DOTTORATO DI RICERCA IN

Ingegneria elettronica

Ciclo XXV

Settore Concorsuale di afferenza: 09/E3 Elettronica

Settore Scientifico disciplinare: ING-INF/01 Elettronica

TITOLO TESI

Ultra-low power WSNs: distributed signal processing and dynamic resource management

Presentata da: Carlo Caione

Coordinatore Dottorato **Relatore**

Prof. Alessandro Vanelli-Coralli **Prof. Luca Benini**

Esame finale anno 2013

To my loving parents, my sister and Giulia

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The emerging field of wireless sensor networks (WSNs) combines sensing, computation, and communication into a single tiny device and while the capabilities of any single device are minimal, the composition of hundreds or thousands of these devices offers radical new technological possibilities.

The real power behind wireless sensor networks is the possibility to deploy large numbers of tiny nodes that assemble and configure themselves in such a way that, usually through advanced mesh networking protocols, they can communicate each other by hopping data from node to node in search of their destination.

These devices can be used for a lot of usage scenarios: from the monitoring of environmental conditions to the monitoring of the health of structures or equipment. Even though we usually refer to these networks as wireless sensor networks, emphasizing the presence of the sensors, the nodes can also be used to pilot actuators extending control from cyberspace into the physical world.

The most common and straightforward application for wireless sensor network technology is to monitor a large and remote environment, sensing and reporting low-frequency data towards a central collector. For example a large building could be easily monitored for temperature, humidity or human presence by hundreds of sensors interconnected by a wireless connection able to trigger in real time the HVAC system for maximizing the comfort of the people inside the building. Unlike traditional wired systems, the deployment costs would be minimal and definitely less intrusive. Using WSNs for monitoring there is no need to wire long cables through walls and conduits but nodes can be easily placed inside pre-

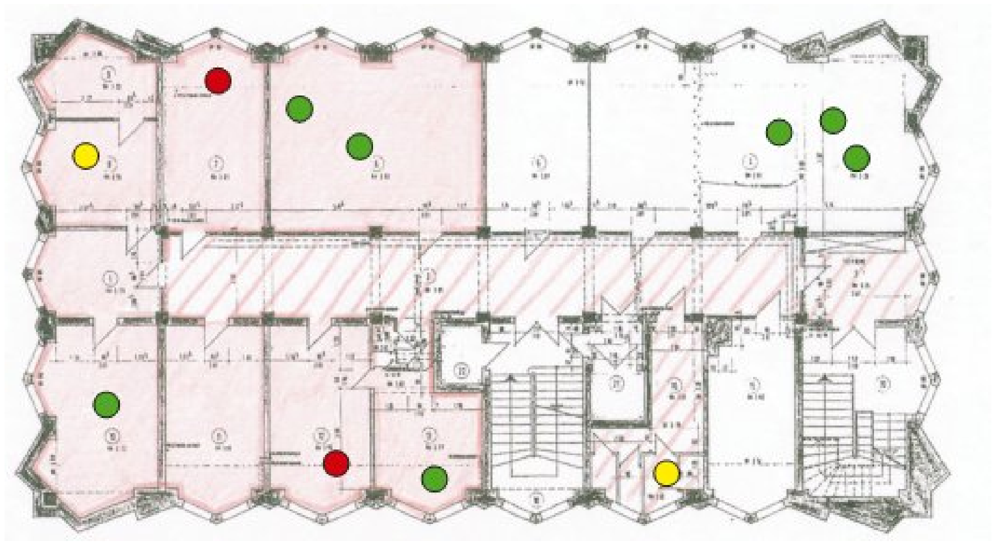


Figure 1.1: Possible deployment of ac-hoc wireless embedded network for smart-building monitoring. Sensors detect temperature, humidity and human presence at hundreds of points across the building and communicate their data over a multi-hop network for analysis

existing buildings without any structural modification. Moreover the network could be incrementally extended by simply adding few more devices without requiring the network reconfiguration.

Another clear advantage in using WSNs for data monitoring and reporting, in addition to saving on installation costs, is the capability of the network to self heal and self adapt to changing environmental conditions. In WSNs, mechanisms do usually exist that can quickly respond to changes in network topology. For example in case of failure of one or few nodes (or in case one node is moved from its original position) the network can reconfigure itself continuing to ensure connection and a reliable data reporting. This behavior is still not feasible with wired networks where the network topology is fixed and not adaptable to changing conditions.

Even though WSNs deal with wireless connection (such as cell phones, tablet, notebooks, ...) they do not rely on expensive network infrastructures. WSNs use small and low-cost embedded devices for a wide range of applications and do not rely on pre-existing infrastructures for their installation and use. In fact, unlike traditional devices, wireless sensor nodes do not directly communicate with a near high-power central control point but they only communicate with their local close peers. A real fixed network infrastructure does not exist, but each node becomes part of the network infrastructure itself. This peculiar network structure requires new protocols to manage the communication among nodes, to shuttle data between the thousands of tiny embedded devices in a multi-hop fashion and eventually self-repair and self-reorganize in the eventuality of a network or point failure.

Another difference with respect to the classical wireless networks is that the real strength of WSNs is in the large number of potential nodes in the network. While for centralized network structures (such as the cell phone network) the number of devices connected to same cell could be a problem when too many devices are active in a small area, at the opposite the interconnection of a WSN just becomes stronger as nodes are added.

Nowadays there is an increasing interest in WSNs and a lot of ongoing research on data aggregation [Nie and Li, 2011][Younis et al., 2006], ad hoc routing [Kassim et al., 2011][Saleem and Farooq, 2007][Lambrou and Panayiotou, 2009][Gajurel and Heiferling, 2010] and especially distributed signal processing within WSNs [Banitalebi et al., 2011][Bal et al., 2009][Conti et al., 2004].

In all the papers in literature it is clear as the main challenge in wireless sensor network deployment is to cope with the resource constraints of the devices. There are several constraints that we have to deal with when working with WSNs and wireless nodes: small memory availability and computational power in embedded processors used for wireless nodes, constraints derived from the vision that these devices have to be small and inexpensive, etc...

Nevertheless it is well known the most difficult resource constrain to meet is the power consumption. This is a straightforward consequence of the reduction of the nodes size: with decreasing in physical dimensions there is a proportional decreasing in the energy capacity of the device. These problems related to energy capacity have also a direct impact on the architectural choices, creating in turn new computational and storage limitations.

While many devices try to reduce their power consumption through the use of specialized communication hardware in ASICs providing low-power implementation of the communication protocols, this is not feasible for WSNs that have to be as general and flexible as possible.

Therefore, while traditional networks and devices aim to achieve high quality of service (QoS) provisions and low power consumption through specialized hardware and always increasing the energy capacity, sensor network protocols must focus primarily on power conservation maintaining small sizes and low-cost design. They must have built-in trade-off mechanisms that give the end user the option of prolonging network lifetime at the cost of lower throughput or higher transmission delay.

In fact, power saving is generally achieved by reducing radio communication through mainly three approaches: (1) using power-aware network protocols [Li et al., 2011]; (2) duty cycling [Wang et al., 2009]; and (3) in-network/in-node processing and compression [Fasolo et al., 2007];

Duty cycling schemes define coordinated sleep / wakeup schedules among nodes in the network. On the other hand, in-network / in-node processing consists in reducing the amount of data to be transmitted by means of compression and / or aggregation techniques. Due to limited processing and storage resources of sensor nodes, data compression in sensor nodes requires the use of ad-hoc algorithms and tools.

With the increasing in network size and number of nodes, direct consequence of having small and low-cost devices, compression and aggregation have become a need in WSNs. The traditional approach to sense and measure environmental informations, e.g. temperature and humidity, through uniform sampling and then reporting data to a fusion center is not energetically sustainable anymore due to the enormous quantitative of data generated by sensor nodes. Fortunately not all the information is really needed since a lot of redundant information is present in data acquired from sensor nodes in a WSN, especially if we are able to transform these signals to some suitable basis.

Among all the frameworks and techniques for data compression and aggregation, in this thesis we deal with Compressive sensing (CS), also called compressed sensing and Sub-Nyquist sampling, that has a surprising property that one can recover sparse signals from far fewer samples than it is predicted by the Nyquist-Shannon sampling theorem [Donoho, 2006][Haupt et al., 2008].

Samples obtained with CS contain a little redundancy in the information level, and the sampling process can accomplish two functions, aggregation and compression, sometimes simultaneously. CS trades off an increase in the computational complexity of post-processing against the convenience of a smaller quantity of data acquisition and definitely a lower demands on the computational capability of the sensor node. This is due to the fact that CS directly acquires the compressed version at sampling time so that no explicit compression is really required.

In this context the main contributions of this work are: (1) an investigation of CS as data aggregation technique in WSNs; (2) the extension of CS to ensemble of nodes in sensor networks; (3) an analysis of trade-offs between power consumption and reconstruction quality for CS and Distributed CS (DCS) when COTS devices are used as hardware for compression; (4) an exploration of the reconstruction performance of CS for highly sub-sampled signals in WSNs; and (5) a comparison between CS and another special technique used for data compression in sensor networks.

This thesis is organized in 5 Chapters. In *Chapter 2* is an introduction on wireless sensor networks with particular focus on the energy problem related to the usage of resource-constrained devices. An overview of different kind of power supplies and batteries for wireless nodes is given and then the techniques for reducing energy consumption, saving on radio and communications, are introduced. Finally the processing subsystem is addressed and high-performance 32-bit micro-controllers are investigated when used for WSNs. *Chapter 3* opens with middlewares and their usage in WSNs. Afterwards in-network aggregation and CS are introduced providing the mathematical background needed to understand the remaining part of the thesis. In the final part of the chapter, a real case of the usage of CS for data aggregation in WSNs is presented. In *Chapter 4* CS is extended to signal ensembles and the two major techniques for CS exploiting multi-dimensionality and multi-signals correlation are introduced, namely: distributed CS (DCS) and Kronecker CS (KCS). These two techniques are then compared against a common data-set. Finally the usage of CS is investigated when signals are sampled at sub-Nyquist frequency and the reconstruction issues are addressed. In *Chapter 5* a comparison between CS and an other promising technique when dealing with sub-Nyquist sampling rate is given. In the chapter, *Chapter 6*, are the conclusions.

Chapter 2

Wireless Sensor Networks

A Wireless Sensor Network (WSN) is a computer network formed by a large number of little and inexpensive wireless devices (the sensors or sensor nodes) that cooperate to monitor the environment using transducers and, in special cases, operate on the environment by means of actuators [Zhao, 2004][Al-Karaki and Kamal, 2004][Goyal and Tripathy, 2012].

Recent technology advances have enabled the possibility to have highly integrated devices with processors and radio systems that can be easily embedded in little multi-purpose and easily programmable devices. In general these devices have on-board also one or more sensing units and an embedded battery or special harvesters to gather energy from the environment. The sensor nodes are usually spread in harsh environments without any predetermined infrastructure with the aim to create a cooperative network of sensors to achieve a common task, usually sensing and reporting environmental data. As a result of the harsh conditions which the nodes are exposed and since the nodes are subject to failures and battery exhaustion, sensor networks must be fault-tolerant and the network must be able to self-heal and self-adjust to configuration changes.

2.1 Applications of wireless sensor networks

As seen in the literature on sensor network architectures, unlike general purpose networks, sensor networks have to make efficient use of limited resources in accomplishing their single goal [Pereira et al., 2011]. Since each network aims at a particular focused objective, it is expected that sensor networks with different goals will be designed differently, with features designed specifically for one application.

2.1.1 Data collection

In this scenario researchers want to collect data and sensor readings from a set of sensors spread in the environment over a period of time in order to detect trends and interdependencies, to control their status and position or to send commands. Data is collected at regular intervals and the nodes have a fixed position in the measurement field. Usually data collection is performed over a long period (months or years) to look for long-term and seasonal trends.

For environmental data collection it is usually not important to sample data at high frequency since the variables to be measured have slow variation in time (the typical environment parameters being monitored, such as temperature, light intensity and humidity, do not change quickly enough to require higher reporting rates) thus these networks generally require very low data rates and extremely long lifetimes [Tovar et al., 2010][Pendock et al., 2007].

The network is characterized by having a large number of nodes continually sensing and transmitting back to a base station where data is permanently stored. Environmental monitoring applications do not have strict latency requirements since in general the data is collected for future analysis, not for real-time operations [Cheng et al., 2011].

In this context in which the topology and the nodes distribution is relatively constant in the environment, it is not needed to develop complex or optimal routing strategy, instead it is more convenient to calculate the optimal routing topology outside the network and then communicate the necessary information to the nodes as required. For this reason, environmental applications typically use tree-based routing topologies where each routing tree is rooted at high-capability nodes. Each node is responsible for forwarding data to its own parent node up the tree-structure until it reaches the sink. In order to permit the flowing of the data from each node toward the collector sink, each communication event must be precisely scheduled. This could be a problem when network is subject to duty cycling to save energy since the sensors remain dormant the majority of the time and they have to wake to transmit or receive data synchronously with children and parent nodes. If the precise schedule is not met, the communication events will fail.

Thus the most important characteristics of the environmental monitoring requirements are long lifetime, precise synchronization, low data rates and relatively static topologies. In this work our focus in mainly on this type of applications.

2.1.2 Events detection

Detect an event is important for security monitoring (IDS) or military applications where network is composed by nodes, placed at fixed locations, that continually monitor the environment to detect an anomaly or report a sporadic event. Differently from sensor nodes used in data collection, for events detection the nodes are not required to gather any data but each node has to frequently check the status of its sensors for anomalies or specific patterns defining an event, reporting back to the collector only in case something suspicious or the event of interest is detected [Bahrepour et al., 2009][Shu and Liang, 2006].

Data collection and event detection are two classes of applications that are very different. While for wireless multi-hop data collection an operational lifetimes on the order of a year or more is required, for the wireless multi-hop event detection sensor networks few days or weeks could be enough. This is because whereas data collection may allow sensor nodes to sleep most of the time, event detection requires that sensors are vigilant most of the time.

In networks for events detection reducing the latency of an alarm transmission is significantly more important than reducing the energy cost of the transmissions. Once detected, a security violation or an event must be communicated to the base station immediately. The latency of the data communication across the network to the base station has a critical impact on application performance and this means that network nodes must be able to respond quickly to requests from their neighbors to forward data. Obviously reducing the transmission latency leads to higher energy consumption because routing nodes must monitor the radio channel more frequently.

2.1.3 Location-tracking

The goal is to trace the roaming paths of moving objects in the network area. Using wireless sensor networks, objects can be tracked by simply tagging them with a small sensor node. The sensor node will be tracked as it moves through a field of sensor nodes that are deployed in the environment at known locations.

Differently from the previous cases, the topology of the network is composed by a stable network formed by nodes at fixed locations and the connectivity of the mobile nodes continually changing. Being known the geographical locations of the monitoring fixed nodes, it is straightforward to track the object moving inside the network [Chen and Feng, 2009][Caceres et al., 2009].

Since the network has to be able to efficiently detect the presence of new nodes that enter the network and to track the existing ones moving within, energetically these sensor networks resemble more the networks for events detection than the networks for data collection. Even in this network the latency and the synchronization among nodes have a critical impact on the performance of the application.

2.1.4 Hybrid networks

In general there are scenarios involving all the aspects of all three categories. This is the case of a network that is able to switch from events detection to data collection when a specific event is detected. In case of smart building monitoring we can have a monitoring network that is able to gather information about the comfort in a specific room only if the human presence is detected inside the monitored environment.

2.2 The energy problem

Each sensor node is basically a tiny device composed of three basic units: a processing unit with very limited memory and computational power, a sensing unit for data acquisition from the surrounding environment and a communication unit, that is usually a radio transceiver to transmit data to the collecting sink. Typically, nodes are powered by small batteries which cannot be generally changed or recharged. Although there have been significant improvements in processor design and computing, the battery technology still struggles to keep the pace, making the energy the most critical resource in WSNs. As a consequence of the energy constraint, a new performance metric, namely, the network lifetime, has become a vitally important benchmark for wireless sensor networks [Dong, 2005][Wang and Xiao, 2005][Rhee et al., 2004].

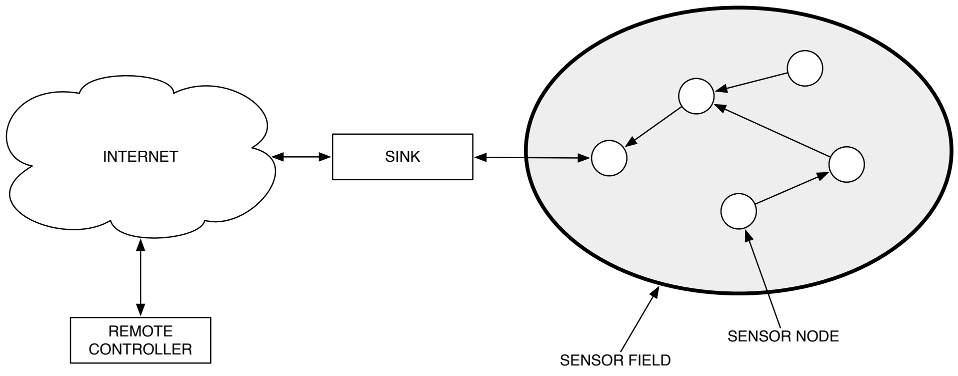


Figure 2.1: Typical sensor network architecture

For all the scenarios seen in Section 2.1 we ideally aim to have nodes active and monitoring, unattended, for months or years. It is important to notice how the nodes are all interconnected each other and, especially for tree-based routing topologies, the correct delivery of data depends on all the nodes on the path toward the sink. Thus in many deployment it is not really important the average node lifetime but rather the minimum node lifetime, since the failure of a single node can determine the failure of the whole network.

In the analysis for energy consumption we will refer to a well-defined network model for a data collection network. The network consists of one sink and a large number of deployed sensor nodes. Data are transferred from the nodes to to the sink through a multi-hop transmission. As seen in Section 2.1.1 the network is intended to be static with a tree-based routing protocols whose routes are calculated off-line. On the other side we can think to the typical sensor node as composed of four main components:

1. POWER SUPPLY SUBSYSTEM to power the whole system
2. SENSING SUBSYSTEM including one or more sensor (with associated ADCs) for data acquisition and digitalization
3. PROCESSING SUBSYSTEM composed by a micro-controller with memory for data storage
4. RADIO SUBSYSTEM for wireless data communications

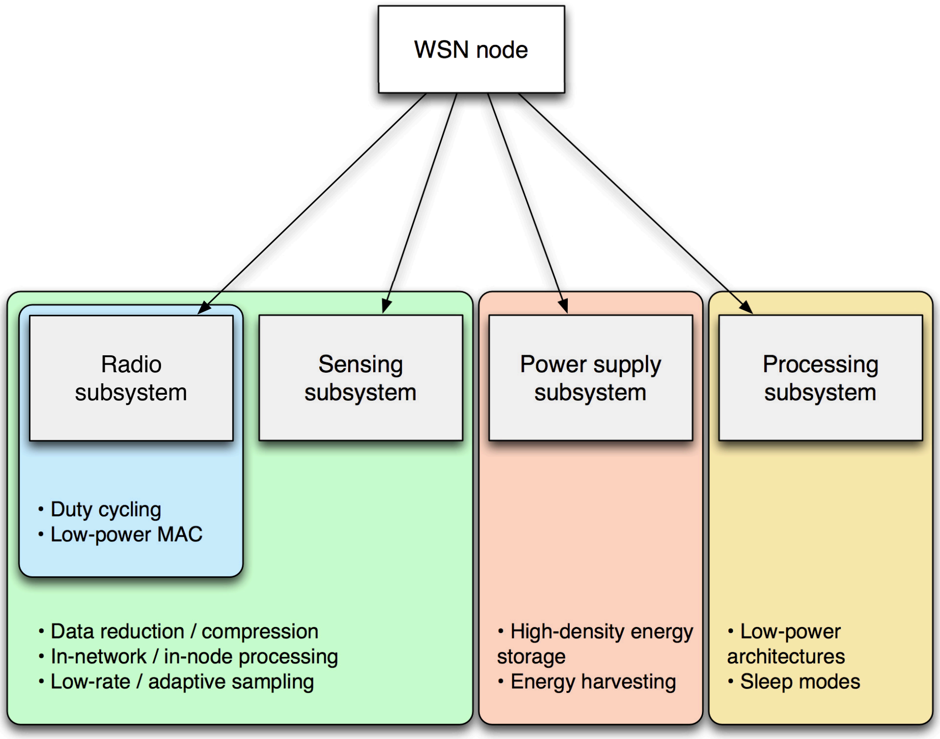

WSN node
Radio subsystem
Sensing subsystem
Power supply subsystem
Processing subsystem
• Duty cycling
• Low-power MAC
• Data reduction / compression
• In-network / in-node processing
• Low-rate / adaptive sampling
• High-density energy storage
• Energy harvesting
• Low-power architectures
• Sleep modes

Figure 2.2: Wireless sensor node power model. For each subsystem some of the major techniques for power consumption reduction are listed

Depending on the specific applications, the sensor nodes may also include additional components such as actuators or GPS. However, as these components are optional, they are occasionally used and are not taken into consideration in this thesis.

Several approaches have to be exploited, even simultaneously, to reduce power consumption or increase lifetime in wireless sensor networks as better depicted in Figure 2.2. Some techniques are just related to one specific subsystem like energy harvesters for power supply or the low-power architectures for the processing subsystem. The sensing subsystem and the radio subsystem are strictly interrelated: the first one provides data to be sent using radio transmissions thus a reduction in the data size reflects in a reduction of power spent in transmission. Specific techniques, however, are specific for the radio subsystem such as duty cycling or the usage of low-power MAC protocols.

In the next paragraphs and chapters we are going to focus on each subsystem and on the major techniques related to that subsystem for energy consumption reduction.

2.3 Battery and power supply

Overall battery capacity is measured in milli-Amp-hours (mAh) or Amp-hours (Ah). In theory a rating of 3 Ah means the battery can supply 3 A for 1 hour or 1 A for 3 hours. In practice this in not always true. Due to battery chemistry, voltage and current levels vary depending on how the energy is extracted from a battery. Sometimes the battery is not quite empty, but there is insufficient potential energy in the battery to get the remaining charge out, for this reason a battery is said to be empty when the potential or voltage drops below a certain level. Typically, for a 1.5 V cell such as a AA-sized cell, the battery is considered empty when the voltage drops to around 0.9 V.

Even thought the technology is trying to scale down size and weight of the batteries used in sensor networks, with the actual technology batteries still are a significant fraction of the total size and weight of the nodes. Due to this strict relation between size and energy storage capacity we can characterize the batteries namely by energy density. The energy density is a term used for the amount of energy stored in a given system or region of space per unit of volume. On the other hand the power density is the amount of power per unit of volume.

Battery Type	Wh/Kg	J/Kg	Wh/L
Lead-acid	41	146000	100
Alkaline long-life	110	400000	320
Carbon-zinc	36	130000	92
NiMH	95	340000	300
NiCad	39	140000	140
Lithium-ion	128	460000	230

Table 2.1: Energy and power density for the most common battery technologies for WSNs

While an ideal energy reservoir should offer both a high energy and a high power density, the typical battery usually features high energy density but a limited power density.

The three main technologies traditionally used in WSNs are Alkaline Batteries, Lithium Batteries and Nickel Metal Hydride (NiMH) Batteries [McDowall, 2000]. In Table 2.1 are reported the energy and power density for the most common battery technologies.

The Alkaline batteries are far the most common type of household battery, and they are very good for a variety of electronics applications. Alkaline batteries are low cost, widely available, and are ideal for low current applications at room temperature. However, they have two major shortcomings: the energy capacity is highly dependent on temperature and they don't tend to work as well under high current draws.

For the Lithium batteries there are many varieties. Although it is difficult to put them all in the same class or use the same model to describe their behavior, there are some common characteristics. For the sake of this discussion, we will focus on the iron-disulfide formula in particular because it is the most common Lithium battery available in AA sizes. This type of battery is regularly used as a replacement for alkaline batteries where longer life, higher current draw, or improved temperature performance is needed. Even thought Lithium iron-disulfide batteries are not rechargeable, the big benefit to Lithium disposable batteries is that they do much better under low temperature and high current rate conditions.

NiMH, unlike its cousin, the Nickel Cadmium (NiCAD) battery, tends to maintain its characteristics through many recharges. Its biggest shortcomings are that it is relatively heavy and that it has a lower energy density, meaning that a typical AA-sized NiMH battery will start at 1.2 V instead of the conventional 1.5 V for an alkaline AA battery.

Recently new technologies have been proposed to try to further shrink down size and at the same time to increase the energy density of the batteries commonly used in very small size portable devices such as wearable and sensor nodes. These new batteries are based on new technologies like thin-film Lithium-ion or Lithium-polymer cells [Owen, 1996]. They are similar to lithium-ion batteries, but they are consisting of thin materials, some only nanometers or micrometers thick, which allows the finished battery to achieve millimeters thickness. The problem with the thin-film batteries is that these batteries are too small for long lasting operations in sensor networks.

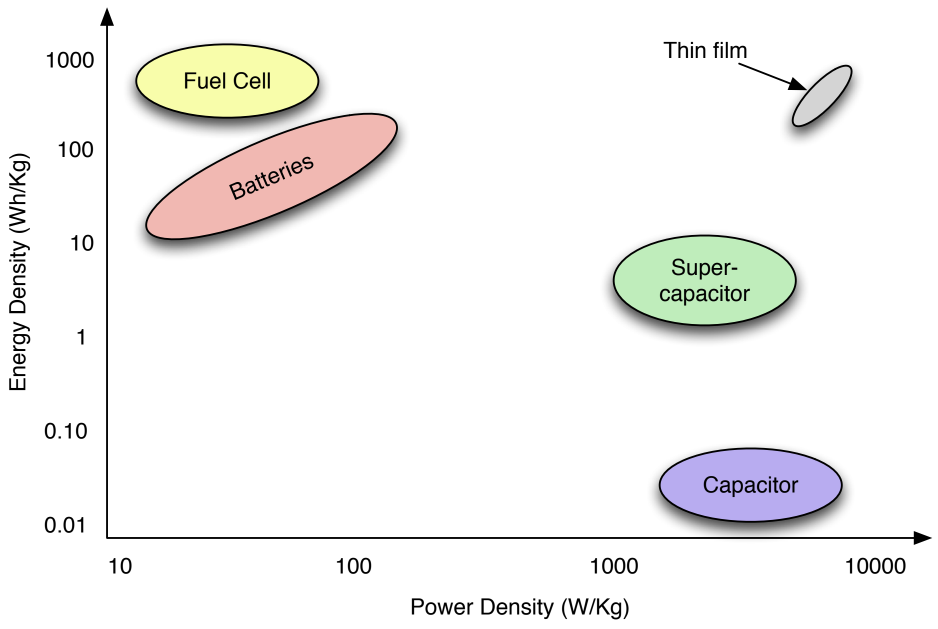


Figure 2.3: Ragone plot power density / power energy plot for typical energy storage technologies

One of the most promising alternatives is the fuel cell [Chraim and Karaki, 2010]. The fuel cell (FC) is an electrochemical device that uses fuel (i.e. Hydrogen) to generate electrical power. FC has a very high energy density if compared to batteries, however the FC cannot respond to sudden changes in the load, thus a system powered solely by the FC is not enough. For this reason a hybrid system where the FC is used to recharge conventional batteries or a super-capacitor is a better solution. The gravimetric energy density of fuel cells is expected to be three to five times larger than Li-ion cells and more than ten times better than Ni-Cd or Ni-MH batteries whereas the volumetric energy density is six to seven times larger than Li-ion. Figure 2.3 shows the Ragone plot for the aforementioned storage systems that helps in understanding the relationship between batteries and capacitors.

In practice no single type of storage element can simultaneously fulfills all of the desired characteristics of an ideal storage system, thus a hybrid solution could overcome the limit of single reservoir and prove to be very much more efficient [Porcarelli et al., 2012].

Nowadays another solution begins to take hold in WSNs to extend the lifetime of node and recharge the batteries: the usage of renewable energy to generate electricity [Morais et al., 2008][Thomas et al., 2006]. The big variety of energy sources in the environment potentially suitable for energy harvesting has driven increased interest in research community. The main sources used for harvesting include solar energy (as well as artificial lightning), vibrations (harvested using piezoelectric, electro-magnetic and capacitive converters), kinetic energy (available in moving water in rivers, pipes and wind flow), magnetic fields (surrounding AC power lines), pressure and heat differentials (harvested using thermoelectric elements). However, due to practical constraints or challenges raised by the low energy density, target power requirements and, in some cases, feasibility of the energy harvesting method, many of these sources could not be considered for energy harvesting [Zhang et al., 2011].

How we can infer from Table 2.2 solar energy is the most efficient natural energy source available for sensor networks in outdoor applications [Brunelli et al., 2009].

All the considerations on battery, harvesting and power supply brings to a power supply unit architecture as depicted in Figure 2.4. This exemplified power unit is composed by the following blocks:

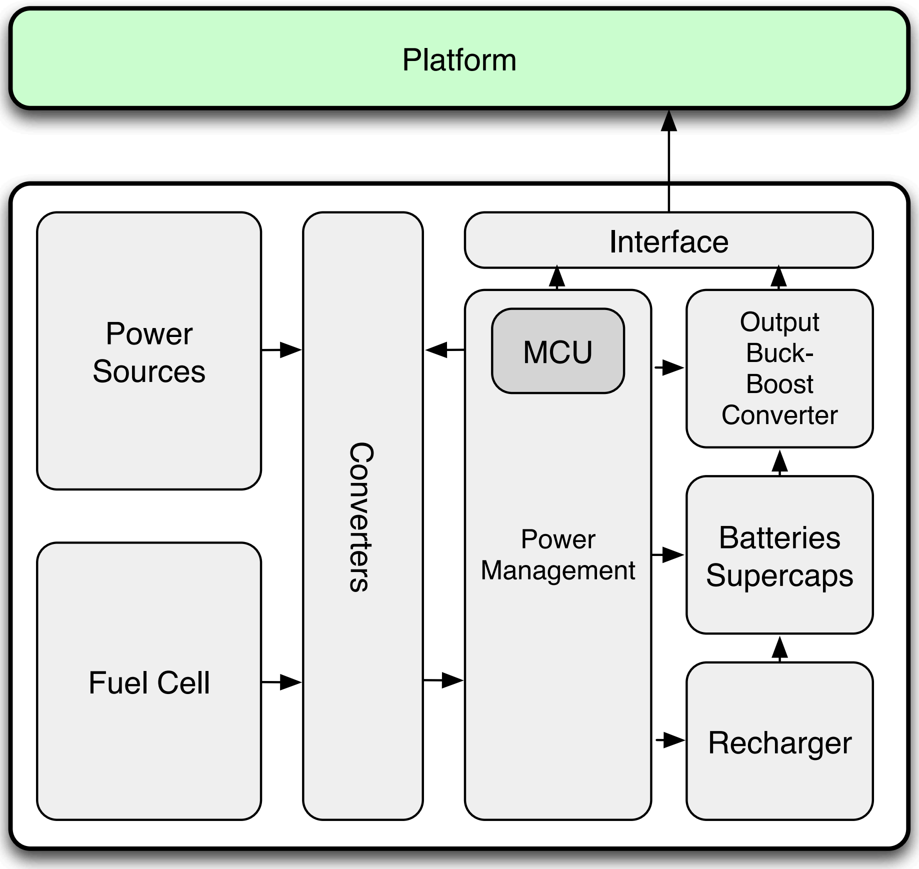


Figure 2.4: Complete power supply architecture for wireless sensor nodes

- **Energy storage**: the output and storage stage implements a double function: it receives the energy from the conversions circuits and stores this energy in specific storage elements (batteries, super-capacitors, etc...).
- **Energy transduction**: to adapt the system to different scenarios, environments, energy sources and different sensor nodes.
- **Fuel cell**: this stage is used to recharge the storage elements when the energy can not be harvester and the storage elements are almost empty.
- **Power unit monitor**: the interactions among the various stages are controlled and managed by a micro-controller through the execution of tasks

Harvesting technology	**Power density**
Solar cells (outdoors at noon)	15 mW/cm^2
Piezoelectric (shoe inserts)	330 μW/cm^3
Vibration (small microwave oven)	116 μW/cm^3
Thermoelectric	(10°C gradient) 40 μW/cm^3
Wind	10W/m^2 at 2.5 m/s

Table 2.2: Power density for different energy harvesting sources

such as measuring the charge status of the storage elements, measuring the voltage levels given by the transducers and the conversion electronics.

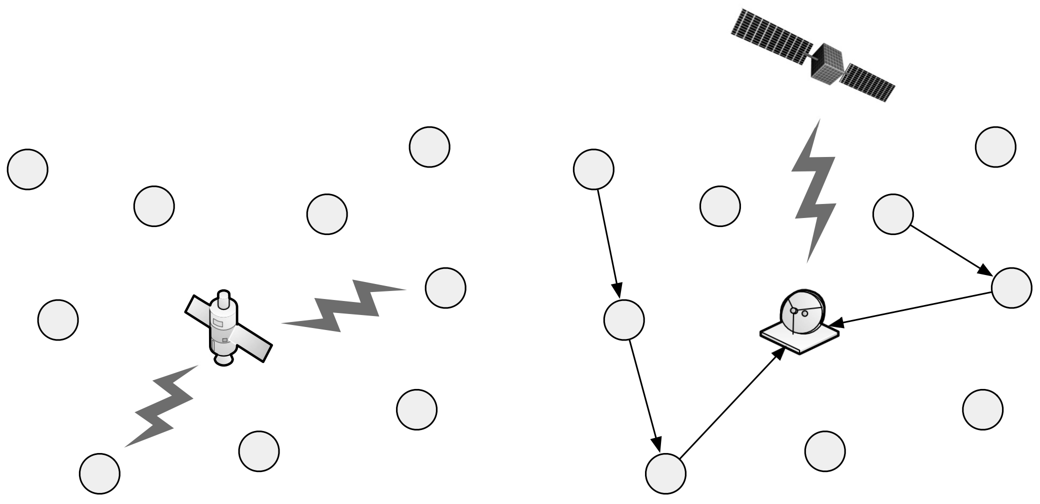
- **Interfaces**: the goal of the interface is to provide adequate information for power management and power policies implementation.

2.4 Radio and communication network

In general, wireless networks can be divided in two main categories: infrastructured and ad-hoc networks. Wireless sensor networks are a special case of ad-hoc networks.

Infrastructured networks The principal characteristic of this type of networks is that they always have one or more central coordination points. The most famous example for this type of networks is the cellular network where each node, to communicate each other, needs to negotiate with the infrastructure hardware of the cell it belongs to [Rahnema, 1993]. Once this communication has taken place the infrastructure takes care of setting up a channel between the peers. In cellular networks the communication between two peers must pass throughout the network infrastructure also if the two devices could be able to communicate to each other because they are in the transmission range of each other.

Ad-hoc networks Differently from infrastructured networks, ad-hoc networks are totally self-organizing and do not relay on a stable infrastructure to enable the communication. Any activity promoted by peers inside the network is carried on



(a) WSN without sink (b) WSN with one sink

Figure 2.5: Different WSN architectures. (a) There is no sink and the nodes communicate directly with the destination node after query. (b) Sink collects data from the sensors via multi-hop communication and forwards collected data to a remote destination

without a central point for coordination. All the peers of an ad-hoc network can play two different roles: (1) the end-point of an active communication (sender or receiver) and (2) the intermediate routing point along the path of an active communication, performing the routing of data for the other peers' communications.

In this type of network we can identify three types of communication pattern: *broadcast*, *convergecast* and *local gossiping*. *Broadcast* is generally used by a base station (sink) to transmit some information to all the sensor nodes of the network (broadcasted information may include queries of sensor, program updates, etc...). In some scenarios, as introduced in Section 2.1.2, the sensors that detect an intruder communicate with each other locally. This type of communication pattern is called *local gossiping* where a sensor sends a message to its neighboring nodes within a range. Lastly *convergecast* is the communication pattern where a group of sensors communicate to a specific sensor or sink.

2.4.1 WSN architecture

Besides the two kind of networks seen in Section 2.4, the WSNs can be considered as a special kind of ad-hoc networks [Akyildiz et al., 2002]. In fact in WSNs each sensor has too little available power to be useful if used alone. Actual WSNs are made up of hundreds (or maybe thousands) of sensors. All sensors cooperate to provide sensing, communication and reliability to all the system.

In WSNs for data collection each node is able to acquire data from the environment, eventually perform some kind of transformation on the acquired data and send out data toward the external world. This can happen in two ways: either every node can communicate to the user or only few nodes can do that. In Figure 2.5 a graphical explanation oh the two cases.

In the first case we can think that each node has enough power to directly communicate the result to the destination point (i.e. using a satellite up-link or GSM connection) or each node can wait to be directly in communication with the destination peer before sending out the data (i.e. a mobile device can directly query the nodes in the network). This case where the communication takes place directly between the sender and the receiver is called single-hop communication and it is fairly rare in WSNs.

Nevertheless the most common is the second case: in the network there are one or more special nodes, namely the sink nodes, which aim is to collect and communicate data to the end user. Usually these nodes are structurally different from the nodes of the network having more computational power and battery. They act like gateways between the user and the WSN and they have usually two interfaces: one toward the WSN for data gathering and the second one used by user to interrogate the network and to extract the needed information. Since the data has to be delivered to the sink that could be physically away for the sensing node, data is routed through all the intermediate nodes on the path toward the sink, in a multi-hop communication.

2.4.2 Energy consumption in radio subsystem

The radio subsystem is the most important system in a wireless sensor node since it is the primary energy consumer in all the application scenarios [Raghunathan et al., 2002]. In general the communication subsystem has a much higher energy consumption than the computation and sensing subsystems. Transmitting one single bit of data consumes as much energy as executing thousands instructions [Shih et al., 2001]. Moreover the radio energy consumption is of the same order in the reception, transmission, and idle states, while the power consumption reduces a lot when the node (and the transceiver) is put in sleep mode. It is clear that the radio has to be managed wisely in order to extend the lifetime of the sensor node and the whole network.

Not all the energy spent by the communication subsystem is wasted and it is possible to identify at least five different cases in which the energy is wasted by radio communications [Demirkol et al., 2006]:

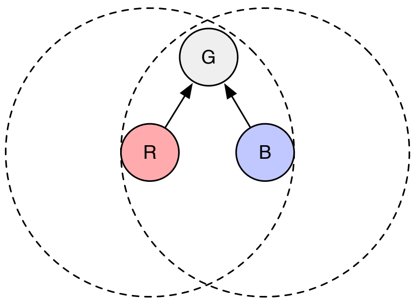
- **Idle listening.** A node is said to be in idle listening when it listens on an idle channel in order to receive possible traffic.

- **Collisions.** A collision occurs whenever a node receives more than one packet at the same time. Each time there is a collision, the packet has to be discarded and the packet retransmitted, increasing the energy consumption. In general it is possible to identify two types of collisions: (1) direct collisions and (2) indirect collisions.

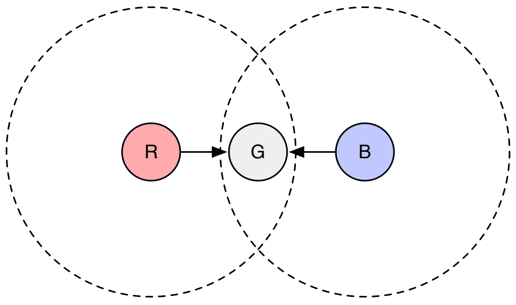
  Direct collisions occur when two or more nodes sending data are in the transmission range of each other and send their data at approximately the same time towards a common receiver as in Figure 2.6a.

  Indirect collisions occur when two or more nodes cannot hear each other but have overlapped transmission ranges and send data at approximately the same time as depicted in Figure 2.6b. In this case there is the so called hidden terminal problem where the nodes in both the transmission ranges of the two sending nodes receive only junk transmission.

- **Overhearing.** A node receives packets that are destined to other nodes. This problem usually arises when a large number of devices try to communicate in the same area. In this case the communications collide frequently and a channel contention problem may rise when many nodes share the same sub-area and all the nodes are able to communicate each-other.



(a) Direct collision



(b) Indirect collision (hidden terminal problem)

Figure 2.6: Different collisions in WSNs. (a) Two nodes are in the transmission range and send data at the same time colliding at node G. (b) Two nodes are not in transmission range but they send data approximately at the same time and again there is collision at node G

- **Control-packet overhead.** This is due to protocol implementations since a minimum number of control packets are required to make the communication feasible.
- **Over-emitting.** It is caused by the transmission of a message when the destination node is not ready. This problem arises when the network synchronization is lost and the nodes are not able to exchange packets.

2.4.3 Energy saving in wireless communications

Duty cycling is the major technique for energy saving in wireless communications [Sadler, 2005]. Normally, a sensor radio has 4 operating modes: transmission, reception, idle listening and sleep. Measurements showed that the most power consumption is due to transmission and in most cases, the power consumption in the idle mode is approximately similar to receiving mode. On the contrary, the energy consumption in sleep mode is much lower. That is the most energy-conserving operation is putting the transceiver in this state whenever the communication is not required. Ideally we could turn on the radio as soon as a new data packet becomes available for sending and switch off it again when the data is sent. This behavior is namely referred to as duty cycling and the duty cycle is defined as the time that the radio spends in an active state as a fraction of the total time under consideration.

The problem with this technique is that in collaborative networks such as WSNs, a coordination among nodes is required to enable data exchanging and forwarding. Thus a sleep / wakeup scheduling algorithm is required to permit the correct functioning of the network. The scheduling can be implemented in two different fashions: (1) on top of an existing MAC protocol (at network or application layer) or (2) strictly integrated within the MAC protocol itself.

In the next section we want to investigate this second case.

2.4.3.1 Low-power MAC protocols for WSNs

The MAC protocol is extremely important in WSN since it creates the network infrastructure by defining the communication channels and shares available communication among nodes. As seen before the radio communications are the most energy expensive actions so the MAC protocol should be properly designed to offer energy saving opportunities by cutting down energy inefficient access to medium.

Besides the energy efficiency other important attributes for MAC protocols are scalability and adaptability to changes. This is because changes in network size or topology are common in WSNs, especially considered the limited node lifetime of the wireless nodes. Moreover the addition of new nodes to the network should be handled rapidly and effectively for a successful adaptation [Wan et al., 2008]. A good MAC protocol should gracefully accommodate such network changes. Other typical important attributes such as latency, throughput and bandwidth utilization may be secondary in sensor networks. Contrary to other wireless networks, fairness among sensor nodes is not usually a design goal, since all sensor nodes share a common task.

