

Packet Loss Performance and MAC Design for Estimation in Wireless Sensor Networks

DAVID PALLASSINI



**KTH Signals
Sensors and Systems**

Master's Degree Project
Stockholm, Sweden 2005

IR-RT-EX-0522

Abstract

Recent advances in wireless communications and electronics has enabled the development of low-cost sensor networks. Wireless sensor networks can be used for various application areas (e.g., military, home, environment). Each application poses its own particular technical issues. One key problem that arises in many applications is how to organize the computation and communication in a sensor network to enable accurate estimation of the state of a dynamic system. As a starting point in this thesis, we demonstrate that packet loss and delays have a strong influence on the performance of optimal estimators. Thus, when we want to use a sensor network platform to collect and forward measurements to a data collector for estimation, it is critical to understand the loss performance of current hardware platforms and, possibly, to design MAC schemes that eliminate losses and minimize data collection latency. We investigate on the problem of packet loss, with practical experiments on a real wireless sensor network built up with Telos platform. The experiments in this thesis are made with single hop and multi-hop protocols in a star and linear network topology. To solve the problem of data delay and to avoid the collisions that lead to packet losses, we propose an overlay TDMA over CSMA. The underlying idea is to have a distributed protocol which is able to schedule the transmission of the nodes in order to have a data flow that converges from the furthest nodes toward the fusion center. To show the benefits of our protocol we simulate the behavior of a network in which nodes are randomly scheduled by the fusion center. Our protocol permits to obtain, in a distributed way, the optimal solution of such a network giving a strong improvement over a casual schedule.

Acknowledgements

First of all I would like to thank my supervisor, Professor Mikael Johansson, for taking me as a student and encouraging me to work abroad. He always helped me when I had questions or wonders, being really kind and helpful throughout this project. Without his great guidance, I would have never completed this work.

I would also like to thank my advisor PhD Pablo Soldati, I am grateful for all the time he spent on introducing me to Stockholm and KTH and helping me with my work.

I would like to express my sincerest gratitude to the following people:

Dr. Carlo Fischione at KTH for his helpful feedbacks on this work.

Prof. Andrea Abrardo in Siena, who is my examiner, for giving me the opportunity to perform this work at KTH.

Also I would like to address my gratitude to the group at exjobbarum which made my time spent at KTH interesting and worthy of remembrance.

Last but not least, my dear parents, my brothers and my girlfriend Manuela for always being there for me, supporting me and encouraging me whenever needed.

David Palladini

Contents

Introduction	1
Literature survey	1
Our field of interest	2
Problem definition	3
Research approach	3
Our work	4
Expected results and thesis outline	5
0.1 Acronyms	6
1 Control and estimation under packet loss	8
1.1 Control under packet loss	8
1.2 Estimation under packet loss	10
1.2.1 Kalman filtering	10
1.2.2 Optimal filtering under packet loss	12
1.3 The dependence on the loss process	13
1.3.1 Gilbert-Elliot model	13
1.4 The influence of delays	15
1.5 Issues for estimation in sensor networks	15
2 Introduction	17
2.1 Sensor networks application	18
2.1.1 Military applications	19
2.1.2 Environmental applications	19
2.1.3 Health applications	19
2.1.4 Home applications	20
2.1.5 Other commercial applications	20
2.2 Factors influencing sensor network design	20
2.2.1 Fault tolerance	20

2.2.2	Scalability	21
2.2.3	Production costs	21
2.2.4	Hardware constraints	21
3	Moteiv Tmote sky	22
3.1	Introduction	22
3.2	Compiling and Loading applications	23
3.3	Communication between motes	24
3.3.1	Packet broadcasting	25
3.3.2	Multihop Routing	25
4	Standard IEEE 802.15.4	28
4.1	Components of a Wireless Personal Area Network (WPAN)	28
4.2	Network topology	29
4.2.1	Star topology	29
4.2.2	Peer-to-peer topology	29
4.2.3	Cluster-tree network	30
4.3	IEEE 802.15.4 Physical layer	31
4.3.1	2400 MHz band	31
4.3.2	Lower bands	32
4.3.3	Modulation technique in IEEE 802.15.4	32
4.4	IEEE 802.25.4 Medium Access Control layer	34
4.4.1	Importance of the duty cycle in a WSN	34
4.4.2	Duty cycle organization	35
4.4.3	Superframe structure	35
4.4.4	CSMA-CA algorithm	37
4.4.5	Beacon generation	38
4.4.6	Synchronization	38
4.4.7	Data transfer	40
4.4.8	Frame format	42
4.4.9	Robust communications	45
5	TinyOS software for measuring radio performance	46
5.1	Single hop	46
5.1.1	CountRadio	46
5.1.2	Meaning of the packet stream	47
5.2	Multihop	49
5.2.1	Aegis	50

5.2.2	Aegis packet format	50
5.2.3	Meaning of the Aegis packet stream	51
5.2.4	Aegis performances	53
5.2.5	Our work on Aegis	53
5.2.6	The Surge application	56
5.2.7	Performance of the Surge application	56
6	Experiments in a real Wireless Sensor Network	58
6.1	Loss rate with AEGIS	58
6.1.1	Normal environment	59
6.1.2	Quiet environment	60
6.1.3	Unexpected result	60
6.2	Our application	61
6.2.1	Results of a star topology	61
6.3	Baud rate and UART	64
6.3.1	Results with different baud rate	67
6.3.2	Experiment with a 115,2 Kb/s baud rate	67
7	Design of an overlay TDMA	69
7.1	Introduction	69
7.1.1	Assumptions	70
7.1.2	Proposed solution	70
7.2	Method analysis	73
7.3	Method M1	73
7.4	Method M2	76
7.5	Conclusions results	77
8	Conclusions & future work	82
8.1	Conclusions	82
8.2	Future work	83

List of Figures

1.1	Lossy model	8
1.2	Estimator performance for different loss rates.	11
1.3	The performance of the optimal estimator.	12
1.4	Gilbert-Elliot loss model	14
1.5	The performance of the optimal estimator under varying time-delay.	16
3.1	Telos Tmote-sky	22
3.2	Example of packet broadcasting	26
3.3	Example of multihop routing	27
4.1	Star and Peer To Peer topology in IEEE 802.15.4 standard . .	30
4.2	Cluster tree topology in IEEE 802.15.4 standard	31
4.3	Frequency bands and data rates	32
4.4	BER versus SNR. Picture taken from http://www.embedded-computing.com	33
4.5	Duty cycle with beacon enable mode in IEEE 802.15.4 standard	35
4.6	Direct data transmission in beacon-enable and non-beacon enable	41
4.7	Indirect data transmission in beacon-enable and non-beacon enable	42
4.8	General MAC frame in the 802.15.4	43
4.9	Frame formats.	44
5.1	General structure of a CountRadio packet	47
5.2	Strings of data	48
5.3	Aegis message headers	51
5.4	Aegis packets in hexadecimal format	52

6.1	Linear topology with 4 sensors	59
6.2	Losses in the normal environment	60
6.3	Losses in the quiet environment	61
6.4	Configuration of our real network	62
6.5	Percentage of lost packets with different sampling rates	63
6.6	Loss rate vs number of sensors	64
6.7	Platform.properties file	66
6.8	Number of packets per second at different baud rates	67
6.9	Comparison between two different baud rate	68
7.1	Procedure to create the schedule	72
7.2	Sensors network	74
7.3	Latency analysis in Method M1.	78
7.4	Latency analysis in Method M2.	79
7.5	Latency analysis in Method M2 in the optimal case.	80
7.6	Possible schedule for the optimal solution	80
7.7	Comparison between Method M2 and M1.	81

List of Tables

1.1	Influence of burstiness on the estimator performance	14
-----	--	----

Introduction

Recent advancement in wireless communications and electronics has enabled the development of low-cost sensor networks. The sensor networks can be used for various application areas (*e.g.*, military, home, environment). Each application poses its own particular technical issues. One of the most important research field is inherent to solve the problem of the estimation of a parameter.

Literature survey

Many algorithms have been developed throughout the last couple of years trying to answer questions like, how can we best derive an estimate of parameters using data collected by sensors? What is the solution to the problem of the optimal power scheduling for decentralized estimation in a sensor network?

In a network of a distributed sensors, each one affected by independent Gaussian noise [36], a scheme based on distributed average consensus in the network is used to compute the maximum-likelihood estimate of parameters. Step by step, each node can compute a local weighted least-square estimate, which converges to the global maximum-likelihood solution.

Another problem to be considered is the optimal power scheduling for the decentralized estimation of a noise-corrupted deterministic signal in an inhomogeneous sensor network. It has been proved that the optimal quantization level and transmission power for each sensor can be determined jointly the channel gains and the local observation noise level [35].

Life time maximization, *i.e.* energy consumption minimization, is closely related to the estimation accuracy: when sensors die the accuracy of the estimation gets worst and worst, mainly if the sensors that are out of

battery are concentrated in the same area. A way to save energy is given in [12], where the authors implement a distributed estimation algorithm, more flexible in energy-accuracy subspace and more robust than the snapshot aggregation and it also brings considerable energy savings for a typical accuracy requirement. This algorithm depends on the chosen threshold, which should be based on both the mean and the variance. In this scheme, a node broadcasts its value only if the difference between the new generated estimate for every neighbor and the old estimate is beyond that preset threshold for any of the neighboring nodes.

A decentralized incremental algorithm for performing in-network optimization can be adopted to get robust estimation techniques that attempt to identify and discard bad measurements from the estimation process [32].

Another way to save energy is given in [20]. Total transmission energy consumption in a sensor network, can be minimized, if quantization levels for sensors are determined jointly by the fusion center using information about correlation of sensor observation. In this paper is also given an approximate suboptimal solution to the energy minimization problem achieving the same target estimation performance as the optimal solution.

Our field of interest

Our work begins with a theoretical investigation on one of the most important problem on WSN: **estimation of the loss process**.

We attempt to answer questions like

- How does the estimator performance depend on the loss process?
- What does the typical loss process look like?
- How does the loss process depend on physical layer parameters?
- How can we do a good combined PHY/MAC & estimator design?

We present our model for the estimation from lossy sensor data and we push through many experiments in a real WSN built up using Telos mote*iv* Tmote sky where both a single hop and a multihop protocol are used.

Problem definition

The motes available at the department S3, (*moteiv* Tmote sky, [4]) in their basic configuration, have shown a very slow sampling rate (around 1Hz) where with *sampling rate* we mean the capability of the base station to sample each sensor. The cause of this problem might be due to several aspects:

- routing overhead
- a bug in the code
- high number of collision since a CSMA scheme is used
- the small output buffer of the node.

This means that if there is a packet waiting to be transmitted (for example, a packet to build the routing tree) and the node tries to send a data packet (for example, with sensor readings), then the second packet will be discarded or corrupted.

We are interested in improving the performances of the motes in terms of sampling rate and analyze the latency of a WSN using a TDMA scheduling.

Research approach

Moteiv do not allow scheduling in the communication level, but use a CSMA scheme to solve the channel contention, hence there is not any form of synchronization. Even this might be sufficient if in a static system, we are interested in seeing the advantages of using an overlay TDMA protocol over CSMA. Since a TDMA scheme is used, in a hierarchical WSN, the fusion center should be able to establish the best schedule that guarantees the minimization of network delay, *i.e.* sensors should be scheduled in order to get the highest number of packets at the fusion center in the minimum amount of time. Basically this requires that sensors are scheduled so that the first ones to transmit are the furthest ones from the fusion center. Using a TDMA scheme in order to minimize the packet latency we have to distinguish between two transmission phases:

- **Distribution phase:** this corresponds to the downlink communication where the base station makes a TDMA schedule with the purpose to minimize the latency and then sends it to the sensors
- **Acquisition phase:** this corresponds to the uplink communication where the sensors use the received schedule to transmit their own data as well as forwarding packets belonging to other users

In particular, the fusion center should broadcast the message containing the TDMA schedule to inform the sensors which time-slot have been them assigned. Note that, in this case, we assume that the fusion center has full knowledge of the network configuration.

Our work

We are interested in improving the performances of the motes in terms of sampling rate and see what happens with the loss rate using the maximal sampling rate. To reach this goal we show how to change the nesC code [16] of the TinyOS [8] applications. We are also interested in seeing the performances of a WSN using a TDMA scheme to solve the channel contention. In particular we want to prove that a TDMA schedule can give a lower latency and lower loss rate compared with a CSMA. To reach this goal, we have adopted two different approaches to generate TDMA schedule in WSN. Due to the symmetric nature of the problem, we will address only on data acquisition phase (from nodes to fusion center). We assume to have a fixed number of nodes and links, say N and L respectively, one fusion center toward which the information converges. We have defined two methods to generate a TDMA schedule:

- **Method M1:** the frame length is fixed to L time-slots (here intended as right of transmission) and each one is divided in L "mini" time-slot. A time slot allows a node to empty its buffer
- **Method M2:** Each time-slot allows one packet to be transmitted but we can decide the length of the frame and thus, a lot more time-slots to certain links.

Expected results and thesis outline

We are going to prove that programming the motes with a new code makes possible to improve their performances in terms of transmission rate. We are also going to analyze and simulate the two schemes above mentioned and show which is the better to obtain a lower latency. Finally we will show that our protocol permits to reach one of the best solution of the two methods in a distributed way.

Chapter 1: Control and estimation under packet loss. This chapter investigates the influence of the packet loss, loss process and delays, on the performance of the Kalman filter. This investigation gives the basis to our work and motivates the experiments in chapter 8.

Chapter 2: Introduction on wireless sensor networks. Wireless sensor networks is a new technology to observe physical phenomena and react to it. In this chapter we give a description of the issues of a wireless sensor network and some general sensor' features.

Chapter 3: Moteiv Tmote sky. This chapter gives a brief description of the motes used in our experiments and shows the procedures to follow in order to use them.

Chapter 4: IEEE 802.15.4 standard. The motes used in our experiments are equipped with a transceiver compliant with the IEEE 802.15.4. In this chapter we give a brief description of this standard focusing on the importance of the beacon order to guarantee the quality of service of a WSN.

Chapter 5: TinyOS software for measuring radio performance. Here are shown the two network topologies adopted in our experiments and the improvements given to the software in this work.

Chapter 6: Experiment in a real wireless sensor network. In this chapter is shown the performance of a real wireless sensor network built with Telos motes.

Chapter 7: Design of an overlay TDMA In this chapter we propose a distributed algorithm that permits to have the minimal latency and to avoid the collisions in a wireless sensor network.

Chapter 8: Conclusions & future work Finally, in the last chapter, conclusions are given and future work is explained.

0.1 Acronyms

AM	Active Message
BE	Back Off Exponent
BER	Bit Error Rate
BI	Beacon Interval
BO	Beacon Order
BPSK	Binary PSK
CAP	Contention Access Period
CHL	Cluster Head
CID	Cluster Identifier
CIR	Carrier-to-Interference Ratio
CDMA	Code Division Multiple Access
CLH	Cluster Head
CFP	Collision Free Period
CSMA	Carrier Sense Multiple Access
CSMA/CA	Carrier Sense Multiple Access / Collision Avoidance
CW	Contention Window
DARPA	Defense Advanced Research Project
FSK	Frequency Shift Keying
FCF	Frame Control Field
FCS	Frame Check Sequence
FFD	Full Function Device
GTS	Guaranteed Time Slot
GUI	Graphical User Interface
IFS	Inter Frame Space
ISM	Industry Scientific and Medical
MAC	Medium Access Control
MEMS	Micro-Electro-Mechanical System
MFR	MAC Footer
MHR	MAC Header
MPDU	MAC Frame
MSDU	MAC Service Data Unit
NEST	Network Embedded Software Technology
PAN	Personal Area Network
PAR	Photosynthetically Active Radiation
PHR	PHY Header
PHY	Physical Layer
PPDU	PHY Data Packet
PSDU	PHY Data Frame Payload
PSK	Phase Shift Keying
QoS	Quality of Service
QPSK	Quadrature PSK
RAM	Random Access Memory
RF	Radio Frequency
RFD	Reduced Function Device
SD	Superframe Duration
SO	Superframe Order

SHR	Synchronization Header
SNR	Signal to Noise Ratio
TDMA	Time Division Multiple Access
TSR	Total Solar Radiation
WI-FI	Wireless Fidelity
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network

Chapter 1

Control and estimation under packet loss

This chapter investigates the influence of the packet loss, the burstiness and the latency, on the performance of the optimal estimator. This investigation gives the basis to our work and motivates the experiments in the next chapters.

1.1 Control under packet loss

We model each sensor as an output of a linear stochastic system with random losses of the sensor output samples. The discrete transitions are modeled as a Markov chain, so that the loss events are modeled as Markov processes.

In our model we consider only one data collector toward which all the information converges. The data collector attempts to estimate a real vector quantity from the data received by the other nodes in the network.

We consider the system model in Figure 1.1.

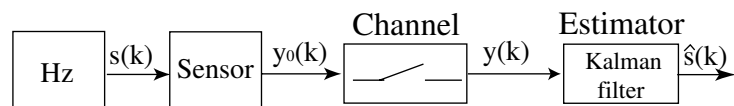


Figure 1.1: Lossy model

All signals are in discrete time.

The detected signal $s[k]$ can be modeled as the output of a linear time-invariant filter H_z . Typically the original signal $s[k]$ is assumed to be slowly varying and H_z is a low pass with a cut off frequency of appropriate bandwidth.

