with the General section. Every OMNeT configuration file has a General section. There the user defines the basic scenario that this file describes; parameters such as the simulation time and number of nodes do not have any default values and must be defined in this section (see Figure 5.3).



```
[ General]
include ../Parameters/Castalia.ini

sim-time-limit = 100s

SN.field_x = 200        # meters
SN.field_y = 200        # meters

SN.numNodes = 3
```

Figure 5.3: *Beginning of a configuration file*

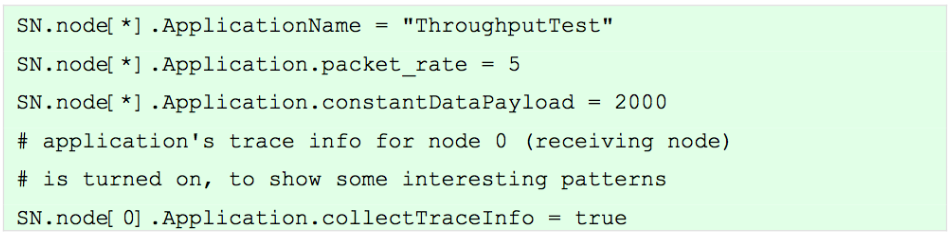
After the setting of the top level parameters, the user must set the module parameters such as radio, MAC layer and routing parameters. When setting this kind of parameters, the full name of the module is given revealing the module hierarchy structure (see Figure 5.4).



```
# important wireless channel switch to allow mobility
SN.wirelessChannel.onlyStaticNodes = false
SN.wirelessChannel.sigma = 0
SN.wirelessChannel.bidirectionalSigma = 0
```

Figure 5.4: *Module parameters in the configuration file*

In the configuration file we need to set the application the node has to run. Given the application, we can change its parameters from the configuration file as illustrated in Figure 5.5.



```
SN.node[ *] .ApplicationName = "ThroughputTest"
SN.node[ *] .Application.packet_rate = 5
SN.node[ *] .Application.constantDataPayload = 2000
# application's trace info for node 0 (receiving node)
# is turned on, to show some interesting patterns
SN.node[ 0] .Application.collectTraceInfo = true
```

Figure 5.5: *Application parameters in the configuration file*

Notice that with $SN.node[*]$ the parameter is applied to all the sensors; specifying the number of the node changes the parameter only to that sensor.

A change in the parameters of a module must be performed according to the modules hierarchy. The following diagram (Figure 5.6) provides an easy way to see all the modules and also show their hierarchical relations.

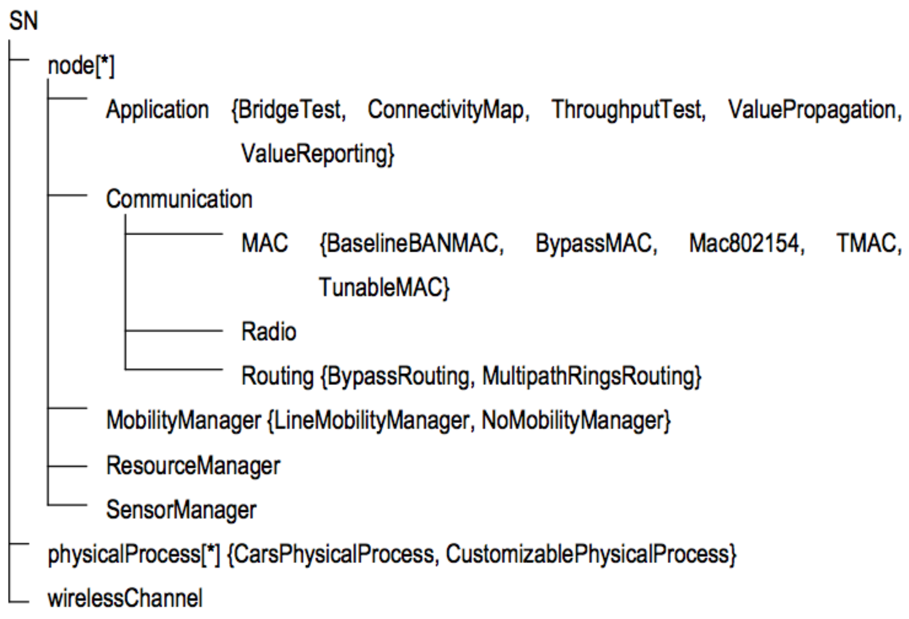

SN
node[*]
Application {BridgeTest, ConnectivityMap, ThroughputTest, ValuePropagation, ValueReporting}
Communication
MAC {BaselineBANMAC, BypassMAC, Mac802154, TMAC, TunableMAC}
Radio
Routing {BypassRouting, MultipathRingsRouting}
MobilityManager {LineMobilityManager, NoMobilityManager}
ResourceManager
SensorManager
physicalProcess[*] {CarsPhysicalProcess, CustomizablePhysicalProcess}
wirelessChannel

Figure 5.6: *Modules hierarchy*

5.1.4 Modeling in Castalia

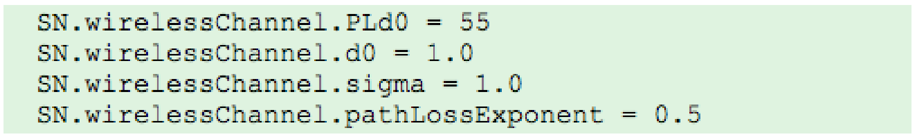
Communication is the most carefully modeled aspect of the Castalia simulator [37]. From the wireless channel to the radio behavior and the implementation of different MAC protocols, the authors tried to capture the essence and many details of their real counterparts.

Concerning the wireless channel, Castalia is the most realistic simulator one could find for WSNs. In fact, "tuning" appropriately the input parameters, the user can obtain results that accurately reflect reality.

One important aspect of the wireless channel modeling is to estimate the average path loss between two nodes, or in general, two points in space. For WSN, lognormal distribution has been shown ([38]) to give accurate estimates for average path loss. This is the formula that returns path loss in dB as a function of the distance between two nodes and a few parameters:

$$PL(d) = PL(d_o) + 10 \cdot \eta \cdot \log\left(\frac{d}{d_o}\right) + X_\sigma$$

where $PL(d)$ is the path loss at distance d, $PL(d_0)$ is the known path loss at a reference distantce d_o, η is the path loss exponent, and X_σ is a gaussian zero-mean random variable with standard deviation σ. All these parameters has default values which can be modified in the configuration file. In the simulations presented in next section we tuned the pathless model with the following values:



```
SN.wirelessChannel.PLd0 = 55
SN.wirelessChannel.d0 = 1.0
SN.wirelessChannel.sigma = 1.0
SN.wirelessChannel.pathLossExponent = 0.5
```

Figure 5.7: *Path loss parameters*

The first two values are as given by default; we changed, instead, the last two values, to better fit our network configuration in terms of distance between sensors and base station.

Another very important aspect of the wireless channel is the temporal variation, especially pronounced in rapidly changing environments. Figure 5.8 shows a typical profile of channel variation. Notice that most of the time the channel is below the average path loss (0dB).

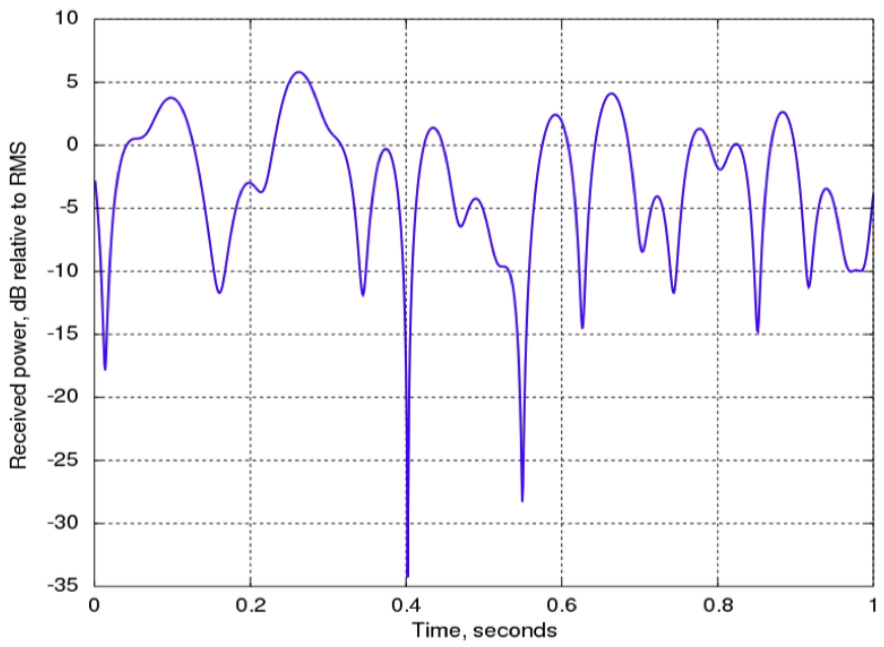


Figure 5.8: *Typical temporal fading in wireless*

In our simulations, we adopted the model in Figure 5.8, given by the authors, by simply specifying in the configuration file the following line:

```
SN.wirelessChannel.temporalModelParametersFile = "../Parameters/WirelessChannel/BANmodels/TemporalModel.txt"
```

The previous setting allows us to reproduce scenarios as close as possible to reality giving to the simulations in next chapter a high level of reliability.

