Politecnico di Milano

Facoltà di Ingegneria dell'Informazione

Corso di Laurea Specialistica in Ingegneria Informatica

Dipartimento di Elettronica e Informazione

A SYSTEM FOR VIDEO STREAMING IN WIRELESS SENSOR NETWORKS

Relatore: Ing. Matteo CESANA

Correlatori: Ing. Luca Pietro BORSANI

Ing. Alessandro REDONDI

Tesi di Laurea di:

Stefano PANIGA

Matr. 721598

Anno Accademico 2009 - 2010

Sommario

Il lavoro proposto illustra le fasi progettuali ed implementative che hanno condotto alla realizzazione di un sistema di *Video Streaming* per reti di sensori wireless. Nella prima parte del documento sono analizzate le reti di sensori allo scopo di individuare vincoli e limitazioni che condizionano le scelte progettuali. Successivamente, tale analisi viene estesa alle reti di sensori multimediali.

Nella sezione teorica del documento viene presentato lo stato dell'arte delle tecniche di compressione di immagini e video. Tale ricerca è stata orientata all'individuazione dell'algoritmo di compressione video più adatto allo scopo di questo lavoro. Successivamente vengono descritte le fasi progettuali ed implementative che hanno portato allo sviluppo del sistema di *Video Streaming*. Viene inoltre descritta la simulazione volta a confrontare il protocollo di rete implementato con l'alternativa proposta.

Infine, le performances del sistema sono testate ed analizzate. Il sistema ottenuto permette l'invio di un flusso video attraverso i diversi nodi componenti la rete di sensori fino al gateway, dove tale flusso viene visualizzato.

Abstract

This dissertation describes the design and the implementation phases which brought to the realization of a *Video Streaming System* on a Wireless Sensor Network. In the first part, the Wireless Sensor Networks are analyzed to understand the general constraints and limitations that underneath the development environment. Subsequently this analysis is extended to the Multimedia Wireless Sensor Networks.

The document reports also a survey on the current data compression techniques. Such methods were studied to find the best data compression algorithm usable in the *Video Streaming System*.

In the implementation section the requirements and the constraints that characterized the work development are presented, together with the design issues encountered and the solutions adopted. Moreover it is reported the simulation aimed to compare the used network protocol with one of the most common alternatives.

Finally the performances of the *Video Streaming System* are tested and analyzed. The presented system allows to send a video through the diverse nodes of a wireless sensor network till reaching the gateway, where the images flow is displayed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

THE considerable scientific interest around Wireless Sensor Networks (WSN) of the past years was mainly due to the great potentialities of this kind of technology. WSNs offer a low cost and easily scalable network for data collection and environmental monitoring purposes. The price of this simplicity is payed in a resource constrained architecture presenting many limitations in terms of performance and functioning autonomy. These architectural limits give rise to a whole set of design issues faced by the scientific community in the last years: need to reduce energy consumptions, reliability of the nodes, processing power of the motes, etc. Many research results have been reached allowing to extent the applications range of this technology from the original military context to other fields like environmental monitoring, healthcare, data collection, etc.

In the recent years, the technology progresses, in particular in image chips development, allowed to equip sensors with multimedia functionalities. Indeed, new CMOS based cameras reduce consistently the amount of energy required for image acquiring, maintaining a fair pictures quality. Such advances, together with the production of more powerful sensors and, at the same time, with reduced energy consumptions, extended further the fields of application of the WSNs in such a way that a new name was coined: Wireless Multimedia Sensor Network (WSNM).

WSNMs offer many new utilization possibilities that range from environments video monitoring, surveillance systems, patients monitoring, etc. But together with these new functionalities also new problems came. The processing efforts required by the multimedia contents strain also the new generation and most powerful sensors. Moreover, such huge amount of data causes problem also in the transmission phase. Indeed, the network must sustain the increase of network traffic with related problems such as congestion and packets losses management. Together with this routing issues, the need of higher transmission rates and the long active periods of the radio chip bring not negligible energy consumption problems.

The developing of the video streaming system presented in this work had to take in account all these design issues. Before starting the software development, we did an accurate research on the current data compression techniques to evaluate whether one of them could perform better than the already implemented Jpeg encoder, for still images and to chose a light and efficient compression method to further reduce transmitted data size in the video subsystem.

The implemented work is based on an image acquisition software available among the *Intel Mote 2* contributions developed by the Stanford University for TinyOS [3]. We exploited the original Jpeg compression algorithm implementation and the drivers of the OV7670 camera chip , installed on the *IMB400* CrossBow multimedia module, to build a Difference Pulse Code Modulation (DPCM) algorithm to efficiently encode the video frames sequences. In a second phase the module for the serial communication with a general purpose computer, provided by the TinyOS contributions, has been extended to support a reliable multihop radio protocol, based on the CC2420 chip drivers provided by the TinyOS 2.1.1 release.

The radio protocol implementation consisted in developing a sender module for the camera mote together with a base station attached to the general purpose machine. The base station has in charge to receive packets through the radio channel and to forward them to the gateway, on the serial connection. Moreover, we implemented the software supporting the insertion in the network of one or more intermediate nodes to

create a multihop network paradigm.

Also a Tossim simulation of the currently used network protocol has been implemented, as well as another losses management solution. This simulation was needed to compare such protocol with one of the most common alternatives, the *Selective Repeat* paradigm. Once the photo and the video subsystems, together with the radio protocol, were completed, the system performances were analyzed measuring image processing and network parameters. The different functionalities were tested varying the number of intermediate nodes, till a maximum of four, and the transmission power levels, going from 0 dBm to -25 dBm. The testbed preparation required some code modifications aimed to collect information without compromising the correctness of the measurements.

The outcome of the work will be deeply reported and explained in the rest of the dissertation:

Chapter 2 introduces the sensor networks describing the most important features and the most common applications.

In Chapter 3, Wireless Multimedia Sensor Networks are analyzed also reporting the new challenges and issues deriving from this new sensor network paradigm.

Chapter 4 presents the results of the initial research work about the state of the art of images and video compression techniques.

In Chapter 6 we describe the design and development phases of the system deployment. Moreover it is described the development of the network protocols simulation.

In Chapter 7, the results of the performance tests for the different system configurations are presented.

Finally, in Chapter 8 considerations about the performances of the implemented system and the work outcomes are presented, together with possible future developments.

Chapter 2

Wireless Sensor Network

Wireless sensor network (WSN) is a relatively new technology gaining a growing attention in the scientific and industrial community. This fact is referable to the diverse noteworthy and peculiar features offered by a sensor network, especially whether developed on a wireless architecture. A WSN, represented in Figure 2.1, consists in a set of sensor nodes (also called motes) communicating each others through a wireless channel, with no constraints on the topology of the network. Those sensors constitute a low cost and low power networked system, particularly well suitable to data collection and environment monitoring, easily adaptable to different aims through software programming. Moreover, this technology offers contained installation costs and a reduced maintenance. Examples of monitored data range from simple measurements, like temperature or humidity values, to more elaborated information, like accelerations or vibrations.

The technological improvements in the hardware design of the last years have permitted to deal with sensors supporting a certain degree of on board computation and consuming a relatively small amount of energy; much help has come also from the large use of energy saving policies that involves network operations as well as the software executed by the sensor.

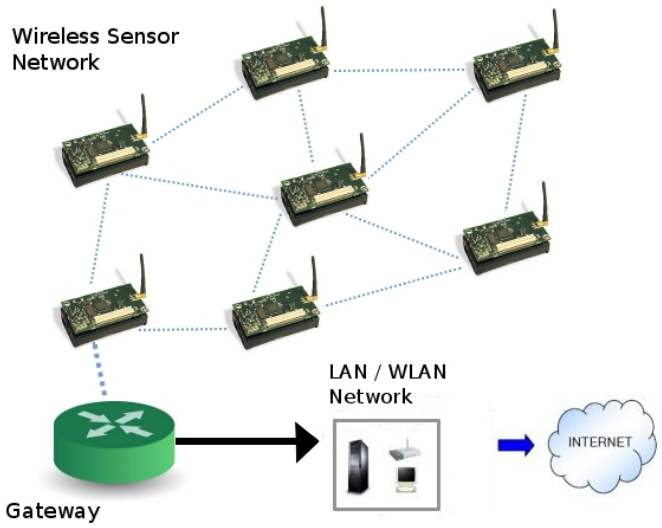


Figure 2.1: Schema of a WSN

The design of this kind of network is similar to the common ad-hoc networks planning methodology but with some differences, imputable to the distinct technology adopted: the number of nodes in a WSN can be notably higher than in an ad-hoc network, motes are densely distributed in the environment and the topology could change frequently. However, WSNs present new problems with respect a common wireless scenario: motes are largely exposed to malfunctions and are subjected to tight energy constraints. Though the network protocol has often been designed to maximize the energy savings communication is the activity with the highest energy dissipation in a generic WSN. Another issue comes from a common wireless network problem, that results amplified by the resource limits of the WSN: this is the low reliability of the wireless channel. While in a wired network the capacity of each link is fixed, in a wireless environment, due to the interference level perceived at the receiver, the channel capacity can vary even consistently. Things get worse in a sensor network scenario, because transmission capacity is already limited and retransmissions cost energy.

2.1 Sensors

The typical hardware configuration of a mote consists of a micro controller for computation, a RAM memory to store dynamic data structures, flash memories where are placed the execution code and long-lived data, a communication module, composed of an antenna and a wireless transceiver, one or more sensors and a power source, usually provided by batteries to support the versatility often required to this kind of network architecture. Some examples of currently available sensors are showed in Figure 2.2

It is possible to classify the sensing units in two separated categories: passive and active. While the first type of sensors does not need to act in an active manner to acquire information, like light and pressure transducers , the other one require a direct and constant action to acquire data, consuming a considerable amount of energy. This is the case, for example, of a sensor equipped with a camera.

One of the most famous motes is the MicaZ: it is based on an Atmel micro controller

Figure 2.2: Wireless Sensors Examples

together with 4K bytes of RAM memory, 128 K bytes of program flash, 512 K bytes of

log memory. On the communication side it has an IEEE 802.15.4 2.4-GHz transceiver that supports a maximum rate of 250 Kbps and a range of about 50 meters. It acquires data through a 10-bit ADC, and runs on two AA batteries. The processor current draw is of 8 mA in active mode while the radio frequency transceiver consumes almost 20 mA in receive mode.

2.2 Network

Inter node communication could be reached through different techniques. Infrared data transmission is tolerant to interference, offers reliable transmissions and is license-free but requires each node to be in line of sight with the others. Another approach exploits Bluetooth proprietary technology. The main issues of this solution are: high energy consumption, strong limitations on the number of nodes in the network and complex overlying MAC layer.

Many solutions bases the network communication on the IEEE 802.15.4 standard. The IEEE 802.15.4 standard was developed to provide a framework and the lower levels for low cost, low power networks. It only provides the MAC(Media Access Control) and PHY(PHYsical) layers, typically for a Personal Area Network (PAN), leaving the upper layers to be developed according to the market needs.

The chief requirements are low-cost, low-speed but ubiquitous communication between devices. The concept of IEEE 802.15.4 is to provide communications over distances up to about 10 meters and with maximum transfer data rates of 250 Kbps. Additionally, when the hardware supports radio sleep mode, it permits to the recovered node a fast re-synchronization with the network.

2.3 Designing guidelines

To realize the typical WSN requirements, innovative mechanisms for a communication network have to be found, as well as new architectures and protocol concepts. One of the main challenges is the need to support specific quality of service, lifetime and maintainability requirements of specific applications. On the other hand, these peculiar mechanisms also have to generalize to a wider range of applications allowing to contain costs.

The wireless medium adopted by these networks imposes the designer some limitations. In particular, communication between over long distances is only possible using prohibitively high transmission power. The use of intermediate nodes as relays can reduce the total required power. Hence, for many WSN implementations, multihop communication is a necessary solution.

The considerable number of nodes and the request of a simple deployment require the ability of the network to configure most of operational parameters autonomously, independent with respect to external configuration. For example nodes should be able to determine their geographical position using only other nodes of the network. Also, the network should be able to tolerate failing nodes or to integrate new nodes.

In some applications, a single sensor is not able to decide whether an event has happened but several sensors have to collaborate to detect an event and only the joint data of many sensors provides enough information. Instead of sending all the data to the edge of the network, where is processed, the information is elaborated in the network itself to achieve a collaboration model reducing the transmission load. An example could be the measurement of the average temperature in a place: while data are propagated through the network they are aggregated to reduce the total number of transmissions.

In a WSN, where nodes are typically deployed redundantly to protect against failures, the identity of the particular supplying data node becomes irrelevant. What is important are the answers and the values themself, not which node has provided them.

Hence, the address-centric approach typical of traditional communication networks leaves room to a data-centric paradigm.
Another property required in a WSN is locality: nodes, who are very limited in memory resources, should attempt to limit the state that they accumulate during protocol processing only to information about their direct neighbors. This help the network to scale to large numbers of nodes without having to rely on powerful processing at each single node.
Obviously, all these properties must be implemented in energy efficient manners, necessary to support long lifetimes.

2.4 Applications

WSN development was originally motivated by military purposes: this technology is particularly suitable in a war environment due of the fault tolerant characteristics and the self organizing capacity of the network. Furthermore, due to the low cost of

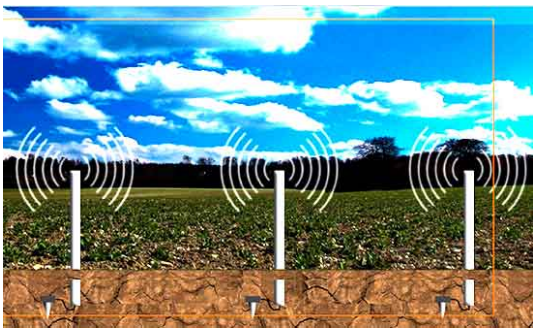


Figure 2.3: WSN for Pipeline Security

the components and the high mote density, the destruction of some devices does not affect the network integrity. Common uses include: battlefield surveillance, targeting and target tracking systems,nuclear, biological or chemical attack detection.

In the last years, WSN are gaining popularity even in a civilian context, where this science has found many fields of application, especially in environment monitoring

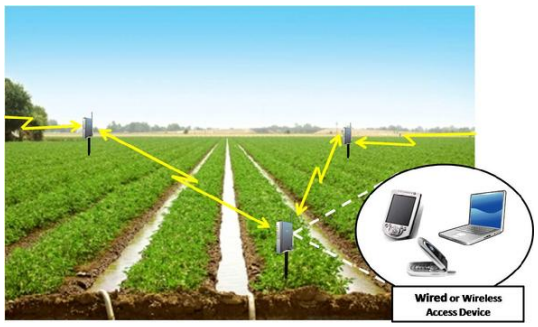


Figure 2.4: Environmental Monitoring Example

like, for example, precision agriculture,tracking of movements of small animals, pollution study. Two examples of WSN applications in a civilian context are showed in Figures 2.3 and 2.4. But WSN find space also in healthcare area, where this technology is used in tracking and monitoring of patients as, for instance, in the Laura project [15], in drug administration in hospitals, providing interface for the disabled people etc. Others applications concern factory automation processes and smart home technologies.

Chapter 3

Wireless Multimedia Sensor Network

WSN were originally focused on the collection of scalar data, simple processing and transmission to remote locations. Here, received data were processed by more powerful devices to obtain the most accurate information. More recently, the availability of inexpensive and low power hardware such as CMOS camera and microphones that allow to capture multimedia content from the environment has fostered the development of Wireless Multimedia Sensor Network (WMSN). These new devices allow retrieving video and audio streams, still images and scalar sensor data. In addition to the ability to collect multimedia data, WMSN are able to process, store and, in case of heterogeneous sources, correlate or fuse different information flows. This new research direction, not only enhance existing sensor network applications such as tracking, home automation, and environmental monitoring, but they will also enable several new ways of employment of this kind of technology:

- *Multimedia surveillance sensor network:*

  wireless video sensor networks will be composed of interconnected, battery-powered miniature cameras, each packaged with a power wireless transceiver capable of processing, sending and receiving data. Video and audio sensors will be used to complement existing surveillance systems.

- *Storage of potentially relevant activities:*

Multimedia sensors could infer and record potentially relevant activities (thefts, car accidents, traffic violations), and make video/audio streams reports available for future query.

- *Traffic avoidance, enforcement and control systems:*
  it will be possible to monitor car traffic in big cities or highways and deploy services that offer traffic routing advice to avoid congestion or simply to collect vehicular traffic data.

- *Advanced health care delivery:*
  Patients will carry medical sensors to monitor parameters such as body temperature, blood pressure, pulse oximetry, ECG, breathing activity. Furthermore, remote medical centers will perform advanced remote monitoring of their patients via video and audio sensors, location sensors, motion or activity sensors, which can also be embedded in wrist devices.

- *Automated assistance for the elderly and family monitor:*
  Multimedia sensor networks can be used to monitor and study the behavior of elderly people. Networks of wearable or video or audio sensors can infer emergency situations and immediately connect elderly patients with remote assistance service or relatives.

- *Environmental monitoring:*
  Several projects on habitat monitoring that use acoustic and video feeds are being envisaged. For example, arrays of video sensors are already used by oceanographers to determine the evolution of sandbars via image processing techniques[10]].

- *Person locator services:*
  Multimedia content such as video streams and still images, along with advanced signal processing techniques, can be used to localize missing people, or identify wanted criminals.

- *Industrial Process Control:*

  Imaging, temperature and pressures for instance, may be used for time-critical industrial process control. For example, in quality control of manufacturing processes, such as those used in semiconductor chips, automobiles, food or pharmaceutical products.

3.1 Multimedia sensors

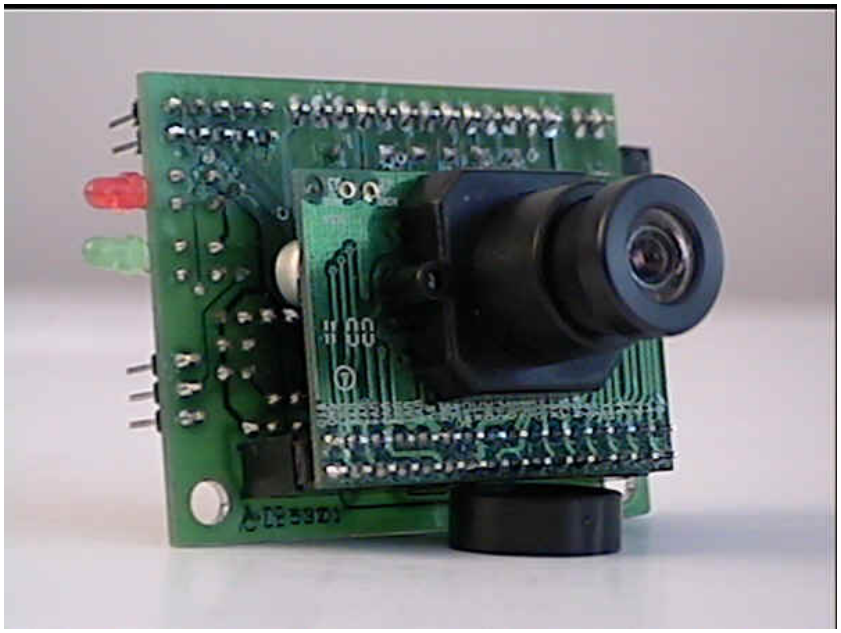


Figure 3.1: CMUcam1, CMOS Based Camera

A multimedia sensor does not differ much from common sensors, as those examined in Chapter 2. Basically they are equipped with a processing unit, a communication subsystem, a coordination system and a storage unit. The great difference is the multimedia module and the correspondent analog-to-digital (ADC) converter. Usually this module is equipped with a microphone but, most of all, an image sensor as the one showed in Figure 3.1. This add new capacities to the mote but increases also the complexity of some components such as the ADC. Since 1975 till few years ago, the de

facto standard for image sensors was the CCD (charge coupled device). As compared to traditional CCD technologies, there is a need for smaller, lighter camera modules that are also cost-effective when bought in large numbers to deploy a WMSN.
In a CCD sensor, the incident light energy is captured as the charge accumulated on a pixel, which is then converted into a voltage and sent to the processing circuit as an analog signal. This architecture brings some disadvantages with respect to its integration on a wireless sensor: they require more electronic circuitry outside the actual image sensor, are more expensive to produce and, above all, they consume up to 100 times power than a CMOS sensor. Unlike the CCD sensor, the CMOS chip incorporates am-

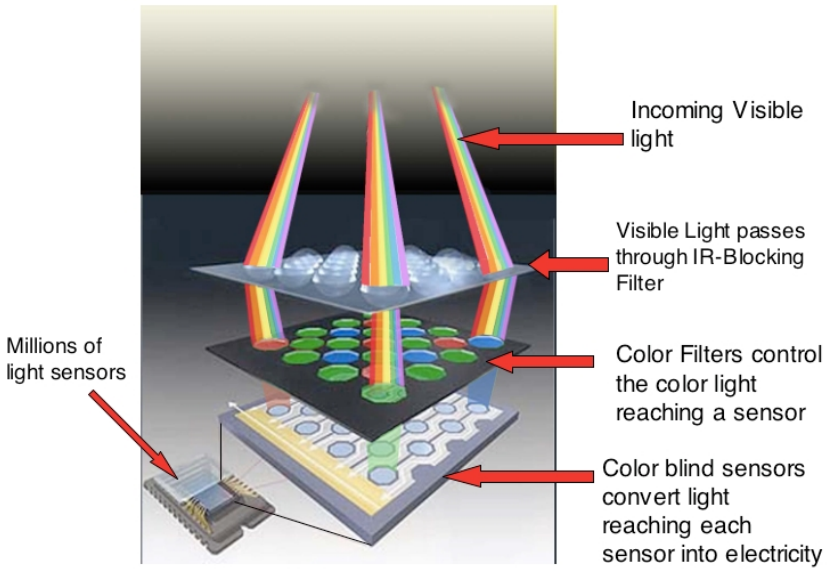


Figure 3.2: CMOS Sensor Functioning Schema

plifiers and A/D-converters, which lowers the cost for cameras since it contains all the logics needed to produce an image. Every CMOS pixel contains conversion electronics. Compared to CCD sensors, CMOS sensors have better integration possibilities and more functions. However, this addition of circuitry inside the chip can lead to a risk of more structured noise, such as stripes and other patterns. CMOS sensors also have a faster readout, lower power consumption, higher noise immunity, and a smaller

system size[11]. In Figure 3.2 it is showed how the CMOS sensor captures the image transforming it in an electrical signal. Depending upon the application environment, such as security surveillance needs or biomedical imaging, these sensors may have different processing capabilities.