The filter H_z can itself be described in a various ways, but in our work we represent it in a state space form as in 1.1

$$\begin{aligned} x[k+1] &= Ax[k] + Bu[k] + v[k] \\ H_z : \quad s[k] &= Cx[k] + e[k] \end{aligned} \tag{1.1}$$

where x is the system state, u is the control, v is the state disturbance, e is the measurement errors and A , B and C are the state-space matrices describing the filter H_z . We will assume that v and e are zero-mean stationary stochastic processes, and that

$$\begin{aligned} \mathbf{E}\{x[0]\} &= 0 \\ \mathbf{E}\{x[k]x^T[k]\} &= R_0 \\ \mathbf{E} \begin{pmatrix} v[k] \\ e[k] \end{pmatrix} \begin{pmatrix} v[k] \\ e[k] \end{pmatrix}^T &= \begin{pmatrix} R_1 & 0 \\ 0 & R_2 \end{pmatrix} \end{aligned}$$

The output of the sensor, due to the internal noise and other imperfections, may differ from the detected signal $y[k]$. The output samples of the sensor are transmitted over a lossy sample-erasures channel. This means that the sample are either received correctly or completely lost. The output of the sensor can be expressed as

$$y_0(k) \quad k = 1, \dots, N$$

Since we model the channel as an erasure channel we can express its output as

$$y[k] = \begin{cases} y_0(k) & \text{if sample } k \text{ is not lost} \\ 0 & \text{if sample } k \text{ is lost} \end{cases} \tag{1.2}$$

This erasure model is well-suited for digital communications where the samples $y_0(k)$ are digitally encoded and transmitted and channel losses

correspond to the losses of the samples. The output of the lossy channel is connected to an estimator which attempts to estimate the original signal $s[k]$ taking into account the imperfections of the sensors and the losses of the channel. The estimator output is $\hat{s}[k]$.

1.2 Estimation under packet loss

In this section, we will assume that no control acts on the system, *i.e.*, that $u[k] = 0$ for all k . This results in the system

$$\begin{aligned}x[k+1] &= Ax[k] + v[k] \\s[k] &= Cx[k] + e[k]\end{aligned}\tag{1.3}$$

We will look for estimators that try to minimize the estimation error variance

$$\mathbf{E} \{ (x[k] - \hat{x}[k])^T (x[k] - \hat{x}[k]) \}$$

1.2.1 Kalman filtering

In this setting, the optimal estimator is known as the *Kalman filter*. The estimator has the structure

$$\hat{x}[k+1] = A\hat{x}[k] + K[k](y[k] - \hat{y}[k])\tag{1.4}$$

where the estimator gain $K[k]$ is given by

$$K[k] = (AP[k]C^T)(CP[k]C^T + R_2)^{-1}\tag{1.5}$$

and $P[k]$ is given by the recursion

$$P[k+1] = AP[k]A^T + R_1 + K[k](CP[k]C^T + R_2)K^T[k]\tag{1.6}$$

Under mild assumptions, the iteration converges and its stationary solution P can be directly obtained via the Riccati equation

$$P = APA^T + R_1 - (APC^T)(CPC^T + R_2)^{-1}(APC^T)^T$$

and the associated stationary filter gain is given by replacing $P[k]$ by P in (1.5).

As an aside, please note that the stationary covariance of the state vector $\mathbf{E}\{xx^T\}$ is given by

$$P = APA^T + R_1$$

and the output variance (under independence of e and v) is $\text{Tr}CPC^T + R_2$.

EXAMPLE 1 (The influence of packet loss). *To estimate the influence of packet loss of the estimator performance, we consider the system*

$$\begin{aligned} x[k+1] &= \begin{bmatrix} 1.9703 & -0.5901 \\ 2.0000 & 0.0000 \end{bmatrix} x[k] + w[k] \\ s[k] &= [0.0397 \quad 0.0197] x[k] + e[k] \end{aligned}$$

(the system is generated by a continuous-time linear system with unit gain, natural frequency one, and damping 0.1 sampled with period $h = 0.1$). We let $R_1 = I_2$ and $R_2 = 1$. Figure 1.2 shows the performance of two estimation schemes for varying packet loss rates. The “hold” scheme simply keeps the estimated state vector unchanged if no new data has arrived, while the “state update” scheme updates the state using a linear prediction (effectively setting $K=0$) in the presence of data loss.

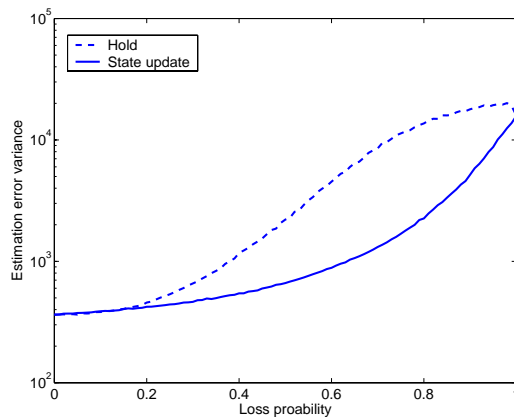


Figure 1.2: Estimator performance for different loss rates.

Although this example illustrates the influence of packet losses on estimation performance and the impact of the behavior of the estimator in case of data loss, none of the simulated schemes are optimal. We will now proceed by studying the optimal estimator under packet loss.

1.2.2 Optimal filtering under packet loss

Intuitively, one can see the packet loss as a data packet with infinite variance (that is $R_2 \rightarrow \infty$ in face of packet loss). The optimal estimator is then the time-varying Kalman filter above as long as data is received, while when no data is received one should perform the state propagation

$$\begin{aligned}\hat{x}[k+1] &= Ax[k] \\ P[k+1] &= AP[k]A^T + R_1\end{aligned}$$

The following example demonstrates the benefits of the time-varying estimation scheme.

EXAMPLE 2. *Optimal estimation under packet loss* Figure 1.3 demonstrates the performance of the optimal estimator in relation to the previous scheme. We note that the performance improvements of a time-varying gain is quite moderate in this case.

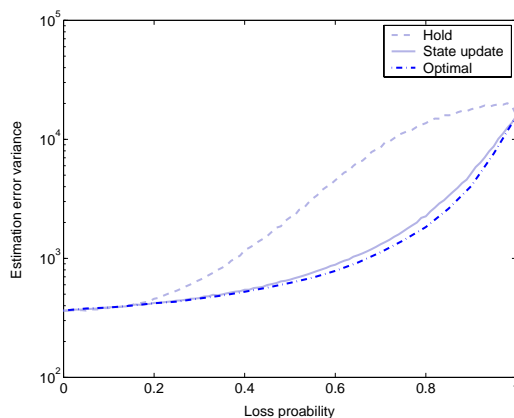


Figure 1.3: The performance of the optimal estimator.

1.3 The dependence on the loss process

So far, we have assumed independent losses. However, in reality it is common that errors and losses occur in bursts. A common model for bursty losses is the so-called Gilbert-Elliot model.

1.3.1 Gilbert-Elliot model

The Gilbert-Elliot model is basically a hidden Markov chain, that is a stochastic process with a countable state space. The Markov chain resides in one of the states at each time instance, and the probability of going to another state is a (time-stationary) function of the present state. With a Markov chain process hidden behind, the statistics of the observation process of a hidden Markov chain is determined by the present state from its associated Markov chain process. We can represent The Gilbert-Elliot model as a two-state hidden Markov chain (Figure 1.4), in which the two states are denoted as “No Loss” (which corresponds to the good state) and “Loss” (which corresponds to the bad state). We can express the transition probability as

$$\begin{aligned} p_{1,1} &= 1 - p_{1,2} = \lambda_1 \\ p_{2,1} &= 1 - p_{2,2} = \lambda_2 \end{aligned} \tag{1.7}$$

For some probabilities λ_1 and λ_2 . The parameter λ_2 represents the probability of going from the non lossy to the lossy state. The permanence of the system in the lossy state is an exponentially distributed time with mean $1/(1 - \lambda_2)$. The stationary distribution of the Markov chain is

$$q_1 = 1 - q_2 = \frac{\lambda_2}{1 - \lambda_1 + \lambda_2} \tag{1.8}$$

The value of q_1 in (1.8) is the overall loss probability and, it reduces to $q_1 = \lambda$ for $\lambda_1 = \lambda_2 = \lambda$ (independent losses).

Usually λ_2 is small hence, the bad state tends to persist for a long time. This emulates the burst error state. Sometime it is helpful to introduce a parameter $\alpha = 1 - \lambda_2 + \lambda_1$, which can range from -1 to 1 and represents the strength of the memory of the channel. The more realistic case is when

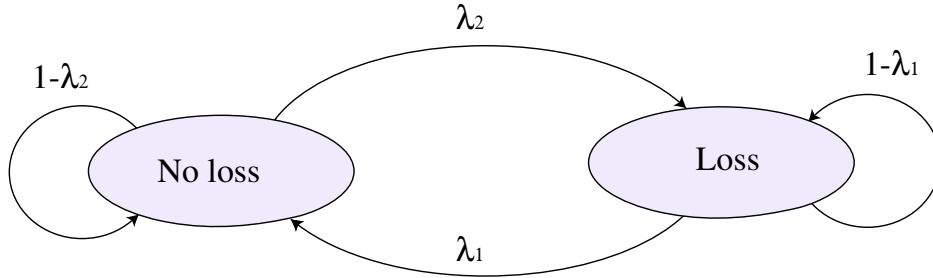


Figure 1.4: Gilbert-Elliot loss model

Expected burst-length	2	5	10	20	50	100
Estimator error variance ($\times 10^3$)	0.883	1.903	3.168	4.410	6.415	7.713

Table 1.1: Influence of burstiness on the estimator performance

α approaches to 1; in this case the channel tends to remain in the same state for a log period of time creating long burst of errors. A less realistic case, is when α tends to -1, which corresponds to a channel which switches rapidly between the two states.

The example below demonstrates how the performance of the optimal estimator depends on the burstiness of the loss process.

EXAMPLE 3 (Dependence of the loss process). *We simulated the optimal estimator for the Gilbert-Elliot model, keeping the loss probability fixed to $p = 0.4$ while altering the expected burst-length m . The results are shown in the Table 1.1*

The simulations illustrate the intuitive: the performance decreases as the burstiness increases. In particular, as the burstiness grow large, the performance tends to $J_0 + p_{\text{loss}}(J_1 - J_0)$ where J_0 is the optimal performance when there are no losses, and J_1 is the optimal performance when no data arrives (that is, the stationary variance of the open-loop system).

1.4 The influence of delays

In addition to packet losses, delays are another factor that limits the achievable performance. To get some insight into how the observation delay influences the estimator performance, we consider an augmented system with state vector

$$z[k] = (x[k] \quad x[k-1] \quad \cdots \quad x[k-d])$$

and the observation is $s[k] = Cx[k-d] + e[k]$. Such systems can be written on the form (1.3), and the optimal estimators can be computed as above. We will not present the general case, but note that for the case of a single delay we have

$$\begin{aligned} z[k+1] &= \begin{bmatrix} A & 0 \\ I & 0 \end{bmatrix} z[k] + w[k] \\ s[k] &= [0 \quad C] z[k] + e[k] \end{aligned}$$

and $\mathbf{E}\{ww^T\} = \begin{pmatrix} I & 0 \\ 0 & R_1 \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}^T$. The following example evaluates the optimal estimator performance for varying delays

EXAMPLE 4 (The influence of delays). *Consider again the linear system used in the examples above. Figure 1.5 shows the optimal performance for varying delay (this investigation does not include any packet losses). Clearly, the latency also plays a critical role for the estimator performance.*

1.5 Issues for estimation in sensor networks

The investigations above demonstrate that packet loss and delays have a strong influence on the performance of estimators. Thus, when we want to use a sensor network platform to collect and forward measurements to a data collector for estimation, it is critical to understand the loss performance of current hardware platforms and, possibly, to design MAC schemes that eliminate losses and minimize data collection latency. These will be the topics for the rest of this thesis.

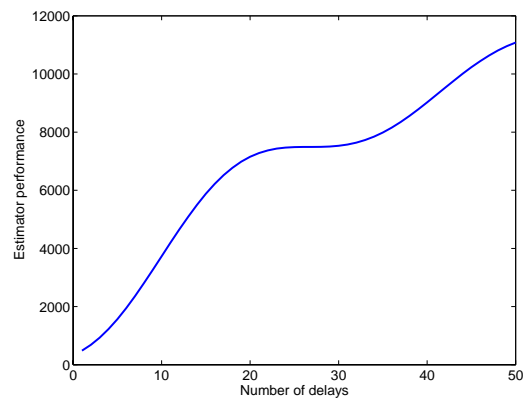


Figure 1.5: The performance of the optimal estimator under varying time-delay.

Chapter 2

Introduction

Wireless sensor networks is a new technology for scientists and engineers to observe physical phenomena and react to it. One recent example of this is the deployment of hundreds of wireless node on an uninhabited island of Maine to monitor the environmental factors affecting the sea birds' comings and goings [25]. In one of Intel's factory, wireless sensors measure the subtle vibrations of various machines to detect malfunctions before the equipment breaks down [7]. At the University of California at Berkeley, sensors embedded in the walls of a building to diagnose its seismic stability after a simulated earthquake [2].

The building blocks of these wireless networks are "motes", self-contained, battery-powered computers that measure light, temperature, humidity, and other environmental factors. Developed by Intel Research in collaboration with the University of California at Berkeley, the motes self-organize into ad-hoc wireless networks. The data is then relayed from a mote to another until it reaches its desired destination for processing.

There are many technological hurdles that must be overcome for a wireless sensor networks to become practical. Nearly everything we take for granted in desktop computing is at a premium in wireless sensor networks: the individual motes are incredibly resource constrained; they have limited processing speed, storage capacity, and communication bandwidth; their lifetime is determined by their ability to conserve power. All of these constraints require new hardware designs, software applications, and network architectures that maximize the motes capabilities while keeping them inexpensive to deploy and maintain. Additionally, it must be easy for non-computer scientists with just basic training in the technology to

extract meaningful real-world data from the networks.

Sensing and measuring itself is not new, but in the past years the cost and bulk of sensing technology meant that only a handful of sensors could be deployed for most applications. However, data averaged from a few sensors does not truly represent the real world. Now, thousands of sensors can be scattered throughout a single physical space, providing a much higher resolution picture of the real world than ever before. This entirely new approach to instrumentation is made possible by the intersection of several technological trends. According to Moore's Law, the number of transistors on a microchip doubles approximately every two years and at the same time, microprocessors with a given computing capacity are becoming smaller and cheaper with every passing year. While silicon scaling marches on, the same semiconductor manufacturing processes are being utilized to build microscopic mechanical structures that interact with the physical world. This technology, called MEMS (micro-electro-mechanical systems), enables the production of velocity sensors, thermometers, and even low-power radio components that fit on the head of a pin and cost just pennies each. These three hardware ingredients (microprocessors, MEMS sensors, and low-power radios) make up a sensor node, or "mote". The "mote" nickname comes from UC Berkeley's Smart Dust project, an effort funded by the Defense Advanced Research Projects Agency's (DARPA) Network Embedded Software Technology (NEST) program to shrink the devices down to dust mote size through the power of Moore's Law.

Even if the low cost and small dimension are an important issue, what enables the deployment of hundreds of these nodes is their multihop networking capabilities.

2.1 Sensor networks application

Sensor networks may consist of many different types of sensors such as seismic, magnetic, thermal, visual, infrared, acoustic and radar, which are able to monitor a wide variety of ambient conditions [9].

Sensor nodes can be used for continuous sensing, event detection, location sensing, and local control of actuators. The concept of micro-sensing and wireless connection of these nodes promise many new application areas. The most important applications are enumerated in the next sec-

tion.

2.1.1 Military applications

The rapid deployment, self-organization and fault tolerance characteristics of sensor networks make them a very promising sensing technique for military command, control, communications and surveillance. Since sensor networks are based on the dense deployment of disposable and low-cost sensor nodes, destruction of some nodes by hostile actions does not affect a military operation as much as the destruction of a traditional sensor, which makes sensor networks concept a better approach for battlefields.

Some of the military applications of sensor networks are monitoring of friendly forces, equipment and ammunition, battlefield surveillance, reconnaissance of opposing forces and terrain, targeting, battle damage assessment and nuclear, biological and chemical (NBC) attack detection and reconnaissance [9].

2.1.2 Environmental applications

Environmental applications of sensor networks include tracking the movements of birds, small animals, and insects; monitoring environmental conditions such pollution or forest fire detection [13, 25]. Since sensor nodes may be strategically, randomly, and densely deployed in a forest, sensor nodes can relay the exact origin of the fire to the end users before the fire is spread uncontrollable. Hundreds of sensor nodes can be deployed and integrated using radio frequencies/optical systems. Also, they may be equipped for example with solar cells [18], because the sensors may be left unattended for months and even years. The sensor nodes will collaborate with each other to perform distributed sensing and overcome obstacles, such as trees and rocks, that block wired sensors' line of sight.

2.1.3 Health applications

Some of the health applications for sensor networks are providing interfaces for the disabled, integrated patient monitoring, diagnostics, drug administration in hospitals, monitoring the movements and internal proces-

ses of insects or other small animals, telemonitoring of human physiological data and tracking and monitoring doctors and patients inside a hospital [29].

2.1.4 Home applications

As technology advances, smart sensor nodes and actuators can be buried in appliances, such as vacuum cleaners, micro-wave ovens and refrigerators. These sensor nodes inside the domestic devices can interact with each other and with the external network via the Internet or Satellite. They allow end users to manage home devices locally and remotely more easily.

2.1.5 Other commercial applications

Some of the commercial applications are monitoring material fatigue, building virtual keyboards, managing inventory, monitoring product quality, constructing smart office spaces, environmental control in office buildings, robot control and guidance in automatic manufacturing environments, interactive toys, interactive museums, local control of actuators, detecting and monitoring car thefts.

2.2 Factors influencing sensor network design

A sensor network design is influenced by many factors, which include fault tolerance; scalability; production costs; operating environment; sensor network topology; hardware constraints; transmission media; and power consumption.

2.2.1 Fault tolerance

Some sensor nodes may fail or be blocked due to lack of power, have physical damage or environmental interference. The failure of sensor nodes should not affect the overall task of the sensor network. This is the reliability or fault tolerance issue

2.2.2 Scalability

The number of sensor nodes deployed in studying a phenomenon may be in the order of hundreds or thousands. The new schemes must be able to work with this number of nodes. They must also utilize the high density nature of the sensor networks.