5.2 Experimentation

5.2.1 Network configuration

The set of simulations presented in this section share the same network configuration. In particular, we set the simulation field as a 20x20 meters square, divided in four equal zones. Each zone is formed by one sensor leader and two other additional sensors, and the base station is situated in the middle of the field as shown in Figure 5.9.

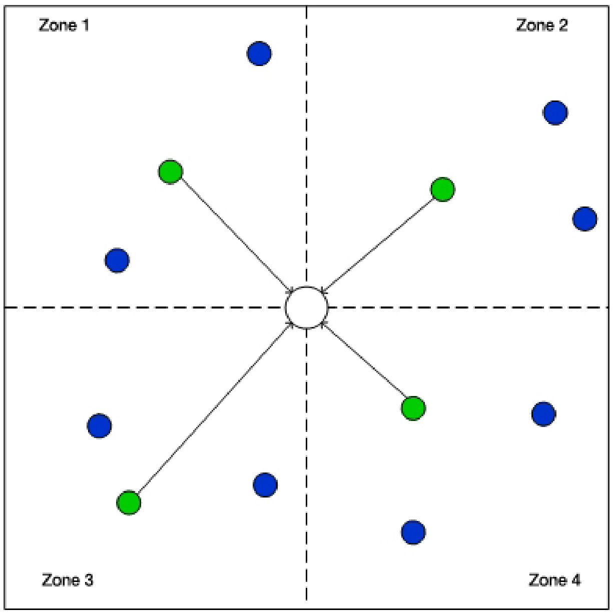


Figure 5.9: *Network configuration during the simulations*

5.2.2 Initial configuration

The following simulations compare the behavior of a general configuration to the behavior of our action-based approach in terms of network quality metrics satisfaction. For this reason the two simulations need to start from the same initial configuration. For the sake of correctness the initial configuration of the general case needs to be chosen wisely, according to the system requirements. In this sense, the initial parameters must be set reflecting the results obtained in Section 4.3.2, taking into account, besides the priority, the number of sensors involved and the data rate.

5.2.3 Simulations

In this section, we report our simulations results. The simulations reflect three different scenarios, each of which characterized by a different order of priority to satisfy:

Completeness. In this scenario, the packet loss is a critical event, as, for instance, in intrusion detection systems. In this case, the system calls for a high value of completeness.

Energy. This scenario aims at reducing the energy consumption rather than enhance the completeness and timeliness dimension. Systems where the monitored phenomena is quite stable, as environment temperature monitoring, reflect this scenario.

Timeliness. In this scenario, the freshness of data is crucial, as, for instance, in sense-and-react systems. For this reason, the systems requires a high value of timeliness.

Furthermore, for each scenario we present two different set of simulations grouped by different sampling intervals: $10ms$ and $100ms$.

As described in Section 4.3.1, the base station, to perform any action, needs to collect a sufficient number of values for each quality metrics. Moreover, after the action is performed, the base station waits a fixed elapse of time, before starting to collect other metrics values. These parameters must be chosen according to the sampling interval. In our simulations we used the values shown in Table 5.1.

Parameter	**Sampling Interval=10ms**	**Sampling Interval=100ms**
minValTimeliness	15	8
minValCompleteness	10	6
minValEnergy	15	8
waitAfterAction [s]	1	4

Table 5.1: *Action-based approach parameters*

Each simulation receives as input signal the one pictured in Figure 5.10 (created by means of Castalia), and lasts for 1200 seconds.

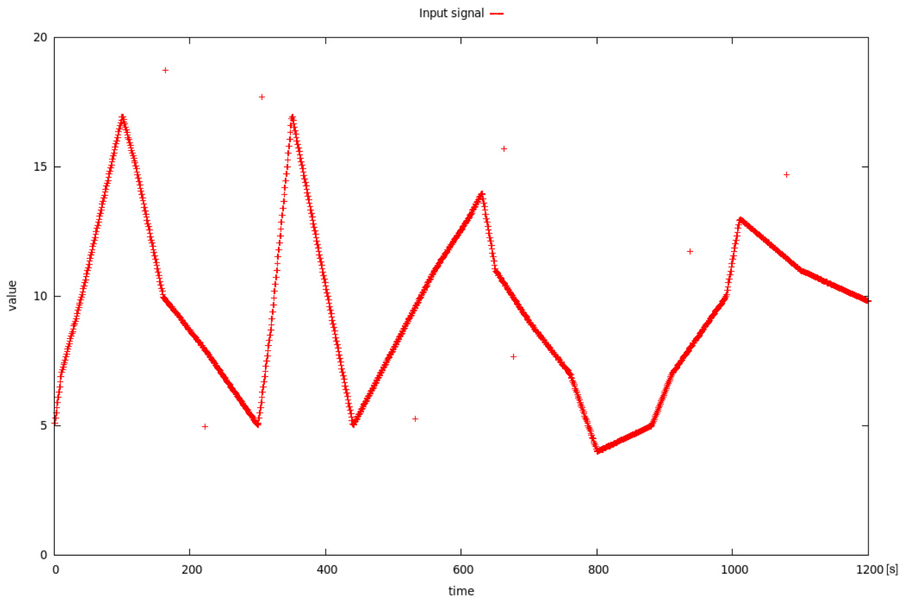


Figure 5.10: *Input signal of each simulation*

Completeness

Let us assume we are monitoring a system in which the completeness dimension is the most important requirement. The value of this metric, then, must be greater or equal to 99%. Since the priority is the completeness, for the other metrics we set less constraining threshold. Figure 5.11 shows the results obtained for a set of different configuration using $samplingInterval = 10ms$. In this case the volatility parameter is equal to 2 seconds.

The figure shows that, thanks to the adaptive-action approach, the first priority is satisfied for every initial configuration, while without using it, the value of the completeness never reaches its threshold. The timeliness dimension results to be higher for the general configuration but since the threshold is set to 0.1 the approach still satisfies the requirements in each configuration. This result is achieved at the expense of the last priority which is, in this case, the power consumption. In this set of simulations, in fact, our approach consumes more energy than the general configuration.