The MAC protocols designed to integrate scheduling policies and power management techniques can be categorized in three different categories: (1) contention-based, (2) TDMA-based and (3) hybrid.

Contention-base MAC protocols are based on the Carrier Sense Multiple Data

Access (CSMA) or better on Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA). Nodes using contention-based MAC do not require coordination among nodes in accessing the communication channel. When a node needs to send data, it verifies the absence of other traffic before transmitting. If the channel is sensed busy before transmission then the transmission is deferred for a random interval. This reduces the probability of collisions on the channel. Colliding nodes will back off for a random duration of time before attempting to access the channel again.

In TDMA-based (Time Division Multiple Access) protocols there is an explicit synchronization among the nodes in the network. The scheduling offers a collision-free scheme by assigning each node a well defined time slot in which communication is permitted. This synchronization avoids interferences between adjacent nodes and, consequently, the energy waste coming from packet collision [Arisha, 2002]. Moreover TDMA-based MAC protocol are immune to the hidden terminal problem seen in Section 2.4.2 since the time slot of each node is unique among its neighbors.

Hybrid MAC protocols have the advantages of both contention-based MAC and TDMA-based MAC protocols. While control packet are transmitted in the random access channel, data packets are transmitted in the scheduled access channel, in this way hybrid protocols can obtain higher energy saving and offer better scalability and flexibility.

In the following paragraphs are few of the most common low-power MAC protocols opportunely designed for wireless sensor networks.

Sensor-MAC (S-MAC) To solve the energy wasting problem the S-MAC [Song et al., 2008] alternates two different states: active state and sleep state, that is periodically listening and sleeping. S-MAC actually reduces the listening time by letting the node go to sleep into periodic sleep mode. Nodes exchange sync packets to coordinate their sleep / wakeup periods and neighboring nodes form virtual clusters to set up a common sleep schedule. If two neighboring nodes reside in two different virtual clusters, they wake up at listen periods of both clusters. The channel access time is split in two parts. In the listen period nodes exchange sync packets and special control packets for collision avoidance while in the remaining period the actual data transfer takes place. Considering that the sender and the destination have to be awake and talk to each other, this brings in a problem since periodic sleep may result in high latency especially for multi-hop routing algorithms, since all immediate nodes have their own sleep schedules. The latency caused by periodic sleeping is called sleep delay. To avoid high latencies in multi-hop environments S-MAC uses an adaptive listening scheme. A node overhearing its neighbor transmissions wakes up at the end of the transmission for a short period of time. If the node is the next hop of the transmitter, the neighbor can send the packet to it without waiting for the next schedule.

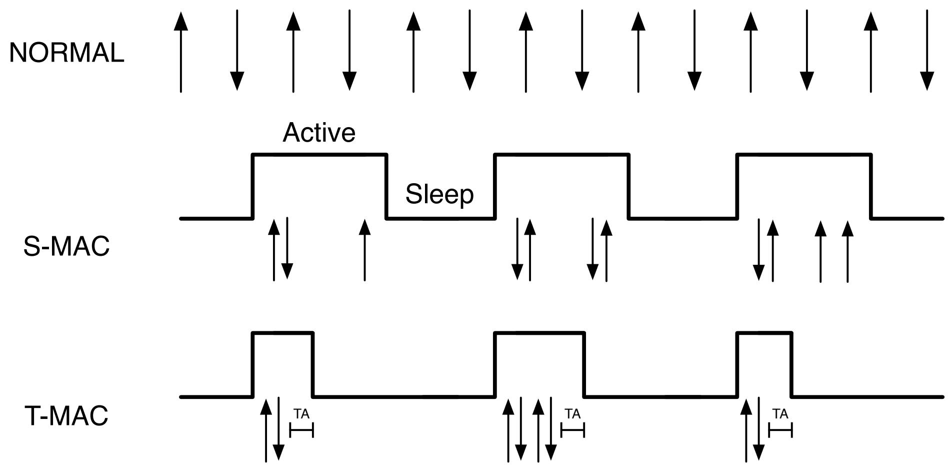


Figure 2.7: Comparison between S-MAC and T-MAC

Timeout-MAC (T-MAC) The problem of sleep delay resulting in high latencies as seen for S-MAC is partially solved in the so called Timeout-MAC (TMAC) [Farjaudon and Hascoet, 1988] that results particularly useful under variable traffic load. In T-MAC listen periods end when no activation event has occurred for a time threshold TA. A comparison between S-MAC and T-MAC is in Figure 2.7.

D-MAC Although duty-cycle based MAC protocols are energy efficient, they suffer sleep latency, i.e., a node must wait until the receiver wakes up before it

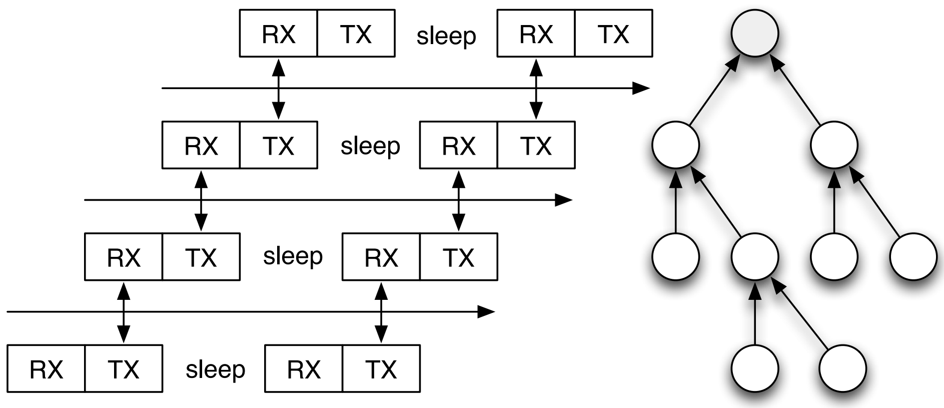


Figure 2.8: A data gathering tree using D-MAC protocol

can forward a packet. This latency increases with the number of hops. In addition, the data forwarding process from the nodes to the sink can experience an interruption problem. This is why the data forwarding process in S-MAC and T-MAC is limited to a few hops. When dealing with deep static tree-like networks, these MAC protocols show their limitations. In this type of networks convergecast is the mostly observed communication pattern within sensor networks. These unidirectional paths from possible sources to the sink could be represented as data gathering trees. D-MAC [Kebkal et al., 2010] is an adaptive duty cycle protocol that is optimized for data-gathering in tree-like networks. In D-MAC based networks the nodes schedules are staggered according their position in the data gathering tree. Each node has a slot which is long enough to permit data transmission hence, during the receive period of a node, all of its child nodes has transmit periods and contend for the medium. Low latency is achieved by assigning subsequent slots to the nodes that are successive in the data transmission path as in Figure 2.8.

Traffic-Adaptive MAC Protocol (TRAMA) One of the most important energy-efficient TDMA protocol for wireless sensor networks is TRAMA [I and Pollini, 1994]. TRAMA divides time in two portions, a random-access period and a scheduled access period. The random access period is devoted to slot reservation and is accessed with a contention-based protocol. On the contrary, the scheduled access period is formed by a number of slots assigned to an individual node. The slot reservation algorithm is composed by a series of subsequent steps. First the nodes derive the two-hop neighborhood information. This information is used to create a collision free scheduler. Afterwards the nodes start an election procedure to associate each slot with a single node. A node becomes the owner of the slot only if its priority, calculated as a hash function of the node identifier and the slot number, is the highest priority among all the nodes. Finally nodes send out a synch packet containing a list of intended neighbor destinations for subsequent transmissions. Consequently nodes can agree on the slots which they must be awake in.

Other low-power MAC protocols Among the TDMA-based MAC protocols it is possible to find: FLAMA (FLow-Aware Medium Access) [Rajendran et al., 2005], LMAC (Lightweight MAC) [Lee et al., 2008], AI-LMAC (Adaptive Information-centric LMAC) [Chatterjea et al., 2004], SPARE-MAC (Slot Periodic Assignment for Reception MAC) [Turati et al., 2009], DEE-MAC (Dynamic Energy Efficient MAC) [Cho et al., 2005]. For contention-based MAC protocols we have: B-MAC (Berkeley MAC) [Fakih et al., 2006], U-MAC (Utilization-based MAC) [Yang et al., 2005]. Hybrid MAC protocols: Z-MAC [Rhee et al., 2008], Wise-MAC [El-Hoiydi and Decotignie, 2004], PTDMA (Probabilistic TDMA) [Oikonomou and Stavrakakis, 2004].

2.4.3.2 Duty cycling on top of MAC protocols

As seen in Section 2.4.3 sleep / wakeup schemes can be defined also as independent protocols (network or application layer) on top of existing MAC protocols. This protocols are usually divided into three main categories: *on-demand*, *scheduled rendezvous* and *asynchronous* schemes.

The basic idea of *on-demand* protocols it that a node should wakeup only when another node wants to communicate with it. The real problem with this approach is how to inform the sleeping node that a child node is willing to communicate. Sometimes the best solution is to implement a wake-on-radio mechanism where a second low-rate and low-power radio or analog circuit is used for signaling and wakeup while a more powerful and more energy hungry radio is used for data transmission.

An alternative solution consist in using a *scheduled rendezvous* approach. The idea is that each node should wake-up at the same time as its neighbors. The wakeup time is scheduled and the node remain active just for the short time interval needed to communicate with their neighbors before going back to sleep and waiting until the next rendezvous time.

Finally an *asynchronous* sleep / wakeup protocol can be used. In this case a generic node can wake up when it wants and it is still able to communicate with its neighbors. This is possible by guaranteeing that neighbors always have overlapped active periods within a specified number of cycles.

These technique for power management are usually implemented on top of an existing MAC protocol. For low-power WSNs the most common protocol is the IEEE 802.15.4.

IEEE 802.15.4 The IEEE 802.15.4 protocol is a standard for low-rate low-power Personal Area Network (PAN). A PAN is composed by a coordinator which manages the whole network (sometimes it is possible to have several coordinators managing subsets of the nodes in the network). Each node in the network must associate with the PAN coordinator in order to communicate. In the original standard the only supported network topologies are: star (single-hop), cluster-tree and mesh.

The IEEE 802.15.4 protocol [Zheng and Lee, 2003] supports two operational modes that may be selected by the PAN coordinator: (1) the *non beacon-enabled mode* in which the MAC is simply ruled by non-slotted CSMA/CS and (2) the *beacon-enabled mode* in which beacons are periodically sent by the coordinator to synchronize nodes that are associated with it. These beacons provide an energy management mechanism based on a duty cycle.

In beacon-enabled mode the coordinator defines a superframe structure as in Figure 2.9 which is constructed defining: (1) the *Beacon Interval (BI)* which defines the time between two consecutive beacon frames, (2) the *Superframe Duration (SD)* which defines the active portion in the *BI*, and is divided into 16 equally-sized time slots during which frame transmissions are allowed.

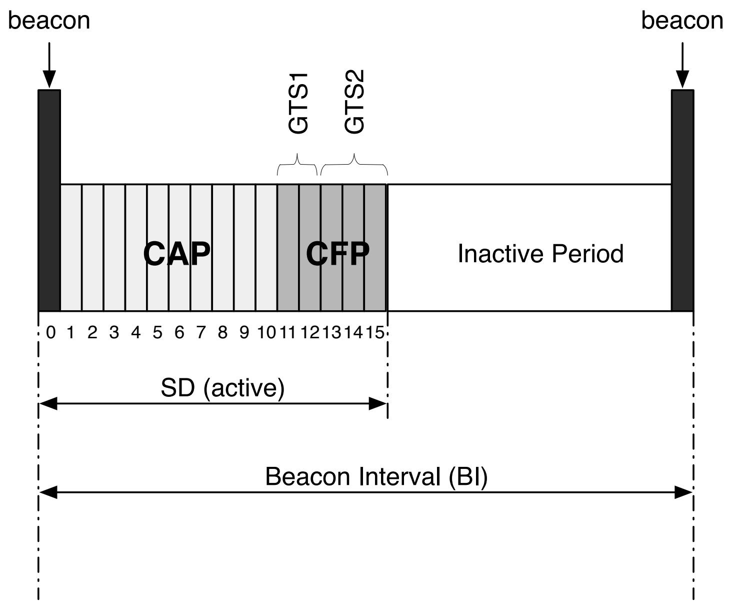


Figure 2.9: Superframe structure for IEEE 802.15.4 protocol

Optionally, an inactive period is defined if *BI* > *SD*. During the inactive period all nodes may enter in sleep mode.

BI and *SD* are determined by two parameters, the *Beacon Order (BO)* and the *Superframe Order (SO)*, respectively, as

$$\left.\begin{aligned} BI &= aBaseSuperframeDuration \cdot 2^{BO} \\ SD &= aBaseSuperframeDuration \cdot 2^{SO} \end{aligned}\right\} \text{for } 0 \leq SO \leq BO \leq 14 \tag{2.1}$$

$aBaseSuperframeDuration = 15.36$ ms denotes the minimum duration of the superframe, corresponding $SO = 0$.

During the *SD*, nodes compete for medium access using slotted CSMA/CA in the *Contention Access Period (CAP)*. For time-sensitive applications, IEEE 802.15.4 enables the definition of *Contention-free period* within the *SD*, by allocation of *Guaranteed Time Slots (GTS)*. It can be easily observed that low duty-cycles can be configured by setting small values of the *superframe order (SO)* as compared to *beacon order (BO)*, resulting in greater sleep (inactive) periods.

The advantage of this synchronization with periodic beacon frame transmissions from the PAN coordinator is that all nodes wake up and enter in sleep mode at the same time. However using this synchronization scheme in a cluster-tree network with multiple coordinators sending beacon frames, each with its own beacon interval, is a challenging problem due to beacon frame collisions.

2.4.3.3 Case study: Conservative Power Scheduling

The most common scheduled rendezvous scheme for tree-based networks is the *staggered wakeup pattern* [Keshavarzian et al., 2006] where nodes located at different levels of the data-gathering tree, wake up at different times. This approach is much more flexible than the classical *fully synchronized pattern* in which all the nodes of the network wake up at the same time according to a periodic pattern as seen in Section 2.4.3.1 for S-MAC and T-MAC. On the contrary the *staggered wakeup pattern* takes advantage of the internal organization of the network by sizing the active times of different nodes according to their position in the data gathering tree.

The staggered scheme has the advantage that at different times, only a subset of the nodes are active thus the probability of collisions is potentially lower than for the fully synchronized pattern, since only few nodes contend for the channel access at the same time. As seen in Section 2.4.2 this permits a reduction in the power consumption since the active period of each node can be significantly shorter. This scheme enables also mechanisms of data aggregation since the parent nodes can wait to receive data from all their children before forwarding to the next node in the path.

The major problem with the staggered scheme is that nodes located at the same level in the gathering tree wake up at the same time so collisions still are a problem. Moreover the scheme has limited flexibility due to the fixed duration of the active and sleep periods that are usually the same for all nodes in the network.

Ideally a low-power protocol should be able to allow different active and sleep times for different nodes in the gathering tree according to the different amount of data managed by the single node.

Following this principle in this paragraph a new power management protocol derived for the *staggered wakeup pattern* is presented. This protocol, namely *Conservative Power Scheduling (CPS)* protocol, is built on top of IEEE 802.15.4 MAC protocol and tries to adapt a slotted approach derived from TDMA schemes to the staggered pattern already presented.

In the following description we will refer to a data collection paradigm where data typically flows from source nodes to a sink node. Nodes are organized in the network to form a static tree rooted at the sink. The routes in the tree are static and each parent node has to be physical neighbor of all its children. Even though it is possible to assume that the routes are static for a long period, the paths can re-computed periodically to take into account variation in the network topology.

Time is assumed to be divided in time slots of duration T_s. Slots are arranged to form period cycles, namely the *communication periods*, where each cycle is made up of m slots and has a duration of $T_c = mT_s$. Each communication period is divided into two parts: an *Active Interval (AI)* during which the node must keep its radio on to receive / transmit from / to other nodes, and the *Sleep Interval (SI)* during which nodes are sleeping, both periods are multiple of the time slot T_s. To permit data exchange, the talk interval between a node and its parent / child must overlap in at least a time slot T_s as clarified in Figure 2.10.

Consider for example a generic node j having node i as parent and node k as child. If T_c is the communication period, AI_{ij} the overlapping active interval between nodes i and j, AI_{jk} the overlapping active interval between j and k, then to ensure the protocol correctness we have to assure that:

$$AI_{ij} + AI_{jk} \leq T_c \tag{2.2}$$

The cycle of duration T_c is the same for all the nodes in the network thus a synchronization is required for the entire network. The most simple method to guarantee synchronization among all the nodes in a tree-based network is to use a synchronization packet sent in broadcast by the tree coordinator. Each synchronization broadcast packet delimits the cycle period T_c

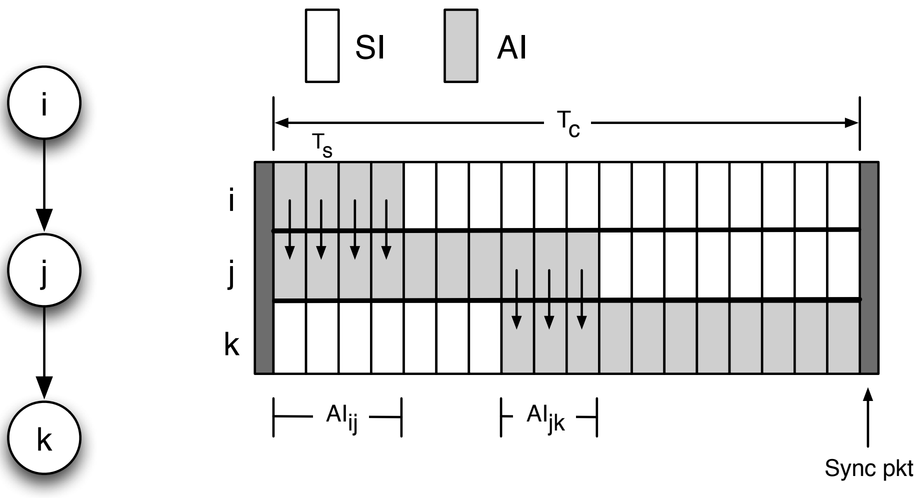


Figure 2.10: Conservative Power Scheduling parameters and communication model

When the CPS protocol is used, to reduce the probability of collisions in the network each time slot T_s is uniquely assigned to a specific node in the whole network. This restriction practically reduces to zero the probability of collision but, on the other hand, it increases the latency experienced by packets to reach the root node. In data-gathering networks for data collection this is not a real issue since the inter-sampling period (in the order on tenth of minutes) is usually extremely larger than the gathering time, thus in this scenario it is more important to preserve the integrity of data avoiding collisions than low data latency.

The real challenge in CPS protocol is to correctly assign the time slots to each node such as to minimize the power consumption. To achieve this goal each node has to be in the active interval for the shortest time possible, waking up just in time for sending sampled data to the parent node and going back to sleep as soon as it has forwarded all the data coming from its child nodes. Differently from other approaches found in literature [Hohlt et al., 2004][Lu et al., 2004], CPS does not calculate adaptively or run-time the slots scheduling but it uses a static scheduler computed off-line. The algorithm for slots assignment takes in input the tree-structure (parent-children relationships for all the nodes) of the network and gives in output the slot times assigned to each node. In practice if k is a generic node, the algorithm is able to calculate the initial slot time D_k^S and the final slot time D_k^E such that $D_k^E - D_k^S = AI_k/T_s$.

The algorithm is articulated in two different steps: (1) *ordering* and (2) *scheduling*.

In the *ordering* phase each node in the network is assigned with an unique ID that defines the priority of the node. Smaller the ID, greater the node priority (the coordinator has always ID = 0). The IDs assignment follows the Depth-first (DFS) algorithm. DFS is an uninformed search that progresses by expanding the first child node of the search tree that appears and thus going deeper and deeper until a goal node is found, or until it hits a node that has no children. Then the search backtracks, returning to the most recent node it hasn't finished exploring.

After each node is opportunely tagged with an ID (from 0 to ID^{MAX}), the *scheduling* step of the algorithm is in charge to calculate the starting time slot for each node. Thus mathematically we calculate $D_{ID}^S \forall ID \in \{1, 2, \ldots, ID^{MAX}\}$ with:

$$D_{ID}^S = D_{ID-1}^S + L_{ID-1} \tag{2.3}$$

where L_{ID-1} is the hop distance of the node with ID=$ID-1$ from the sink (having $L_0 = 0$).

After the starting time slot, we can derive the ending time slot $D_{ID}^E \forall ID \in \{1, 2, \ldots, ID^{MAX}\}$ with:

$$D_{ID}^E = \begin{cases} \max\left(D_{chld\_ID}^S + 2\right), & \text{if } chld\_ID \neq \varnothing \\ D_{ID}^S + 1, & \text{if } chld\_ID = \varnothing \end{cases} \tag{2.4}$$

where $D_{chld\_ID}^S$ define the set of starting time slots of the child nodes and $chld\_ID$ the set of their IDs.

The rationale behind Equation 2.3 is straightforward: each node waits that data sent by its own parent has reached the sink before sending out its own data,

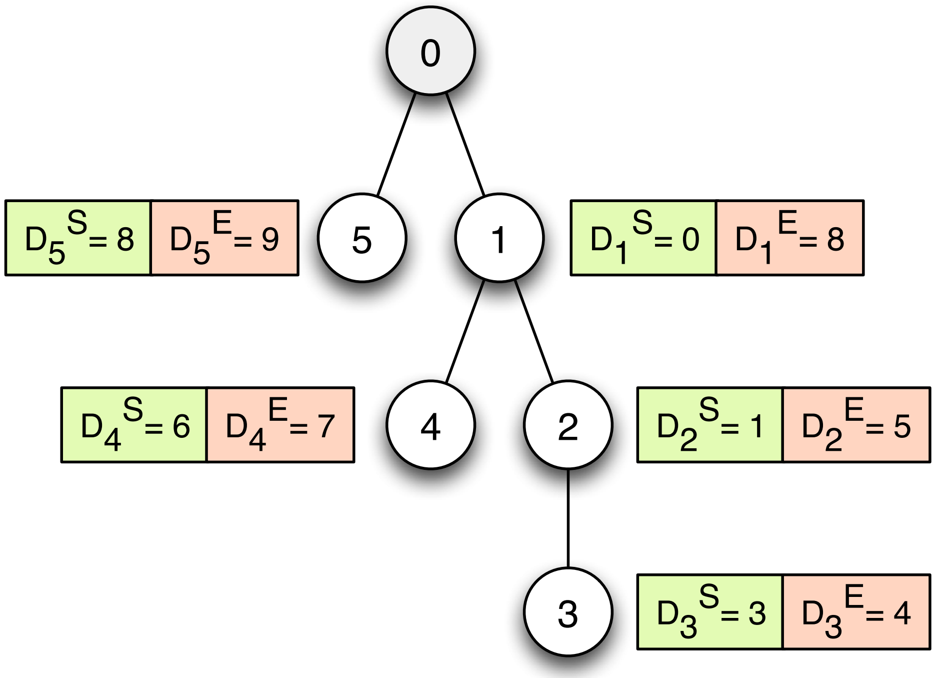


Figure 2.11: Tree-based network with associated IDs, starting and ending time slots for CPS protocol

avoiding collisions in the network. The same for Equation 2.4 that indicates how the node can switch in sleep mode only after having forwarded data coming the child nodes. In Figure 2.11 the resulting tree with IDs after the ordering step is reported, in the boxes next each node the values of D_{ID}^S and D_{ID}^E are also indicated.

In Figure 2.12 the temporal scheduling of the network seen in Figure 2.11 is clearly defined. Is is possible to see in the plot how each node sends data as soon as it wakes up and returns in low-power mode after all the child nodes have sent its own data. In every time slice T_s only one node is active at time, thus there is no energy wasting for data collision management.

In practical implementation of CPS, the synchronization using broadcast packets from the tree coordinator can be problematic due to delay in packet relay. For this reason the margins for synchronization are relaxed accounting for clock drift

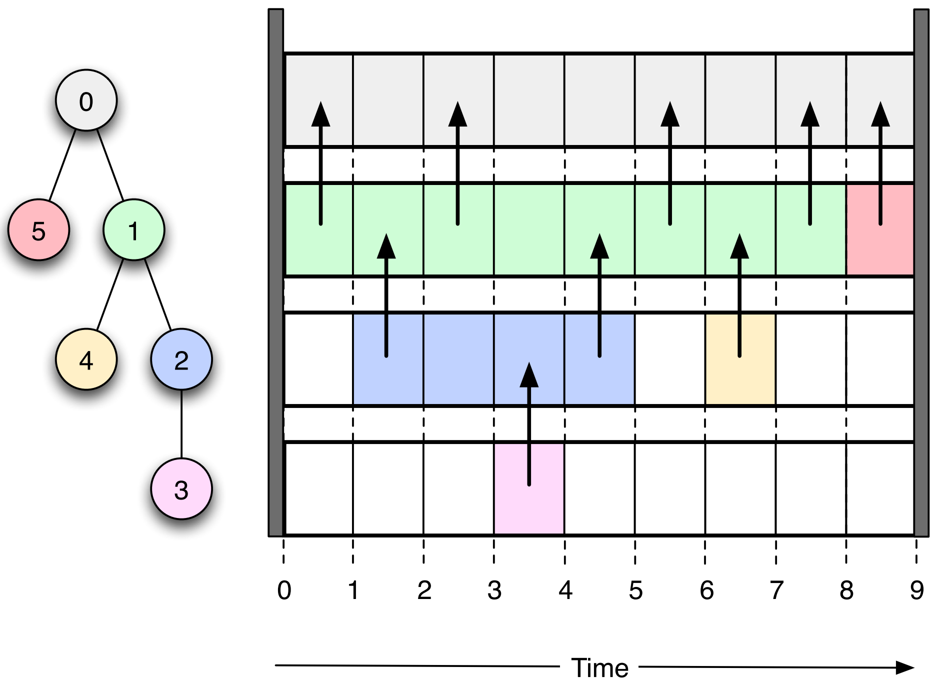


Figure 2.12: Scheduled network using CPS protocol

and delay in transmission of synchronization packets. Practically within a communication period T_c synchronization among nodes becomes sensible in mainly three temporal instant: (1) when the node has to wake up to receive the synchronization packet, (2) at D_{ID}^{S} since it has to send sampled data to parent node and (3) at D_{ID}^{E} when it needs to forward data to the next hop toward the sink.

For these temporal markers when implementing the protocol on real hardware it is always needed to consider a margin of error for synchronization, as clearly showed in Figure 2.13. In the plot is reported the detailed time-line for a communication period of a generic node implementing the CPS protocol.

First it is possible to see how, accounting for a non-perfect synchronization, the node wakes up 100 ms before the scheduled appointment for the reception of the synchronization packet. In this way the synchronization packet can be received also in presence of temporal drifts. After the reception of the synchronization packet the node turns off the radio to save power and starts the sampling of sensors, at the end of which the node, if with $D_{ID}^{S} \neq 0$, goes to sleep, entering in the *Sleep Interval*.

Usually using a wake up timer, the node wakes up slightly in advance of the set D_{ID}^{S} and, after sending and forwarding data, returns in sleep mode again at D_{ID}^{E}, until the next communication period.

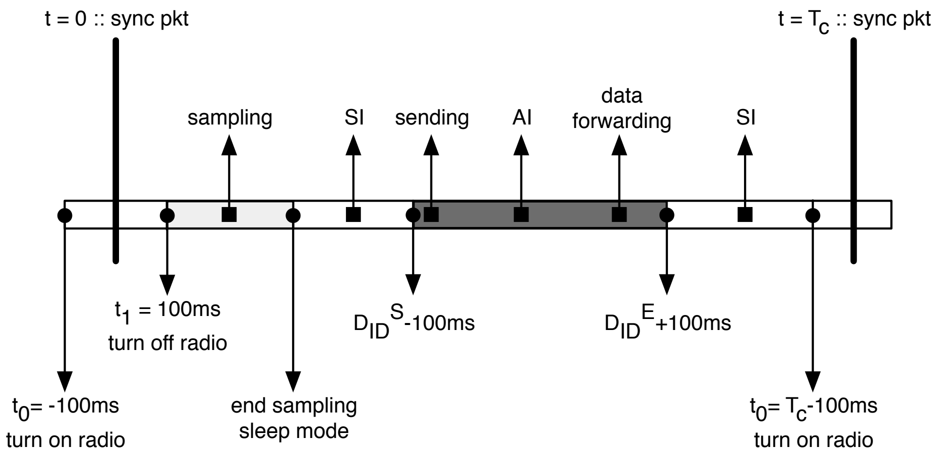


Figure 2.13: Detailed view of a communication period for practical implementation of CPS protocol

2.5 The processing subsystem

In WSNs a balance must be maintained between capability and power consumption to best address the application needs. This is particularly important for the processing subsystem in sensor nodes that is in charge to elaborate data and managing the connection status of the node.

2.5.1 Hardware characteristics

Modern micro-controllers integrate flash storage, RAM, analog-to-digital converters and digital I/O one a single integrated circuit therefore their tight integration makes them ideal for use in deeply embedded systems like wireless sensor networks. The key points to take into account when selecting a micro-controller family are mainly energy consumption and support for external peripherals but also wake-up time and speed.

Power consumption The power consumption is strictly dependent on the micro-controller family and architecture. Usually the trade-off between power consumption and computational power is expressed in mA per MHz. Standard micro-controllers consume between 0.250 to 2.5 mA per MHz. This parameter is usually referred to peak power consumption or power consumption when the MCU is active and loaded. For WSNs the most important parameter to evaluate the energy saving is the stand-by or sleep power consumption, that is the consumption of the MCU needed by the MCU to only maintain its memory and the synchronization mechanisms (wake-up timers) to properly wake up when necessary.

Sleep current consumption varies between 1 uA and 50 uA across controller families. This parameter is extremely important since in usual applications for WSNs the CPU is expected to be idle (thus in stand-by or sleep mode) 99.9% of the time.

Wake-up time Other important parameter for MCU is the wake-up time. The wake-up timer is the delay of a certain micro-controller to start after a sleep period and stabilize the system clocks. The faster a controller can enter and leave the low-power state, the more often the sleep state can be used. If the wake-up time is short it is possible to put the MCU to sleep even between each bit of the transmission to save energy.

Speed In the past the main duties of the micro-controller were just to manage the communication protocols, control the radio and interact with sensors. Most of these operations did not require very fast CPU, for this reason the CPU used in sensor nodes were relatively slow CPU (usually between 1 and 8 MHz). Nowadays the availability of very powerful and fast processors with a relatively small power consumption is moving the research toward complex data processing and compression techniques able to greatly reduce the amount of data to send. Since the power consumption of the radio is usually higher than the power spent on calculations by CPU, it is more energy efficient trying to compress data as much as possible before sending them out, saving on communication costs.

Another important feature of several controller families is the ability to dynamically change the operating frequency to reduce the power consumption. However the execution time is inversely proportional to frequency, therefore the choice between lowering the working frequency or increasing the execution time is dependent on the architecture and application.

Memory In general sensor nodes only require small amounts of storage and program memory since data is only stored long enough for it to be analyzed and then transmitted. In general, modern flash-based micro-controllers contain between 16 and 512 KB of on-chip program storage that can be used as both program memory and as temporary data storage. Additionally they contain between 8 and 64KB of data ram that can be used for program execution. while SRAM requires more energy to retain data over time and does not require as much energy for the initial storage operation, Flash, on the other hand, is a persistent storage technology that requires no energy to maintain data.

Operating systems support Programming wireless sensor nodes is generally not an easy task because of the constrained memory and processing power, the requirement to manage the radio communication and the need to conserve as much energy as possible. Fortunately a number of operating systems are now available for wireless sensor networks to help in the development process. The OS manages the node's hardware and provides a high level interface to this hardware for the programmer.

The most common operating systems for WSN are: TinyOS [Levis et al., 2005], Contiki [Dunkels et al., 2004], MANTIS [Bhatti et al., 2005], Nano-RK [Eswaran et al., 2005] and Lite-OS [Cao et al., 2008].

Platform	CPU	Clock (MHZ)	RAM/Flash	Radio transceiver	OS
Rene 1	Atmel AT90LS8535	4	512/8K	RFM TR1000	TinyOS
Rene 2	Atmel Atmega 163	8	1K/16K	RFM TR1000	TinyOS
Mica	Atmel Atmega 128L	4	4K/128K	RFM TR1000	TinyOS
BT Node	Atmel Atmega 128L	8	4K/128K	Chipcon CC1000	TinyOS
Mica2	Atmel Atmega 128L	8	4K/128K	Chipcon CC1000	TinyOS
iMote	Zeevo ZV4002 (ARM)	12-48	64K/128K	Zeevo BT	TinyOS
Nymph	Atmel Atmega 128L	4	4K/128K	Chipcon CC1000	TinyOS
MicaZ	Atmel Atmega 128L	8	4K/128K	Chipcon CC2420	TinyOS
iMote2	Intel PXA 271	13-104	256K/32M	Chipcon CC2420	TinyOS
TelosB	TI MSP430F1611	8	10K/48K	Chipcon CC2420	TinyOS
EM250	Atmel Atmega 128L	8	4K/128K	Ember 250	EmberNet
Sun Spot	Atmel AT91FR40162S	75	256K/2M	Chipcon CC2420	Squawk VM
SHIMMER	TI MSP430F1611	4/8	10K/2G	Chipcon CC2420	TinyOS
IRIS	Atmel ATmega 1281	8	8K/640K	Atmel ATRF230	TinyOS

Table 2.3: Selection of sensor network nodes commercially available and their hardware characteristics

2.5.2 Sensor nodes for wireless sensor networks

Considering the stringent hardware requirements for sensor nodes several platforms have been developed trying to address every point highlighted in the previous section. In the following paragraph are presented just few of the most important platform nodes publicly available on the market. Table 2.3 shows a longer list of the most common node platforms commercially available with hardware characteristics.

MicaZ [Hill and Culler, 2002] The MicaZ platform, designed at UC Berkeley, combines communication, computation, power management, and sensing into a small experimental platform (the size is comparable to a pair of AA batteries). The main micro-controller is an Atmel Atmega103L or Atmega128 running at 8 MHz, 128KB flash program memory, 4KB static RAM and the radio module consists of a Chipcon CC2420. The standby current for Mica's components is a few microamps and the platform includes an expansion bus that connects to a wide array of sensor boards.

iMote2 [Nachman et al., 2008] The iMote2 is the natural evolution of the original IMOTE platform. Built around a low power XScale PXA271 processor, it integrates an 802.15.4 radio (ChipCon CC2420) along with an embedded 2.4GHz chip antenna. In addition to the traditional TinyOS, the iMote2 also supports

Linux and the Microsoft .net micro framework.

TelosB TelosB platform is built around an 8 MHz TI MSP430 micro-controller with 10kB RAM, 1MB external flash for data logging and an IEEE 802.15.4/ZigBee compliant RF transceiver with integrated on-board antenna. Even it his case TinyOS is the default OS.

2.5.3 High-performance 32-bit micro-controllers for wireless sensor networks

Even though, as seen in Table 2.3, the majority of the nodes nowadays used in the WSN field are small 8/16-bit micro-controllers, the gap in power consumption (but also size and price) between the 8/16-bit and the more powerful 32-bit micro-controllers is shrinking. In Table 2.4 a short list of 32-bit sensor nodes available is presented. We can identify 4 different areas in which the usage of 32-bit architectures is becoming more convenient with respect to the old 8 and 16-bit architectures: (1) price, (2) performance, (3) code size and (4) energy consumption.

Price Even though the price of the 32-bit micro-controller is still higher than the price for 8/16-bit MCUs, the gap is going to be reduced in the next years. This reduction is mainly consequence of the large increasing in production for 32-bit MCUs that are gradually replacing the old architectures in almost every application.

Performance It is straightforward to have more performance per MHz delivered by the 32-bit micro-controllers. This increased performance is not only due to the increase in clock speed but also to hardware accelerators and new technologies (i.e. Thumb technology) to improve code density and execution speed.

Code size 32-bit processors usually offer superior code density to 8-bit and 16-bit architectures. This has significant advantages in terms of reduced memory requirements and maximizing the usage of precious on-chip Flash memory. For example with Thumb-2 technology, the ARM processors support a fundamental base of 16-bit Thumb instructions, extended to include more powerful 32-bit instructions. In many cases a C compiler will use the 16-bit version of the instruction unless the operation can be carried out more efficiently using a 32-bit version.

Power consumption The energy saving derived from the usage of 32-bit MCU is due to several factors. First, when considering ARM micro-controllers with support for Thumb and Thumb-2 instructions, the instruction efficiency of these MCUs is definitely higher. There are many circumstances where a single Thumb instruction equates to several 8/16-bit micro-controller instructions; this means that the 32-bit micro-controllers can achieve the same task at lower bus speed.

The possibility to run at lower MHz with a higher efficiency means that a task can be achieved with shorter activity periods, saving energy as depicted in Figure 2.14.

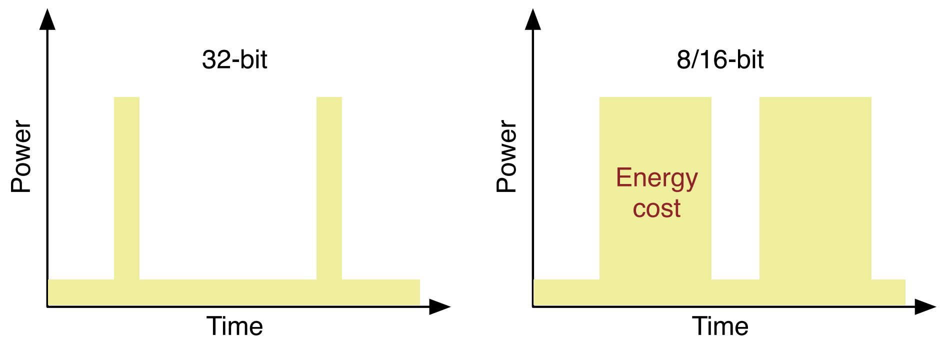


Figure 2.14: 32-bit faster micro-controllers achieve tasks with less energy in comparison to old 8/16-bit micro-controllers

Another way to save energy is the possibility to put the MCU in sleep modes. 32-bit architectures usually support more sleep modes that the older architectures. The increasing in the granularity of these sleep modes allows the developer to always choose the better mode to achieve the minimum power consumption compatibly with the application requirements.

Platform	CPU	Clock (MHZ)	RAM/Flash	Radio transceiver
STM32W108	ARM Cortex-M3	6/12/24	8-16KB/64-256K	Embedded
Ember EM35x	ARM Cortex-M3	6/12/24	12K/128-192K	Embedded
iMote2	Intel PXA 271	13-104	256K/32M	Chipcon CC2420
Jennic JN5148	RISC	4-32	128K/128KB	Embedded

Table 2.4: Selection of 32-bit sensor network nodes commercially available

Finally the increasing demand for connectivity and sophisticated analog sensors has resulted in the need to more tightly integrated analog devices with digital functionality to pre-process and communicate data and most 8/16-bit devices usually do not offer performance to sustain these tasks without significant increase in MHz and power consumption.

2.5.4 Case study: ultra-low power device for aircraft structural health monitoring

Damages to spacecraft and high-speed vehicle caused by the impact of debris and particles is a critical concern for automotive and avionic systems especially because this type of damage cannot be easily detected. The acronym BVID (Barely Visible Impact Damage) is usually employed to indicate this type of damage. There exist a number of different inspection techniques to localize impact damages and to determine their severity (X-ray, C-scan, thermography, etc...) [Roemer et al., 2005].

Among the number of structural health monitoring (SHM) technologies, the one based on guided waves (GW) is considered as the most promising and versatile. When the impact occurs at high speed this produces detectable acoustic and ultrasonic waves on the structural materials that can be used to compute the location of the impact and eventually to assess the damage [Bovio and Lecce, 2005]. The passive approach based on ultrasonic Lamb waves and conventional piezoelectric transducers (PZT discs) is capable of achieving high localization performance using a dispersion compensation algorithm with low computational cost.

A lot of literature has been produced on the use of these sensor-array-based methods for high-speed acquisition and processing of data from a large number of individual sensors that usually are bulky, heavy and require wiring back to a central location. Moreover when large-scale deployments are implied, the power consumption of the system is hardly sustainable by the ordinary generation system present on board.

For all these reasons, in contrast to these traditional transducers, wireless sensors technologies integrating small sensors and wireless communication are becoming vital in SHM, guaranteeing: (1) less wiring among sensors and between sensors and central unit; (2) lower weight; (3) reduced power consumption; (4) real-time monitoring even in harsh environmental conditions and (5) a more scalable solution.

Exploiting the advantages and the opportunities given by modern 32-bit processors it is possible to think and design a new PZT-based wireless embedded ultrasonic structural monitoring system for impact localization with advantages of compactness, light weight, low-power consumption, high efficiency and precision.

The passive approach based on ultrasonic Lamb waves and conventional piezoelectric transducers (PZT discs) is capable of achieving high localization performance using a dispersion compensation algorithm with low computational cost. The device samples the signals in passive mode using 4 different piezoelectric transducers and the signals are elaborated on a Cortex-M4 based micro-controller. By cross-correlating the dispersion-compensated signals, the impact point can be determined via hyperbolic positioning. When the impact occurs, the monitoring position result is recorded only in a small amount of data and sent to the central system through wireless transmission.

2.5.4.1 SHM design

The structure of the new SHM system is illustrated in Figure 2.15. In the new SHM system, the signal conditioning, amplification and A/D converting circuits present in conventional SHM systems are replaced by a simple comparator circuit, in which the response signal from a PZT sensor is directly changed into a digital queue by comparing it with a preset trigger value.