2.2.3 Production costs

Since the sensor networks consist of a large number of sensor nodes, the cost of a single node is very important to justify the overall cost of the networks. If the cost of the network is more expensive than deploying traditional sensors, then the sensor network is not cost-justified. As a result, the cost of each sensor node has to be kept low.

2.2.4 Hardware constraints

While the particular size and type of motes that form a network are mostly determined by the intended application, all of the devices face the same overarching design constraint: due to the difficulty and sometime impossibility to replace a mote, it must have the ability to conserve power. Ideally, each mote should be able to survive on its own for at least a year on a pair of AA batteries. This means that the motes need to be able to run at extremely low duty cycles. A mote should stay in its low-power mode for most of its life-time and “wake up” only to take scheduled readings or to transmit or receive data from neighboring devices.

Motes’ hardware and software components are designed to support low duty cycles. Simple microcontrollers like those that function as a mote’s brain can operate with just a mW of power when active, or 1-10 μ W in standby mode. A mote’s memory must also be limited due to the energy constraints. Each mote typically has less than 10 kilobytes of RAM and around one hundred kilobytes of software. All told, that’s approximately 10,000 times less data storage than a desktop PC.

Chapter 3

Mote*iv* Tmote sky

In this chapter we give a brief description of the Telos motes (Tmote sky, Fig. 3.1) available in the S3 department at KTH Stockholm and we will explain what is necessary to know in order to use them.



Figure 3.1: Telos Tmote-sky

3.1 Introduction

Tmote sky is the next-generation mote platform for extremely low power, high data-rate, sensor network applications designed with the dual goal of fault tolerance and development ease. Tmote Sky boasts the largest on-chip RAM size (10kB) of any mote, the first IEEE 802.15.4 radio, and an integrated on-board antenna providing up to 125 meters range outdoors and 50 meters indoor. For more details see [4] and [15]

Each node has a set of onboard integrated sensors and can operate as an independent node that can communicate with the other nodes collecting, receiving and sending data in the network.

The Tmote can be programmed using TinyOS [8]. TinyOS is a modular open-source operating system suitable for sensor network requirements. It provides a set of modules that can be linked together and loaded to sensors in order to obtain a complex application.

The default TinyOS package comes with a library composed by data acquisition tools, sensors drivers, network protocols and other useful interfaces. All the tools are directly available for the user and can also be improved or modified in order to create a own user's application.

The modularity of TinyOS assures the minimization of the software loaded into the mote, since only the files one really needs in order to run his application are loaded, thus minimizing the total amount of needed memory.

3.2 Compiling and Loading applications

In order to use the motes we need first to compile the required application on the PC and then to load it into the motes by a bootstrap loader.

Applications are written in nesC [16], a new language especially developed for network embedded systems such as sensor networks. NesC has a C-like syntax, but supports the TinyOS concurrency model. This makes a distinction between asynchronous code, which can execute in interrupt context, and synchronous code, which can only execute in tasks. Essentially, synchronous code is non-preemptive; it runs to completion with respect to other synchronous code. Asynchronous code, however, can preempt itself and synchronous code. This means that if a variable is read or written in asynchronous code, there is a possible race condition; it can preempt a concurrent access. Variables that can be modified by asynchronous code must be protected by atomic sections, which ensure atomic access (for further information about task, atomic and interrupt see [5])

NesC supports mechanisms for structuring and linking together software components into robust network embedded systems. Components are linked together during the compilation in order to form an executable. Every component provides and uses interfaces which are the only point of access to the component. Interfaces specify a multi-function inte-

reaction channel between two components, the *provider* and the *user*. The interface specifies a set of named functions, called *commands*, supplied by the interfaces provider and a set of named functions, called *events*, to be implemented by the interfaces user.

There are two types of components in nesC: **modules** and **configurations**. Interfaces are implemented by modules while configurations are used to assemble other components together, connecting interfaces used by components to interfaces provided by others. This procedure is called wiring. Every nesC application is described by a top-level configuration that wires together the components inside.

Applications are compiled in Cygwin [3], a linux-like environment included in the TinyOS distribution. When an application is compiled, the binary code generated includes both your application and the components of the operating system that you selected when the application was written (see [5]).

3.3 Communication between motes

In a sensor network generally we need to send data from a sensor to another sensor or to a central point, called *base station* or *data collector*.

In order to read data from the network, the base station needs to be attached to a gateway that for example can be a personal computer. To make this possible, a simple way is to connect one of the motes to the USB port of a computer and have it receive data from the rest of the motes. For this, we need a program which is able to read data packets from the serial port and to forward them to a logical port in the computer, so that we can have more than one application running and reading data from the network. This program is included in the TinyOS distribution and it consists in a java tool called **SerialForwarder**. TinyOS provides also another program called **Listen**. Running the Listen Java tool makes it possible to read raw data on the screen in hexadecimal.

Once a wireless sensor network is set up two way of communication are possible: packet broadcasting and multihop routing.

3.3.1 Packet broadcasting

Packet broadcasting is a communication technique whereby data is sent from one node in a net to another by attaching address information to the data to form a packet. The packet is then broadcasted over a communication channel shared by a large number of nodes in the network. As the packet is received by these nodes the address is scanned and the packet is accepted by the proper addressee and ignored by the others.

However, packet broadcasting has a major problem: communication between two nodes is only possible if both nodes are located within each other's range. If we want all nodes to transmit data to the data collector, all of them need to be able to reach it. If a node can not reach directly the base station, (because it is too far or the radio signal strength decreases because the node is running out of batteries), the sending node will be isolated and the base station will not be able to receive data from it, even if other nodes can (see Figure 3.2). In such a solution, intermediate nodes will not forward packets received from the nodes in their range. A solution is needed in order to ensure that all the nodes can communicate to the base, this goal is reached using a multihop routing.

3.3.2 Multihop Routing

A multihop protocol ensures that a packet is forwarded from one to the destination, even if source and destination are not in each other radio range. Each node is able to receive and forward packets coming from the other nodes toward their final destination. Of course, appropriate routing protocols are necessary to discover routes between the source and the destination, or even to determine the presence or absence of a path to the destination node.

A multihop network has several issues: for example, in many applications, we might need tens or hundreds of wireless sensor nodes that may be placed either regularly or irregularly. These nodes can be exposed to highly dynamic and hostile environments, and therefore, the network must be tolerant to the failure of individual nodes. This means that if a node fails, the network must be auto-reconfigurable so that each node can communicate with the data collector.

In [14] authors describe the general properties of self organization as applied to communications, signal processing, and resource (e.g., energy)

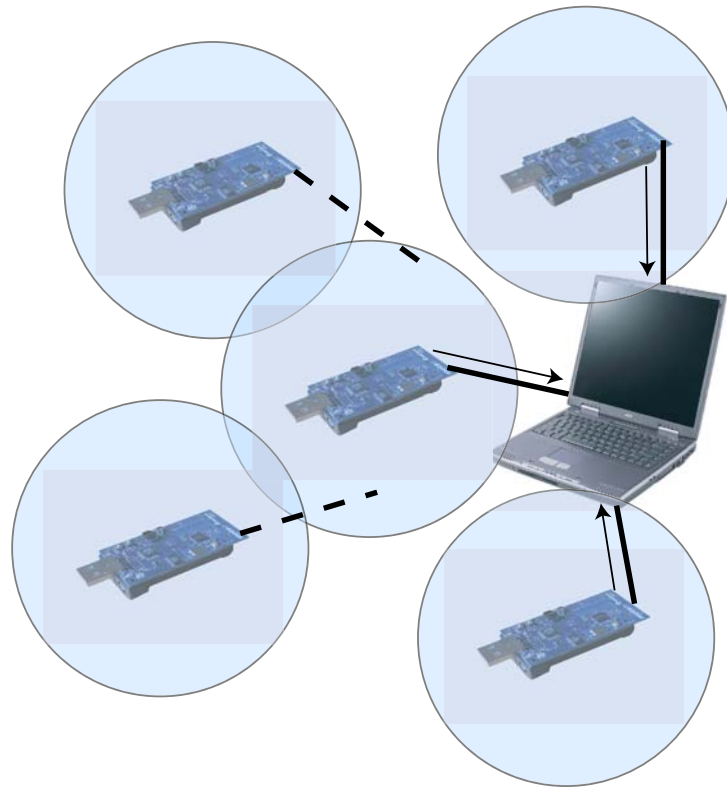


Figure 3.2: Example of packet broadcasting

management in distributed sensor networks. The paper focused on self-organization as applied to the communications network.

Others aspects of wireless sensor networks are described in [19] where they also present a set of algorithms for establishing and maintaining connectivity in wireless sensor networks.

Algorithms for wireless sensor networks must be distributed to prevent single points of failure, and self-organizing for scalable deployment [23] and multihop protocols make all this possible.

The TinyOS releases and includes library components that provide ad-hoc multi-hop routing for sensor network applications. The implementation uses a shortest-path-first algorithm with a single destination node (the data collector) and active two-way link estimation. Multihop routing in TinyOS is divided in several components so that each of them can be easily tested .

Figure 3.3 shows a situation where there is not a direct connection among some nodes and the base station, but using a multihop routing, they use the neighbor nodes so that their packets can reach the data collector.

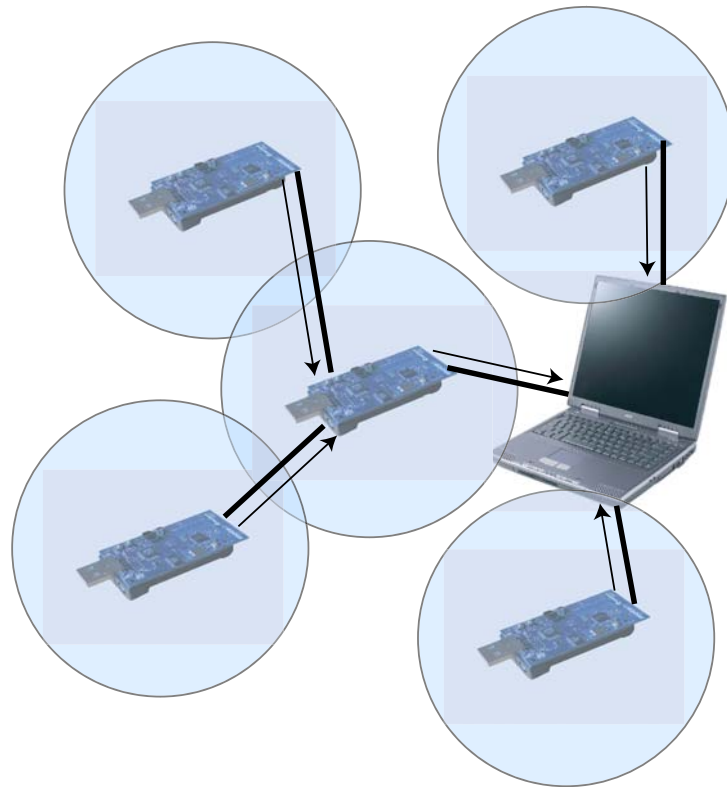


Figure 3.3: Example of multihop routing

However, there are two main drawbacks using the multihop routing:

- The delivery of data is not reliable. Since there is not an acknowledgment mechanism, corrupted data are discarded and lost.
- Multihop routing protocol introduces a huge overhead that limits the performance reducing the bandwidth available to send application data.

Chapter 4

Standard IEEE 802.15.4

The IEEE 802.15.4 standard has been created to address the need for low-rate, low-cost and low-power wireless networking. It also specifies interoperable wireless physical and medium access control layers targeted to sensor node radios [30, 17]. Our platform, the *Moteiv Tmote sky* uses the CC2420 transceiver [1] designed specifically for low-power, low-voltage RF applications in the 2.4 GHz unlicensed ISM band. It is a RF transceiver compliant with the IEEE 802.15.4 standard (for further details see [6]). Therefore in this chapter we are going to give a brief description of the standard.

4.1 Components of a Wireless Personal Area Network (WPAN)

A wireless personal area network (WPAN) consists of several components. The most basic is the device. A device can be a full-function device (FFD) or reduced-function device (RFD).

An FFD can operate in three modes serving as a PAN coordinator, a coordinator, or a device.

FFDs contain all of the features of 802.15.4 and can talk to both RFDs and FFDs.

A PAN coordinator is the primary controller of the network, and it must be a FFD. There can be only one PAN controller per network. A PAN controller is required for an 802.15.4 network.

A coordinator is a FFD that provides synchronization services by transmitting beacons.

A RFD can operate only as a device. RFDs contain a subset of the features of 802.15.4 and are intended to be high-volume, low cost devices. They can be duty-cycled to reduce power consumption. RFD devices can talk only to FFDs. This means that RFDs have no routing capability, so they must be on the perimeters of a mesh network. Conceptually, each network would have one FFD that acted as the PAN coordinator and several more FFDs that formed the mesh network. The number and position of FFDs in the network would determine the coverage of the network.

4.2 Network topology

There are three different topologies supported by IEEE 802.15.4: Star topology, Peer-to-peer topology and Cluster-tree network

4.2.1 Star topology

In the star topology, the communication is established between devices and a single central controller, called the PAN coordinator (Figure 4.1). The PAN coordinator may be main powered while the devices usually are battery powered. Applications that benefit from this topology include home automation, personal computer (PC) peripherals, toys and games. After an FFD is activated for the first time, it may establish its own network and become the PAN coordinator. Each star network chooses a PAN identifier, which is not currently used by any other network within the radio range. This allows each star network to operate independently.

4.2.2 Peer-to-peer topology

In peer-to-peer topology Figure 4.1, there is also one PAN coordinator, which acts as the root of the network. In contrast to star topology, any device can communicate with any other device as long as they are in range of one another. Applications such as industrial control and monitoring, and wireless sensor networks, would benefit from such a topology. It also allows multiple hops to route messages from any device to any other device in the network. It can provide reliability by multipath routing.

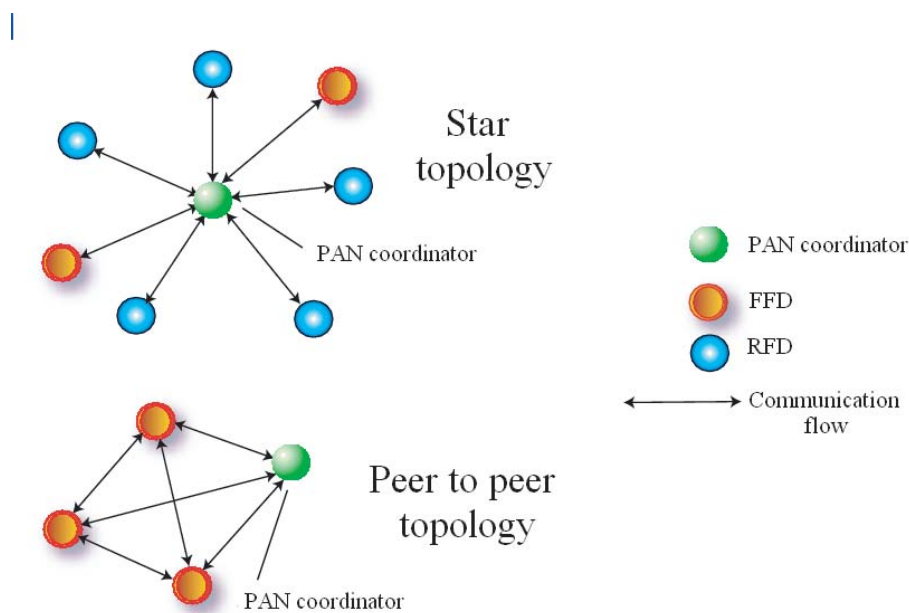


Figure 4.1: Star and Peer To Peer topology in IEEE 802.15.4 standard

4.2.3 Cluster-tree network

Cluster-tree network is a special case of a peer-to-peer network in which most devices are FFDs and an RFD may connect to a cluster-tree network as a leaf node at the end of a branch, Figure 4.2. Any of the FFDs can act as a coordinator and provides synchronization services to other devices and coordinators. Only one of these coordinators however is the PAN coordinator. The PAN coordinator forms the first cluster by establishing itself as the cluster head (CLH) with a cluster identifier (CID) of zero, choosing an unused PAN identifier, and broadcasting beacon frames to neighboring devices. A candidate device receiving a beacon frame may request to join the network at the CLH. If the PAN coordinator permits the device to join, it will add this new device as a child device in its neighbor list. The newly joined device will add the CLH as its parent in its neighbor list and it begins transmitting periodic beacons such that other candidate devices may then join the network at that device. Once application or network requirements are met, the PAN coordinator may instruct a device to become the CLH of a new cluster adjacent to the first one. The advantage of this clustered structure is the increased coverage area at the cost of increased

message latency.

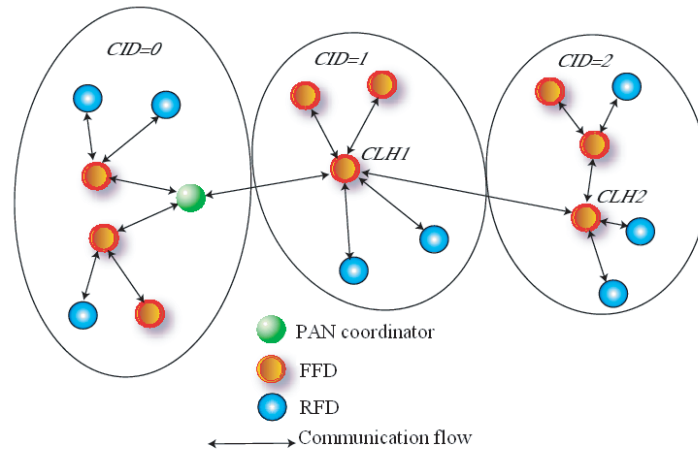


Figure 4.2: Cluster tree topology in IEEE 802.15.4 standard

4.3 IEEE 802.15.4 Physical layer

IEEE 802.15.4 defines specific RF frequencies, modulation formats, data rates, and coding techniques. It also specifies how individual packets are structured, and the interaction between two ends of a data link. IEEE 802.15.4 specifies 27 RF channels in the three frequency bands shown in Figure 4.3.