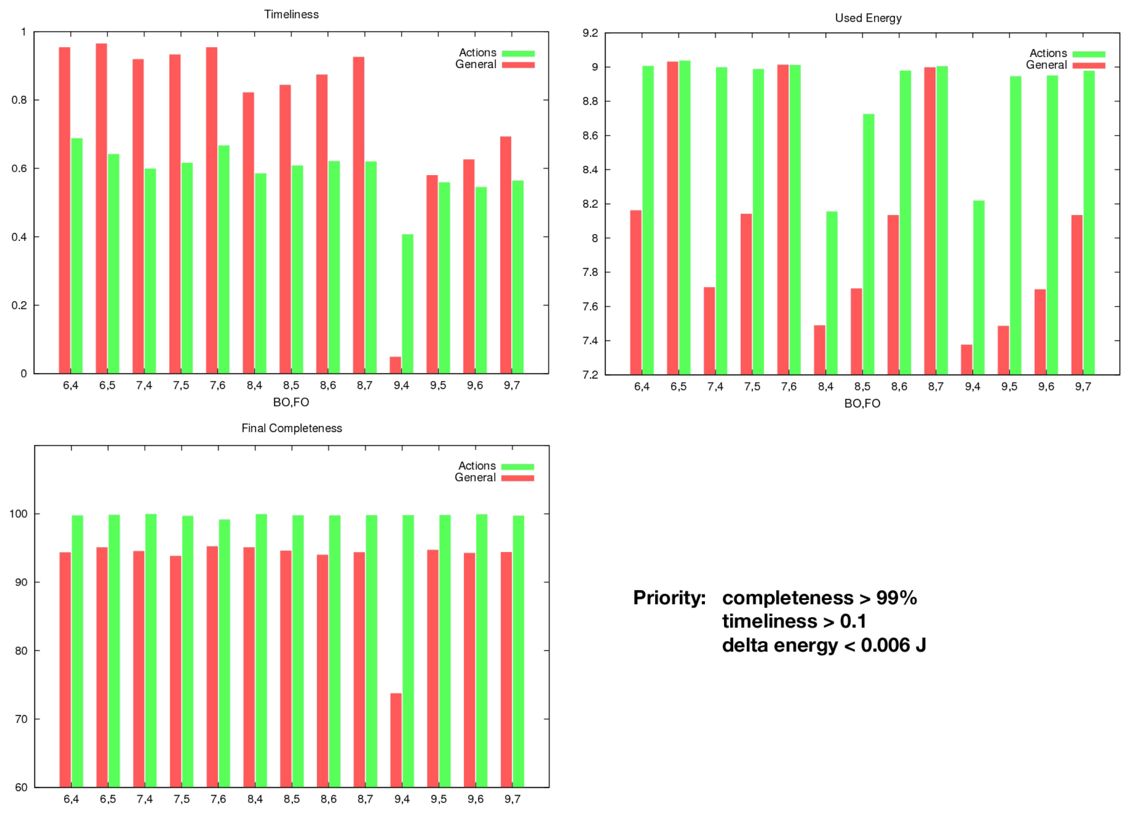


Figure 5.11: *Dimension values for different configuration as the completeness has the priority. (*$Samplinginterval = 10ms$*)*

Figure 5.11 reports the results of the same experiment with $samplingInterval = 100$

The results are similar to those obtained with $samplingInterval = 10$, except for the energy. In this case in fact, the energy consumption is more similar to the general configuration because the traffic load is lower than the one in the previous example and then it is easier for the base station to preserve the completeness without affecting the energy consumption.

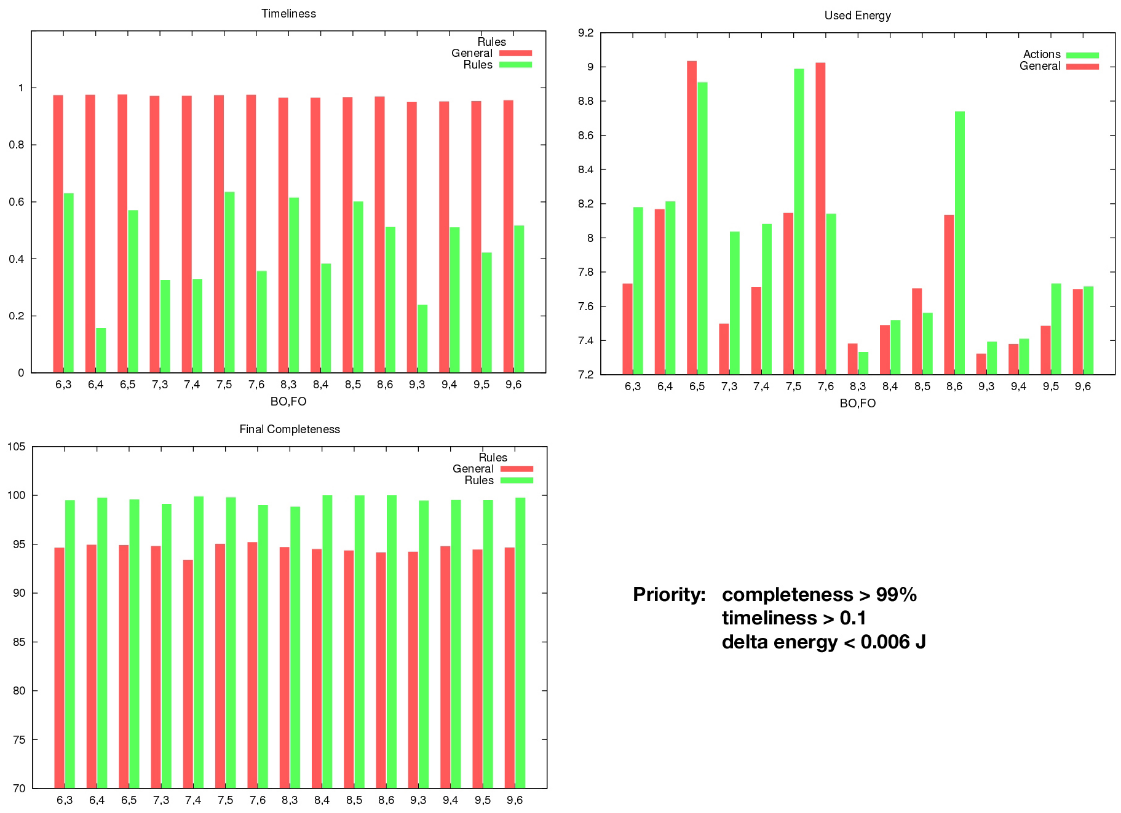


Figure 5.12: *Dimension values for different configuration as the completeness has the priority.* $(Sampling interval = 100ms)$

Energy

Let us consider, now, a scenario where the priority of the system is the energy saving. In this case, we set as first priority the delta energy consumption. Figure 5.13 shows the results of the simulations with the interval of sampling equal to 10 ms. The constraints for the other dimension are smoother and from the figure we can evince that the action-based approach satisfies all the requirements keeping the energy consumption at the lowest possible level. With $samplingInterval = 100$ (see Figure 5.14) the trend of the results is very close to the previous experiment.

This is a remarkable result because a sampling period of 10 ms leads to a very high network traffic load, but the behavior does not change thanks to the effectiveness of the