3.2 Design Issues

The volume of data collected as well as the complexity of the necessary in-network stream content processing provides a diverse set of challenges in comparison to generic scalar sensor network research. Multimedia motes have to support tasks increasingly demanding in terms of number and computational complexity of operations. Due to the considerable energy dissipation spent in transmission, acquired data have to be elaborated in order to reduce size. This task could require a lot of energy resources if low complexity and efficient coding techniques are not used. Moreover, acquiring more and more accurate data, increase dramatically the load in the entire network. For this matter, it is necessary, to combine data compression techniques with new transmission policies aimed at reduce the network traffic.

The wide variety of applications envisaged will have different requirements. In addition to data delivery modes typical of scalar sensor networks, multimedia data include content based requirements. This force to consider QoS (Quality of Service) and application specific requisites. These requirements may pertain to multiple domains and can be expressed, amongst others, in terms of a combination of bounds on energy consumption, delay, reliability, distortion, or network lifetime.

Multimedia contents, especially video streams, need transmission bandwidth that is orders of magnitude higher than that supported by currently available sensors. For example, the nominal transmission rate of state-of-the-art IEEE 802.15.4 is 250 Kbit/s. Data rates at least one order of magnitude higher may be required for high-end multimedia sensors, with comparable power consumption. Therefore we need to balance the trade-off between transmission performance and energy consumption. A help could

come from new transmission techniques such as UWB (Ultra Wide Band)[4]
Because of the considerable amount of data generated by multimedia sensors, efficient processing techniques for lossy compression are necessary for this kind of networks. Traditional video coding methods are based on reducing bits generated by the source encoder by exploiting the source statistics. More commons encoders rely on intra-frame compression techniques to reduce the redundancy within the same frame and inter-frame compression to exploit redundancy between subsequent frames. The two correlation types are showed in Figure 3.3. Since predictive (inter-frame) coding re-

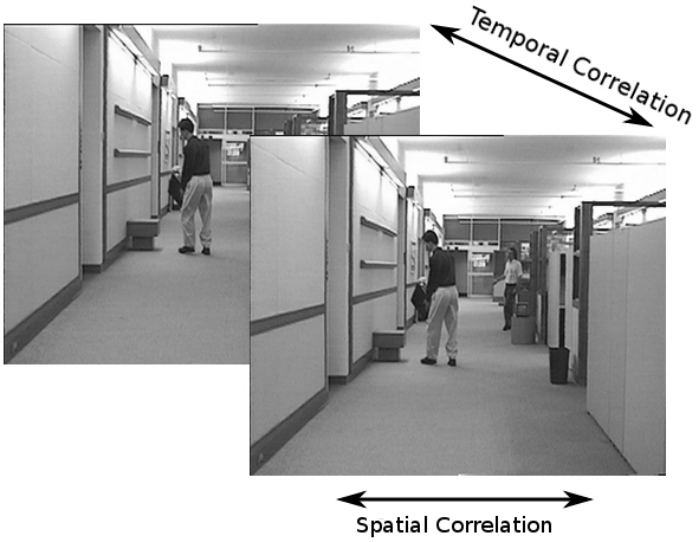


Figure 3.3: Spatial and Temporal Correlation Example

quires powerful processing algorithms it may not be suited for low-cost multimedia sensors. The solutions adopted try to move as much computation as possible to the decoder, designing lightweight and distributed source encoders that produce different small data flows put together by the decoder [9].
WMSN allow performing multimedia in-network processing algorithms on the raw data extracted from the environment. This requires new architectures for collaborative, distributed and resource constrained processing that allow for filtering and extraction of semantically relevant information at the edge of the sensor network. In this way it,

is possible to increase the system scalability by reducing the transmission of redundant information, merging data originated from multiple views, on different media and with multiple resolutions.
Obviously power consumption has a central role in the deployment of WMSN. The considerable amount of data and its elaboration require energy saving oriented algorithms, even more than in traditional WSN. In fact, while the energy consumption of traditional sensor nodes is known to be dominated by the communication facilities, it may not necessarily be true in WMSN.
Wireless multimedia sensor networks will support several heterogeneous and independent applications with different requirements. It is necessary to develop flexible, hierarchical architectures that can accommodate the requirements of all these applications in the same infrastructure.

3.3 State Of The Art

The field of video sensor networks is a relatively new area of interest that offers different research opportunities. The add of multimedia content to the sensor networks required a rethinking of the network layers and architecture. Many of the last years researches were focused on answering to the requirements of the new network paradigm. With the increase of the required bandwidth, the old MAC (Medium Access Control) protocols became inefficient. Indeed, existing schemes are based mostly on variants of CSMA/CA (Carrier Sense Multiple Access, with Collision Avoidance) MAC protocol and present notable limitations in terms of latency and coordination complexity. The channel contention could be significantly reduced using two radios in which to one is delegated the task of channel monitoring and is responsible for waking up the main radio for data communication. The disadvantages of this approach reside in the distinct channels assignment and in the increased hardware complexity.
A contention free alternative could be the implementation of a TDMA (Time Division Multiple Access) protocol. With this paradigm, channel resources are assigned

in a small time interval with a contention based method while the rest of the frame is contention-free and divided on the basis of the QoS (Quality of Service) requirements. The main problems of this technique concern the network scalability and the complex network-wide scheduling.

A third way reproduces MIMO (Multiple Input Multiple Output) antenna systems, where each sensor functions as a single antenna, sharing information and simulating a multiple antenna array.

The unreliability of the wireless channel requires error correction mechanisms. The main link layer classes of error correction protocols are FEC (Forward Error Control) and ARQ (Automatic Repeat Request). While the former applies different degrees of redundancy to different parts of the video stream, on the basis of the part importance, the ARQ protocols use the bandwidth efficiently at the cost of additional latency involved with the retransmission process. Recent comparisons between the two techniques showed that for certain FEC block codes (BCH), longer routes decrease both the energy consumption and the end-to-end latency [21].

In a WMSN could exist nodes with different capabilities. Consequently there could be different routes with different characteristics based on the diverse features of the nodes along the path. Moreover the routing could be based on video stream content. For instance, streams belonging to cameras with the same orientation could follow the same route to facilitate redundancy removal.

Another approach could speed up the routing procedures differentiating between flows with different delay and reliability requirements. Each node selects its next hop based on link-layer delay measurements [8].

Transport layer protocols can follow UDP (User Datagram Protocol) or TCP (Transport Control Protocol) models. Usually, for multimedia contents, UDP paradigm is more appreciated. It allows to give more importance to timeliness constraints than reliability requirements. The Real-time Transport Control Protocol (RTCP), based on UDP, permits a dynamic adaptation to the network conditions, allowing bandwidth scaling and integration of different images in a single composite. In addition, with

application level framing (ALF) it is possible to ensure a better compatibility between different networks, such as WSN and IP-networks, encoding specific instructions in the packets header.

On the other side, TCP approaches support a more selective management of the network traffic. For example, in MPEG protocol, some frames, called I-frames, can not be retrieved by interpolation and dropping packets indiscriminately, as in the case of UDP, may cause discernible disruptions in the multimedia content. It would be better to introduce a sort of selective reliability mechanism that increases the protection of these packets with respect to less important parts.

Although many of the cited works were developed in a simulated environment, some of them were implemented. For example in the surveillance systems, where WMSN allow costs reduction and more flexibility in the system installation, new way of monitoring are offered, like target classification and tracking [7].

Alongside of the existent development environments, new frameworks are developed with the aim of helping the designers in implementing new multimedia applications that can exploit the potentialities offered by sensors with multimedia capabilities [16].

The increase of the network traffic, due to the multimedia content of the network, requires the developing of new routing protocols to efficiently drain the huge amount of data. Older protocols must be updated or substituted by newer and more reliable traffic management methods [19].

Other research studies concern testbeds to test new protocols and applications developed for Wireless Multimedia Sensor Networks or to measure how the existent techniques perform with respect to the new and more problematic multimedia approaches [13].

Chapter 4

Image Coding Techniques

THE considerable amount of information produced by a multimedia sensor together with the limited resources of commons WSN, require data compression techniques to reduce the network load and the intermediate node's processing efforts.

Images are made of pixels, each pixel is described by a variable number of bits depending on the image type, grayscale or colored, and quality. The number of bits required to represent an image may be substantially lower because of redundancy. This redundancy can be classified in three different types: spatial redundancy, due to the correlation between neighboring pixel values, spectral redundancy, generated by the correlation between different color planes or spectral bands and temporal redundancy, due to the correlation between different frames in a sequence.

Image compression research aims to reduce the number of bits required to represent an image by removing these redundancies.

The most general categories in which a compression method could fall are: lossless and lossy compression. While lossless methods allow to reconstruct an image identical to the original, on a pixel-per-pixel base, lossy compression drops some information with respect to the original, so that the result of reconstruction operations contains degradations.

In a WMSN environment, the choice of a technique aimed to reduce the size of an image captured by a node must be driven by the performance limits of the mote's hardware and by the energy constraints to which it is subjected.
The major part of compression techniques elaborate the signal transforming it and processing data in frequency domain to exploit properties of the transform with the aim of a more efficient compression of the information.

4.1 Fourier Transforms

There exist many types of transforms: Fourier, Cosine, Hadamard, Karhunen Loeve, Tchebichef, etc. Each one has different properties and complexity. In the image compression field discrete Fourier transformation (DFT) is frequently used because it can condense most of the image information in few coefficients. The main issue is the high computational complexity. On the other side, Hadamard transformation is very fast and performs better than Tchebichef or discrete cosine transformation (DCT) with images containing rapid gradient variations while loses effectiveness for continuous tone images[12].
One of the most used transforms, adopted in JPEG and MPEG standards, is DCT. This technique is used when the input signal contains only real parts: it is referable to the Fourier transform where the the imaginary part is always zero. It is appreciated for the compromise between performance and effectiveness offered and for some owned properties like the fact that for real inputs it gives real output so that less storage space is needed and for the ability of exploiting the correlation between adjacent pixels. DCT is able to concentrate most of the signal in the lower spatial frequencies so that, in the subsequent quantization operations, most of high frequencies coefficients have a zero or negligible value and can be cut. Another advantage of the DCT is the possibility of computing the transform with a reduced complexity. Although the direct application of these formulas would requires $O(N^2)$ operations, it is possible to compute the same

thing with only $O(NlogN)$ complexity by factorizing the computation similarly to the fast Fourier transform (FFT) [6].

The DCT represents a finite set of points in a sum of cosine functions,oscillating at different frequencies.

The one dimension DCT formula is:

$$\sum_{0}^{n} x_n cos[\frac{\pi}{N}(n+\frac{1}{2})k] \qquad k = 0,...,N-1 \tag{4.1}$$

For image processing, which obviously deal with two dimensions arrays, it is sufficient to apply such transformation two times.

Consider a signal X of size M by N. Then the 2D DCT of X is given by Y which is also a M by N signal, where Y is given by:

$$Y_{p_q} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} X_{mn} cos(\pi \frac{2m+1}{2M} p) cos(\pi \frac{2n+1}{2N} q) \tag{4.2}$$

$$\alpha_p = \begin{cases} \frac{1}{\sqrt{M}} & if \quad p == 0 \\ \frac{2}{\sqrt{M}} & else \end{cases} \qquad \alpha_q = \begin{cases} \frac{1}{\sqrt{N}} & if \quad q == 0 \\ \frac{2}{\sqrt{N}} & else \end{cases}$$

The purpose of the transformation is to prepare the data set for the compression. The output of the DCT algorithm contains the same information of the original image, it is a lossless step. Usually this transformation is followed by two others elaborations: quantization and entropy encoding. These will be examined in details in the next paragraphs.

4.2 Wavelets

The Fourier transform is only able to retrieve the global frequency content of the signal, while the time information is lost. This is overcome by Short Time Fourier Transform (STFT) which, applying a constant temporal window in the calculation of

the Fourier transform, can retrieve also temporal information. Notice that, due to the fixed window length, they are extracted constant-time and constant-frequency resolution samples. This works well for low frequencies where often a good frequency resolution is required over a good time resolution while, for high frequencies, time resolution is more important than frequency resolution.

The wavelet transform allows a multi-resolution analysis. It is calculated similarly to the Fourier transform but trigonometric analysis functions are replaced by a wavelet function. A wavelet is a short oscillating function that contains both the analysis function and the window. Time information is obtained by shifting the wavelet over the signal while the frequencies are changed by contraction and dilatation of the wavelet function.

The discrete wavelet transform (DWT) is gaining increasing importance in digital image processing. It uses filter banks to perform the wavelet analysis. The DWT transform decomposes the signal into wavelet coefficients from which the original signal can be reconstructed again. The wavelet coefficients represent the signal in various frequency bands. The coefficients can be processed in several ways, giving the DWT attractive properties over linear filtering.

The DWT is defined as:

$$C(j,k) = \sum_{n \in \mathbb{Z}} f(n)\psi_{jk}(n) \tag{4.3}$$

$$\textit{where} \qquad \psi_{jk}(n) = 2^{-\frac{j}{2}}\psi(2^{-jn} - k)$$

Images are analyzed and synthesized by bi-dimensional filter banks. The low frequencies, extracted by high scale wavelet functions, represent flat background, the high frequencies represent regions with textures. The compression is performed with a multi level filter bank that divides the signal in subbands. Lower bands, that give an approximation of the original image and are supposed to last for the entire duration of the signal, must be codified with less bits whereas higher frequencies, that represent image details and are assumed to appear from time to time, should have fewer bits.

After the DWT analysis, bit allocation and quantization is performed on the coefficients. The coefficient are grouped scanning and coded for compression. The process of entropy coding can be split in a modeling part where probabilities are assigned to the coefficients and a coding part where each coefficient is coded. An image could be compressed also ignoring coefficient under a threshold to obtain lossy compression. This technique finds application also in video compression.

4.3 Image Compression Algorithms

There is a considerable number of compression techniques in literature. As for transforms algorithms each one has peculiar advantages and disadvantages. It is not in the purposes of this work to discuss all the existent methods, therefore only some of them will be taken in account in order to give an idea of the diverse approaches that can be used in image processing.

4.3.1 Lossless compression

Lossless compression algorithms allow the reconstruction of the exact original data set starting from compressed data. This kind of compression is used in many applications such as ZIP or GZIP file formats. It is mainly adopted for text or source code compression or in specific image format such as PNG or GIF. All the lossless compression techniques rely on Shannon's noiseless coding theorem [18] that guarantees the correct decoding of the compressed data as long as the average number of symbols out of the decoder exceeds the source entropy by an arbitrary small amount.
As showed in Figure 4.1, almost all lossless compression methods consist of two stages: decorrelation and entropy coding. Decorrelation means remove the redundancies present in the original data source while entropy coding consists, basically, in assigning smallest codewords to more frequent source symbols.
A notable technique that does not follow the described schema is Bit-plane coding,

employed especially in grayscale images coding. This method considers the binary representation of each pixel and builds as many matrixes as the number of bits that constitute the pixel word, filling each matrix with bits that covers the same position in the different words (bit planes). Each bit-plane is then separately compressed using run-length encoding.

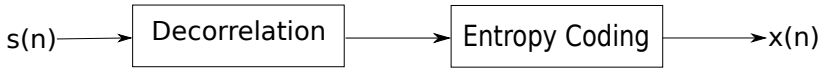


Figure 4.1: Lossless Compression Schema

Decorrelation

One noteworthy decorrelation technique is linear prediction. It consists basically in a differential pulse code modulation (DPCM) without quantization. For each sample it builds a prediction from the weighted sum of neighboring samples. Decorrelation is accomplished subtracting predictions to the actual sample values so that the entropy of the source results diminished.

Another decorrelation method is associated to transforms. Since to avoid losses we can not apply quantization, the approach consist in coding the information otherwise discarded. This is accomplished subtracting the output of the lossy coder (transformation and quantization) from the original data set and entropy coding the result. It must be said that this coding technique does not reach a noteworthy compression ratio.

Alongside of these technologies there is a set of multi resolution techniques including hierarchical interpolation (HINT), the Laplacian pyramid and the S-Transform. These methods form a hierarchy of data sets representing the original data with varying resolution and supporting therefore progressive transmission which allows data to be decoded in several stages. For example, HINT starts with a subsample version of the original data set that has been coded with any lossless technique. Linear prediction

is then applied to subsampled data to estimate intermediate samples. Subsequently the difference between estimated intermediate sample and actual intermediate samples is entropy coded. The process is repeated until all the intermediate samples have been estimated.

Entropy Encoding

The probability mass function (PMF), in probability theory, is a function that gives the probability that a discrete random variable assume a certain value. Unless the PMF of the decorrelated data is uniform, entropy coding can be applied to further compress the data. The majority of compression schemes employ Huffman coding or arithmetic coding.

Huffman coders generate, given a source alphabet and the corresponding PMF, the optimal set of variable length binary codewords of minimum average length. The Huffman codes length is always within one bit of the entropy bound. Most practical Huffman encoders are adaptive and estimate the source symbols probabilities from the data. A Huffman code is built in such a way that a word can not be a prefix of another word and that words with highest number of occurrences are coded with shorter codewords.

Arithmetic coding is a generalization of Huffman coding. The principle is the same as Huffman coding: it is a variable length encoding scheme where frequently used characters will be stored with fewer bits. Rather than separate the input into component symbols replacing each with a codeword, the arithmetic encoding codes all the message in a single number, a fraction n where $0.0 < n < 1.0$. The decoding process start dividing the real interval $[0.0, 1[$ in subintervals corresponding to the PMF of the symbols. In this way exists a direct relation between an interval and a source alphabet symbol. The first decoded word is found looking up in which interval the received number falls. Then the individuated interval is subdivided in the same manner as before and the procedure is recursively repeated until all the message has been decoded.

The Lempel-Ziv-Welch (LZW) coder was developed for text compression but was also applied in signal compression with limited success. It is a dictionary based technique: when a sequence of symbols matches a word stored in the dictionary, its index is sent, otherwise the sequence is forwarded without coding and is subsequently added to the dictionary. This algorithm does not need to be preceded by decorrelation procedures because exploits source correlation. The main issue of this method is that the dictionary size increases rapidly with increasing sample resolution, making it unsuitable for high performance applications.
Another frequently used technique is the run-length coding, particularly useful for compression of binary source data where long sequences of the same symbol have high probability. The idea at the ground of this method is to transmit the length of the runs of repeated symbols instead of transmitting each symbol.

4.3.2 Lossy compression

Lossy compression is a data encoding method in which some information is deliberately discarded to achieve a better compression ratio. This methods are mainly applied in the discretization of continuous sources in a finite set of discrete values but find also employment in the transformation of a discrete source in another with a smaller alphabet. Obviously the applying of this techniques introduce a distortion in the resulting output signal. This degradation is usually quantified through dedicated metrics that, with the reached compression ratio (the ratio between compressed size and original size), is used to estimate the fairness of the algorithm.

Quantization

Quantization is the non invertible operation used to discretize a set of continuous random variables. It can be distinguished in: scalar quantization, referred to the quantization of a single random variable, and vector quantization, referred to the quantization of a block of random variables simultaneously. Moreover, it can be applied

to already discrete sources to obtain a coarser representation of the original data set which allows the possibility of higher compression ratios. The method is based on a set of output values, called reproduction levels, and on a set of decision regions stated with the constraint that each one of the source symbols belongs to one interval. The quantization function substitutes each value of the source with the corresponding reproduction level. In this way the initial alphabet is reduced to a smaller set of symbols. Vector quantization is the generalization of the scalar quantizer of a single variable to the case of a block or a vector of random variables. Augmenting the samples length increases the coding capacity because the simultaneous coding of a group of random variables allows a more efficient representation of the source information. The functioning is the same for the single variable case but here the decision regions are in R^n. The design of an optimal generic quantizer for a source with given statistics, consists in finding the codebook and the partition that minimize the distortion. There are different optimization methods: for scalar quantizer it is often used the Lloyd-Max algorithm from which it has been derived the Linde-Buzo-Gray (LBG) algorithm for vector quantizers.

Jpeg

Jpeg is the acronym for Joint Photographic Expert Group. It has published diverse standards for image compression: among those we mention Jpeg, lossy compression algorithm of continuous tone still images, Jpeg-LS, lossless and near lossless compression algorithm for continuous tone still images and Jpeg2000, scalable coding of continuous tone still images, lossless and lossy.