4.3.1 2400 MHz band

Regarding the 2400 MHz band, this is tremendously valuable because it allows unlicensed operation nearly anywhere in the world. There are 16 RF channels in this band. The channels start at 2400 MHz, and are spaced 5 MHz apart up to 2483.5 MHz.

The channels do not directly coincide with Wi-Fi channels. Therefore, IEEE 802.15.4 systems can coexist with Wi-Fi systems with little physical separation. The data rate is fast enough to allow very short packets, yet slow enough to make sure that the required energy per bit is compatible with the goal of very long primary battery life and good range.

PHY (MHz)	Frequency Band (MHz)	Spreading parameters		Data parameters		
		Chip rate (Kchip/s)	Modulation	Bit rate (Kb/s)	Symbol rate (Ksymbol/s)	Symbols
868/915	868-868.6	300	BPSK	20	20	Binary
	902-928	600	BPSK	40	40	Binary
2450	2400-2483.5	2000	Q-QPSK	250	62.5	16-ary orthogonal

Figure 4.3: Frequency bands and data rates

4.3.2 Lower bands

Transceivers that provide low-band operation support both a single channel (868 to 870 MHz) for European unlicensed applications, and 10 channels (902 to 928 MHz) for North American applications.

The ability of 802.15.4 to operate in the 915 MHz and 868 MHz ISM bands presents an attractive alternative to the congested 2.4 GHz spectrum. However there is a significant reduction in data rate when operating in these bands.

4.3.3 Modulation technique in IEEE 802.15.4

IEEE 802.15.4 relies on a very robust modulation technique known as Phase-Shift Keying (PSK), instead of Frequency-Shift Keying (FSK). FSK is a less efficient, but simpler to implement and is used in Bluetooth and many other applications that range from toys to cheap two-way data solutions.

The 2400 MHz band uses Offset Quadrature-PSK, while the lower bands use Binary-PSK. Both modulation modes offer extremely good low bit error rate (BER) performance at low Signal-to-Noise Ratio (SNR). Figure 4.4 compares the performance of the 802.15.4/ZigBee modulation technique to Wi-Fi, Bluetooth, and other proprietary FSK modulation formats.

In all cases, both forms of PSK are anywhere from 7 to 18 dB better,

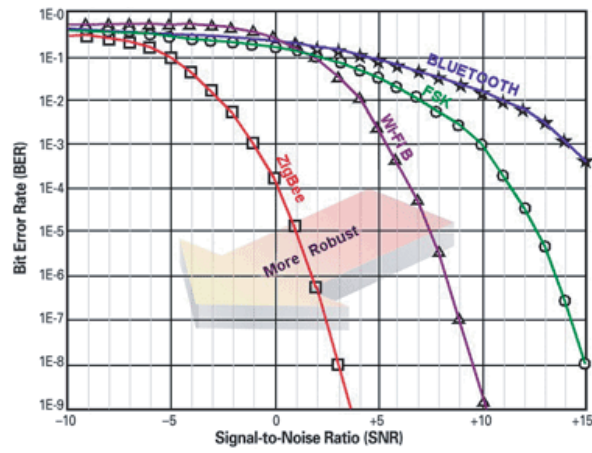


Figure 4.4: BER versus SNR. Picture taken from <http://www.embedded-computing.com>

which directly translates to a range that increases from 2 to 8 times the distance using the same energy per bit, or an exponential increase in reliability at any given range.

4.4 IEEE 802.15.4 Medium Access Control layer

The MAC protocol in IEEE 802.15.4 can operate in both beacon and non beacon mode.

One of the problems in many sensor network applications is the problem of ensuring the desired quality of service, expressed as the mean data rate obtained from a specific spatial area, while simultaneously maximizing the lifetime of the network.

This tradeoff is mostly given by the **beacon order**(or duty cycle).

The features of the MAC layer are beacon management, channel access, GTS management, data transfer, acknowledged frame delivery and frame validation. In the beaconless mode, the protocol is a CSMA-CA protocol. We will discuss this technique in Paragraph 4.4.4.

In the beacon mode, the IEEE 802.15.4 uses a superframe structure.

4.4.1 Importance of the duty cycle in a WSN

Among the most important requirements for sensor networks is the maximization of their lifetime due to high costs (and sometimes even infeasibility) of maintenance activities.

Sensor lifetime can be extended by adjusting the frequency and ratio of active and inactive periods of sensor nodes [28].

This approach is supported by the 802.15.4 standard in its beacon enabled mode with slotted CSMA-CA, where the interval between the two beacons is divided into active and inactive parts, and the sensors can switch to low-power mode during the inactive period. However, many sensor applications (such as surveillance, health care, and structural health monitoring) require continuous monitoring of relevant variables and events, and letting the network sleep for some time is simply out of the question.

In such cases, duty cycle management can be applied at the level of individual sensor nodes, provided the number of sensors covering a given physical area is larger than the minimum number based on the required data rate. The desired data rate received at the sink can then be achieved by adjusting the number of sensors that are active at any given time.

When redundant sensors are used, their duty cycle can be reduced to increase the lifetime of the network while maintaining the desired QoS. In this manner, the costly maintenance and human intervention can be reduced, which may well offset the increased initial cost of deployment. As

an added bonus, the failure of individual sensors will not affect network performance since the duty cycle of the remaining ones can be increased to compensate for loss.

4.4.2 Duty cycle organization

In a beacon-enabled network the PAN has some important parameters which determine the time between two successive beacons (*beacon interval BI*), the duration of the active phase (*superframe duration SD*) and the conditions for the guaranteed time slots (**GTS**). These parameters are transmitted by the PAN to all the devices in the network so that each one knows the duty cycle and when it is allowed to send messages. the structure of the duty cycle is shown in Figure 4.5.

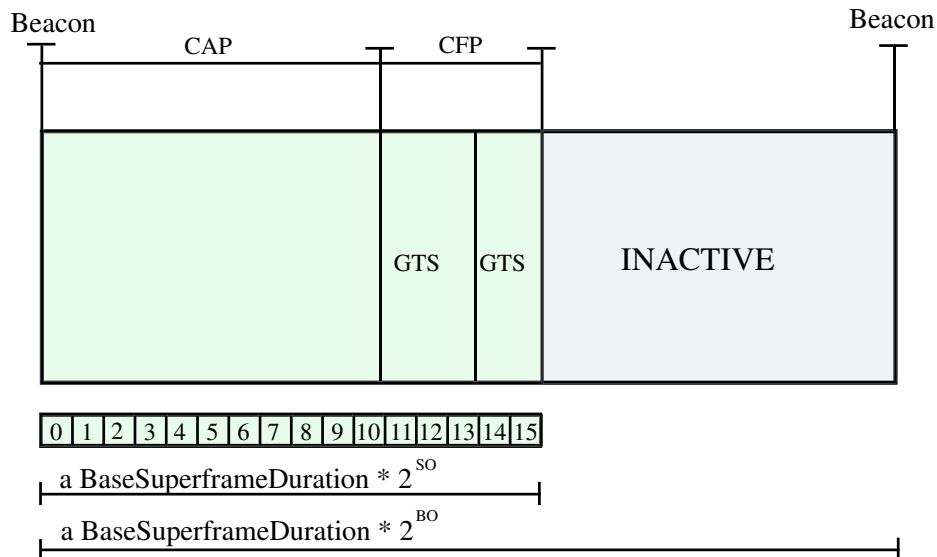


Figure 4.5: Duty cycle with beacon enable mode in IEEE 802.15.4 standard

4.4.3 Superframe structure

IEEE 802.15.4 MAC layer allows the optional use of a superframe structure. The format of the superframe is defined by the PAN and is bounded

by network beacons. The beacon frame is sent in the first slot of each superframe. If a coordinator does not want to use the superframe structure, it may turn off the beacon transmissions.

The beacons are used to synchronize the attached devices, to identify the PAN and to describe the structure of superframe and occur in an interval that can range from 15ms to 245s.

The superframe contains one active and one non-active portion. During the inactive portion, devices shall not interact with the PAN and may enter a low-power mode. The active portion of each superframe is divided into 16 equally sized slots and consists in three parts:

- Beacon
- Contention access period (**CAP**)
- contention free period (**CFP**)

Any device wishing to communicate during the CAP shall compete with other devices using a slotted CSMA-CA mechanism. On the other hand, the CFP contains guaranteed time slots (**GTS**). The GTS always appear at the end of the active superframe part. The PAN coordinator may allocate up to seven of these GTS and each one can occupy more than one slot period. The duration of different portions of the superframe is described by the values of **macBeaconOrder** and **macSuperFrameOrder**. **MacBeaconOrder** describes the interval at which the coordinator shall transmit its beacon frames. The beacon interval, **BI**, is related to the **macBeaconOrder**, **BO**, as follows:

$$BI = aBaseSuperFrameDuration \cdot 2^{BO}, \quad 0 \leq BO \leq 14. \quad (4.1)$$

The superframe is ignored if $BO = 15$.

The value of **macSuperFrameOrder** describes the length of the active portion of the superframe. The superframe duration, **SD**, is related to **macSuperFrameOrder**, **SO**, as follows:

$$SD = aBaseSuperFrameDuration \cdot 2^{SO}, \quad 0 \leq SO \leq 14 \quad (4.2)$$

If $SO = 15$, the superframe should not remain active after the beacon.

The active portion of each superframe is divided into a **aNumSuperFrameSlots** equally spaced slots of duration $aBaseSlotDuration \cdot 2^{SO}$ and

is composed of three parts: a beacon, a CAP and CFP. The beacon is transmitted at the start of slot 0 without the use of CSMA.

The CAP starts immediately after the beacon. The CAP shall be at least `aMinCAPLength` symbols unless additional space is needed to temporarily accommodate the increase in the beacon frame length to perform GTS maintenance.

All frames except acknowledgement frames or any data frame that immediately follows the acknowledgement of a data request command that are transmitted in the CAP shall use slotted CSMA-CA to access the channel. A transmission in the CAP shall be complete one **IFS** period before the end of the CAP (where the IFS period is the time necessary to process the received packet to the physical layer). If this is not possible, it defers its transmission until the CAP of the following superframe. The CFP, if present, shall start on a slot boundary immediately following the CAP and extends to the end of the active portion of the superframe. The length of the CFP is determined by the total length of all of the combined GTS. No transmissions within the CFP shall use a CSMA-CA mechanism.

A device transmitting in the CFP shall ensure that its transmissions are complete one IFS period before the end of its GTS. Transmitted frames shall be followed by an IFS period. The length of IFS depends on the size of the frame that has just been transmitted.

The PAN that does not wish to use the superframe in a nonbeacon-enabled shall set both `macBeaconOrder` and `macSuperFrameOrder` to 15. In this kind of network, a coordinator shall not transmit any beacons, all transmissions except the acknowledgement frame shall use unslotted CSMA-CA to access channel, GTS shall not be permitted.

4.4.4 CSMA-CA algorithm

According to the slotted CSMA/CA algorithm, a node must sense the channel free at least twice before being able to transmit.

The first sense must be delayed by a random delay chosen between 0 and $2^{BE} - 1$, where BE is the backoff exponent. This randomness serves to reduce the probability of collision when two nodes simultaneously sense the channel, assess it free and decide to transmit at the same time.

When the channel is sensed busy, transmission may not occur and the next channel sense is scheduled after a new random delay computed with

an incremented backoff exponent. If the latter has been incremented twice and the channel is not sensed to be free, a transmission failure is notified and the procedure is aborted.

When a packet collides or is corrupted, it can be retransmitted after a new contention procedure. The contention procedure starts immediately after the end of the beacon transmission. All channel senses or transmissions must be aligned with the CSMA slot boundaries.

This contention procedure introduces a significant overhead in energy consumption since devices wishing to transmit have to wait longer before entering in low-powered mode. Therefore the 802.15.4 standard supports the so called Battery Life Extension mode, which shortens the CAP reducing the active part of the superframe. However, in dense network conditions, this mode would result in an excessive collision rate.

4.4.5 Beacon generation

A FFD (Full Function Device) may either operate in a beaconless mode or may begin beacon transmissions either as the PAN coordinator or as a device on a previously established PAN. An FFD that is not the PAN coordinator shall begin transmitting beacon frames only when it has successfully associated with a PAN. The time of the transmission of the most recent beacon shall be recorded and computed so that its value is taken at the same symbol boundary in each beacon frame, the location of which is implementation specific.

4.4.6 Synchronization

For PAN supporting beacons, synchronization is performed by receiving and decoding beacon frames. For PAN that does not support beacons, the synchronization is performed by polling the coordinator for data. In a beacon enabled network, devices shall be permitted to acquire synchronization only with beacons containing the PAN identifier. There are two methods of synchronization:

- Tracking
- Non tracking

In tracking mode the device shall attempt to acquire the beacon and keep track of it by regular and timely activation of its receiver. It shall enable its receiver at a time prior to the next expected beacon frame transmission, i.e. just before the known start of the next superframe. The tracking mode leads to a consume of energy since the receiver is periodically activated.

In the non tracking mode, the device shall attempt to acquire the beacon only once. The device does not spend energy on periodical beacon reception, but in the other hand, it has to enable its receiver immediately to search for a beacon when it need to communicate with the coordinator, since it does not know where the beacon appears. This can lead to an high energy consumption due to the long idle period.

There are tradeoff between tracking and non tracking depending on the duty cycle and data rate. When duty cycle is low the number of beacon frames is reduced, so that the beacon tracking consumes less energy to receive the beacon frames. As drawback, non tracking devices, need to wait for a longer time statistically before they can receive a beacon frame, which increases energy consumption.

With higher data rate, there are statistically more packets waiting to be transmitted in a beacon interval. At the same time there is also a need to acquire the synchronization for non-tracking devices, and thus the energy consumption would also increase.

In [24] authors show that low duty cycle enables significant energy saving but at the cost of significantly higher latency and lower bandwidth.

Considering the IEEE 802.15.4 network in beacon enabled mode, [26] have analyzed the performance under two duty cycle management algorithms. Both algorithms are fully distributed, i.e., the nodes autonomously control their sleep period according to local information only. In the first algorithm, the sleep period is a geometrically distributed random variable with a given mean value; in the second, the sleep period is inversely proportional to the most recent duration of the packet service time. They show that the network reliability is function of the MAC backoff parameters and of the duty cycle management discipline.

An evaluation on the performance of IEEE 802.15.4 standard is given in [11]. In this paper the authors have studied how this standard can be used to support communication in dense, data-gathering networks.

4.4.7 Data transfer

For both beacon and non-beacon networks data transfer can happen in three different ways:

- From a device to a coordinator
- From a coordinator to a device
- from one peer to another in a peer-to-peer multi-hop network

The mechanisms for data transfer depend on whether the network supports the transmission of beacons. A beacon-enabled network is used for supporting low-latency devices, such as PC peripherals. If the network does not need to support such devices, it can elect not to use the beacon for normal transfers.

Direct data transmission: This data transfer transaction is the mechanism to transfer data from a device to a coordinator.

In a beacon-enabled network, when a device wishes to transfer data to a coordinator, it first listens for the network beacon, as shown in Figure 4.6(a). When the beacon is found, the device synchronizes to the super-frame structure.

At the appropriate point, the device transmits its data frame, using slotted CSMA-CA, to the coordinator.

The coordinator acknowledges the successful reception of the data by transmitting an acknowledgment frame.

On the other hand, in a nonbeacon-enabled network, when a device wishes to transfer data, it simply transmits its data frame, using unslotted CSMA-CA, to the coordinator.

The coordinator acknowledges the successful reception of the data by transmitting an acknowledgment frame, as shown in Figure 4.6(b).

Indirect data transmission: This data transfer transaction is the mechanism for transferring data from a coordinator to a device.

In a beacon-enabled network, when the coordinator wishes to transfer data to a device, it indicates in the network beacon that the data message is pending. The device periodically listens to the network beacon and, if a message is pending, transmits a MAC command requesting the data, using slotted CSMA-CA.

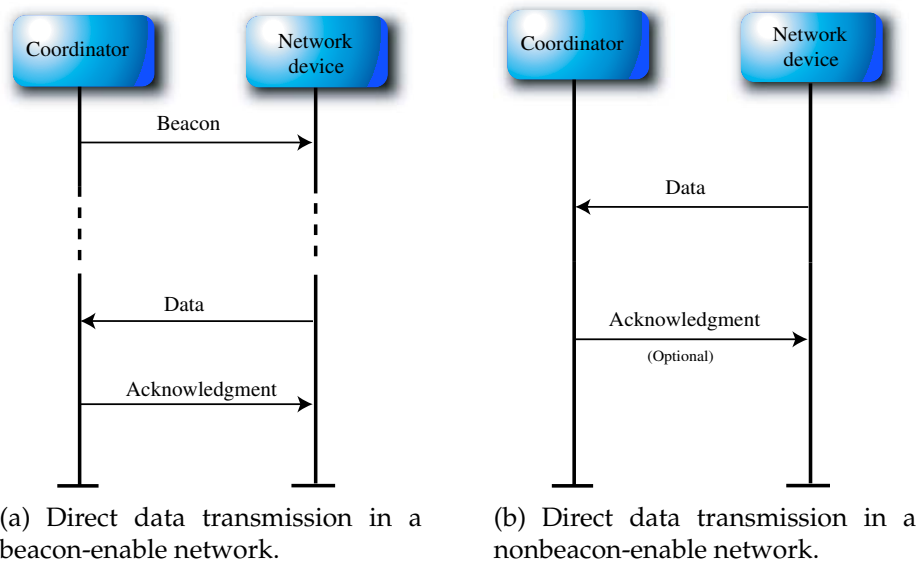


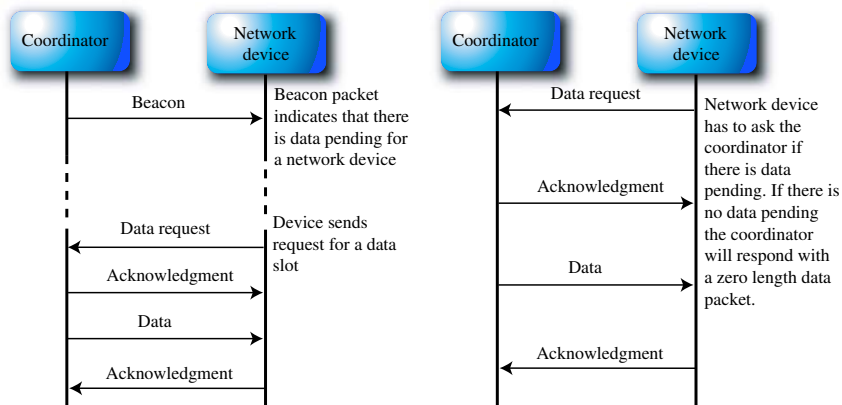
Figure 4.6: Direct data transmission in beacon-enable and non-beacon enable

The coordinator acknowledges the successful reception of the data request by transmitting an acknowledgment frame. The pending data frame is then sent using slotted CSMA-CA.