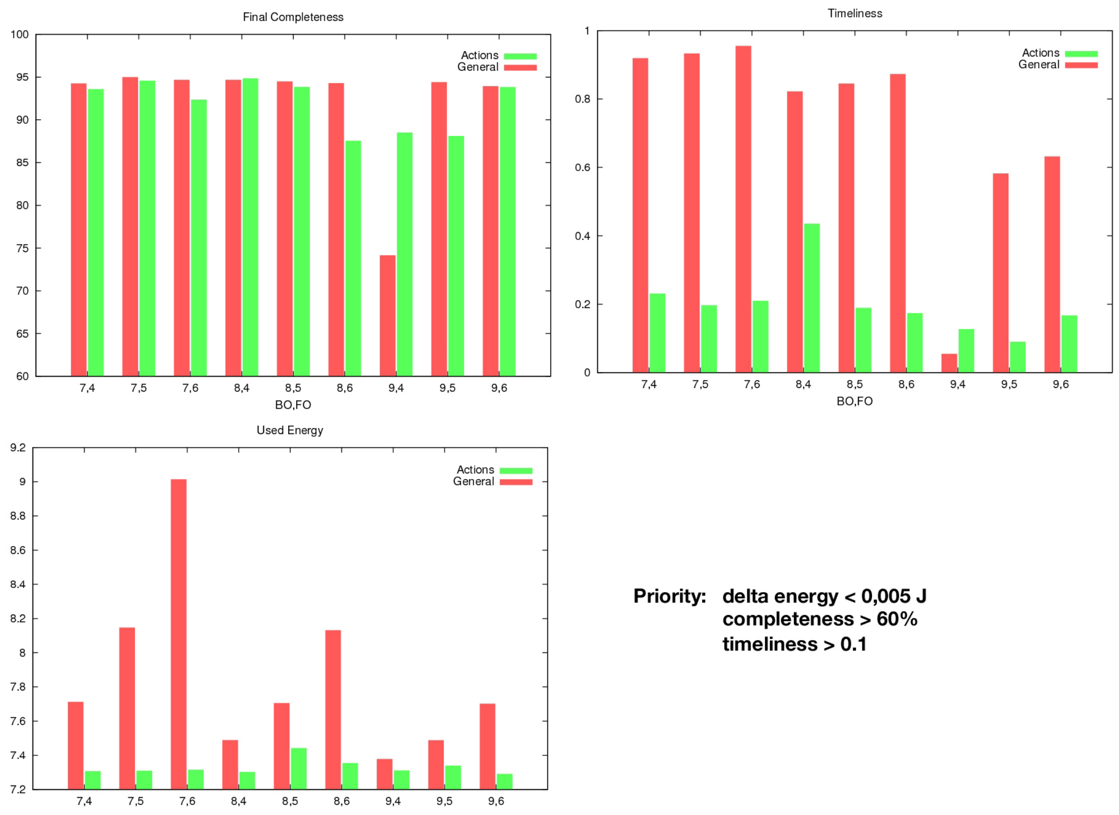
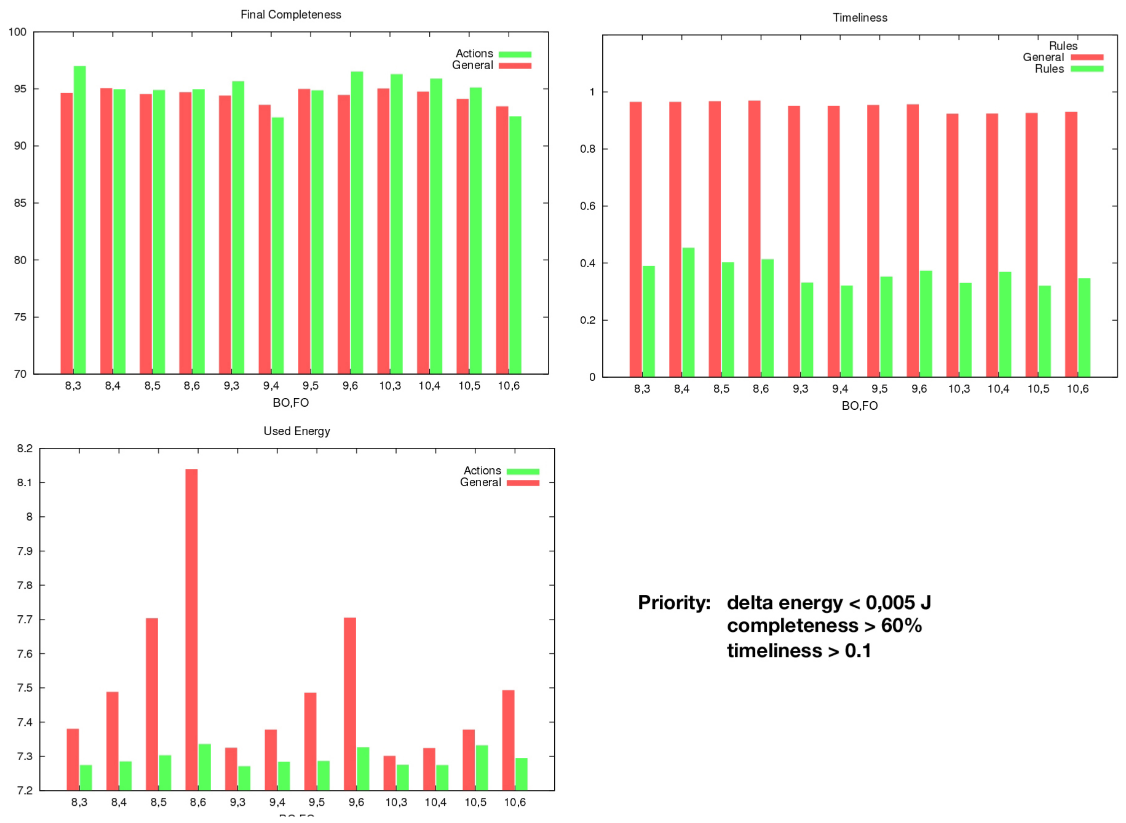


Figure 5.13: *Dimension values for different configuration as the energy has the priority.* $(Samplinginterval = 10ms)$

aggregation data algorithm installed on the sensor. In fact, the base station adjusts the width window of the aggregation algorithm according to the trend of the input signal., decreasing in this way the number of packets sent to the base station. In our case, the signal is not too irregular and the algorithm can perform a good aggregation even with a larger window.



Priority: **delta energy < 0,005 J**
completeness > 60%
timeliness > 0.1

Figure 5.14: *Dimension values for different configuration as the energy has the priority.* $(Samplinginterval = 100ms)$

Timeliness

The last set of simulations describes a scenario where the timeliness has the priority among the other metrics. In this sense, the system requires the received data to be as fresh as possible. For this reason we set the timeliness threshold to 0.8. Figure 5.15 and 5.16 show the results in the two different configuration.

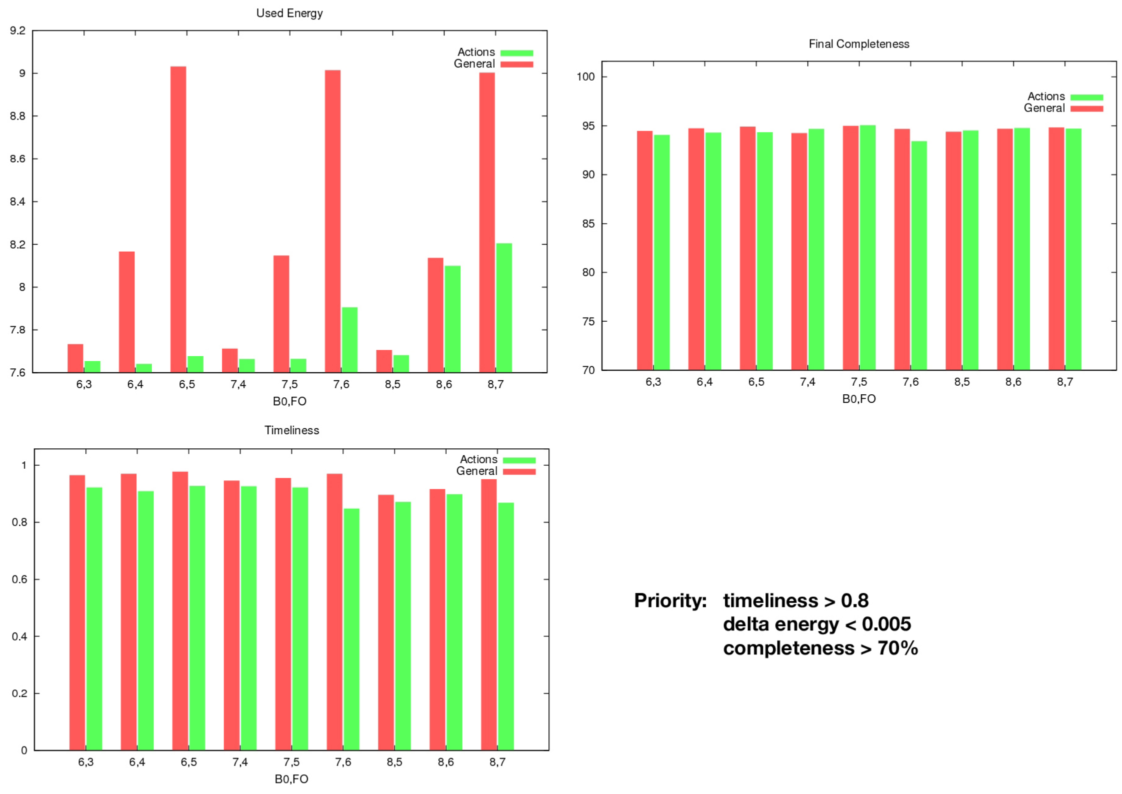
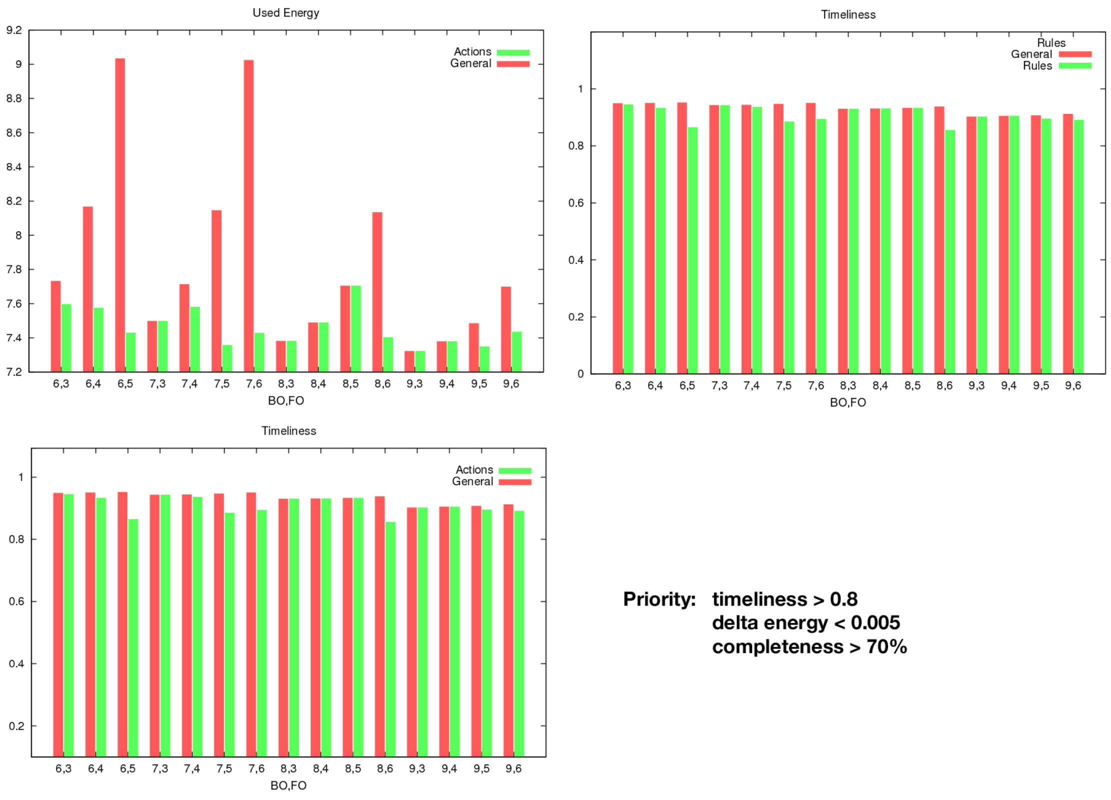