The most famous of them, Jpeg, has been developed in different implementations. The baseline version of the standard includes the following phases: each color component is divided in blocks of 8x8 pixels each, then each block is processed independently applying DCT transform, subsequently the output of the DCT step is quantized cutting off the transform coefficients with a value minor than a threshold. At last the data flow

is run length and Huffman encoded. The quantization step could be influenced tuning the quality factor of the Jpeg algorithm: this number is usually comprised between 0 and 100 and its value is inversely proportional to the quality of the output image. A schema of the Jpeg compression steps is showed in Figure 4.2.

The Jpeg standard is appreciated for its low complexity, the limited memory requirements and the reasonable coding efficiency. On the other side it presents also many issues such as blocking artifacts at low bit rate, does not have lossless capabilities, poor error resilience etc.

Some of the problems have been fixed in the Jpeg2000 standard. Here we find an improved coding efficiency, full quality scalability from lossless to lossy at different bit rates, division of the image in subimages (tiling), improved error resilience and region of interests, part of the image that can require a better quality compression. Another interesting characteristic in Jpeg2000 standard is the use of wavelet transformation in place of the discrete cosine transformation. This brings consistently advantages in scalability management: while in Fourier transformations, to which DCT belongs, the basis functions cover the entire signal range, varying in frequency only, wavelet basis functions vary in frequency as well as spatial coordinate. This allow to achieve spatial scalability reconstructing from only low resolution coefficients as well as quality scalability decoding only sets of bit planes, starting from the most significant bit plane.

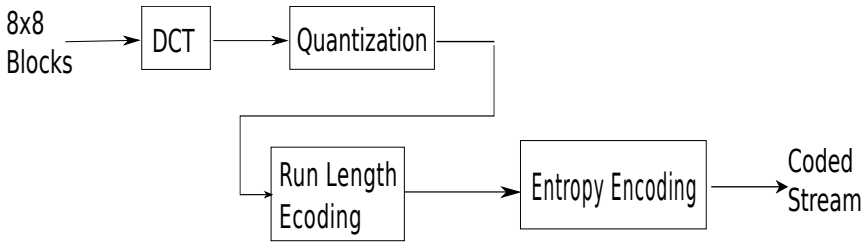


Figure 4.2: Jpeg Compression Schema

Embedded Zerotree Wavelet and Set Partitioning in Hierarchical Trees

An embedded coding is a process of encoding the transform coefficients that allows for progressive transmission of the compressed image. Zerotrees are a concept that allows for a concise encoding of the positions of significant values that result during the embedded coding process. The embedding process used in EZW is called bit-plane encoding. The EZW algorithm is specially designed to use with wavelet transformation.

The method is divided in multiple steps, showed in Figure 4.3. First of all it must be chosen the threshold value, so that there exist at least one wavelet coefficient greater than the threshold. Subsequently, the threshold value is halved and then, in the *significance pass*, the algorithm builds the set of wavelet quantized coefficients assigning $w_q = T_k$ and sending in output the coefficient sign whether the absolute value of the correspondent wavelet coefficient is greater than the threshold, otherwise leaving $w_q = 0$. In the *refinement pass* the algorithm scans through significant values: if $w(m) \in [w_q(m), w_q(m) + T_k]$ it outputs the bit 0, if $w(m) \in [w_q(m) + T_k, w_q(m) + 2T_k]$ it outputs the bit 1.

In few words, the bit-plane encoding of the EZW algorithm consists in computing binary expansions for the transform values using the initial threshold as unit and then recording in magnitude order only the significant bits of these expansions. During the decoding process the signs and the bits output can be used to reconstruct an approximate wavelet transform to any desired degree of accuracy. Once obtained the desired detail level, it is possible to decide to stop the decoding process.

Zerotrees are exploited to reduce the number of bits sent to the decoder. It must be noticed that natural images in general have a low pass spectrum. When an image is wavelet transformed the energy in the subbands decreases as the scale decreases (low scale means high resolution), so the wavelet coefficients will, on average, be smaller in the higher subbands than in the lower subbands. Moreover, large wavelet coefficients are more important than smaller wavelet coefficient. There are coefficients in different

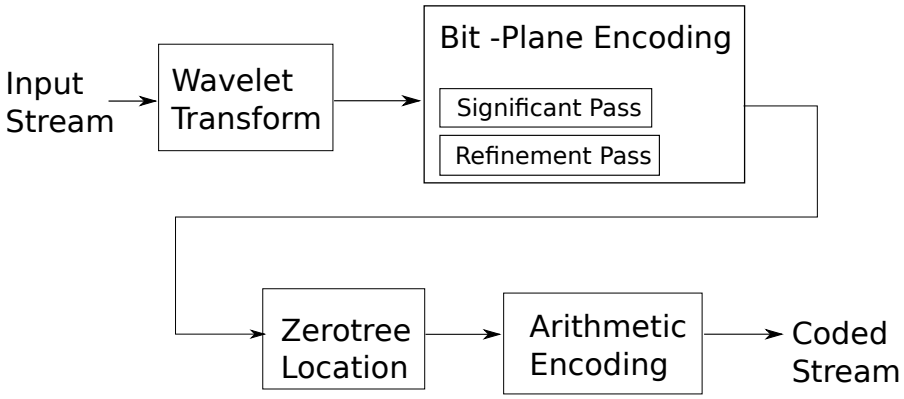


Figure 4.3: EZW Encoding Schema

subbands that represent the same spatial location in the image and this spatial relation can be depicted by a quad tree except for the root node at top left corner representing the DC coefficient which only has three children nodes. A quad tree is defined as a tree of locations in the wavelet transform. If the tree root is in [i,j] its children are located at [2i,2j], [2i+1,2j], [2i, 2j+1] and [2i+1,2j+1]. A quad tree can have multiple levels when a child is the root of another tree.

A zerotree is defined as a quad tree which, for a given threshold T, has insignificant wavelet transform values at each of its locations.

Once wavelet coefficient have been encoded, with respect to the current threshold, in ones (higher than T) and zeros (lower than T), the EZW exploits the zerotrees based on the observation that wavelet coefficients decrease with scale. It assumes that there will be a very high probability that all the coefficients in a quad tree will be smaller than a certain threshold if the root is smaller than this threshold. In this case the whole tree can be encoded with the zerotree symbol. Fortunately, with wavelet transform of natural scenes, the multi resolution structure of the wavelets does produce many zerotrees allowing to reduce notably the compressed data size. After identified the zerotrees, the data are encoded using an arithmetic encoding algorithm.

The set partitioning in hierarchical trees (SPIHT) algorithm is a highly refined

version of the EZW. It offers highest PSNR (Pixel Signal Noise Ratio) for given compression ratios, consequently it is probably the most widely used wavelet algorithm for image compression. Set partitioning refers to the way these quadtrees divide up, partition, the wavelet transform values at a given threshold. The analysis of this partitioning of transform values has been exploited to improve EZW algorithm.
SPIHT makes use of three lists: the List of Significant Pixels (LSP), List of Insignificant Pixels (LIP) and List of Insignificant Sets (LIS). These are coefficient location lists that contain their coordinates. After the initialization, the algorithm takes two stages for each level of threshold: the sorting pass (in which lists are organized) and the refinement pass (which does the actual progressive coding transmission). The result is in the form of a bitstream.
SPHIT uses a state transition model to encode zerotree information. Every index in the baseline scan order is assigned to a state depending on its value with respect the current threshold and the output of a significant function. From one threshold to the next the locations of transform values undergo state transitions. State transitions are coded with a smaller amount o¡f bits with respect the whole states: the states are four and only from one state all the others can be reached; the second and the third state can reach only two states while the last state is final. Encoding only the state transitions allows SPHIT to reduce the number of bits needed.
This algorithm performs better than other more sophisticated algorithms. For example it outperforms Jpeg both in perceptual quality and in terms of PSNR. Moreover it is less exposed to artifacts. It also is more efficient and more effective than EZW [17].

4.4 Video Compression Algorithms

Today's digital video coding paradigm represented by the ITU-T and MPEG standards mainly relies on a hybrid of block based transform and interframe predictive coding approaches. In this coding framework, the encoder architecture has the task to exploit both the temporal and spatial redundancies present in the video sequence,

which is a rather complex exercise. As a consequence, all standard video encoders have a much higher computational complexity than the decoder (typically five to ten times more complex), mainly due to the temporal correlation exploitation tools, notably the motion estimation process. This type of architecture is well-suited for applications where the video is encoded once and decoded many times, i.e., one-to-many topologies, such as broadcasting or video-on-demand, where the cost of the decoder is more critical than the cost of the encoder. On a sensor network the encoders have limited processing capacities while the decoders usually are run on machines out of the sensor network that can face major computational complexities.

Distributed Video Coding

The emerging technology for video streaming processing on wireless sensor networks is the Distributed Video Coding (DVC). The signals are captured independently by different sources and subsequently merged by a central base station that has the capability to jointly decode them.

The theoretical foundation of this technology can be found in the Slepian-Wolf coding theorem. It establishes that, if two sources are coded separately, provided that they are decoded jointly and that their correlation is known to both the encoder and the decoder, the lossless compression rate bound represented by the joint entropy H(X,Y) can be approached with a vanishing error probability[20]. Another contribution was given by Winer-Ziv studies. They showed that for correlated Gaussian source and a mean square error distortion measure, there is no rate loss in the independent coding of the sources with respect to the joint coding and decoding of the two sources [22].

One of the first applications in DVC is PRISM, by the Berkeley's group [14]. The algorithm divides the stream in group of pictures on which a complete cycle of coding decoding process is applied. The first frame of this group of pictures is encoded in a traditional way, for example with AVC/H.264 Intra [5]. This first frame is then used

to encode the remaining frames of the group with an hybrid technique, showed in Figure 4.4 which combines distributed and traditional coding. Each frame is split in 8x8 blocks and then transformed. Each block is considered as a separate unite and encoded independently from its spatially neighboring blocks. Then, the encoder estimates the current block's correlation level with the previous frame by using a zero-motion block matching. Further the blocks are classified into different encoding classes depending on the level of estimated correlation. The blocks with a lower level of correlation are encoded using conventional coding methods while high correlated blocks are not coded. The medium correlation blocks are encoded with a distributed approach where the number of bits used depends on the correlation level. The encoder computes syndrome bits used, at decoder side, to correct different predictors.
Another implementation is called Stanford codec. Video sequence is again split in groups of pictures: the first picture of the group is called *key frame* and is coded with traditional methods. The remaining frames are completely encoded in a distributed fashion passing through a quantization phase and then Turbo encoded. At decoder side intermediate frames are decoded by interpolation between the key frames. The decoder corrects its prediction using parity bits received from the encoder.

Interframe Video Compression

Interframe compression includes those techniques applied to a sequence of video frames and not only within an image. Interframe compression exploits the similarities between successive frames, known as temporal redundancy, to reduce the volume of data required to describe the sequence.
The most established and implemented strategy is the block based motion compensation technique, employed in MPEG or ITU-T video compression standards. This technique is based on motion vector estimation: the image is divided into disjoint blocks of pixels and each block is compared to areas of similar size of the previous frames to find an area that is similar. A block from the current frame for which a similar area is

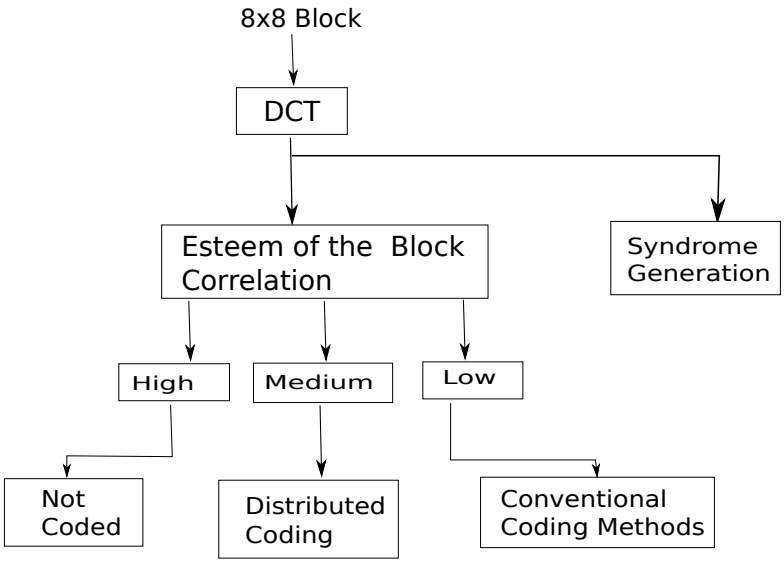


Figure 4.4: PRISM Encoding Schema

sought is known as a target block. The location of the similar or matching block in the past frame might be different from the location of the target block in the current frame. The relative difference in locations is known as the motion vector. If the target block and matching block are found at the same location in their respective frames then the motion vector that describes their difference is known as a zero vector. In the coding phase, instead coding blocks that simply moved through the image, the encoder codes the motion vectors describing such a movement. During decompression the decoder uses the motion vectors to find the matching blocks in the past frame and copies those blocks in the right position.

The effectiveness of compression techniques that use block based motion compensation depends on some assumptions: objects move on a plane that is parallel to the camera plane, illumination is spatially and temporal uniform and occlusion of one object by another does not happen.

Another interesting technique often adopted for its simplicity is Differential Pulse Code Modulation(DPCM). This coding system merges predictive coding and scalar

quantization. The technique exploits the correlation between two subsequent frames coding the prediction residual. This residual, due to inter-symbol correlations assumes small values with high probability, thus having a smaller variance than the source. With continuous signals this small variance allow to reduce the quantization error variance, that is proportional to the variance of the quantizer input. Moreover it is possible to design adaptive quantizers that compensate slowly varying input signal power by dynamically scaling the quantizer output.

DPCM is also suitable for discrete signals as video sequences. The basic idea exploits again the correlation between closest frames but this time predicted residuals have not to be quantized. The residual coefficients are zeros or near zeros values and constitute the ideal input for the subsequent compression phases as, for instance, transforms and run length coding. Due to the proximity and the smallness of those values the mentioned techniques can achieve better compression results exploiting the reduced amount of information carried by the data set. A schema of the DPCM encoder/decoder is showed in Figure 4.5.

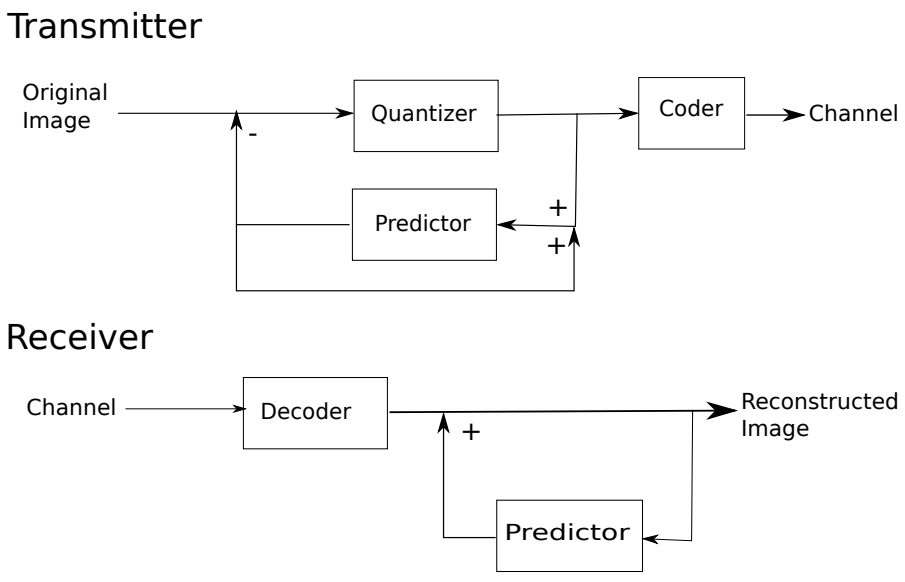


Figure 4.5: DPCM Transmitter and Receiver Schema

Chapter 5

Hardware and Software

In this Chapter the hardware and software components used to implement the video streaming on the sensor network will be presented in details.
An accurate analysis of the hardware features of the adopted sensors was necessary for the developing of the entire system. Quality parameters and performances of the components had to be considered to avoid bottlenecks and to tune properly all the variables involved.

5.1 Crossbow Intel Mote 2

The Intel Mote 2, Figure 5.1, is an advanced sensor network node platform designed for demanding wireless sensor network applications requiring high CPU/DSP and wireless link performance and reliability.
The platform is built around a low power XScale processor, PXA271. It integrates an 802.15.4 radio (ChipCon 2420) and a built in 2.4 GHz antenna. It exposes a "basic sensor board" interface, consisting of two connectors on one side of the board, and an "advanced sensor board" interface, consisting of two high density connectors on the other side of the board. The Intel Mote 2 is a modular stackable platform and can be stacked with sensor boards to customize the system to a specific application, along

with a “power board” to supply power to the system.

Figure 5.1: Intel Mote 2

Processor

The Intel Mote 2 contains the PXA271 processor. This processor can operate in a low voltage (0.85V) and a low frequency (13 MHz) mode, hence enabling low power operation. The frequency can be scaled to 104 MHz at the lowest voltage level, and can be increased up to 416MHz with Dynamic Voltage Scaling (DVS). Currently DVS is not supported by TinyOS 2.x and the voltage level can only be set to 13, 104 or 208 MHz. The processor has many low power modes, including sleep and deep sleep modes. It also integrates 256 KB of SRAM divided into 4 equal banks of 64 KB. The PXA271 is a multi-chip module that includes three chips in a single package, the processor, 32 MB SDRAM and 32 MB of flash. The processor integrates many I/O options making it extremely flexible in supporting different sensors, A/Ds, radio options, etc. These I/O options include I2C, 3 Synchronous Serial Ports one of which dedicated to the radio, 3 high speed UARTs, GPIOs, SDIO, USB client and host, AC97 and I2S audio codec interfaces, fast infrared port, PWM, Camera Interface and a high speed bus (Mobile Scaleable Link). The processor also adds many timers and a real time clock. The PXA271 also includes a wireless MMX coprocessor to accelerate multi-

media operations. It adds 30 new media processor instructions, support for alignment and video operations and compatibility with Intel MMX and SSE integer instructions.

Radio

The Intel Mote 2 integrates an 802.15.4 radio transceiver from ChipCon (CC2420). 802.15.4 is an IEEE standard describing the physical and MAC layers of a low power low range radio, aimed at control and monitoring applications. The CC2420 supports a 250 kb/s data rate with 16 channels in the 2.4 GHz band. The Intel Mote 2 platform integrates a 2.4 GHz surface mount antenna which provides a nominal range of about 30 meters. If a longer range is desired, an SMA connector can be soldered directly to the board to connect to an external antenna. Other external radio modules such as 802.11 and Bluetooth can be enabled through the supported interfaces (SDIO, UART, SPI, etc). The power levels available with this radio chip are described below together with the current consumptions.

Power Level	**Output Power [dBm]**	**Current [mA]**
31	0	17.4
27	-1	16.5
23	-3	15.2
19	-5	13.9
15	-7	12.5
11	-10	11.2
7	-15	9.9
3	-25	8.9

Table 5.1: CC2420 Power Levels

Power Supply

To supply the processor with all the required voltage domains, the Intel Mote 2 includes a Power Management Integrated Circuit(IC). This PMIC supplies 9 voltage

domains to the processor in addition to the Dynamic Voltage Scaling capability. It also includes a battery charging option and battery voltage monitoring. Two of the PMIC voltage regulators (1.8 V and 3.0 V) are used to supply the sensor boards with the desired regulated supplies at a maximum current of 200 mA. The processor communicates with the PMIC over a dedicated I2C bus (PWRI2C). The Intel Mote 2 platform was designed to support primary and rechargeable battery options, in addition to being powered via USB.

5.2 Crossbow IMB400

The IMB400, Figure 5.2, adds multimedia capabilities to the Imote2 platform. It allows for capturing images, video and audio as well as for audio playback. All data is digitally captured for storage, transmission or further processing on the Imote2 main-board. In addition, the IMB400 features a PIR sensor for platform wake-up from sleep if movement is detected.
The IMB400 attaches to the advanced connector set of the Imote2 main-board.

Figure 5.2: IMB400 Module

Specifications

The camera sensor supports different resolutions. It can acquire VGA, QVGA, CIF and QCIF images with RGB, YCbCr or YUV formats. Moreover it supports hardware image scaling and filtering and drivers allow to tune some of the camera acquirement parameters such as automatic exposure, gain, white balance and black level. Image controls include also saturation, hue, gamma and sharpness.
The IMB400 camera chip is the OV7670 VGA from Omnivision. The OV7670 can operate at 30 frame per seconds (fps) in VGA with full user control over image quality, formatting and output data transfer.
The audio codec supports sampling rates up to 48 kHz in mono mode. Signal to noise ratio is major than 94 dB while the total harmonic distortion (THD) is less than -80 dB. It allow also to set programmable filters for noise suppression.
It also includes a microphone and a line input, an on board miniature speaker and a line output.
The passive infrared sensor (PIR) supports a maximum range of almost 5 meters and detection angles of 80-100°.