The device acknowledges the successful reception of the data by transmitting an acknowledgment frame. Upon receiving the acknowledgment, the message is removed from the list of pending messages in the beacon. This sequence is summarized in Figure 4.7(a). On the other hand, in a nonbeacon-enabled network, when a coordinator wishes to transfer data to a device, it stores the data for the appropriate device to make contact and request the data. A device may make contact by transmitting a MAC command requesting the data, using unslotted CSMA-CA, to its coordinator at an application-defined polling rate, as shown in Figure 4.7(b).

The coordinator acknowledges the successful reception of the data request by transmitting an acknowledgment frame. If data are pending, the coordinator transmits the data frame, using unslotted CSMA-CA, to the device. If data are not pending, the coordinator transmits a data frame with a zero-length payload to indicate that no data were pending. The device acknowledges the successful reception of the data by transmitting an

acknowledgment frame.



(a) Indirect data transmission in a beacon-enabled network. (b) Indirect data transmission in a nonbeacon-enabled network.

Figure 4.7: Indirect data transmission in beacon-enabled and non-beacon enabled

4.4.8 Frame format

The general MAC frame format is given in Figure 4.8

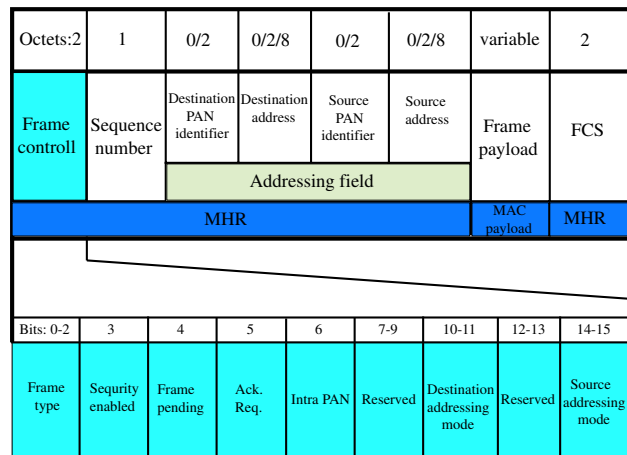
There are four frame types for IEEE 802.15.4 transmission:

- Data frame (Fig. 4.9(a))
- Acknowledgment frame (Fig. 4.9(b))
- MAC command frame (Fig. 4.9(c))
- Beacon frame (Fig. 4.9(d))

Frame commonality

All frames have the following common components:

- **Preamble:** All of the frames begin with a 40-bit preamble that helps the receiving station to pick the transmission out of noisy environments.



Frame controll field

Figure 4.8: General MAC frame in the 802.15.4

- **Frame Length field:** Tells the receiving station exactly how long the frame is.
- **Sequence Number:** An 8-bit value that is incremented each time a device transmits a new, unique frame.
- **Frame Check Sequence (FCS):** Each frame ends with a 16-bit mathematical sequence that allows the receiving station to validate that the packet was received without error.

The **data packet** is the major one that affects the data throughput of the network. The format of the 802.15.4 **data frame** is shown in Figure 4.9(a).

The MAC frame, i.e. the MPDU, is composed of an MAC header (MHR), MAC service data unit (MSDU), and MAC footer (MFR). The first field of the MAC header is the frame control field (FCF). It indicates the type of MAC frame being transmitted, specifies the format of the address field, and controls the acknowledgment.

In short, the frame control field specifies how the rest of the frame looks and what it contains.

A data frame may contain both source and destination information with the size of the address field between 4 and 20 bytes. The payload field is variable in length. However, the maximum MAC data payload,

Octets:2	1	4 or 10	2	variable	variable	variable	2
Frame control	Beacon Sequence number	Source address information	Superframe specification	GTS fields	Pending address field	Beacon payload	FCS
MAC header			MAC payload				MHR

Bits: 0-3	4-7	8-11	12	13	14	15
Beacon order	Superframe order	Final CAP slot	Battery life extension	Reserved	PAN coordinator	Association permit

(a) Schematic view of the data frame.

Octets:2	1	2
Frame control	Data Sequence number	FCS
MAC header		MHR

(b) Schematic view of the acknowledgment frame.

Octets:2	1	4 or 20	1	variable	2
Frame control	Data Sequence number	Address information	Command type	Command payload	FCS
MAC header			MAC payload		MHR

(c) Schematic view of the MAC command frame.

Octets:2	1	4 or 20	variable	2
Frame control	Data Sequence number	Address information	Data payload	FCS
MAC header		MAC payload		MHR

(d) Schematic view of the beacon frame.

Figure 4.9: Frame formats.

$aMaxMACFrameSize$, is equal to $aMaxPHYPacketSize$ (127 bytes) $aMaxFrameOverhead$ (25 bytes) = 102 bytes. The MPDU is passed to the PHY as the PHY data frame payload, *i.e.*, PSDU. The PSDU is prefixed with a synchronization header (SHR) and a PHY header (PHR), together with PSDU to form the PHY data packet, *i.e.*, PPDU.

Acknowledgment frame

An IEEE 802.15.4 system uses the Acknowledgement frame (Figure 4.9(b)) to tell the transmitting station that the data arrived successfully. This takes place as shown in the following data exchange between stations A and B:

- A checks the RF channel to ensure that another station is not transmitting.
- A transmits a data frame addressed to Station B.
- B receives the frame, and uses the FCS to make sure that the packet was received without error.

- At 192 microseconds after the end of A's transmission, B responds with the Acknowledgement packet which is addressed to A, and contains the Frame Number specified in A's original frame. This allows A to know that B received the data without error.
- If this Acknowledgement is not received successfully, then the process begins anew with A transmitting the Data Frame a second time.

Even with multiple retransmissions, the entire process takes well under 10 milliseconds, making for a timely, robust and efficient data exchange process.

4.4.9 Robust communications

IEEE 802.15.4 is a communications system that contains robust mechanisms that start at the RF channel itself. They include the careful choice of radio frequencies, the use of the most robust and simple modulation techniques, and the use of communication frame features to make sure that the message can get through. Finally, the acknowledgement frame closes the loop, which allows the sender to know that the message got through.

Chapter 5

TinyOS software for measuring radio performance

A key objective of our work is to understand the loss performance of the Telos platform. However, as far as we know, there is no software in the TinyOS distribution that estimates the radio performance, hence we have written our own software for this purpose. This chapter summarizes our effort.

5.1 Single hop

As regards the single hop, we have considered one of the simplest applications: The *CountRadio* application.

5.1.1 CountRadio

CountRadio is a simple program that maintains an internal counter and communicates its value via radio regardless of the node address. The value of the counter increases by one in each packet. It does not implement any multihop mechanism, but it simply sends its packet to the broadcast address.

The first step has been to compile the application as it is written in the *readme.CountRadio* file in the TinyOS package and run it on one Mote. Then we connected another mote (as data collector) to the pc and ran the **TOSBase** application on it. The TOSBase application acts as a gateway,

relaying the motes' packets to the PC and viceversa, *i.e* it receives and displays the data by listening to the broadcast address and forwards the packets to the computer.

In order to read and display the data on the screen one just has to open a cygwin [3] window, run the SerialForwarder Java tool and then the Listen Java tool. Now it is possible to see raw data appear in the screen in hexadecimal. This is not the best way to read and understand information, but at least we can check that the packets are coming and verify if they are as expected. The general packet format of the CountRadio application is shown in Figure 5.1.

Bits: 1	2	1	4	1	1	2	2
Payload length	Frame controll	Counter	Broadcast address	Message type	ID group	Counter	Source address

Figure 5.1: General structure of a CountRadio packet

5.1.2 Meaning of the packet stream

Having the knowledge of the structure of the AM packet, allows to get a complete information from the data displayed on the screen.

In [34] the authors help us to understand the general structure of an AM packet even if the final Telos packet structure is defined in AM.h file and it is a little bit different from the general AM message format in TinyOS (some fields are the same, but in a different position).

Note that the data obtained running the Listen Java tool are in hexadecimal in **little endian** format which means that the most significative byte appears on the right of the string.

Running Listen Java tool once the SerialForwarder is already running, we might obtain data strings like in figure 5.2:

```

/cygdrive/c/program-files/ucb/cygwin/opt/tinyos-1.x
04 01 08 F7 FF FF FF FF 04 7D 78 CA 01 00
04 01 08 F8 FF FF FF FF 04 7D 7B CA 01 00
04 01 08 F9 FF FF FF FF 04 7D 7F CA 01 00
04 01 08 FA FF FF FF FF 04 7D 83 CA 01 00
04 01 08 FB FF FF FF FF 04 7D 85 CA 01 00
04 01 08 FC FF FF FF FF 04 7D 89 CA 01 00
04 01 08 FD FF FF FF FF 04 7D 8B CA 01 00
04 01 08 FE FF FF FF FF 04 7D 8F CA 01 00
04 01 08 FF FF FF FF FF 04 7D 91 CA 01 00
04 01 08 00 FF FF FF FF 04 7D 95 CA 01 00
04 01 08 01 FF FF FF FF 04 7D 98 CA 01 00
04 01 08 02 FF FF FF FF 04 7D 9C CA 01 00
04 01 08 03 FF FF FF FF 04 7D 9E CA 01 00
04 01 08 04 FF FF FF FF 04 7D A2 CA 01 00
04 01 08 05 FF FF FF FF 04 7D A5 CA 01 00
04 01 08 06 FF FF FF FF 04 7D A8 CA 01 00
04 01 08 07 FF FF FF FF 04 7D AA CA 01 00
04 01 08 08 FF FF FF FF 04 7D AD CA 01 00
04 01 08 09 FF FF FF FF 04 7D AF CA 01 00
04 01 08 0A FF FF FF FF 04 7D B2 CA 01 00
04 01 08 0B FF FF FF FF 04 7D B5 CA 01 00
04 01 08 0C FF FF FF FF 04 7D B8 CA 01 00
04 01 08 0D FF FF FF FF 04 7D BA CA 01 00
04 01 08 0E FF FF FF FF 04 7D BD CA 01 00

```

Figure 5.2: Strings of data

- The first byte (0x04) represents the payload length (4 byte in this case).
- The second and third byte are Telos specific fields for frame control.
- The fourth byte is a counter, it increases by one at each step (considering it in all the following strings you can also see if you have some missed packet).
- The fifth and sixth fields have two bytes each (0xFFFF and 0xFFFF), they represent a special destination identifier and the destination address. Since CountRadio does not implement a routing protocol but packets are simply broadcasted, the two field are set to the broadcast address which is 0xFFFF by default.

- The seventh field represents the message type (0x04 for CountRadio packets).
- The next field is the ID group (0x7D). Motes can communicate to each others only if they belong to the same group.
- The next two fields are both a counter.
- The following two bytes (0x0100) represent the address of the mote that has originated the packet.

At the beginning of the experiment we have been obtaining a very low sampling rate (around 4 packets per second). Therefore, we were interesting in obtain better performances in terms of sampling rate.

Looking at the Cygwin window, we can see that when we compile the CountRadio application, two modules are invoked: CountSendC.nc and CountSendM.nc. Particularly changing the code to the CountSendM.nc permits us to change some default values:

```
command result_t StdControl.start()
{
call Timer.start( TIMER_REPEAT, XXX );
return SUCCESS;
}
```

Increasing or decreasing XXX we can set the sampling rate as we like, up to 1 packet every 5ms, beyond that the microcontroller is saturated.

5.2 Multihop

Since Telos motes ere mainly intended for monitoring applications, nodes need to communicate with each other. Particularly, they should be able to communicate with the base station, even if it is not within their radio range. To make this possible, we need multihop functionality. One application that uses a multihop protocol is the **Aegis** application [27].

5.2.1 Aegis

Aegis a nesC application developed over TinyOS for Telos Tmote sky. The Aegis application is complemented by the Aegis Java Tool that provides a graphical interface that allows to visualize the logic tree of the network.

A node running Aegis reads data (temperature, Humidity, Total Solar Radiation and Photosynthetically Active Radiation) from its onboard sensors and sends them inside a data packet to the base station.

In Aegis, nodes build a routing tree where the node with address "0" acts like a base station. Aegis can be used for both single hop and multihop routing. All the multihop routing is managed by the TinyOS component LQIMultiHopRouter, which offers the interface needed to send messages through the routing tree up to the base station.

5.2.2 Aegis packet format

The Aegis application uses two different kind of packets: one containing the sensor readings sent from one node to the base station (**Aegis message**) and one containing a command sent from the base station to the nodes (**SimpleCmdMsg**). As the structure of the SimpleCmdMsg is described in the corresponding SimpleCmdMsg.h file in the Aegis package, we are going to describe only the Aegis message.

In Figure 5.3 is shown the general header structure for an Aegis message. (More details are given in [27], § 5.1.4)

- The Aegis message starts with a header which consists on the message type (sensor reading in our case).
- Then a multihop header, containing among other fields the source address (address of the node that has generated the packet), the forwarding address, (last node that has forwarded the packet) and a hop count (number of times the packet has been forwarded until now), is added (10 byte).
- The last header is the Active Message (AM) header. It contains the destination address, the payload length and the AM message type (17 for Aegis message).
- The Aegis tail contains the parent node address and a 32 bit sequence number.

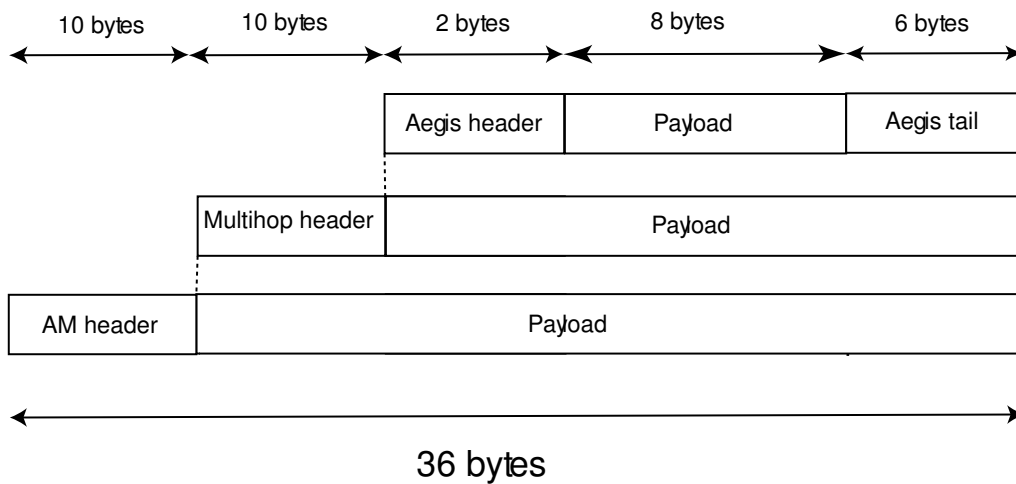


Figure 5.3: Aegis message headers

5.2.3 Meaning of the Aegis packet stream

Using Aegis, running SerialForwarder Java tool and Listen Java tool, we should see packets in hexadecimal sliding on the screen like in Figure 5.4:

We can now analyze all the packet fields:

- The first 10 bytes belong to the AM packet header:
 - 0x1A defines the payload length which is 26 bytes for all the packets.
 - the next three fields are the Telos specific fields for frame control.
 - 0xFF and 0xFF are the destination PAN identifier (also specific for Telos).
 - 0x7E and 0x00 is the destination address (the serial port (UART) in this case).
 - 0x11 is the AM message type (17 in this case).
 - 0x7D is the group ID.
- The next 10 bytes belong to the Multihop packet header:

```

/cygdrive/c/program-files/UCB/cygwin/opt/tinyos-1.x
david@xjob-rt /cygdrive/c/program-files/UCB/cygwin/opt/tinyos-1.x
$ java net.tinyos.tools.Listen
1A 00 00 00 00 00 7E 00 11 7D 00 00 00 00 00 00 8F 00 00 00 00 00 5C 1A 7A 04 01
00 01 00 7E 00 ED 08 00 00
1A 21 08 33 FF FF 7E 00 11 7D 00 00 01 00 08 01 08 01 01 00 00 00 94 19 F2 04 01
00 01 00 00 00 F1 00 00 00
1A 21 08 0A FF FF 7E 00 11 7D 00 00 02 00 05 01 05 01 01 00 00 00 99 19 F8 04 01
00 01 00 00 00 EB 00 00 00
1A 00 00 00 00 00 7E 00 11 7D 00 00 00 00 00 00 8F 00 00 00 00 00 5C 1A 7A 04 02
00 01 00 7E 00 EE 08 00 00
1A 21 08 34 FF FF 7E 00 11 7D 00 00 01 00 09 01 09 01 01 00 00 00 93 19 F0 04 01
00 01 00 00 00 F2 00 00 00
1A 21 08 0B FF FF 7E 00 11 7D 00 00 02 00 06 01 06 01 01 00 00 00 99 19 F7 04 01
00 01 00 00 00 EC 00 00 00
1A 00 00 00 00 00 7E 00 11 7D 00 00 00 00 00 00 8F 00 00 00 00 00 5C 1A 7A 04 02
00 02 00 7E 00 EF 08 00 00
$

```

Figure 5.4: Aegis packets in hexadecimal format

- 0x00 is the forwarding node; in this case is the base station towards the UART port.
 - 0x01 0x00 is the originating node which is node with address 1 in this case.
 - 0x08 0x01 is a sequence number, it increases by one in every packet.
 - 0x08 0x01 is a sequence number set by the origin (in this case it has not been modify)
 - 0x01 0x00 is the hop count: in this case it indicates that node with address 1 is one hop away from the data collector.
- The next 2 bytes 0x00 and 0x00 belong to the Aegis packet header and represent the Aegis packet type. Since we are considering only the type “sensor reading” the only value used is “0”.
 - The following 8 bytes are the sensor readings:
 - **Temperature** (0x94 0x19)
 - **Humidity** (0xF2 0x04)
 - **TSR** (Total Solar Radiation 0x01 0x00)
 - **PAR** (Photosynthetically Active Radiation 0x01 0x00)

- The last 6 bytes belong to the Aegis tail:
 - 0x00 0x00 is the parent address of the originating node. In this case the parent address of the mote with number “1” is the base station which has address “0”.
 - The last bytes 0xF1 0x00 0x00 0x00 is a four-byte sequence number for Aegis.