Figure 5.15: *Dimension values for different configuration as the timeliness has the priority.* $(Samplinginterval = 10ms)$

The timeliness is respected in all the different frameOrder-beaconOrder configurations for either the general approach and for the action-based approach. In the latter, though, the energy consumption is always lower compared to the general configuration, because at run-time, once the main timeliness constraint is respected, the algorithm successfully performs additional actions to reduce the energy consumption.



Priority: **timeliness > 0.8**
delta energy < 0.005
completeness > 70%

Figure 5.16: *Dimension values for different configuration as the timeliness has the priority.* $(Samplinginterval = 100ms)$

5.3 Comparison with other compression data algorithms

In the previous simulations we compared the results obtained running a general configuration and another implementing our adaptive-action approach. Both configurations, though, aggregate the input data using the algorithm described in Chapter 3. In this section we show the behavior of a general and an adaptive-action configuration, both implementing other compression data algorithms.

In particular, we want to analyze how the results change, using the data compression algorithms proposed by Schoellhammer et al. in [15] and by Lazaridis et al. in [18], mentioned in Section 2.2. The algorithm proposed in [15] receives as input a data stream, and starts considering the first two values z and v. The first is left as it is, the second is considered as a segment adding to the value y-coordinate a tolerance value $(y + e, y - e)$. It is, then, determined the sheaf of straight lines originated from z towards the superior and inferior limit of the segment generated by v. The algorithm keeps considering the next point, turning it as well into a new segment. This procedure stops when the segment generated by the new value is outside the sheaf of straight lines generated by z. In this case, z is sent to the base station.

The approach in [18], instead, considers as input a temporal series. It evaluates, value by value, the maximum and the minimum values of the series and, as long as the difference between these values respect the accuracy tolerance, the algorithm effects the average between the two values. Once a new value does not respect the accuracy tolerance anymore, the algorithm sends the mean between the minimum and maximum tolerance, previously calculated and re-starts the process from the last value received.

Concerning the interaction between one sensor and the base station, a comparison among the algorithms, in terms of accuracy and energy consumption is presented in [2]. The main results are shown in Table 5.2 and Figure 5.17.

	MAE	**MA%E**
Cappiello and Schreiber	0,0021	1,30%
Lazaridis et al.	0,0030	1,88%
Schoelhammer et al.	0,0010	0,63%

Table 5.2: *Comparison of correctness results*

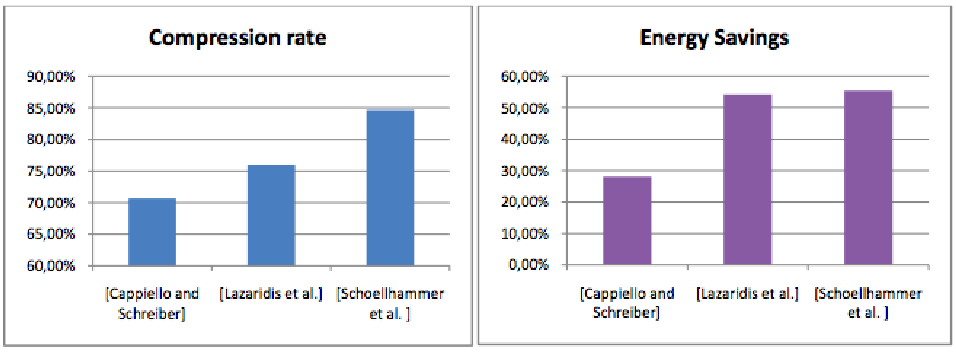


Figure 5.17: *Comparison among the three aggregation algorithms*

Our approach, though, considers a larger scenario, adding to the network several sensors, and adopts a MAC layer protocol to handle the communication among the sensors.

To compare the different algorithms behaviors, in terms of our quality dimension, we ran the same simulations of the previous section using the other data aggregation algorithms. The following figures highlight the differences between the algorithms.

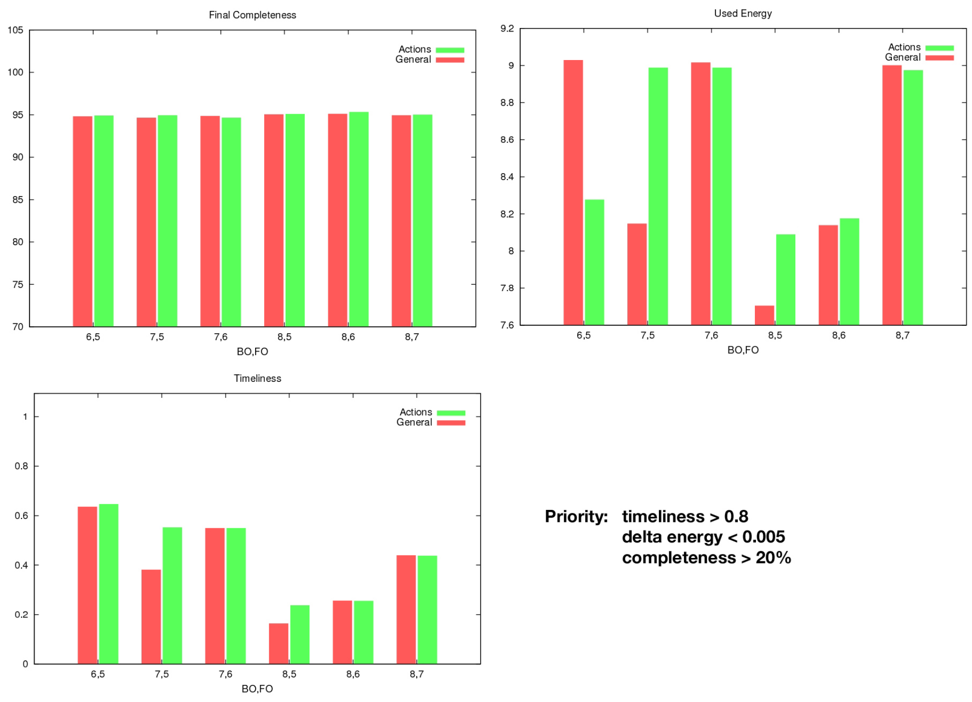


Figure 5.18: *Dimension values for different configuration as the timeliness has the priority, using the LTC algorithm. (Samplinginterval* $= 10ms$*)*

The main issue in the Lightweight Temporal Compression (LTC) algorithm proposed by Schoellhammer et al. is its stream, rather than the temporal series, approach. In fact, the algorithm keeps aggregating the input values as long as they respect the accuracy tolerance. In this way, if the input signal is not too irregular, the elapse of time of the aggregation grows affecting negatively the value of the timeliness. As shown in Figure 5.18, the timeliness value in the general configuration is always below the required threshold, and even the action-based approach is not able to enhance that value.

Since the value of completeness is strongly related to the MAC parameters and the network traffic rather than the aggregation algorithm itself, for this quality dimension the results satisfy the required threshold and are not very different from the ones obtained using our algorithm.

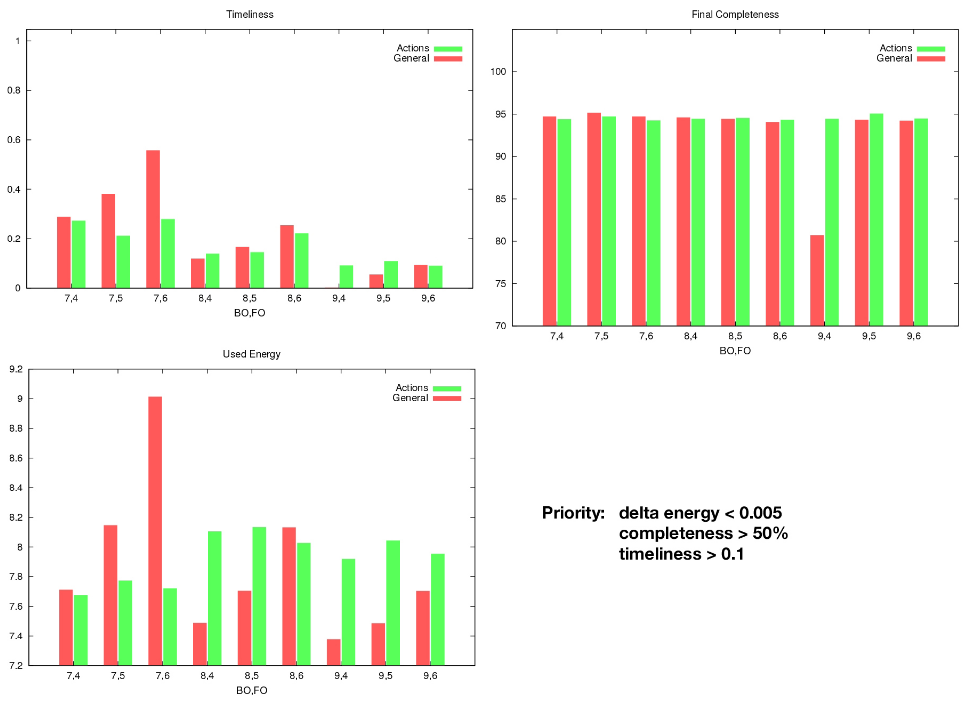


Figure 5.19: *Dimension values for different configuration as the energy saving has the priority, using the LTC algorithm. (Samplinginterval* $= 10ms$*)*

Finally, regarding the energy consumption we have a remarkable result. In fact, since the approach does not exploit the windowing approach, the action-based approach cannot use the window width increase action. With our algorithm, that action remarkably improved the energy consumption dimension, even whit a short sampling interval (see Figure 5.13). Figure 5.19 shows the former considerations.

On the other hand, the data compression algorithm proposed by Lazaridis et al., uses a temporal series approach as our algorithm does, and, for this reason, the obtained results reflects quite similarly our results, as shown in Figure 5.20 compared to Figure 5.14.

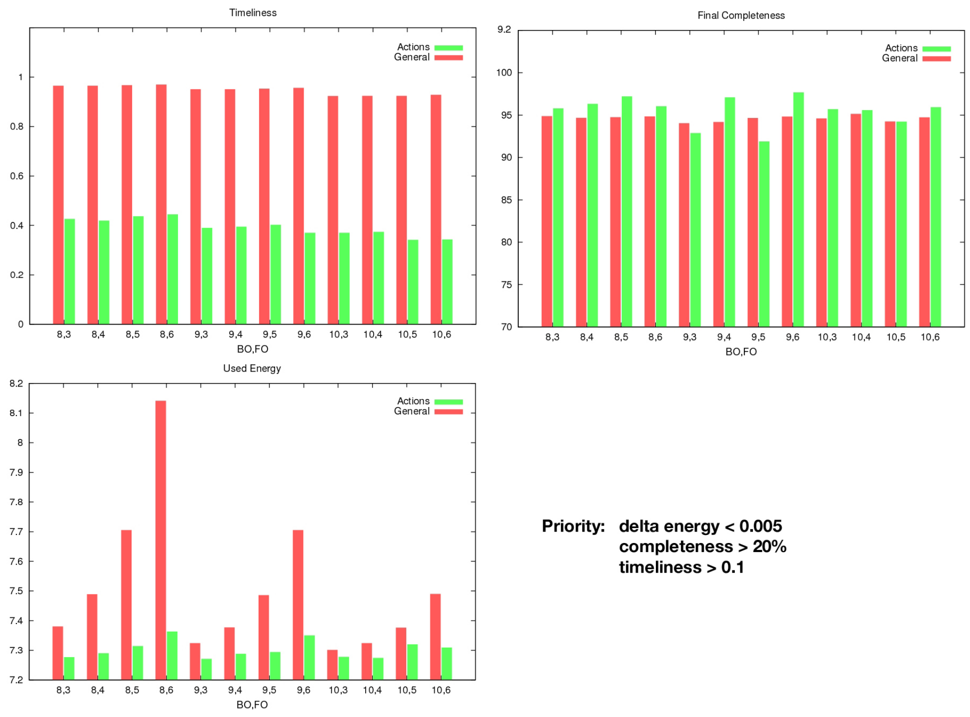


Figure 5.20: *Dimension values for different configuration as the energy saving has the priority, using the Lazaridis et al. algorithm. ($Samplinginterval = 100ms$)*

Nevertheless, it is important to underly that our aggregation algorithm provides the base station with additional information in each sent packet. In fact, the sensor classifies, at run-time, each packet in *aggregated*, *slow change*, *fast change* and *outlier*, providing, in this way, the base station with not only the value but also its trend nature compared to the one of the previous values in window.

Figure 5.21, 5.22 and 5.23 clarify this concept. We take into consideration an input signal with some outliers and we run a short simulation (100 seconds) for each algorithm. All the figures share the same pattern, in the top part there is the input signal (green dots) and the values received at the base station (blue circles). In the bottom part, instead, it is reported the input signal again and the interpolation made using the received values (blue line).

Nevertheless, our algorithm provides more information. In Figure 5.23, in fact, we can see also purple circles, which stand for values detecting a slow change in the trend, and more important, red circles that stand for outlier values. Thanks to this information, the base station is able to rebuild the input signal more precisely respect the two other approaches (see Figure 5.23b), by pointing out the outlier values from the signal trend. Notice how the input signal generated by Castalia has a "step" trend and only our algorithm menages to detect it.