5.3 TinyOS

TinyOS [3] is an open source operative system developed by the University of California at Berkeley to develop WSN software. The main difference with the common operative systems is that TinyOS is not based on a *kernel* that interface all the components. Instead it offers a direct access to the hardware. Furthermore it has been developed thinking to the low power and low resources WSN environment.
In order to cope with the severe hardware constraints of sensor nodes, TinyOS only allows for static memory allocation. This makes it very space and time efficient because there is no need for maintaining an additional data structure managing the dynamic heap. The only necessary data structure is the call stack keeping track of local vari-

ables and a few pointers. The first version of the operative system was released in the October of 2002 and was followed by many updates until version 1.15, released in the June of 2005. This first version presented some limitations due to the not very intuitive structure and the high dependency of many components that obstructed the reuse of the code.
The need of a more modular system brought to the 2.x version of TinyOS. The software was completely redesigned to create a new system, adaptable to different hardware platforms, more simple and modifiable.
In this thesis work it has been developed the video streaming system on the 2.1.1 source code.
TinyOS is implemented with the *NesC* programming language. It has been created for the developing of embedded systems applications. Because an embedded system has functionalities already known at design time, it could be built keeping in account hardware as well as software needs, allowing to develop a more efficient product.

5.4 NesC

NesC is a dialect of the C programming language. It offers a system to create and assembly modular components to obtain robust and easily modifiable embedded systems.
The main features of this language are:

- *Construction and composition separation:* the development of a NesC application is based on the assembling of a series of components that can be pre existent or ad-hoc created. Every component is assembled through configuration files and must define and implement its specifications.

- *Component specifications through interfaces:* a component must declare external interfaces used and those provided by itself. Because every interface is defined as a set of function prototypes it result easier to reuse the code for new

applications.

- *Bidirectional interfaces:* other than the functions provided by the component, the latter must specify a series of events to be managed by the user of that module.

- *Static structure:* the graph interconnecting the components is static and can not be modified run-time.

- *Concurrency:* the NesC model is compatible with the TinyOS concurrency model and consists in processes that execute their activities on shared data in the same time.

In NesC language they are defined three types of functions: *commands*, *events*, *tasks*.

Commands and Events

A component can require services offered by another component through commands. Commands are functions declared in the interface definition which accept input parameters and output a return value.
Events, instead, are generated by a module and must be managed by the application that uses the module. They represent hardware interrupts managers. The interrupts can take place due to external factors such as the reception of a message or for internal reasons like, for instance, a timer expiration or the dispatch of a packet. The concurrency behavior of both commands and events can be synchronous or asynchronous. Asynchronous commands or events can be generated by hardware interrupts and their execution starts once the interrupt has been triggered, stopping all the other operations. The execution will be restarted from the point where it was interrupted once the interrupt manager finished its work. On the other side, commands or events declared as synchronous (default behavior) can not manage hardware interrupts and can be called only by tasks.

Tasks

Task are pieces of code that can be executed without being blocked by other synchronous commands. A scheduled task is put in the tasks queue. Here the elements of the queue are executed with a First In First Out (FIFO) policy. This kind of function can execute other tasks, activate commands or generate events. Moreover the non-preemptive nature of this function allow to avoid race conditions on shared variables. The only functions that can preempt a task are those explicitly defined as asynchronous with the *async* keyword.

Interfaces and Components

Interfaces are files describing functions and events the module associated with the interface respectively provides and triggers. This paradigm allow to easily write modular code, modifiable and reusable.
In NesC language a component can be a *module* or a *configuration*. Modules implement the functions and events described in the interface. In the configurations it is declared how the components are connected with each other and can also provide interfaces.

5.5 Hardware Abstraction Architecture

Hardware Abstraction Architecture (HAA) for TinyOS 2.0 is an hardware abstraction model that balances the conflicting requirements of code reusability and portability on the one hand and efficiency and performance optimization on the other. It is composed by three main parts:

- *Hardware Presentation Layer (HPL)* This layer is positioned directly over the HW/SW interface. It includes modules that manage interrupts directly and modify register values. The HPL components sh expose an interface that is fully determined by the capabilities of the hardware module that is abstracted. The

interrupt service routines in the HPL components perform only the most time critical operations (like copying a single value, clearing some flags, etc.), and delegate the rest of the processing to the higher level components that possess extended knowledge about the state of the system.

This HPL does not provide any substantial abstraction over the hardware beyond automating frequently used command sequences. Nonetheless, it hides the most hardware-dependent code and opens the way for developing higher-level abstraction components. These higher abstractions can be used with different HPL hardware-modules of the same class.

- *Hardware Abstraction Layer (HAL)* This layer is still dependent on the hardware layer and masks to the upper layers the hardware complexities. In contrast to the HPL components, they are allowed to maintain state that can be used for performing arbitration and resource control. Instead of hiding the individual features of the hardware class behind generic models, HAL interfaces expose specific features and provide the ”best” possible abstraction that streamlines application development while maintaining effective use of resources.

- *Hardware Interface Layer (HIL)* The final tier in the architecture is formed by the HIL components that take the platform-specific abstractions provided by the HAL and convert them to hardware-independent interfaces used by cross-platform applications. These interfaces provide a platform independent abstraction over the hardware that simplifies the development of the application software by hiding the hardware differences.

5.6 Tossim

In the distribution of TinyOS considered in this thesis, a simulator of a sensor network is implemented. This simulator, called Tossim, becomes useful in diverse aspects of the software development phases: first of all it comes in help as a support in the de-

bugging phase. Because of the embedded nature of the sensors, it is often hard to clearly understand what is going on at run time, therefore simulating the buggy code can save much time. On the other hand, before implementing the code on the sensors it could be useful to simulate the behavior of the system to decide, for instance, in which way it must be developed.

Tossim simulates entire TinyOS applications, it works by replacing components with simulation implementation. The level at which a component is replaced is very flexible, it can simulate a packet level component as well as a low level radio chip.

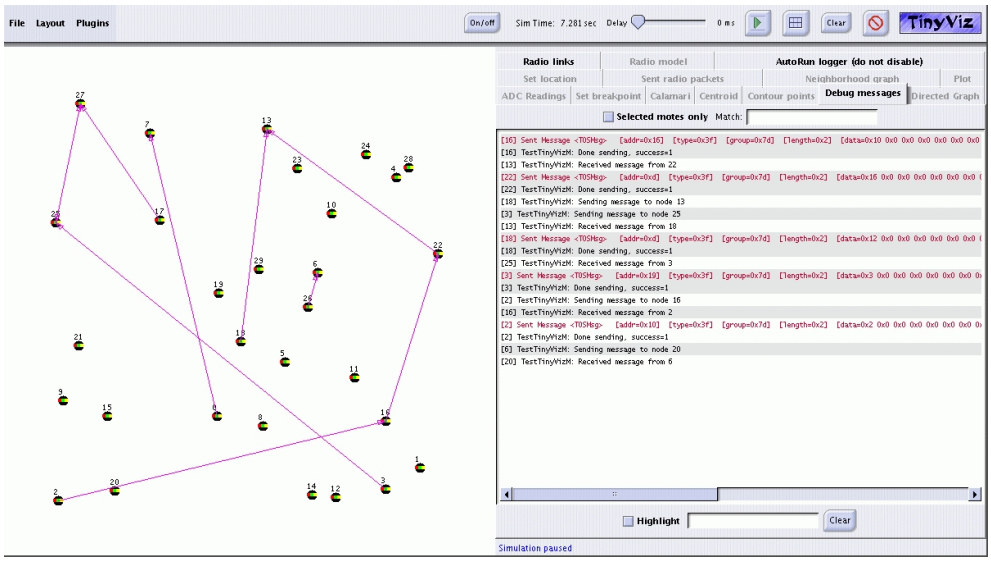


Figure 5.3: TinyViz Screenshot

Tossim is a discrete time simulator, it pulls event on an events queue and executes them. Simulation events can represent hardware interrupts as well as high level system events. Also tasks are simulation events.

Indeed Tossim is a library, it is necessary to run a program that configure a simulation and runs it. Supported languages are Python and C++.[2]

By default Tossim captures TinyOS behavior at a very low level, for instance it simulates the network at a bit level and replicates each individual ADC capture and each interrupt of the real system but it does not model the real word. Instead it implements

abstractions of real world phenomena. For example, instead modeling radio propagation, it provides abstraction of direct independent bit errors between two nodes; it also does not model draw or energy consumption, however it is simple to add annotations to components that consume power to provide information on when their power state changes. After the simulation is run a user can apply a model to these transitions.

There exists a Java visualization and actuation environment for Tossim called *TinyViz*. It is not only a visualizer but also a framework where plugins can provide desired functionalities. Plugins can be dynamically registered and deregistered. For example, when in Tossim a node sends a packet, a networking plugin can listen for packet send events and update TinyViz node state and draw an animation of the communication.

Chapter 6

Video Streaming System Implementation

In this Chapter they will be described the diverse phases that led to the development of a video streaming system on a wireless sensor network.

Initially it was necessary an accurate analysis of the already available software for compression and transmission of the data. Moreover it was examined the state of the art reached in developing low power and low complexity compression algorithms to establish whether other methods could fit better the requirements.

Most of the time spent in this work was taken by the code implementation of the system, also because debugging operations of embedded systems presents a lot of difficulties. Furthermore TinyOS and the used hardware had some unexpected behaviors in particular situations. This abnormalities required adjunctive efforts to be understood and fixed.

6.1 Architecture Of The Existent System

The initial system architecture is constituted by one multimedia node directly connected to the gateway through a serial cable. The components constituting the mote are the Intel mote 2 node an IMB400 multimedia board and the programming board used for serial communication.
The software already available from previous works is made of three major parts: the image acquisition and compression module, a serial transmission module and the Java application used to interact with the user. The first two modules implement sensor side operations while the third handle the data on the gateway.
On the sensor side it is possible to conceptually divide the code in data processing code and transmission code. The former manages the multimedia section of the sensor setting camera parameters, acquiring the raw array of pixels and eventually compressing the image for the delivering. The transmission code divides the image array in bursts and sends them on the serial cable to the gateway.
The gateway is any multi purpose machine supporting the Java environment and running applications devoted to data collection and display.

Implementation

The original implementation belonged to the TinyOS code contributions[1] lightly reworked to add the possibility of decompress and save received Jpeg images.
The cameraJpegTestSerial application, as the original implementation is called, is composed by two modules devoted to image processing: cameraJpegTestM.nc and JpegM.nc. In the camera module there are primitives to receive commands from serial channel and to acquire pictures in the format specified by those commands. The available commands allow to choose between color and grayscale images and between QVGA and VGA formats.
The JpegM.nc module implements the code for Jpeg compression of the original image through discrete cosine transform, quantization, run length encoding of the zeros

and Huffman encoding.

Transmission operations are implemented in the SendBigMsgM.nc module where the image array received from cameraJpegTestM is divided in 64 bytes bursts and each packet is sent on the serial channel to the Java application running on the gateway.

There were two Java applications the computer that had to elaborate data needs to run. One is the Serial Forwarder program that is used to directly read packets incoming on the serial port to subsequently handle or simply observing them. The other application, called CameraGUI, provides a graphical environment to easily interact with the camera mote. This program allows to select picture resolutions and formats and to send an acquire command to the mote. In Figure 6.1 is described the cameraGUI behavior on the reception of data packets. Once the image has been sent, the program receives the packets forwarded by the Serial Forwarder, reconstructs the initial image array and saves it on disk adding the *png* or *ppm* header. If the image received is compressed the CameraGUI application exploits the C decoder coupled with the JpegM.nc compression module to reconstruct the image in a human readable format.

6.2 Development Of The Video Streaming System

The development phase of the system presented in this work can be divided in three steps. The first part focuses on the compression algorithm and its behavior. Then it is implemented a first version of the video streaming on the mote allowing only the serial and direct communication with the gateway. In the last phase, it is implemented the radio communication: initially the mote can send data only with a direct radio connection to the central computer, then it is added the multihop support with a static routing. The entire work required an evaluation on the techniques to adopt in the design of the system. This included mainly the compression technique used to reduce the data sent on the network. Due to the fact that an optimized version of the Jpeg compression algorithm was already implemented and that no method were found during the mentioned research that overcome entirely such Jpeg implementation, it is decided to

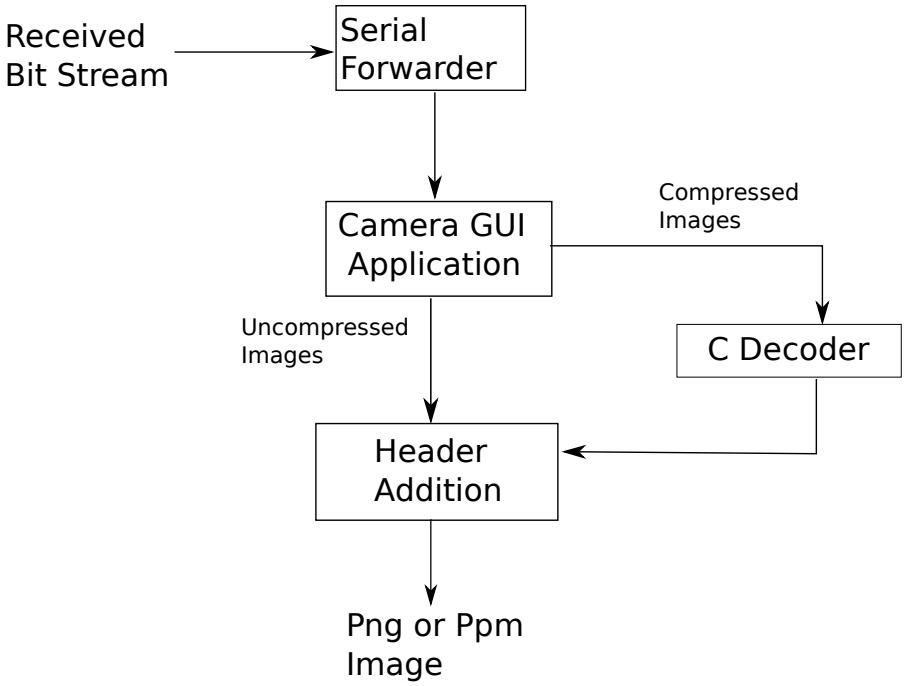


Figure 6.1: Camera GUI On Receipt Operations

use the existent compression system as the basis on which build the video streaming application.

6.2.1 Analysis Of The Existent System

The core of the starting system is the compression algorithm. It is based on the Jpeg standard and foresee the following steps. Initially it receives in input an array containing the image matrix read with a per row order. After some initialization operations, from the input data are generated discrete cosine transform coefficients applying a fast version of the DCT algorithm and quantizing the coefficients through a quantization table. These operations outputs a new array of coefficients with values cut to fill a one byte signed integer variable except for the DC coefficient that can assume values between 0 and 255.

The set of data elaborated by the fast DCT algorithm goes then in input to the run

length encoder. The original implementation of the zeros encoder parsed the array counting how many zeros each sequence of consecutive zeros have and saving that value in a counter. The maximum counter value is limited to 128 while all the other coefficients, not involved in the coding, are cut to a seven bit signed value (-64, +63). This happens because the implementation of the algorithm foresee to mark with the most significant bit set to one the coefficients not coded while the counters exposed zero as first bit.

This coding procedure, even if very efficient in the way it distinguishes coefficients and counters, cuts coefficients major than 63 and minor than -64. The output of the DCT, already cut to 8 bit signed values, is constituted by few high magnitude values representing the low frequencies and many zero or low value coefficients representing details of the images. Using the described run length encoding technique many of the most important low frequencies values would be cut so that the image loses quality. For this matter the coding algorithm, showed in Figure 6.2, has been reimplemented in a way that avoid coefficients cuts. Now, counters are preceded by a zero while untouched values can be coded with a number between -128 and 127. The byte used for signalling a counter is partially retrieved by the extension of the counter range from 128 to 255.

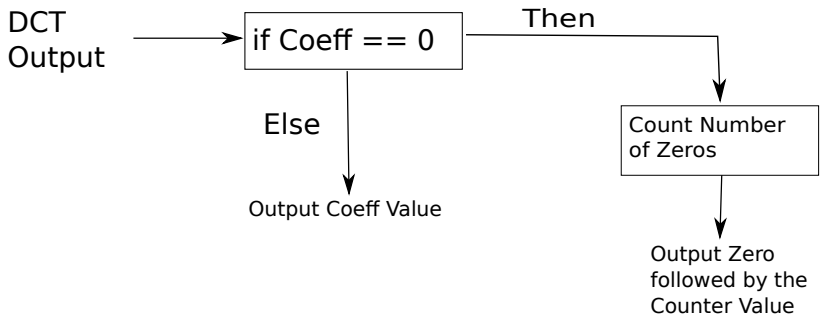


Figure 6.2: Run Length Encoding

The tests done with the new implementation showed a behavior varying from image to image. Substantially the compression rate is maintained with a difference between the

two methods in order of the hundreds of bytes in the compressed data size. The number of values that would be cut by the old algorithm is not very high, the percentage is around the 1% of the all coefficients but those values contains the major part of the image information.

The data exiting from the run length encoding are then Huffman encoded to further reduce the transmitted information.

Another series of code manipulations concerned the C implementation of the decoder. As previously said, the images coded on the sensors through the Jpeg *nesC* implementation are decoded gateway-side using the correspondent decoder, implemented in C programming language. Initially this implementation supported only QVGA format images and had a little bug in the code images decoding. In this first phase of the work it is added the support for VGA decoding extending the existent one and some bugs has been fixed.

Subsequently it has been necessary to tune the camera parameters to obtain the best quality from the device. This step required a large set of experiments to test many of the different configurations allowed for the camera chip. The main parameters set include gamma curves values, dimensions of the acquire window, scaling parameters, etc. Moreover it has been implemented the support for another image format: QCIF, useful to reduce the data flow of the video stream in the initial stages of the development.

Before setting these parameters the image chip output was shifted with respect the actual image acquired and therefore some columns on the extreme right of the image were showed on the extreme left. Also this problem has been solved by correcting chip's settings. An example of a shifted image is showed in Figure 6.3

The serial communication part is substantially left unchanged. The image array is divided in data segments of 64 bytes. Every packet contains a 16 bit ID to allow order reception checking duplicate or lost packets. At this time the system is capable to communicate on the serial channel to the gateway supporting QVGA, VGA and QCIF formats and the Jpeg compression of the images.

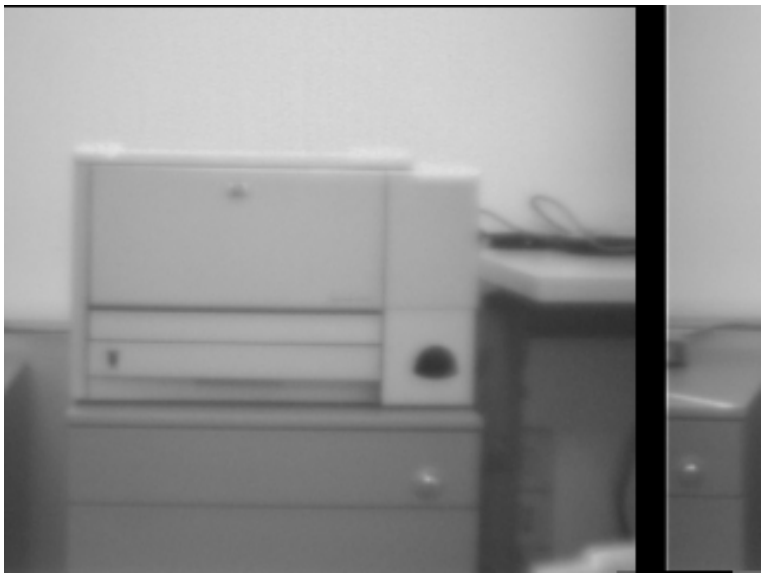


Figure 6.3: Example of a Shifted Picture

The general behavior of the existent system is the following. An image acquisition starts with the correspondent command message sent by the CameraGUI application. Once received by the camera mote they are set the parameters and the variables necessary for image acquisition and processing as, for instance, image resolution and format and it is called the acquire task that has in charge to fill a buffer, located in the *sdram* memory, with the matrix of the acquired picture read row per row. The resulting array is then processed by the image processing task to reduce the number of bytes per pixel, if the raw image is sent, or to compress the data. Subsequently it is passed to the transmission module where the data array is fragmented and sent in multiple packets. The steps of the picture acquire operation are showed in Figure 6.4

It must be said that the color images acquisition is not optimal yet. The read of the different colors is not well implemented and the resulting images show chromatic inconsistencies. For this matter and with the aim of limit at most the transmitted data size, the video streaming system is based only on grayscale images.

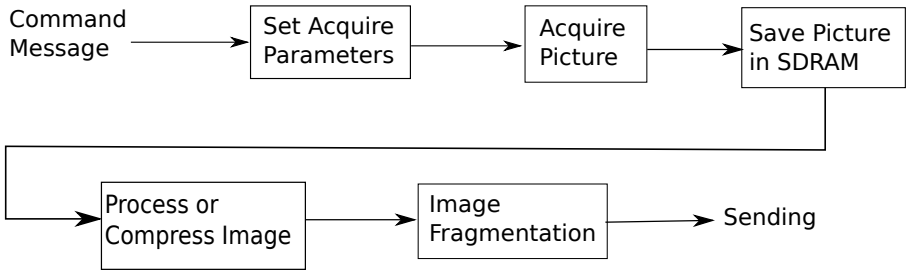


Figure 6.4: Picture Acquire Operations

6.2.2 Video System Implementation

Once obtained a working environment for pictures acquisition, it starts the design of the algorithm implementing the video system.
The first image format adopted is, as said, grayscale image type with a QVGA resolution corresponding to a window of $320x240$ pixels. Each pixel is coded with only one byte.
Working with limited band capacities, it was necessary to employ some kind of reduction of the amount of data transmitted. After a research on the state of the art of video coding techniques, due to the fact that the intraframe coding technique (Jpeg) is already implemented and that a light and efficient interframe coding method is needed, it has been decided to develop a DPCM algorithm with the residual given by the difference between previous and current frame.