The system is composed by 4 different elements: (1) piezoelectric sensors, (2) processing electronic unit, (3) acquisition chain and (4) wireless transmission

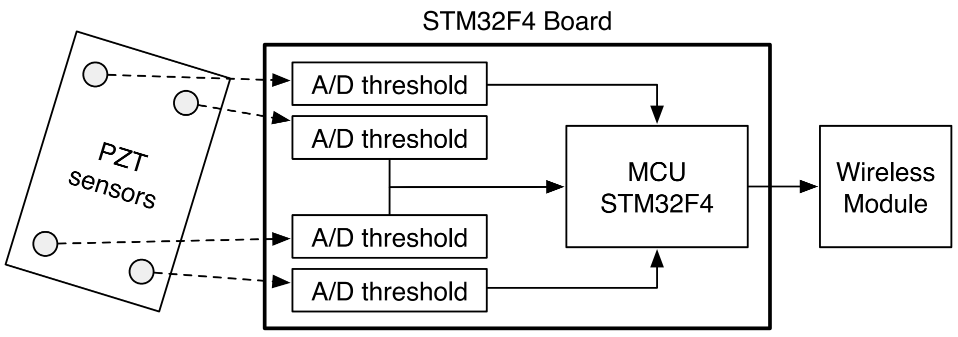


Figure 2.15: Structure of the embedded SHM device for impact detection

module.

Piezoelectric sensors When an impact occurs on an elastic structure, a stress wave is created and it propagates across the surface of the structure, radially from the point of impact. The stress wave can be caught by PZT sensors in a passive way. The proposed system implies at least 4 conventional piezoelectric transducers arranged in a geometrical fashion. The sensor array of piezoelectric elements needs to contain at least four sensor elements in order to provide reliable triangularization capabilities.

Processing electronic unit At the center of the system is the processing core which contains function modules for data collection, processing and communication control. A Cortex-M4 based board is selected as main chip in the processing core. The MCU is specifically a STM32F4 evaluation board featuring a STM32F407VGT6 micro-controller with 1 MB Flash and 192 KB RAM.

The strength point of the core is the CPU with FPU, adaptive real-time accelerator allowing 0-wait state execution from Flash memory and frequency up to 168 MHz.

Acquisition chain The proposed signal elaboration technique is based on an innovative time-frequency analysis techniques: by cross-correlating the dispersion-compensated signals the impact point can be determined via hyperbolic positioning.

This solution allows a noticeable reduction in cabling and circuitry, improving at the same time the energy efficiency due to implicit data compression and for the reduced number of transducers to be used during the actuation and sensing phase. PZT transducers are connected directly with the ADC ports of the STM32F4 board and each ADC channel is configured in dual mode with 250 kHz maximum sapling frequency since that spectral components of the Lamb waves lower rapidly above 60-100 kHz. The acquired values are stored in a DMA circular buffer; when the maximum value of the buffer exceeds the threshold value the trigger is sent and the MCU performs the localization algorithm.

Wireless transmission module When the device is used to monitor the structural health of large structures, it is possible to think to the network of devices as distributed intelligence. Each node in the network monitors a specific portion of the structure surface, eventually reporting to a central location in case of detected damage.

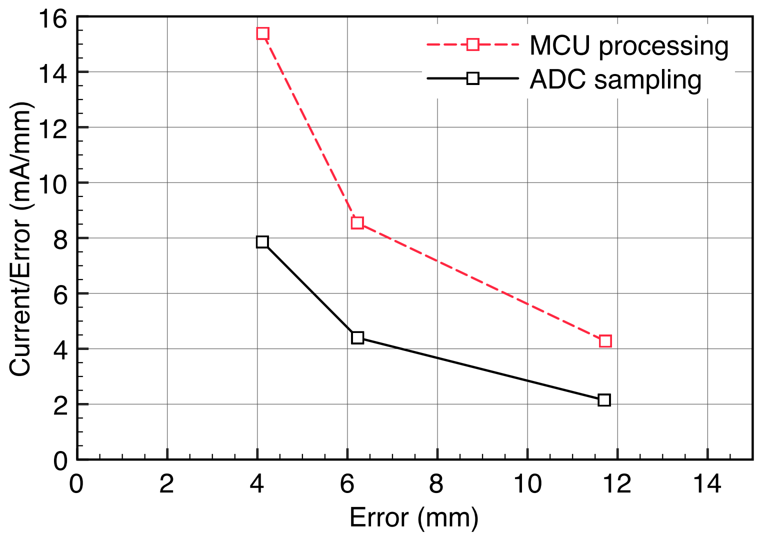
The wireless communication technology allows long distance data transmission without wiring, simplifying the difficulties in multi-device network monitoring.

To be compliant with the low-power requirements the device presents a RF wireless module ZigBee/IEEE802.15.4 compliant, connected to the main board using an SPI interface. The network topology suitable for this type of applications is mesh or star network, where each node in the network is able to communicate with the central gateway.

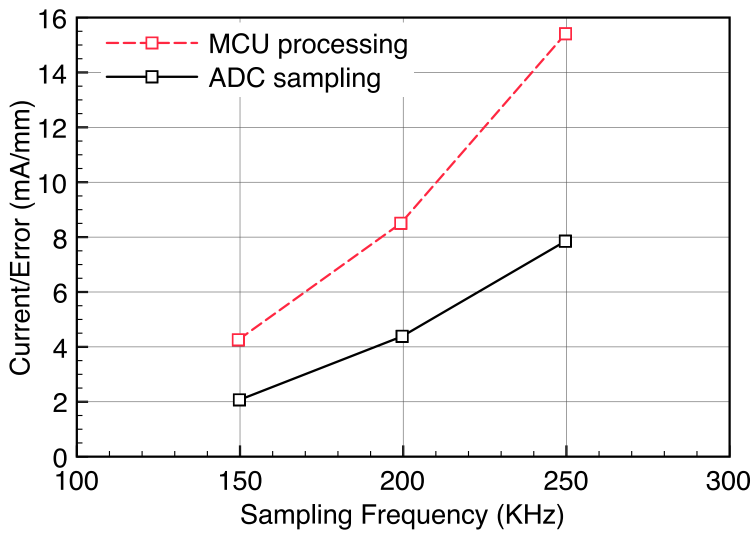
2.5.4.2 Experimental verification

The proposed system has been exploited to locate impacts in an aluminum 1050A square plate 1 m $\times$ 1 m and 3 mm thick. Four PZT discs (PIC181, diameter 10 mm, thickness 1 mm) were placed asymmetrically at the corners of a square.

Guided waves were excited by hitting the plate with a pencil orthogonally to the surface. The ADC channels of the STM32F4 discovery board are set in dual mode and continuously acquire signals with a maximum sampling frequency of 250 kHz. Data are recorded in a circular DMA buffer and acquisitions are triggered when the signal received from one of the PZT discs reaches a threshold



(a)



(b)

Figure 2.16: Current consumption and localization error vs (a) the error itself and (b) the sampling frequency

level of 50 mV. The sampling frequency was sufficiently high to avoid aliasing effects, as the frequency content of the acquired signal vanishes above 60 kHz. In order to analyze the dependency of the power consumption and the localization performances with the sampling frequency, experiments were carried changing the sampling frequency in the range [150-250] kHz.

Results in Table 2.5 show how lowering the sampling frequency, the current consumption decreases but not in a linear way; furthermore the MCU elaboration step is very sensible to the sampling frequency since the algorithm complexity is proportional to the sample buffer length which is reduced if the sampling frequency is lower.

Mean Current Consumption		
	ADC sampling	Signal Processing
$f_s = 250$ kHz	32 mA	63 mA
$f_s = 200$ kHz	27 mA	53 mA
$f_s = 150$ kHz	24 mA	50 mA

Table 2.5: Current consumption for different sampling frequencies in the SHM device

On the other hand lowering the sampling frequency the positioning error rises; in contexts such as wing monitoring, the high localization resolution is an important constrain because facilitates the decision to be taken in critical phases such as aircraft takeoff and optimizes the number of sensors to be used to monitor large areas. A good parameter able to take into account both the current consumption and the spatial resolution is the radio among the current consumption and the error as seen in Figure 2.16a while Figure 2.16b shows this ratio when the sampling frequency is changed.

Finally the frequency sampling choice depends on the localization and current constraints, knowing that if the location is inaccurate it is necessary to scan in a larger area since that the defect induced from the impact can be below the surface and therefore not immediately visible and then the control times get longer.

Chapter 3

Data reduction in WSNs

In the previous chapter the energy consumption for wireless sensor nodes has been analyzed with particular focus on radio and processing unit. As seen in Section 2.2, depending on the specific application, the sensing subsystem might be another significant source of energy consumption, thus its power consumption has to be reduced as well.

The techniques used in the reduction of the energy consumption derived from the sensing subsystem are usually referred to as *data-driven approaches*. The reduction in power consumption given by these techniques affects both the sensing subsystem and the radio subsystem since these techniques aim to reduce the number of bytes to be sent by the radio transceiver.

Data reduction through data-driven approaches has usually two different forms: (1) data aggregation and (2) encoding and compression.

In general these two techniques are supported in modern WSNs by middlewares. Middleware should provide data aggregation and data compression algorithms, giving the applications above the ability to choose the algorithm that best suits them. In addition, it may be useful to let the application injects its own in-network processing functions in the network.

Data aggregation Data aggregation, also called data fusion, is one of the most used techniques of in-network processing. The benefit of this technique (power saving) comes from the fact that transmitting data is considerably more expensive than even complex computations. This particular form of data management is called aggregation since data collected from different nodes is aggregated into a condensed form along the path from the node to the sink.

Encoding and compression While data aggregation does not transmit all the bits of measured data from sensors, with data encoding is still possible to reduce the number of transmitted bits but still obtain the full information about all sensor readings at the sink.This is feasible using special compression and encoding techniques. Moreover since sensor nodes are deployed close to each other in the physical environments, usually the readings of adjacent nodes are correlated. The compression and encoding techniques exploit such correlation, which can be a spatial one (in adjacent nodes) or a temporal one (readings at the same moment).

3.1 Middlewares

The trend in WSN of the usage of powerful 32-bit micro-controllers as seen in Section 2.5.3 is gradually shifting the problem to manage the complexity from the collecting sink to the nodes. The nodes are becoming powerful enough to be used actively for data elaboration and transformation but also for more complex operating systems and distributed applications. Even though this prospective introduces new possibilities in terms of data management and power consumption reduction, all this complexity has to be managed carefully.

To cope with this complexity one possible solution is to introduce abstraction layers and middlewares. Middleware refers to software and tools that can help hide the complexity and heterogeneity of the underlying hardware and network platforms, ease the management of system resources, and increase the predictability of application executions. WSN middleware is a particular kind of middleware providing the desired services for pervasive computing application using sensors and distributed networking.

The motivation behind the research on middlewares for WSNs derives from the gap between the high-level application requirements and the complexity of the underlying structure of WSNs. The requirement includes flexibility, re-usability and reliability while the complexity is due constrained resources, dynamic network topology and variety in hardware configuration.

In the early time of the research on WSNs there was no need for middlewares since the simplicity of the applications, due also to the poor performance of the 8/16-bit micro-controllers, did not require an intermediate layer of abstraction on top of the hardware. Along with the rapid evolution in this area, the usage of middlewares has become a requirement.

Practically a WSN middleware: (1) provides system abstraction so the application programmer can focus on the application logic without caring about low-level or algorithmic implementation, (2) provides hardware abstraction, decoupling the application layer from low-level details about hardware, (3) helps the programmer in network infrastructure management and power management.

While operating systems provide a useful environment for resource management and task scheduling, they are too general and usually do not facilitate the development of high level applications because of the wide variety of potential applications for WSN and the resource constraints of a limited platform.

For this reason nowadays middlewares are emerging as a valid alternative and in particular frameworks are becoming a valid tool to provide abstraction toward hardware, ensuring at same time a layer that improves interoperability and allows the reduction of development time.

The architecture of a framework is generally composed by three main components: (1) an application core to be included in each node of the WSN which provides a stub interposed between hardware and high level application hiding low level details, (2) a gateway forwarding sensor data from WSN to a backend server and (3) the server application which is responsible for data analysis and processing.

The opportunity to easily program and monitor a large number of nodes over a large area poses new challenges in terms of design, information processing and aggregation techniques given by these new frameworks.

A framework provides the programmer an intermediate abstract layer between hardware and software that avoids the additional work of reimplementing the same algorithm in different way according to the underlying specific hardware. Moreover the overhead involved in design protocols for data aggregation and dissemination can be prevented by delegating the problem to the framework. Such a middleware permits to implement applications, algorithms and policy in WSN hiding low-level details about hardware and effective communication protocols to the application developers.

In literature there are several middlewares or frameworks that at a certain extent share the same structure: DexterNet [Kuryloski et al., 2009], MobiCare [Chakravorty, 2006], Senceive [Chakravorty, 2006], CodeBlue [Kambourakis et al., 2007], SPINE1.3 [Iyengar et al., 2008] and others.

3.1.1 Case study: SPINE2

SPINE2 is a programming framework designed to speed up the prototyping and implementation of collaborative WSN applications. It provides a platform-independent layer making easier the porting process from one platform to another one [Fortino et al., 2009b]. It is composed by libraries of protocols, utilities and processing functions besides a set of java APIs used on java server interfaced with local gateway providing a set of routines to manage and control informations and data to and from nodes [Fortino et al., 2009a].

More in detail SPINE2 uses a task-driven model to program the sensor nodes, where each task corresponds to a different function and can be discovered, created, activated, scheduled and controlled independently on each node by the gateway using apposite java routines. Moreover a sensing and elaboration task can be performed entirely on node itself, aiming at a significant reduction of transmitted data and then increasing the battery life and lifetime of the system. Since the central coordinator has a general view of the state of WSN and it is responsible for tasks allocation, a dynamic approach can be used about distribution of the work, mapping different tasks according to changes in context, goals and state of the single node. This can be done using proper messages exchange permitting load balancing, exploiting parallelism in computation and granting QoS and faults tolerance as well.

SPINE2 is composed by a core written in C which adapts the hardware to different platform through specific interfaces as in Figure 3.1. The choice of using C is due to the large scale deployment of this language in the world of sensor platforms.

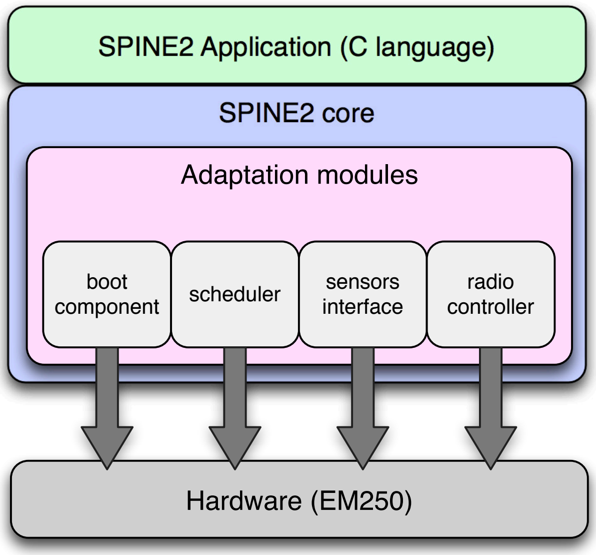


Figure 3.1: SPINE2 stack. The spine core separates and connects hardware and application level. The link between the core and the hardware goes through general interfaces to adapt to the specific node

With the software layering approach and the creation of the framework, the development of high level software and the framework itself can be decoupled, keeping the process of maintaining the code efficient and structurally separated.

In our cases SPINE2 framework has been ported on a Ember EM250 with EmberZNet ZigBee stack, a ZigBee System-on-Chip that combines a 2.4GHz IEEE 802.15.4 compliant radio transceiver with a 12MHz 16-bit microprocessor equipped with 128Kb of flash memory and 5Kb of RAM.

To port the framework on this hardware some software interfaces have been implemented and adapted to the node: (1) the spine boot component containing the initialization code, (2) the scheduler to manage tasks execution, (3) the sensor drivers through which real sensors data can be accessed and (4) the radio component which passes and forwards packet between SPINE2 core and the underlying radio controller.

3.1.1.1 SPINE2 on Ember EM250 platform

The chosen platform is a XAP2 core which is very versatile and has low power components. Albeit SPINE2 has a small foot print, during the porting several decisions about implementation of hardware dependent routines have been taken trying to overcome the limitations due to the lack of RAM memory and computational power. In Table 3.1 the summary of the memory occupation for the ported version of the SPINE2 is reported.

RAM Usage	
Used by globals, static, and call stack	3844 bytes
APPLICATION_CONFIGURATION_HEADER usage	1174 bytes
Available for future use	102 bytes
Flash Usage	
CODE segment	89180 bytes
Available for future CODE	15140 bytes
CONST segment	3442 bytes
Available for future CONST/INITC	4562 bytes

Table 3.1: Flash and RAM usage of SPINE2 framework

The table shows how the most critical point in the architecture is the RAM. Because of the lack of RAM we do not allow dynamic memory allocation, working only with statically allocated buffers. These two aspects rise the big problem to write code with preallocated buffers saving as much RAM as possible. Therefore each array in the code is accurately weighted paying attention to not allocate unnecessary space.

This is actually a limitation not directly related to SPINE2 but it has to be taken into account during the development of specific application functions (e.g. feature extraction). In fact heavy RAM-hungry functions should be carefully optimized to be integrated into SPINE2 and fit into the limited RAM free space.

Another point to take into account is the additional delay in operations brought by the presence of the interposed middleware. It increases also the energy consumption on the node which is largely compensated by the power saving in communication activity.

In the following paragraphs each interface of the frametwork is presented emphasizing how the abstraction is implemented against a limited resource hardware.

Boot component It is composed by preliminary code used to call the init functions for sensors and buffer pools.

Scheduler SPINE2 adopts a timer-driven task oriented model where each task (sensing, feature extraction, aggregation, data transmission) can be dynamically created, started and deallocated by specific packets from the coordinator. Each single operation performed by node is then represented by a periodic task linked to a handler.

Since using the Ember APIs it is not possible to create from scratch a periodic task, it is necessary to create a small set of generic periodic tasks which will be differently specialized according to the incoming configuration packets. The drawback of this approach is that the maximum number of tasks allowed is fixed and simultaneously the tasks cannot be too many since each additional task takes 34 bytes of RAM.

```
typedef struct {
  EmberEventControl control;
  taskDescription td;
  unsigned char set;
} td_ember;
...
td_ember task[MAX_EVENTS];
EmberEventData events[MAX_EVENTS+1] =
{
  { &(task[0].control), schedulerHandler },
  { &(task[1].control), schedulerHandler },
  { &(task[2].control), schedulerHandler },
  { NULL, NULL }
};
```

Generally three tasks are required: one to retrieve a sample from sensors, the second one to extract a feature every n samples and the last one to send out the data packet containing the result of the operation of feature extraction.

The *schedulerHandler()* routine is the generic function responsible for handling and periodic rescheduling of the task. Even if the restrict number of allocable tasks limits the set of possible operations of the node, these are enough to permit the use of SPINE2 and Ember motes in a real environment. Considering the limited resources available on the node, performing more than 3 or 4 tasks in parallel is not advisable, this is why the prefixed number of allocable tasks is not a major limitation in SPINE2.

Sensor interface The sensor interface deals with sensors access and data acquisition. It is a very critical part since each additional delay introduced by the middleware limits the maximum sampling frequency of the sensors. In our test a synchronous reading process is implemented: during the access to ADC or SPI interfaces the normal program execution is paused until the reading is completed.

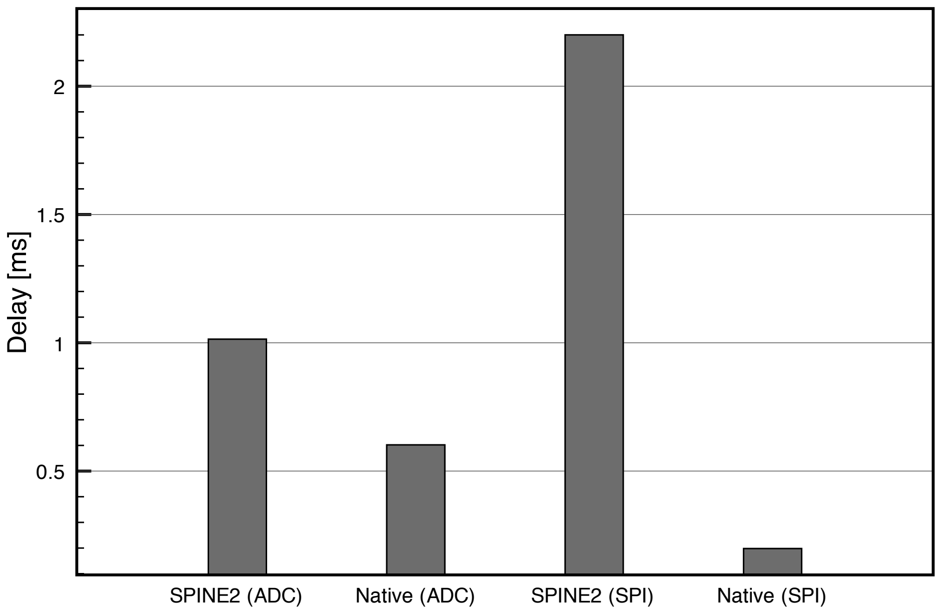


Figure 3.2: Comparison of delay in reading sensor data between SPINE2 and native Ember routines

In Figure 3.2 delays in performing a sensor reading (both ADC and SPI) are presented. It is clear, especially using a SPI channel, that the presence of intermediate code and functions between application and low level routines for accessing sensors involves a high delay, actually bounding the sampling frequency for these sensors.

Although SPINE2 cannot be employed in tasks with strict real-time constraints and applications requiring a very high sampling frequency, the delay introduced is small enough to permit the use in generic applications.

Radio controller The radio controller is responsible for sending data over the air. SPINE2 defines the payload content and the lower interfaces assemble a real packet and send it to coordinator using hardware specific APIs.

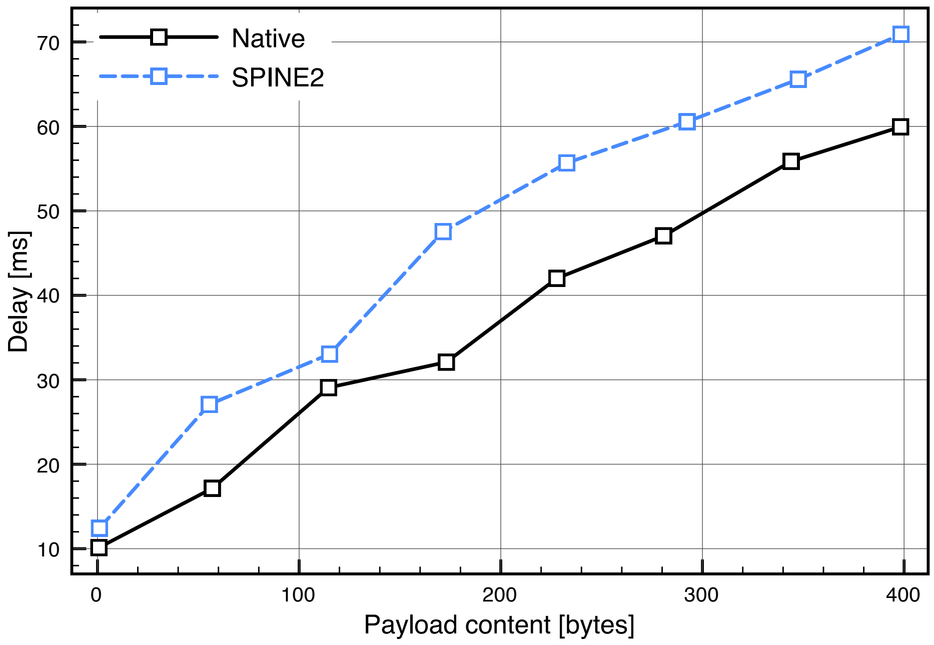


Figure 3.3: Delay for sending data in function of data size. Delays are measured considering the time elapsed between the call of the sending routine and the callback function indicating the successful dispatch of the data packet

As seen in Figure 3.3 the presence of SPINE2 brings in more delay in sending data. This is due to two different reasons: (i) a computational overhead to assemble the packet constituted by the payload and an additional SPINE2 specific header, and (ii) the specific header itself that increases the number of packets to send keeping unchanged the bytes of data to incapsulate (in normal circumstances the maximum payload allowed by Ember APIs in each packet is around 100 bytes).

This increases the delay in communication between nodes and power consumption due to the additional number of packets sent over the network. Therefore the choice of using SPINE2 should be narrowed to applications in which the delay is not a major problem and the additional power consumption given by few more packets is compensated by a better overall energy saving policy.

3.1.1.2 Application scenarios

The limitations of hardware and implementation stressed in the previous paragraph regarding the delays in sensor reading and data sending do not prevent the usage of SPINE2 in practical cases where the use conditions are not extreme or atypical.

We show two interesting real applications in which, despite limitations, SPINE2 achieves a general behavior of the system that is not achievable without using a framework as intermediate abstraction layer.

These case studies point out different aspects in which SPINE2 can be successfully exploited: (1) energy and power savings: the possibility to operate in-network processing and in-node computation avoiding useless and power hungry communications, (2) load balancing and redistribution: the WSN coordinator can map different tasks to each node according to the application context, local computational load and amount of energy available, (3) data aggregation and dissemination: data can easily spread inside the network without caring about lower network protocols and hardware, focusing only on application and algorithm implementation.

Fall detection

The major contributions to power consumption in sensor nodes are: (1) the power consumed by the digital part for acquisition and elaboration of sensor data and (2) power for inbound and outbound transmissions.

Therefore to apply power-aware techniques we can operate in two different domains: (1) communication protocols and (2) communication-aware software.

Consider a network where a node has to gather data from sensors on which it is necessary to perform some kind of operation. Data analysis can be performed on another node in the same network or on the node itself. This two choices have a very different impact on energy consumption, involving a different amount of packets sent within the WSN. If digital signal processing is performed by the acquiring node, the output transmission is avoided, saving the energy spent in communication.

According to this new vision the node is not only a passive device that senses and sends data to a coordinator but it becomes a true computational unit. Shifting the analysis from a centralized point to a distributed elaboration creates new problems in coordination and control that cannot be faced without resorting to a new approach. The energy saving obtained with a local computation is compensate by an increasing difficulty in configuration and control over the network, opening a new trade-off between power saving and configuration.

The simpler case is that one in which a specific algorithm has to be implemented over a set of data acquired by a remote sensor node. In this case the node is used simply as a data forwarder towards central coordinator that can be configured by the user with specific parameters. Since each sample coming from environment is sent toward the gateway a lot of energy is wasted in communication, proportionally to sampling period and acquisition time.

Moreover with this configuration the coordinator should have enough computational power to serve each request and elaborate each data stream, hence the system is not scalable at all.

Through the usage of a specific framework for signal processing in node environment (SPINE2) the allocation of specific tasks on sensor nodes and the acquisition of the result is a very simple procedure. SPINE2 permits to dynamically allocate and start (but also to stop and to deallocate) tasks on a specific node, configuring it over-the-air. The data effectively sent to the coordinator is only the result of the on-node elaboration.

A very successful clinical application of this framework is the fall detection in which the acceleration is taken as parameter to monitor for detecting the fall.

A series of nodes equipped with accelerometers are posed on several parts of the body with the aim to capture accelerations along the three axis: according to the value of a special parametric function a movement is classified as a fall or not. The classification is done by comparing the value of the estimation function with an opportune threshold [Chen et al., 2005]; if the value of acceleration is above the threshold then the movement is classified as a fall.

A very common scenario proposed in literature is that one in which the system is composed by several sensor nodes connected over a WSN to a central coordinator, that generally is a portable device able to run Java and log events. If the framework is not available the developer has to implement fixed estimation functions with immutable threshold, proceeding to write a new firmware for each change in parameters.

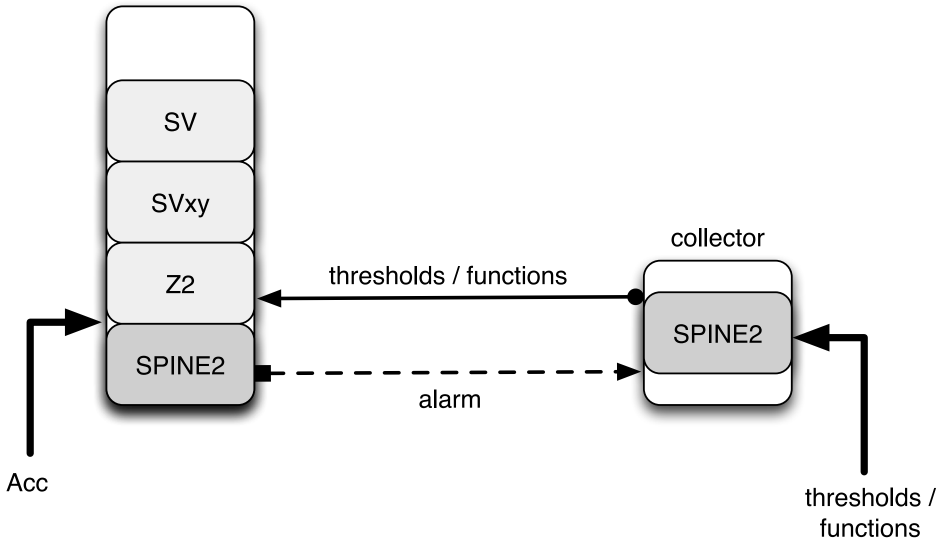


Figure 3.4: SPINE2 in fall detection applications reduces energy consumption of the node. The coordinator can choose and change the evaluation function and value of thresholds

In this context SPINE2 provides an abstract and configurable layer making available the entire set of functions for fall detection, as represented in Figure 3.4. Coordinator can choose and exchange at run-time the function used to evaluate the fall using few configuration packets. Moreover using Java APIs it is possible to modify value and typology of thresholds used to identify the fall.

Since this type of characterization is node oriented, different kind of functions and thresholds can be used for different sensors. This highly dynamic configuration mechanism is extremely power-aware: a single packet of alarm is sent back to coordinator when a fall is detected. Every other data processing is performed on the node using parameters received by coordinator.

During our tests we have evaluated through an oscilloscope the power consumption involved in sending data. The trace extracted shows the voltage across a precision resistor used to evaluate the absorbed current by the device. The visible spike is related to a peak in power consumption up to 120mW in correspondence to the packet sent to coordinator. The power consumption is always at minimum level since the radio is always off except when a packet has to be sent and when the node polls the coordinator asking for configuration packets.

This is the case in which the problem of the increased number of packets circulating within the network because of SPINE2 is largely compensated by the number of those ones not sent over-the-air (using dynamic on-node processing permitted by middleware). Even though SPINE2 brings additional extra packets and then energy consumption, the global energy balance is positive and favorable to adoption of the middleware that permits to exploit better packet management policy and save the number of payload byte sent.

Concluding the use of a framework infrastructure like SPINE2 makes the system globally energy-aware with the advantage of a system highly configurable on-the-fly.

Model Predictive Control

The second case study is Model Predictive Control (MPC). It is a very interesting feedback strategy in which linear models are used to predict the system dynamics even though the dynamics of the closed-loop system is nonlinear. The main idea of MPC is to select a control action solving on-line an optimal control problem. The aim is to minimize a specific cost function over a future horizon considering the constraints on the manipulated input and output, where the future behavior is extracted according to a model of the plant [Nikolaou, 2001].

Practically at a certain time t the current state plant is sampled and the state is used as parameter in a cost function to minimize on-the-fly, obtaining a cost-minimizing control strategy for a short time horizon in the future $[t, t + T]$. After the implementation of the control strategy, the horizon is shifted one step ahead and the algorithm computes the new strategy. An application using MPC involves the deployment of sensor nodes to gather data about the system state and solving an optimization problem requires a lot of computational effort. The limited resources of nodes in a WSN are not enough to face such a complexity. For this reason in most cases data containing the state of the controlled system are sent to a device with more computational power able to solve the optimization problem. Once the new vector containing the control policy is computed, this is sent back to the nodes that can perform the control.

A typical situation is that one in which the two sensor nodes represented are working on the same system or the inputs are correlated in some way (for example they are in the same environment and performing some kind of measurements on it). This means that the actions performed are not independent one each other and the information about the state of a node is correlated to the state of the other one.

For instance we can have two nodes that receive their energy from solar cells as in [Moser et al., 2007]. In this case during the computation of the best state trajectory it is necessary not only dealing with the minimization of power consumption but also considering the time-varying amount of energy available.

The system model is presented in Figure 3.5 in which two SPINE2 running nodes are powered by energy harvesting devices that at each time t supply energy $E_S1(t)$ and $E_S2(t)$ to the energy storages. At the same interval t part of the energy, $E_D1(t)$ and $E_D2(t)$, is consumed by the node, leaving an amount of energy $E_C1(t)$ and $E_C2(t)$ for further use.

On the device there are two main tasks running: the application (depending on the goal of the WSN) and the estimator that predicts future energy production of the harvester, based on past history.

The controller is left outside the nodes, for two main reasons: (1) it is computationally the most intensive part and nodes are not able to perform such a heavy calculations on a resource-constrained hardware as an embedded system and (2) a

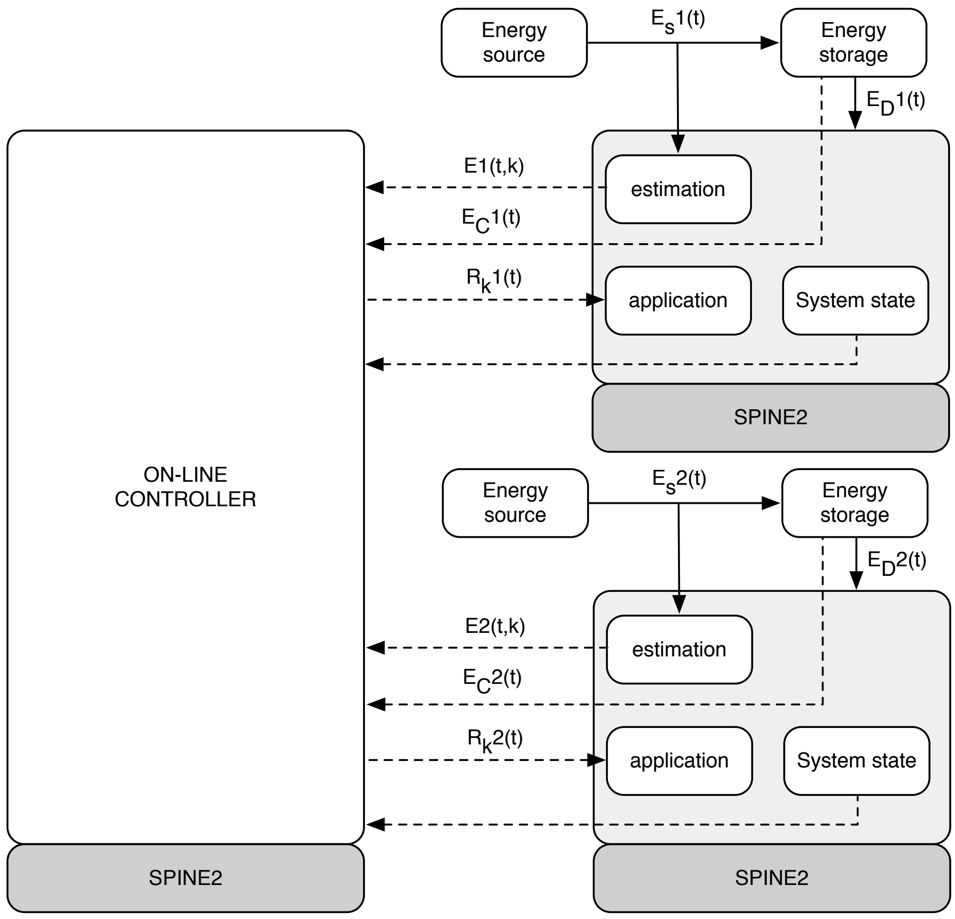


Figure 3.5: MPC scheme using SPINE2 as underlying framework. Data elaboration is not performed on-node but an external controller process data coming from WSN to extract an optimal solution

centralized controller can consider the performance and state of both the sensor nodes (and in general of the whole network), to develop a better control strategy.

The controller adapts properties of the applications, $R_K1(t)$ and $R_K2(t)$, based on the estimation of future available energy, the energy currently stored and the information about the system state, to optimize the overall objective respecting the system constraints.

The solution of the LP problem involves not only a non-negligible computational power but also the knowledge of the parameters of two systems. For this reason the on-line controller cannot be inside the nodes, but it has to be an extern system in charge of computing the optimal solution.

Even in this case the framework infrastructure of SPINE2 allows to build up the entire system without caring useless details not related to MPC problem. The SPINE2 core on the nodes takes care of communication between node and controller while the gateway is a proxy toward an implementation of resolution algorithm for the LP problem that can reside on the gateway itself.

SPINE2 makes easy the process of aggregation and dissemination of packets inside the network but the performance of the network itself is very important to ensure the correct application of the control command. Since the framework hides the protocols details about network communication it is always necessary to be sure that delays does not affect the packet relay.

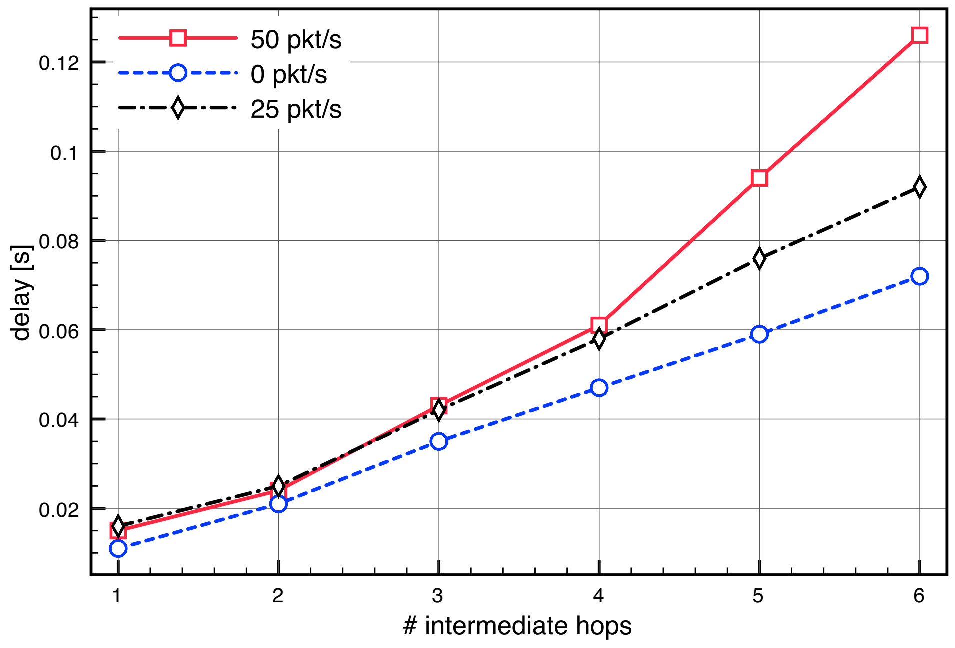


Figure 3.6: Transmission delay vs number of hops in a multi-hop ZigBee network when varying the traffic traversing each intermediate hops

This is true especially in multi-hop networks where the path of packets is not fixed but it is function of a series of parameters that cannot be monitored run-time. In Figure 3.6 a plot is presented representing the delay inside a ZigBee network in function of the number of hops traversed by a packet, using as parameter the outbound traffic (in packet/s) affecting each hop. The experimental setup is composed by ZigBee Ember EM250 nodes configured as routers. The path to follow within the network is hard coded inside the payload of the packet itself while the nodes are subject to a high rate of packets sent towards the coordinator.

For a big network, a long path of 8 hops and with high traffic the delay is not above 130ms. This value is not able to affect significantly the control considering that the resolution process of the LP problem is longer and therefore determines the throughput of the entire system.

As seen before SPINE2 introduces also a slightly delay in sending routines and data acquisition. If the sampling period of sensors is not too short the delay in reading sensors is negligible. In particular, the MPC can add some constraints on sampling rate to avoid a too frequent access to sensors. The delay in sending routines can be neglected because the additional average delay of 10ms does not compromise the resolution of the linear programming problem which is more intense and time demanding.

3.2 In-network aggregation

In the typical scenario for WSN data is gathered by the several sensor nodes of the network and data is made available at same central sink node, where it is processes, analyzed and used by application. In many cases the data gathered and sent by sensors can be jointly processed while being forwarded to the data sink. In-network processing techniques deal with this distributed processing of data within the network.

Data aggregation techniques are tightly coupled with data representation at sensor nodes and how packer are route through the network. Both these aspects have a significant impact on energy consumption and network efficiency.

Briefly in-network aggregation is the process of gathering and routing information through a multi-hop network processing the routed data at the intermediate nodes such as to reduce the energy consumption and then to increase the network lifetime.

In general it is possible to distinguish between two general approaches: (1) in-network aggregation without size reduction and (2) in-network aggregation with size reduction.

In *in-network aggregation without size reduction* data are not elaborated by intermediate nodes but different data can still be packed in less packets exploiting remaining free space.

In *in-network aggregation with size reduction* the aim is to reduce the amount of data that has to be sent by each node in the network. This is usually done combining and compressing data coming from different sources. This approach is better than the previous one in terms of energy saving but usually it reduces the accuracy of the reconstruction at the sink since it is not possible to perfectly reconstruct all the original data while with the first approach preserves the original information.

Moreover the in-network aggregation with size reduction can adopt two different approaches: lossy and lossless. In the first case the original values cannot be recovered exactly after having merged data during aggregation. In addition we may lose precision with respect to transmit all recordings uncompressed. The lossless approach allows us to compress the data by preserving the original information.

Among the most recent and famous approach for data compression through in-network aggregation is the so called Compressive Sampling (CS).

3.3 Compressive Sensing (CS)

CS theory [Candes et al., 2006][Donoho, 2006] shows that if a signal has a sparse representation in one basis then the signal can be recovered from a small number of projections onto a second basis that is *incoherent* with the first.

CS relies on tractable recovery procedures that can provide exact recovery of a signal of length N and sparsity K, i.e., a signal that can be written as a sum of K basis functions from some known basis, where K can be orders of magnitude less than N.

The implications of CS are promising for many applications, especially sensing signals that have a sparse representation in some basis. Instead of sampling a

K-sparse signal N times, only cK incoherent measurements suffice, where K can be orders of magnitude less than N. Therefore, a sensor can transmit far fewer measurements to a receiver, which can reconstruct the signal and then process it in any manner. This is particularly important for sensor nodes in WSN where to prolong the node lifetime is necessary to reduce as much as possible the number of bytes sent toward the coordinator.

Moreover, the cK measurements need not be manipulated in any way before being transmitted, except possibly for some quantization.