5.2.4 Aegis performances

Regardless the kind of network structure, (star topology or linear topology) using Aegis we have been obtaining very bad performances. In the basic configuration the sampling rate is one packets every two seconds and as we are trying to increase it up to one packet per second, using the control panel provided by java tool, the network fails drastically.

Quoting from the TinyOS documentation on multihop at http://www.tinyos.net/tinyos-1.x/doc/multihop/multihop_routing.html:

One limitation of multi-hop, however, is the aggregate data rate. Applications should maintain average message frequency at or below one message every two seconds. Higher rates can lead to congestion and or overflow of the communication queue.

Nonetheless we are interested in finding a way to solve this problem.

5.2.5 Our work on Aegis

The Aegis application is compiled using several modules and components. Among them we focus on the “AegisM.nc” module.

That is an application based on SurgeM.nc, OscilloscopeM.nc, SimpleCmdM.nc and BcastM.nc (provided in the TinyOS package) where we can set all the parameters of the network as well as the sampling rate.

```
Execute the command
switch (cmd->type) {
  case AEGIS_TYPE_LED_ON:
    call Leds.greenOn();
    break;
```

```

case AEGIS_TYPE_LED_OFF:
    call Leds.greenOff();
    break;
case AEGIS_TYPE_SETRATE:
    ...
    call Timer.start(TIMER_REPEAT, timer_rate);
    break;
case AEGIS_TYPE_SLEEP:
    ...
    break;
case AEGIS_TYPE_WAKEUP:
    ...
    call Timer.start(TIMER_REPEAT, timer_rate);
    ...
}
break;
case AEGIS_TYPE_FOCUS:
    ...
    call Timer.start(TIMER_REPEAT, FOCUS_TIMER_RATE);
} else {
    ...
    call Timer.start(TIMER_REPEAT, FOCUS_NOTME_TIMER_RATE);
}
break;
case AEGIS_TYPE_UNFOCUS:
    ...
    call Timer.start(TIMER_REPEAT, timer_rate);
    break;
default:
    //call Leds.yellowToggle();
}
pending =0;
signal ProcessCmd.done(msg, SUCCESS);
}

...
...
...

```

```

event result_t Timer.fired() {
    if (initTimer) {
        initTimer = FALSE;
        return call Timer.start(TIMER_REPEAT, timer_rate);
    }
    dbg(DBG_USR1, "AegisM: Timer fired\n");
    timer_ticks++;
    if (timer_ticks % TIMER_GETADC_COUNT == 0) {
        call TSR.getData();
    }
    return SUCCESS;
}

```

Acting on this portion of code we can set the sampling rate, the transmission rate as well as the parameters used in the Aegis Java Tool to control the network.

Limitation of the sampling rate

There is nevertheless a limitation on the sampling rate; Telos motes are equipped with Humidity and Temperature sensors provided by SENSIRION. After the request for a measurement, the controller has to wait for the measurement to be ready. This takes approximatively 210ms for the temperature sensor plus 55ms for the humidity sensor [33]. These values are retained to the default measurement resolutions of 14bit (for temperature sensor) and 12bit (for humidity sensor). These values can be reduced to 12 and 8 bit respectively to obtain high speed or extremely low power applications. Using the default values we experience a maximum sampling rate of three packets per second.

In order to test the radio performance of the Telos, we need to have an application that allows us to obtain an higher rate. One of the applications provided by the TinyOS package that uses a multihop protocol is the **Surge** application. Even if the default rate of this application is very poor (one packet every two second), we will show later what one needs in order to obtain an high rate.

5.2.6 The Surge application

Surge is a simple application that performs periodic sensor sampling of the internal voltage and uses ad-hoc multi-hop routing over the wireless network to deliver samples to the base station.

Surge motes organize themselves into a spanning tree rooted at the base station. Each mote maintains the address of its parent and its depth in the tree, advertising its depth in each radio message (either sensor sample or forwarded message) that it transmits. A node selects an initial parent by listening to messages and choosing the node with the smallest depth. In order to generate the spanning tree, the base station periodically broadcasts beacon messages with depth 0. Nodes estimate parent link quality and if it falls below some threshold, they select a new parent from their neighbor set based on link-quality estimates and depth.

When a node receives a message from another node, it forwards the message to its parent. Sensor samples are collected at the base station where they can be analyzed or visualized.

Surge programmed nodes belonging to the same class, set by the command `“java net.tinyos.surge.MainClass <group>”`, are processed via a GenericBase station. The java tool snoops the multihop headers to provide a graphical view of the logical network topology. It also permits variation of the sample rates and sending pre-defined commands to the surge nodes.

5.2.7 Performance of the Surge application

By default the Surge application gives a very poor performances in terms of sampling rate (approximately one packet per two seconds). As in the CountRadio application, we need to increase the sampling rate in order to observe the limitation of the radio channel.

Looking once again to the nesC code, we can see that when Surge is compiled the `Surge.nc` configuration and the `SurgeM.nc` module are invoked. Particulary the last one permits us to change the sampling rate.

.....
.....
.....

```

event result_t Timer.fired() {
    if (initTimer) {
        initTimer = FALSE;
        return call Timer.start(TIMER_REPEAT, timer_rate);
    }
    dbg(DBG_USR1, "SurgeM: Timer fired\n");
    timer_ticks++;
    if (timer_ticks % TIMER_GETADC_COUNT == 0) {
        call ADC.getData();
    }
    return SUCCESS;
}
.....
.....
.....

```

Once again we can change the sampling rate acting on this part of the nesC code of the module SurgeM.nc.

By default Surge application uses the DemoSensorC component so, as said above, nodes send internal voltage measurement. We can modify the application to send data from any other sensor, but unfortunately the graphical interface for Surge provided in the TinyOS package does not work for Telos.

Chapter 6

Experiments in a real Wireless Sensor Network

In this chapter we focus on the performances of a real wireless sensor network, where the motes are the Telos Tmote sky.

6.1 Loss rate with AEGIS

We are interested in seeing what happens to the loss rate running Aegis with the maximum sample rate of three packets per second.

We focus on a linear topology where 4 sensors, of which one acting as the base station, are arranged as shown in Figure 6.1

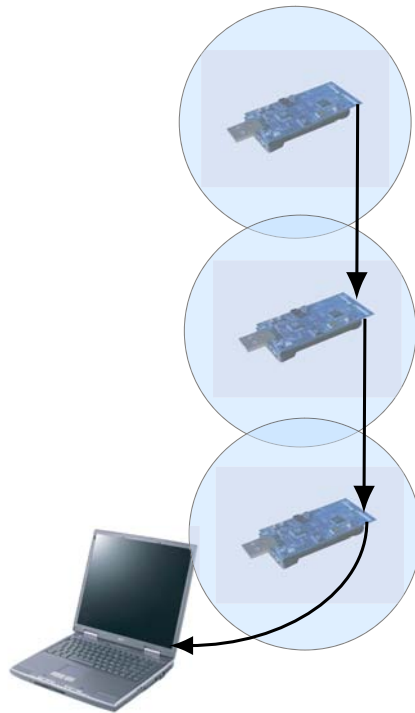


Figure 6.1: Linear topology with 4 sensors

Since the motes' behavior is influenced by a lot of factors, such as the orientation and the height above the ground, the radio activity of the department as well as the presence of human beings, we have repeated the experiments for both normal (with people walking around) and quiet environment (late in the evening with none around).

For both cases, the behavior of the network is the one we expected: the number of loss packets increases with the number of hops from the base station.

6.1.1 Normal environment

Taking into account the normal environment, we can see in Figure 6.2, that there are a lot of losses due to the normal activity of the department, that is, people are walking around during the experiment.

Node 3 which is three hops away from the base station loses the biggest amount of packets. This is plausible with the theory, since its packets have

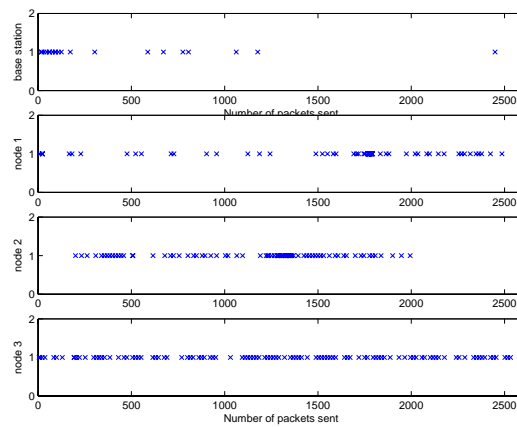


Figure 6.2: Losses in the normal environment

to be forwarded from one node to another for three times before reaching the base station and this increases the loss probability. Furthermore node 3, being approximately 15 meters away from the base station, is likely to lose its packets due to people walking around.

6.1.2 Quiet environment

In this case the losses are negligible (see Figure 6.3) as in a number of 2500 packets sent only 5 are lost by the furthest node, 3 by the next node and none by the node nearest to the base station. The total amount of loss packets is around the 0.2%.

This shows that the motes are very sensitive to the interferences and to the human presence.

6.1.3 Unexpected result

Looking at Figure 6.2 and 6.3 we can see that the base station loses some of its own packets. This is a very strange behavior since the base station is connected to the USB port of the PC and does not send any data packet via radio. It only sends a routing packet once every 10 seconds (by default). We still don't understand the reason for that, but this is not a real problem since data collected by the base station should be discarded as they are altered by the closeness to the PC which is radiating heat.

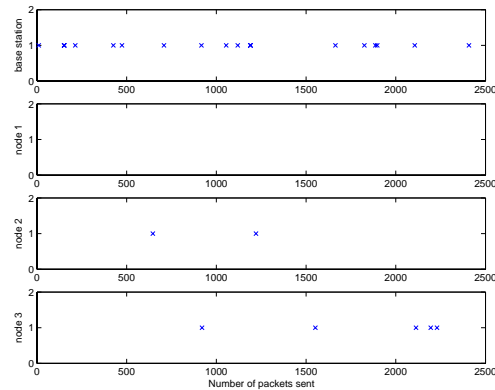


Figure 6.3: Losses in the quiet environment

We tried to contact the TinyOS forum to get any possible help but none has still replied.

6.2 Our application

What we have been doing so far has been trying a way to improve the performances of the motes in terms of sampling rate. Now we would like to see what happens to the loss rate in the network running the motes at different sampling rates.

To achieve this task we cannot use AEGIS since its performances are restricted to three packets per second, that is the time needed by the onboard sensors to sample the measurements of temperature, humidity, PAR and TSR. What we did therefore, was to write a modification of AEGIS, so that there is not need anymore to wait for the measurements from the onboard sensors before sending a packet. This should allow us to send packets in a very high rate.

Once again we did our experiments in both normal and quiet environment.

6.2.1 Results of a star topology

We focus on a four-sensors network like in Figure 6.4

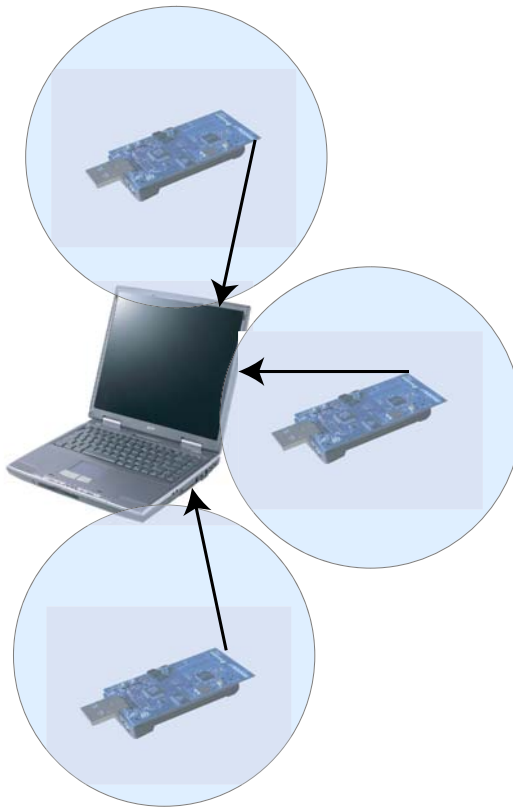


Figure 6.4: Configuration of our real network

We run the experiment first in the normal environment with different sampling rate and then in the quiet environment using the same rates.

What we expect is a loss rate that increases enhancing the sampling rate and changing from a normal to quiet environment.

Looking at the Berkeley application note on Telos [31], a node can achieve

$$\text{node_capacity} = 125\text{Kb/s} \quad (6.1)$$

Considering that a Surge packet is 30 bytes

$$\text{packet_length} = 30\text{bytes} = 240\text{bits} \quad (6.2)$$

and taking into account 6.2 and 6.1, we can compute the maximum rate for each node

$$\text{max_rate} = 125 \cdot 1000 / 240 = 520\text{pkt/s} \quad (6.3)$$

This calculation of course does not take into account the features of the microcontroller and the physical time needed to generate the packet. What we experienced indeed, is a maximum sampling rate of around 120 packets per second for the all network regardless the number of sensors.

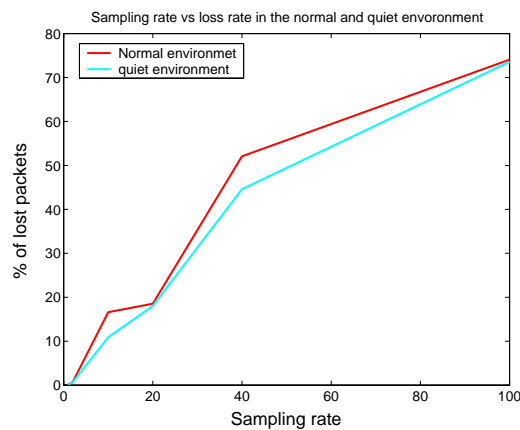


Figure 6.5: Percentage of lost packets with different sampling rates

Looking at Figure 6.5 we can notice that there is not a huge difference between the normal and the quiet environment as it was for the AEGIS application. This is because now we are using a star topology where all the sensors are in the same room, no more than five meters away from the base station and are not spread in a linear topology as in the previous case.

Since in this case there is not a relevant difference between the normal and the quiet environment, we are interested in seeing what happens to the loss rate adding one by one the sensors in the network.

This effect is clearly seen in Fig. 6.6 where the number of sensors varies from one to four.

These results show that when a variable number of motes is used, the loss rate is lower when the transmission rate is below the threshold of 100 packets per second for all the network. When this value increases the network fails drastically until the case where the collisions due to the CSMA contention mechanism are so high that is impossible for the motes to access the channel and only the base station sends its packets.

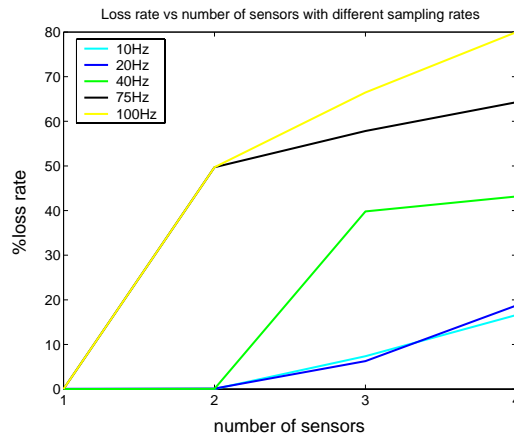


Figure 6.6: Loss rate vs number of sensors

6.3 Baud rate and UART

Due to their energy-consumption constraints, in their basic configuration, motes are not intended for very-high rate applications. Therefore if we want them to perform faster there are some default settings which have to be changed. In addition to changing the code in the required application we need to edit other codes.