DPCM

The developing of this system passed through three different implementation steps: initially a Matlab version of the encoder and the decoder functions was produced, exploiting the Matlab framework to obtain in a short time a functioning model of the system. Subsequently the code has been translated to C implementing all the functions in a language very close to the final one. The last step consisted in bringing the C code on the sensor adapting it to the nesC dialect. The encoder was thus implemented on

the sensor attached to the camera, in the DpcmM.nc module. With this file the configuration file DpcmC.nc and the interface declaration Dpcm.nc were also associated. The DpcmM.nc file contains one main function called *dpcm_encode* that does all the work taking in input the addresses of the arrays containing the image to process and the result of the elaboration, the dimensions of the image, the Jpeg quality parameter, the number of the current frame, the maximum bandwidth value and the buffer address.

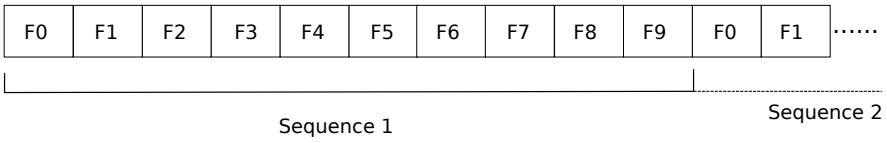


Figure 6.5: Video Streaming Frames Sequences

The function behavior is different with respect to the kind of frame processed. The frame number zero, f_0 is always treated as a common Jpeg image while the other frames $f_x \quad x = 1..9$ are coded using the inter frame correlation between them and the previous one. The frame sequence is periodic with a ten frame period and is showed in Figure 6.5. The frame number in input to the function is considered after calculating the modulo 10 operation on it. However this is a parametric value, tunable on the necessities.

Once the f_0 frame is coded, the original not coded frame is saved in a buffer for later use. Instead encoding the second frame entirely, it is first subtracted to the content of the buffer containing f_0 and then only the residual r of this operation is compressed with the Jpeg algorithm described above. This allow to diminish the information content of the data set so that it can be reached a better compression ratio. After the coding of the differences between f_0 and f_1, those differences are summed to the content of the buffer b to obtain the same prediction available at the decoder. Now the buffer b contains the predicted values of the f_1 frame. Such content will be subtracted to the f_2 frame to send only the differences r. The process is repeated until the f_10 frame for which the buffer content will be ignored coding such frame as a Jpeg image.

Before coding the frame array with DCT it is performed another operation to reduce further the data size. The image array is subsampled coding each four pixels block with a coefficient obtained averaging the four pixels values.

On the decoder side the process is inverted. The first frame, f_0 is decoded as a simple Jpeg image and the result of the decoding process is saved in a buffer. Subsequently the output of the inverse DCT, run length and Huffman uncompress operations for all the f_x frames of the sequence is summed to the buffer content to reconstruct the original frame content. The first frame of the subsequent sequence, f_10, is not summed to the buffer content and is entirely stored in such buffer. Before displaying the image, the picture resolution is increased to QVGA format reconstructing the missing pixels by bicubic interpolation. As on the encoder side the stand alone frame recourse every ten pictures.

The entire video coding and decoding process is showed in Figure 6.6 and 6.7.

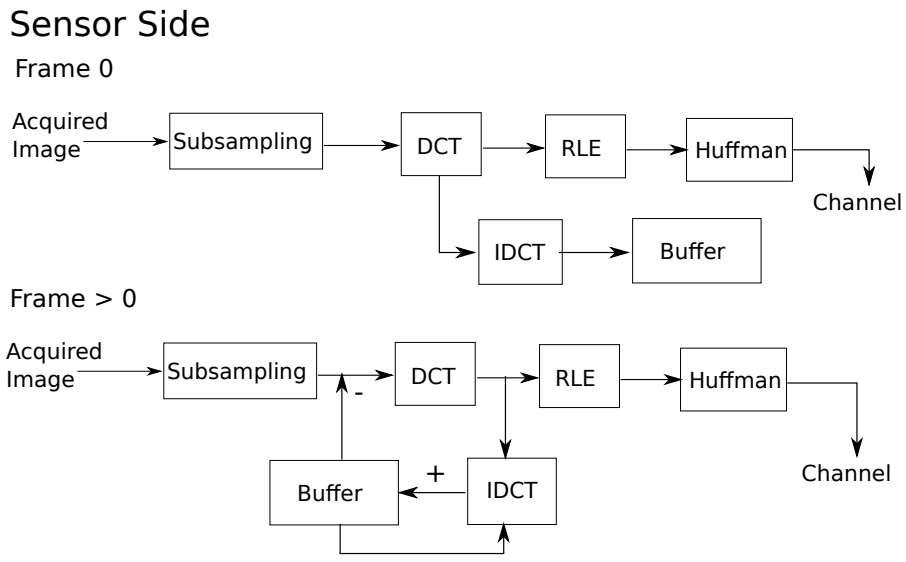


Figure 6.6: Video Coding Process

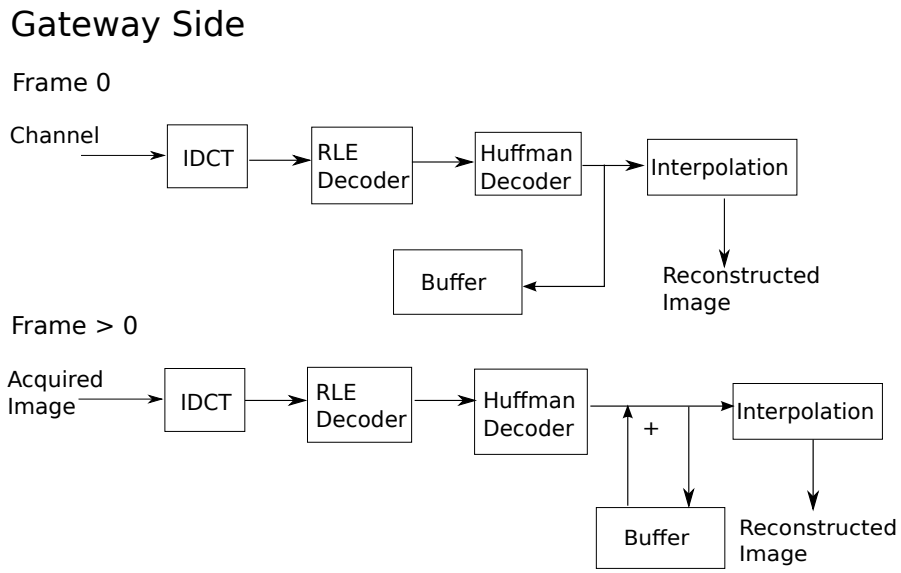


Figure 6.7: Video Decoding Process

Extension Of The CameraGUI Software

The introduction of the video functionality required the implementation of the relative support in the Java environment to allow the interaction between the user and the sensor network and to display the video stream. The original CameraGUI application was split in a multi thread program to support the constant flow of information incoming and the contemporary elaboration of such data.

Alongside the existing process they are added two threads: one called receiver thread while the other is named display thread.

The receiver thread has in charge to receive the data packets from the Serial Forwarder and inspect the payload content to ensure the correct packet sequence within the frame is respected, furthermore it has to control the frame number in such a way that the entire frames can be decoded as simple Jpeg pictures while intermediate frames are decoded as a difference and then summed to the buffer content. The structure of a video packet is showed in Figure 6.9. To do this the thread is synchronized with the

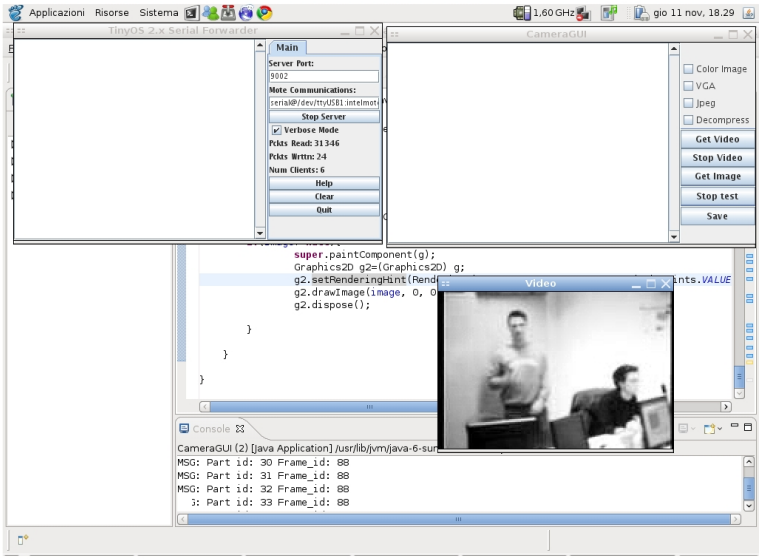


Figure 6.8: Screenshot of the Gateway Side Application

camera sensor node that is sending the packets. Every correct frame received a counter is incremented so that the DPCM decoder could know the position of the frame within the sequence and consequently its type. A correct frame is a frame for which all the parts in which the sender divide the coded data for the delivering are received without errors. If a frame is not received correctly the subsequent frames must be discarded until the synchronization is restored. The corruption of a frame is generally due to the loss of one or more packets that compromises the correct construction of the entire frame. All the subsequent frames depending on the discarded one become invalid because the buffer is corrupted.

The synchronization is naturally restored with the correct reception of the subsequent *frame zero* that does not depend on the previous ones. This could affect the performances of the system because some transmitted data became completely unuseful. To reduce this time lost the algorithm uses a mechanism to anticipate the sending of the entire frame. When the receiver thread finds that the sequence of the inner parts of a frame or the frames flow are broken, it stops saving the incoming data and send a command message to the camera mote. On the reception of this message, the camera mote

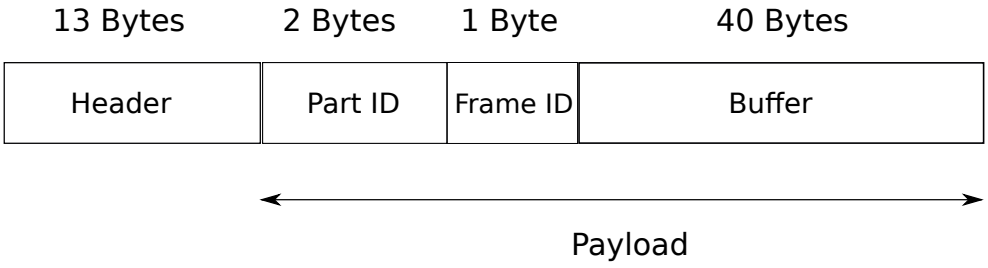


Figure 6.9: Video Packet Schema

forces the restart of the frames enumeration, acquiring a full Jpeg frame and marking it with frame id zero.

All the correctly received frames are saved on disk and the path to every file is inserted in the *frameBuffer* array. Every time the sequence is restarted unexpectedly the current position of the array is set to null while the path of the first frame of the reset sequence is saved in the next position. As illustrated in the following this mechanism allow to easily synchronize the display thread.

The display thread reads the files from the disk, decode properly the different frames and display those using the Java Swing libraries. The receive thread and the display thread read and write respectively the *frameBuffer* with synchronized methods. Once the display thread acquires the path to the current file to be elaborated, it passes such path to the DPCM decoder together with the frame's number. This number is used by the decoding algorithm to discriminate between the whole Jpeg pictures and the coded differences as it is done during the encoding process. Afterward, the decoded image is acquired reading the decoder's output and displayed through the Java Swing libraries.

As the receiver thread has to be maintained synchronized with the camera node, in the same way the display thread must be aware of the interruption of the sequence and the subsequent restart. This is achieved, as described before, inserting a null value in the *frameBuffer* every time the frames flow brakes. In this way, reading the null value, the display thread knows when to reset the frame counter so that the current frame can be

decoded properly. A schema of the receiver and display thread interaction is presented in figure 6.10

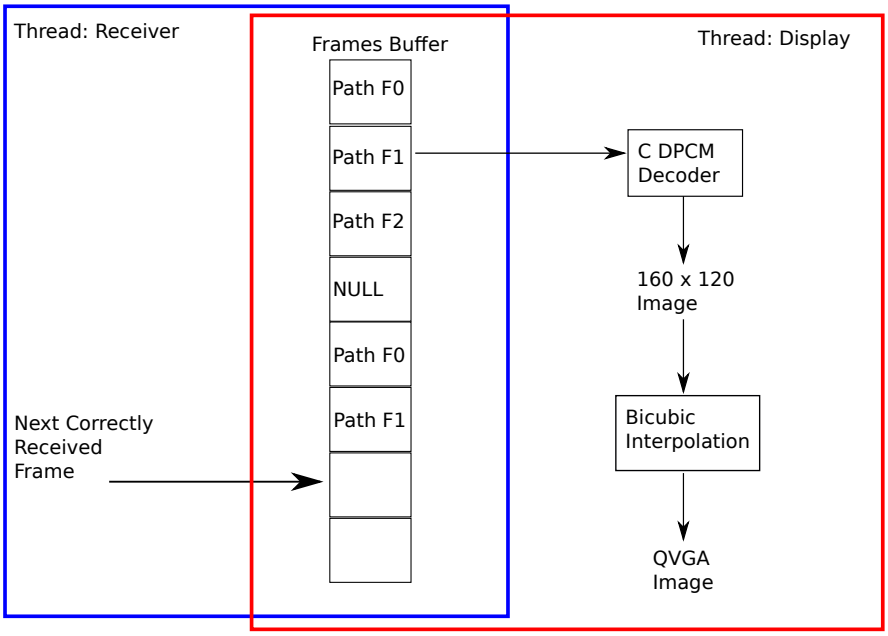


Figure 6.10: Interaction of Receiver and Display Threads

The display thread is started only when the *frameBuffer* contains a predetermined number of elements. Once started it consumes the content of the buffer decoding and displaying it and then the thread is send in sleep mode for 450 milliseconds. After this sleeping time it wakes up, read the second position of the buffer and elaborates it, and so on. This guarantees a discreetly fluent video reproduction but consumes the buffer content too rapidly with respect to the buffer refilling speed that is always slower even on the serial channel. Once the *frameBuffer* is empty the display thread stops calling a *wait()* primitive on the *frameBuffer* object and pauses until the number of buffer elements does not exceed the defined refilling threshold. When the latter is overcame the receiver buffer calls a *notify()* on the same object and the display thread restarts its task.

Extension Of The Node Transmission Module

To support the transmission of the new data flow it has been necessary to extend also the communication module of the camera mote. The existent module allowed only the transmission of images through the fragmentation of the image array in multiple packets and the sent of these packets on the serial channel. To add the possibility of sending video frames the mote has been equipped with functions similar to those of the image sending. The main difference is the packets size, extended from 64 to 100 bytes to reduce the number of packets transmitted on the channel and thus the network traffic. Another modification concerned the packet format. As for the image case, messages contain a buffer of unsigned 8 bits integers that brings the bursts of the actual data yield by the camera. While in the photo messages the only ID required for the packets numbering is the 16 bit part identification number within the fragmented sequence, here it is also necessary to distinguish the different frames and guarantee the correct order delivering. To do this it is inserted another 8 bits ID reset to zero every 250 frames.

Design Issues

The problem encountered during this developing phase are mainly two. Both concern the video fluidity: the first one affects the image elaboration process while the second concerns the transmission operations.
The video system is slightly different than the image acquisition process described before Here, in fact, it must be acquired and sent an arbitrarily long sequence of frames. The series of images must be taken with a higher rate if compared to the single image acquisition where the subsequent picture could be acquired only after the previous one was completely received by the gateway. Moreover, smaller is the acquiring (and the processing) time, earlier the system could start sending the data. These considerations do not find confirmation in the software behavior which apparently could acquire at the really slow rate of two frames per seconds. Inspecting the source code of the camera it

was found a piece of code implementing a busy wait that stopped the whole acquiring process to allow the camera chip to settle after the image acquisition. To be able to remove that busy wait the mote's working frequency had to be augmented from 13 MHz to 208 MHz. Subsequently that frequency was set to 104 MHz due to transmission problems. This allowed to speed up the image acquisition rate to almost 5 frames per second and had also the benefit of reducing image processing time. Obviously all of this was achieved at the price of an increased energy consumption.

The second problem encountered in the developing of the system concerned the volume of data sent by the multimedia mote. Every frame was composed of 76800 bytes that compressed and fragmented in 100 bytes bursts have an average size of 50-60 packets for the full frames and 25-35 packets for coded differences. Moreover in case of highly dynamic scenes the size of the intermediate frames could increase notably and the size of the images augments proportionally to the number of details captured by the camera.

With this characteristics the frame transmission took too much time and the frame rate of the display thread must be set to a value major than one frame per second to be comparable to the arrival rate of the frames. Because a video stream with that frame rate is totally unacceptable it raised the need of reducing transmitted data size. The first solution adopted interested video resolution that passed from QVGA to QCIF with a window of $176x144$ pixels, each coded by one byte. In this way the transmission time is consistently reduced and the reproduction rate can be augmented till almost three frame per second. This time the price of the problem fixing has been a notable reduction of the displayed video's window size.

To overcome also this limitation it was used a well known technique called subsampling. The basic idea underneath this fixing is to acquire a complete QVGA image, reduce the coding to one byte per pixel as for all the processed images and then reducing further the image size dividing the matrix in groups of four pixels and sending only the average value of these four pixels. This technique halves both the dimensions of the original matrix containing the image. The image delivered to the gateway has

now a $160x120$ resolution. To restore the original image size it is used a method offered by the Java Swing library that reconstructs a QVGA image through a bi-cubic interpolation technique with a tolerable quality loss.

6.2.3 Radio Transmission Implementation

The second step consists in the introduction of a base station directly connected to the gateway through a serial cable and implementing a radio communication between the camera mote and the mentioned base station.

The transmission protocol implemented was based on the acknowledgement of each transmitted packet. Initially it does not limit the retransmissions number in case of lost packets or acks. Subsequently this paradigm has been conserved only for images delivering while video frames packets can not be retransmitted more than 5 times thus if the fifth sent fails the data burst is lost.

The communication between the control application (CameraGUI) and the camera mote is bidirectional. The traffic going from the node to the base station and then to the Java program is mainly composed by data packets. On the other side, from the gateway to the motes they are sent control packets such as commands to start and stop operations and to control the information flow. For instance, the command messages that reset the frames count are sent directly by the Java application. Moreover there are all the acknowledgements responses for each packet correctly transmitted between camera mote and the base station.

The base station task consists only in forwarding packets to the gateway. Every message received is saved in a FIFO queue. A task has in charge to consume the queue content by sending messages on the serial channel to the Java application that will elaborate them. Actually this queue contains for most of the time only the packet just received. In fact, the serial communication is faster than the radio one because packets there are no losses and the packets are not acked, thus messages are forwarded immediately avoiding queue waits.

In a second time the system was extended to support a multihop paradigm. This required the design of intermediate nodes with the simple assignment to forward every kind of packets passing through them. From this moment the simple communication between nodes seen till now assumes more a network connotation. It raises therefore a question on which kind of routing paradigm is better to use. To simplify the network design and to avoid augmenting the network traffic with routing messages, it was chosen a static protocol. Every intermediate node has a source and a destination corresponding to the TOS_NODE_ID (the node identifier,unique within a network) of the previous and the subsequent node in the path from camera node to base station node. Obviously the camera node has only a destination address stored while the base station has only a source address.

The checks that intermediate nodes and base station node apply on the incoming packets are the same. They control that the information flow respects the original order by checking the identification numbers inserted within the packets payload. The behavior is slightly different for photo and video packets. The photo packets, in fact, are acked through all the hops they made till the base station. For these reason all the packets must arrive in the correct order to guarantee the final picture could be reconstructed and each node checks that the ID of the current message is the subsequent to the previous one. On the other side, video packets are subjected to a maximum number of retransmissions attempts. Due to this fact a packet could be lost in any step of the path and the sequence could result broken. Thus the nodes check only that the current packet has an ID major than the last one received. This mechanism was initially studied to avoid duplicated packets to reduce the number of forwarded packets and, in the development phase, helped in finding packets sequence errors. In a scenario with many intermediate nodes it could be useful exploiting these checks to anticipate the broken sequence detection and the sending of the resynchronization command to the camera mote in such nodes to reduce notably the unuseful traffic.

The nodes implement a queue to store data packets of both video and photo type. Others kinds of messages, such as command or control messages, are not put in a queue on

their arrive for two reasons: first because implementing more than one queue give rise to inconsistencies within the queue system itself, as will be described in the following *design issue* section, secondly because there are no cases in which it is necessary to send more than one command in a short time and the loss of these messages in the reception phase compromises the system behavior.
The main difference between the base station and the intermediate node is that in the first case received messages are sent on the fast and reliable serial channel while in the second case the sent is again on the radio medium that is subjected to errors and retransmissions. Due to this fact, the queue of the data packets on the forwarder nodes can often saturate. To avoid this scenario, in which further incoming packets would be lost, a mechanism to regulate the incoming traffic has been introduced. There exist two thresholds called respectively *stop threshold* and *restart threshold* that allow the system to know if the queue is near saturation and to take the proper countermeasures. When the incoming messages queue size is equal or major than the stop threshold, the mote continues accepting new packets but sends a control message to its source node. On the reception of this message the source node stops to send packets and remains in that state until it does not receive another control message from its destination node meaning that it can restart sending operations. The latter message is sent by the full queue node only when its queue size decreases under the restart threshold value. Obviously when a node can not send messages it easily will fill its own queue and will ask its source node to stop sending messages. This stop condition will be propagated until the the stop of the camera node that is the actual source of data packets.

Design Issues

The main problems in the radio communication development concerned hardware and software limits more than real design problems. The first issue concerned the TinyOS acknowledgements. After several tests it was found that the higher frequency to which motes can work, 208 MHz, gives rise to problems with the delivery and the

reception of the acknowledgements. It was necessary to slow down the working frequency of each node to 104 MHz to obtain from the system the expected behavior.
The second problem encountered interests the already mentioned queues. Allocating more than one queue to store not only the data messages but also control and test messages, the queue system become totally unreliable. Packets correctly received and saved in the queue showed inconsistencies and wrong values once extracted and read. To solve this problem it was necessary to deallocate all the queues except for the data one, the only necessary for the system.
The last difficulty found concerned the messages reception. The symptoms of the problem rised when to the well tested system composed by a camera node, an intermediate node and the base station they were added an arbitrary number of intermediate nodes. The system seemed to work well for the first moments, all the packets were received correctly by the gateway application passing through all the sequence checks of all the intermediate nodes and of the base station and through the hardware checks like Cyclic Redundancy Check (CRC) or concerning the message type or the header flags. But once the video frame or the entire image were displayed they present notable degradation till being totally corrupted. After a meticulous analysis of the entire system it was clear that the problem did not belong to the implemented software. Moreover the 2 hops system worked well and on all the intermediate nodes the same code was installed. Firstly we though about the increased network traffic but the symptoms of the problem did not go in that way. Packets were not lost, they were corrupted but at the same time they passed all the controls. The header was intact but the payload was not. After diverse tests it was clear the problem was located between the hardware checks and the triggering of the receive event by the TinyOS software of the intermediate nodes. Basically each intermediate mote could not face the elaboration of the packets after the rate at which the packets arrived was augmented due to the addition of more intermediate nodes. The nodes were stressed by the increasing number of checks required to discriminate whether accept a message or not and could not elaborate correctly the accepted packets. To overcome this situation it was reduced

consistently the payload size of each packet from 100 and 64 bytes, respectively for video and photo, to 40 bytes for both the data types. In this way the number of packets sent for the same data size augmented but each packet is much more easily handled by the reception stack of the intermediate nodes. Furthermore, as a confirmation of the fairness of the solution adopted, it is possible to see how the corruption of the packets increases augmenting the payload size till a limit, around 50 bytes, when many packets are received corrupted. Another observation that could be made is that adding more and more intermediate nodes the problem could represent. Already with three intermediate nodes sometimes it is possible to notice images degradations even if slight. In Figure 6.8 the video streaming system functioning is showed.

6.3 Symulation Of A Different Network Protocol

The implemented communication protocol is very simple. Every packet sent, at each step of the path between the camera node and the base station, is aknowledged before sending the subsequent. Even with the introduction of the 5 retransmissions limit for the video packets, this kind of paradigm generates an high packets traffic and a notable delay between one transmission and the subsequent.

To speed up the transmission operations it has been proposed an alternative protocol, called *Selective Repeat*. The basic idea of this protocol consists in sending a certain number of packets without requiring acknowledgements in any of the path steps. Once the base station received the last packet, or once a control timer is fired, the received packets are checked and for those not received is sent a message to the camera node with a retransmission request. The retransmissions follow the same paradigm: they are sent without checking the correct reception. If even one of the retransmitted packets is lost, the base station node sends another retransmission request. Obviously this protocol is based on the assumption that the channel has a low error rate so that the retransmissions number is little.

The *Selective Repeat* version of the video system sends, without waiting for acknowl-

edgements, a burst of 64 packets and then stops waiting for a feedback from the network. All the sequence of packets sent is stored in the camera node's memory. The base station waits for the incoming packets. Once the last message of a sequence is received, the base station checks how many packets of such sequence were correctly received. If all the packets were received, the base station sends an acknowledgement to the camera node, otherwise it sends a retransmission request indicating which packets were lost.

A specific control message is used to communicate to the camera which packets must be retransmitted. This message contains a 64 bit integer as payload. The integer represent a bitmap, initially all the bits are zero. For every correctly received packet, the base station sets the correspondent bit in the bitmap to one. Once the camera receives the control message scans the bitmap and retransmits the packets corresponding to the positions of the bits set to zero.

It must be noticed that the base station needs more information to correctly send retransmission requests. First of all, the base station must know that all the packets of a sequence has been sent even if the last message is lost. Secondly, the last burst of packets of the entire data source could be composed of less than 64 packets. The first problem is solved setting a timer which, once fired, sends the retransmission message containing the 64 bit mask built till that moment. The timer value must be proportional to the average transmission time of a burst. To correctly set the timer value, we collected a series of burst transmission times. The timer expiration is set to the maximum of the collected values further augmented by the 5%of its value. To solve the second problem it has been necessary to send the size of the source data within an acknowledged packet.This description packet brings the information on the image size that allow to compute the number of the last sequence and the last packet ID.

To study the behavior of this new paradigm, a simulated version for Tossim has been implemented . To compare the behavior of the current system with the new one also the old system has been simulated. In the next Chapter the results achieved will be examined.

Chapter 7

Performances Evaluation

AFTER the implementation phase, described in Chapter 6, a detailed analysis of the system performances is now presented. The test performed focused on observing the system functioning in both the operating modes: pictures acquisition and video streaming.

Several parameters are taken into account to evaluate the two subsystems. These measurements are taken varying environment variables such as nodes' transmission powers and the network population. While video quality is fixed the photo subsystem is tested also varying the pictures format. All the photo part tests are done using compressed pictures, distinguishing between QVGA and VGA format.

In order to measure the different parameters without conditioning the system functioning, data collection operations required a series of code modifications, especially in monitoring the network traffic. In this Chapter we will describe the methods used to collect and to elaborate data and how the system responds to the different transmission power and topology conditions.

7.1 Testbed

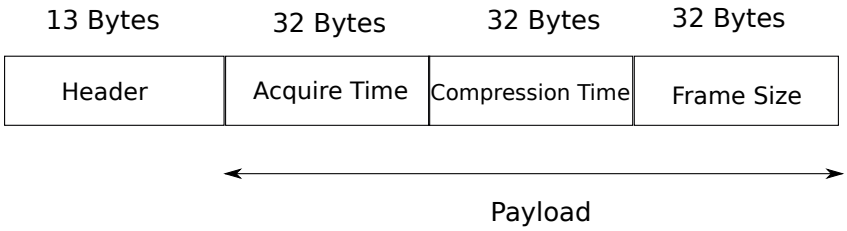
The preparation of the test environment required to insert several data collectors all over the developed software.

The monitoring of images processing operations is split between source and destination of the data flow. Acquire time (τ_A) and compression time (τ_C) are collected through timers inserted in the camera module, cameraJpegTestM.nc. While the former measures the temporal interval necessary to take the picture and to store it in a buffer allocated in the *SDRAM*, the latter is the time taken by the the frame to pass through all the compression steps of the Jpeg algorithm, in the photo case, or of the Dpcm algorithm, in the video case. On the destination side, it is measured how long the C decoder takes to decompress the received data (τ_D). Also here the photo and the video cases must be distinguished. While for the first function all the received data are Jpeg compressed pictures, only some of the video frames constitute an entire image, the others are residuals that must be decompressed and added to the buffer content. Therefore in decompression time of the video case are included also all these operations. Moreover, for all the frames, independently from their type (entire images or residuals), the compressed data size is recorded .

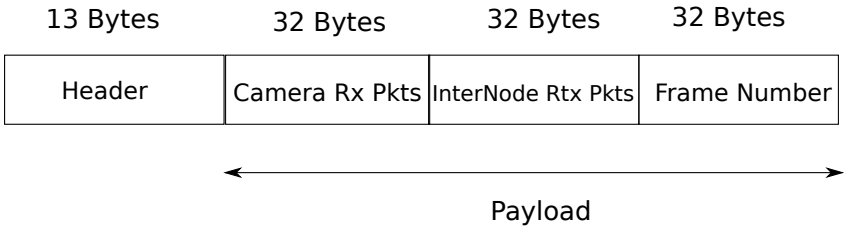
For each frame these information are collected and sent to the gateway in a time message packet, showed in Figure 7.1. In the video streaming monitoring, the correct sending of these packets is not checked with acknowledgements, to reduce the influence on the network traffic to the minimum.

Another series of collectors are used to monitor the network traffic. The camera mote and each of the intermediate nodes register the number of sent packets (N_{send}) and the number of retransmissions (N_{rtx}) required to correctly deliver a message.

This statistic message is sent by the camera mote once the transmission is finished: in the pictures case, the message is sent after the last data data packet has been acked while in video streaming mode the test packet is sent once the video transmission has been stopped. In both cases the packet collects the values of N_{send} and N_{rtx} located



(a) Time Test Packet



(b) Statistics Test Packet

Figure 7.1: Schemas of the Test Packets

at each step of its path from the camera mote to the gateway. To be sure to collect updated data, at each step the packet is elaborated and forwarded only once the data message queue is empty. The statistic message format is showed in figure 7.1.

All these measurements are collected, with the respective differences, for both photo and video modes. To better understand the behavior of the video streaming, it is necessary to add some other measurements. These values characterize the performances of the DPCM algorithm over the entire video system. The time in which the buffer is filled with new frames (τ_{refill}) and how long does it takes to such buffer to be empty again (τ_{empty}). The latter timers are placed directly on gateway application.

All the test are conducted with a linear network topology where the nodes are in line of sight with each other. Each test is executed for a number of hops that ranges from 1 to 4. Each of these is performed for three different power levels. The higher transmission power adopted is of $1mW$ (0 dBm). The lowest is of $3\mu W$ (-25 dBm) while the intermediate level is of $0.1mW$ (-10 dBm). For each of these test cases the photo

system is operated varying the image format. On the other side, the video streaming performance is distinguished on the basis of the movement of the subject in high and low motion. Indeed in videos with high degree of movement the subsequent frames show a lower correlation, increasing the size of the residual produced by the DPCM algorithm and consequently the number of packets transmitted.

7.2 Image System Analysis

The image processing performance analysis was done considering a sample of ten pictures for each test case. Moreover, to test the network performances, due to the fact that the brightness of the environment and even the slighter movement of the camera can influence the compressed data size, a single QVGA picture is considered, storing it in memory and sending it multiple times. We chose an image rich of details to analyze the case in which the compression algorithm is much more stressed and the resulting compressed data size is considerable. An example of the high details image considered during the tests is showed in Figure 7.2.

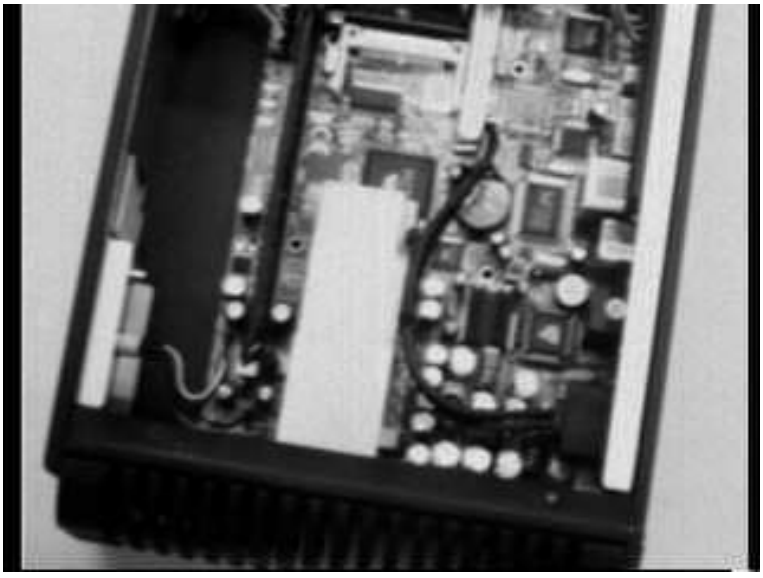


Figure 7.2: Picture Sample Used for Network Performances Monitoring

In the table 7.1 the average τ_A, τ_C and τ_D values are presented considering all the test cases, over a population of 120 samples. Time values are reported in milliseconds while size is in bytes. Table 7.1 shows how the acquire time of the image is substan-

	Acquire Process [ms]	**Compression Process [ms]**	**Decompression Process [ms]**	**Image Size [byte]**
QVGA	92.6583	470.0750	87.0750	8.6095e+03
VGA	90.3667	1.3351e+03	210.0750	2.3221e+04

Table 7.1: Pictures Processing Times

tially constant for the two formats while the data size is strongly influenced by the picture resolution. Due to the increased size also the τ_C and τ_D on the VGA file is more than two times those of the QVGA data. The table also justifies the adoption of techniques, described in Paragraph 6.2, to further reduce the transmitted data in video streaming system. Indeed every frame would require more than 500 millisec-

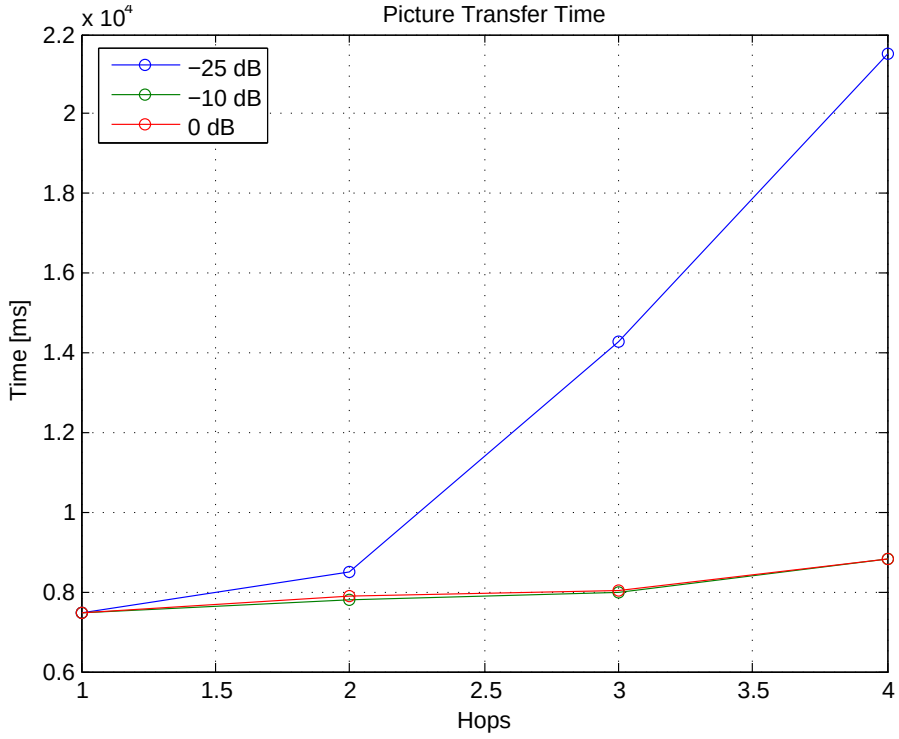


Figure 7.3: Graph of the Transfer Times of a QVGA Picture

onds only to prepare the image to send. Moreover it must be considered the sending time to the closest node of the network that, as we will see in the following graph, is around 700 milliseconds. This would mean that the minimum reproduction frame rate at the gateway is less than one frame per second. Figure 7.3 shows the photo system performance in sending a picture of a fixed size varying the number of hops and the transmission powers. As explained before in this Chapter, the picture sent is always the same so that the measurements are independent from data size. For each test 240 packets are sent. The timer is started when the gateway sends the *Get photo* command and is stopped once all the packets have been received. From the Figure it is clear that the network behavior remains constant for 0 *dBm* and -10 *dBm* transmission powers, with the transfer time that slightly increases augmenting the number of hops but does not exceeds the second.

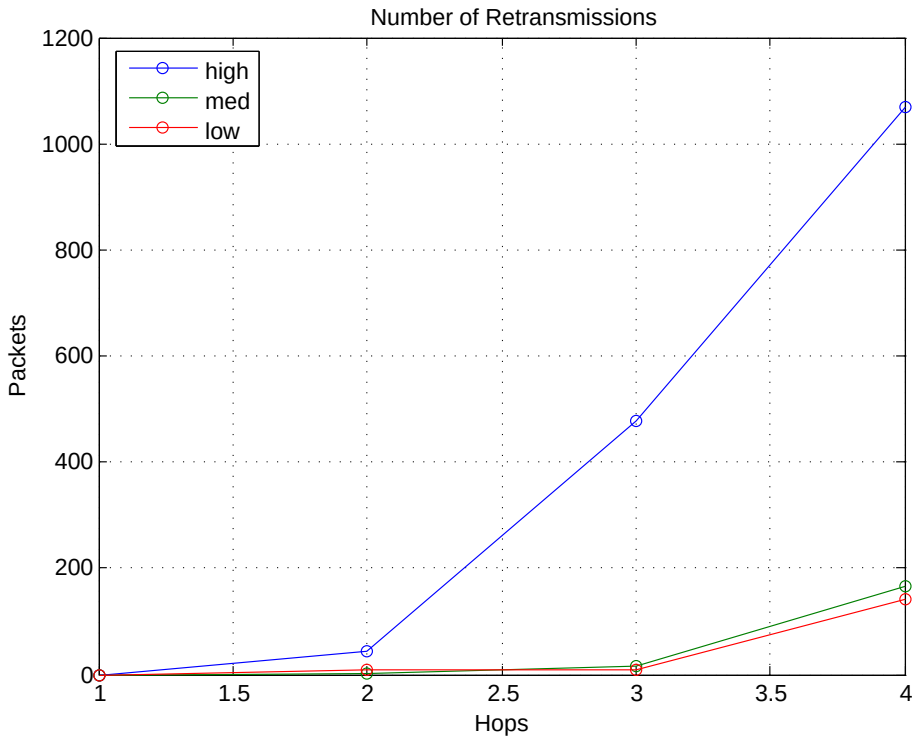


Figure 7.4: Graph of Packets Retransmissions of a QVGA Picture

Things get worst with the lowest power for which the transfer time is equal or compa-

rable to the other two powers in the first part of the graph but subsequently diverges for three and four hops, reaching values higher than 20 seconds. Another interesting parameter to measure is the retransmissions number N_{rtx}. We expect that graph 7.3 and 7.4 are strongly correlated because the transfer delays depends on the number of lost packets. Due to the fact that no losts of packets are allowed, all the messages are retransmitted until they are correctly delivered and the time losses caused by these transmission attempts are reflected in the delays graph. We can notice how the two behaviors are similar. With the two higher powers the functioning is almost the same for all the network configurations while with the power transmission level set to -25 *dBm* the retransmissions number augments considerably in the three and four hops cases.
In general, the photo system behaves very good with a number of intermediate nodes minor than three and starts losing efficiency since the add of the third forwarder node, provided that we are in the higher or intermediate power level. Otherwise the performance degradation is visible since the add of the first intermediate node and shows a notable worsening adding the second and the third. This correlation between the number of nodes and the increase of lost packets can be explained by the mutual interference generated by the motes.

7.3 Video System Analysis

To measure the performance of the video system for all the test cases, we considered video streams of 5 minutes. For each test case two types of videos were recorded: the first one was produced filming an almost static scene, with slow movements of the subject and a fix background while the second type concerned high motion scenes on a fix background.
Obviously it was not possible to reproduce the same motion quantity for all the test cases, and it was not possible to avoid brightness difference between different tests. Due to this fact the graphs could appear slightly less consistent with what we expect.
Another note concerns the type of parameters considered in the tests. Due to the fact

that the first video frame is always an entire jpeg image, it was meaningless to measure the delay from the starting of the video acquisition to the reception of the first frame, because the times are exactly the same of the photo transfer time except for the limit on the number of retransmissions, as explained in Paragraph 6.2.3. Moreover the received picture is not immediately displayed but is put in a buffer. For this matter the filling and emptying times of such buffer are considered taking into account the worst case corresponding to the maximum filling time $\tau_{maxEmpty}$ and the minimum emptying time $\tau_{minRefill}$.

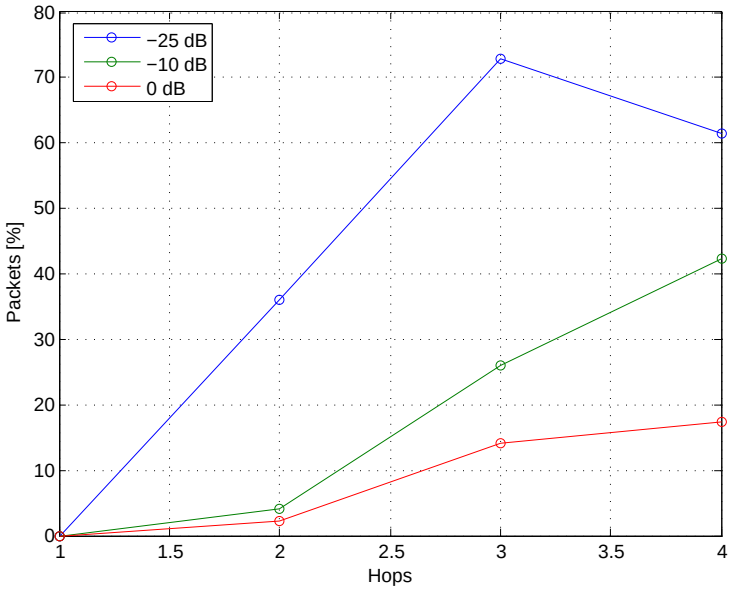
As done for the photo system, firstly we present a table with some significant values describing the image elaboration process. These values are τ_A, τ_C, τ_D and the frame size. As expected the acquire time does not differ much either from the photo case nor between the motion types. The differences are instead visible for the compression and decompression values. Here it takes less time to compress and decompress a frame because the size of the data are notably decreased with respect to the photo case. It must be noticed that here we called frame both the entire Jpeg images and the residuals produced by the DPCM algorithm. The mean size of the frames is almost 8 times

	Acquire Process [ms]	**Compression Process [ms]**	**Decompression Process [ms]**	**Frame Size [Byte]**
Low Motion	91.7597	268.3625	29.1431	1.1128e+03
High Motion	92.6904	269.4769	28.5767	1.1701e+03

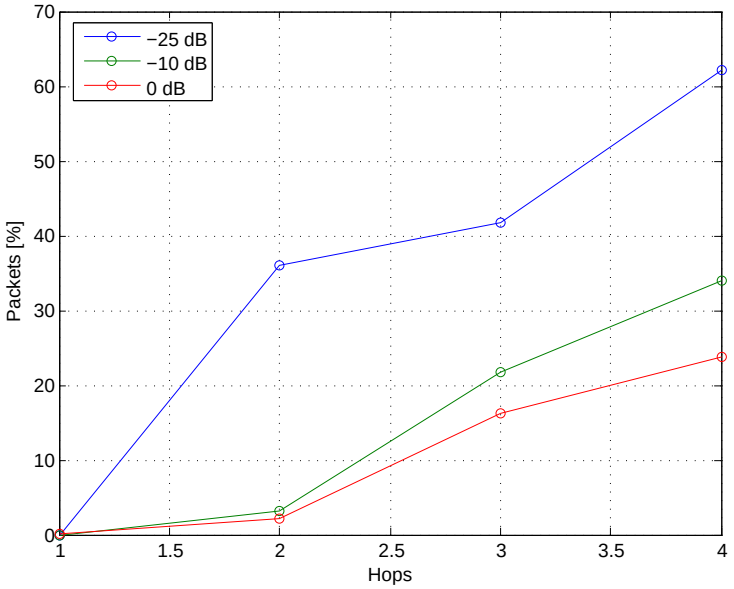
Table 7.2: Video Processing Times

smaller than the QVGA case of the table 7.1. This is one of the major results reached in this work. Indeed also the video format is QVGA but thanks to DPCM algorithm and image subsampling technique, described in Paragraph 6.2.2 the mean data size transmitted in the video mode is strongly smaller with respect to the photo system.

Another measure similar to the photo case is N_{rtx}, showed in Figure 7.5. In the following we present the percentage retransmitted packets with respect to the three different power levels and the different network configurations for both the high and the low



(a) High Motion Retransmissions



(b) Low Motion Retransmissions

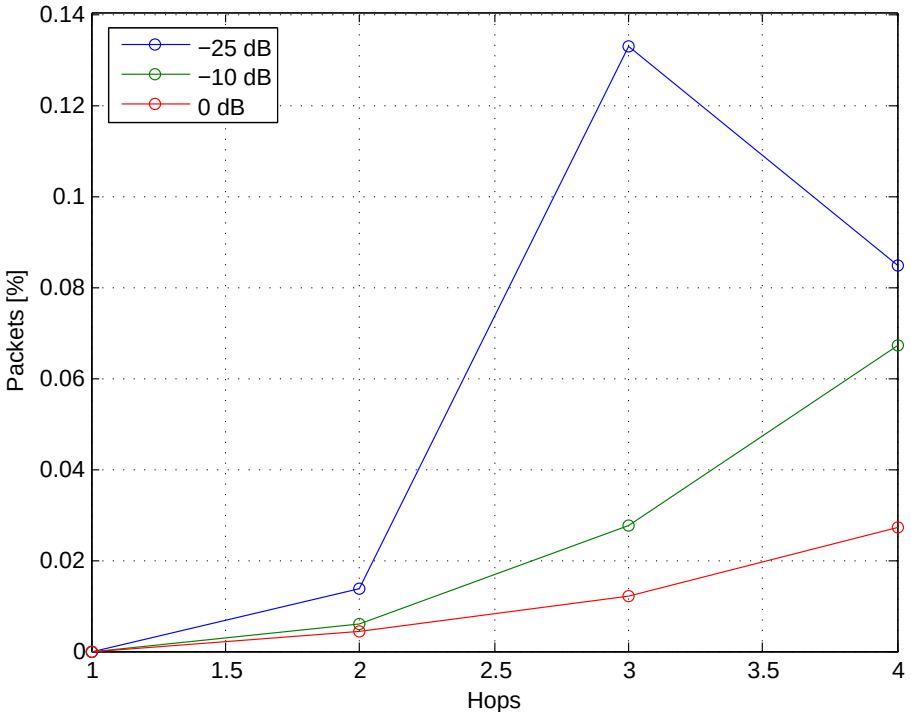
Figure 7.5: Percentage of Retransmitted Packets in the Video Streaming System

motion cases.

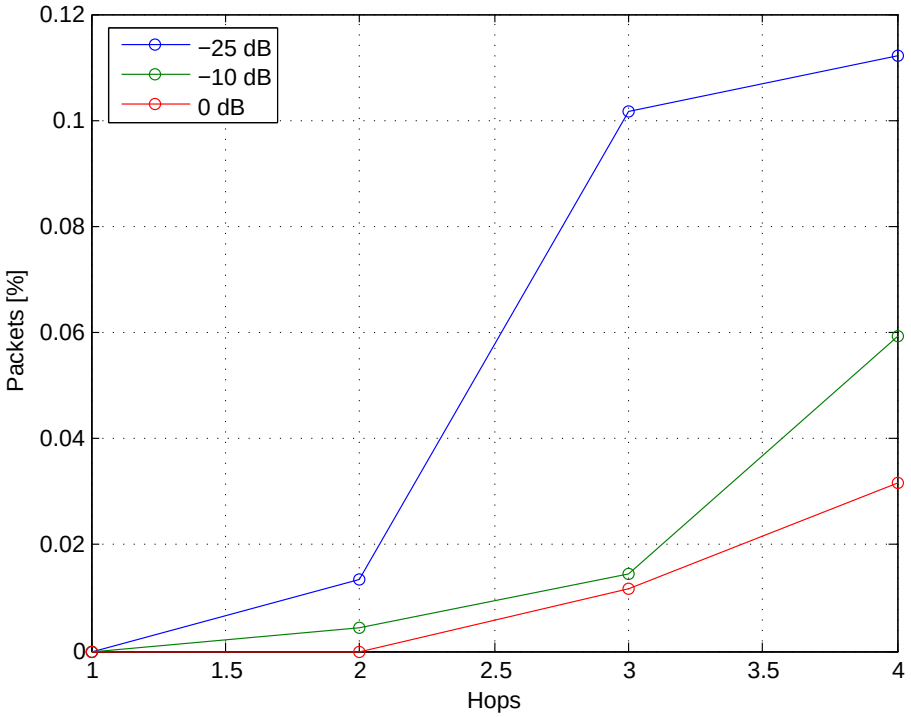
The two graphs are very similar. For all the three power levels there are no retransmissions in the one hop configuration. The worst behavior is registered when the power level is set to -25 *dBm*. The low motion case behaves slightly better due to the fact that the generated traffic is less because the mean data size is smaller but the differences are not marked. The two higher power levels perform very good also in presence of an intermediate node. The retransmissions number for high and low motion cases with transmission power level -10 *dBm* settle around the 5% of the transmitted packets while with a 0 *dBm* power that value decreases to the 2% for both cases. In the other two network configurations even the intermediate power loses some efficiency with the percentage of retransmitted packets that is more near the 30%. On the other side the max power level case performs better in all the situations, remaining almost always below the 20%. Only with the low motion and four hops configuration the amount of retransmissions is over the 20%, as for the intermediate power case. Probably this bad behavior is also influenced by the external and environmental factors such as brightness changes or degradations in the radio channel quality.

As described in the Paragraph 6.2.3, it is possible, due to the video transmission protocol implementation, to lose packets. Indeed every node try to retransmit the unsuccessfully delivered packet for a maximum of 5 attempts after which the packets is considered lost. The following graphs, presented in figures 7.6(a) and 7.6(b) show the behavior of the video system respectively in high and low motion cases, in terms of packets losses. Notice that this measurements are strongly subjected to the channel quality variations between two different test cases.

Both the graphs show the expected behavior with the packets lost number that increases adding new intermediate nodes and augmenting the hops in the path. Again the worst performance is registered by the lower power level that loses more than the 10% of the transmitted packets in the configuration with four hops, in low motion mode. The higher power levels show a good behavior also with the three hops configuration while in the lowest power level case more than the 12% of the packets are lost. A consistent



(a) High Motion Packets Losses



(b) Low Motion Packets Losses

Figure 7.6: Percentage of Packets Lost in the Video Streaming System

efficiency reduction also for the 0 *dBm* and the -10 *dBm* cases is visible only with three intermediate nodes. The two graphs does not present a great spread between the registered values due to the motion type. The most important characterization of the

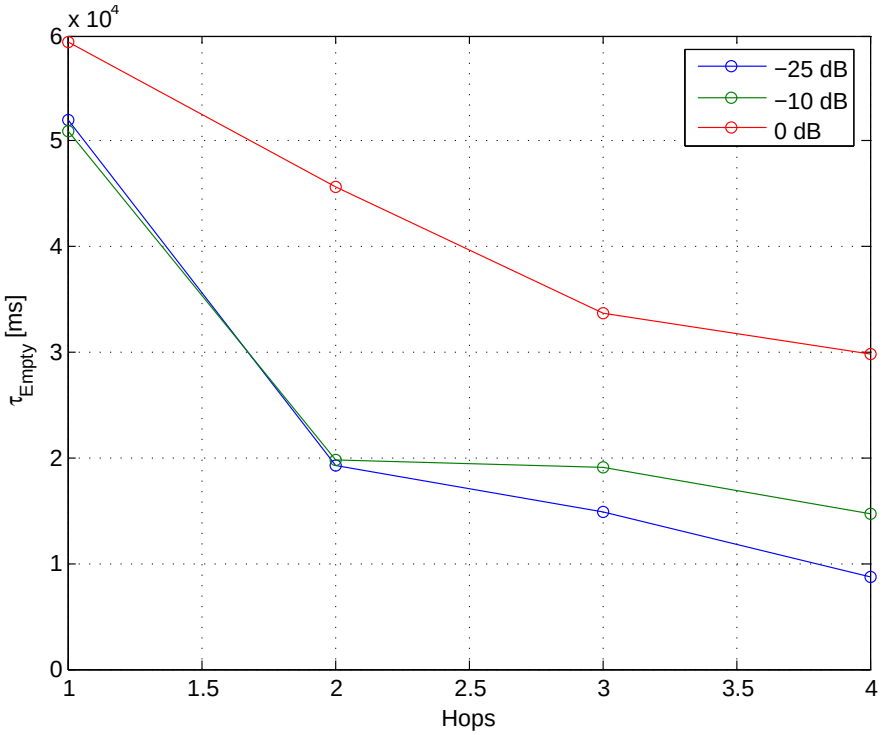


Figure 7.7: Graph of Buffer Emptying Times in High Motion Mode

video streaming behavior considers the time delays. As described in Chapter 6, the current implementation uses a buffer to store the received frames and starts displaying the buffer content in a second moment, when the number of elements in the buffer reached a certain threshold. Such threshold is now set to 50 frames. In the same manner, when the buffer content is consumed, the reproduction stops to allow the system to refill the buffer. For this matter, it is difficult to consider parameters that show the reproduction delay directly because it depends on the position of the received frame in the buffer. It has been decided to describe the system functioning through the τ_{empty} and τ_{refill} times of the buffer. In this way we can not know the exact display delays but we can indirectly estimate the system behavior considering that τ_{empty} is the period in

which the video is displayed while τ_{refill} is the period in which the video stops waiting for the new frames.

In Figure 7.7, we showed the minimum buffer emptying times for the different network configurations, in high motion mode. In this case the high power level shows a considerable better behavior with respect to the other two levels. In the worst case, represented by the minimum emptying buffer time with the four hops configuration, the video is on for 30 seconds. It is a good result if compared to the worst case value, 40 seconds, of the refilling time, showed in Figure 7.9(a). It means that with a four hops architecture, provided that the power level is set to 0 dBm, in the worst case for every 70 seconds of the video streaming activity we can monitor the subject for at least 30 seconds. On the other side, the graph shows also that for the other two power levels things get rapidly worse. Indeed apart the good behavior registered for the one hop configuration, the other configurations show emptying times minor than 20 seconds.

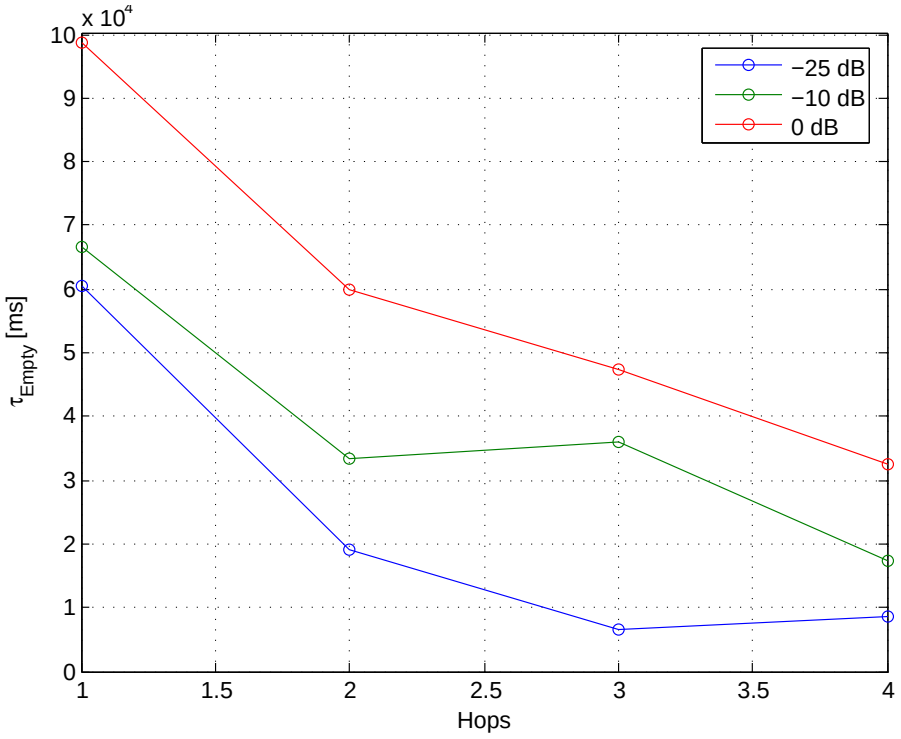
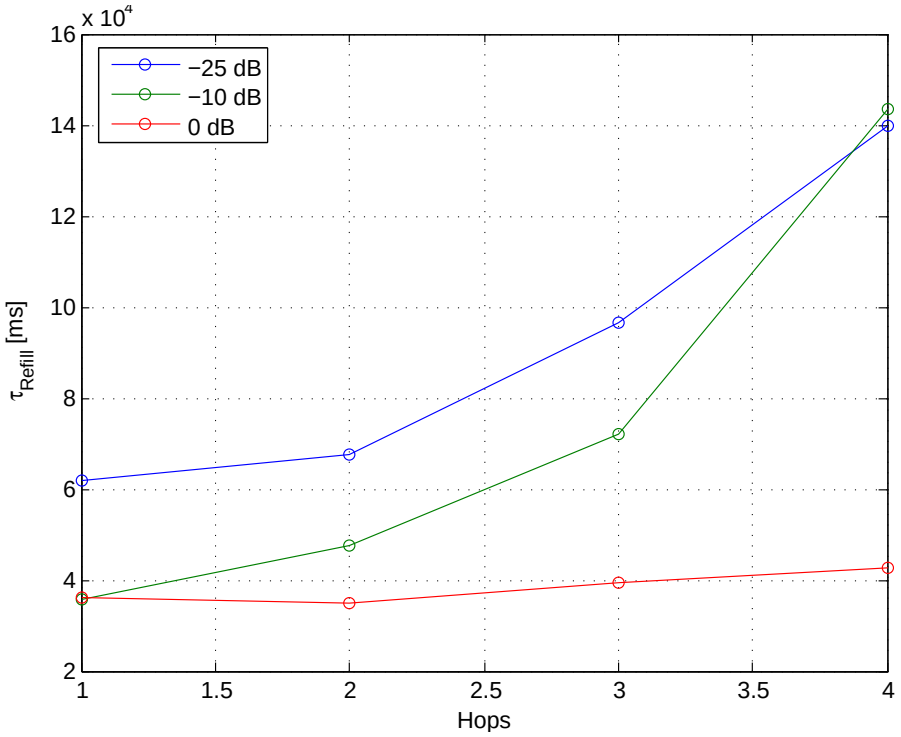


Figure 7.8: Graph of Buffer Emptying Times in Low Motion Mode

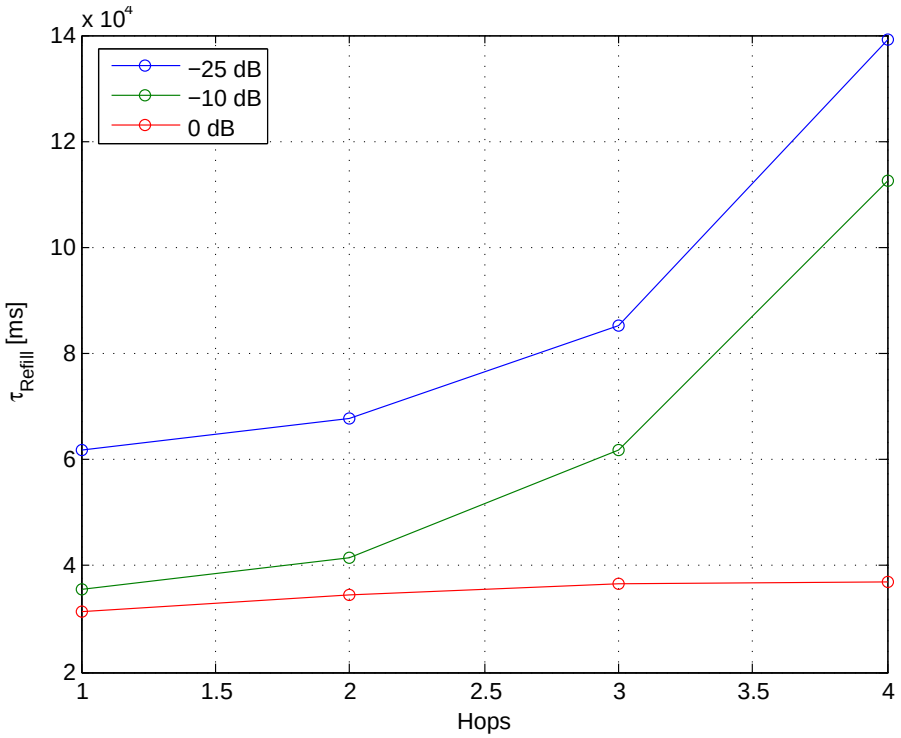
It is important to notice that this time the motion type influences substantially the system behavior. In fact, especially in configurations with less hops the buffer emptying time is longer for the low motion mode, where the mean packet size is smaller. This difference is less marked with a major number of intermediate nodes due to the presence of more retransmissions that made the emptying time value more and more dependent on the environment and channel interferences. The global progress of the graph in Figure 7.8 is substantially similar to the high motion case but with a more marked difference between the two lower levels. Moreover, considering also the refilling time graph of Figure 7.9(b) we can notice that with the maximum power, in the worst case of the four hops configuration, we can watch almost 30 seconds of video every minute.

Analyzing the refilling time graphs, in figures 7.9(a) and 7.9(b) we can have a better view of the periods of time in which the video is off, waiting for the buffer to be filled. It is possible to see how the refilling time is almost constant through the change of the network configurations for the higher power level. We can infer that, once the buffer is empty, it will take almost forty seconds before the video restarts. Even here,especially for the high power level, the low motion video shows slightly smaller values, especially in the configurations with less intermediate nodes. The two lower power levels show a completely different behavior with respect to the 0 dBm level, with the refilling time that grows with the addition of intermediate nodes. This is probably due to the increasing number of retransmitted packets that slows down the entire refilling operation.

The last parameter taken in account is referred to the resynchronization time of the DPCM algorithm. As described in Paragraph 6.2.3, when the Java application of the gateway detects a lost packet, it sends a *sequence broken* message to the camera mote to restart the DPCM sequence sending an entire Jpeg image. This resynchronization operation costs time and the graphs in Figure 7.10 describe this time loss. The timer is started when the application receive an out of sequence packet and is stopped when the first packet of the Jpeg frame is received. Here the worst case is again considered,



(a) High Motion Refilling Times



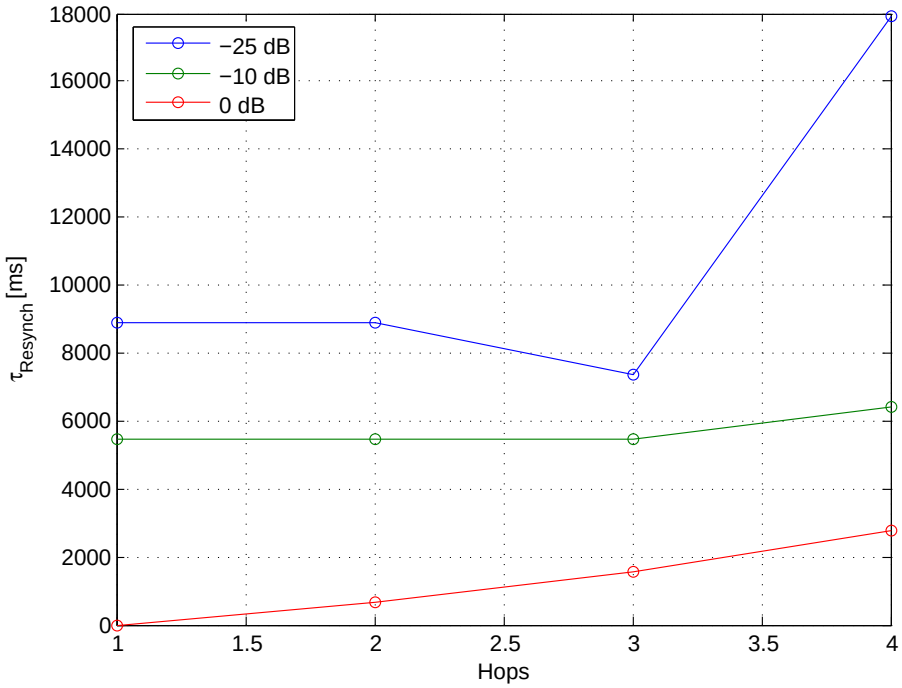
(b) Low Motion Refilling Times

Figure 7.9: Buffer Refilling Times in the Video Streaming System

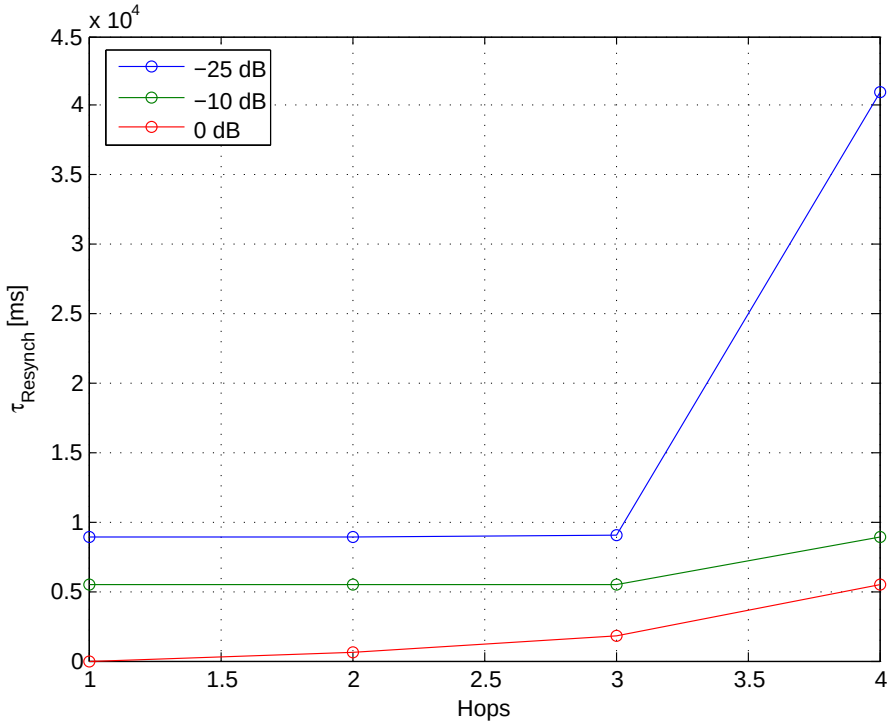
reporting the higher time values recorded.

For the intermediate power level the time values are different between the power levels but almost constant for all the network configurations, except for the 4 hops configuration where the time value slightly increases. For the lower power level the time values are constantly under 10 seconds for the first three network configurations. In the fourth network case the resynchronization time diverges till reaching 18 seconds in the high motion case and overcoming 40 seconds in the low motion case. Even here the higher power level performs better remaining under five seconds even in the worst case, with three intermediate nodes.

Where the resynchronization time is zero, it means that the streaming never broke.



(a) High Motion Case



(b) Low Motion Case

Figure 7.10: Resynchronization Time in the Video Streaming System

7.4 Network Protocols Analysis

To improve the efficiency of the system, as described in Paragraph 6.3, a new network protocol is proposed. Due to the fact that the major limitation of the current system comes from the high number of transmitted packets, the proposed algorithm try to reduce the network traffic limiting the number of acknowledgements sent by each node.

The proposed paradigm, called *Selective Repeat* is based on the transmission of a 64 packets burst, from the camera node to the base station, without requiring any acknowledgement. Only once all the 64 packets has been sent, the sender waits for a cumulative acknowledgement. If some of the transmitted packets are lost along the path, the base station asks for their retransmission.

To evaluate the possibility of implementing a *go back n* version of the video streaming system the protocol has been simulated using Tossim, the sensor network simulator described in Section 5.6. Moreover, to compare the *go back n* with the paradigm used by the system, also the *Stop and Wait* protocol currently implemented is simulated.

	Sending Time [ms]	**Retransmissions Number**	**Packets Lost**
Stop and Wait	1.0467e+004	0	0
Selective Repeat	9.9537e+003	0	0

Table 7.3: Low Noise Simulation Results

The simulation software reproduces a sensor network composed by the camera node and a base station. The camera node generates a fixed size matrix representing the image produced by the camera node. Indeed, the matrix has 320 columns and 240 rows, as a QVGA image. Once the simulated image is generated, the data are divided in 1200 packets with a 64 bytes payload. Moreover, the 1200 packets are subdivided in 18 sequences of 64 packets each. Finally, two simulations of 5000 seconds are performed for each protocol, changing the noise level in the network.

In the following the results of the simulations are presented for the high noise and

	Sending Time [ms]	Retransmissions Number	Packets Lost
Stop and Wait	1.0668e+004	17.7107	0
Selective Repeat	1.0478e+004	17.5656	0.0164

Table 7.4: High Noise Simulation Results

the low noise cases. In tables 7.3 and 7.4 the average sending time (τ_s), the average retransmissions number (N_{Rtx}) and the average number of packets lost (N_{Lost}) are reported for both the protocols implementations. The mean is computed on a per photo basis.

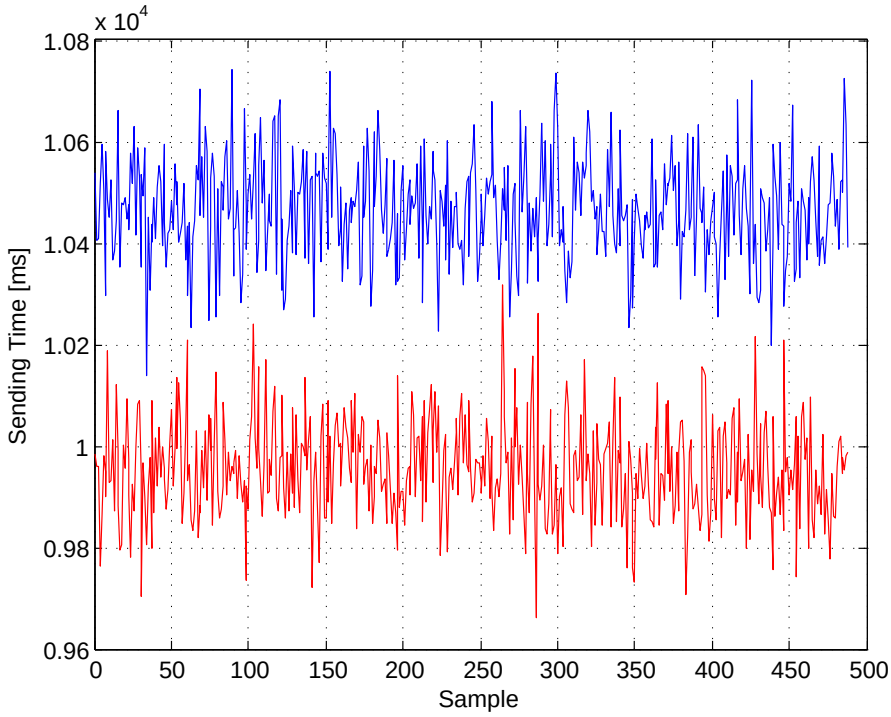


Figure 7.11: Graph of The Simulated Sending Times of the Low Noise Simulation

The table 7.3 shows how for the low noise simulation, where the system does not retransmit any packet, the *Selective Repeat* protocol performs notably better than the *Stop and Wait*. The mean sending time, measured over a population of 488 simulated photos, is almost 500 milliseconds lower in the *Selective Repeat* case. The sending

times progress with respect to each sample of the population is showed in Figure 7.11. The red graph represent the performances of the *Selective Repeat* protocol, while with the blue color is traced the *Stop and Wait* behavior.

The differences are less marked if the channel noise model is worsened. Indeed, as showed in the table 7.4 with the same N_{Rtx}, the behavior of the two paradigms, in terms of average sending times τ_s, is now comparable even if also in this case the *Selective Repeat* protocol performs slightly better. Moreover, comparing the table 7.3 with the 7.4, we can notice how the both the protocol lose performances with the increase of the channel noise but, above all, how the worsening of the *Selective Repeat* paradigm is much more marked.

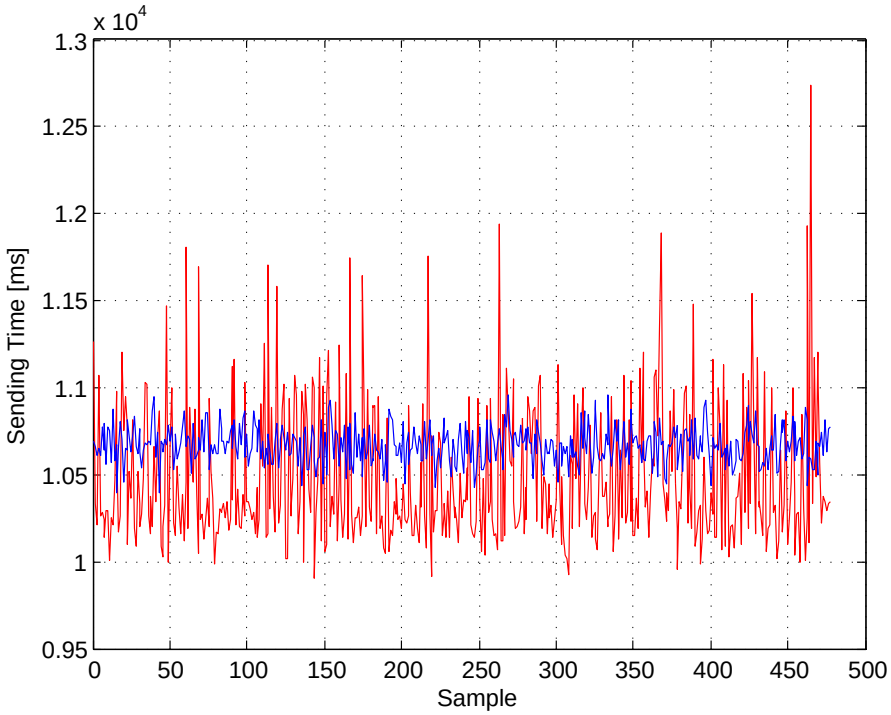


Figure 7.12: Graph of The Simulated Sending Times of the High Noise Simulation

Another interesting characteristic emerging from the protocols simulations is the most major variance of the τ_s values showed by the *Selective Repeat*, especially in the high noise case. This suggests, as we expected, that the *Selective Repeat* performances

suffer more the uncorrelated channel quality degradations. Indeed while a moment in which the channel is particularly bad can cause a small amount of retransmissions in the *Stop and Wait* paradigm, in the *Selective Repeat* protocol this situation force the base station to send back the retransmission request which must be elaborated by the source node and performed, requiring a not negligible time overhead. Moreover the uncorrelation of the errors lean to distribute the errors on different bursts so that the described procedures must be repeated multiple times during a single photo send.

As in the low noise case, the two protocols behaviors are showed graphically, in Figure 7.12. The high noise data are collected over a population of 477 simulated photos. Again, the *Stop and Wait* performance is signed in blue while the *Selective Repeat* is represented with the red color.

Chapter 8

Conclusions

THE DISSERTATION has focused first on a survey of Wireless Sensor Networks with the analysis of features and limitations of this kind of networks. Subsequently we introduced the Wireless Multimedia Sensor Networks showing how the capabilities of the system and the design issues change in this new approach. Afterward, the research results on pictures and video coding techniques are reported, presenting the main mathematical theory foundations, such as DCT and Wavelet transforms, and reporting principal lossy and lossless image coding methods as well as video coding algorithms.

In the second part of the work, the developing phases of the video streaming system are described in details, underlining the encountered problems and the solutions adopted to overcome such difficulties.

Finally the photo and the video subsystems are tested measuring how they perform through the collection of most significant parameters. Some values, such as those describing image processing performances (τ_A, τ_C and τ_D) or the retransmissions percentage, are meaningful for both the subsystems while others are needed to better describe the more complex video system. The latter parameters include buffer timings, τ_{Empty} and τ_{Refill}, as well as packets lost percentages.

From the performed tests emerges that the most important results come from the used

compression techniques. Indeed, the DPCM algorithm implemented produces frames almost 8 times smaller than Jpeg compressed images of the same format (QVGA). This is obtained at the price of a slight but tolerable image quality degradation in the reproduced video. Moreover the average DPCM compression and decompression times are notable smaller than those of the Jpeg QVGA images.

In the video case, on the network performances side, the tests underline a very good system behavior for the one and two hops configurations provided the use of the maximum or the medium power transmission levels. With the three hops configuration is also achieved a fair performance level with both the higher power levels. Moreover, the maximum transmission power level supports pretty well also the add of the third forwarder node. It must be said that the behavior of the lower transmission power level is very different with respect to the other two. In this case, the system performs reasonably well only in the one hop configuration while adding even only one intermediate node reduces notably the video performances.

The photo subsystem behavior is instead very good till the three hops configuration included, for both the higher and the medium transmission power levels. It loses some goodness in the four hops configuration. This time the lower transmission power level performs well for the first two configurations, diverging from the other monitored levels only when the second and the third intermediate nodes are added.

Finally, from the simulation performed with Tossim for the evaluation of a the *Selective Repeat* network protocol it is emerged the fairness of the adopted solution in case of noise channels.

8.1 Future Developments

The performances tests outcome underlines the main limitations of the video streaming architecture and the bottlenecks to which the system is subjected. Almost all of these limitations concerns the constrained resources with which we have to relate. Some of them are negotiable through a redesign of the most critical sections of the

system.

Packets Losses

With the current implementation the video system presents, in a few cases, a really inefficient behavior. Indeed, once a packet constituting a frame is lost, the reconstruction of that frame at the gateway is compromised: without such packet the C decoder does not have enough information to decode the entire frame. For this matter, losing a packet is the same as losing an entire frame. To overcome this situation it should be reimplemented the image coding process on the sensor side. Instead encoding the whole image matrix in a single compressed file, it could be convenient to divide the image in blocks coding them independently so that a packet lost would correspond to the loss of a single block. After all or some of the blocks have been correctly received, the gateway application can reassemble such subparts to reconstruct the original image.
However, to avoid notable video quality losses, also in the subsequent residuals frames, the block size must be as small as possible while to better exploit the intra frame correlation the block size should be as big as possible. This trade-off should be conveniently resolved to obtain the best performances from the system.

Multihop Interference

Another problem encountered during the tests is the increase of the inter-nodes interference. Due to the fact that the medium access follows the Carrier Sense Multiple Access with Collisions Avoidance (CSMA/CA) paradigm, the presence of many nodes in the network transmitting on the same frequency channel could generate many collisions and, as a consequence, many retransmissions or transmissions delays. A solution to this problem could be the implementation of a Time Division Multiple Access protocol that assigns the resource univocally. With this paradigm a support for different Quality of Service flows could also be implemented.

Different Network Protocol

Another possible future development could interests the network protocol. The current implementation checks the correct transmission of every packet with acknowledgements. When the acknowledgement signalling the correct reception of a packet is not received, the latter is retransmitted. In the photo case, the number of retransmissions has not limitations while a video packet could be resent, at most, 5 times.
If, due to the channel quality and the network configuration the majority of the transmitted packets is received after the first transmission, the delay yield by the ack wait could be avoided changing the network paradigm. For instance, only one cumulative ack could be sent after the delivery of a fixed number of packets. If some of the packets did not arrive to the destination node, the protocol could require their retransmission.

Bibliography

[1] Tinyos contributions source for intel mote 2: http://tinyos.cvs.sourceforge.net/viewvc/tinyos/tinyos-2.x-contrib/intelmote2/.

[2] Tossim website, http://docs.tinyos.net/index.php/tossim.

[3] University of california at berkeley. tiny os website, http://www.tinyos.net/.

[4] G.R. Aiello and G.D. Robertson. Ultra-wideband wireless systems. *IEEE Microwave magazine*, 2003.

[5] G. Bjntegaard, A. Luthra, T. Wiegand, and GJ Sullivan. Overview of the H. 264/AVC Video Coding Standard. *IEEE Transactions On Circuit and Systems for Video Technology*, 2003.

[6] E.O. Brigham and RE Morrow. The fast Fourier transform. *Spectrum, IEEE*, 4(12):63–70, 2009.

[7] R.R. Brooks, P. Ramanathan, and A.M. Sayeed. Distributed target classification and tracking in sensor networks. *Proceedings of the IEEE*, 91(8):1163–1171, 2003.

[8] E. Felemban, C.G. Lee, and E. Ekici. MMSPEED: Multipath multi-SPEED protocol for QoS guarantee of reliability and timeliness in wireless sensor networks. *IEEE Transactions on Mobile Computing*, pages 738–754, 2006.

[9] B. Girod, A.M. Aaron, S. Rane, and D. Rebollo-Monedero. Distributed video coding. *Proceedings of the IEEE*, 93(1):71–83, 2004.

[10] R. Holman, J. Stanley, and T. Ozkan-Haller. Applying video sensor networks to nearshore environment monitoring. *Pervasive Computing, IEEE*, 2(4):14–21, 2003.

[11] D. James, L. Klibanov, G. Tompkins, and S.J. Dixon-Warren. Inside CMOS Image Sensor Technology. *Chipworks White Paper*.

[12] R. Mukundan and O. Hunt. A comparison of discrete orthogonal basis functions for image compression. In *Proc. Conf. Image and Vision Computing New Zealand 04*, pages 53–58, 2004.

[13] G. Pekhteryev, Z. Sahinoglu, P. Orlik, and G. Bhatti. Image transmission over IEEE 802.15. 4 and ZigBee networks. In *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, pages 3539–3542. IEEE, 2005.

[14] R. Puri and K. Ramchandran. PRISM: A new robust video coding architecture based on distributed compression principles. In *PROCEEDINGS OF THE ANNUAL ALLERTON CONFERENCE ON COMMUNICATION CONTROL AND COMPUTING*, volume 40, pages 586–595. Citeseer, 2002.

[15] A. Redondi, M. Tagliasacchi, M. Cesana, L. Borsani, P. Tarrio, and F. Salice. Laura- localization and ubiquitous monitoring of patients for health care support. *IEEE APLEC, Istanbul, Turkey*, September 2010.

[16] A. Rowe, D. Goel, and R. Rajkumar. Firefly mosaic: A vision-enabled wireless sensor networking system. In *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International*, pages 459–468. IEEE, 2007.

[17] A. Said and W.A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *Circuits and Systems for Video Technology, IEEE Transactions on*, 6(3):243–250, 2002.

[18] CE Shannon. A Mathematical Theory of Communications. Bell Systems Technical Journal, 27: 623, 1948.

[19] L. Shu, Z.B. Zhou, M. Hauswirth, D. Le Phuoc, P. Yu, and L. Zhang. Transmitting streaming data in wireless multimedia sensor networks with holes. In *Proceedings of the 6th international conference on Mobile and ubiquitous multimedia*, pages 24–33. ACM, 2007.

[20] D. Slepian and J. Wolf. Noiseless coding of correlated information sources. *Information Theory, IEEE Transactions on*, 19(4):471–480, 2002.

[21] M.C. Vuran and I.F. Akyildiz. Cross-layer analysis of error control in wireless sensor networks. In *Sensor and Ad Hoc Communications and Networks, 2006. SECON'06. 2006 3rd Annual IEEE Communications Society on*, volume 2, pages 585–594. IEEE, 2007.

[22] A.D. Wyner. The rate-distortion function for source coding with side information at the decoder\ 3-II: General sources. *Information and Control*, 38(1):60–80, 1978.