Independent and identically distributed (i.i.d.) Gaussian or Bernoulli (random ± 1) vectors provide a useful universal basis that is incoherent with all the others. Hence, when using a random basis, CS is universal in the sense that the sensor node can apply the same measurement mechanism no matter what basis the signal is sparse in (and thus the coding algorithm is independent of the sparsity-inducing basis).

3.3.1 CS: a mathematical background

Consider a generic signal x of length N indexed as $x(n), n \in \{1, 2, \ldots, N\}$. Suppose that the bases $\Psi = [\psi_1, \ldots, \psi_N]$ provides a K-sparse representation of x, that is:

$$x = \sum_{n=1}^{N} \theta(n)\psi_n = \sum_{\ell=1}^{K} \theta(n_\ell)\psi_{n_\ell} \tag{3.1}$$

where x is a linear combination of K vectors chosen from the basis Ψ, $\{n_\ell\}$ are the indices of those vectors and $\{\theta(n)\}$ are the coefficients. The same formula can be written in matrix notation:

$$x = \Psi\theta \tag{3.2}$$

where x is an $N \times 1$ column vector, the *sparse basis* matrix Ψ is $N \times N$ with the basis vectors ψ_n as columns and θ is an $N \times 1$ column vector with K nonzero elements.

The K-nonzero elements define the sparsity of the signal that can be indicated using the $\| \cdot \|_p$ notation where the ℓ_p norm is defined as:

$$\|x\|_p = \left(\sum_{i=0}^{N-1} |x_i|^p \right)^{1/p} \tag{3.3}$$

therefore we can write that $\|\theta\|_0 = K$.

The CS applies not only to exactly K-sparse signals but also for signals which coefficients decay rapidly but not to zero.

The standard procedure to compress signals, known as *transform coding*, is (1) to acquire all the N samples of the signal x, (2) to compute the complete set of transform coefficients $\{\theta(n)\}$, (3) to locate the largest K coefficient and discard the remaining, (4) to encode values and locations of these largest coefficients.

This procedure is highly inefficient since we must sample the all signal gathering all the N samples and we must encode all the transformation coefficients to be able to retain just the most significant. Finally the encoder has the encode value and location for each coefficient.

CS just find solution to these problems directly esimating the set of the largest coefficients without going through the sampling of the whole signal.

3.3.2 Incoherent projections

To be more precisely with CS we are not able to directly acquire the largest coefficients. Instead we measure and encode $M < N$ projections $y(m) = \langle x, \phi_m^T \rangle$ of the signal onto a second set of basis functions $\{\phi_m\}$ with $m = 1, 2, \ldots, M$, where ϕ_m^T denotes the transpose and $\langle \cdot, \cdot \rangle$ is the inner product.

So we have:

$$y = \Phi x \tag{3.4}$$

where y is an $M \times 1$ column vector and Φ is the $M \times N$ *measurement basis* with each row is a basis vector ϕ_m.

Is is clear that being $M < N$ the problem in Equation (3.4) is ill-posed.

Actually CS states that under certain conditions it is possible to recover the original signal. In particular when the basis $\{\phi_m\}$ cannot sparsely represent the element of the basis $\{\psi_n\}$ (*incoherence*) and the number of measurements M is large enough then it is possible to find solution to Equation (3.4).

Fortunately, random matrices are largely incoherent with any fixed basis Ψ. We can construct the random measurement basis Φ in several ways:

- *Gaussian measurements.* In this case we assume that the entries of the matrix Φ are independently sampled from the normal distribution with zero mean and variance $1/K$ if matrix is $K \times N$.

- *Binary measurements.* The entries of the $K \times N$ matrix Φ are independently sampled from the symmetric Bernoulli distribution with $P(\Phi_{ki} = \pm 1/\sqrt{K}) = 1/2$

- *Fourier measurements.* We suppose that the matrix Φ is a partial Fourier matrix obtained by selecting K rows uniformly at random as before and normalizing the columns.

3.3.3 Signal recovery

To recover the original signal from its compressed version we can resort to different algorithms.

Signal recovery via ℓ_0 optimization

The recovery of the coefficients $\{\theta(n\}$ can be achieved solving optimization problems by searching for the signal with ℓ_0-sparsest coefficients that agree with the M observed measurements y.

The problem with this approach is that solving the ℓ_0 optimization problem is extremely complex and known to be NP-complete.

Signal recovery via ℓ_1 optimization

A much easier problem with an equivalent solution is the ℓ_1 minimization. We need only to solve for the ℓ_1-sparsest coefficient θ that have the measurements y, such that:

$$\hat{\theta} = \arg\min \|\theta\|_1 \qquad \text{s.t.} \qquad y = \Phi\Psi\theta \tag{3.5}$$

this problem is known as *Basis Pursuit* [Chen et al., 1998] and it can be solved with traditional linear programming techniques polynomial in N.

To recover the original signal we require $M > cK$ measurements where $c > 1$ is an oversampling factor.

Usually we can assume that the sparsity scales linearly with N; that is $K = SN$ with S is the *sparsity rate*. CS theory states that if we have $0 < S \ll 1$ then there exist an oversampling factor $c(S) = \mathcal{O}(\log(1/S)) > 1$ such that, for a K-sparse signal x in basis Ψ, the probability of recovering x via Basis Pursuit from $(c(S) \pm \epsilon)K$ random projections, $\epsilon > 0$, converges to one as $N \to 0$. Clearly the probability increases with the number of measurements $M = cK$.

A rule of thumb for defining $c(S)$ is:

$$c(S) \approx \log_2(1 + S^{-1}) \tag{3.6}$$

3.3.4 Random measurements

In addition to offering substantially reduced measurement rates, CS has many attractive and interesting properties, particularly when we employ random projections at the sensors. Random measurements are universal in the sense that any sparse basis can be used, allowing the same encoding strategy to be applied in different sensing environments. Random measurements are also future-proof: if a better sparsity-inducing basis is found for the signals, then the same measurements can be used to recover a more accurate view of the environment. Random coding is also robust: the measurements coming from each sensor have equal priority, unlike Fourier or wavelet coefficients in current coders. Finally, random measurements allow a progressively better recovery of the data as more measurements are obtained; one or more measurements can also be lost without corrupting the entire recovery.

3.4 Data aggregation using CS in WSNs

As seen in the previous sections CS does not require any priori information about original data but sparsity rate and it could be used to performs in-network aggregation without a central coordination. When considering CS in WSNs we can think to the original signal to compress as generated in the sensor network itself. In particular a generic signal originated in a WSN with N nodes can be seen as a data vector created by sensor readings. This vector can be referred to as $x \in \mathbb{R}^N$ and indexed as $x(n), n \in \{1, 2, \dots, N\}$.

Compression can be performed jointly with routing while decoding affects only the collector that is usually a power grid connected gateway capable of complex calculations. The problem with this approach is that even though CS is a promising approach for in-network distributed data compression, it presents several challenges mainly related to packet size constraints imposed by standard media access control (MAC) protocols.

3.4.1 Practical case study

As we have seen before, the performance of CS is strictly related to the characteristics of the original signal and, in particular, to its sparsity or compressibility. Even though natural signals are not exactly sparse, fortunately CS does not apply only to sparse signals but also to compressible signals. We can say that a generic signal x is S-compressible in Ψ if considering the coefficients vector $\theta = \Psi^T x$ sorted by absolute value, namely $\hat{\theta}$, it has entries with magnitude decaying with $|\hat{\theta}(n)| \leq C \cdot n^{-S-1/2}$ for all $n = 1, \dots, N$ and $C < \infty$. In this case the compression is able to keep only the greatest coefficients carrying the most informative content.

Since the difference in terms of compressibility between natural and artificial signals is well demonstrated in literature [Quer et al., 2009] it is always important to deal with real signals obtained by real deployments. As practical example we can consider the sets of measurements gathered by LUCE deployment of EPFL SensorScope WSN, reporting ambient temperature recorded by 100 weather station deployed on the EPFL campus. Since for each node the GPS location is provided, it is possible to reconstruct the temperature field by interpolating and extrapolating data on the square area considered, as shown in Figure 3.7.

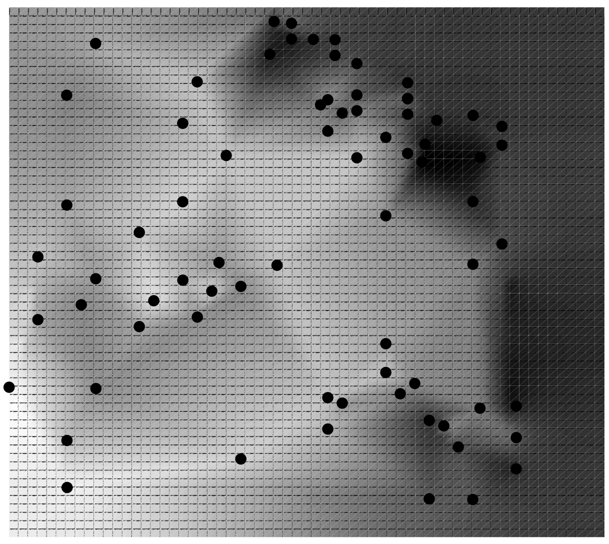


Figure 3.7: Temperature field reconstruction from LUCE deployment (March 28, 2007, 12.00AM)

As discussed in Section 3.3.3, the quality of the reconstruction and the number of measurements needed to achieve a perfect reconstruction with high probability is a function of the sparseness or compressibility of the original signal. In particular the solution of the problem in Equation (3.5) requires $M > cK$ random projections where $c(S) = \mathcal{O}(\log(1/S))$ where S is the sparsity rate of the signal. If the sparsity rate is function of the signal dimension we can define $S = K/N$ and give a more practical formulation for $c(S)$ that is $c(S) \approx \log_2(1 + S^{-1})$.

Using these relations the number of required measurements is:

$$M(N) > \log_2 (1 + S^{-1}) \cdot S \cdot N \tag{3.7}$$

where the number of measurements is mainly function of: (1) the sparsity rate S and (2) the length N of the original signal.

Regarding N, the bigger the network (and the size of the signal), the greater the number of random projections we need. On the contrary, S is considered constant and depending only on the signal to encode.

Since our aim is to monitor ambient temperature, S has to be obtained by such a kind of spatial monitoring. To evaluate the sparsity rate S we can randomly choose at least 10 days equally distributed during the solar year (to take into account the variation of temperature during the year) and for each hour of the day we can reconstruct the temperature field. To each reconstructed temperature field a DCT (Discrete Cosine Transform) can be applied to obtain the transform coefficients. The choice of DCT is due to the consideration that temperature signal appears to be smooth and therefore suitable for a transform coding like DCT.

The goal is to evaluate the coefficients percentage that we require to reconstruct the original signal with a good reconstruction quality to have an indication about the value of sparsity rate S. The reconstruction quality is evaluated using the *Normalized Mean Square Error* (NMSE) defined as:

$$\text{NMSE} = \frac{\sum_{n=1}^{N} (x(n) - \hat{x}(n))^2}{\sum_{n=1}^{N} x(n)^2} \tag{3.8}$$

where $x(n)$ is the original signal and $\hat{x}(n)$ is its reconstruction.

If we indicate with $\hat{x}(S, N)$ the reconstruction of the signal $x(n)$ when only a fraction S of its greatest DCT coefficients are left different from 0 (meaning that the fraction $(1 - S)$ of its smallest coefficients are set to 0) then NMSE(S) provides and indication about the real sparsity of the signal. In fact, if NMSE(S) is sufficiently close to zero, then the S percentage of the DCT coefficients are enough to represent the whole original signal with a good approximation. In this case, the temperature, as the natural signals in general, is compressible in the sense that the sorted magnitudes of the DCT coefficients decay quickly; then $x(n)$ is well approximated by $\hat{x}(n)$ and therefore the reconstruction error is small.

In performed tests, we set a NMSE threshold to 10^{-5} and then we evaluate the sparsity rate as $S = \{S|\text{NMSE}(S) \simeq 10^{-5}\}$. This value for NMSE is small enough to achieve almost a perfect reconstruction of the original signal through an inverse DCT transformation.

In Figure 3.9 the trend for the sparsity rate S is presented averaged on 10 days. In the simulations, $S = 0.05$ seems to be a good compromise and it is used as reference sparsity for the temperature signal considered.

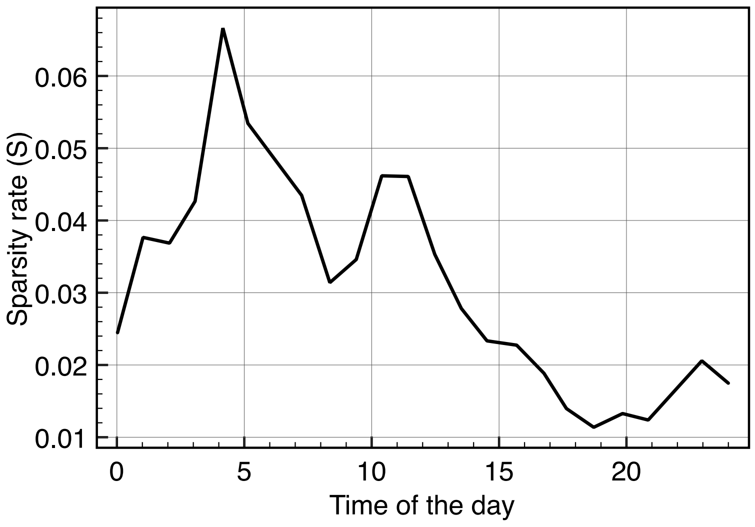
With regard to the network, we consider N nodes to be deployed in a square area with nodes placed in a uniform fashion. If the sensors are placed in a random position in a regular grid, the location of each node can be seen as a sampling location greatly simplifying calculations [Haupt et al., 2008]. The assumption is that the scalar value of the temperature to be recorded by the node is constant within each cell.

The sink is placed at the center and in the network a geographical routing is adopted [Al-Karaki and Kamal, 2004]. Each node can communicate with just one other node in the network that is the closest one to the sink within the communication range R_c. In the tests, R_c is chosen to guarantee communication among all nodes.

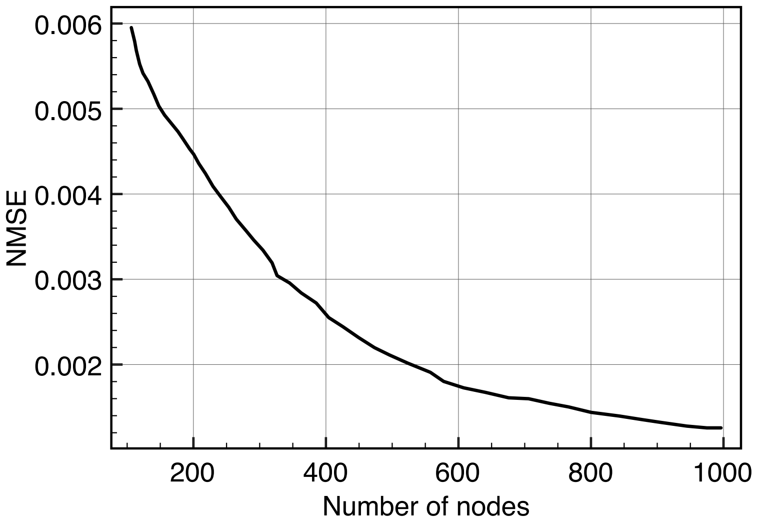
In Figure 3.9 an example network is represented where geographical routing is adopted.

Temperature is not a signal subject to fast changes; thus, data collection is performed every 1 minute, spending the rest of time in sleep mode to preserve battery power. For the sake of simplicity, the coordination among nodes is neglected and a perfect synchronization among nodes can be assumed.

To evaluate whether the chosen value for S is good enough to permit a reliable reconstruction of the signal, CS is used to reconstruct several random temperature signals averaging the NMSE of the reconstruction over 100 simulations using the proposed network model and a measurement vector of length calculated using Equation (3.7). The results are presented in Figure 3.8b. It is possible to infer that the value chosen for S permits a reliable reconstruction of the original



(a)



(b)

Figure 3.8: (a) Sparsity rate S function of the time of the day averaged on 10 days (NMSE=10^{-5}) (b) NMSE for 10 reconstructed temperature signals using CS averaged over 100 runs

temperature field, especially for big-sized networks. To increase the reconstruction quality if we need high accuracy, it is possible to increase the size M of the measurement vector.

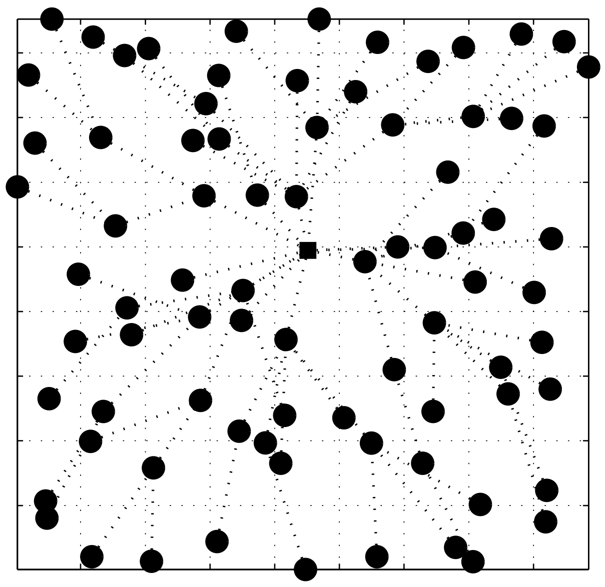


Figure 3.9: 9 × 9 sensor network example adopting geographical routing

3.4.2 Data gathering and compression

Given the dataset and scenario introduced in the previous sections, we want to investigate the performance of CS as in-network aggregation technique to be used in WSN. Therefore we consider a real-life scenario where a ZigBee / IEEE 802.15.4 network is used for temperature monitoring and we investigate the performance of both CS and a classical data gathering technique performing in-network aggregation without size reduction.

3.4.2.1 Pack and Forward (PF)

PF is an energy-safe strategy in which each node tries to encapsulate data in the most efficient way, minimizing the number of outgoing packets. This is a slightly modified version of the classical relay scheme usually adopted in WSNs in which each node on the path toward the sink forwards the incoming packets to the next hop according to its own routing table [Fasolo et al., 2007]. If the network has just one aggregation point and the number of nodes is large, this technique turns out to be very energy unaware, wasting communication power proportionally to the number of packets to relay.

PF takes advantage of the fact that nodes in the network are fixed in space and, assuming that the routing tables are built during the network bootstrap, each node knows the number of children nodes [Cuomo et al., 2007]. In this way, the node, before sending its own data to the next hop, waits for the complete transmission by its child nodes, computing an aggregated payload before sending out the final aggregated data. Since, in general, the size of transmitted data is smaller than the IEEE 802.15.4 payload, which is 127 bytes in size, the PF mechanism allows to pack several payloads within a single packet.

The information the decoder requires to know for signal reconstruction is constituted by two field: (1) the scalar sensor reading and (2) an identification ID of the node used during the reconstruction phase to identify the source and to associate a scalar reading to a cell in the grid.

If we indicate with B_{data} the number of bytes needed to encode the scalar reading and with B_{ID} the size in bytes of the identification field, the number of packets sent by the k-th node without fragmentation is:

$$P_k^{PF} = \lceil ((N_{tch} + 1) \cdot (B_{ID} + B_{data}))/B_{payload} \rceil \tag{3.9}$$

where in case of IEEE 802.15.4 $B_{payload} = 127$ bytes. N_{tch} is the number of nodes performing relay on the node. Considering that the path is always directed toward the sink, the value of N_{tch} is greater for nodes closer to the sink. Moreover on equal hop distance from the central sink, the value increases with increasing in network size and number of nodes. Thus being P_k^{PF} function of N_{tch}, in general the number of total packets circulating in the network increases when the number of nodes increases.

Since the node is not aware of the value N_{tch} but it only knows the number of its children nodes, this relation can be rewritten as:

$$P_k^{PF} = \left\lceil \left(\sum_{j=1}^{H} B_j + (B_{ID} + B_{data}) \right) / B_{payload} \right\rceil \tag{3.10}$$

where H is the number of child nodes for node k and the term B_j is the number of bytes received by the j-th child node.

3.4.2.2 Compressed Sensing (CS)

CS is a powerful approach to network data compression because it does not require any information about signal to compress (except the sparsity rate of the signal) and data encoding is performed jointly with routing in a distributed fashion.

As we have seen in Section 3.3.2 we can write the process of data acquisition and compressing as:

$$y = \Phi x \tag{3.11}$$

where $\Phi = [\phi_1, \phi_2, \ldots, \phi_N]$ is the measurement matrix of size $M \times N$ with $M < N$, y is the $M \times 1$ compressed version of x.

Each element y_i of the compressed version is obtained by the whole vector x weighted with a row taken from Φ. The equation Equation (3.11) can be written also as:

$$y_i = \sum_{j=1}^{N} \Phi_{ij} x_j \qquad i \in \{1, 2, \ldots, M\} \tag{3.12}$$

where Φ_{ij} is the (i, j)-th element of the measurement matrix, x_j is the scalar data sensed by the j-th node and y_i is the i-th element of the compressed output vector. It is clear that each node can add its own contribution $\{\phi_{ij}\}_{i=1}^{M} x_j = \phi_j x_j$ to the global sum independently one each other. The only information required by the node, apart from the scalar reading, is the random column vector ϕ_j. Therefore CS compression is performed by each node in two steps.

1. Each of the N sensors locally computes the M elements of the random vector $\phi_j = \{\phi_{ij}\}_{i=1}^{M}$ using its own address (or any other unique number known also by decoder) as seed for the pseudo-random sequence.

2. The j-th sensor multiplies its sensor reading for the vector ϕ_j just computed obtaining a new $M \times 1$ vector. Then the sensor waits until it receives by each of its children nodes the compressed vector. Once received, it simply sums all the vectors together and sends the newly computed vector to its own parent.

The generation of the random vector using the node ID is a requirement since during the reconstruction phase the decoder must know the whole Φ matrix for the resolution of the optimization problem in Equation (3.5). In this way the decoder is able to reconstruct the matrix for the reconstruction process from node ID (or address).

An easy visual explanation of the two approaches is in Figure 2.16.

According to Equation (3.12) the number of packets sent by a node performing CS is:

$$P_k^{CS} = \lceil (M \cdot B_{data})/B_{payload} \rceil \tag{3.13}$$

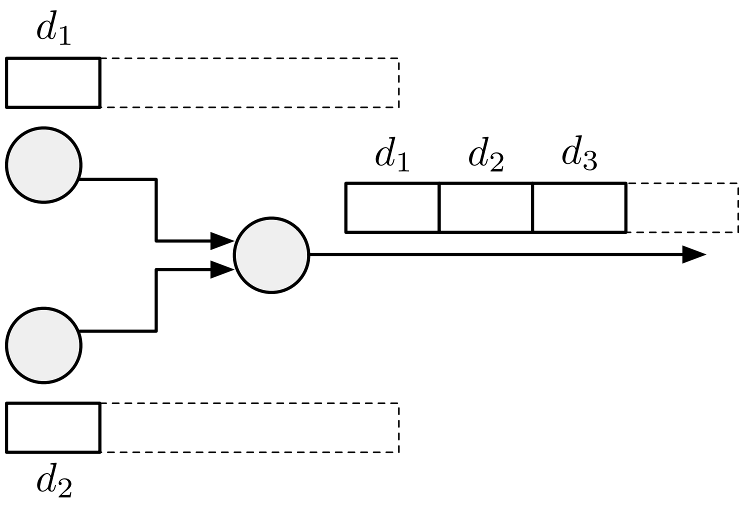
Differently from PF the number of outbound packets is not function of a geographic parameter as N_{tch} in Equation (3.9). In this case each node in the network sends the same number of packets, independently from the position in the directed routing path from the node to the sink.

Nevertheless P_k^{CS} is function of M, which is depending on the network size N as in Equation (3.7). Therefore even though each node sends the same number of packets, this number is function, for the whole network, of the number of deployed nodes. The major drawback of this technique is that a slightly increasing in the value of M determines a large increment in data circulating within network.

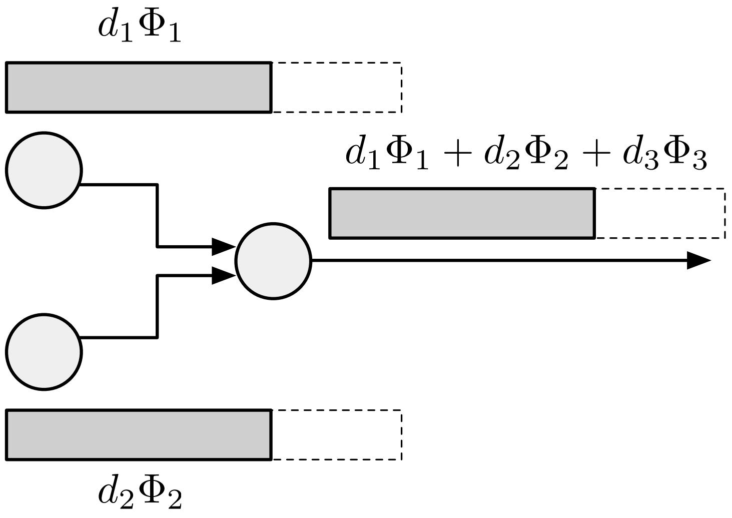
3.4.2.3 Mixed algorithm: between PF and CS

Event though as seen in previous sections both the techniques present some critical limitations, it is possible to overcome these limitations with a mixed algorithm to achieve a better result in terms of lifetime and energy saving.

While the performance of PF scheme, in terms of P_k^{PF}, is strictly related to parameter N_{tch}, that is bigger for nodes proximal to the sink, on the contrary,



(a) PF: data incoming in each node are packed using data aggregation



(b) CS: the outgoing data size is independent from the number of child nodes

Figure 3.10: PF and CS data aggregation techniques. Dashed rectangle is the IEEE 802.15.4 packet

performance for CS is only connected with parameter M that is proportional to network size. Therefore, neither PF nor CS are suitable schemes to adopt in every condition and network topology.

One can assume that each node has enough information to take a proper decision with the goal of reducing the outgoing packets. Thus a generic node can autonomously choose to adopt PF or CS to save battery power.

In formula we have that the number of packets sent by a generic k node is:

$$P_k = \begin{cases} P_k^{CS}, & \text{if } P_k^{CS} \leq P_k^{PF} \\ P_k^{PF}, & \text{if } P_k^{CS} > P_k^{PF} \end{cases} \tag{3.14}$$

Where P_k is the number of packets actually sent by the node. Each node can independently take a decision about what kind of gathering scheme, between the two illustrated, to adopt aiming to reduce the number of transmitted packets.

Thus the general algorithm is composed by three steps:

1. If the node receives from one children node a packet compressed with CS, it must compress data using the same scheme, converting data received by the other children nodes and aggregate data as described in Section 3.4.2.2
2. If all data received is aggregated using PF, the node chooses to adopt PF or DCS aiming to send the lowest number of packets evaluating Equation (3.14).
3. If it is not possible to take a decision since the number of outgoing packets is the same, CS is performed.

See the pseudo-code of the Algorithm (1) for further details.

It is important to stress that when a node in the networks decides to compress data using CS (step 2), the node has to compress incoming data coming from all the other nodes. This means that the node has to compress each scalar reading x_j, not yet compressed, coming from the j-th node. As seen before the compression goes through a vector multiplication as expressed in Equation (3.12). This implies that the node has to know the measurement vector ϕ_j for each node j and perform $N_{tch} \times M$ multiplications to compress data. This is quite expensive for the node if the vector M is large, for this reason the energy spent in compression is not always negligible

Algorithm 1 Selection of data gathering scheme

```
for $i = 1$ to number_children_nodes() do
    pkt($i$) = receive_pkt_from_child_node()
end for
for $i = 1$ to number_children_nodes() do
    if is_compressed_CS( pkt($i$) ) then
        compress_data_CS_and_send(pkt)
        return
    end if
end for
if pkt_CS(pkt) $\leq$ pkt_PF(pkt)  then
    compress_data_CS_and_send(pkt)
else
    compress_data_PF_and_send(pkt)
end if
```

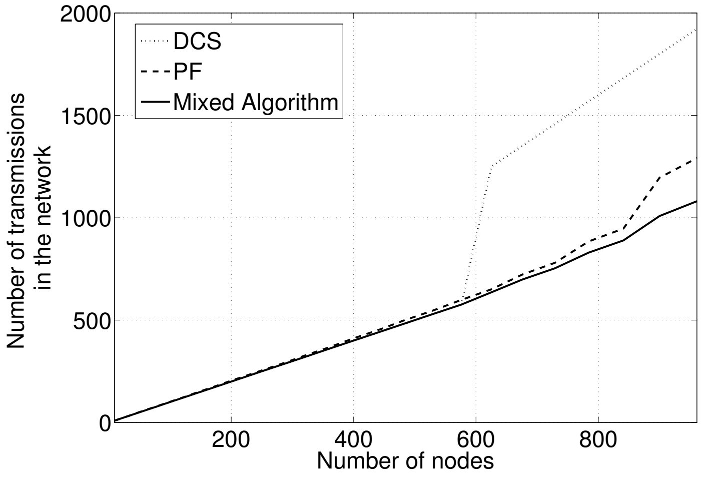
3.4.3 Mixed algorithm simulation results

To evaluate the behavior of the network and the performance of the solution, a framework in MATLAB has been developed. In the preliminary tests, parameters are set to $B_{ID} = 8$ bytes and $B_{data} = 1$ byte. The former is used because it is possible to use the EUI64 address, composed by 64 bits, to identify the node and because EUI64 is unambiguous for each node in the network.

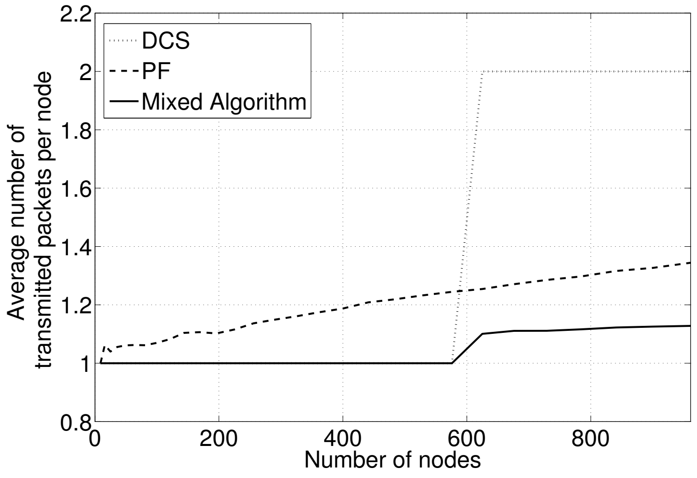
In Figure 3.11a, the simulation result is shown when network dimension varies from 9 to 961 nodes. For the PF technique the plot shows a slightly increasing trend with the network size. This behavior is due to the parameter N_{tch} as seen in Equation (3.9).

More interesting is the behavior for CS. For a particular network size the number of sent packets explodes, presenting a large increment. For this specific network size, indicated from now on with N_{crit}, the value for P_k^{CS} increases from 1 to 2, since $M(N) \cdot B_{data} > B_{payload}$. This increment regards each single node within the network that has to send out two packets instead of one. This is reported clearly in Figure 3.11b where the average number of transmissions is plotted: for $N = N_{crit}$ the number of transmissions doubles, whereas for PF the increasing is much more moderated.

The mixed algorithm performs better than the two previous schemes, presenting a number of sent packets that is always smaller than both PF and CS.



(a) Total number of packets circulating within the network



(b) Average number of transmissions per node

Figure 3.11: Comparison among PF, DCS, and mixed algorithm

For small-sized networks the proposed solution approaches CS. This is why the number of packets sent with PF or CS is the same. Therefore according to the algorithm proposed, the node compresses data using CS. The interesting part of the plot is for values of $N > N_{crit}$ where a mixed behavior arises. Since P_k^{PF} depends on N_{tch} for big networks this value becomes considerably large especially for nodes proximal to the central sink, whereas it is smaller for distal nodes. For those nodes with high N_{tch} it becomes $P_k^{CS} \leq P_k^{PF}$ and the algorithm switches to using CS. This creates an inner zone in the network where CS is performed, leaving the outside nodes gathering data through classical PF, as represented in Figure 3.12. The nodes with the higher workload in this case are those ones on the frontier since they are responsible for data compression using CS.

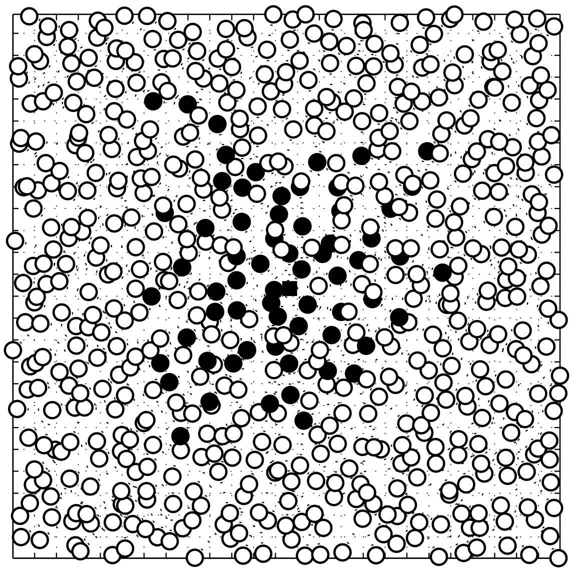


Figure 3.12: For network with size $N > N_{crit}$ it is clearly visible the inner zone performing CS (black circles) and the external one composed by nodes performing PF (white circles)

The main interesting parameters in WSNs are lifetime and power consumption. Lifetime, in this context, is intended as the time until the death of the first node in the network when a $500mAh$ battery is used. To evaluate the power consumption the hardware taken as reference for current and timing measurements is an Ember EM250 node. It is a ZigBee System-on-chip combining a 2.4GHz IEEE 802.15.4 radio transceiver with a 12MHz 16-bit processor having 128Kb of flash memory and 5Kb of RAM.

The parameters for simulations regarding power consumption are measured on the wireless node. In particular we have a current consumption of $36mA$ for transmitting and receiving data at $3.3V$, while the time needed to send or receive a single packet is measured to be around $5ms$. Apart from the time spent in packets transmission and reception, nodes are in sleep mode: a low power state in which they consume just $1.5\mu A$. The node wakes up, eventually compresses and transmits data just for the time needed. Then it turns into sleep mode as soon as it finishes to forward data to the parent.

The power consumption for compression has been measured using a precision resistor and evaluating the current absorption during CS. Moreover we have connected an oscilloscope to an unused pin on the micro-controller pin for measuring the time spent in compression. We drive a pin immediately before calling the compression algorithm, and release the pin immediately upon returning from the algorithm. During the tests the absorbed current has been of $9.3mA$ whereas the elaboration time is almost proportional to node providing data to compress (N_{tch}).

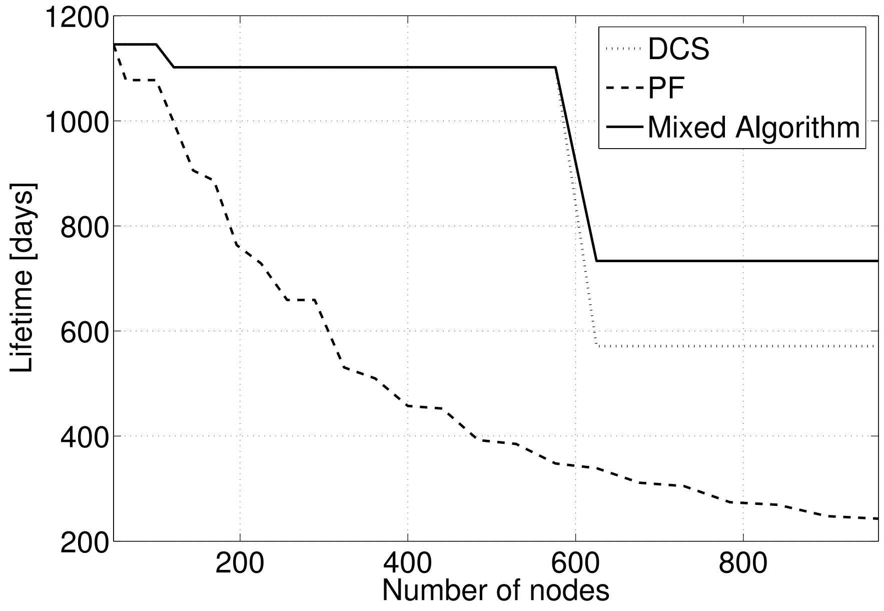


Figure 3.13: Network lifetime using Ember EM250 nodes (B_{ID} = 8 bytes B_{data} = 1 byte)

In Figure 3.13 the simulation result is reported. The lifetime with PF as expected is always decreasing. When in the network CS is used, a decrement in lifetime is visible for $N = N_{crit}$ due to the doubling of P_k^{CS}. The interesting trend in the graph is that one related to the mixed algorithm. It is possible to identify two different parts in the plot: (1) for small networks with $N < N_{crit}$ the algorithm is coincident with CS as also seen in Section 3.4.2.3 while (2) for $N \sim N_{crit}$ we have an abrupt decrement in lifetime.

For this network size every node in the network, except few nodes proximal to the sink, changes its own compression scheme from CS to PF. This mixed behavior implies a greater lifetime for the whole network, gaining a 35% increase in lifetime compared to classical CS scheme.

As seen in Section 3.4.2 simulations depend on two parameters: B_{ID} and B_{data}.

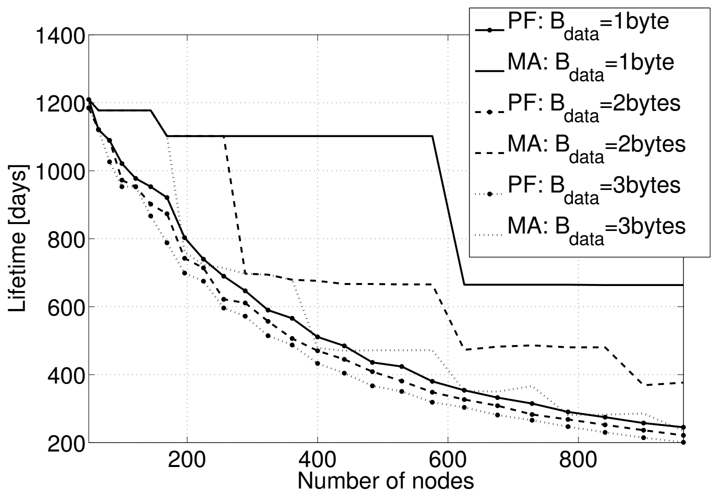
B_{data} results are the most critical. Both PF and CS depend on this parameter, but doubling the value has a deep impact mainly on CS since each nodes within the network doubles the number of sent packets, whereas PF is only slightly affected since B_{data} is an additional contribution to the sum in Equation (3.9). The result of simulations when B_{data} is doubled is reported in Figure 3.14a.

Even if the influence of the parameter B_{data} mainly affects the performance of the mixed algorithm, the lifetime is always better for our algorithm than classical PF scheme.

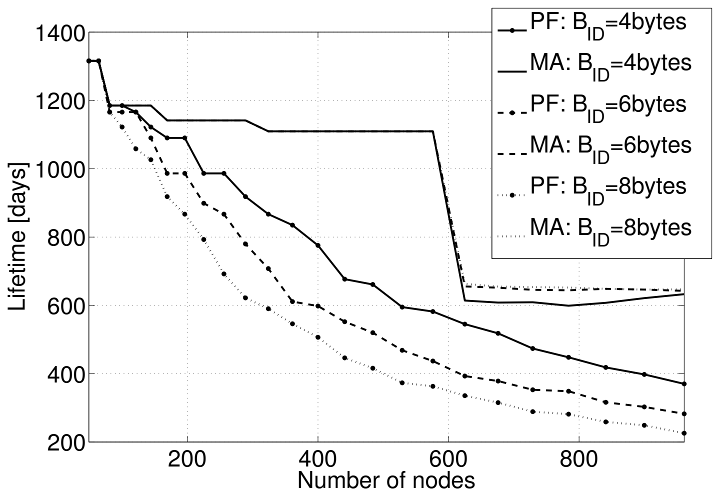
In Figure 3.14b the simulation when B_{ID} is varied from 2 bytes to 8 bytes is reported. On the contrary in this case the increasing in the dimension of the ID field of the ZigBee packet affects much more PF than the mixed algorithm. The increase in the number of bytes used for ID has the main effect to enlarge the central region for DCS since PF performs poorly when it has to manage too many bytes per node that is what happens within the network near the sink.

3.4.4 Energy consumption optimization

Although simulations performed in the previous section take into account the energy spent in compression, the real contribution of the power consumption for elaboration is strictly related to the radius of the central region of the network performing compression, as seen in Figure 3.12. Closer the inner zone is to



(a) Varying B_{ID}



(b) Varying B_{data}

Figure 3.14: (a) Lifetime of the network: comparison between PF and Mixed Algorithm (MA)

the sink, more the nodes on the boundary have data to compress coming from external nodes that forward data using PF. Using the notation in Equation (3.9), this means that bigger is N_{tch} for a boundary node, higher will be the energy consumption for data compression. We can then write $E_{comp}(N_{tch})$ where E_{comp} is the energy spent in compression.

Moreover according to Equation (3.14) a boundary node is the node in which the equality:

$$P_k^{CS} = P_k^{PF} = P_k \tag{3.15}$$

is first realized in the network when $N > N_{crit}$.

To evaluate E_{comp} for boundary nodes, in equation (3.15) we can consider M and B_{data} as constants: the former is related to the network size and the latter is function of the sensor data encoding. Then neglecting the ceiling operator the equality becomes:

$$N_{tch} = \frac{M \cdot B_{data}}{B_{ID} + B_{data}} - 1 \tag{3.16}$$

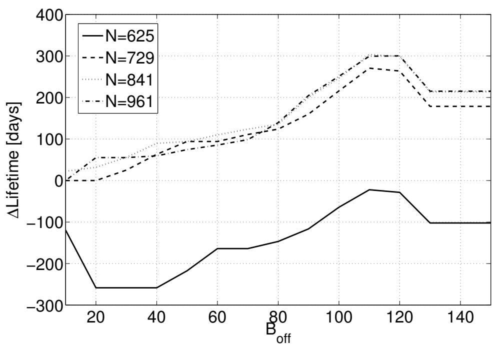
that is N_{tch} is inversely proportional to B_{ID}. A relation between E_{comp} and B_{ID} does exist in the network. Fewer bytes are used to represent the location field in the payload, higher will be the power consumption and the energy involved in data compression for boundary nodes.

Since the problem of the compression regards only nodes on the boundary of the central region, the optimization problem of the algorithm focuses on network sizes greater than N_{crit}. For a number of nodes $N < N_{crit}$ there is no central region performing CS since each node compress each own data using CS.

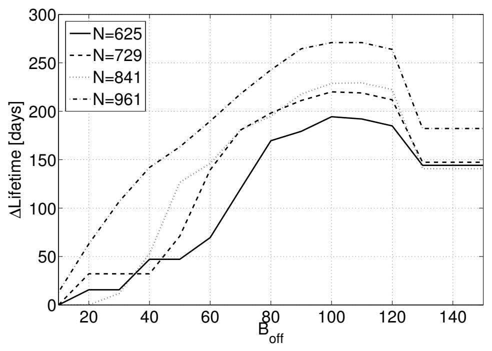
In Figure 3.15 the contribution of the compression energy is presented versus the number of nodes N, using B_{ID} as parameter. The plot is related to the first dying boundary node in the network.

The figure shows how the compression energy becomes predominant when reducing B_{ID} and the corresponding increasing in compression work.

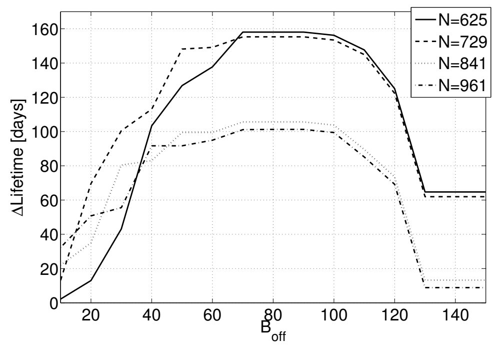
The plot presents decreasing values for B_{ID}. In particular $B_{ID} = log_2(N)$ bytes is the case in which the minimum number of bytes is used to represent the nodes in the network. For $B_{ID} = 0$ bytes the mapping between data and node position is obtained through data ordering within the packet payload. Both nodes and sink have to know the network configuration and the routing path supposed static. Since there is a shared knowledge between sink and nodes, the



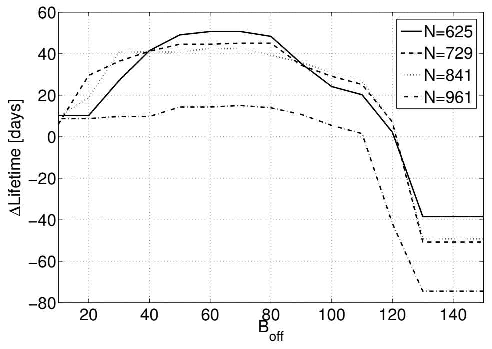
(a) $B_{ID} = 0$ bytes



(b) $B_{ID} = log_2(N)$ bytes



(c) $B_{ID} = 2$ bytes

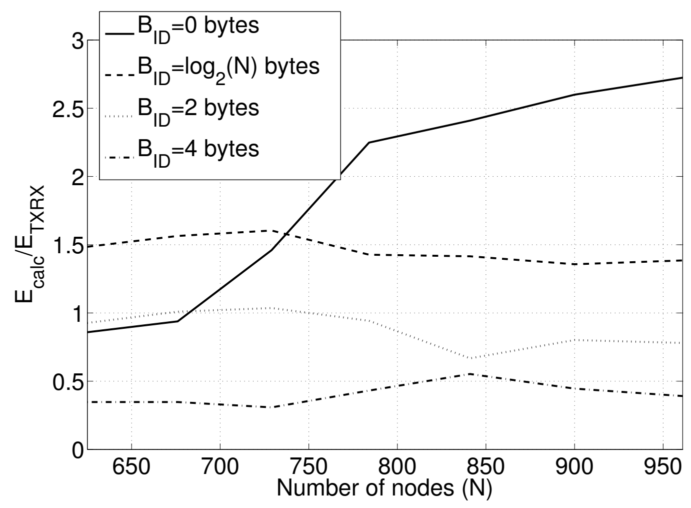


(d) $B_{ID} = 4$ bytes

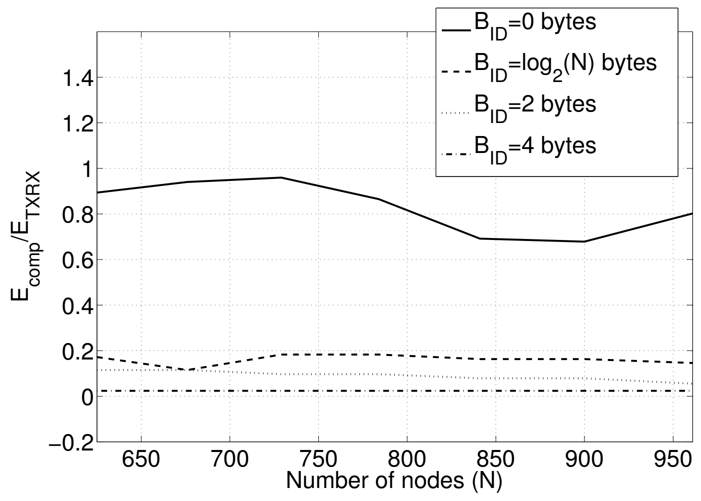
Figure 3.15: Lifetime extension of the modified algorithm with respect to the standard mixed algorithm vs. B_{off} using network size N as parameter

use of B_{ID} becomes redundant. This mechanism is based on the assumption that each node assigns a different priority value to its own children nodes where this priority defines the position inside the forwarded packet. When a node receives data coming from its children, it orders data according to the priority value of the source node. The knowledge of the priority values and the static routing path, is enough to associate sensor value and node based on data offset inside the payload.

In the trend reported in Figure 3.16a it is possible to see how with decreasing in the size of B_{ID} the power consumption for compression greatly increases, becoming the major contribution in energy consumption. The slightly increasing trend for the $B_{ID} = 0$ curve is due to the collapse of the central region in very



(a) Mixed Algorithm



(b) Modified algorithm with $B_{off} = 90$ bytes

Figure 3.16: Ratio between the energy spent in compression E_{comp} and energy for data transmission and reception E_{TXRX} for the first dead node in the network vs. network size ($B_{data} = 1$ byte)

few nodes around the sink that becomes part of the boundary.

The mixed algorithm aims to the minimization of the number of packets transmitted by each node to save energy. If the compression scheme is not energy-aware it could represent the main source of energy consumption. To take into account the energy expenditure for compression we introduce a modification to the original algorithm. The idea behind the modification is to reduce the contribution of E_{comp} with respect to E_{TXRX}. As we have seen before E_{comp} is function of the value N_{tch}, that is the number of nodes doing relay on the boundary node responsible for data compression. Since in general N_{tch} increases the closer we get to the sink, the main idea is to shift the boundary toward the leaf nodes, making bigger the inner zone of the network performing CS and reducing the value of N_{tch}.

As we said before the boundary is characterized by those nodes for which is verified the equality (3.15), therefore if we want to reduce the value of N_{tch} to reduce the power consumption, we have to modify Equation (3.9) so that the equality continues to be verified.

Therefore in our new algorithm Equation (3.14) is modified in:

$$P_k = \begin{cases} P_k^{CS}, & \text{if } P_k^{CS} \leq \hat{P}_k^{PF} \\ P_k^{PF}, & \text{if } P_k^{CS} > \hat{P}_k^{PF} \end{cases} \tag{3.17}$$

where:

$$\hat{P}_k^{PF} = \lceil ((N_{tch}+1)\cdot(B_{ID}+B_{data})+B_{off})/B_{payload} \rceil \tag{3.18}$$

That is a slightly modified version of the equation (3.9) in which a term B_{off} has been added. This is a dummy contribution used in the comparison to take a decision about the mechanism to adopt. Accordingly the new boundary condition becomes:

$$P_k^{CS} = \hat{P}_k^{PF} = P_k \tag{3.19}$$

that is verified for smaller values of N_{tch} shifting the boundary toward more external nodes.

The practical implementation of the solution on the nodes is done using algorithm 2 reported using pseudo-code.

Differently from the implementation of the mixed algorithm seen in Section

Algorithm 2 Selection of gathering scheme with modified mixed algorithm

```
Require: offset $\geq 0$
  for $i = 1$ to number_children_nodes() do
    pkt($i$) = receive_pkt_from_child_node()
  end for
  for $i = 1$ to number_children_nodes() do
    if is_compressed_CS( pkt($i$) ) then
      compress_data_CS_and_send(pkt)
      return
    end if
  end for
  if pkt_CS_plus_offset(pkt,offset) $\leq$ pkt_PF(pkt)  then
    compress_data_CS_and_send(pkt)
  else
    compress_data_PF_and_send(pkt)
  end if
```

3.4.2.3 that minimizes the number of packet transmissions, the modified version increases the number of transmissions in the network trying to obtain a better overall lifetime of the system reducing the energy used in compression by boundary nodes.

In Figure 3.15 the simulation results are reported showing the increasing in lifetime of the modified algorithm with respect to the classical implementation of the mixed algorithm when different values for B_{off} are used in the simulation using the network size N as parameter in the plots.

In each plot it is possible to identify a general behavior for the network with increasing value of B_{off}. (1) For small increasing values of B_{off} the lifetime increases since the contribution of the energy spent in compression decreases although in the network expands the inner portion of nodes performing DCS, reducing the contribution of E_{comp} for each boundary node. (2) For a certain value of B_{off} ranging from 80 to 100 bytes, the lifetime is maximized. For this value a perfect trade-off between power consumed for transmission and for elaboration is achieved. (3) After the peak in lifetime the power for packet sending and receiving increases much more than the power saving for compression, resulting in a decreasing lifetime. (4) When $B_{off} \geq B_{payload} = 127$ bytes the relation $P_k^{CS} \leq \hat{P}_k^{PF}$ in Equation (3.17) is never verified since the value for $\hat{P}_k^{PF}$ in Eq

(3.18) is always greater than P_k^{CS} because of the ceiling operator, resulting in the whole network performing CS. For small network sizes and $B_{ID} = 0$ bytes we can see again the atypical behavior already highlighted in Figure 3.16 where the collapsing of the inner zone on the sink, creates a critical situation in which the modified algorithm poorly performs.

In general an optimal value for the dummy B_{off} does exist and it minimizes the power spent in compression at the expenses of number of transmissions. Moreover the lifetime increasing strictly depends on the value of B_{ID}. In Figure 3.16b we report the results of simulations when an optimal value of 90 bytes is used for B_{off} evaluating the ratio between the energy spent in compression and energy used for data transmission. Differently from Figure. 3.16a the ratio is almost under 1 for every network size, meaning that the contribution of compression in energy consumption is much more well-balanced than before.

Chapter 4

Compressive Sensing for signal ensembles

The theory developed for data aggregation in the previous chapter has been developed with the goal to capture the spatial information spread in WSN. Nevertheless many applications involve multiple signals varying also in time, for which the classical CS cannot be applied without modifications.

More specifically we can consider a generic WSN in which a large number of distributed sensor nodes observe the same phenomena. The ensemble of signals they acquire can be expected to posses some joint structure, or *inter-signal correlation*, in addition to *intra-signal correlation* in each individual sensor's measurements.

For example, we can imagine a WSN composed by sensor nodes sensing temperature or humidity at several points in space. The time-series acquired at a given sensor might have considerable intra-signal (temporal) correlation and might be sparsely represented in a local Fourier basis. In addition, the ensemble of time-series acquired at all sensors might have considerable inter-signal (spatial) correlation, since all sensors gather data from to the same environment. In such settings, distributed source coding that exploits both intra- and inter-signal correlations might allow the network to save on the communication costs involved in exporting the ensemble of signals to the collection point [Slepian and Wolf, 1973].

A number of distributed coding algorithms have been developed that involve collaboration amongst the sensors, including several based on predictive coding [Cristescu et al., 2004], a distributed KLT [Gastpar et al., 2002], and distributed wavelet transforms [Wagner et al., 2006]. Three-dimensional wavelets have been proposed to exploit both inter- and intra-signal correlations [Ganesan et al., 2003]. Note, however, that any collaboration involves some amount of inter-sensor communication overhead.

4.1 Techniques for signal ensemble compression and reconstruction

The two most prominent frameworks dealing with sparsity and compressibility of multidimensional signals and signal ensembles are Distributed Compressive Sensing (DCS) [Chen et al., 2011] and Kronecker Compressive Sensing (KCS) [Duarte and Baraniuk, 2012].

4.1.1 Distributed Compressed Sensing (DCS)

DCS is a distributed algorithm that exploit both intra- and inter-signal correlation structures to improve objects reconstruction.

In a typical DCS scenario a certain number of sensors measure signals that are each individually sparse in some basis but also with correlation from sensor to sensor. While each sensor independently encodes its own signal with classical CS theory and transmits just a few of the resulting coefficients to a single collection point, the reconstruction is performed at the sink that can reconstruct each of the signals precisely.

Differently from the usual compression performed using CS, DCS theory rests on the concept of *joint sparsity* of a signal ensemble. The joint sparsity is often smaller than the aggregate over individual signal sparsities. Therefore, DCS offers a reduction in the number of measurements.

Unlike the single-signal definition of sparsity, however, there are numerous plausible ways in which joint sparsity could be defined.

In total we can consider three different *joint sparsity models* (JSMs) each suitable for a different situation. If we call them JSM-1, JSM-2 and JSM-3 we can say that in the first two models each signal is itself sparse and we use DCS to exploit a *joint representation* for the ensemble to use few measurements and a better recovery. In JSM-3, on the contrary, no signal is itself sparse but the existing joint sparsity among signals allows recovery using DCS.

If the considered network is composed by J nodes, each sensing a different signal, we can denote each signal as x_j with $j \in \{1, 2, \dots, J\}$ and, if each signal is composed by N samples, $x_j \in \mathbb{R}^N$. With $x_j(n)$ we indicate the n sample in time of the node (signal) j. We also assume that there exist a sparse basis $\Psi \in \mathbb{R}^{N \times N}$ in which the x_j can be sparsely represented.

We define with Φ_j the measurement matrix for the node (signal) j, $\Phi_j \in \mathbb{R}^{M_j \times N}$ and it is different for each node.

According to these definition we can rewrite Equation (3.4) as $y_j = \Phi_j x_j$ where y_j is the $M_j \times N$ vector of the incoherent measurements of x_j.

As seen in Section 3.3.2 Φ_j is special random matrix (Gaussian, Bernoulli, etc...).

4.1.1.1 JSM-1

In this model each signal x_j is composed by a common sparse component shared by all the signals and a sparse innovation component specific for each node. In formula:

$$x_j = z_C + z_j, \qquad j \in \{1, 2, \dots, J\} \tag{4.1}$$

where:

$$z_C = \Psi\theta_C,\ \|\theta_C\|_0 = K_C \qquad \text{and} \qquad z_j = \Psi\theta_j,\ \|\theta_j\|_0 = K_j \tag{4.2}$$

In the formulas the signal z_C is common to all of the signal x_j with sparsity K_C while the innovation part z_j is different for each node that has a different K_j sparsity.

Signals characterized by a sparse common component and innovation components (JSM-1) are considered to be typical of large-scale WSNs, when global phenomena (sun, wind, temperature, etc...) affect all sensors while local phenomena (shade, water, human presence, etc...) affect individual sensors.

As in the general case the resolution of JSM-1 based problems goes through the resolution of the minimization problem in Equation (3.5) with small variations.

For simplicity we can consider the case of $J = 2$ and sparsity basis $\Psi = I_N$. We can formulate the recovery problem using matrices and vectors as:

$$z = \begin{bmatrix} z_C \\ z_1 \\ z_2 \end{bmatrix}, \; x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \; y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$\Phi = \begin{bmatrix} \Phi_1 & 0 \\ 0 & \Phi_2 \end{bmatrix}, \; \tilde{\Psi} = \begin{bmatrix} \Psi & \Psi & 0 \\ \Psi & 0 & \Psi \end{bmatrix}$$

Using the frame $\tilde{\Psi}$ we can represent the data vector x sparsely using the coefficient vector z which contains $K_C + K_1 + K_2$ nonzero coefficients, to obtain $x = \tilde{\Psi} z$. The measurement vector$y$ is computed from separate measurements of the signal x_j using the measurement basis Φ.

We can then recover a vector $\hat{z}$ which is a viable representation for x by solving:

$$\hat{z} = \arg\min \|z\|_1 \qquad \text{s.t.} \qquad y = \Phi\tilde{\Psi} z \tag{4.3}$$

The vector z enables the reconstruction of the original signal x_1 an x_2.

Alternatively we can use a γ-weighted ℓ_1 formulation where we try to minimize the modified ℓ_1 metric:

$$\gamma_C \|z_C\|_1 + \gamma_1 \|z_1\|_1 + \gamma_2 \|z_2\|_1 \tag{4.4}$$

where $\gamma_C, \gamma_1, \gamma_2 \geq 0$.

4.1.1.2 JSM-2

In this modes all signals are constructed from the same sparse set of basis vector but with different coefficients. In formula:

$$x_j = \Psi\theta_j, \qquad j \in \{1, 2, \dots, J\} \tag{4.5}$$

where each θ_j is nonzero only on the common coefficient set $\Omega \subset \{1, 2, \dots, N\}$ with $|\Omega| = K$.

The JSM-2 sparsity model is directly applicable when all the sensors acquire a replica of the same sparse signal but with phase shifts and attenuations caused by signal propagation. This is the case when sensors are deployed in the same environment and they acquire the same signal but with slight differences due to positioning.

Differently from JSM-1 the sparse approximation can be recovered via greedy algorithms such as the Simultaneous Orthogonal Matching Pursuit (SOMP) [Aravind et al., 2011] or the DCS-SOMP algorithm [Baron et al., 2005].

It is possible to demonstrate that DCS-SOMP succeeds in optimal reconstruction with $\hat{c} < c(S)$.

4.1.1.3 JSM-3

This model extends JSM-1 but the common component is no longer sparse in any basis. That is:

$$x_j = z_C + z_j, \qquad j \in \{1, 2, \dots, J\} \tag{4.6}$$

with:

$$z_C = \Psi\theta_C \qquad \text{and} \qquad z_j = \Psi\theta_j,\ \|\theta_j\|_0 = K_j \tag{4.7}$$

but Z_C is not necessarily sparse in the basis Ψ.

A practical situation well-modeled by JSM-3 is where several sources are recorded by different sensors together with a background signal that is not sparse in any basis.

The algorithm used for signal recovering in JSM-3 is the ACIE algorithm based on Orthogonal Matching Pursuit (OMP) [Aravind et al., 2011].

4.1.2 Kronecker Compressive Sensing (KCS)

Data sensed in a WSN by several nodes spread in the environment can be seen as a multi-dimensional signal with at least two dimensions: temporal and spatial dimension.

Kronecker product matrices are a natural way to generate sparsifying and measurement matrices for CS of multidimensional signals. Kronecker product sparsity bases combine the structures encoded by the sparsity bases for each signal dimension into a single matrix. Kronecker product measurement matrices for multidimensional signals can be implemented by performing a sequence of separate multiplexing operations on each dimension.

We can use Kronecker product matrices as sparsifying bases for multidimensional signals to jointly model the signal structure along each one of its dimensions.

Mathematically the Kronecker product of two matrices A and B of sizes $P \times Q$ and $R \times S$ respectively is defined as:

$$A \otimes B := \begin{bmatrix} A(1,1)B & A(1,2)B & \dots & A(1,Q)B \\ A(2,1)B & A(2,2)B & \dots & A(2,Q)B \\ \vdots & \vdots & \ddots & \vdots \\ A(P,1)B & A(P,2)B & \dots & A(P,Q)B \end{bmatrix} \tag{4.8}$$

Thus $A \otimes B$ is a matrix of size $PR \times QS$.

4.1.2.1 KCS for distributed sensing in WSN

In distributed sensing problem we aim to acquire an ensemble of signals $x_1, \dots, x_j \in \mathbb{R}^N$ that vary in time, space, etc... As also seen for DCS we assume that each signal's structure can be encoded using sparsity with an appropriate basis Ψ.

This ensemble can be expressed and an $N \times J$ matrix $X = [x_1 x_2 \dots x_J] = [x^{1^T} x^{2^T} \dots x^{N^T}]^T$ where the individual signals $x_1, \dots, x_J$ corresponding to columns of the matrix and where the rows $x^1, \dots, x^N$ of the matrix correspond to different snapshots of the signal ensembles at different values of time, space, etc... The structure of each signal is observable on each of the columns of the matrix, while the structure of each snapshot (spanning all the signals) is present on each of the rows of the matrix X.

If the multidimensional signal presents different sparsity properties along each of its dimension it is possible to obtain a single sparsity and measurement basis for the entire multidimensional signal to jointly compress and recover the original components (dimensions) of the signal.

More formally we can denote the individual signal as $x_j \in \mathbb{R}^N, 1 \leq j \leq J$ and the individual snapshots as $x^n \in \mathbb{R}^J, 1 \leq n \leq N$. The multidimensional signal X is then in $\mathbb{R}^N \otimes \mathbb{R}^J \cong \mathbb{R}^{NJ}$ where the columns are the signal gathered by the j-th node and the rows correspond to a snapshot in time shared among all the nodes.

We assume that the snapshots x^n are sparse or compressible in a basis Ψ and that the signals x_j are sparse or compressible in Ψ'. We then can obtain a sparsity basis for X obtained from Kronecker products as $\bar{\Psi} = \Psi \otimes \Psi' = \{\psi \otimes \psi', \psi \in \Psi, \psi' \in \Psi'\}$, and obtain a coefficient vector Θ for the signal ensemble so that $\tilde{X} = \bar{\Psi}\Theta$ where $\tilde{X} = [x_1^T x_2^T \dots x_J^T]^T$ is a vector-reshaped representation of X.

The same matrix construction method is feasible for measurement matrices that are formulated as Kronecker products. Such matrices correspond to measurement process that operate first on each individual signals / snapshots, followed by operations on the measurements obtained for the different signals / snapshots respectively. The resulting measurement matrix can be expressed as $\tilde{\Phi} = \Phi \otimes \Phi'$ with $\Phi \in \mathbb{R}^{M_1 \times N}$, $\Phi' \in \mathbb{R}^{M2 \times J}$ and $\tilde{\Phi} \in \mathbb{R}^{M_1 M_2 \times NJ}$. This results in a matrix that provides $M = M_1 M_2$ measurements of the multidimensional signal X.

In case of a WSN each node obtains independent measurements $y_i = \Phi_j x_j$ with an individual measurement matrix being applied to each signal. In practice the measurement ensemble can be written as:

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_J \end{bmatrix} \qquad \text{and} \qquad \tilde{\Phi} = \begin{bmatrix} \Phi_1 & 0 & \dots & 0 \\ 0 & \Phi_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \Phi_J \end{bmatrix} \tag{4.9}$$

where 0 is a matrix of appropriate size with all entries equal to 0. We then have $Y = \tilde{\Phi}\tilde{X}$. If a matrix $\Phi_J = \Phi'$ is used at each sensor to obtain its individual measurements, then we can express the joint measurement matrix the matrix $\tilde{\Phi} = I_J \otimes \Phi'$ where I_J denotes the $J \times J$ identity matrix

4.2 A comparison between KCS and DCS

In this section we first compare the two frameworks and further investigate how, in a real deployment, the compressed sensing techniques for signal ensembles can be used to reduce the power consumption and prolong lifetime.

4.2.1 Compressibility of signal ensembles

We discuss and compare the different techniques proposed in previous sections against a common set of artificial signals properly built to embody the main characteristics of natural signals.

Signals characterized by a sparse common component and different innovation components (JSM-1) are considered to be typical of large-scale WSNs, when global phenomena (sun, wind, temperature, etc...) affect all sensors while local phenomena (shade, water, human presence, etc...) affect individual sensors.

To recreate this situation in our first simulation setup we consider $J = 16$ nodes sampling a fictitious signal and gathering $N = 50$ samples. In our first simulation each signal presents a common component that is sparse in a DCT basis with sparsity $K_C = 10$ representing the common temperature field and an innovation component supposed sparse in a identity basis $\mathbf{I}_N$ with sparsity $K = 2$ representing abnormal sporadic sensor readings. We let the number of measurements M_j varying from 0 to N and for each value of M_j we perform

100 iterations by generating the sparse signals and using measurement matrices with i.i.d. Gaussian entries to compress the readings. Then we measure the probability of successful recovery for each value of M_j where a success is declared if the reconstructed signal $\hat{\mathbf{x}}$ obeys $\epsilon = \|\mathbf{x} - \hat{\mathbf{x}}\|_2 / \|\mathbf{x}\|_2 \leq 10^{-2}$.

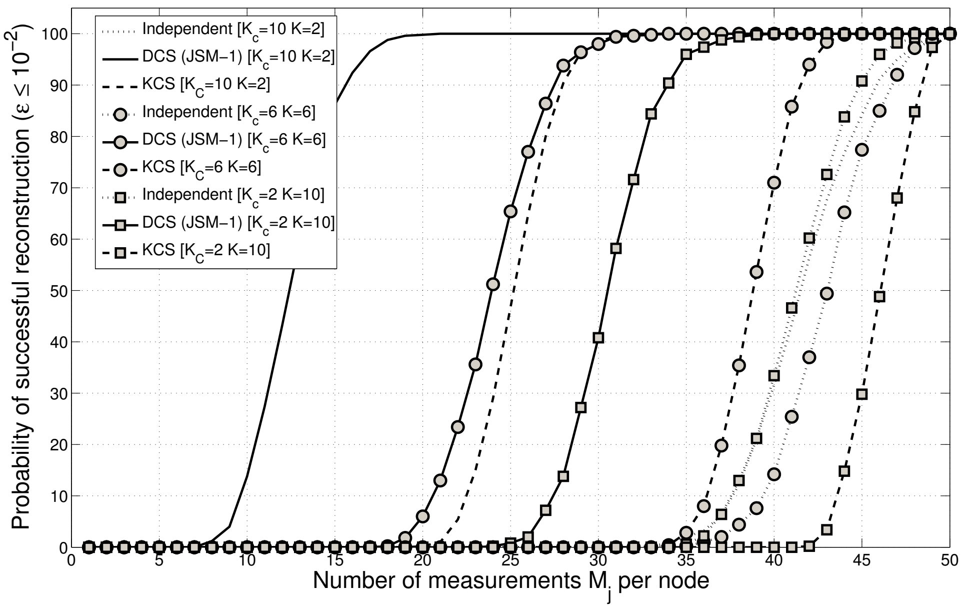


Figure 4.1: Joint reconstruction quality comparison among DCS (JSM-1), KCS and separate decoding. The considered $J = 16$ signals of length $N = 50$ have common sparsity K_C and innovation with sparsity K

In Figure 4.1 the simulation results are shown. JSM-1 reconstruction is performed using YALL1 Matlab® package, jointly solving the basis pursuit problem. In the independent reconstruction we try to reconstruct the data, recovering each signal independently and exploiting an over-complete representation basis [Donoho et al., 2006] $\mathbf{\Lambda} = [\mathbf{\Psi}\ \mathbf{I}]$ as sparsifying basis where $\mathbf{\Psi}$ is the DCT matrix and $\mathbf{I}$ is the identity matrix. KCS problem is solved using the SparseLab Matlab® package with a sparsifying matrix $\mathbf{\Delta} = \mathbf{\Pi} \otimes \mathbf{\Lambda}$ where $\mathbf{\Pi}$ is the matrix of the *Horz-diff* transformation as proposed in [Quer et al., 2009] to exploit the spatial correlation of the signals.

The results show how the JSM-1 guarantees a perfect reconstruction with only 20 measurements per node with respect to KCS and independent recovery.

The independent recovery in particular is inefficient since it does not take advantage of the signals correlation during recovery. In this case we need almost 50 measurements for every node to be able to reconstruct the signal ensembles. For the KCS we have an intermediate result because the sparsifying matrix is good enough to take advantage of the signals correlation in the reconstruction phase.

To evaluate also how DCS, KCS and independent reconstruction perform when the signals present more innovation with respect to common sparsity components, in the simulations we have tried to change the parameters K_C and K and conduct again the reconstruction tests. Results are summarized in Figure 4.1 and confirm that DCS performs better with respect to KCS and independent reconstruction even when the signals present more innovation components. The figure shows also how the general performance of DCS and KCS decreases, this is because when reducing the prevalence of common components among signals both frameworks are not able to exploit the inter-signals correlation to improve reconstruction quality. Independent reconstruction, on the other hand, is not affected by signal ensembles characteristics and maintains the same reconstruction performance regardless of the ratio between K_C and K.

The JSM-2 sparsity model is directly applicable when all the sensors acquire a replica of the same sparse signal but with phase shifts and attenuations caused by signal propagation. This is the case when sensors are deployed in the same environment and they acquire the same signal but with slight differences due to positioning.

To compare DCS and KCS against this type of signals in the second simulation we use $J = 16$ nodes with a signal of length $N = 64$ that is $K = 10$ sparse in a 4-level db4 wavelet transform basis $\mathbf{\Psi}$ (results are the same using a DCT matrix as sparsifying matrix). As in the previous simulation, 100 simulation trials are performed for each value of M_j. JSM-2 recovery is performed using the DCS-SOMP algorithm as in [Baron et al., 2005] whereas KCS reconstruction is obtained using a sparsifying matrix $\mathbf{\Delta} = \mathbf{I} \otimes \mathbf{\Psi}$ where the symbols are to be intended as usual.

The result in Figure 4.2 shows how the DCS-SOMP algorithm outperforms both KCS and independent reconstruction. More specifically KCS performs worse than before because the spatial dimension in this case is hardly sparse and KCS

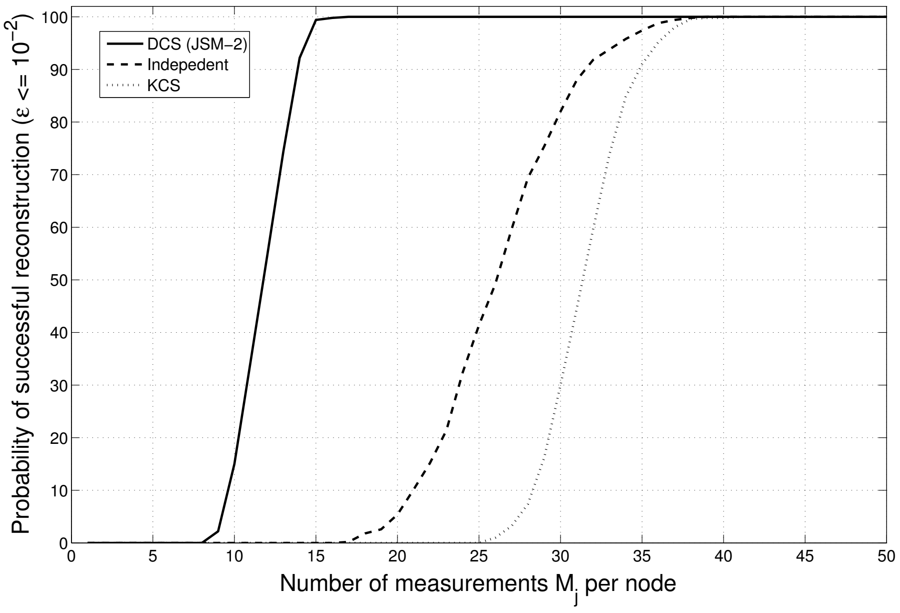


Figure 4.2: Comparison of joint reconstruction for DCS (JSM-2), KCS and separate decoding. All $J = 16$ signals of length $N = 64$ have the same sparsity $K = 10$

is not able to take advantage of the sparsity on this dimension.

The different reconstruction performance for KCS and DCS is mainly due to the different requirements about the original signal to compress. KCS requires that the signals are sparse along all the considered dimensions and this is sometimes a hard requirement to fulfill, especially for sensors in WSNs where signals are more correlated in time than in space. The requirements for DCS are less stringent since the algorithm just needs that the signals share the same support in one single basis along one single dimension.

The synthetic signals used in the simulations resemble realistic data that exhibit little spatial correlation: this is due to two common issues in real WSNs that is (a) irregular distribution of the node locations and (b) readings corrupted by noise and sporadic events. Then since the multidimensional signal (generated by the signal ensembles) usually is not really sparse along the spatial dimension, this prevents KCS to perform as well as DCS that can reconstruct the signal with more accuracy since it requires sparsity only along the temporal dimension.

The irregular distribution issue can be in general faced with special sparsifying transformation techniques like graph wavelets [Crovella and Kolaczyk, 2003] or diffusion wavelets that, however, require a strict and fixed tree network topology. The second issue of readings corrupted by noise can be addressed using data reordering schemes as proposed in [Luo et al., 2010] which unfortunately are not feasible when dealing with multidimensional signals.

From these considerations and from the simulation results it follows that in general DCS performs better than KCS when we want to exploit the inter- and intra-correlation among signals to achieve a better compression factor preserving a high reconstruction quality. Moreover even though JSM-1 and JSM-2 are both suitable models for the kind of natural signals we usually deal with in WSNs, JSM-2 reconstruction algorithm has a lower complexity as the reconstruction problem goes through a fast greedy algorithm. The resolution of the basis pursuit problem (3.5) for JSM-1 is too heavy when tens or hundreds of nodes are considered. Moreover the optimization problem (3.5) is often rewritten as an γ-weighted ℓ_1-norm problem in which the optimal choice of the parameters cannot be determined analytically and whose calculation is computational intensive.

In this section we have not dealt with JSM-3 since it does not match well the natural signals we are interested in, being this model well suited for video reconstruction problems.

4.2.2 JSM-2 model for real signal ensembles

To study the usage and the performance of DCS with JSM-2 signal ensembles in this section we consider data coming from the LUCE deployment that is a measurement campaign which took place on the EPFL campus since July 2006. This deployment is suitable for our research because it is characterized by temporal and spatial high density measurements covering heterogeneous areas. In evaluating DCS, we use three different sets of data: temperature, humidity and light. The set considered in our simulations spans the course of approximately 4 days with sampling periods in the order of 5 minutes. Due to the high fragmentation and incompleteness of the data in the sensor recordings, we consider the sensor readings of $J_{th} = 57$ nodes for temperature and humidity and $J_l = 14$ nodes for light recordings with a resampled signal length of $N = 1024$.

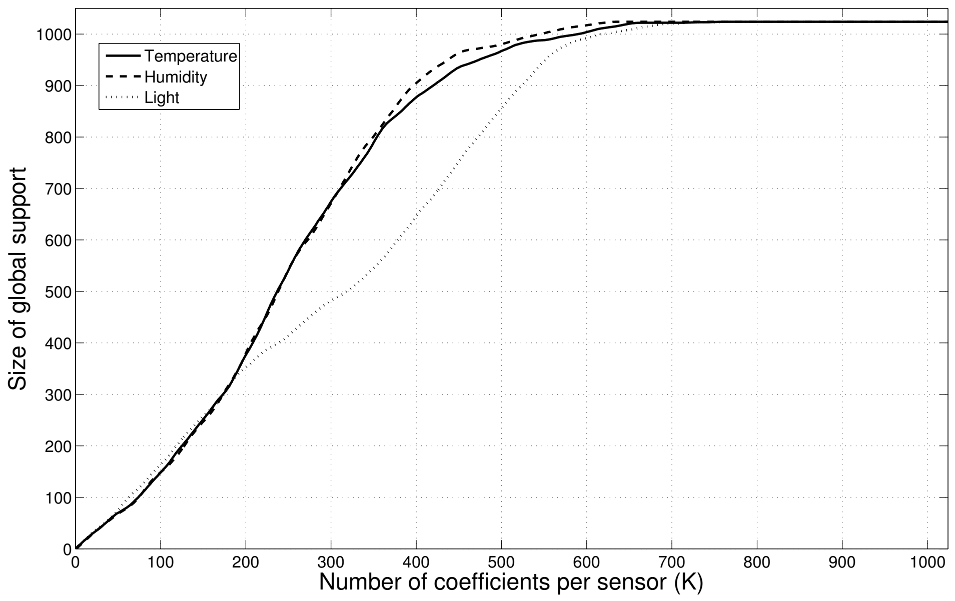


Figure 4.3: Number of wavelet vectors required to include the K largest wavelet coefficients for each signal

To see whether these natural signals can be recovered by DCS-SOMP algorithm we investigate the size of the global support $|\mathbf{\Omega}|$ for each type of signal as illustrated in Section 4.1.1.2. In Figure 4.3 the number of transform coefficients required to include the K largest coefficients for each signal is reported when a 8-level db8 wavelet transform is used. Is well-known in literature that the wavelet basis are suitable sparsifying basis (together with DCT also used in Section 4.2.1) for environmental signals and it is also known that with 6- or 8-layer wavelet representation it is possible to achieve a good sparsity for natural signals [Luo et al., 2010].

The slope of the curves show that, even if the signals could not exactly share the same support, the supports of the compressible signals overlap thus the ensemble is well represented by the JSM-2 model.

To further check whether the DCS-SOMP algorithm is able to reconstruct the original signals taking advantage of the inter-signals correlation and to investigate its performance, in Figure 4.4 we recover the original signals comparing

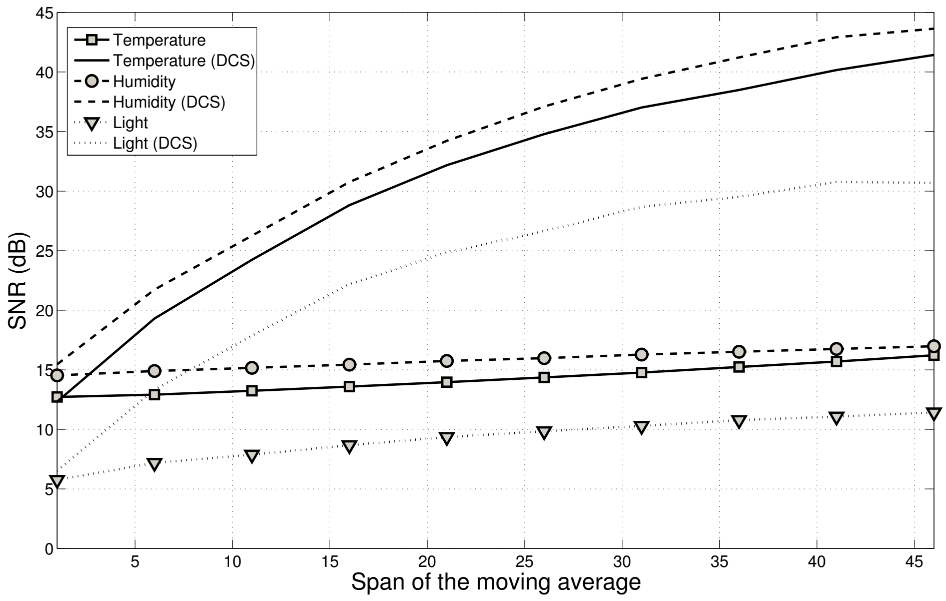


Figure 4.4: Reconstruction quality for DCS and separate decoding varying the span of the lowpass filter ($N = 1024$, $M = 120$)

the DCS approach with a separate recovery of the signals using CS when only $M = 120$ random projections are taken thus achieving almost a $1 : 10$ compression ratio. The reconstruction quality is measured in terms of SNR intended as $SNR(db) = 20 \cdot \log_{10}(\|\mathbf{x}\|/\|\mathbf{x} - \hat{\mathbf{x}}\|)$. To investigate the influence of the noise on the reconstruction, we apply a lowpass moving average filter with filter coefficients equal to the reciprocal of the span and evaluate the decoding performance for each filter window size. This is done because the signal is naturally affected by noise and in literature it is not well investigated how much the noise impacts on the joint reconstruction.

The results in the plot show how the DCS outperforms almost always the separate independent CS reconstructions and that averaging 20-25 samples is an optimal trade-off point between complexity in filtering, delays and reconstruction quality. It is important to notice also how the independent reconstruction is barely affected by the smoothing process, showing just a small improvement in the reconstruction quality with the increasing in the span of the moving filter.

The real advantage in using DCS considering JSM-2 signal ensemble is actually that we can send fewer bits over the air due to a shorter vector of random projections, saving in transmission costs guaranteeing at the same time on optimal reconstruction quality at collection point.

In general we can say that the interval of feasible number of random projections is delimited by the different values: (1) M_q considered function of the minimum reconstruction quality of the signal we can tolerate and (2) M_e depending on the energy consumption required for compression. That is $M_q \leq M \leq M_e$.

Classical CS theory Liu et al. [2010] claims that, for independent reconstruction, M_q depends on the signal sparsity according to the relation $M_q = \log(N)\left[\epsilon / \left(C_p \cdot S\right)\right]^{1/(0.5-1/p)}$ where N is the length of the original signal, p and C_p are constants, S is the sparsity and ϵ is the MSE between the original and the reconstructed signal. When DCS is used, the value of M_q is lowered since we can achieve a better reconstruction with less measurements. On the contrary M_e is determined by the maximum energy consumption for compression allowed by the node.

When the compression level has to be defined in a network, these two values have to be taken into account. If a better reconstruction is required by application, the value of M is chosen closer to (but not above) M_e, while a consistent energy saving is achievable for values of M closer to M_q at expenses of signal recovered quality. This is why DCS is a powerful tool for reducing the energy expenditure: we can choose value of M closer to M_q if we are interested in energy saving or, for the same value of M we can achieve a better reconstruction with the same energy consumption.

The following section investigates the energy threshold M_e and the methodologies to reduce the energy required by DCS.

4.2.3 Efficient DCS implementation

One of the major problems, often underestimated in literature, is the energy spent in compression. CS data compression goes through a matrix-vector multiplication, as seen in Section 3.3.2, that is not performed at zero cost. The energy expenditure for compression, not always negligible, that determines also the value M_e defined in Section 4.2.2, is mainly function of two different parameters: (1)

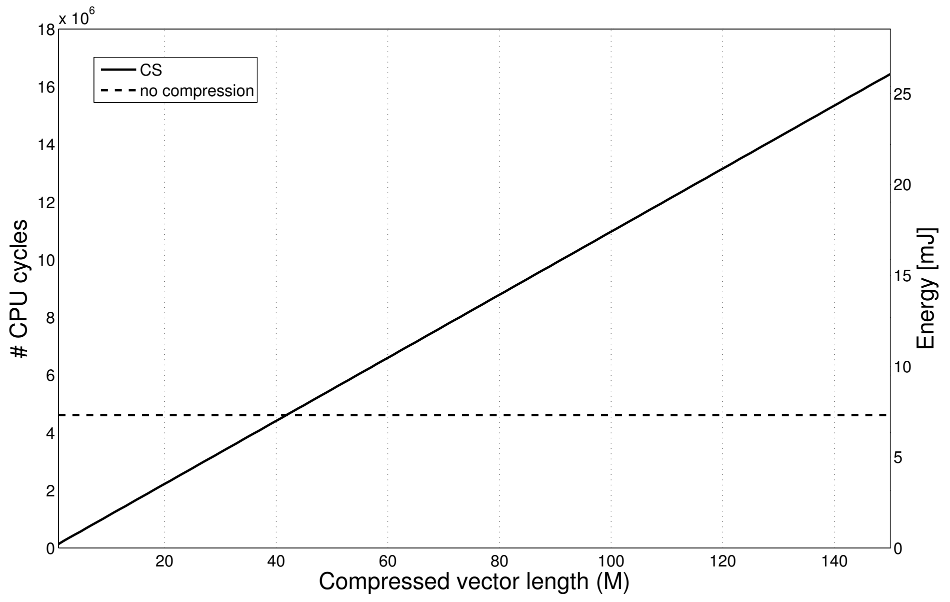


Figure 4.5: Comparison between the energy spent in compression and transmission for CS and the energy for transmission when no compression is performed. On the second axis is also reported the number of cycles required to compress data using a random orthogonal matrix generated on the node ($N = 512$)

the hardware used to implement DCS and (2) how the compression is performed.

Hardware The hardware used as reference in our tests is a wireless node by ST microelectronics, the STM32W108 that is a fully integrated System-on-Chip having a 2.4 GHz, IEEE 802.15.4-compliant transceiver, 32-bit 24MHz ARM Cortex-M3 microprocessor, 128K-byte Flash and 8K-byte RAM memory and peripherals of use to designers of 802.15.4-based systems. In our tests we assume that the microprocessor is in deep sleep mode when it is not busy in performing compression, with a quiescent current of $1.3\mu A$ @ $3.6V$. The processor and peripheral (considering CPU, RAM and flash memory) consume around $7.5mA$ @ $3.6V$ whereas the most important contribution to power consumption comes from radio, with a total current consumption in reception of $26.5mA$ @ $3.6V$ and in transmission $43.5mA$ @ $3.6V$. The processor does not have a floating point unit in hardware, but it can manage floating point calculations through software emulation. The compiler used in all the simulations and tests is CodeSourcery

G++ Lite 2010q1-188 and the code is compiled with -O3 optimization. Moreover to measure the time involved in compression and sending, we use the debug registers in the ARM core that are able to measure the number of cycles used in performing a certain operation; hence, knowing the working frequency, it is straightforward to obtain the elapsed time and the energy spent.

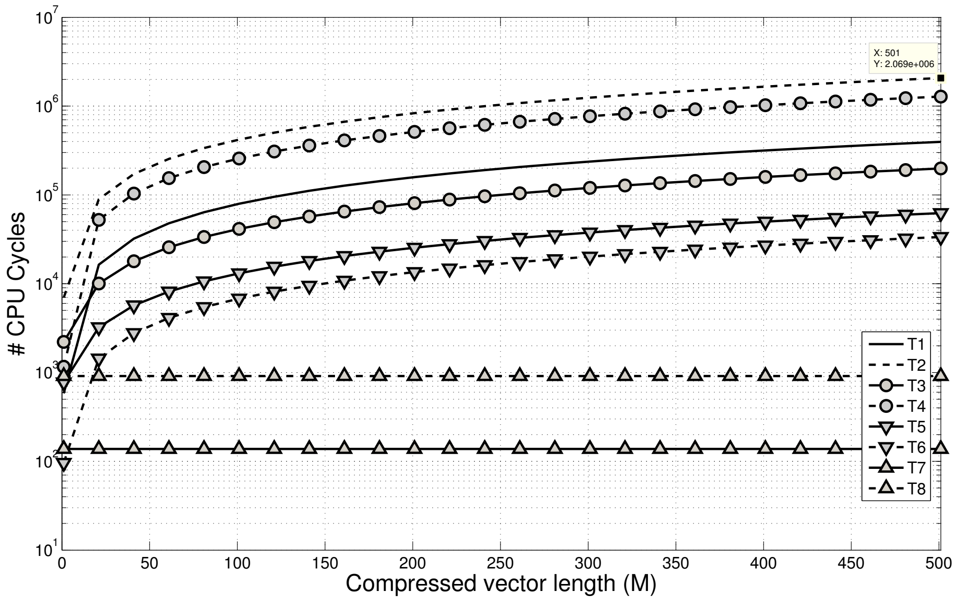


Figure 4.6: Number of CPU cycles required to compress a sample using different random $\mathbf{\Phi}$ matrices varying the compression factor. (**T1**) Matrix with random 16bit fixed-point values. (**T2**) Gaussian matrix generated using a Box-Muller transformation with mean zero and variance $1/M$. (**T3**) Matrix with random floating point values. (**T4**) Same as **T2** but the matrix is generated with the Ziggurat method. (**T5**) Entries of the matrix are generated from the symmetric Bernoulli distribution with $P(\Phi_{jk} = \pm 1\sqrt{M}) = 1/2$. (**T6**) Same as **T5** with $P(\Phi_{jk} = \pm 1) = 1/2$. (**T7**) Binary sparse matrix with $d = 1$. (**T8**) Binary sparse matrix with $d = 10$

In Figure 4.5 the compression and transmission energy versus compressed vector length M is plotted when CS and no compression is used for a random signal of length $N = 512$. The length of compressed vector at the intersection point defines the value M_e at which it is no more convenient to compress data since the energy for compression is higher than the energy for transmission of uncompressed data. While the energy spent when data is not compressed is constant and only function of the size of the outgoing packet, the energy spent in compression using CS increases with the increasing in the length M of the compressed vector.

From the plot follows that the energy spent for CS strictly depends on the compression energy that turns out to be function of how the compression itself is performed.

Compression CS theory claims that for CS it is possible to use several kind of random compression matrices $\mathbf{\Phi}$ derived from random matrix ensembles: uniform spherical ensemble, random signs ensemble, partial Fourier ensemble, etc... Each matrix has different characteristics and hence its usage implies a different power consumption. In particular, the power consumption is due to the contribution of three different factors: (1) energy spent in matrix generation, (2) energy for matrix-vector multiplication and (3) energy spent for transmitting the resultant compressed vector. In Figure 4.6 the number of CPU cycles required to compress a sample using different random $\mathbf{\Phi}$ matrices varying the compression factor is reported. It is possible to notice how substantial differences in the computational workload do exist among the different kind of random matrices. It is known in literature [Mamaghanian et al., 2011] that this difference is mainly due to the generation of the random columns of the matrix. In certain cases the matrix generation implies the use of complex functions such as log or sqrt that require a lot of CPU cycles to be performed.

For each random matrix in the plot, we have in general a different power consumption for data compression and a different reconstruction quality of the original signal. The smaller the power consumption for compression the greater the value of M_e; that is DCS becomes more convenient for a greater range of measurement vectors and in general more energy-aware for each value of M.

From the measurements it turns out that the more energy-aware random matrix is the binary sparse random matrix which allows a very fast and efficient implementation of the large matrix multiplication required by CS. This is a peculiar matrix having only a fixed small number d of "ones" in each column and all the other entries are equal to zero. The generation of the columns of this type

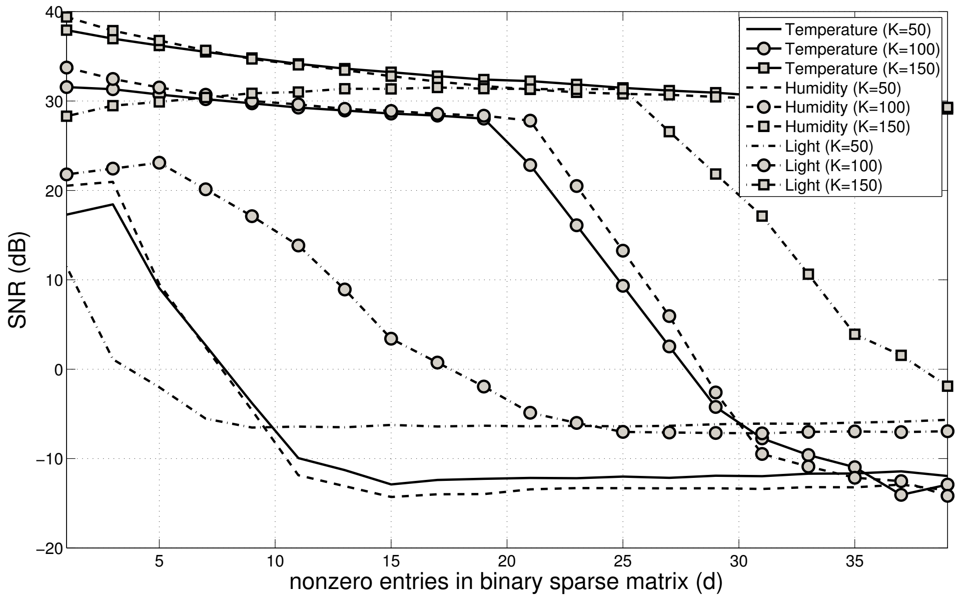


Figure 4.7: Average SNR varying the sparsity d of the binary sensing matrix. Signals of length $N = 1024$ are compressed using a 8-level db8 wavelet matrix with a compressed vector length of $M = 50$, $M = 100$ and $M = 150$

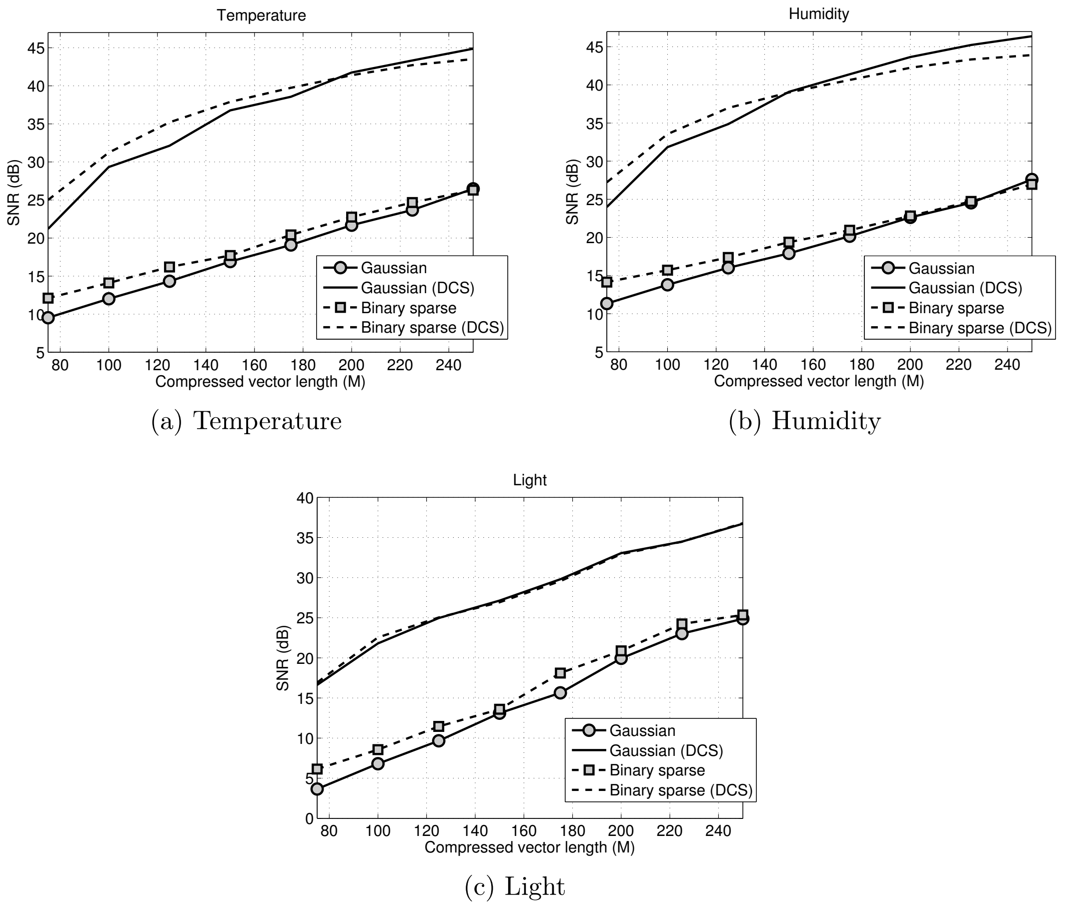
of measurement matrix can be obtained with a computational cost that does not depend on the dimension M of the compressed vector.

It has been shown [Gilbert and Indyk, 2010][Mamaghanian et al., 2011] that this type of matrix satisfies a weaker form of the RIP property and that, in practice, these binary sparse matrices are as good as random Gaussian or Fourier matrices.

Although these matrices have been used in literature, to the best of our knowledge this is the first work addressing this type of measurement matrices in conjunction with DCS and JSM-2 signals.

4.2.4 DCS with sparse random matrices

As first step we are interested in identifying the minimum value of d nonzero entries in each column of $\mathbf{\Phi}$ that guarantees the optimal trade-off between execution time, signal reconstruction error and power consumption. To get the most suitable value, sparse binary sensing matrices are applied to the deployment data



(a) Temperature (b) Humidity (c) Light

Figure 4.8: Comparison between reconstruction performance of Gaussian matrix and binary sparse matrix ($d = 1$) for temperature, humidity and light signals. DCS and independent reconstruction using CS are compared. The original signals of length $N = 1024$ are sparsified using an 8-level db8 wavelet matrix

and the output SNR of the reconstructed signals is measured for DCS.

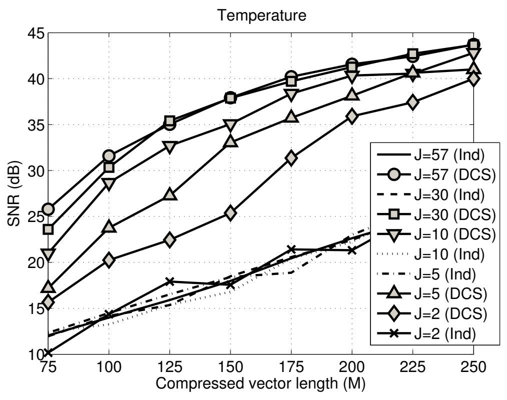
Figure 4.7 reports the resulting average output SNR versus the number of nonzero elements d in the sparse binary sensing matrix $\mathbf{\Phi}$ when a 8-level db8 sparsifying matrix is used for reconstructing each signal, varying the length of the compressed vector from $M = 50$ to $M = 150$. The general trend reported in the plot is that for small values of d the quality of the reconstruction using DCS-SOMP is higher than for greater values of d. That is the performance of DCS-SOMP with sparse binary matrices decreases with increasing in the value of d. This is due to information aliasing when the DCS-SOMP algorithm is used.

Thus from the simulation results it turns out that the best binary sparse matrix to be used with DCS is a sparse binary matrix with $d = 1$, that is a matrix having just a single one in each column in a random position.

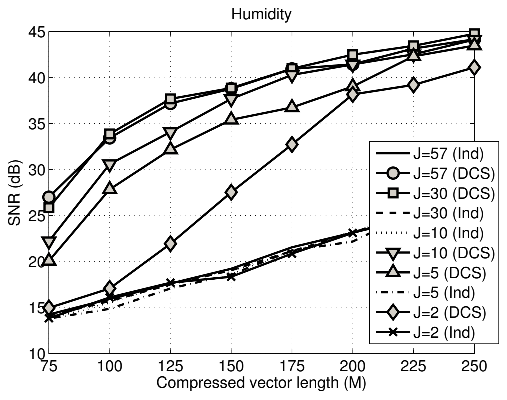
We have also performed several simulations to compare the reconstruction performance of sparse matrices with respect to the classical Gaussian matrices for DCS and independent reconstruction using CS. From the results in Figure 4.8 we can infer that the binary sparse matrix is optimal for DCS reconstruction, since it is able to guarantee a reconstruction quality similar to that one obtained from Gaussian matrices. In particular increasing the length of the compressed vector causes the increasing in the reconstruction quality for both DCS and independently reconstructed signals.

Since the DCS performance are strictly related to the number of nodes involved in reconstruction, in Figure 4.9 we have performed the same reconstruction for DCS as in Figure 4.8 but varying the number of sensor nodes used for recovery. The trend in the figure clearly shows how reducing the number of nodes of the ensemble, the reconstruction quality obtained by DCS decreases whereas the independent reconstruction is not affected at all by this parameter. This is an expected behavior for DCS since its performance are directly related to the number of signals considered.

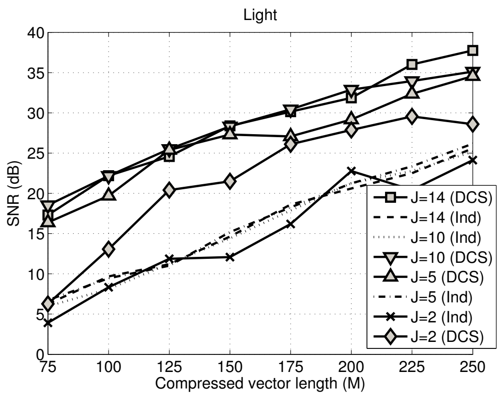
Once determined how DCS can be effectively used to achieve a better reconstruction quality, in the last set of trials we have calculated the energy consumption of the DCS when Gaussian matrices and sparse binary matrices are used. Results are in Figure 4.10. The plot shows how using DCS with sparse binary matrices permits a near-optimal reconstruction quality with less energy than that one required by both Gaussian matrices and transmission of data without compression. According the plot in Figure 4.6 all the other matrices are placed between these two cases. The plot clearly shows three different groups: (1) when no compression is performed the energy spent is always the same and it is only due to data transmission, (2) when Gaussian matrix is used form compression we need higher energy to compress data, mainly due to the work of generating the measurement matrix on the node, (3) very small energy is required for compression using the sparse binary matrix and this permits to save energy prolonging



(a) Temperature



(b) Humidity



(c) Light

Figure 4.9: Signals recovery using DCS and independent reconstruction when a binary sparse matrix ($d = 1$) is used as measurement matrix. The number J of the nodes involved in reconstruction is changed for each signal considered and the reconstruction quality of the reconstruction is evaluated

the nodes lifetime.

4.3 CS with sub-Nyquist sampling rate

The great majority of the papers addressing CS and DCS deals with a purely digital implementation of CS where the signal is sampled at a given frequency (e.g. Nyquist or above) and then compressed using CS. Nevertheless when natural signals have a relatively low information content as measured by the sparsity of

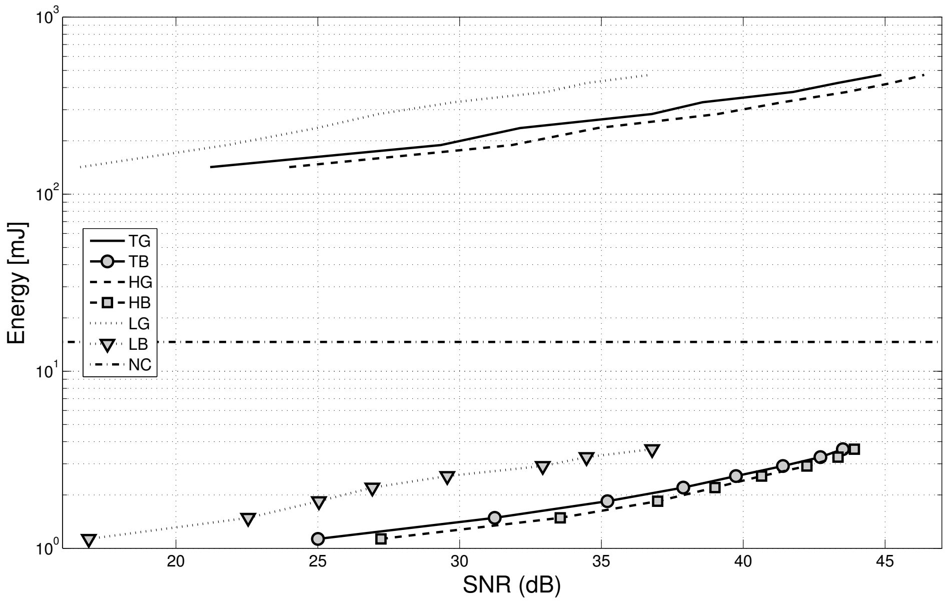


Figure 4.10: Energy consumption for data transmission when no compression (**NC**) is applied and energy for compression and transmission when Gaussian or sparse binary matrices are used in DCS versus the average reconstruction quality of the signals. (**TG**) Temperature (Gaussian). (**TB**) Temperature (Binary sparse). (**HG**) Humidity (Gaussian). (**HB**) Humidity (Binary sparse). (**LG**) Light (Gaussian). (**LB**) Light (Binary sparse). The length of the signals is $N = 1024$

their spectrum, the theory of CS suggests that randomized low-rate sampling may provide an efficient alternative to high-rate uniform sampling. This technique is usually referred to as analog CS and it is a novel strategy to sample and process sparse signals at sub-Nyquist rate [Ranieri et al., 2010].

4.4 CS in embedded systems

4.4.1 Hardware and compression

In this section we want to analyze the real potential of CS aiming at low-complexity energy-efficient data compression on resource constrained WSN platforms.

CS is usually considered a suitable approach for data acquisition and compression in WSNs. It is claimed [Caione et al., 2012] to be particularly attractive for energy-constrained devices for at least two reasons: (1) the compression is agnostic to the specific properties of the signal and it is performed through a small number of linear independent measurements and (2) the small number of measurements can be transmitted to a remote gathering center where they can be accurately reconstructed using complex, nonlinear and energy expensive decoders.

Nevertheless the energy spent in compression is often underestimated in literature. When implemented in software, data compression goes through several matrix-vector multiplications as seen in Section 3.3.2 that are not negligible, especially when resource-constrained nodes are used for compression and for the generation of the measurement matrix.

The hardware used as reference in our tests is a wireless node by ST microelectronics, the STM32W108 that is a fully integrated SoC with a 2.4GHz IEEE802.15.4-compliant transceiver, 32bit 24MHz ARM Cortex-M3 microprocessor, 128KByte Flash and 8KByte of RAM memory. Two additional sensors Sensirion SHT21 are considered on the board. The micro-controller has no floating point unit and it uses software emulation to overcome this limitation. The compiler used for compiling benchmarks is Sourcery CodeBench Lite Edition and the code is compiled with -O3 optimization. The time measurement is performed using the debug registers in the ARM core capable to accurately measure the number of cycles spent in performing a certain operation.

Data for power consumption of the various subsystems are not here reported for lack of space. For reference the reader can refer to the data-sheets of micro-controller and sensors. Our tests and simulations track the reported data-sheet values with high fidelity.

Compression using CS can be performed using different kind of compression matrices $\mathbf{\Phi}$. In literature it is possible to find a plethora of papers arguing on different kind of sensing matrices [Gilbert and Indyk, 2010]. As seen in Section 3.3.2 the only requirement is that the sensing matrix is highly incoherent with the sparsifying basis $\mathbf{\Psi}$. Such property is practically verified for random matrices such as random matrices with independent identically distributed (i.i.d.) entries.

Interestingly, many efficient sensing matrices can be generated having different characteristics and hence different memory and power footprint, moreover they require a different number of bytes for encoding and then storing.

In Figure 4.6 the number of cycles required by micro-controller to generate the compression matrix and perform the compression of a single sample for different kind of measurement matrices is shown. The differences are mainly due to (1) the computational workload required for generating the random vectors for the compression since in some cases the generation implies the use of complex and computationally intensive functions such as *sqrt* or *log* and (2) the time spent in multiplication of the vector against the sample that, especially in the case of floating point numbers, is not negligible.

4.4.2 Power consumption model

When CS is used to perform compression in a WSN, the type of compression matrix strongly affects also the power consumption of other subsystems: (1) the longer the time necessary to compress the data is, the longer the node has to be awake before switching back to sleep mode to save energy; (2) the number of bytes required to encode the compression matrix is not the same for all the matrices $\mathbf{\Phi}$; (3) following from the previous point, the time and space required by the micro-controller to store data in non volatile memory is different and (4) the energy spent in transmission is different for measurement matrices.

To evaluate the influence of the choice of measurement matrix and other compression parameters, in this subsection we introduce an architecture level power consumption model to evaluate the power consumption of the nodes when compression is performed using different parameters for compression and we compare the results against the power spent to transmit data without any kind of compression. Using this power model and feeding it with data coming from real hardware we can easily evaluate how changing the parameters influences the energy consumption of the system enabling design space exploration.

The hardware taken as reference (already described in this section) is a STM32W108 node acquiring data from the two on-board sensors. The network is organized as a star, a very common topology for practical WSN deployments [Rajagopalan and Varshney, 2006].

During the simulation involving no compression the node wakes up, samples data from the two sensors and sends them out to a collector center. Afterwards it goes back in sleep mode waiting for next cycle. The energy spent in each cycle can be written as:

$$E = E_{\text{sleep}} + E_{\text{setup}} + E_{\text{sample}} + E_{\text{trans}} \tag{4.10}$$

where E_{sleep} is the energy spent in sleep mode, E_{setup} is the energy used for waking up and setting up the device, E_{sample} is the energy for sampling each sensors and E_{send} is the energy used to send the acquired data. Expanding each term we have:

$$\begin{aligned} E = {} & T_{\text{sleep}} \cdot (P_{\text{sleep}} + P_{\text{soff}} + P_{\text{toff}}) + \\ & T_{\text{setup}} \cdot (P_{\text{mcu}} + P_{\text{soff}} + P_{\text{toff}}) + \\ & T_{\text{sample}} \cdot (P_{\text{sample}} + P_{\text{sactive}} + P_{\text{toff}}) + \\ & T_{\text{trans}} \cdot (P_{\text{comm}} + P_{\text{soff}} + P_{\text{trans}}) \end{aligned} \tag{4.11}$$

where $T_{\text{sleep}}, T_{\text{setup},} T_{\text{sample}}, T_{\text{trans}}$ are the duration of each respective phase. P_{sleep} is the power consumed in sleep mode, P_{soff} is the power absorbed from sensors when sleeping, P_{toff} is the power consumption of the transceiver when the node is in sleep mode. P_{mcu} is the power consumed by the MCU, P_{sample} is the power spent for data acquisition, P_{sactive} is the power consumed by sensors, P_{comm} is the power consumption for filling the transceiver output buffer and finally P_{trans} is the power for sending data. All the values for the power consumption or timing are taken from the datasheets or actually measured on the hardware.

When CS is used to compress data, the compression is performed after the node has acquired N_{acc} samples. Thus the energy consumption in each cycle is:

$$\begin{aligned} E_{\text{CS}} = (&N_{\text{acc}} \cdot (E_{\text{sleep}} + E_{\text{setup}} + E_{\text{sample}} + E_{\text{store}}) + \\ &E_{\text{nv}} + E_{\text{comp}} + E_{\text{trans}})/N_{\text{acc}} \end{aligned} \tag{4.12}$$

Where E_{store} is the energy to store the acquired sample in non volatile memory, E_{nv} is the energy spent during the recovery of the data from non volatile memory and E_{comp} is the energy for compression. In detail:

$$\begin{aligned} E_{\text{CS}} = (&N_{\text{acc}} \cdot (T_{\text{sleep}} \cdot (P_{\text{sleep}} + P_{\text{soff}} + P_{\text{toff}}) + \\ &T_{\text{setup}} \cdot (P_{\text{mcu}} + P_{\text{soff}} + P_{\text{toff}}) + \\ &T_{\text{sample}} \cdot (P_{\text{sample}} + P_{\text{sactive}} + P_{\text{toff}}) + \\ &T_{\text{nv}} \cdot (P_{\text{store}} + P_{\text{soff}} + P_{\text{toff}})) + \\ &T_{\text{store}} \cdot (P_{\text{soff}} + P_{\text{toff}} + P_{\text{store}}) + \\ &T_{\text{comp}} \cdot (P_{\text{soff}} + P_{\text{toff}} + P_{\text{comp}}) + \\ &T_{\text{trans}} \cdot (P_{\text{comm}} + P_{\text{soff}} + P_{\text{trans}}))/N_{\text{acc}} \end{aligned} \tag{4.13}$$

with obvious meaning of the symbols.

In Figure 4.11 the result of simulations is reported when $N_{\text{acc}} = 512$, $M = 100$, $T_{\text{sleep}} = 10s$ with an overhead of 10 bytes for each packet sent. The other parameters in Equation (4.11) and (4.13) are derived from these values and the hardware specification data in datasheets. The two compression matrices used in the simulation when CS is performed are: (**T2**) Gaussian matrix generated using a Box-Muller transformation with mean zero and variance $1/M$ and (**T6**) matrix generated from the symmetric Bernoulli distribution $P(\Phi_{jk} = \pm 1) = 1/2$. According to Figure 4.6 these two matrices define the energy consumption boundary for CS.

The result of the simulation clearly shows how not always compressing data with CS determines an actual saving in power consumption. For all the cases the energy spent in sleep mode, the energy for sampling and the energy for setting up the node after the sleep is obviously the same. The differences are related to the energy for compression and for sending the data.

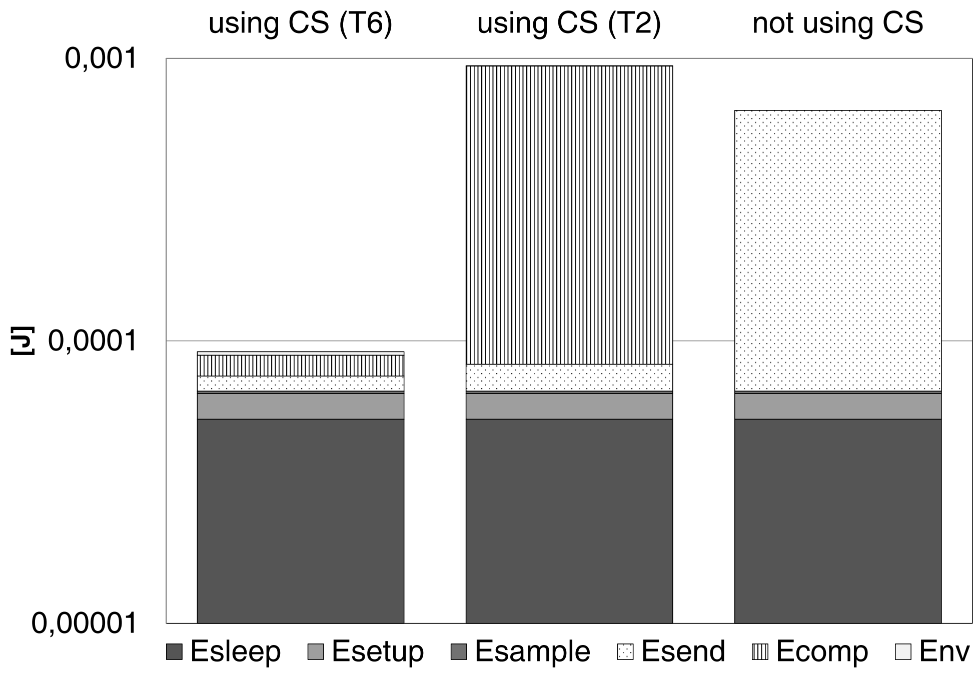


Figure 4.11: Energy spent in one sampling cycle when CS is used to compress the sample compared to the energy consumed when the sample is sent without compression. The first bar refers to CS when measurement matrix is obtained from a Bernoulli distribution (**T6**) while for the second bar the compression is performed using a Gaussian matrix (**T2**). (Simulation parameters: $N_{\text{acc}} = 512$, $M = 100$, $T_{\text{sleep}} = 10s$)

Using a complex compression matrix (**T2**) is really expensive in terms of energy consumption thus the overall power consumption is higher with CS than without any compression. Differently, when a simpler matrix is used (**T6**) the energy for compression becomes negligible, and the power consumption abruptly decreases. A large difference between using CS and not using compression is also in the power for sending data due to two different factors: (1) the number of bytes sent and (2) a better packetization since the compressed vector is sent at the end of the N_{acc} cycles permitting to maximize the number of compressed samples that fit in the packet payload.

4.4.3 Low-Rate CS

In this section we want to investigate how it is possible to further reduce the energy consumption by means of simpler sparse measurement matrices and acting on the number of samples gathered by the node.

In classical acquisition systems (as in the digital CS seen before), samples are taken regularly on the time axis at a given rate (usually not less than the Nyquist one). A particular form of CS, called analog CS, relies on random sampling to avoid this regularity and aims to produce a number of measurements that, on the average, are less than those produced by Nyquist sampling, while still allowing the reconstruction of the whole signal resorting to sparsity and other priors.

While usually analog CS is performed by means of specialized hardware encoders, we want to study whether analog CS is a suitable technique to be performed on WSNs nodes and whether this peculiar form of compression, that we call from now on Low-Rate CS (LR-CS), is still able to reconstruct the original signals of interest with satisfying quality.

From a mathematical point of view the problem is still the same as seen in Equation (3.4), what is different is the form of the measurement matrix $\mathbf{\Phi}$. Let $\mathbf{B}$ denote an M-dimensional vector each element of which contains a unique entry chosen randomly between 1 and N. In analog CS the measurement matrix $\mathbf{\Phi}$ is a sparse $M \times N$ matrix where the i^{th} row of the matrix is an all-zero vector with 1 at the location given by the i^{th} element of $\mathbf{B}$. This is very simple measurement matrix, energetically cheap to generate, store and permits also to save on the number of samples to gather.

Practically using this type of measurement matrix means that the node is required only to randomly gather M samples with an under-sampling ratio of order $\rho = M/N$. As done before the energy consumption on average after N_{acc} sampling period is:

$$\begin{aligned} E_{\text{sub}} =& (M \cdot (E_{\text{setup}} + E_{\text{sampl}} + E_{\text{store}}) + \\ & N_{\text{acc}} \cdot E_{\text{sleep}} + E_{\text{nv}} + E_{\text{send}})/N_{\text{acc}} \end{aligned} \tag{4.14}$$

In Figure 4.12 the comparison between digital and low-rate CS is reported.

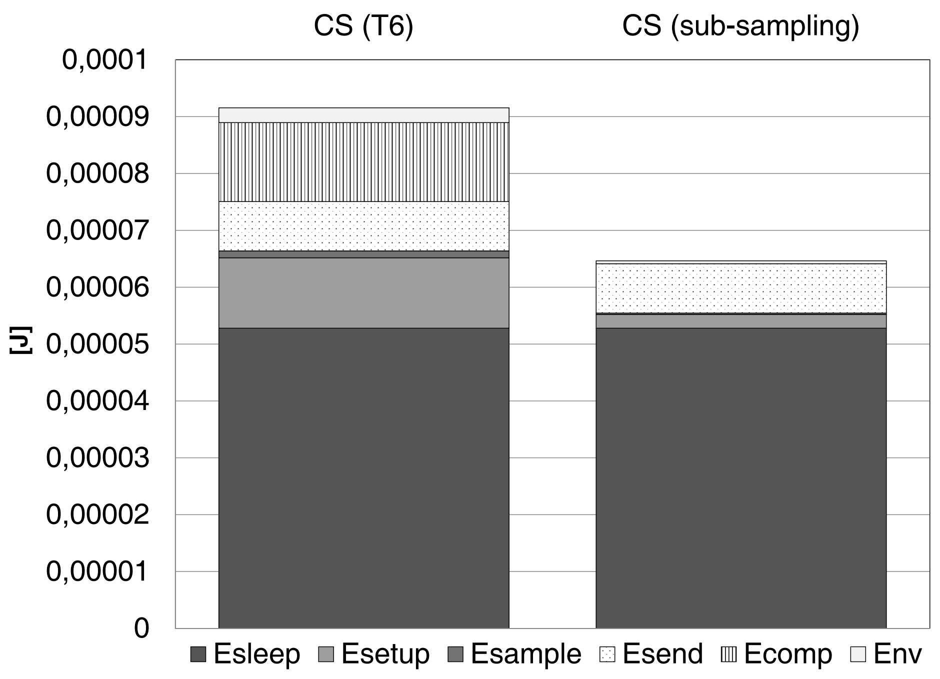


Figure 4.12: Energy comparison between digital and low-rate CS. (Simulation parameters: $N_{\text{acc}} = 512$, $M = 100$, $T_{\text{sleep}} = 10s$)

As inferred from Equation (4.12) and (4.14) the energy saving is mainly due to three to factors: (1) there is no energy spent in compression for the analog version of CS, (2) the contribution of E_{setup}, E_{sampl} and E_{store} is reduced by a factor ρ and (3) E_{nv} is decreased since the number of bytes to store in flash is reduced.

In Figure 4.13 the comparison between the energy spent for low-rate and digital CS is reported, normalizing the energy with respect to the energy spent when no compression is applied. The low-rate CS is *always* more convenient with respect to the digital CS. In the plot is also visible the influence of the packet overhead on the power consumption that creates small abrupt increases in energy consumption when an additional packet has to be sent.

Having verified that using Low-Rate CS and a sparse measurement matrix the node can save energy the problem shifts to verify whether low-rate CS can be used in practice to reconstruct signals gathered by WSNs nodes deployed in a real environment.

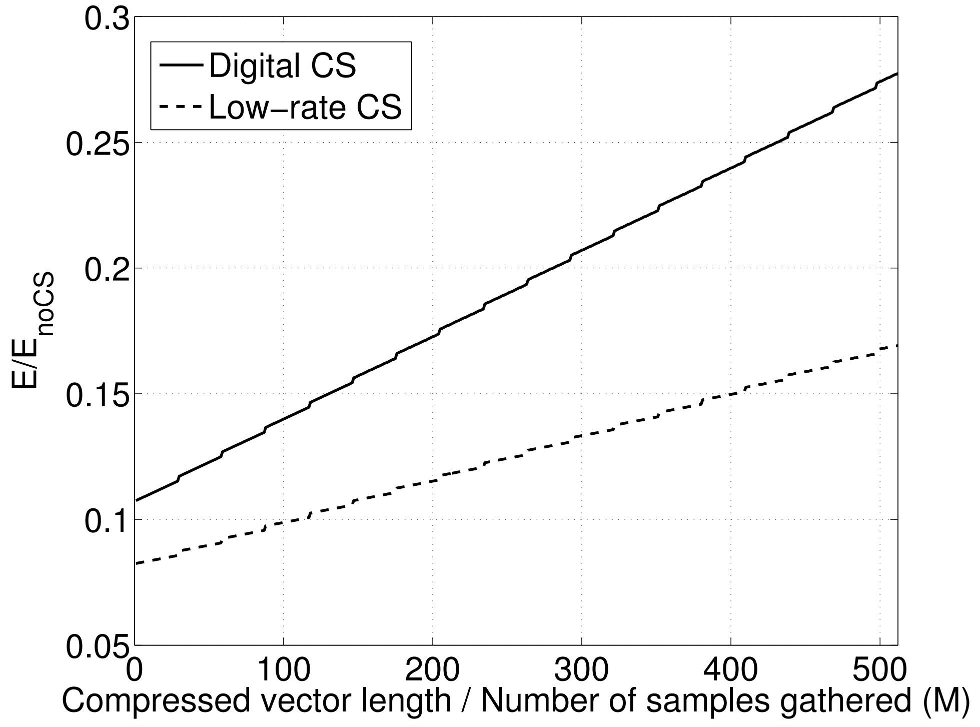


Figure 4.13: Energy comparison between digital and low-rate CS varying the compressed vector size for digital CS and the number of samples gathered for the low-rate CS

4.4.4 WSN data reconstruction for Low-Rate CS

In this section we want to investigate the performance of several reconstruction algorithms to check if there is an algorithm that better than others is able to cope with low-rate CS and that can guarantee a good signal recovery. Moreover we want to address the problem of choosing a suitable sampling pattern for the low-rate CS since the sampling pattern chosen is strictly related to the quality of the recovered signal during the reconstruction phase.

In our experiments we consider data coming from the CIMIS dataset that manages a network of over 120 automated weather stations in the state of California. We take as reference the data collected during the 23rd week of the 2012 by seven different weather stations near Monterey (CA). For our simulations we

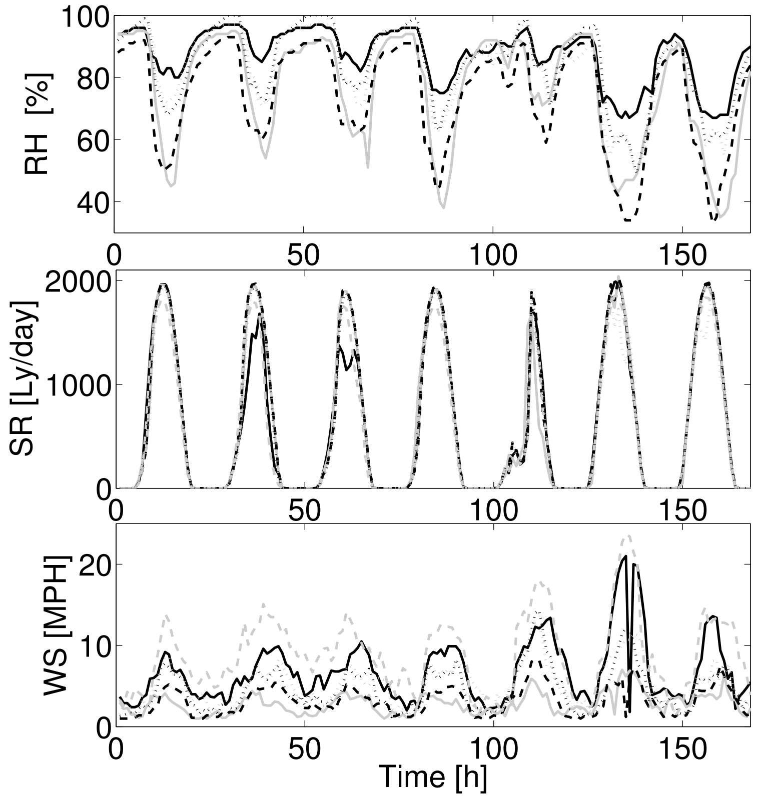


Figure 4.14: Signals ensembles for (a) relative humidity (**RH**), (b) solar radiation (**SR**) and (c) wind speed (**WS**) for seven different weather stations near Monterey (CA). Each different line in each sensor plot is referred to a different node: each kind of sensor presents a different level of correlation among different nodes.

refer to three different kind of sensors: temperature, relative humidity and wind speed, as reported in Figure 4.14. The ensemble of signals is chosen such that it includes periodic and highly correlated signals (temperature and relative humidity) with less-correlated signals (wind speed).