First of all, for any microcontroller to communicate with external digital hardware, it must do so through one of many standard data buses. These include the UART. We can change the baud rate of the mote setting a different rate in the HPLUARTM.nc platform (in the line pointed by the command “call USARTControl.setClockRate”) and compiling the required application specifying in the Cygwin window the new baud rate.

```
includes msp430baudrates;

module HPLUARTM {
  provides interface HPLUART as UART;
  uses interface HPLUSARTControl as USARTControl;
  uses interface HPLUSARTFeedback as USARTData;
} implementation {
```

```

async command result_t UART.init() {
    // set up the USART to be a UART
    call USARTControl.setModeUART();
    // use SMCLK
    call USARTControl.setClockSource(SSEL_SMCLK);
    // set the bitrate to newrate
    call USARTControl.setClockRate(UBR_SMCLK_newrate, UMCTL_SMCLK_newrate);
    // enable interrupts

    call USARTControl.enableRxIntr();
    call USARTControl.enableTxIntr();
    return SUCCESS;
}

async command result_t UART.stop() {
    call USARTControl.disableRxIntr();
    call USARTControl.disableTxIntr();
    // disable the USART modules so that we can go in deeper LowPower
    call USARTControl.disableUART();
    return SUCCESS;
}

async event result_t USARTData.rxDone(uint8_t b) {
    return signal UART.get(b);
}

async event result_t USARTData.txDone() {
    return signal UART.putDone();
}

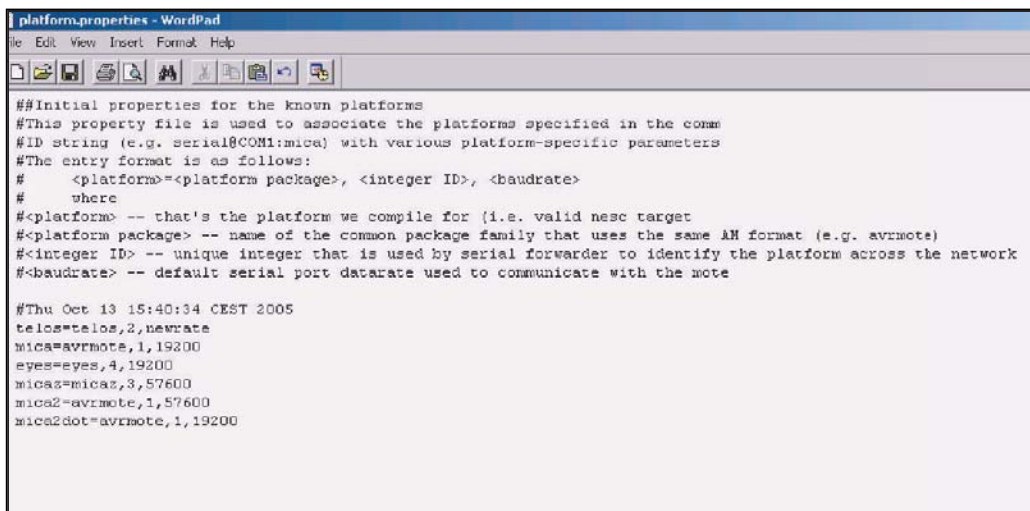
async command result_t UART.put(uint8_t data) {
    return call USARTControl.tx(data);
    .....
    .....
    .....
}

```

All the baud rates available are listed in the `mcp430baudrates.h` file.
Another bottleneck is represented by the serial port UART port in the

PC. A UART (Universal Asynchronous Receiver/Transmitter) is the microchip that controls a computer's interface to its attached serial devices. Specifically, it provides the computer with the RS-232C Data Terminal Equipment (DTE) interface so that it can "talk" to and exchange data with serial devices. By default this link between the mote and the PC only runs at 57.6 Kb/s, thus represents a bottleneck.

Finally, to run a mote with the new rate, we need to edit our `platform.properties` file in order for java to understand the new baud rate. To make this possible, we need the `$TOS.PLATFORM` path to use a Windows paths and not a Cygwin paths, then edit the file pointed to by `$TOS.PLATFORM` and change the baudrate specification for telos to the new rate Figure 6.7.



```
platform.properties - WordPad
File Edit View Insert Format Help

##Initial properties for the known platforms
#This property file is used to associate the platforms specified in the comm
#ID string (e.g. serial@COM1:mica) with various platform-specific parameters
#The entry format is as follows:
#   <platform>=<platform package>, <integer ID>, <baudrate>
#   where
#<platform> -- that's the platform we compile for (i.e. valid nesc target)
#<platform package> -- name of the common package family that uses the same AH format (e.g. avrmote)
#<integer ID> -- unique integer that is used by serial forwarder to identify the platform across the network
#<baudrate> -- default serial port data rate used to communicate with the mote

#Thu Oct 13 15:40:34 CEST 2005
telos=telos,2,newrate
mica=avrmote,1,19200
eyes=eyes,4,19200
micas=micas,3,57600
mica2=avrmote,1,57600
mica2dot=avrmote,1,19200
```

Figure 6.7: Platform.properties file

6.3.1 Results with different baud rate

We can increase the value of the baud rate up to 230.4 Kb/s which allow us to send around 220 packet per second with the Surge application.

Figure 6.8 shows the number of packet sent by a single mote using different baud rates. We can notice that we have a linear behavior until we approach to 57.6 Kb/s, after this value the microcontroller comes close to the saturation and the behavior is no more linear.

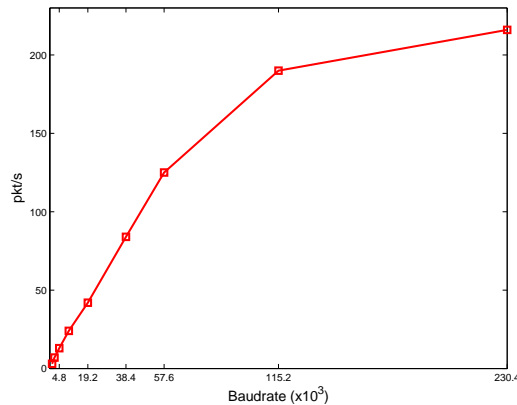


Figure 6.8: Number of packets per second at different baud rates

Note that actual measurements, were only made for the baud rates indicated by the markers. The full lines are interpolated and the exact break point is unknown.

6.3.2 Experiment with a 115,2 Kb/s baud rate

In this paragraph we repeat some of the experiments done in 6.2.1 with the new baud rate of 115,2Kb/s. Particulary we want to know which are the benefits, in terms of number of packets per second, of increasing the baud rate up to 115.2Kb/s compared to the default standard of 57.6 Kb/s. Considering that each node is sending with a 40Hz rate, we can see that, the usage of an higher baud rate, leads to a considerable improvement in the performance, when the total amount of packets in the network goes over the rate of 100 packets per second (Figure 6.9).

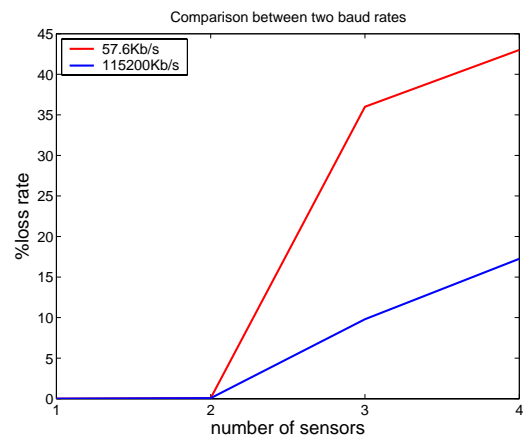


Figure 6.9: Comparison between two different baud rate

Chapter 7

Design of an overlay TDMA

7.1 Introduction

Wireless sensor networks are similar to mobile ad-hoc networks (MANETs) in that both involve multi-hop communications. However, they differ from each other for the the nature of the applications and routing requirements. First, the typical mode of communication in a sensor network is from multiple data sources to a data collector or base station, rather than communication between any pair of nodes. Second, in most scenarios, sensors are not mobile (even if the sensed phenomena may be), so the nature of the dynamics in the two networks is different. Finally, wireless sensor networks have the huge constraint of saving energy. This situation is more pronounced than the one in MANETs, where the communicating devices can be replaced or recharged relatively often. The scale of wireless sensor networks and the necessity of unattended operation for months at a time means that energy resources have to be managed even more carefully. For these reasons, many of the end-to-end routing protocols proposed for MANETs in recent years are not suitable for wireless sensor networks [10, 21, 22]. Furthermore, in many applications of wireless sensor networks it is desirable that data from sensors arrive to the fusion center within a latency requirement.

The goal of this chapter is to design an overlay TDMA with the minimal latency. The idea behind this approach is to demonstrate that in WSN, the usage of a TDMA protocol which is able to schedule the transmissions of the nodes in order to have a convergent data flow from the furthest no-

des toward the base station, might reduce the latency of the network as well as avoid the collisions due to the prevailing CSMA protocol.

7.1.1 Assumptions

We focus our attention on a single network flow that is assumed to consist of a single fusion center attempting to gather information from a number N of data sources. Our protocol makes the following assumption:

- a fixed number of nodes and links. Let us say N and L respectively.
- A potentially multiple routing trees rooted at the fusion center.
- All nodes produce packets at fixed uniform rate
- Each node knows the number of its children ¹.

Most of the above assumptions are natural for sensor networks applications where one uniformly (over time) collects data of some variable (temperature, humidity). The fourth assumption is more technical, but this information can be derived from the routing table.

7.1.2 Proposed solution

The underlying idea of the protocol is to schedule the transmissions of the nodes in order to have a data flow that converges from the furthest node toward the base station. In such a solution, when nodes one hop away from the base station get a transmission right, can send their information with the information coming from their children, thus reducing the number of transmissions needed to have all the data collected at the base station.

Each step of the protocol implementation is described below:

1. The fusion center generates an empty reservation packet which consists of as many time slots as the number of nodes minus one. This assumption requires that the fusion center knows exactly the number of nodes N in the network.

¹we consider all the nodes from the lower levels as children

2. Once the reservation packet is generated, the fusion center makes as many copies of it as the number of its branches. Then for each of its branches, it sends one copy of the packet with as many free slots as the number of its children in that branch and the other slots blocked.
3. At the lower level, the node receiving the reservation packet, assigns itself the last free time slot inside the free field assigned him by its father. Then it acts like the fusion center, *i.e* it makes as many copies of the received packet as the number of its branches and again reserves/blocks the remaining slots (inside the free field let available for him by its father) for its branches taking into account once again the number of children per each branch.
4. This procedure keeps going until the reservation packet reaches the leaf nodes.
5. Each leaf node finds available only one slot, reserved for it by its father in the reservation packet. This allows the leaf nodes to transmit before their parents. Therefore, the data flow begins to converge toward the base station.

In Fig. 7.1, we consider an example of this procedure: the fusion center has three branches with 1 children in the first, 2 in the second and 3 in the third. It generates the reservation packet composed by 6 slots, makes three copies of it and sends to the first branch a copy with the first slot available and the 5 remaining blocked. Then it sends to the second branch the second copy with the first slot and the last three blocked and only the two in the middle available. Finally the third copy with only the last three slots available is sent to the third branch.

Node 2 gets the reservation packet, assigns itself the last free time slot inside the free field assigned to him by its father and sends the packet to its child (the leaf node 1) which finds only one time slot available.

Likewise, node 4 gets the reservation packet, assigns itself the last free time slot inside the free field assigned him by its father, makes two copies (one for each branch) of the reservation packet and sends it to its children leaving free only one slot for each.

Nodes are now scheduled in the way that ensures the minimal latency, that is the information converges from the furthest nodes toward the base

station. This optimal solution is not unique, but there exist several possibilities to reach it. The relevant aspect is that at each level, children should transmit before their parents, so that the information can converge toward the fusion center.

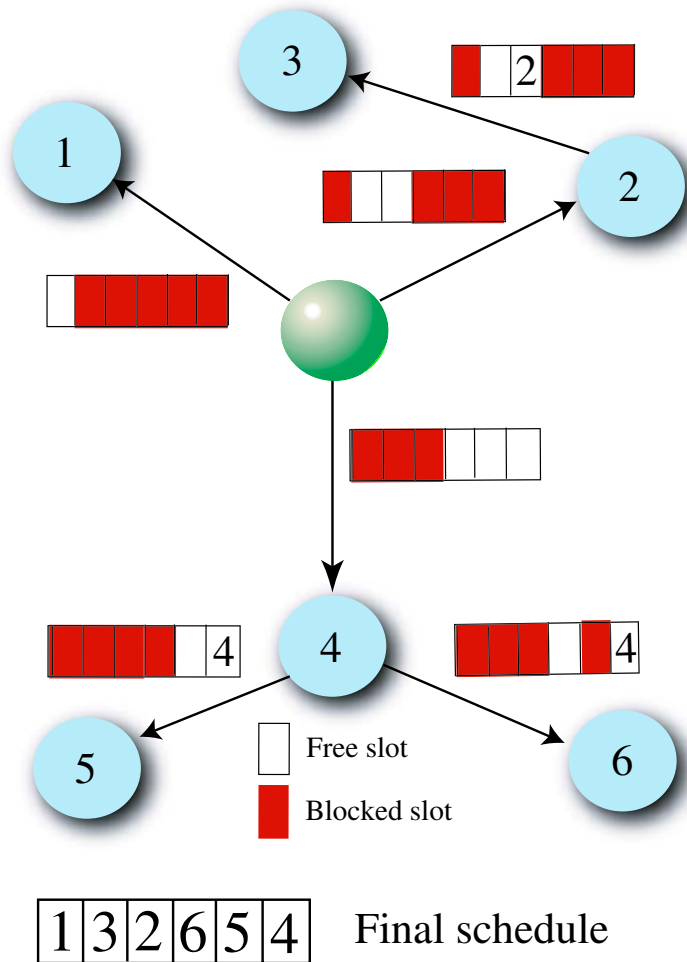


Figure 7.1: Procedure to create the schedule

7.2 Method analysis

In this section present two methods (M1 and M2) used to generate a TDMA schedule in a wireless sensor network and we demonstrate that our protocol represents the framework of a distributed algorithm that allows to get the optimal solution in terms of latency in the two methods.

- **Method M1:** The frame length fixed to L time slots. One transmission right is guaranteed to each link in every frame. Furthermore each time slot is virtually divided into L mini-slots, so that a node is allowed to empty its buffer in the worst case (linear topology)
- **Method M2:** Each time-slot allows only one packet to be transmitted but we can decide the length of the frame and thus, a lot more time-slots to certain links. Once again the frame is assumed to be complete when all the nodes have transmitted at least once, then the same frame is repeated until all packets have reached the fusion center.

In the next analysis we will evaluate the performances of M1 and M2 where the schedule is generated by a random assignment of transmission rights. Both method are implemented for the network in Figure 7.2 where we assume that each node has only one packet to transmit. In our analysis we consider the latency in terms of mini-slots, as they are directly related to the forwarding delay in seconds (a mini-slot corresponds to the transmission of a packet). However, for a more complete overview, we report the plots of the latency also in terms number of frames per schedule and number of empty-slots per schedule.

The network behavior has been simulated for 1000 times, generating different schedules by a **random selection** among all the nodes. Finally we will show the benefits that our structured TDMA scheduling can apport to both methods

7.3 Method M1

In method M1 the frame-length is fixed to L time slots (here *virtual slots*), each one divided in L mini-slots (or *physical slots*) so that each virtual slot allows a node to empty its buffer in the worst case (linear topology).

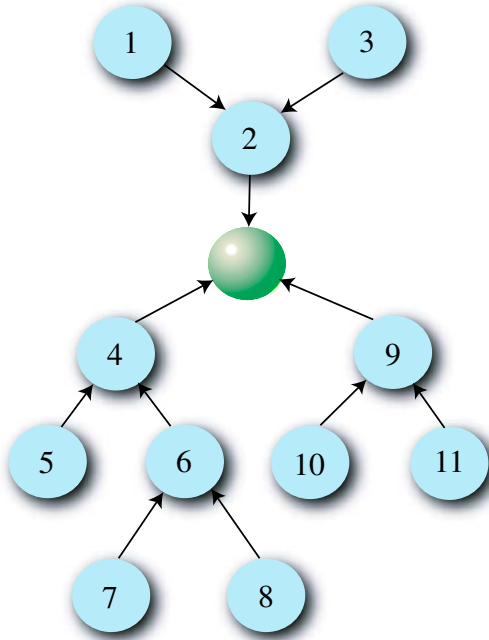


Figure 7.2: Sensors network

In this setting, the worst-case latency is $M = L \cdot |\max L(p)|$ mini-slots, that is the time needed to make all packets reach the fusion center. Here L is the length of the frame which in this case is equal to the number of links and $|\max L(p)|$ is the length of the longest path. This happens when a frame is created so that the nearest nodes to the fusion center, are the first ones to transmit.

To summarize the results obtained with method M1, Figure 7.3(a) shows the frequency of frames per schedule in a number of 1000 independent experiments. Figure 7.3(b) illustrates the frequency of the schedule size in mini-slots and Figure 7.3(c) highlights the statistic number of empty mini-slots per schedule.

Since the frame is randomly generated it might happen that all packets reach the fusion center using only one frame. This is the optimal solution and we can reach it when all the information converges from the furthest nodes toward the fusion center.

Even if there exist several combinations that satisfy the optimal requi-

rements, they represent a small fraction of the $L!$ possible combinations. In our simulation obtained the optimal solution by chance in the 1% of the cases.

To obtain the optimal solution in a centralized way is a trivial problem, since the fusion center has the knowledge of the network and can easily generate the optimal schedule. However in a wireless sensor network, as nodes may fail due either to the unflattering conditions or to the limited battery-life, a distributed algorithm is more desirable.

Our overlay TDMA permits to reach one of the optimal solution of M1 in a distributed way.

7.4 Method M2

In this method, as in the other one, a frame is finished when all nodes have got at least one transmission right. However, in this case, the frame length is not fixed a priori since a node can get more transmission rights consecutively, in particular when a node gets a transmission right it is allowed to use more time-slots adjacent as long as it has emptied its buffer (remember that in this method a time slot allows only one packet to be sent). Then the same node can get others transmission rights in the same frame, if some of its predecessors still has to transmit. Looking at Matlab simulations, we can see that this method leads to a solution that depends on the schedule (Figure 7.4(a), 7.4(b) and 7.4(c)). This means that the order of transmission is very important. If the frame is generated so that the furthest nodes from the fusion center are the first ones to transmit we can obtain the best solution which consists of only one frame with 21 time slots used and none empty as we can see in Figure 7.5(a), 7.5(b) and 7.5(c).

As in the previous case, the optimal solution is not unique, however a random assignment of transmission rights shows that the minimal latency may be reached only in the 10% of the cases.