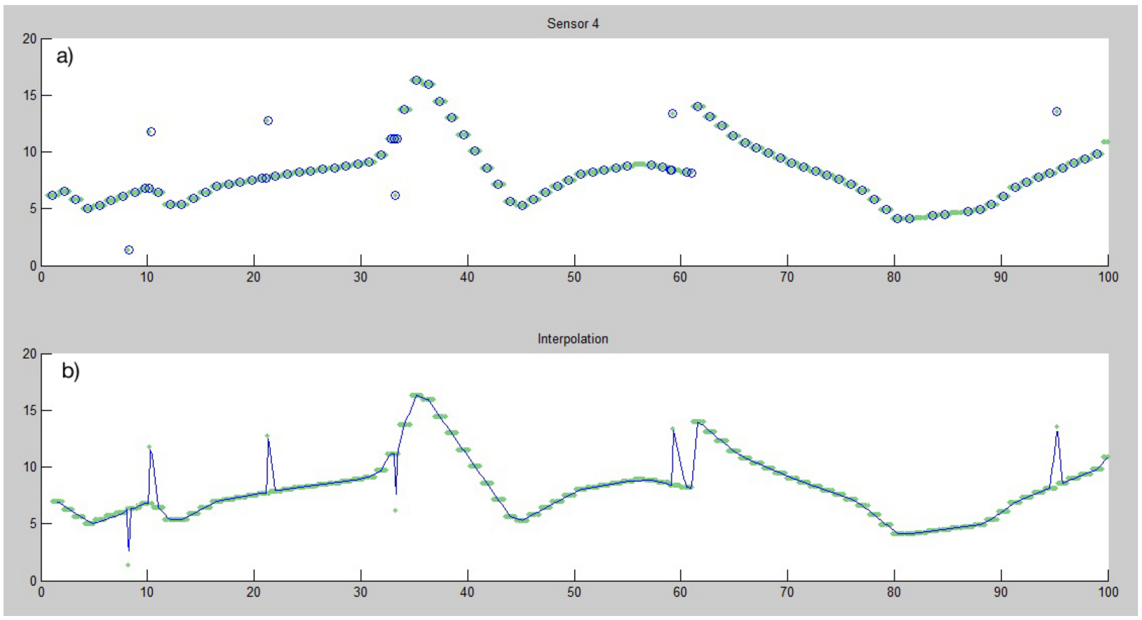


Figure 5.21: *a) Original signal (green) and value received by the base station (blue); b) Original signal (green) and interpolation of received data (blue) using Schoellhammer et al. algorithm*

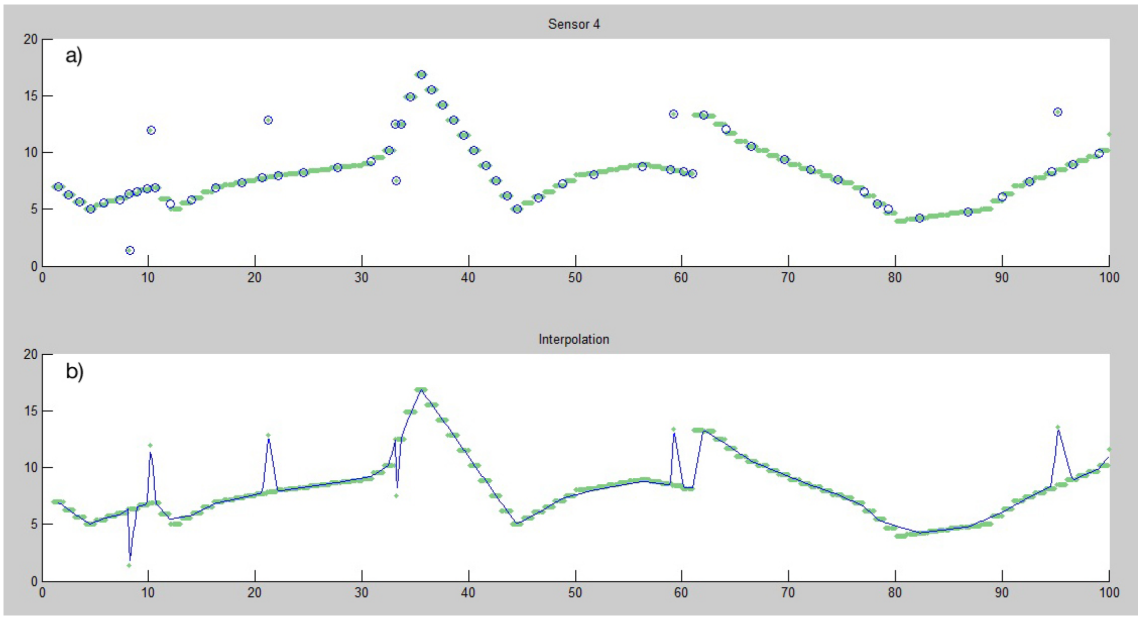


Figure 5.22: *a) Original signal (green) and value received by the base station (blue); b) Original signal (green) and interpolation of received data (blue), using Lazaridis et al. algorithm*

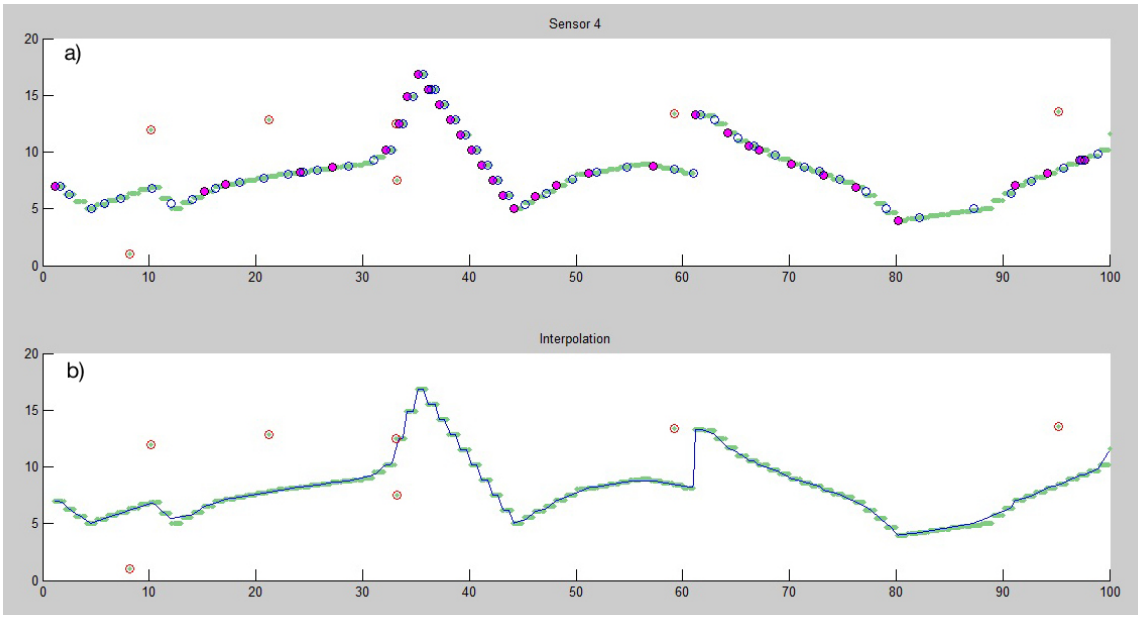


Figure 5.23: *a) Original signal (green) and value received by the base station; b) Original signal (green) and interpolation of received data, using our algorithm*

6 Conclusions

Considering the obtained results shown in the previous chapter we can state that adopting the action-based approach results always in an improvement of the overall quality dimensions. The graphs in Chapter 5, in fact, illustrate how, given a set of priority, the algorithm is able to satisfy them as much as possible. This approach best fits those scenarios in which one quality dimension is more important then the others. To better understand the obtained results we need to analyze them separately, grouped by the dimension with the highest priority.

Completeness. First of all, it is important to notice that, without our functionalities, the completeness is always below the required threshold and, in particular, it is never above the 94%, whilst, with our algorithm, the completeness is always between 99% and 100%, even when the network traffic is elevated.

In this scenario, the second priority concerns the timeliness. Even if the threshold is respected, the timeliness value in the general configuration results to be always higher compared to the one obtained with our approach. This, of course, is due to the low timeliness contribution given by the retrieved packet, needed to satisfy the completeness (see Section 4.3.2).

The satisfaction of the completeness is reached at the expense of the energy consumption. Especially in the high packet rate case, the approach needs to increase as much as it can the active period in order to reduce the packet loss, resulting in a more expensive configuration in terms of energy. When the packet rate is lower, instead, the trend of the energy consumption is closer to the one obtained with the general configuration, and sometimes it is even lower. This happens because, since the network traffic is lower, the packet retrieval functionality is sufficient to guarantee the completeness, and thus the algorithm is able to increase the beacon period (BO) in order to reduce the energy consumption.

Energy. The results concerning the energy saving are remarkable. In fact, as Figure 5.13 and 5.13 illustrate, the energy consumption using our approach is always much lower than the consumption in the general configuration, even when the packet rate is high. Nevertheless, the completeness is always higher than the required threshold. The last priority, in this case was the timeliness. As we can see from the figures, the timeliness value is always lower than the one in the general configuration, but still, it always respects the threshold requirement. This occurs because, in order to reduce the power consumption, the base station increases the beacon order, affecting, in this way, the timeliness value (see Section 4.3.2).

This important result is achieved thanks to the potentiality of our data aggregation algorithm, described in Chapter 3. In fact, its *windowing approach*, allows us to enlarge the window of compression, in terms of number of values considered during the compression. In this case, since the input signal we produced through Castalia is not too irregular, the base station decides that, if the number of packet sent per window is below a certain percentage (in our case 20%), it can try to enlarge the window. In this way, the higher the number taken into account for the compression, the lower the number of packets sent to the base station. Obviously, we set, at design time, a maximum value for the window width, according to the timeliness requirement, in order to respect the threshold as much as possible.

It is important to underly that, even if the compression takes into consideration a larger number of values, the aggregated value is still accurate since it must respect the algorithm's ε_{acc} given at design time.

Timeliness. Timeliness dimension is the easiest to satisfy. It is important to notice, though, that, as both the general and the action-based configuration fully satisfy the timeliness and completeness requirements, our approach manages to remarkably reduce the energy consumption in both of the two different packet rate scenarios (see Figure 5.15 and 5.15).

Giving the same initial configuration, thus, using our approach leads into two different scenarios:

a) The approach satisfies the major quality requirement and enhances its value respect the general configuration, improving, whenever it is possible the other requirements.

b) Both the general and the action-based approach equally satisfy the major quality requirement, but our approach remarkably enhances also the other metrics according to the priority threshold.

Even if the enhancements in certain configuration are not extremely remarkable (for instance and enhancement of just 0.2 J for the energy dimension), it is necessary to notice that with our approach, the quality requirements are constantly under control, and in case an external factor affects somehow the network (for instance, an unexpected variation in the wireless channel), our approach is able to detect it and to change its configuration in order to constantly satisfy the quality requirements.

Finally, it is important to notice that good performances of our approach are strictly related to the parameters setting phase in terms of both the algorithm parameter such as the ones mentioned in Table 5.1 and the set of priority given as input. In fact, as mentioned in Section 4.3.1, the set of priorities must be formed by: a) a number of priorities with low thresholds that reflects a default and safe configuration; b) the priorities with the threshold we aim to satisfy. This is necessary because, omitting the first set of priority, if the first threshold to be enhanced is too strict, the system tries in vain to satisfy it, leading the network into a bad configuration.

References

[1] L. Mottola and G. P. Picco. Programming wireless sensor networks: Fundamental concepts and state of the art. *ACM Computing Surveys. Volume 43, Issue 3*, 2011.

[2] C. Cappiello and F. A. Schreiber. Quality- and energy-aware data compression by aggregation in wsn data streams. *Personal Communication*, 2010.

[3] F. Rossini and M. Rugginenti. Implementazione e analisi dei risultati per un algoritmo di compressione dati su sensori wireless. Bachelor's Thesis, Politecnico di Milano, 2009.

[4] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communication Mag. 40, 8*, 2002.

[5] L. Bertossi, F. Rizzolo, and Lei Jiang. Data quality is context dependent. *BIRTE 2010: 52-67*, 2010.

[6] A. Varriale. Algoritmo di compressione dati di tipo quality-aware per la compressione di flussi di dati in una wsn. *Progetto di Technologies for IS*, pages 4–7, 2008.

[7] N. Kimura and S. Latifi. A survey on data compression in wireless sensor networks. *ITCC*, pages 8–13, 2005.

[8] L. Golab and M. T. Özsu. Issues in data stream management. *SIGMOD Record 32*, pages 5–14, 2003.

[9] J. Chou, D. Petrovic, and K. Ramchandran. A distributed and adaptive signal processing approach to reducing energy consumption in sensor networks. In *INFOCOM*, 2003.

[10] J. Gama and M. M. Gaber. Learning from data streams. *Springer*, 2007.

[11] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *OSDI*, 2002.

[12] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Compressing historical information in sensor networks. In *SIGMOD Conference*, pages 527–538, 2004.

[13] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robus communication paradigm for sensor networks. In *MobiCom*, pages 56–67, 2000.

[14] R Cardell-Oliver. Rope: a reactive, opportunistic protocol for environment monitoring sensor networks. In *EmNetS-II*, 2005.

[15] T. Schoellhammer, E. Osterweil, B. Greenstein, M. Wimbrow, and D. Estrin. Lightweight temporal compression of microclimate datasets. In *LCN*, pages 516–524, 2004.

[16] Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang. Multi-dimensional regression analysis of time-series data streams. In *VLDB*, pages 323–334, 2002.

[17] D. Tulone and S. Madden. Paq: time series forecasting for approximate query answering in sensor networks. In *EWSN*, pages 21–37, 2006.

[18] I. Lazaridis and S. Mehrotra. Capturing sensor-generated time series with quality guarantees. In *ICDE*, pages 429–439, 2003.

[19] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proc of the 21st Int. Conf. on Computer Communications (INFOCOM)*, 2002.

[20] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proc of the 2nd Int. Conf. on Embedded Networked Sensor System (SenSys)*, 2004.

[21] T. Van Dam and K. Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proc of the 1st Conf. on Embedded Networked Sensor System (SenSys)*, 2002.

[22] V. Rajendran, J. Obraczka, and J. J. Garcia-Luna-Aceves. Energy-efficient, collision-free medium access control for wireless sensor networks. *Wireless Networks 12*, page 1, 2006.

[23] Y.-C. Tu, M. Hefeeda, Y. Xia, S. Prabhakar, and S. Liu. Control-based quality adaptation in data stream management system. In *DEXA*, pages 746–755, 2005.

[24] H. Hu, C.-H. Jiang, K.-Y. Cai, and W. E. Wong. A control-theoretic approach to qos adaptation in data stream management systems design. In *COMPSAC (2)*, pages 237–248, 2007.

[25] D. J. Yates, E. M. Nahum, J. F. Kurose, and P. J. Shenoy. Data quality and query cost in pervasive sensing system. *Pervasive and Mobile Computing 4 (6)*, pages 851–870, 2008.

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

[27] M. A. Sharaf, J. Beaver, A. Labrinidis, and P. K. Chrysanthis. Balancing energy efficiency and quality of aggregate data in sensor networks. *BLDB J. 13 (4)*, pages 384–403, 2004.

[28] I. Akyildiz and I. H. Kasimoglu. Wireless sensor and actor networks: Research challenges. *Ad Hoc Netowkrs Journal 2*, page 4, 2004.

[29] T. Liu and M. Martonosi. Impala: A middleware system for managing autonimc, parallel sensor systems. In *Proc of the 9^{th} SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 2003.

[30] R. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: Modeling and analysis of a three-tier architecture for sparse sensor networks. *Elsevier Ad Hoc Netowkrs Journal 1*, pages 2–3, 2003.

[31] F. A. Schreiber. Data management in pervasive systems. *On line article: http://www.scitopics.com/Data Management in Pervasive Systems.html*, 2008.

[32] ISO ISO/IEC Guide 99-12:2007. International vocabulary of metrology basic and general concepts and associated terms.

[33] M. A. Yigitel, O. D. Incel, and C. Ersoy. Qos-aware mac protocols for wireless sensor networks: A survey. *Computer Networks, Volume 55, Issue 4*, 2011.

[34] The IEEE 802.15.4 standard. *standards.ieee.org/getieee802/download/802.15.4-2006.pdf*, 2006.

[35] P. R. Grassi and F. A. Schreiber. Methodologies and models for the analysis and control of the stability of context-aware self-adaptive systems. *Personal Communication*, 2011.

[36] A. K. Dey. Understanding and using context. *Personal Ubiquitous Computing*, 5(1):4–7, January 2001.

[37] A. Boulis. Castalia. a simulator for wireless sensor networks and body area networks - user manual., 2010.

[38] M. Zuniga and B. Krishnamachari. Analyzing the transitional region in low power wireless links. In *First IEEE International Conference on Sensor and Ad Hoc Com-*

munication and Networks (SECON), 2004, pages 517–526, Santa Clara, CA, October 2004.