In our model the seven nodes are deployed in a IEEE 802.15.4 star network. The power consumption for each node adheres to the same model as described in Section 4.4. In each simulation cycle, each node samples the signal for a certain period, called acquisition period, collecting a certain number of samples before compressing these samples and sending out the compressed vector toward a central collector. The acquisition period is supposed be the same for each node and each node uses the low-rate CS for compressing data. The sparse compression matrix $\mathbf{\Phi}$ used for compression is locally generated by each node using its own id and the time-stamp as seed for generation. The compressed vectors are gathered by the central coordinator and here the original signals are recovered using different algorithms.

Two different sampling patterns for the generation of the measurement matrix $\mathbf{\Phi}$ are considered in this section: (1) uniform sampling (US) pattern and (2) non-uniform sampling (NUS) pattern. In the uniform sampling pattern the inter-measurements intervals are constant $\Delta k_j = k_{j+1} - k_j = \Delta k = \gamma \Delta k_{\min}$ where $\Delta k_{\min}$ is the minimum sampling period of the ADC and $\gamma = \lceil N/M \rceil$ whereas in the non-uniform sampling pattern the inter-sample period is randomly chosen between $[\Delta k_{\min}, \infty]$.

We carry the reconstruction using several algorithms, distributed and non-distributed, and evaluate the quality of reconstruction using the SNR expressed in dB:

$$\mathrm{SNR_{dB}} = 20 \cdot \log_{10} \frac{\|\mathbf{x}\|_2}{\|\mathbf{x} - \hat{\mathbf{x}}\|_2} \tag{4.15}$$

where $\mathbf{x}$ is the original signal and $\hat{\mathbf{x}}$ is its recovered version.

In particular we try the reconstruction using: (1) Basis Pursuit on single nodes averaging the quality of reconstruction over all the seven different signals. (2) DCS-SOMP algorithm considering a JSM-2 model for the signals ensemble. (3) Joint-sparse basis pursuit model (JS-BP) solved with YALL1 MATLAB package. (4) Gradient Projection base Sparse Reconstruction (GPSR).

While the BP does not exploit any correlation or priori information and DCS-SOMP and JS-BP try to exploit inter-correlations existing among the different nodes, the GPSR algorithm is well-suited both for periodic and correlated signals since it presents a weighting factor that can be used to give to the reconstruction algorithms some hints about reconstruction.

With the same nomenclature as in the previous section, the problem of signal reconstruction for GPSR can be expressed as:

$$\text{minimize } \left[\|\mathbf{\Psi}\mathbf{\Phi}\boldsymbol{\alpha} - \mathbf{y}\|_2^2 + \tau \|\mathbf{W}\boldsymbol{\alpha}\|_1 \right] \tag{4.16}$$

where τ is a non-negative parameter providing relative weight of the ℓ_1-norm and ℓ_2-norm in the cost function while $\mathbf{W}$ is a diagonal matrix with $\omega_1, \dots, \omega_n$ on the diagonal and:

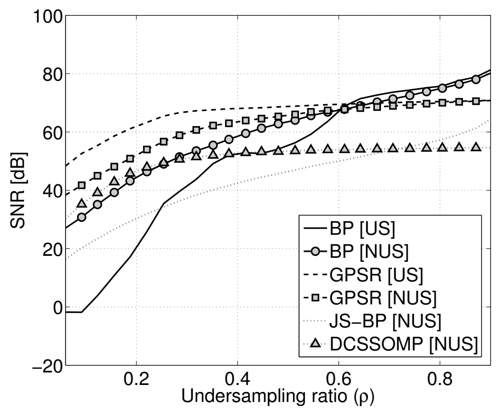
$$\omega_i = \frac{1}{|\eta_i| + \epsilon} \tag{4.17}$$

where $\epsilon > 0$ is in order to provide stability and in general the weights η_i are free parameters in the convex relaxation whose values could improve the signal reconstruction. The matrix $\mathbf{W}$ can be in fact used to incorporate a priori information about sparsity and can be estimated on-line from inter o intra-correlation data between sensors and nodes.

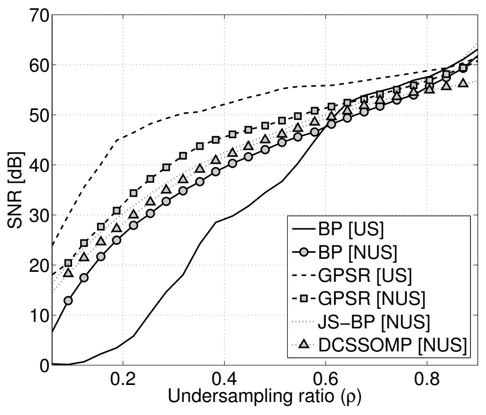
In this section we use the data from the same sensor the day before those ones involved in the reconstruction as training information for each sensor to obtain the $\mathbf{W}$ matrix, exploiting the temporal intra-correlation of each node.

In the simulations the acquisition period before sending out the compressed data toward the base station is 2 days (more precisely 42 hours). During this period each sensor of each node is sampled and M samples are gathered by the node according to the generated $\mathbf{\Phi}$ matrix. The minimum wake-up time (the minimum inter-sample period) is 5 min, so a maximum number of $N_{\text{acc}} = 512$ samples can be gathered by each node for each sensor in one acquisition period. The sparsifying matrix $\mathbf{\Psi}$ is a DCT matrix that is already demonstrated be a good basis for compressible natural signals [Caione et al., 2012]. Each simulation cycle is performed for 100 trials and for each run both the measurement matrix and the sampling pattern for the non-uniform random sampling are randomly generated.

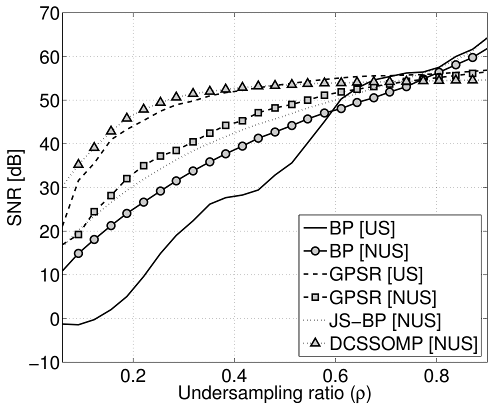
In Figure 4.15 the reconstruction quality for each kind of signal averaged over all the seven nodes is reported. The plot is done against the under-sampling ratio $\rho = M/N$ defined as the fraction of the samples actually taken respect to



(a) Humidity



(b) Solar



(c) Wind

Figure 4.15: Quality of reconstruction vs the under sampling ratio for the three kind of signals taken into consideration. Each signal is reconstructed using all the algorithms investigated in the paper, varying also the under-sampling pattern

the number of total samples.

The results clearly show how BP does not perform well for all the three signals when low under-sampling ratios are considered achieving a SNR that is lower than that one obtained with all the other algorithms. Algorithms involving the exploitation of spatial inter-correlation between nodes or temporal intra-correlation achieve a much better reconstruction quality for all the signals considered. In general results show that the better reconstruction quality is obtained using the

GPSR algorithm. This much higher SNR for reconstruction using GPSR is obtained by giving the reconstruction algorithm useful hints about the signal to reconstruct as seen in Equation (4.16). For the wind speed the reconstruction quality guaranteed by GPSR is comparable to that on achieved by DCSSOMP, this is due to the fact that the wind speed, among all the signals, presents a lower temporal correlation.

The plot also shows that, while for GPSR the uniform sampling (US) outperforms the non-uniform sampling pattern (NUS), for BP is the opposite.

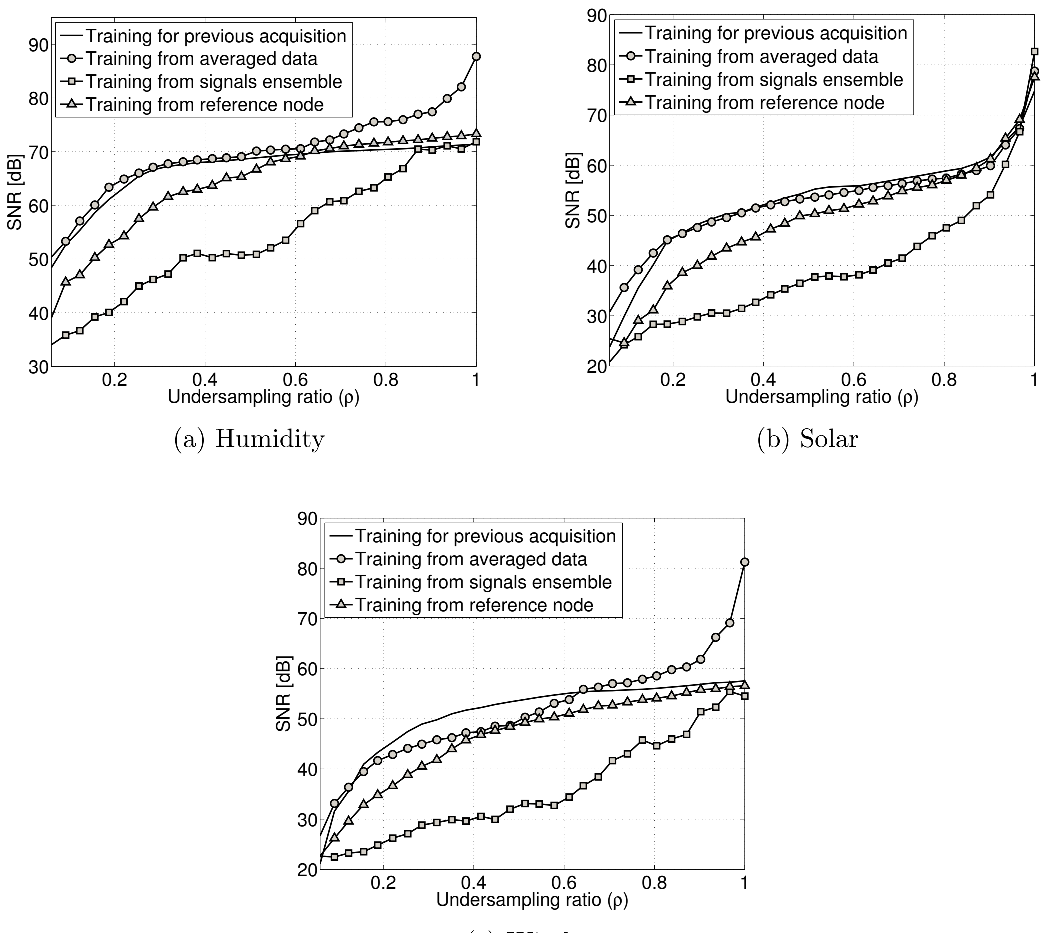
4.4.4.1 Training data for GPSR

From the results collected follows that the best algorithm able to provide a good reconstruction of the signals is GPSR. In this section we want to investigate how the training data (the parameters under the form of the $\mathbf{W}$ matrix in Equation (4.16)) can influence the reconstruction. This is particularly significant in WSNs where spatial and temporal correlations do exist between different nodes and within the node itself.

In our simulations we investigate four different scenarios, each aimed to exploit spatial correlation among nodes or temporal correlation within the sensor of interest to create a suitable data training for the GPSR reconstruction.

As seen in Figure 4.16 our training data is obtained: (1) exploiting temporal correlation by using data of the same sensor on the same node reconstructed in the previous acquisition cycle; (2) by averaging a maximum of 10 signals reconstructed in the previous acquisition cycles; (3) by using a pseudo-signal obtained combining the raw data gathered by neighbor nodes; (4) by using a line-powered node taken as reference providing uncompressed data placed near the compressing node. This last point is a fictitious case taken as reference since it is not always possible to have a line-powered node providing continuous stream of data, but it is useful to evaluate the recovery when spatial-correlated data is used for reconstruction.

The first result inferred from the simulations output is that, exploiting the spatial correlation using as training for the algorithm the pseudo-signal, is not convenient since the quality of the reconstruction is lower than that one obtained



(a) Humidity (b) Solar

(c) Wind

Figure 4.16: Quality of reconstruction varying the training data used in GPSR algorithm

using the other methods.

In general a better recovery is achieved when data temporally correlated with the signal that we want to recover is used as training data. This is particularly true for periodic signals such the environmental signals of interests. The best results in the compression range of interest are obtained by using as training for the GPSR algorithm data coming from the same sensor and node but gathered in a previous acquisition cycle. This can guarantee the maximum temporal (and obviously spatial) correlation giving helpful hints to the reconstruction algorithm to correctly recover the signal.

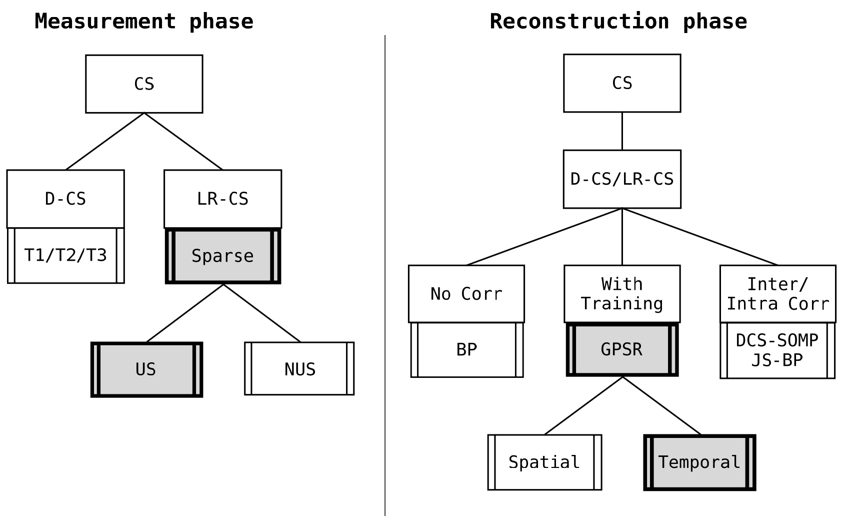


Figure 4.17: With the same nomenclature previously introduced this plot highlights the different choices for measurement and reconstruction phase that permit to achieve the better reconstruction with the minimum energy expenditure.

4.4.4.2 Energetically optimal reconstruction

In Sections 4.4 and 4.4.3 we have investigated the compression phase, coming to the conclusion that a sparse measurement matrix is the best compression matrix to save energy in compression. Afterwards in Section 4.4.4 we have obtained that, among several reconstruction algorithms and using this sparse measurement matrix, GPSR is the best algorithm capable to guarantee the higher reconstruction quality. In Figure 4.17 a graphical review of the best choices in measurement and reconstruction phase is reported.

Since we have investigated both the power consumption in compression and the reconstruction quality using GPSR it is possible to address the problem to find the optimal compression parameters able to guarantee good reconstruction quality with the minimum energy expenditure.

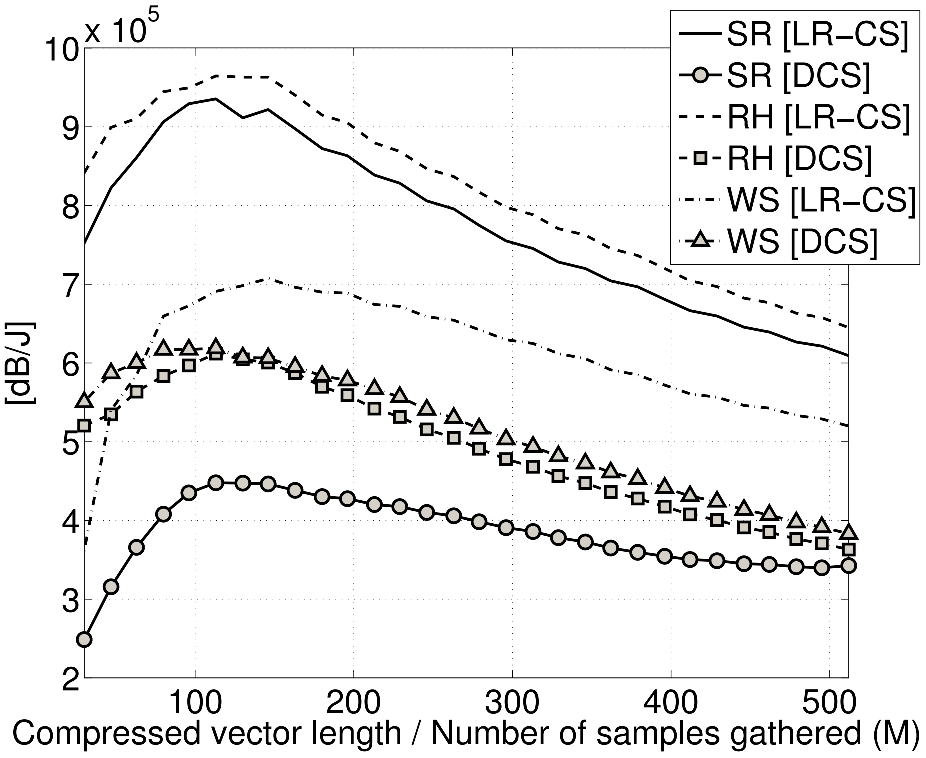


Figure 4.18: Trade-off between reconstruction and energy consumption for compression. (**SR**: Solar radiation. **RH**: Relative humidity. **WS**: Wind speed. **LR-CS**: low rate CS. **DCS**: digital CS.)

In Figure 4.18 the trade-off between quality of signal recovery and power consumption is reported plotting the ratio between quality of reconstruction and the energy spent in compression varying the under-sampling ratio ρ for low-rate CS and the compression vector size M for digital CS.

Looking at the plot we can see how the curves for LR-CS are always above the curves for the digital CS, meaning that for LR-CS the compression is energetically cheaper. More precisely this means that each dB in reconstruction is obtained using less Joules of energy during the compression phase.

Moreover within the same class of curves we have a range of compression values M and ρ (between $M = 100$ and $M = 200$ for digital CS and $\rho = 0.2$ and $\rho = 0.4$ for the low-rate CS) for which the curves present a maximum, identifying the best trade-off between reconstruction quality and power consumption for compression.

Comparing these values with the plots in Figure 4.15 we can see how in this range the quality of reconstruction is always > 30dB that is a very good reconstruction quality for our goals.

Thus the low-rate CS with an under-sampling ratio $0.2 \leq \rho \leq 0.4$ when reconstruction is performed using GPSR with temporally correlated data as training data is able to guarantee an optimal reconstruction > 30dB with minimum energy used for compression.

Chapter 5

Sub-sampling frameworks comparison

As seen in the previous chapter when the considered signals present spatial-temporal correlations, they can be used to reconstruct the desired data from only a limited portion of collected samples. The ability to reconstruct missing data enables the adoption of aggressive duty cycling policies on individual sensor nodes, sampling only the most informative parts of the data, thereby reducing the overall energy consumption.

Besides DCS another promising technique capable to recover the signal from an highly incomplete sub-sampled version is a special data-driven statistical model based on latent variables (LV). This approach extends the standard latent variable factorization model, which typically considers only dyadic interactions in data, to multivariate spatial-temporal data, by applying tensor decomposition techniques [Yang et al., 2008]. The key advantage of using a latent variable model is that it provides a compact representation of the gathered data that can be used to recover the missing samples. To perform well under extreme sub-sampling conditions, the standard technique is extended to explicitly incorporate the spatial, temporal, and inter-sensor correlations present in the considered data.

5.1 Group sparsity with CS

Beside the distributed techniques already explored in Section 4.1 and following subsection, there are several other techniques able to deal with groups of signals that are sparse. In practice, a wide class of solutions are known to have certain *group sparsity* structure. Namely, the solution has a natural grouping of its components, and the components within a group are likely to be either all zeros or all non-zeros. Encoding the group sparsity structure can reduce the degrees of freedom in the solution, thereby leading to better recovery performance.

The group sparse reconstruction problem has been well studied recently. A favorable approach in the literature is to use the mixed $\ell_{2,1}$-regularization. Suppose $x \in \mathbb{R}^n$ is an unknown group sparse solution. Let $\{x_{g_i} \in \mathbb{R}^{n_i} \;:\; i = 1, \dots, s\}$ be the grouping of x, where $g_i \subseteq \{1, 2, \dots, n\}$ is an index set corresponding to the i-th group and x_g denotes the sub-vector of x indexed by g_i. Generally, g_i's can be any index sets, and they are predefined based on prior knowledge. The $\ell_{2,1}$-norm is defined as follows:

$$\|x\|_{2,1} := \sum_{i=1}^{s} \|x_{g_i}\|_2 \tag{5.1}$$

just like the use of ℓ_1-regularization for sparse recovery, the $\ell_{2,1}$-regularization is known to facilitate group sparsity and result in a convex problem.

In general, instead of using the $\ell_{2,1}$-norm (5.1), it is convenient to consider the weighted $\ell_{w,2,1}$-norm defined by:

$$\|x\|_{w,2,1} := \sum_{i=1}^{s} w_i \|x_{g_i}\|_2 \tag{5.2}$$

where $w_i \geq 0$ are weights associated with each group.

5.1.1 Joint sparsity and MMV problem

An interesting special case of the group sparsity structure is called *joint sparsity* [van den Berg and Friedlander, 2008]. Jointly sparse solutions, namely, a set of sparse solutions that share a common nonzero support, have been already introduced in Section 4.1.1. Differently, in this case, the reconstruction of jointly sparse solutions, is known as the multiple measurement vector (MMV) problem.

This problem can be formulated as:

$$\min_X \|X\|_{w,2,1} := \sum_{i=1}^{n} w_i \|x^i\|_2 \qquad \text{s.t.} \qquad AX = B \tag{5.3}$$

where $X = [x_1, \dots, x_l] \in \mathbb{R}^{n \times l}$ denotes a collection of l jointly sparse solutions, $A \in \mathbb{R}^{m \times n}$ with $m < n$, $B \in \mathbb{R}^{m \times l}$, $w_i \geq 0$ for $i = 1, \dots, n$ and x^i and x_j denote the i-th row and j-th column of X respectively.

When dealing with WSN, we can consider an ensemble of J signals we can denote each signal with $\mathbf{x}_j \in \mathbb{R}^N$ with $j \in \{1, 2, \dots, J\}$. For each signal $\mathbf{x}_j$ in the ensemble we have a sparsifying basis $\mathbf{\Psi} \in \mathbb{R}^{N \times N}$ and a measurement matrix $\mathbf{\Phi}_j \in \mathbb{R}^{M \times N}$ such that as before $\mathbf{y}_j = \mathbf{\Phi}_j \mathbf{x}_j$ with $M_j \ll N$ and $\mathbf{x}_j = \mathbf{\Psi} \boldsymbol{\alpha}_j$.

In this case the problem (5.3) can be rewritten as:

$$\min_{\tilde{\boldsymbol{\alpha}}} \|\tilde{\boldsymbol{\alpha}}\|_{w,2,1} := \sum_{i=1}^{n} w_i \|\tilde{\boldsymbol{\alpha}}_i\|_2 \qquad \text{s.t.} \qquad \tilde{\mathbf{\Theta}} \tilde{\boldsymbol{\alpha}} = \tilde{\mathbf{Y}} \tag{5.4}$$

where $\tilde{\mathbf{Y}} = \left[\mathbf{y}_1^T \mathbf{y}_2^T \dots \mathbf{y}_J^T\right]$, $\tilde{\boldsymbol{\alpha}} = \left[\boldsymbol{\alpha}_1^T \boldsymbol{\alpha}_2^T \dots \boldsymbol{\alpha}_J^T\right]$, w_i is the weight and $\tilde{\mathbf{\Theta}} \in \mathbb{R}^{JM \times JN}$ is a matrix having on the diagonal matrices $\mathbf{\Theta}_j = \mathbf{\Phi}_j \mathbf{\Psi}$ for $j \in \{1, 2, \dots, J\}$.

We can refer to this technique with the name Group-Sparsity CS (GS-CS).

5.2 Latent variables and tensor factorization

This technique is based on the well-known variable based factorization already used in several application domains [Koren et al., 2009]. The main idea behind this framework is to model the large number of observed variables (the observed data) in terms of a much smaller number of unobserved variables (the latent variables). The latent variables are learned from the observed data and are used to estimate the missing samples, modeling complex interactions between the observed variables through simple interactions between the latent variables.

Let say we are able to collect some multivariate data by a heterogeneous WSN, we can organized this data in a 3-dimensional matrix. Each of the three dimensions corresponds to a different variate of a particular measurement. In our case the three dimensions are: time, spatial data evolution and kind of sensor sampled. Once the data is organized in this fashion we can associate a low-dimensional latent variable with each unique location, time slice and sensor type. It turns out that a particular observation can be seen as a noisy combination of the associated latent variables. The aim is to obtain a good set of latent variables that can represent the observed data.

Mathematically we can define with $n \in \{1, 2, \ldots, N\}$ the time instance, with $s \in \{1, 2, \ldots, S\}$ the sensor type and with $j \in \{1, 2, \ldots, J\}$ the node location. In vectors we have $\mathbf{a}_n$, $\mathbf{b}_s$ and $\mathbf{c}_j$ respectively. We have not direct access to these vectors, called the latent factors, but these are assumed to control the location, time and sensor-specific interactions present in the observed data.

Given S different sensor types collected by J nodes at N different time instances, we can think of this data as a single $[N \times S \times J]$ tensor $\mathcal{X}$. $\mathcal{X}$ can also contain missing entries due to problem in transmission or to save on sampling power.

We assume that each reading x_{nsj} is a noisy realization of the underlying true reading that is obtained by the interaction of the time specific latent variable $\mathbf{a}_n$, with the sensor specific latent variable $\mathbf{b}_s$ and with the location specific variable $\mathbf{c}_j$. That is,

$$x_{nsj} = \sum_{k=1}^{K} a_{nk} b_{sk} c_{jk} + \varepsilon, \tag{5.5}$$

where ε is modeled as independent zero-mean Gaussian noise ($\varepsilon \sim \mathcal{N}(0, \sigma^2)$). Once one has x_{nsj} the goal is to find the most predictive set of vectors $\mathbf{a}_n$, $\mathbf{b}_s$ and $\mathbf{c}_j$ for all $n = 1, \ldots, N$, $s = 1, \ldots, S$, $j = 1, \ldots, J$.

Using the formulation in Equation (5.5) the entire model is represented by just $[K \cdot (N + S + J)]$ modeling parameters. The K parameter defines the trade-off between quality of the reconstruction and computational power. Moreover while a large K increases the number of modeling parameters and thus can help model the observed data exactly, on the other hand this lacks the capability on predicting unobserved / missing data due to over-fitting. On the contrary a small K escapes the over-fitting problem, but the corresponding model lacks sufficient richness to capture salient data trends.

5.2.1 Learning process

By [Kolda and Bader, 2009] we know that it is possible to find the optimal set of K-dimensional latent variables by factorizing the given tensor into three matrices of rank at most K. This can be done, assuming that all the data is known, by employing the CanDecomp / ParaFac (CP) tensor factorization. Using this technique it is possible to decompose a generic third order tensor in three matrix factors A, B and C.

The resolution of the problem to find the suitable A, B and C matrices is solved by alternating least square approach (ALS) [Acar and Yener, 2009] which iteratively optimizes for one matrix factor at time while keeping the other two fixed.

If this technique is suitable and already explored when working with dense matrices, in the case of WSNs sensor nodes can periodically go off-line due to duty-cycling or run out of energy, creating holes in the matrices filled with zeros. Therefore it is needed to extend the basic model to deal with data missing from multiple sensors or nodes. This could be done explicitly exploiting spatial, temporal and sensor specific information from neighboring observations by learning and enforcing the corresponding correlations.

5.2.2 Exploiting correlations in LV

We explicitly model spatial, temporal and sensor-specific trends within each of our latent variables $\mathbf{a}_n$, $\mathbf{b}_s$ and $\mathbf{c}_j$. Such trends ensure that the latent variables $\mathbf{a}_n$ and $\mathbf{a}_{n'}$ (respectively $\mathbf{b}_s$ and $\mathbf{b}_{s'}$, and $\mathbf{c}_j$ and $\mathbf{c}_{j'}$) take similar values when times

n and n' are “similar" (respectively sensors types s and s', and locations j and j').

The similarity constraints are modeled in the same way for all the three sets of latent variables, and here we illustrate the case for the $\mathbf{a}_n$'s.

Since each $\mathbf{a}_n$ is a K-dimensional variable, let a_n^k denote its k^{th} coordinate. We model a_n^k (independently for each coordinate k) as

$$\begin{aligned} a_:^k &= \mu_a^k + \alpha_:^k \\ \alpha_:^k &\sim \mathcal{N}(0, \Sigma_a). \end{aligned} \tag{5.6}$$

Here $a_:^k$ represents the collection of all $\mathbf{a}_n$'s (across $n = 1, \ldots, N$) in the k^{th} coordinate and μ_a represents their mean value. The distributional constraint over $\alpha_:^k$ (as $\mathcal{N}(0, \Sigma_a)$) enforces the similarity constraints via the $N \times N$ covariance matrix Σ_a. By changing the n, n' entry of Σ_a we can encourage / discourage the corresponding $\mathbf{a}_n$ and $\mathbf{a}_{n'}$ to take similar values.

To get the right similarity constraints Σ_a, Σ_b and Σ_c (for latent variables $\mathbf{a}_n$, $\mathbf{b}_s$ and $\mathbf{c}_j$), it is possible to compute the empirical correlations from the data. That is, for spatial similarity constraints the averaged pairwise Pearson correlation coefficient between data from different pairs of locations (across sensors and times) is used. The same is done to approximate inter-sensor and temporal similarities.

5.2.3 Parameter learning

The underlying latent variables are estimated in a probabilistic framework using a *maximum a posteriori* (MAP) estimate.

In particular, let θ denote all the model parameters (i.e. $\theta = \{\{\mathbf{a}_n\}, \{\mathbf{b}_s\}, \{\mathbf{c}_j\}, \sigma\}$), then the optimum choice of parameters θ_{MAP} given the data $\mathcal{X}$ is obtained by:

$$\theta_{MAP}(\mathcal{X}) := \arg\max_{\theta} \underbrace{p(\mathcal{X} \mid \theta)}_{\text{likelihood}} \underbrace{p(\theta)}_{\text{prior}} =$$
$$\arg\max_{\theta} \sum_{n,s,j \in \text{observed}} \log p(x_{nsj} \mid \mathbf{a}_n, \mathbf{b}_s, \mathbf{c}_j, \sigma) +$$
$$\sum_{k=1}^{K} \log p(a^k_{:}) + \sum_{k=1}^{K} \log p(b^k_{:}) + \sum_{k=1}^{K} \log p(c^k_{:}).$$

The first term (the likelihood) takes the form of Equation (5.5), and the other terms represent the priors for each latent variable and each one of them takes the form of Equation (5.6). We take a uniform prior over σ, the standard deviation of the residuals in Equation 5.5 so it is not explicitly shown in the equation.

This optimization does not have a closed form thus an alternating hill-climb approach is used by optimizing the value of one variable while keeping all others fixed to get a good solution.

5.3 Comparison between GC-CS and LV

Before directly comparing the two frameworks it is needed to define the models for hardware, network and power consumption used in the simulations.

Hardware As already seen in the previous chapters, the hardware taken as reference for simulations and tests is the wireless node by ST Microelectronics STM32W108. The sensors considered in this case are: a Sensirion SHT21 for sensing of temperature and humidity and a BH1715 light sensor. Timing in performing the operations used in the power model are obtained either using the values reported in the data-sheet or measured using a GPIO trigger connected to an oscilloscope.

Data for power consumption of the various subsystems are not reported for lack of space but the reader can refer to the data-sheets of the components for further reference.

Network model Data used for the simulations is taken from the 3ENCULT [3ENCULT, 2012] that is a structural health monitoring deployment composed by 23 low-power sensor nodes spread across the three floors of the historic building *Palazzina della Viola* at the University of Bologna, Italy. The topology considered is a star network in which data is relayed toward a central sink that is in charge to collect and elaborate data.

In a star network we can consider that each node samples the signals for a period of time T, called acquisition period, ideally gathering $N = T/f_s$ number of samples at a f_s sampling frequency, before sending the data towards the collecting point. If each node adopts a sub-sampling policy with an under-sampling ratio ρ then the number of samples actually gathered by the node is $M = \rho N$.

The under-sampling pattern is locally generated by each node using its own id and the timestamp as seed for randomization. In the random sampling pattern the inter-measurements intervals are always multiple of the minimum sampling period $T_k = T/N$.

Power model The architecture power model is derived from the power model introduced in Section 4.4.2. Considering the characteristics of the network model and the power consumption model already introduced we can define the average energy consumption in each period of duration T_k, when a sub-sampling factor ρ, is:

$$E_k = \rho \left(E_{\text{setup}} + E_{\text{sampl}} + E_{\text{store}}\right) + E_{\text{sleep}} + N^{-1} \left(E_{\text{nv}} + E_{\text{send}}\right) \tag{5.7}$$

where E_{sleep} is the energy spent in sleep mode, E_{setup} is the energy used for waking up and setting up the device, E_{sample} is the energy for sampling each sensors, E_{send} is the energy used to send the acquired data, E_{store} is the energy to store the acquired sample in non volatile memory and E_{nv} is the energy spent during the recovery of the data from non volatile memory.

5.3.1 Reconstruction quality / lifetime trade-off analysis

The data considered to investigate the performance of GC-CS and LV is composed by data gathered by temperature, humidity and light sensors.

The compression phase is the same for both frameworks: each node samples the signals of interest gathering a sub-set M of the needed samples ($M = \rho N$), with $0 < \rho < 1$. After the acquisition period $T = NT_k$ the gathered data is sent to the collecting sink through the network. The sampling time T_k in the following simulations is set to $600s$ and the results are averaged over 100 trials. Each trial is characterized by a different sampling pattern and a different considered portion of the signal. The reconstruction phase is fairly different and determines the recovery quality of the original signal. For CS the DCT matrix is used as sparsifying matrix that is already been demonstrated being a good sparsifying matrix for natural signals [Caione et al., 2012].

In the first simulation we evaluate the reconstruction quality without exploiting any correlations among signals or sensors but just averaging the reconstruction quality over all the reconstructed signals using a signal length of $N = 512$. For the latent variables approach here we used the standard CP tensor factorization technique, without the contribution of any correlations in the data. The comparison is carried evaluating the signal to noise ratio (SNR) defined as:

$$\mathrm{SNR}_{\mathrm{dB}} = 20 \cdot \log_{10} \frac{\|\mathbf{x}\|_2}{\|\mathbf{x} - \hat{\mathbf{x}}\|_2} \tag{5.8}$$

where $\mathbf{x}$ is the original signal and $\hat{\mathbf{x}}$ is its recovered version. We show the average SNR across all the network nodes.

From the plot, shown in Figure 5.1, we can infer how different the performance is for the two techniques: while the reconstruction performance for LV is pretty stable varying the sub-sampling factor ρ, CS is much more affected by the compression factor. Recovery with CS achieves a better reconstruction almost for every sub-sampling factor with respect to the latent variable based technique.

Moreover from the plot we can infer how the recovery of light signals is much more difficult then the two other signals. This is due to the nature of the light signal that is recorded inside the building. While for temperature and humidity

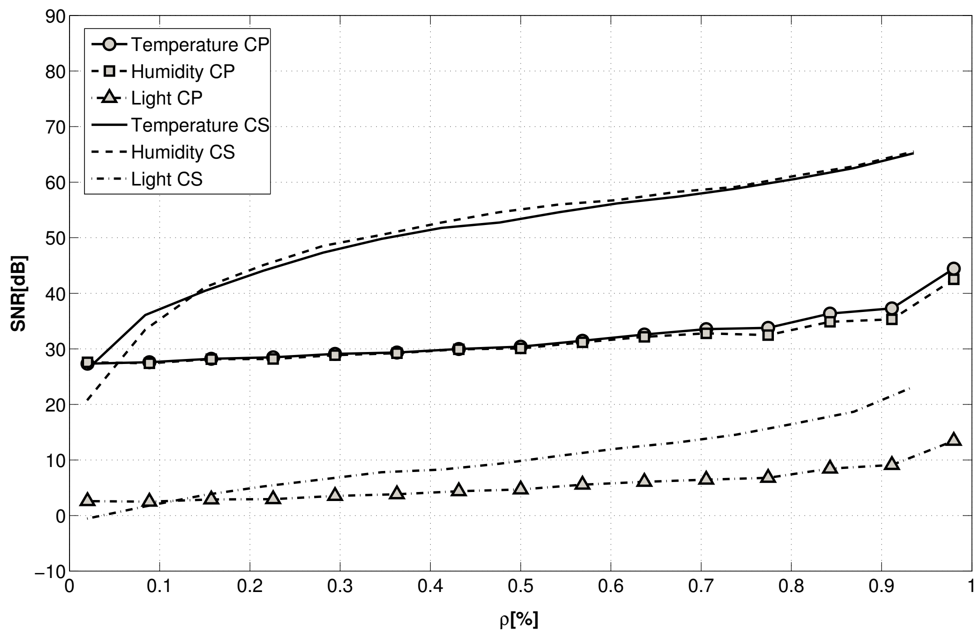


Figure 5.1: Recovery comparison between CS and latent variable (LV) method when reconstructing the original signals from sub-sampled version averaging the reconstruction quality over all the nodes ($N = 512$)

the gathered signals are continuous signals and smoothly affected by the human presence, the light signal is highly irregular and highly influenced by the artificial lighting in the single rooms. Moreover, some of the nodes are placed in the basement where the light level is under the noise threshold of the light sensors, providing extremely noisy data.

Once investigated the reconstruction of the single signals, it is possible to evaluate the reconstruction when correlations among sensors and nodes is exploited. We performed the recovery against the same dataset using for CS the group sparse optimization (GS-CS) exploiting the joint sparsity of the signals and for LV the maximum a posteriori optimization (LV-MAP).

Results are reported in Figure 5.2. While the performance for CS remains almost the same, the LV-MAP method guarantees a significant improvement in reconstruction resulting better than CS for small values of ρ, especially in relation to humidity and temperature signals.

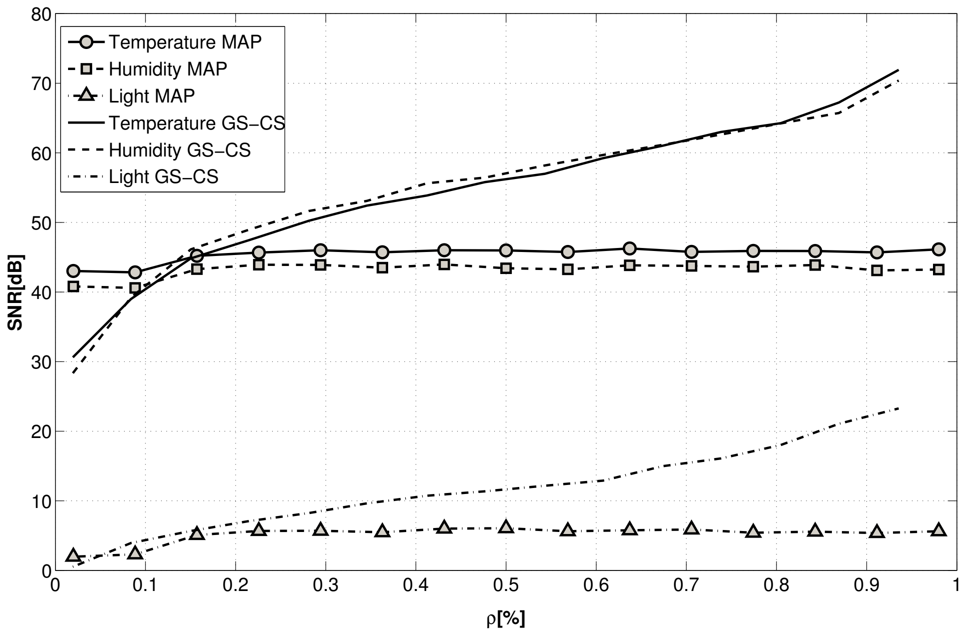


Figure 5.2: Recovery comparison between GC-CS and MAP. The recovery is obtained exploiting the correlations among sensors and nodes. ($N = 512$)

The behavior of CS could be explained looking at Figure 5.3. Here it's shown how the union over all signals of the K best DCT basis vectors per signal has a size definitely greater than K. Practically this means that GS-CS is able to exploit the inter-nodes correlation only at a small extent since the shared information among different nodes is limited and the recovery algorithm is not able to exploit this information to improve the recovery quality.

According to the model the simulations are performed with a sampling frequency $f_s = 1/600$[Hz], and since the length of the data is $N = 512$ this brings in a delay in the data delivery towards the data collector of 3.5 days. Thus the size of the recovered signal spawns across 3.5 days. In practice having high values of N means that we have to wait a longer time to proceed with data recovery. Therefore we want now to investigate how the length of the block of data gathered by sensors affects the two frameworks and whether a correlation between recovery performance and the N parameter does exist for GS-CS and LV-MAP.

In Figure 5.4a the results for GS-CS when N is changed are presented. From

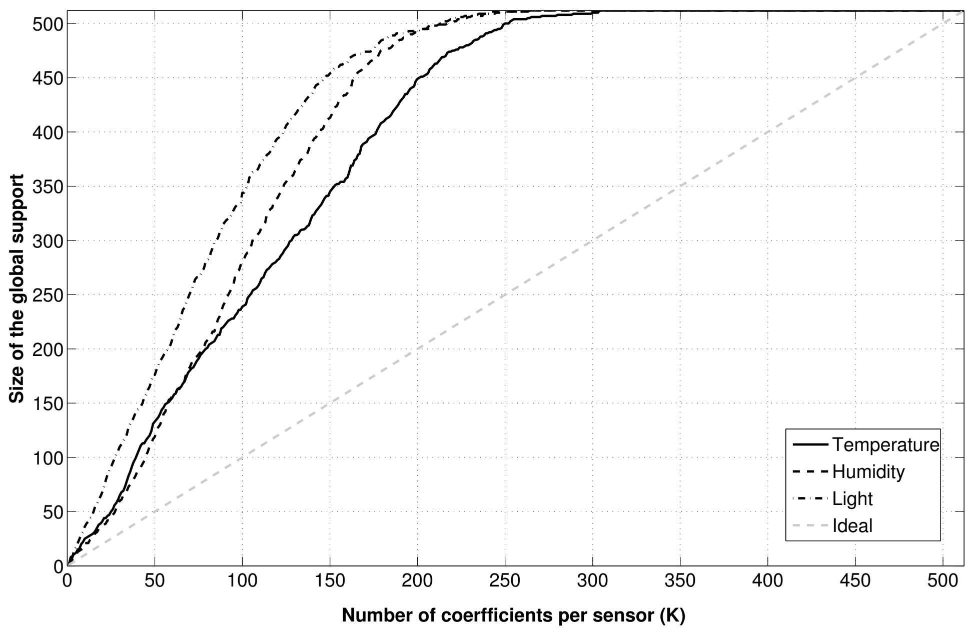
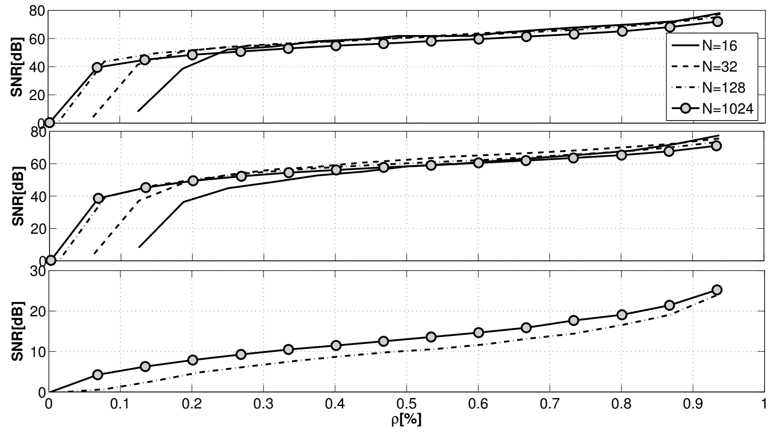


Figure 5.3: Number of DCT coefficients necessary to include the K largest coefficients for each signal. ($N = 512$)

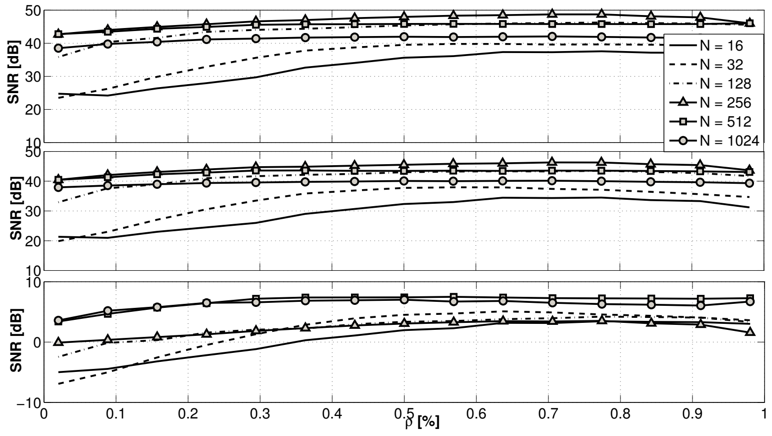
the plot we can infer how the length of the signal N does not greatly affect the reconstruction quality for all the signals taken into consideration. Rather we can see how the influence of the parameter N (and then of the delay in the data collection) is only visible for small values of the sub-sampling factor ρ. Differently from temperature and humidity, the light signal presents a peculiar behavior showing an increased reconstruction quality with the increasing in the number of acquired samples.

The same results for the LV-MAP approach are presented in Figure 5.4b. The difference in the reconstruction error for the various values of N is more evident than in the GS-CS case. With small values of N we registered difficulties to reconstruct the desired signals. The best recovery performance is achieved when considering 256-512 samples at a time, identifying the optimal trade-off between delay and reconstruction accuracy, since larger blocks of data present again a loss of accuracy.

For a delay smaller than $N = 128$ the reconstruction of the light signal is not



(a) GS-CS

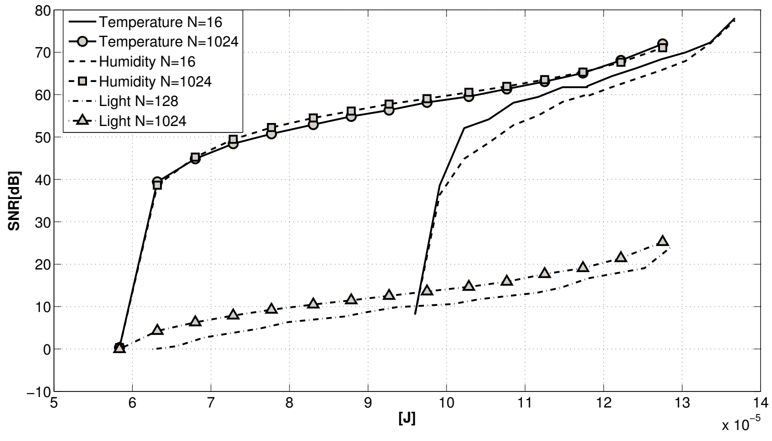


(b) LV

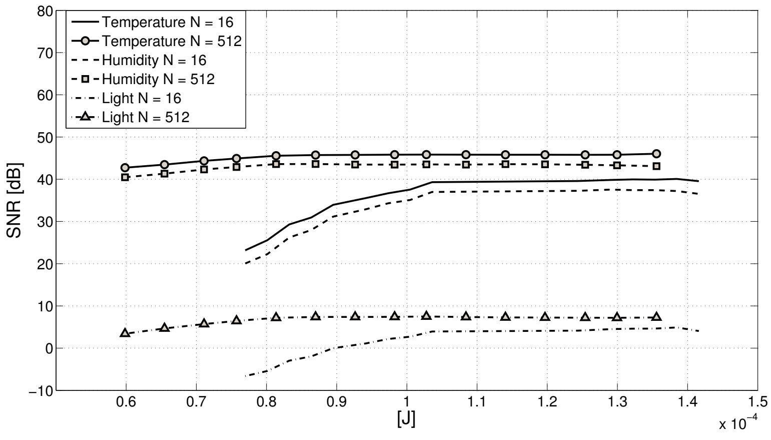
Figure 5.4: From top to bottom: temperature, humidity, light. GS-CS and LV reconstruction quality for the different signals varying the sub-sampling factor ρ and using the signal length N as parameter.)

feasible in both cases, since the majority of the samples gathered are zeros due to the lack of light at night.

Having evaluated the recovery performance and the influence of the gathering delay on reconstruction, it is interesting to investigate the power consumption involved with compression according to the power model. In Figure 5.5 we report the reconstruction quality against the energy consumption for one acquisition cycle. The plot clearly shows how a trade-off between energy consumption and transmission delay does exist in GS-CS case (Figure5.5a). Higher values of N,



(a) GS-CS



(b) LV

Figure 5.5: Reconstruction quality vs. averaged per cycle energy consumption varying the parameter N for the signals of interest for the considered techniques: (a) GS-CS and (b) LV

thus higher delays in transmission, are able to guarantee a better reconstruction quality with definitely less energy than the $N = 16$ case (all the other cases are not considered in the plot since they are between these two boundaries). The light signal is a special case but we can draw the same conclusions as before. The LV-MAP case (Figure5.5b) presents a similar behavior, but with a less accentuated increase in energy efficiency corresponding to the increase in data size N. When comparing the two graphs, we can observe how the GS-CS case exhibits a slightly higher energy efficiency, allowing a higher reconstruction when considering the same energy consumptions as the LV-MAP case. Only for extremely sub-sampled signals the LV-MAP approach results better, having a major benefit from the explicit correlation models incorporated in the data reconstruction. In both cases, we are able to obtain a better accuracy (or the same reconstruction quality with less energy) if we are willing to wait for an higher number of gathered samples before proceeding with the reconstruction.

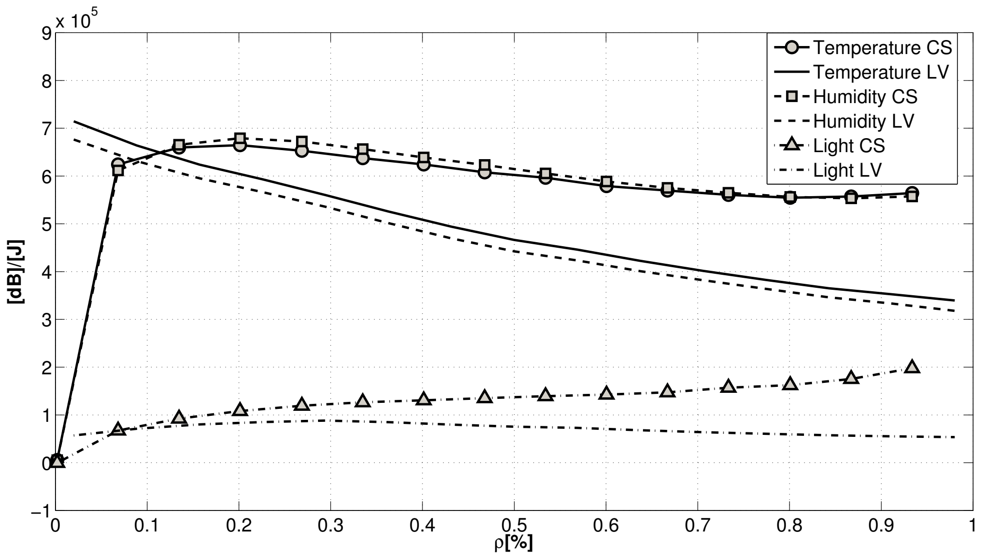


Figure 5.6: Ratio between the recovery quality and energy spent in compression varying the sub-sampling factor ρ for the two approaches

The same conclusions are explicated in Figure 5.6 where the ratio between the reconstruction quality and the consumed energy is plotted against the sub-sampling factor ρ. Here we can see a direct comparison of the two techniques for the case with the best reconstruction performance ($N = 512$). Again we can see how the GS-CS case has a higher ratio when compared to the LV-MAP case, for almost all the sub-sampling policies. Only when dealing with a really small amount of sampled data (20%), the LV-MAP case shows a better performance. This result can be a guide for WSN developers, suggesting the adoption of the LV method only when a really aggressive power saving technique is needed.

Chapter 6

Conclusions

This thesis has presented several data processing and compression techniques capable of addressing the strict requirements of wireless sensor networks.

In the introduction we have given a general overview of the several kind of sensor networks it is possible to deal with, stressing for each one issues and strength points. From here the energy problem has been introduced, dividing the different approaches according to the different subsystem they try to optimize. After having introduced the major technologies for batteries and energy supplies, the focus of the research has been shifted toward the radio subsystem. Since it is well-known that the energy expenditure for communications is usually the largest contribution in the power consumption, several low-power MAC protocols have been addressed and one particular case study, the Conservative Power Scheduling, has been described in detail. Finally the processing subsystem issues have been introduced. We are strongly convinced that a future trend in the WSNs field will be moving from the 8 and 16-bit architectures toward the more powerful 32-bit architectures thus an in-depth analysis of the usage of this type of micro-controllers has been done. A special case-study is also given at the end of the chapter describing an ultra-low power device for aircraft structural health monitoring.

In Chapter 3, data reduction techniques for data managing in WSNs have been introduced. To manage the complexity brought by these techniques, a quick overview of the most common middlewares for WSNs is given, describing also in detail SPINE2, a framework for data processing in the node environment. The remaining part of the chapter has been focused on the in-network aggregation techniques, used to reduce data sent by the network nodes trying to prolong the network lifetime as long as possible. Among the several techniques, the approach that seems the most promising is the Compressive Sensing (CS). The theory behind this framework has been given in the same chapter and a practical implementation of the algorithm is compared with a simpler aggregation scheme, deriving a mixed algorithm able to successfully reduce the power consumption.

Still focusing on CS, the step in Chapter 4 has been to move from compression implemented on single nodes to CS for signal ensembles. The rationale behind this is to try to exploit the correlations among sensors and nodes to improve compression and reconstruction quality. The two main techniques for signal ensembles, Distributed CS (DCS) and Kronecker CS (KCS), have been introduced and compared against a common set of data gathered by a real deployment. The best trade-off between reconstruction quality and power consumption has been obtained by DCS and the JSM-2 sparsity model has resulted to be the most suitable to describe the sparsity that characterizes the natural signals of interest for WSNs applications. Finally the usage of CS has been investigated when the signal of interest is sampled at a Sub-Nyquist rate, evaluating the reconstruction performance.

In the last chapter of the thesis the group sparsity CS (GS-CS) has been compared to another well-known technique for the reconstruction of signal from an highly sub-sampled version derived from the latent variables and tensor factorization technique. These two frameworks are compared again against a real data-set and an insightful analysis of the trade-off between reconstruction quality and lifetime is given.

Briefly:

- Compressive Sensing is a powerful tool for energy reduction in WSNs
- Temporal and spatial correlations among sensors and nodes can be exploited to improve reconstruction (DCS and KCS)
- CS is powerful to compress data and to reduce the amount of data to transmit but the energy consumption for compression cannot be considered negligible especially when considering resource-constrained nodes

- Several kind of compression matrices have been investigated in relation to recovery performance and energy spent for compression
- Several techniques and their recovery performance have been investigated always trying to achieve the better trade-off between reconstruction quality and energy consumption
- Further energy reduction can be achieved using a peculiar form of CS, low-rate CS, opportunely sampling the signal of interest at sub-Nyquist rate
- CS is able to obtain very good results also when compared to other techniques for data recovery from highly incomplete version of the original signal

References

3ENCULT. http://www.3encult.eu/en/casestudies/default.html, 2012.

E. Acar and B. Yener. Unsupervised multiway data analysis: A literature survey. *Knowledge and Data Engineering, IEEE Transactions on*, 21(1):6 –20, jan. 2009. ISSN 1041-4347. doi: 10.1109/TKDE.2008.112.

I.F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *Communications Magazine, IEEE*, 40(8):102 – 114, aug 2002. ISSN 0163-6804. doi: 10.1109/MCOM.2002.1024422.

J.N. Al-Karaki and A.E. Kamal. Routing techniques in wireless sensor networks: a survey. *Wireless Communications, IEEE*, 11(6):6 – 28, dec. 2004. ISSN 1536-1284. doi: 10.1109/MWC.2004.1368893.

N.V. Aravind, K. Abhinandan, V.V. Acharya, and D.S. Sumam. Comparison of omp and somp in the reconstruction of compressively sensed hyperspectral images. In *Communications and Signal Processing (ICCSP), 2011 International Conference on*, pages 188 –192, feb. 2011. doi: 10.1109/ICCSP.2011.5739298.

Khaled A. Arisha. Energy-aware tdma-based mac for sensor networks. In *IEEE Workshop on Integrated Management of Power Aware Communications, Computing and Networking (IMPACCT 2002*, 2002.

M. Bahrepour, N. Meratnia, and P.J.M. Havinga. Sensor fusion-based event detection in wireless sensor networks. In *Mobile and Ubiquitous Systems: Networking Services, MobiQuitous, 2009. MobiQuitous '09. 6th Annual International*, pages 1 –8, july 2009. doi: 10.4108/ICST.MOBIQUITOUS2009.7056.

M. Bal, Weiming Shen, and H. Ghenniwa. Collaborative signal and information processing in wireless sensor networks: A review. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 3151 –3156, oct. 2009. doi: 10.1109/ICSMC.2009.5346152.

B. Banitalebi, D. Gordon, S. Sigg, T. Miyaki, and M. Beigl. Collaborative channel equalization: Analysis and performance evaluation of distributed aggregation algorithms in wsns. In *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*, pages 450 –459, oct. 2011. doi: 10.1109/MASS.2011.51.

Dror Baron, Michael B. Wakin, Marco F. Duarte, Shriram Sarvotham, and Richard G. Baraniuk. Distributed compressed sensing. Technical report, 2005.

Shah Bhatti, James Carlson, Hui Dai, Jing Deng, Jeff Rose, Anmol Sheth, Brian Shucker, Charles Gruenwald, Adam Torgerson, and Richard Han. Mantis os: an embedded multithreaded operating system for wireless micro sensor platforms. *Mob. Netw. Appl.*, 10(4):563–579, August 2005. ISSN 1383-469X. URL http://dl.acm.org/citation.cfm?id=1160162.1160178.

I. Bovio and L. Lecce. Health monitoring: new techniques based on vibrations measurements and identification algorithms. In *Aerospace Conference, 2005 IEEE*, pages 3601 –3609, march 2005. doi: 10.1109/AERO.2005.1559665.

D. Brunelli, D. Dondi, A. Bertacchini, L. Larcher, P. Pavan, and L. Benini. Photovoltaic scavenging systems: Modeling and optimization. *Microelectronics Journal*, 40(9):1337 – 1344, 2009. ISSN 0026-2692. doi: 10.1016/j.mejo.2008.08.013. URL http://www.sciencedirect.com/science/article/pii/S0026269208004631.

M.A. Caceres, F. Sottile, and M.A. Spirito. Adaptive location tracking by kalman filter in wireless sensor networks. In *Wireless and Mobile Computing, Networking and Communications, 2009. WIMOB 2009. IEEE International Conference on*, pages 123 –128, oct. 2009. doi: 10.1109/WiMob.2009.30.

C. Caione, D. Brunelli, and L. Benini. Distributed compressive sampling for lifetime optimization in dense wireless sensor networks. *Industrial Informatics,* *IEEE Transactions on*, 8(1):30 –40, feb. 2012. ISSN 1551-3203. doi: 10.1109/TII.2011.2173500.

E.J. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489 – 509, feb. 2006. ISSN 0018-9448. doi: 10.1109/TIT.2005.862083.

Qing Cao, T. Abdelzaher, J. Stankovic, and Tian He. The liteos operating system: Towards unix-like abstractions for wireless sensor networks. In *Information Processing in Sensor Networks, 2008. IPSN '08. International Conference on*, pages 233 –244, april 2008. doi: 10.1109/IPSN.2008.54.

R. Chakravorty. A programmable service architecture for mobile medical care. In *Pervasive Computing and Communications Workshops, 2006. PerCom Workshops 2006. Fourth Annual IEEE International Conference on*, pages 5 pp. –536, march 2006. doi: 10.1109/PERCOMW.2006.11.

S. Chatterjea, L.F.W. van Hoesel, and P.J.M. Havinga. Ai-lmac: an adaptive, information-centric and lightweight mac protocol for wireless sensor networks. In *Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004. Proceedings of the 2004*, pages 381 – 388, dec. 2004. doi: 10.1109/ISSNIP.2004.1417492.

Chien-Hua Chen and Kai-Ten Feng. Asynchronous location tracking algorithms for distributed power-saving wireless sensor networks. In *Wireless Communications and Networking Conference, 2009. WCNC 2009. IEEE*, pages 1 –6, april 2009. doi: 10.1109/WCNC.2009.4917605.

J. Chen, Karric Kwong, D. Chang, J. Luk, and R. Bajcsy. Wearable sensors for reliable fall detection. In *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*, pages 3551 –3554, jan. 2005. doi: 10.1109/IEMBS.2005.1617246.

Scott Shaobing Chen, David L. Donoho, Michael, and A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20: 33–61, 1998.

Wei Chen, M.R.D. Rodrigues, and I.J. Wassell. Distributed compressive sensing reconstruction via common support discovery. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1 –5, june 2011. doi: 10.1109/icc.2011.5962798.

Chi-Tsun Cheng, C.K. Tse, and F.C.M. Lau. A delay-aware data collection network structure for wireless sensor networks. *Sensors Journal, IEEE*, 11(3):699 –710, march 2011. ISSN 1530-437X. doi: 10.1109/JSEN.2010.2063020.

S. Cho, K. Kanuri, Jin-Woong Cho, Jang-Yeon Lee, and Sun-Do June. Dynamic energy efficient tdma-based mac protocol forwireless sensor networks. In *Autonomic and Autonomous Systems and International Conference on Networking and Services, 2005. ICAS-ICNS 2005. Joint International Conference on*, page 48, oct. 2005. doi: 10.1109/ICAS-ICNS.2005.43.

F. Chraim and S. Karaki. Fuel cell applications in wireless sensor networks. In *Instrumentation and Measurement Technology Conference (I2MTC), 2010 IEEE*, pages 1320 –1325, may 2010. doi: 10.1109/IMTC.2010.5488214.

A. Conti, D. Dardari, and R. Verdone. Collaborative signal processing for energy-efficient self-organizing wireless sensor network. In *Wireless Ad-Hoc Networks, 2004 International Workshop on*, pages 99 – 104, may-3 june 2004. doi: 10.1109/IWWAN.2004.1525550.

R. Cristescu, B. Beferull-Lozano, and M. Vetterli. On network correlated data gathering. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2571 – 2582 vol.4, march 2004. doi: 10.1109/INFCOM.2004.1354677.

M. Crovella and E. Kolaczyk. Graph wavelets for spatial traffic analysis. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1848 – 1857 vol.3, march-3 april 2003. doi: 10.1109/INFCOM.2003.1209207.

F. Cuomo, S. Della Luna, U. Monaco, and F. Melodia. Routing in zigbee: Benefits from exploiting the ieee 802.15.4 association tree. In *Communications, 2007.*

ICC '07. IEEE International Conference on, pages 3271 –3276, june 2007. doi: 10.1109/ICC.2007.542.

I. Demirkol, C. Ersoy, and F. Alagoz. Mac protocols for wireless sensor networks: a survey. *Communications Magazine, IEEE*, 44(4):115 – 121, april 2006. ISSN 0163-6804. doi: 10.1109/MCOM.2006.1632658.

Q. Dong. Maximizing system lifetime in wireless sensor networks. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 13 – 19, april 2005. doi: 10.1109/IPSN.2005.1440886.

D.L. Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289 –1306, april 2006. ISSN 0018-9448. doi: 10.1109/TIT.2006.871582.

D.L. Donoho, M. Elad, and V.N. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *Information Theory, IEEE Transactions on*, 52(1):6 – 18, jan. 2006. ISSN 0018-9448. doi: 10.1109/TIT.2005.860430.

M.F. Duarte and R.G. Baraniuk. Kronecker compressive sensing. *Image Processing, IEEE Transactions on*, 21(2):494 –504, feb. 2012. ISSN 1057-7149. doi: 10.1109/TIP.2011.2165289.

A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455 – 462, nov. 2004. doi: 10.1109/LCN.2004.38.

A. El-Hoiydi and J.-D. Decotignie. Wisemac: an ultra low power mac protocol for the downlink of infrastructure wireless sensor networks. In *Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium on*, volume 1, pages 244 – 251 Vol.1, june-1 july 2004. doi: 10.1109/ISCC.2004.1358412.

A. Eswaran, A. Rowe, and R. Rajkumar. Nano-rk: an energy-aware resource-centric rtos for sensor networks. In *Real-Time Systems Symposium, 2005. RTSS*

2005. 26th IEEE International, pages 10 pp. –265, dec. 2005. doi: 10.1109/RTSS.2005.30.

K. Fakih, J.F. Diouris, and G. Andrieux. Bmac: Beamformed mac protocol with channel tracker in manet using smart antennas. In *Wireless Technology, 2006. The 9th European Conference on*, pages 185 –188, sept. 2006. doi: 10.1109/ECWT.2006.280466.

T. Farjaudon and J. Hascoet. T-mac. a new member of the mac family for eng links. In *Broadcasting Convention, 1988. IBC 1988., International*, pages 98 –100, sep 1988.

E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi. In-network aggregation techniques for wireless sensor networks: a survey. *Wireless Communications, IEEE*, 14(2): 70 –87, april 2007. ISSN 1536-1284. doi: 10.1109/MWC.2007.358967.

G. Fortino, A. Guerrieri, F. Bellifemine, and R. Giannantonio. Platform-independent development of collaborative wireless body sensor network applications: Spine2. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 3144 –3150, oct. 2009a. doi: 10.1109/ICSMC.2009.5346155.

G. Fortino, A. Guerrieri, F.L. Bellifemine, and R. Giannantonio. Spine2: developing bsn applications on heterogeneous sensor nodes. In *Industrial Embedded Systems, 2009. SIES '09. IEEE International Symposium on*, pages 128 –131, july 2009b. doi: 10.1109/SIES.2009.5196205.

S. Gajurel and M. Heiferling. Swarm intelligent routing solution for wireless sensor networks. In *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, pages 707 –714, oct. 2010. doi: 10.1109/LCN.2010.5735797.

Deepak Ganesan, Ben Greenstein, Denis Perelyubskiy, Deborah Estrin, and John Heidemann. An evaluation of multi-resolution storage for sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, SenSys '03, pages 89–102, New York, NY, USA, 2003. ACM. ISBN 1-58113-707-9. doi: 10.1145/958491.958502. URL http://doi.acm.org/10.1145/958491.958502.

M. Gastpar, P.L. Dragotti, and M. Vetterli. The distributed karhunen-loeve transform. In *Multimedia Signal Processing, 2002 IEEE Workshop on*, pages 57 – 60, dec. 2002. doi: 10.1109/MMSP.2002.1203247.

A. Gilbert and P. Indyk. Sparse recovery using sparse matrices. *Proceedings of the IEEE*, 98(6):937 –947, june 2010. ISSN 0018-9219. doi: 10.1109/JPROC.2010.2045092.

D. Goyal and M.R. Tripathy. Routing protocols in wireless sensor networks: A survey. In *Advanced Computing Communication Technologies (ACCT), 2012 Second International Conference on*, pages 474 –480, jan. 2012. doi: 10.1109/ACCT.2012.98.

J. Haupt, W.U. Bajwa, M. Rabbat, and R. Nowak. Compressed sensing for networked data. *Signal Processing Magazine, IEEE*, 25(2):92 –101, march 2008. ISSN 1053-5888. doi: 10.1109/MSP.2007.914732.

S. Hayat, N. Javaid, Z.A. Khan, A. Shareef, A. Mahmood, and S.H. Bouk. Energy efficient mac protocols. In *High Performance Computing and Communication 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on*, pages 1185 – 1192, june 2012. doi: 10.1109/HPCC.2012.174.

J.L. Hill and D.E. Culler. Mica: a wireless platform for deeply embedded networks. *Micro, IEEE*, 22(6):12 – 24, nov/dec 2002. ISSN 0272-1732. doi: 10.1109/MM.2002.1134340.

B. Hohlt, L. Doherty, and E. Brewer. Flexible power scheduling for sensor networks. In *Information Processing in Sensor Networks, 2004. IPSN 2004. Third International Symposium on*, pages 205 – 214, april 2004. doi: 10.1109/IPSN.2004.1307340.

Chih-Lin I and G.P. Pollini. The tree-search resource auction multiple access (trama) protocol for wireless personal communications. In *Vehicular Technology Conference, 1994 IEEE 44th*, pages 1170 –1174 vol.2, jun 1994. doi: 10.1109/VETEC.1994.345276.

Sameer Iyengar, Filippo Tempia Bonda, Raffaele Gravina, Antonio Guerrieri, Giancarlo Fortino, and Alberto Sangiovanni-Vincentelli. A framework for creating healthcare monitoring applications using wireless body sensor networks. In *Proceedings of the ICST 3rd international conference on Body area networks*, BodyNets '08, pages 8:1–8:2, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). ISBN 978-963-9799-17-2. URL http://dl.acm.org/citation.cfm?id=1460257.1460268.

Georgios Kambourakis, Eleni Klaoudatou, and Stefanos Gritzalis. Securing medical sensor environments: The codeblue framework case. In *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*, pages 637 –643, april 2007. doi: 10.1109/ARES.2007.135.

M. Kassim, R. Ab Rahman, and R. Mustapha. Mobile ad hoc network (manet) routing protocols comparison for wireless sensor network. In *System Engineering and Technology (ICSET), 2011 IEEE International Conference on*, pages 148 –152, june 2011. doi: 10.1109/ICSEngT.2011.5993439.

O. Kebkal, M. Komar, and K. Kebkal. D-mac: Hybrid media access control for underwater acoustic sensor networks. In *Communications Workshops (ICC), 2010 IEEE International Conference on*, pages 1 –5, may 2010. doi: 10.1109/ICCW.2010.5503951.

Abtin Keshavarzian, Huang Lee, and Lakshmi Venkatraman. Wakeup scheduling in wireless sensor networks. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '06, pages 322–333, New York, NY, USA, 2006. ACM. ISBN 1-59593-368-9. doi: 10.1145/1132905.1132941. URL http://doi.acm.org/10.1145/1132905.1132941.

Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Rev.*, 51(3):455–500, August 2009. ISSN 0036-1445. doi: 10.1137/07070111X. URL http://dx.doi.org/10.1137/07070111X.

Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30 –37, aug. 2009. ISSN 0018-9162. doi: 10.1109/MC.2009.263.

P. Kuryloski, A. Giani, R. Giannantonio, K. Gilani, R. Gravina, V.-P. Seppa, E. Seto, V. Shia, C. Wang, P. Yan, A.Y. Yang, J. Hyttinen, S. Sastry, S. Wicker, and R. Bajcsy. Dexternet: An open platform for heterogeneous body sensor networks and its applications. In *Wearable and Implantable Body Sensor Networks, 2009. BSN 2009. Sixth International Workshop on*, pages 92 –97, june 2009. doi: 10.1109/BSN.2009.31.

T.P. Lambrou and C.G. Panayiotou. A survey on routing techniques supporting mobility in sensor networks. In *Mobile Ad-hoc and Sensor Networks, 2009. MSN '09. 5th International Conference on*, pages 78 –85, dec. 2009. doi: 10.1109/MSN.2009.37.

Ang-Hsi Lee, Ming-Hui Jing, and Cheng-Yan Kao. Lmac: An energy-latency trade-off mac protocol for wireless sensor networks. In *Computer Science and its Applications, 2008. CSA '08. International Symposium on*, pages 233 –238, oct. 2008. doi: 10.1109/CSA.2008.47.

P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. Tinyos: An operating system for sensor networks. In Werner Weber, JanM. Rabaey, and Emile Aarts, editors, *Ambient Intelligence*, pages 115–148. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-23867-6. doi: 10.1007/3-540-27139-2_7. URL http://dx.doi.org/10.1007/3-540-27139-2_7.

Rongxin Li, Chaomei Zheng, and Yunru Zhang. Study of power-aware routing protocal in wireless sensor networks. In *Electrical and Control Engineering (ICECE), 2011 International Conference on*, pages 3173 –3176, sept. 2011. doi: 10.1109/ICECENG.2011.6057824.

Zhaorui Liu, H.V. Zhao, and A.Y. Elezzabi. Block-based adaptive compressed sensing for video. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 1649 –1652, sept. 2010. doi: 10.1109/ICIP.2010.5654000.

G. Lu, B. Krishnamachari, and C.S. Raghavendra. An adaptive energy-efficient and low-latency mac for data gathering in wireless sensor networks. In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, page 224, april 2004. doi: 10.1109/IPDPS.2004.1303264.

Chong Luo, Feng Wu, Jun Sun, and Chang Wen Chen. Efficient measurement generation and pervasive sparsity for compressive data gathering. *Wireless Communications, IEEE Transactions on*, 9(12):3728 –3738, december 2010. ISSN 1536-1276. doi: 10.1109/TWC.2010.092810.100063.

H. Mamaghanian, N. Khaled, D. Atienza, and P. Vandergheynst. Compressed sensing for real-time energy-efficient ecg compression on wireless body sensor nodes. *Biomedical Engineering, IEEE Transactions on*, 58(9):2456 –2466, sept. 2011. ISSN 0018-9294. doi: 10.1109/TBME.2011.2156795.

J. McDowall. Conventional battery technologies-present and future. In *Power Engineering Society Summer Meeting, 2000. IEEE*, volume 3, pages 1538 –1540 vol. 3, 2000. doi: 10.1109/PESS.2000.868757.

Raul Morais, Samuel G. Matos, Miguel A. Fernandes, AntÃşnio L.G. Valente, Salviano F.S.P. Soares, P.J.S.G. Ferreira, and M.J.C.S. Reis. Sun, wind and water flow as energy supply for small stationary data acquisition platforms. *Computers and Electronics in Agriculture*, 64(2):120 – 132, 2008. ISSN 0168-1699. doi: 10.1016/j.compag.2008.04.005. URL http://www.sciencedirect.com/science/article/pii/S0168169908001257.

C. Moser, L. Thiele, D. Brunelli, and L. Benini. Adaptive power management in energy harvesting systems. In *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, pages 1 –6, april 2007. doi: 10.1109/DATE.2007.364689.

L. Nachman, J. Huang, J. Shahabdeen, R. Adler, and R. Kling. Imote2: Serious computation at the edge. In *Wireless Communications and Mobile Computing Conference, 2008. IWCMC '08. International*, pages 1118 –1123, aug. 2008. doi: 10.1109/IWCMC.2008.194.

Pin Nie and Bo Li. A cluster-based data aggregation architecture in wsn for structural health monitoring. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, pages 546 –552, july 2011. doi: 10.1109/IWCMC.2011.5982592.

Michael Nikolaou. Model predictive controllers: A critical synthesis of theory and industrial needs. volume 26 of *Advances in Chemical Engineering*, pages 131 – 204. Academic Press, 2001. doi: 10.1016/S0065-2377(01)26003-7. URL http://www.sciencedirect.com/science/article/pii/S0065237701260037.

K. Oikonomou and I. Stavrakakis. Analysis of a probabilistic topology-unaware tdma mac policy for ad hoc networks. *Selected Areas in Communications, IEEE Journal on*, 22(7):1286 – 1300, sept. 2004. ISSN 0733-8716. doi: 10.1109/JSAC.2004.829345.

J.R. Owen. Prospects for thin film lithium batteries. In *Compact Power Sources (Digest No. 96/107), IEE Colloquium on*, pages 2/1 –2/3, may 1996. doi: 10.1049/ic:19960677.

G.J. Pendock, L. Evans, and G. Coulson. Wireless sensor module for habitat monitoring. In *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on*, pages 699 –702, dec. 2007. doi: 10.1109/ISSNIP.2007.4496928.

V. Pereira, J.S. Silva, J. Granjal, R. Silva, E. Monteiro, and Qiang Pan. A taxonomy of wireless sensor networks with qos. In *New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on*, pages 1 –4, feb. 2011. doi: 10.1109/NTMS.2011.5720617.

D. Porcarelli, D. Brunelli, M. Magno, and L. Benini. A multi-harvester architecture with hybrid storage devices and smart capabilities for low power systems. In *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2012 International Symposium on*, pages 946 –951, june 2012. doi: 10.1109/SPEEDAM.2012.6264533.

G. Quer, R. Masiero, D. Munaretto, M. Rossi, J. Widmer, and M. Zorzi. On the interplay between routing and signal representation for compressive sensing in wireless sensor networks. In *Information Theory and Applications Workshop, 2009*, pages 206 –215, feb. 2009. doi: 10.1109/ITA.2009.5044947.

V. Raghunathan, C. Schurgers, Sung Park, and M.B. Srivastava. Energy-aware wireless microsensor networks. *Signal Processing Magazine, IEEE*, 19(2):40 –50, mar 2002. ISSN 1053-5888. doi: 10.1109/79.985679.

M. Rahnema. Overview of the gsm system and protocol architecture. *Communications Magazine, IEEE*, 31(4):92 –100, april 1993. ISSN 0163-6804. doi: 10.1109/35.210402.

Ramesh Rajagopalan and Pramod K. Varshney. Data aggregation techniques in sensor networks: A survey. *Comm. Surveys & Tutorials, IEEE*, 8:48–63, 2006.

V. Rajendran, J.J. Garcia-Luna-Aveces, and K. Obraczka. Energy-efficient, application-aware medium access for sensor networks. In *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, pages 8 pp. –630, nov. 2005. doi: 10.1109/MAHSS.2005.1542852.

J. Ranieri, R. Rovatti, and G. Setti. Compressive sensing of localized signals: Application to analog-to-information conversion. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 3513 – 3516, 30 2010-june 2 2010. doi: 10.1109/ISCAS.2010.5537820.

Injong Rhee, A. Warrier, M. Aia, Jeongki Min, and M.L. Sichitiu. Z-mac: A hybrid mac for wireless sensor networks. *Networking, IEEE/ACM Transactions on*, 16(3):511 –524, june 2008. ISSN 1063-6692. doi: 10.1109/TNET.2007.900704.

S. Rhee, D. Seetharam, and S. Liu. Techniques for minimizing power consumption in low data-rate wireless sensor networks. In *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, volume 3, pages 1727 – 1731 Vol.3, march 2004. doi: 10.1109/WCNC.2004.1311813.

M.J. Roemer, Jianhua Ge, A. Liberson, G.P. Tandon, and R.Y. Kim. Autonomous impact damage detection and isolation prediction for aerospace structures. In

Aerospace Conference, 2005 IEEE, pages 3592 –3600, march 2005. doi: 10.1109/AERO.2005.1559664.

B.M. Sadler. Fundamentals of energy-constrained sensor network systems. *Aerospace and Electronic Systems Magazine, IEEE*, 20(8):17 –35, aug. 2005. ISSN 0885-8985. doi: 10.1109/MAES.2005.1499273.

M. Saleem and M. Farooq. A framework for empirical evaluation of nature inspired routing protocols for wireless sensor networks. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 751 –758, sept. 2007. doi: 10.1109/CEC.2007.4424546.

E. Shih, B.H. Calhoun, Seong Hwan Cho, and A.P. Chandrakasan. Energy-efficient link layer for wireless microsensor networks. In *VLSI, 2001. Proceedings. IEEE Computer Society Workshop on*, pages 16 –21, may 2001. doi: 10.1109/IWV.2001.923134.

Feng Shu and T. Sakurai. Analysis of an energy conserving csma-ca. In *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, pages 2536 –2540, nov. 2007. doi: 10.1109/GLOCOM.2007.482.

Haining Shu and Qilian Liang. Fundamental performance analysis of event detection in wireless sensor networks. In *Wireless Communications and Networking Conference, 2006. WCNC 2006. IEEE*, volume 4, pages 2187 –2192, april 2006. doi: 10.1109/WCNC.2006.1696635.

D. Slepian and J. Wolf. Noiseless coding of correlated information sources. *Information Theory, IEEE Transactions on*, 19(4):471 – 480, jul 1973. ISSN 0018-9448. doi: 10.1109/TIT.1973.1055037.

Wen-Miao Song, Yan-Ming Liu, and S.-E. Zhang. Research on smac protocol for wsn. In *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on*, pages 1 –4, oct. 2008. doi: 10.1109/WiCom.2008.921.

James P. Thomas, Muhammad A. Qidwai, and James C. Kellogg. Energy scavenging for small-scale unmanned systems. *Journal of Power Sources*, 159(2):1494 –

1509, 2006. ISSN 0378-7753. doi: 10.1016/j.jpowsour.2005.12.084. URL http://www.sciencedirect.com/science/article/pii/S0378775306001121.

A. Tovar, T. Friesen, K. Ferens, and B. McLeod. A dtn wireless sensor network for wildlife habitat monitoring. In *Electrical and Computer Engineering (CCECE), 2010 23rd Canadian Conference on*, pages 1 –5, may 2010. doi: 10.1109/CCECE.2010.5575142.

F. Turati, M. Cesana, and L. Campelli. Spare mac enhanced: A dynamic tdma protocol for wireless sensor networks. In *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pages 1 –6, 30 2009-dec. 4 2009. doi: 10.1109/GLOCOM.2009.5425974.

B.E. Usevitch. A tutorial on modern lossy wavelet image compression: foundations of jpeg 2000. *Signal Processing Magazine, IEEE*, 18(5):22 –35, sep 2001. ISSN 1053-5888. doi: 10.1109/79.952803.

E. van den Berg and M. P. Friedlander. Probing the pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing*, 31(2):890–912, 2008. doi: 10.1137/080714488. URL http://link.aip.org/link/?SCE/31/890.

R.S. Wagner, R.G. Baraniuk, S. Du, D.B. Johnson, and A. Cohen. An architecture for distributed wavelet analysis and processing in sensor networks. In *Information Processing in Sensor Networks, 2006. IPSN 2006. The Fifth International Conference on*, pages 243 –250, 0-0 2006. doi: 10.1109/IPSN.2006.243753.

Zhiwen Wan, Jinsong Zhang, Hao Zhu, K. Makki, and N. Pissinou. On energy-efficient and low-latency medium access control in wireless sensor networks. In *Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE*, pages 1905 –1910, 31 2008-april 3 2008. doi: 10.1109/WCNC.2008.339.

Jie Wang, Qinghua Gao, Hongyu Wang, and Wenzhu Sun. A method to prolong the lifetime of wireless sensor network. In *Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on*, pages 1 –4, sept. 2009. doi: 10.1109/WICOM.2009.5300990.

L. Wang and Y. Xiao. Energy saving mechanisms in sensor networks. In *Broadband Networks, 2005. BroadNets 2005. 2nd International Conference on*, pages 724 – 732 Vol. 1, oct. 2005. doi: 10.1109/ICBN.2005.1589678.

Hongli Yang, Guoping He, and Yulin Dong. Nonnegative tensor decomposition and it's applications in image processing. In *Computer Science and Information Technology, 2008. ICCSIT '08. International Conference on*, pages 212 –217, 29 2008-sept. 2 2008. doi: 10.1109/ICCSIT.2008.99.

Shih-Hsien Yang, Hung-Wei Tseng, E.H.-K. Wu, and Gen-Huey Chen. Utilization based duty cycle tuning mac protocol for wireless sensor networks. In *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, volume 6, pages 5 pp. –3262, dec. 2005. doi: 10.1109/GLOCOM.2005.1578377.

O. Younis, M. Krunz, and S. Ramasubramanian. Node clustering in wireless sensor networks: recent developments and deployment challenges. *Network, IEEE*, 20(3):20 – 25, may-june 2006. ISSN 0890-8044. doi: 10.1109/MNET.2006.1637928.

Pengfei Zhang, Gaoxi Xiao, and Hwee-Pink Tan. A preliminary study on lifetime maximization in clustered wireless sensor networks with energy harvesting nodes. In *Information, Communications and Signal Processing (ICICS) 2011 8th International Conference on*, pages 1 –5, dec. 2011. doi: 10.1109/ICICS.2011.6174242.

Feng Zhao. Wireless sensor networks: a new computing platform for tomorrow's internet. In *Emerging Technologies: Frontiers of Mobile and Wireless Communication, 2004. Proceedings of the IEEE 6th Circuits and Systems Symposium on*, volume 1, pages I – 27 Vol.1, may-2 june 2004. doi: 10.1109/CASSET.2004.1322900.

Jianliang Zheng and Myung J. Lee. A comprehensive performance study of IEEE 802.15.4. page 14, 2003.