The analogy between the best solution of this method with that one reached with our protocol, is not easy to recognize at first sight.

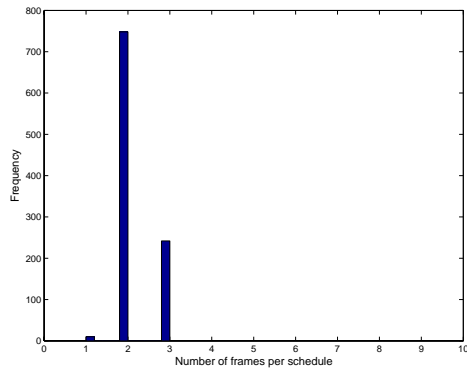
Our protocol can be simply adapted to suit with this method simply changing one assumption: rather than knowing the number of children, each node needs to know the total number of packets which have to be forwarded by itself when it acts as a forward node. For a network composed by a few nodes, this information could be stored in the routing table, thus this assumption does not add any computational cost. The reservation packet is composed by a number of slots that can be derived by (7.1)

$$\text{Number of slots} = S = \sum_{i=1}^N h_n \quad (7.1)$$

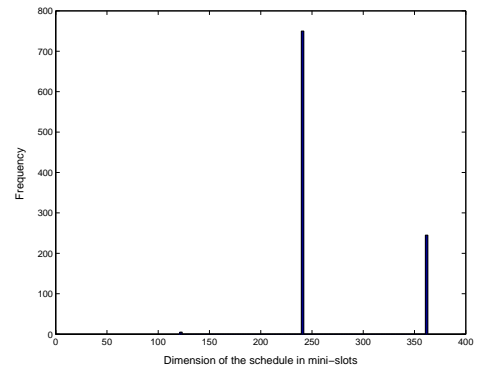
where N is the number of nodes and h_n represents the number of hops of the n -node to the base station. Bringing back to Fig 7.2 one of the possible optimal solutions is given by the schedule in Fig 7.6

7.5 Conclusions results

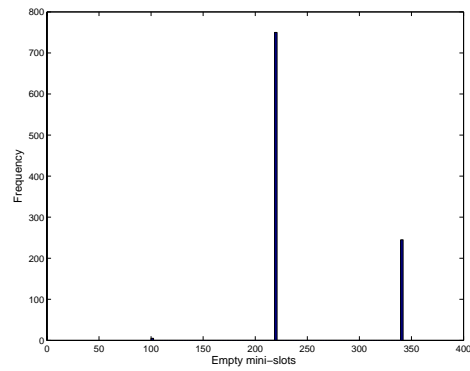
We have explained in detail two methods to evaluate the latency in the case of a simple network consisting of 11 nodes and a fusion center, which is able to generate a schedule randomly selecting among all the users. In terms of latency (remember that we refer to the number of mini-slot to calculate the latency), method M2 is clearly the better one: in our simple network the latency is 21 mini-slot compared to 121 mini-slots of method M1. Our protocol is well-suited for both methods and gives a strong improvement over a casual schedule, Fig. 7.7.



(a) Latency in terms of frame per schedule. Best method M1 has 1 frame

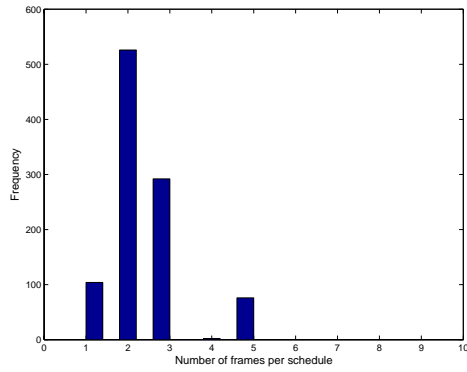


(b) Dimension of the schedule in mini-slots. Optimal dimension obtained by method M1 is 121 mini-slots

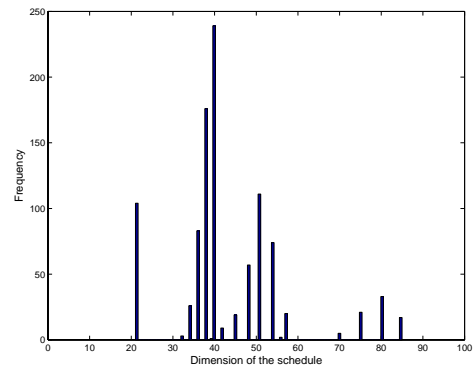


(c) Number of empty mini-slots per schedule. The optimal solution consists in 100 empty mini-slots

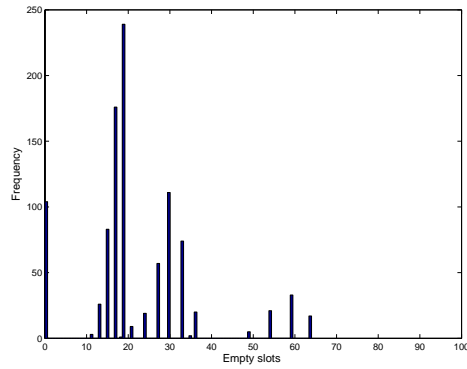
Figure 7.3: Latency analysis in Method M1.



(a) Latency in terms of frame per schedule.

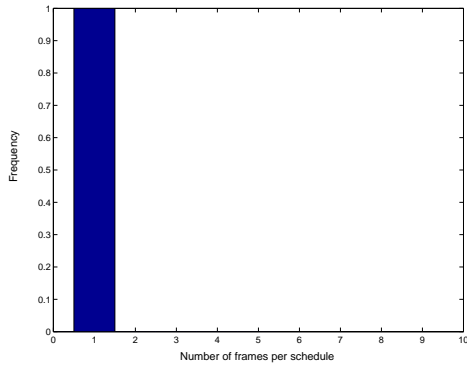


(b) Dimension of the schedule.

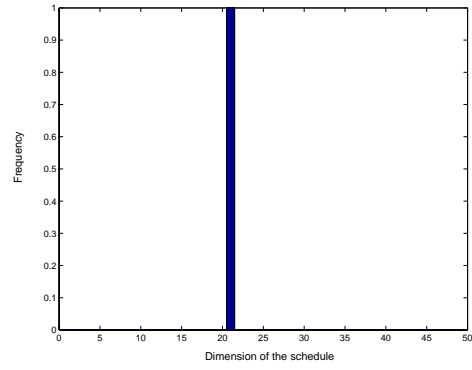


(c) Number of empty-slots per schedule.

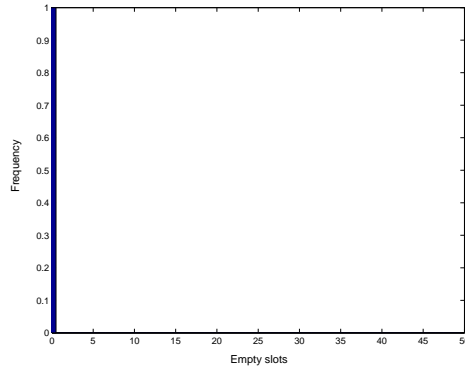
Figure 7.4: Latency analysis in Method M2.



(a) Latency in terms of frame per schedule.



(b) Dimension of the schedule.

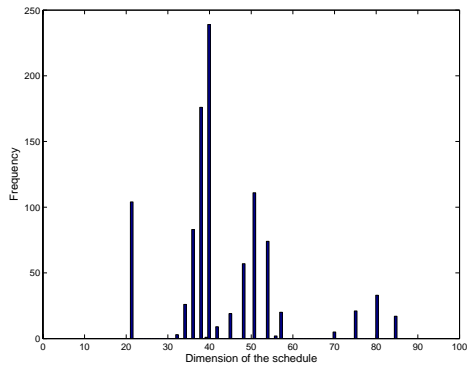


(c) Number of empty-slots per schedule.

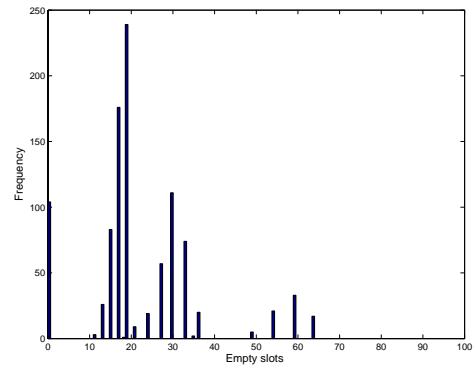
Figure 7.5: Latency analysis in Method M2 in the optimal case.

7	8	6	6	6	5	4	4	4	4	4	10	11	9	9	9	1	3	2	2	2
---	---	---	---	---	---	---	---	---	---	---	----	----	---	---	---	---	---	---	---	---

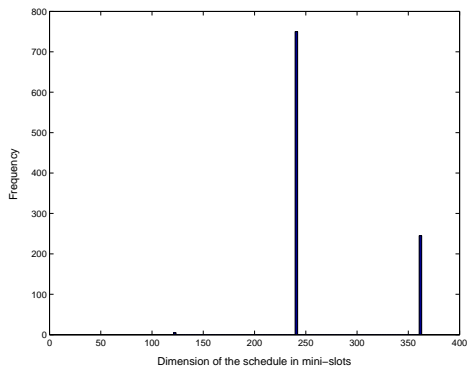
Figure 7.6: Possible schedule for the optimal solution



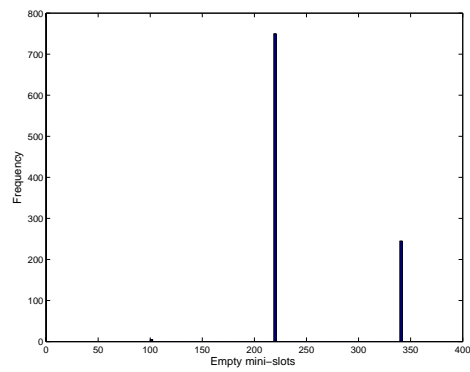
(a) Dimension of the schedule in M2.



(b) Number of empty-slots per schedule in M2.



(c) Dimension of the schedule in M1.



(d) Number of empty-slots per schedule in M1.

Figure 7.7: Comparison between Method M2 and M1.

Chapter 8

Conclusions & future work

8.1 Conclusions

Wireless sensor networks can be used for a wide range of application areas(e.g., military, home, environment), each of which poses its own particular technical issues. One key problem that arises in many applications, is how to organize the computation and communication in a sensor network to enable accurate estimation of the state of a dynamic system.

As starting point in this thesis, we demonstrate that the performances of the optimal estimator are influenced by packet loss and delays. Thus, when we want to use a sensor network platform to collect data and forward measurements to a data collector, it is critical to understand the loss performance of the current platform and, possibly, to design MAC schemes that eliminate losses and minimize latency.

The main effort of this thesis has been to improve the performance of the Telos Tmote sky in terms of data rate. As the sensors are not intended for high rate applications the default throughput in terms of packet per second was very poor. Investigating on the hardware and software of the motes we have been able to obtain a throughput suitable for high rate applications.

The natural effect of increasing the number of packets per second results in an huge amount of traffic in the network. Telos motes use a CSMA/CA protocol to access the channel, *i.e.* there is not synchronization a priori, this results into a great collision probability. To solve the problem of data delay and to avoid the collisions that lead to packet losses, we propose

an overlay TDMA over CSMA. Using a distributed algorithm, as it is our overlay TDMA, we can schedule the nodes to have the minimal latency and to avoid collisions. To show the benefits of our protocol we simulate the behavior of a general network in which nodes are randomly scheduled by the fusion center. Our protocol permits to obtain, in a distributed way, the optimal solution of such a network resulting in a strong improvement over a casual schedule.

8.2 Future work

A good point for a future work could be a more detailed understanding of the performance bottlenecks. It could be useful to understand if the bottlenecks for the limited rate depend on the saturation of the CPU, or on the collision of packets or on some external interference.

As we did a theoretical investigation on how delays influence the optimal estimator performance, we would like to write an application that stamps a timer into the payload of the node's packet, so that when this packet is received by the base station, we can check its latency.

Finally another good point for a future work could be to implement our MAC protocol on the sensors and see the practical performance.

Bibliography

- [1] Chipcon web site. In <http://www.chipcon.com>.
- [2] Citris annual report 2004-2004. In <http://citris.ucdavis.edu/annualreport.pdf>.
- [3] Cygwin web site. In <http://www.cygwin.com>.
- [4] Moteiv web page. In <http://www.moteiv.com>.
- [5] Tinyos tutorial. In <http://www.tinyos.net/tinyos-1.x/doc/tutorial/>, September 2003.
- [6] Chipcon data sheet. In http://www.chipcon.com/files/CC2420_Data_Sheet_1_2.pdf, September 2004.
- [7] Intel research group. In *Instrumenting the world*, 2004.
- [8] Tinyos community forum. In <http://www.tinyos.net>, 2004.
- [9] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. In *IEEE Communications Magazine*, volume 40, pages 102–114, August 2002.
- [10] A. Bonivento, C. Fischione, A. Sangiovanni-Vincentelli, F. Graziosi, and F. Santucci. Seran: A semi random protocol solution for clustered wireless sensor networks. In *International Workshop on Localized Communication and Topology Protocols for Ad hoc Networks*, November 2005. To appear.

- [11] B. Bougard, F. Catthoor, D. Daly, A. Chandrakasan, and W. Dehaene. Energy efficiency of the IEEE 802.15.4 standard in dense wireless microsensor networks: modeling and improvement perspectives. In *Design, Automation and Test in Europe, 2005. Proceedings*, volume 1, pages 196–201, 2005.
- [12] A. Boulis, S. Ganeriwal, and M. Srivastava. Aggregation in sensor networks: an energy-accuracy trade-off. In *IEEE First International Workshop on Sensor Network Protocols and Applications.*, pages 128–138, May 2003.
- [13] A. Cerpa, J. Elson, M. Hamilton, and J. Zhao. Habitat monitoring: application driven for wireless communications technology. In *ACM SIGCOMM'2000*, April 2001.
- [14] L. P. Clarea, G. J. Pottie, and J. R. Agre. Self-organizing distributed sensor networks.
- [15] M. Corporation. Tmote sky quick start guide. In <http://www.moteiv.com/products/docs/telos-revb-quickstart.pdf>, 2004.
- [16] D. Gay, D. Culler, P. Levis, and E. Brewer. nesc language reference manual. May 2003.
- [17] J. Gutierrez, M. Naeve, E. Callaway, M. Bourgeois, V. Mitter, and B. Heile. IEEE 802.15.4: a developing standard for low-power low-cost wireless personal area networks. In *IEEE Network*, volume 15, pages 12–19, Sept.-Oct.
- [18] X. Jiang, J. Polastre, and D. Culler. Perpetual environmentally powered sensor networks. In *Fourth International Symposium on Information Processing in Sensor Networks, 2005. IPSN 2005*, pages 463–468, April 2005.
- [19] V. A. K. Sohrabi, J. Gao and G. J. Pottie. Protocols for self-organization of a wireless sensor network. In *Personal Communications, IEEE [see also IEEE Wireless Communications]*, volume 7, pages 16 – 27, October 2000.
- [20] A. Krasnopeev, J.-J. Xiao, and Z.-Q. Luo. Minimum energy decentralized estimation in sensor network with correlated sensor noise. In

Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005 ICASSP'05, volume 3, pages 673–676, March 2005.

- [21] L. Krishnamachari, D. Estrin, and S. Wicker. The impact of data aggregation in wireless sensor networks. In *22nd International Conference on Distributed Computing Systems Workshops, 2002*, pages 574–578, July 2002.
- [22] L. Krishnamachari, D. Estrin, and S. Wicker. Modelling data-centric routing in wireless sensor networks. In *IEEE INFCOM, 2002*, 2002.
- [23] R. Krishnan and D. Starobinski. Message-efficient self-organization of wireless sensor networks. In *IEEE Wireless Communications and Networking, 2003. WCNC 2003.*, volume 3, pages 1603 – 1608, 16-20 March 2003.
- [24] G. Lu, B. Krishnamachari, and C. Raghavendra. Performance evaluation of the IEEE 802.15.4 mac for low-rate low-power wireless networks. In *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.*, page 224, April 2004.
- [25] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *WSNA'02*, September 2002.
- [26] J. Misic and V. Misic. Duty cycle management in sensor networks based on 802.15.4 beacon enabled MAC. In *Ad hoc & sensor wireless networks*, volume 1, pages 207–233, March 2005.
- [27] I. C. Molero. Telos and aegis quick start guide. April 2005.
- [28] M. Neugebauer, J. Plonnigs, and K. Kabitzsch. A new beacon order adaptation algorithm for IEEE 802.15.4 networks. In *Proceedings of the Second European Workshop on Wireless Sensor Networks, 2005.*, pages 302 – 311, 31 Jan.-2 Feb 2005.
- [29] N. Noury, T. Herv, V. Rialle, G. Virone, E. Mercier, G. Morey, A. Moro, and T. Porcheron. Monitoring behavior in home using a smart fall sensor and position sensors. In *1st Annual International, Conference*

On Microtechnologies in Medicine and Biology, pages 607–610, October 2000.

- [30] I. of Electrical and E. Engineers. IEEE std 802.15.4. Thecnical report, IEEE, May 2003.
- [31] J. Polastre, R. Szewczyk, and D. Culler. *Telos: Enabling Ultra-Low Power Wireless Research*. 26-27 April 2005.
- [32] M. Rabbat and R. Nowak. Distributed optimization in sensor networks. In *Third International Symposium on Information Processing in Sensor Networks 2004*, pages 20–27, April 2004.
- [33] SENSIRION. STH1x/STH7x humidity & temperature sensors. In <http://www.sensirion.com/images/getFile?id=25>, July 2004.
- [34] J. Thorn. Deciphering tinyos serial packets. In *Octave Tech Brief #5-01*, March 2005.
- [35] J.-J. Xiao, S. Cui, and A. J. Goldsmith. Optimal power scheduling of universal decentralized estimation in sensor networks. November 2004.
- [36] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus.