المدرسة العليا للمواصلات بتونس

ÉCOLE SUPÉRIEURE DES COMMUNICATIONS DE TUNIS

Telecommunication Engineering

Option:

Network Engineering (IRES)

Graduation Project Report

Topic:

Evaluation of Smart-HOP: A Handoff Approach for Mobile Wireless Sensor Networks

By:

Meriam THAMRI

Supervisors:

Prof. Màrio ALVES
Mr. Hossein FOTOUHI
Mrs. Houda KHEDHER

Work proposed and elaborated within:

CISTER - Research Centre in Real-Time & Embedded Computing Systems

Academic Year : 2011/2012

Dédicaces

Je dédie ce travail,

À mes très chers parents, Taoufik et Olfa, que j'aime infiniment, dont le soutien, le dévouement et les sacrifices m'ont permis d'être là où je suis,

À ma petite sœur Salma,

à sa fraîcheur et à son sourire qui illuminent ma vie,

À mes amis, Houssem, Salma, Amira, Ghassen, Hichem, Hassene, Oussama, les meilleurs au monde, pour leur présence, leur patience, leur affection, leurs encouragements

À mes grands-parents, à tous mes oncles, mes trésors,

À tous ceux que j'aime,

Qu'ils trouvent dans ce travail le témoignage de mon profond amour et l'expression de ma sincère reconnaissance.

Votre Meriam

FORWARD

This document reports the work that was carried out as part of my graduation project at SUP'COM, the Higher School of Communication of Tunis, in collaboration with the research lab CISTER, to obtain the engineering degree in Telecommunications.

CISTER (Research Centre in Real-Time Computing Systems) is a top-ranked Research Unit based at the School of Engineering (ISEP) of the Polytechnic Institute of Porto (IPP), Portugal. The IPP-HURRAY research group, created in mid 1997, is the core and genesis of the CISTER Research Unit.

ABSTRACT

Mobility management is a new emerging issue in wireless sensor networks. Characteristics and resources specifications and limitations represent a real challenge for the algorithm designers in order to meet with the various applications requirements.

Many handoff mechanisms are proposed to deal with this concern. Smart-HOP is a handoff process which is currently developed at The CISTER research unit and specifically designed for WSNs. An evaluation of this algorithm performances and behavior is needed to setup its parameters. For this purpose, an analytical analysis and a simulation model were developed and, based on their results, the implementation of the algorithm was performed.

KEYWORDS: WIRELESS SENSOR NETWORKS, MOBILITY MANAGEMENT, HANDOFF, TINYOS

ACKNOWLEDGMENT

I would like to express my gratitude to Mr. Eduardo TOVAR, the Head of IPP-HURRAY research group, for having hosted me in CISTER.

My grateful thanks also go to Prof. Màrio ALVES and Mr. Hossein Fotouhi for sharing their precious time and positive insights and for their consistent and generous support during the project schedule.

The special thanks go to Mrs. Houda KHEDHER, conference master SUPCOM assistant for the unwavering assistance and guidance throughout this project.

Not forget, great appreciation go to the rest of CISTER members and staff who helped me as much as they could during the project.

My sincere gratitude to all the persons who have made the completion of this work possible and who have never ceased helping me until this project ended.

Last but not least, I would like to thank all the members of my evaluation committee at SUP'COM for their acceptance to asses my work.

CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

A

AP:	ACCESS POINT

B

BER:	BIT ERROR RATE

C

CDMA:	CODE DIVISION MULTIPLE ACCESS
CPU:	CENTRAL PROCESSING UNIT

D

DSP:	DIGITAL SIGNAL PROCESSOR

G

GPS:	GLOBAL POSITIONING SYSTEM

H

HO:	HANDOVER OR HANDOFF
HM:	HYSTERESIS MARGIN

I

ISM:	THE INDUSTRIAL, SCIENTIFIC AND MEDICAL RADIO BANDS

M

MAC:	MEDIA ACCESS CONTROL
MN:	MOBILE NODE

N

NESC:	NETWORK EMBEDDED SYSTEMS C

O

OS:	OPERATING SYSTEM

P

PC:	PERSONAL COMPUTER

Q

QOS:	QUALITY OF SERVICE

R

RAM:	RANDOM ACCESS MEMORY
ROM:	READ ONLY MEMORY
RF:	RADIO FREQUENCY
RSSI:	RECEIVED SIGNAL STRENGTH INDICATION OR INDICATOR

S

SNR:	SIGNAL-TO-NOISE RATIO

T

TDMA:	TIME DIVISION MULTIPLE ACCESS

U

UART:	UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER

W

WSN :	WIRELESS SENSOR NETWORK

General Introduction

During the past few decades, the environment data observation, tracking and real time monitoring need arose rapidly especially with the recent progress in networking and wireless communication fields. Nowadays, monitoring applications are designed with very strict technical requirements in terms of delay and performances.

In this context and in order to meet with these expectations, an emerging branch of telecommunications researches was created, introducing the wireless sensor networks concept. In this type of networks, mobile wireless sensor nodes report valuable and precise information concerning their immediate environment or the target they are tracking.

The mobility management in such networks consists a capital issue to maintain a reliable and continuous data stream and ensure the high efficiencies demanded.

The CISTER research unit is currently developing the smart-HOP mechanism under MASQOTS FCT project. Smart-HOP is a solution which was specially designed for applications such as health-care monitoring where sensors collect the vital signs of various patients thanks to embedded wireless sensing devices. Smart-HOP describes a handoff mechanism where mobile nodes select the best available access point for their data transmission.

For further evaluation of the proposed Smart-HOP algorithm and investigate the functionality of the mechanism in various situations we conducted several analytical and simulation comparisons. The main goal of this first stage was to find out the importance of handoff process. The analysis would help to setup and perform more wisely experiments that consider the findings of preliminary and simple tests that were done previously.

During this project, an analytical analysis and a simulation model of the proposed algorithm were established. The obtained results were used in setting up the configuration parameters calibration of the smart-HOP mechanism.

This report is divided into four chapters. In the first chapter, a general description of wireless sensor networks, their architecture, their characteristics and their main applications is presented. In the second chapter, the mobility challenge major issues in these networks are highlighted and we will give then an overview of the smart-HOP mechanism. After that, in the third chapter, we will focus on the smart-HOP analytical analysis and its simulation model. The last chapter will be dedicated to the implementation of The smart-HOP algorithm in TelosB motes with TinyOS.

CHAPTER I:

WIRELESS SENSOR NETWORKS

Introduction

A wireless sensor network, or WSN, consists of a collection of wireless sensor nodes spatially distributed realizing different kinds of measurements related to their environment or to physical monitoring. The most common data that can be reported by these sensors are the temperature, the humidity, the brightness, pressure and etc. The sensing data collected by sensor nodes is propagated through the wireless channel to be used for a specific task. It may be utilized for an online and distributed decision making purpose or to be collected at a central node known as sink/destination node. The requirements of applications impose various challenges in the protocol design of WSNs.

In this chapter, first we explain the architecture of sensor nodes followed by their main components. Then the main applications will be briefly introduced. The challenges which are of paramount importance will be defined in this chapter. Finally, we will explain the mobility management as one of the new challenges and the handoff model as a solution to cope with it.

I.1 Sensor Architecture

In the literature, a sensor, which is also called detector, is basically a converter that measures a physical quantity and converts it into a signal that can be read directly by an observer or by an instrument, mostly electronic. The difference between a sensor and a measuring device is that the sensor is nothing more than the simple interface between the physical variable and its detected value. On the other hand, the measuring instrument, can be self-sufficient by providing a display and a database to store and collected[1].

The deployment of a wireless sensor network and its characteristics must perfectly match with the special requirements of a given application. The nodes should be meet criteria in terms of processing capability (one or more microcontrollers, CPUs or DSP chips), memory capacity (program, data and flash memories), radio communication support (RF transceiver), power source management (batteries and solar cells), number of sensors and actuators accommodation, size, price, etc[2].

The wireless sensors constructors designed different types of these devices, with various characteristics that can correspond to distinct uses and requirements. However, they have similar structure and general architecture.

I.1.1 Components of the Sensor

The architecture of a wireless sensor is basically made up of these main modules as shown in the figure bellow:

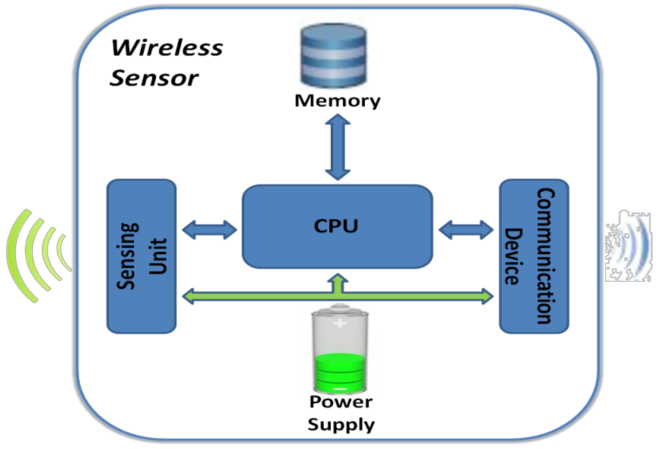


Fig I.1: Wireless sensor architecture

- Central Processing Unit
- Memory
- Communication device
- Power supply
- The sensing unit

I.1.1.1 Central Processing Unit

The Central Processing Unit abbreviated CPU, is the hardware which is responsible for executing the instructions execution by performing the basic arithmetic, logical and input/output operations. The CPU is in charge of managing the data transmission between the nodes, the treatment of the different tasks and the running of the protocols.

The energy consumption, which is one of the major topics in WSNs, depends on the amount of the local operations, the occurrence of the events in the network (packets exchange) and the complexity of the algorithm.

I.1.1.2 Memory

The random access memory, RAM, is used to record different measurements and store in buffers after receiving packets and before sending packets. The nonvolatile storage or read only memory, ROM, is used to save the program code before executing. Generally, sensor nodes only require relatively small amounts of program and storage memory.

I.1.1.3 The Communication Device

The radio communication device is responsible for ensuring the nodes packets exchange. Its principle characteristics are the carrier frequency, the data rate, the modulation type, the receiving and transmitting power level, consumption control, the gain, and the coverage.

The transceiver can be controlled and be powered only when it is needed to reduce the energy consumption. By optimizing this procedure, the small capacity batteries that supply the mobile nodes are more efficient.

I.1.1.4 The Sensing Unit

The probe captures the observed phenomenon and converts it from analog signal to digital. The value is reported to the Central processing unit for further processing.

The energy consumption of the sensing unit depends on its type. The table bellow shows the most commonly available sensor types with their specific characteristics in terms of current, sampling time and voltage requirement.

	Current	Discrete Sample Time	Voltage requirement	Manufacturer
Photo	1,9 mA	330 uS	2,7 - 5,5 V	Taos
Temperature	1 mA	400 mS	2,5 - 5,5 V	Dallas Semiconductor
Humidity	550 uA	300 mS	2,4 - 5,5 V	Sensirion
Pressure	1 mA	35 mS	2,2 - 3,6 V	intersema
Magnetic fields	4 mA	30 uS	any	Honeywell
Acceleration	2 mA	10 mS	2,5 - 3,3 V	Analog Devices
Acoustic	5 mA	1 mS	2 - 10 V	Panasonic
Smoke	5 uA	-	6 - 12 V	Motorola
Passive IR (Motion)	0 mA	1 mS	any	Melixis
Photosynthetic light	0 mA	1 mS	any	Li-Cor
Soil Moisture	2 mA	10 mS	2 - 5 V	Ech2o

Tab I.1: Commonly available sensor characteristics

I.1.1.5 The Power Supply

This Unit provides the required energy to the sensor, it performs the measurements related to the power and estimates the remaining lifetime of the mote.

The mobile nodes are minted to operate for relatively long periods, but as they are battery powered, the control of this consumption is very important. In order to minimize it, some components, like the transceiver, of the mote are switched off.

To get a general picture, you can find in the Appendix 1 the characteristics of some wireless sensors, from the UC Berkeley Family.

In our case, we will use TelosB motes which are working with the TinyOS open-source operating system specially designed for wireless sensors. More details are attached in the Appendix 2.

I.1.2 Software Architecture

The software technologies design is closely related to the hardware platform in the wireless sensor networks due to the numerous constraints. The demand is very specific and the resources are limited. Hence, the operating systems for WSN nodes have to be much less complex than other general purpose operating systems, they must fulfill these requirements:

- ***Robustness:*** the sensor network should be able to provide a long term service. The nodes are deployed in a field and are supposed to work for months or years.
- ***Low resource usage:*** a sensor network node, as we already said, is battery powered. Hence the algorithm should be efficient to occupy minimum memory, processing and communication usage.
- ***Adaptability to continuous evolutions:*** sensor motes are employed in various applications. It is possible to encounter different situations in terms of environment or network changes which oblige the nodes to be adaptative.
- ***Adaptability to the application requirements:*** each application has its very specific requirements in terms of sensing, communication, lifetime… On the other hand, these applications do not require the same interactivity as for PCs. The operating system for wireless sensor networks does not need to include any support for the user interface.

I.2 WSN architecture

Wireless sensor networks are basically ad-hoc networks. The nodes are distributed and cooperate to monitor physical or also environmental conditions of the considered area.

As we already mentioned, this type of networks has some resource constraints, mainly the low processing speed, limited storage capacity and communication bandwidth. Also fluctuations in the physical environment where the network is deployed can cause many changes in its connectivity and affect the networking protocols.

Due to all these considerations, the protocols designed for WSNs are very specific and must take account of these factors [3]:

- ***Fault tolerance:*** a node must be with no trouble and rapidly substitutable and its networking functionalities supported by anther if any interruption occurs. The failure of one node, which is very frequent because of a lack of power, a physical damage, or an environmental interference, should not affect the overall task of the sensor network.
- ***Scalability:*** a wireless sensor network can be reduced or enlarged very easily.
- ***Deployment:*** the deployment of the nodes must be optimized in terms of coverage and location of the different nodes.
- ***Power management:*** The nodes lifetime must be maximized.

I.2.1 General Architecture of WSNs

A wireless sensor network is essentially constituted by a certain number of sensor nodes, one sink node, and a data processing center as represented in the following figure.

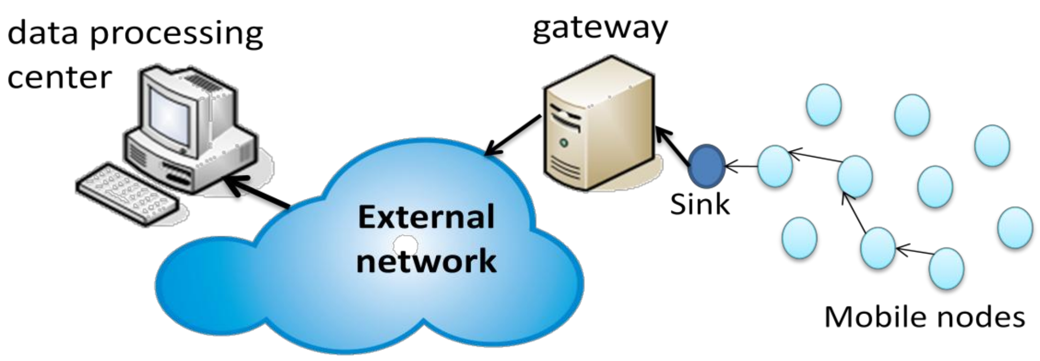


Fig I.2: Organization of a WSN

The sensor nodes are spatially distributed, according to the application criteria. They are generally battery powered.

The sink node is a particular node in the wireless sensor network, since it is responsible of gathering data from all the sensor nodes in the network. Given the importance of this node, it must be powered by a reliable energy source.

The data processing center receives all the data needed from the sink node, it treats the information and provides it to the specific application. The sink and the data processing center may be not directly linked, so that the data needs to be transferred through other networks. This is why a gateway is possibly required then.

I.2.2 Protocol Stack

The Architectural layers of a WSN are represented in the following figure:

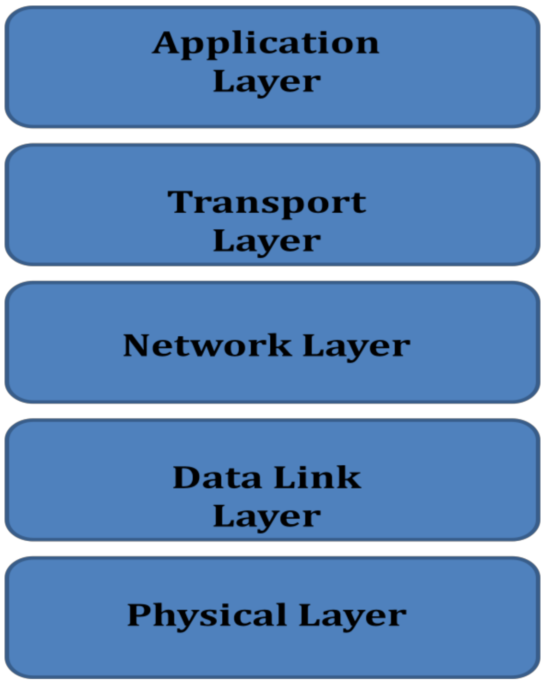


Fig I.3: Architectural layers of a WSN

The protocol stack of wireless sensor networks is organized into five main layers.

- ***Application layer:*** This layer defines a certain set of applications and services and primitives available and ready to be used by the programmer which can be implemented on various platforms.

- ***Transport layer:*** This layer is involved in maintaining the flow of data in order to meet the applications requirements, especially if the wireless sensor network needs to be accessed through external networks, for instance the Internet.
- ***Network layer:*** The aim of this layer is routing data and path selection in the packets exchange in the network.
- ***Data link layer:*** The main purpose of this layer is to manage the media access control (MAC) and to insure the multiplexing of the data streams.
- ***Physical layer:*** It represents the lowest layer in WSNs, and so is responsible of the data encryption, modulation and the frequency and power level selection.

I.3 Applications of WSNs

The WSNs operation consists basically of the monitored event detection and its reporting to the base station which will take the right decision and action according to the received information.

The deployment of WSNs is very interesting in many applications fields. Such networks are considered as highly promising and with an extremely great potential.

I.3.1 Military Applications

As for many other technologies, the military motivation has been a strong driving force at the beginning of wireless sensor development. Their easy deployment, self-organization, low cost, and fault tolerance are valued qualities in such an area.

The sensors are mainly used in unknown or difficult to access sectors before sending troops or as motion detector. They can be very effective in the detection and recognition of nuclear, biological and chemical attack.

I.3.2 Security Applications

A real time monitoring of the traffic or the energy distribution network can be ensured by the WSNs. This is very practical with the aircraft structure. The sensors are also used as alarm system to prevent structural alterations of the buildings, highways, bridges, etc. They can be incorporated to the construction, periodically activated, without power or wired connections, and be functioning for years and even decades.

I.3.3 Environmental Applications

These applications include the weather and environment conditions monitoring, the livestock or insects tracking and natural disasters alert system in case of fire or floods by giving give a damage assessment.

Sensors can be very helpful also in the pollution detection and its follow-up.

I.3.4 Commercial Applications

Manufacturing companies, through wireless sensor networks would be able to follow the production process from raw material to final delivered product. The sensors can be useful also in end-of-life management of products. They can give bring precious indicators in order to decide whether the product should be recycled or reused.

I.3.5 Medical and Veterinary Applications

Vital functions monitoring of a living organism could in future be facilitated and optimized by micro-sensors swallowed or implanted under the skin permitting a faster and more precise diagnosis of some diseases and patients' real time tracking in the hospital or from their houses.

I.4 Need for Mobility Support

Mobility management represents a main consideration in several emerging omnipresent and pervasive sensor network applications, especially in fields such as the health-care monitoring, the intelligent transportation systems and the industrial automation [4-6]. In most of these scenarios, mobile nodes are supposed to exchange data in real-time with a fixed-node infrastructure. For instance, in the clinical monitoring [7], patients carry embedded wireless sensing devices that are responsible of reporting data through a fixed wireless sensor network infrastructure. In this type of scenarios, it is essential to provide a constant and reliable stream of information.

Conclusion

In this first chapter we provided an overview on wireless sensor networks and presented its main applications. In the next chapter, we will focus on mobility in WSNs and its management characteristics in such networks.

CHAPTER II:

MOBILITY MANAGEMENT

Introduction

Mobility management is one of the new challenges in Wireless Sensor Networks. In the literature, most of protocol and algorithm designs consider stationary sensor nodes. Some of them allow physical mobility in terms of joining and leaving of nodes. However, the physical mobility which stands for the change of sensor node from one location to another was not considered. By advancing the technologies in WSNs and arising new applications, the requirements of the protocols changed. Nowadays, the mobility support has become one of the serious challenges in some applications such as clinical health-care monitoring, smart cities and industrial automation.

II.1 Mobility in WSN

II.1.1 Mobility Types in WSNs

The mobility in Wireless sensor networking can be classified into 4 basic types [8]:

- ***Stationary***: The node does not change its position and stays in the original location.
- ***Random***: The node moves with a random fashion. The motion may be followed within a probabilistic model characterized by parameters.
- ***Predictable***: The node's future location is known and cannot be modified.
- ***Controlled***: A user can control the node's position. There may be some constraints on the node's motion, mainly, the maximum speed or the maximum acceleration.

II.1.2 The Mobility Management Challenge

A naive solution in WSN applications for this mobility management is for sensor nodes to simply broadcast their data packets to all the neighboring access points within range. This approach has obviously numerous limitations. The major one is that broadcasting leads to redundant information at the reached APs, since several of them are receiving the same packets. This automatically implies that the fixed infrastructure will necessarily have to either waste huge resources in forwarding exactly the same information until the end point, or it will need very complex procedures and schemes, such as data fusion, to eliminate the duplicated packets locally.

A more operational solution is that the mobile node needs to be attached and connected to only one access point at any given time. The link between the mobile node and the access point must ensure a reliable quality of service and an optimized data exchange. When this communication is no more guaranteed, the mobile node needs to select a new access point.

II.1.3 Handoff Definition

The handoff, or handover, process is a capital issue in the implementation of the wireless networks concept. It defines the event where a mobile node performs the selection and the transfer of data from one serving AP to the "best" available access point. The selection criteria may depend on the terminal or the application requirements as well as the network status and the environment characteristics.

A handoff mechanism is basically the data packet destination address changing from one access point to a “better” other one. There are three main steps in the handoff process:

i. Monitoring the radio channel,

ii. Making a decision for handoff,

iii. Selection of the new AP

This procedure must take into consideration the specifications of the WSN considered.

II.2 Handoff Design Considerations in WSNs

Mobility management in wireless networks and more specifically WSNs is based on the handoff process. The issues considered when designing a suitable HO mechanism depend on the wireless network’s technology. Complexity, constraints and triggering criteria of this procedure are directly related to the expected performances of the network and its acceptable quality of service level.

The WSNs have very specific constraints that must be taken into account in the design of handoff mechanism.

II.2.1 Handoff Type

There are mainly two types of handoffs, Hard handoff and Soft handoff.

II.2.1.1 Soft Handoff

In Soft handoffs, also called *make-before-break*, the node’s radio devise must be able to use and sense multiple radio channels at the same time. This feature enables the mobile node to assess and compare neighboring access points link qualities without interrupting the data exchange with the current serving AP or disconnecting from it. This characteristic allows the mobile node to maintain the communication with its serving AP until it gets connected with the new selected one.

Soft handoffs are used for instance in code division multiple access, CDMA [9].

WSN nodes communication devise typically consists of low-power radio transceivers that can only operate on a single radio channel at a time, such as the widely used CC2420, see Appendix 3 for more details.

This constraint is the reason why Soft handoffs are not suitable to WSNs.

II.2.1.2 Hard Handoff

Hard handoffs, also known as *break-before-make*, do not allow communication with more than one AP at any given time. The mobile node has to be disconnected from its serving AP and then assess the others link qualities. Less complex than the Soft handoffs, Hard Handoffs introduce however an additional delay due to the selection of the “best” AP and the reconnection procedures after the mobile node’s disconnection from its previous serving AP. This delay must be minimized as it represents a disconnection period for the mobile node that may seriously affect the network performances and so the application in process. The most relevant challenge in Hard Handoffs is the optimization of this disconnection delay.

This type of handoff matches with most wireless sensors characteristics.

II.2.2 Links in WSNs

As already said, wireless sensors rely on low-voltage, low-power radio transceivers. In the other hand, such wireless networks are characterized by short coverage and unreliable links.

II.2.2.1 Short coverage

For Wireless sensor networks nodes, short coverage implies low density of reachable access points, contrary to cellular networks in which mobile nodes can commonly be within the range of tens of APs. The relatively big range in cellular networks permits to the mobile node to stay quite conservative with the quality of service requirements and to select a link with very high reliability.

In the case of WSNs, the deployment’s low density involves more relaxed and tolerant link quality requirements. In practice, the handoff parameters should be very carefully adjusted within the transitional and unreliable region.

II.2.2.2 Links High Variability

Link variability represents a particular concern in wireless sensor networks. In fact, a vast majority of them are working on the ISM radio bands, the same bands used by cordless telephones in the USA, Wi-Fi standards 802.1b, 802.11g and 802.11n, Bluetooth devices, microwaves, ZigBee / IEEE 802.15.4 wireless data networks, etc. We can notice then that the problem due to interferences and packet collisions is pervasive.

The links high variability and sudden fluctuations have an important impact in stability of WSNs. Actually, if not designed appropriately, handoffs may significantly degrade the network performances due to the ping-pong effect, especially when the mobile node moves in the frontiers of two neighboring APs.

II.2.2.3 Ping-Pong Effect

The ping pong effect consists basically in mobile nodes having unneeded and successive handoffs between two APs. This phenomenon occurs usually when moving at the frontiers of the APs coverage, where the serving AP's link quality is degrading and its assessment significantly affected by noise.

In order to minimize this ping-pong event redundancy, avoid useless disconnections, and in the other hand keep satisfying the link quality requirements the handoff mechanism metrics must be well calibrated. Taking into account these considerations preserve the communication stability and continuous data stream despite the variance of the wireless links.

II.3 Handoff Metrics

The decision of starting a handoff and selecting a new serving AP is governed by a number of metrics that are established in advance and can differ from one handoff algorithm to another.

In WSNs, these criteria depend on the link quality requirements, the power management and the CPU capacity.

II.3.1 The RSSI

The RSSI, which stands of the Received Signal Strength Indication, is the most used metric in handoff algorithms. The RSSI between one MN and one AP decreases when the distance between the two of them increases.

The received signal strength indication can be considered in two ways; absolute RSSI or relative RSSI. In absolute measurements, the algorithm bases its metrics only on the value of the RSSI obtained from one AP. In relative RSSI, the measurements are based on a comparison, or more precisely the ratio between couples of neighboring APs [10].

II.3.1.1 Thresholds

The decision to start a handoff may occur when the RSSI (R_a in the following figure) between the mobile node and its serving AP (AP_a) gets below a certain threshold level (T_1). For the selection of the new serving AP, the RSSI (R_b) between the mobile node and this new AP (AP_b) must be greater than a threshold (T_2).

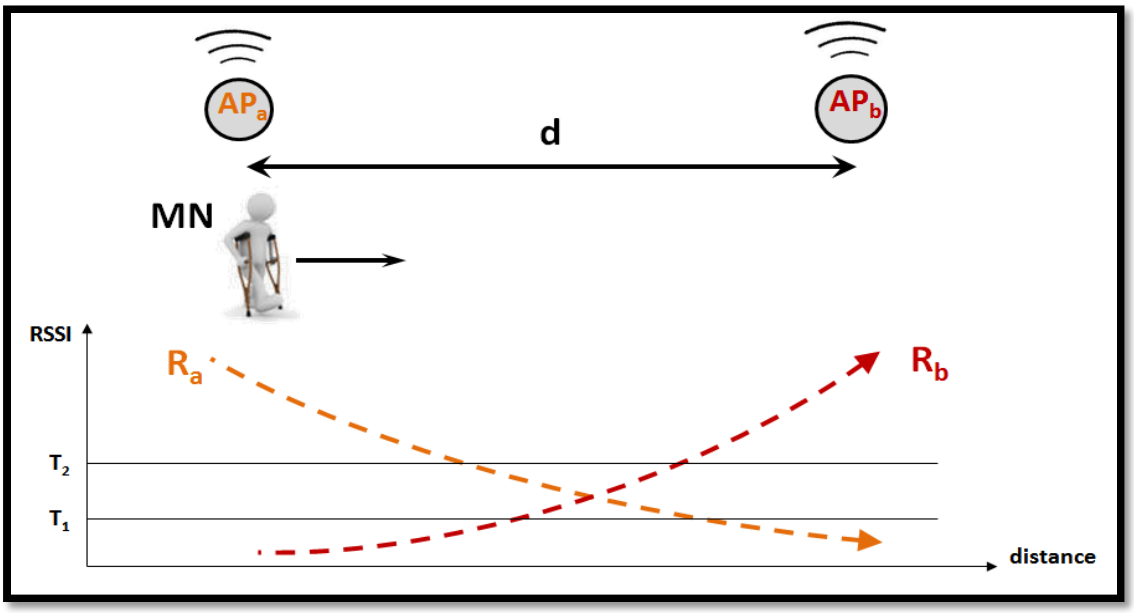


Fig II.1: Scenario considered

II.3.1.2 Hysteresis Margin

The hysteresis margin is basically the difference between the starting handoff threshold (T_1) and the end of handoff threshold (T_2). If the hysteresis margin is very low or equal to 0, in the APs coverage boundaries, even little links fluctuations can seriously affect the stability of the network due to the ping pong effect.

II.3.1.2.1 Constant Hysteresis Margin

In the vast majority of handoff algorithms, the hysteresis margin is fixed in advance, taking into account the bearable link quality degradation and the noise floor. This method is very simple but may be not optimized in some particular conditions where the link quality requirements are strict. The definitive selection of this HM will be a capital issue in this kind of algorithms however its value will probably not be satisfying in all environments.

II.3.1.2.2 Dynamic Hysteresis Margin

The hysteresis margin can also be variable. Indeed, some HO algorithms proposed a new strategy to calibrate and recalculate its value depending on the distance between the MN and the serving AP [11].

In this case, the HM will decrease when the distance increases with a certain trend. This procedure improves the HO performances in terms of delay, but requires that the mobile node has its localization information or at least knows its distance with the serving AP. This algorithm is complex and its process can be considered as hard to handle by wireless sensors CPU.

II.3.2 Distances

As we already saw with the case of dynamic HM, the distance may be a decisive criterion to initiate a handoff process. Numerous HO algorithms are based on distance as a main and direct metric like for the RSSI [12].

For wireless sensor networks mobile nodes, the location can be performed thanks to a GPS module. However, this cannot be afforded in all situations due to the cost and the power required for such devises. Another simpler method to estimate the distance between the MN and AP is the time difference of arrival signals from neighboring APs.

II.3.3 SNR

Signal-to-noise ratio is a compares the level of a received signal strength to the level of background noise. It is defined as the ratio of the first signal power to the noise power. When considering the SNR, algorithms just deal with this value instead of the RSSI. The noise is easily estimated by just assessing the noise floor periodically.

II.3.4 BER

BER stands from bit error rate. As its name suggests, this measurement represents an estimation of the links quality and the error ratio it induces. Methods to calculate this parameter are proposed but they have limited reliance and are still very complicated and complex, especially for wireless nodes characterized by their low computing power [13].

II.4 Handoff Performance Criteria

For all the handoff algorithms, there are common criteria to be optimized so that we can judge if the HO performances are satisfying or not. The most relevant challenge is to decrease the total number of unnecessary handoffs and reducing the ping pong effect. A good handoff mechanism should also distinguish necessary handoffs in time.

Furthermore, the size of handoff area has to be controlled and minimized. This region, called the transitional region is the area where the mobile node is the most likely meant to start a handoff. In this region is characterized by a low signal level and is sensitive to the noise floor and interferences.

The most important handoff performance criteria to be considered when designing a handoff algorithm are:

- Average number of handoffs,
- Average handoff delay,
- Standard deviation of handoff location,
- Average signal strength during handoff,
- Probability of link degradation.

II.5 Smart-HOP Design

The smart-HOP is a handoff algorithm especially designed for WSNs. Its conception highlights the importance of three parameters: link monitoring, hysteresis thresholds and stability monitoring. The figure below shows the time diagram of the smart-HOP mechanism [14].

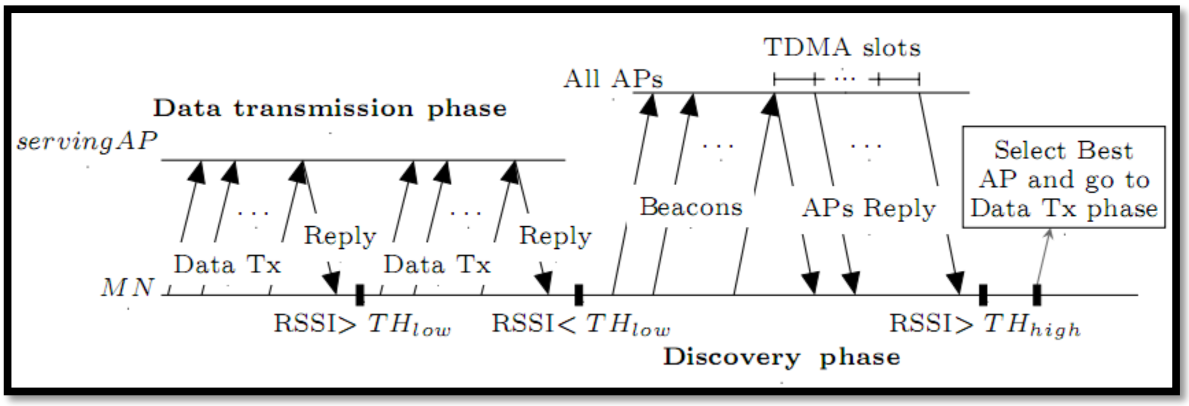


Fig II.2: Time Diagram of the smart-HOP Mechanism

Let's assume that initially the mobile is in the Data transmission phase. In this phase, the mobile node is attached to its serving AP and the link between them is reliable. The MN monitors the link quality level by receiving reply packets from its serving AP. In fact, upon receiving a number of n data packets in a given window w, the serving AP replies with the average received signal strength (RSSI), or the signal-to noise ratio (SNR), of the n packets. If it receives nothing, the AP takes obviously no action. The first important parameter to consider in the smart-HOP handoff process is to determine how many times the link monitoring should be done. The Window Size (*ws*) parameter represents the number of packets required to estimate the link quality.

The MN switches to the Discovery Phase when the link quality goes below T_1 and searchs for APs that are above the reliable threshold T_2. During this phase, the MN sends *ws* continuous beacons, and the neighboring reachable APs reply with the average RSSI or SNR of the beacons they receive. In order to reduce the packets collisions, the APs use a simple TDMA MAC. Regarding the high variability of wireless links, the MN may detect an AP that is, for a short time, above T_2 but the link quality may decline shortly after being chosen. In order to avoid this, it is important to assess the stability of the AP candidate. After detecting an AP above T_2, the MN sends *m* further *ws*-burst beacons to validate the stability of that AP. The value of *m* will be considered as the stability parameter.

Conclusion

In the second chapter, we described the mobility management new challenges in WSNs. We discussed also the handoff design considerations and constraints in such networks. The smart-HOP is one of a mechanism proposed to perform this operation. In order to predict the smart-HOP behavior in different conditions and to properly tune the settings of this algorithm, an analytical analysis and a simulation model will be developed in next chapter.

CHAPTER III:

SMART-HOP ANALYSIS

Introduction

In this phase, a thorough probabilistic analysis and simulation evaluations are conducted to study the impact of relevant parameters on the performance of handoff. The results of these analyses are guidelines for putting experiments with more rational assumptions.

For further evaluation of the proposed algorithm and investigate the functionality of the algorithm in various situations we conduct several analytical and simulation comparisons. The main goal in this stage is to find out the importance of handoff process. This would help to setup and perform more wisely experiments that consider the findings of preliminary experiments and the extensive analysis.

III.1 Analytical Analysis

We are extending the analytical and simulation model in order to get more intuition into an efficient handoff process.

III.1.1 System Model

A basic system consisting of two access points (APs) separated by a distance of d is considered in this work. It is assumed that the mobile node (MN) is moving along a straight line with a constant velocity $v=1\ m/s$ (close to normal human walk speed) between the two APs, labeled AP_a and AP_b – see Figure below. To model received signal strength (RSS), the log-normal shadowing path loss model is used which is given by:

$$RSSI(d) = P_t - PL(d_0) - 10\eta \log(d/d_0) - X_\delta$$

Where P_t indicates the transmitting power, $PL(d_0)$ is the path loss at $d_0=1meter$, η is the path loss factor, d is the distance between one MN and the AP expressed in meters which is directly proportional to the time slot number i as the velocity is constant and equal to 1. $X\delta$ is a log-normal variable with standard deviation of δ (in dB). In our model, D is the distance between the two APs.

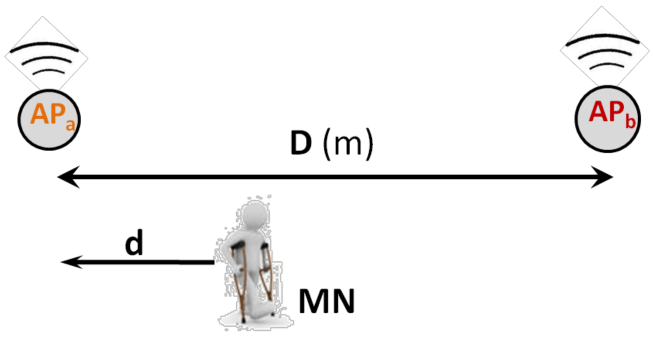


Fig III.1: System Model

To provide an insight and to make the solution tractable analytically, we assume that the shadow fading is not correlated. The probabilities of going below a predefined threshold level T and observing a high quality link is expressed as follows:

$$P(RSSI(i) < T) = Q(\frac{-T + RSSI(i)}{\delta})$$

Where Q( ) is the Q-function (complementary cumulative distribution function, i.e $Q(x)=\int_{x}^{\infty}\frac{1}{\sqrt{2\pi}}e^{t^2/t}\,dt$), Ra(i) and Rb(i) indicate the RSS mean values of APa and APb at time instant i. The two threshold levels for starting a handoff (T1) and ending a handoff process (T2) are depicted in the next figure.

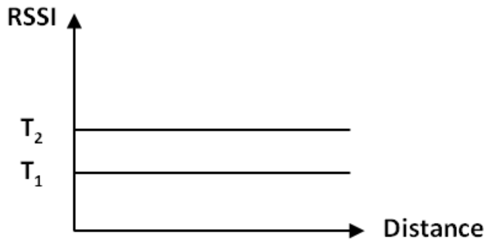


Fig III.2: The low and high threshold levels

III.1.2 Probabilistic Analysis

In this section, we will study the case were *ws* and the stability parameter *m* are both equal to 1.

To evaluate these metrics, it is required to define the chance of starting and ending a handoff process at each location. As explained earlier, WSNs only support hard handoff process which obliges the MN to communicate with a single AP at each time. The MN is initially connected to AP$_a$ –see Figure 2. It tracks the likelihood of the RSSI being above a threshold level, T$_1$, at each sampling interval. By observing the first slot with low link level, the handoff process starts. The primary assumption made for this analysis is that, the two parameters of window size and stability monitoring are ignored (*ws*=*m*=1). Adding these parameters enters various conditions which increases the equations complexity. The probability of starting a handoff at slot $s \in [2,k)$ is defined as follows:

$$P(Start(s))=\left[\prod_{i=1}^{s-1}P(R_a(i)>T_1)\right]\times P(R_a(s)<T_1)$$

The travelling path is separated into k samples. The first part of the equation indicates the observation of a number of slots with good/acceptable link quality level (above T_1). The second part denotes the observation of the low link quality for the first time (below T_1).

With the hard handoff process, the MN is able to communicate with one AP in each slot at each specific time. After disconnecting from AP_a at slot/location *s*, the MN assesses the other AP according to higher threshold level, T_2. The handoff finishes when the MN observes a high link quality (above T_2), as follows:

$$P(End(e)|Start(s)) = \left[\prod_{i=s+1}^{e-1} P(R_b(i) < T_2)\right] \times P(R_b(e) > T_2)$$

The probability of ending a handoff at a specific slot $e \in (2,k]$ is related to the time in which the handoff process has been started at *s*. The ending slot takes place in a location after occurrence of starting slot.

The time spent from the starting slot to the ending slot is called handoff period. We characterize the starting and ending moment of handoff with probabilities. The expected handoff delay is employed by the weighted sum of all possible handoff periods. It is defined as the product of the time spent in each possible handoff process by the respected probabilities for starting at slot *s* and ending at slot *e*.

$$E_{delay}(s) = \sum_{e=s+1}^{k} (e - s) \times P(End(e)|Start(s))$$

In order to show the ping-pong effect in Smart-HOP, a new term is defined which is called probability of restarting a handoff. This happens when a MN performs an unsuccessful handoff which causes redundant handoff or ping-pong effect. This imposes the MN to reconnect to AP_a again. The restarting of a handoff always occurs after successfully ending the first handoff at slot $r \in (2,k]$ as follows:

$$P(\mathrm{Re}\,start(r)|End(e)) = \left[\prod_{i=e+1}^{r-1} P(R_b(i) > T_1)\right] \times P(R_b(r) < T_1)$$

III.1.3 Impact of Parameters

The proposed probabilistic model derived in the previous section provides a general framework to evaluate the handoff process based on smart-HOP assumptions. In general, there are two set of variables, namely channel parameters and network parameters. In this section, we study impact of each parameter change on the performance metrics.

Due to the numerous mobile applications envisioned for a handoff process, the sensor node is required to work in different environments. The 2-tuples of environment evaluation are the path loss exponent, n, and the shadowing standard deviation, δ. The 3-tuple of network is set to ($ws=1,m=1,HM=5$).

III.1.3.1 Impact of Channel Parameters

We assume two APs located 10 meters far from each other, while the MN is moving from one side to the other in a one dimensional space. The reference values for performing all analysis are as follows:

P_t=0 dBm, d_0=1 m, $T1$=-90 dBm, HM=5 dBm, *data rate*=100 ms, *speed*=1 m/s, m=1 and ws=1.

III.1.3.1.1 Impact of the path loss exponent

In non-free space area, path loss exponent varies between 2 to 5 depending on the environment. Also depending on Coherence Bandwidth, path loss may vary for different frequencies in the signal [14]. The environmental change directly affects the path loss exponent. This parameter may be less dynamic in some applications with stationary sensors and static environment or may be highly variable in some other situations like mobile WSN applications [15].

First, we set the δ to 4 and assign n equal to values of 2 to 6. Figure 3 shows the impact of increasing the path loss exponent on the *RSS* and performance of handoff process. The next figure shows the impact of increasing the path loss exponent on the *RSS* and performance of handoff process.

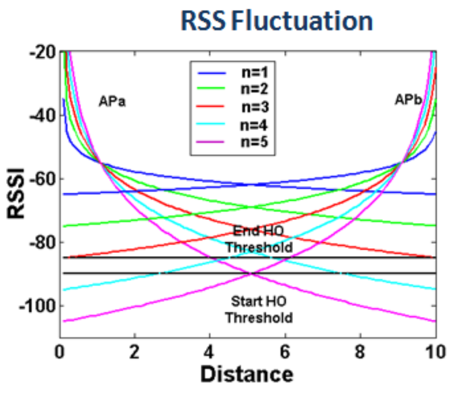


Fig III.3: RSS Fluctuation with various values of path loss exponent

This figure illustrates the variation of the *RSS* at AP_a and AP_b when the MN is moving from the location of AP_a toward AP_b. The horizontal lines show the threshold level for starting a handoff at -90 dBm and the threshold level for ending a handoff at -85 dBm. The larger the path loss exponent is the higher the slope of RSS reduction will be. It is apparent that this slight change has a huge impact on the RSS values. In smart-HOP design, we are aiming at choosing a handoff starting level below the intersection of received power from AP_a and AP_b. The ending level is supposed to be at this intersection or slightly higher than this point.

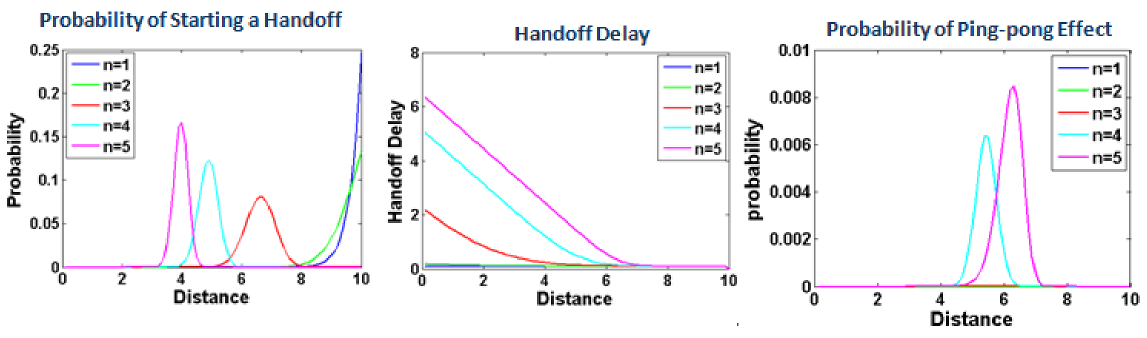


Fig III.4: Results obtained with various values of path loss exponent

The probability of starting a handoff process is shown in the left figure. The chance of starting a handoff increases when the RSS of the current AP goes below -90 dBm which is the preferred threshold level. The result indicates that with smaller path loss exponent, the longer the MN can move to reach an unreliable link situation.

The figure in the middle reports impact of path loss change on the expected handoff delay. The decreasing trend in all cases indicates that at the beginning stages of the MN trip, any disconnection from the current point of attachment till finding a good link at the other point takes longer. This is obvious since the received power is always stronger at closer distances of the APs. The larger path loss exponent imposes huge amount of handoff delay since the MN can just capture good links at very close distances to the neighboring AP.

Figure in the left shows the chance of performing a ping-pong effect at each time slot if and only if the MN had a successful handoff process before. A successful process stands for the cases where the MN enters into a Discovery phase, searches for new AP with better link quality, and finally associates with a new AP. But a careless parameter tuning may result in an inefficient process which causes a new handoff. The probability of ping-pong effect is higher when the MN is at highly unreliable link with AP_b which is more closer to the lower bound of the transitional region, -90 dBm. This region is farther to AP_b in environments with higher path loss exponent.

III.1.3.1.2 Impact of the Shadow Fading

The shadow fading is deviation of the attenuation affecting the signal level strength over the wireless propagation. In the figure below, we plot the RSS fluctuation with different values of the standard deviation of δ. We can clearly notice the greater δ is, the more important and significant are the oscillations of the RSS.

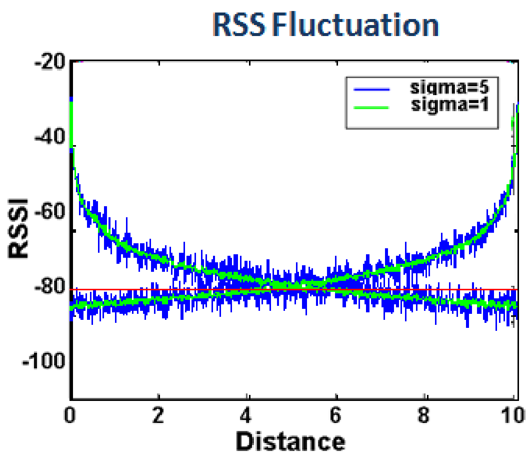


Fig III.5: RSS Fluctuation with various values of standard deviation

The amount of handoff delay is highly dependent on the intrinsic characteristic of starting probability and ending a handoff probability. The standard deviation compresses the signal with a spike as the scale parameters goes to smaller values – see next figures. On the other hand, an extended signal increases the chance of starting and ending a handoff at shorter period.

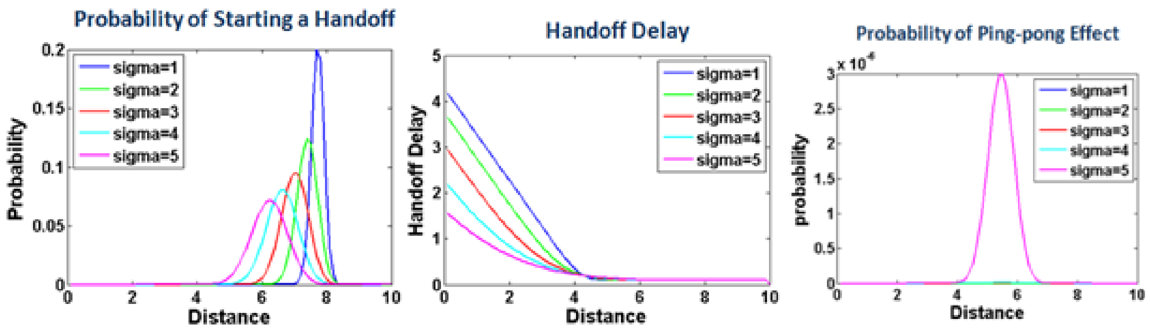


Fig III.6: Results obtained with various values of the standard deviation

The larger shadow fading imposes earlier moments of restarting a handoff and also increasing the chance of re-establishing a link by at most 3.0E-6 with large standard deviation compare to the least value with no possibility of ping-pong effect. This means that the handoff process is highly dependent on the environmental changes. However, the shadow fading value for indoor environments is not fluctuating too much and most of application designer set it to 4.

III.1.3.1.3 Impact of Speed and Data Rate

Here we can see some results when varying the speed and the data rate.

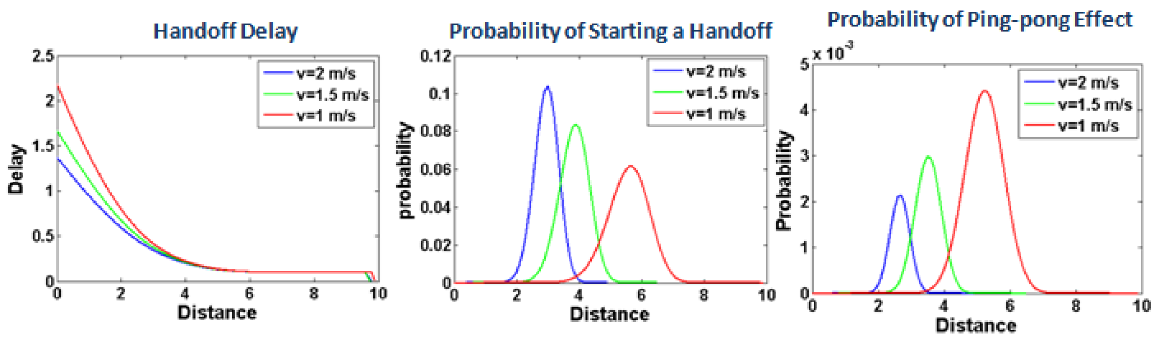


Fig III.7: Results obtained with various values of velocity

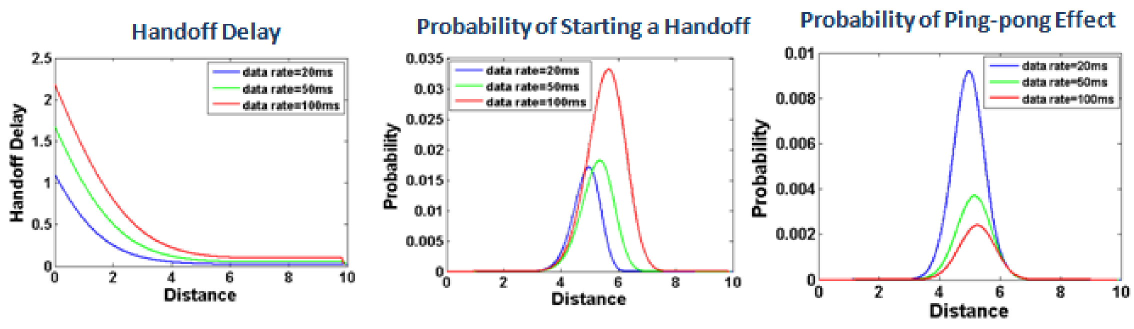


Fig III.8: Results obtained with various values of data rates

III.1.3.2 Impact of Network Parameters

There are three channel parameters: the Hysteresis Margin, *ws* and stability *m*.

III.1.3.2.1 Impact of Hysteresis Margin

Studying the impact of three network parameters involved in smart-HOP gives very interesting hints in calibrating a handoff process. Basically, the received signal is averaged over time using an averaging window known as *ws* to remove momentary fading due to geographical and environmental factors. The change of window size has a meaningful effect on the staring moment of a handoff since it directly varies the time spent for monitoring the current link with low quality – see figure below. A low window size imposes the MN to enter an assessment phase earlier. The ping-pong effect signal covers a wider region with higher probability since the lower window size reduces the accuracy of performing an efficient handoff process as shown in figures.

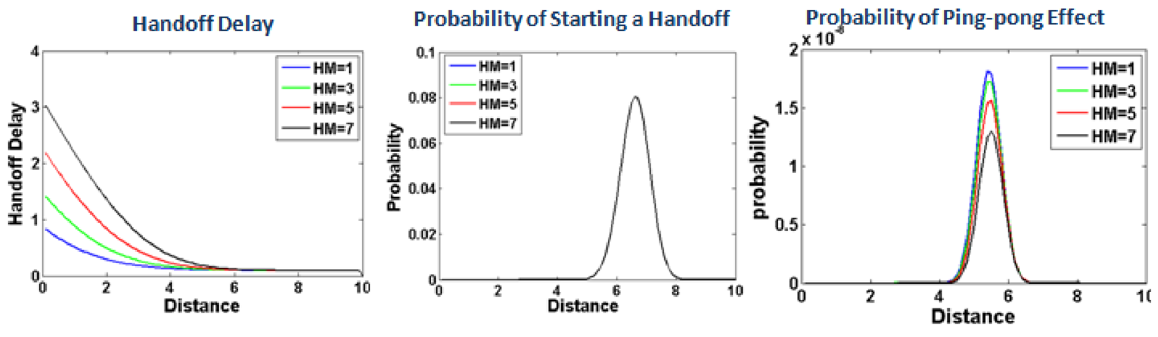


Fig III.9: Results obtained with various values of HM

The smart-HOP performances with the HM=5 are satisfying.

III.1.3.2.2 Impact of *ws* and *stability*

Encoding the events “good/acceptable link quality level (above T_1)” and “low link quality level (below T_1)” will be necessary for the needs of the following method.

“Good/acceptable link quality level (above T_1)” will be encoded by 0.

“Low link quality level (below T_1)” will be encoded by 1.

In the previous situation (*ws*=1) starting a handoff at slot $s \in [2,k)$ is equivalent to a sequence of s-1 successive “Good/acceptable link quality level (above T_1)” and a “Low link quality level (below T_1)” at slot s.

$$\underbrace{0\;0\;0\;\;\dots\;\;0}_{\text{s-1 first slots}}\quad\underbrace{1}_{\text{slot n° } s}$$

For a given position “s” and for WS=1 a unique sequence matches which makes the calculation of the probability simple, it will not be the case with *ws*=2 and *ws*=3 where multiple sequences are possible.

The probabilities being variable depending on slot position no general formula of the probability can be generated and the calculations must be realized individually for each position.

- For *ws*=2

Starting a handoff at s=2 is equivalent to the sequence: $\mathrm{Mat}_2 = 1\ 1$

Starting a handoff at s=3 is equivalent to the sequence: $\mathrm{Mat}_3 = 0\ 1\ 1$

To Start a handoff at s=4 multiple sequences are possible:

✓ Situation1: the sequence starts with a “good/acceptable link quality level (above T_1)”.

- ✓ Situation2: the sequence starts with a "Low link quality level (below T_1)" and followed by a "good/acceptable link quality level (above T_1)" (otherwise the handoff will start at s=2).

In situation 1: the rest of the sequence is completed by Mat_3 like following:

0, 0, 1 ,1

Situation 1, Mat_3

In situation 2: The rest of the sequence is completed by Mat_2.

0, 1 1, 1

Situation 2 Mat_2

The sequences can be arranged as a matrix Mat_4= 0 0 1 1

1 0 1 1

In general, to start handoff at position S=k the matrix of the possible sequences is given by:

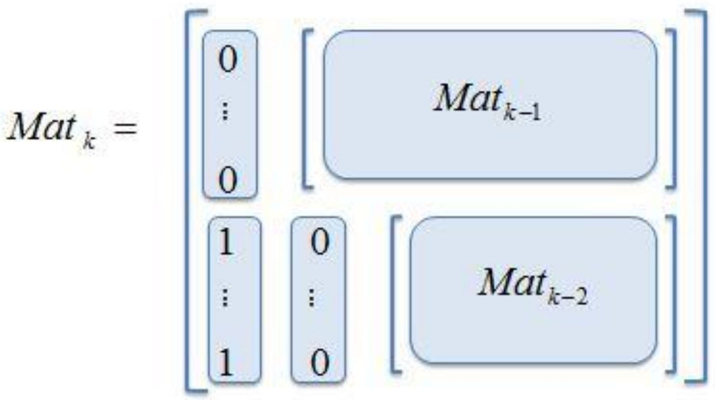


Fig III.10: Matrix generation for ws=2

So the probability to start a handoff at position s=k is given by:

$$\sum_{i=1}^{n}\prod_{j=1}^{k} P(R_a(j) > T_1) \times 1_{(Mat_k[i,j]=0)} + P(R_a(j) < T_1) \times 1_{(Mat_k[i,j]=1)}$$

- For *ws*=3

The same method is used, for every slot position we generate the matrix of the possible sequences that lead to start a handoff at this position.

The general form of the matrix is given by:

$$Mat_k = \left[\begin{array}{llll}
\boxed{\begin{matrix}0\\ \vdots\\ 0\end{matrix}} & \left[\boxed{\quad Mat_{k-1} \quad}\right] & & \\
\boxed{\begin{matrix}1\\ \vdots\\ 1\end{matrix}} & \boxed{\begin{matrix}0\\ \vdots\\ 0\end{matrix}} & \left[\boxed{\quad Mat_{k-2} \quad}\right] & \\
\boxed{\begin{matrix}1\\ \vdots\\ 1\end{matrix}} & \boxed{\begin{matrix}1\\ \vdots\\ 1\end{matrix}} & \boxed{\begin{matrix}0\\ \vdots\\ 0\end{matrix}} & \left[\boxed{\quad Mat_{k-3} \quad}\right]
\end{array}\right]$$

Fig III.11: Matrix generation for ws=3

And the probability to start a handoff at position S=k is still given by:

$$\sum_{i=1}^{n}\prod_{j=1}^{k} P(R_a(j)>T_1) \times 1_{(\mathrm{Mat_k}\,[\mathrm{i,j}]=0)} + P(R_a(j)<T_1) \times 1_{(\mathrm{Mat_k}\,[\mathrm{i,j}]=1)}$$

In practice, we cannot calculate the probabilities with this method. In fact, the size of the matrix increases exponentially and computing this method results is impossible, at least with Matlab how runs this until the slot s=17.

III.2 Simulation

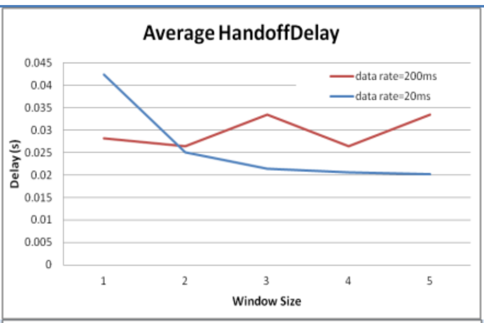
To explain the impact of window size and stability which were not considered in the probabilistic analysis, we performed a simple simulation. In this model, according to the sampling rate of MN, the travelling path has separated to a number of slots. According to the distance of MN to each neighboring AP, the RSSI is generated at that specific slot. In this scenario, we assume that initially at each slot the MN is connected to AP_a. By considering a sliding window and the low threshold for starting a handoff (T_1), the MN decides for the handoff starting slot. Then by having the RSSI value of AP_b and considering the stability parameter, the MN decides for ending the handoff. This process repeats for 10,000 trips and the results are getting averaged at the end of simulation.

III.2.1 Impact of *ws*

With Assumptions: data rate=0.2 (s), v=1 m/s, we run the simulation for 10,000 trips. The results are summarized in the following tables.

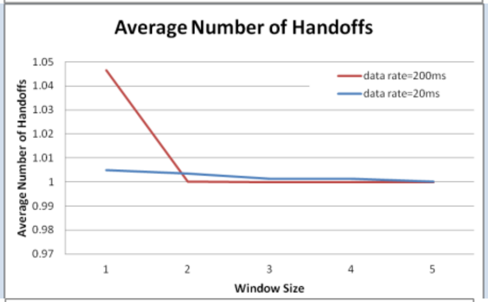
Average HO delay

ws	data rate=200ms	data rate=20ms
1	0.0282	0.0424
2	0.0264	0.025
3	0.0335	0.0214
4	0.0264	0.0206
5	**0.0335**	**0.0202**



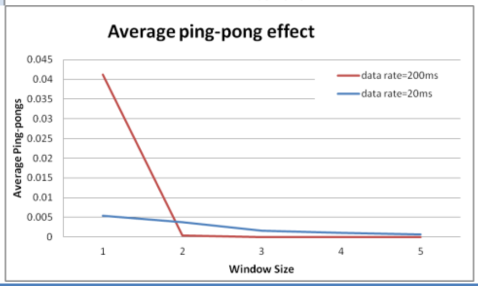
Average number of handoff

ws	data rate=200ms	data rate=20ms
1	1.0465	1.005
2	1.0002	1.0034
3	1	1.0014
4	1	1.0013
5	1	1.0002



Average ping-pong effect

ws	data rate=200ms	data rate=20ms
1	0.0412	0.0054
2	0.0004	0.0037
3	0	0.0016
4	0	0.0011
5	0	0.0007



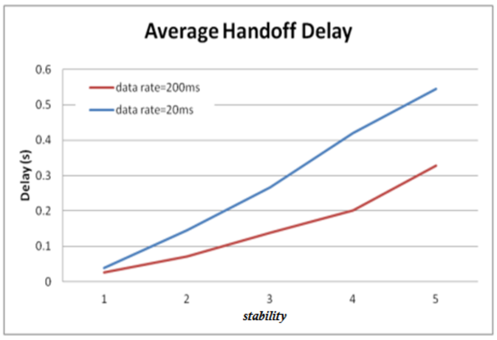
Tab III.1,2,3: Simulation results with different *ws* values

In theory, window size does not have considerable impact on the handoff delay. Assessing more than one packet reduces number of handoffs and ping-pong effect slightly. We conclude that in general window size is not a critical parameter for a handoff process. However, in practice we should assume a number of packets in order to compensate the sudden fluctuations of RSSI.

III.2.2 Impact of Stability

Average handoff delay in each run of simulation (total of 10,000 run)

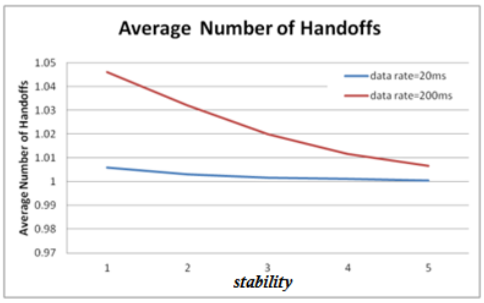
m	data rate=200ms	data rate=20ms
1	0.0269	0.0384
2	0.0711 (+62.2%)	0.1449 (73.5%)
3	0.1376 (+80.5%)	0.2663 (85.6%)
4	0.2016 (+86.7%)	0.4201 (90.9%)
5	0.3279 (+91.8%)	0.5461 (93%)



Tab III.4: Simulation results on handoff delay with different *m* values

Average number of handoff in each run of simulation (total of 10,000 run)

m	data rate=200ms	data rate=20ms
1	1.0461	1.0059
2	1.032 (-1.35%)	1.0031 (-0.28%)
3	1.0198 (-2.5%)	1.0016 (-0.43%)
4	1.0117 (-3.29%)	1.0011 (-0.48%)
5	1.0065 (-3.79%)	1.0004 (-0.55%)

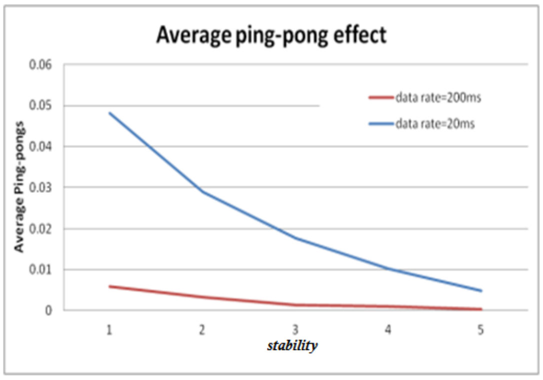


Tab III.5: Simulation Average Number of HO with different *m* values

Increasing the stability monitoring and hysteresis margin reduces the link variability. The results indicate that the impact of stability monitoring is much more than the hysteresis margin. It seems that considering a very small stability increases the handoff delay significantly. It is important to consider the ping-pong effect in these cases as well.

Average ping-pong effect

m	data rate=200ms	data rate=20ms
1	0.0058	0.0482
2	0.0032 (-44.8%)	0.0289 (-40%)
3	0.0014 (-75.8%)	0.0176 (-63.5%)
4	0.0011 (-81%)	0.0102 (-78.8%)
5	0.0004 (-93.1%)	0.0048 (-90%)



Tab III.6: Simulation results on Average ping-pong effect with different *m* values

The performances obtained with *ws*=3 and *m*=1 are satisfying according to this results.

Conclusion

In this chapter, thanks to the analytical analysis and simulation results, we decided that according to our application requirements and its fault tolerance, the HM, *ws* and *m* values were chosen. In the fourth chapter, we will give details on the implementation of the smart-HOP algorithm.

CHAPTER IV:

SMART-HOP IMPLEMENTATION AND TESTS

Introduction

After getting the results of both analytical and simulation models of the Smart-HOP algorithm, experiments will bring out some conclusions about the algorithm's behavior in realistic scenario.

In order to test the Smart-HOP mechanism, an implementation phase in TinyOS was carried out to extend the proposed code and modify it so to increase the protocol reliability in harsh environment and to make it support different practical scenarios.

IV.1 TinyOS

TinyOS [16] is an open source event-driven operating system especially designed for wireless sensor networks, its libraries and applications are written in the nesC language which will be described in the section below. TinyOS is designed to manage the operation of many types of mote devices as well as the sensors that are attached to them. TinyOS offers a networking stack that allows the motes to constitute an ad-hoc network.

As already said TinyOS has been built using nesC, it was built using numerous components and varied interfaces. This permits a lot of flexibility in the selection of different components like different networking modules that implement different protocols.

To customize the motes for several purposes, a single application can be implemented with the TinyOS structure. In many new mote versions the application can be changed remotely over the network after deployment. When compiling TinyOS, all the selected components, as well as custom components and the application are integrated into one single multi-threaded program.

The TinyOS executable consists mainly of two threads. The first one is used for the tasks execution and the other one is used to run the hardware event handlers. Tasks are basically functions which can be postponed. After they have been scheduled, the tasks run to completion, without pre-empting or blocking each other. The hardware event handlers are called if a hardware interrupt arises. These handlers again run to completion but are allowed to obstruct the execution of a task or any other hardware event handler.

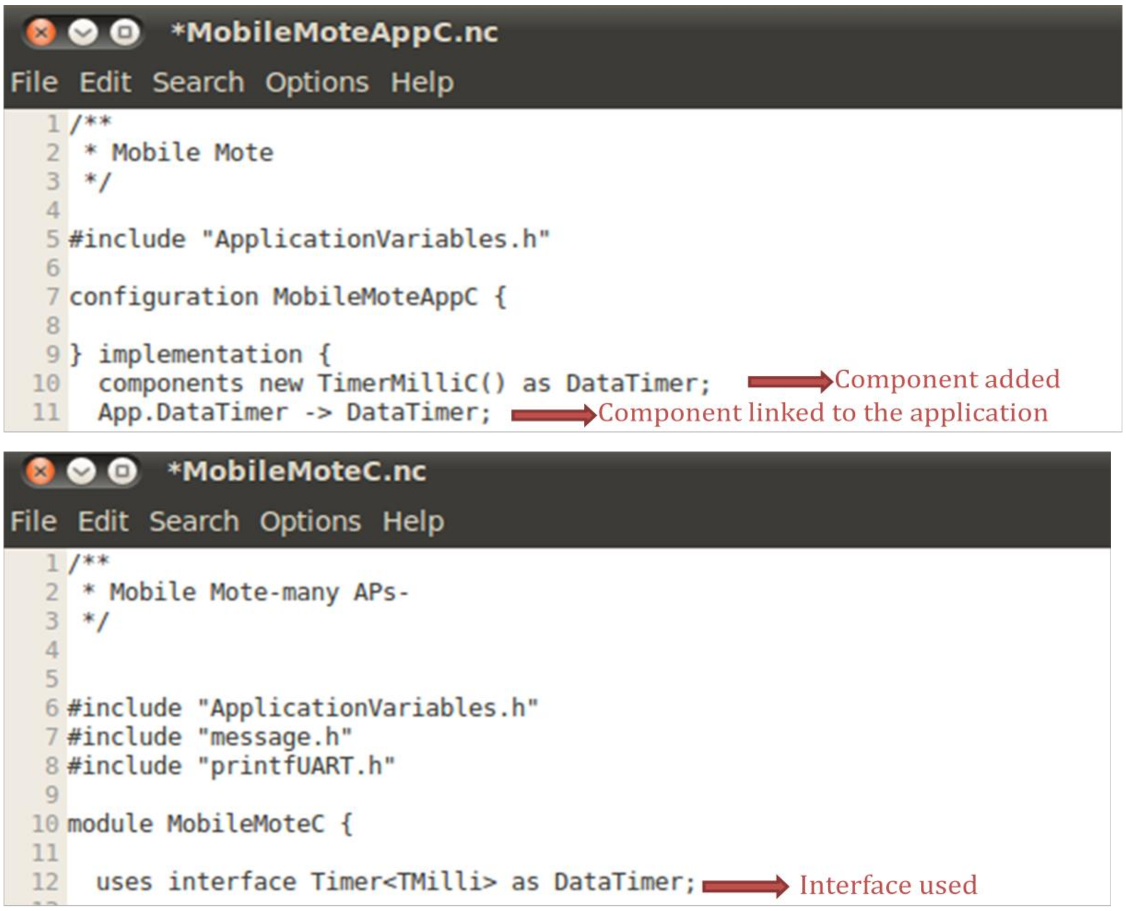
In the implementation, the used version is TinyOS 2.0.2.

VI.2 NesC

The nesC language [17] is an extension of the C programming language. It was designed to ease the implementation of the execution model and the structuring concepts of the TinyOS operating system. This programming language is created to be run on the operating system of embedded systems such as the sensor networks.

In nesC there is a clear separation between construction and composition. Programs are built out of various components which are linked together to form the whole program. The nesC components provide and use bidirectional interfaces which are the only way to access the component. An interface declares a set of commands which has to be implemented by the interface provider and a series of events which must be implemented by the user of each interface. In other words, if a component wishes to call commands in a specific interface it must first implement the events associated with that interface.

Fig IV.1 shows an example of creating DataTimer which is the period of data transmission. We create it similar to the TiimerMillic() which is a timer with precision in milliseconds. Then, it is attached to the application and eventually the interface is used in the configuration file.


*MobileMoteAppC.nc
File Edit Search Options Help

```
/**
 * Mobile Mote
 */

#include "ApplicationVariables.h"

configuration MobileMoteAppC {

} implementation {
  components new TimerMilliC() as DataTimer;
  App.DataTimer -> DataTimer;
```

Component added
Component linked to the application
*MobileMoteC.nc
File Edit Search Options Help

```
/**
 * Mobile Mote-many APs-
 */


#include "ApplicationVariables.h"
#include "message.h"
#include "printfUART.h"

module MobileMoteC {

  uses interface Timer<TMilli> as DataTimer;
```

Fig VI.1: An example of using component and interfaces in our smart-HOP code

The nesC language has two main types of components, the modules and the configurations. Modules contain the program code, which implements the defined interfaces to use. The Configurations are specially used to connect and link interfaces used by modules to the interfaces provided by other modules. Every application in nesC must have a top level configuration that wires the modules of that application together.

Concurrency is also supported in nesC, similar to TinyOS. Concurrency in nesC is based on run-to-completion tasks and can interrupt handlers.

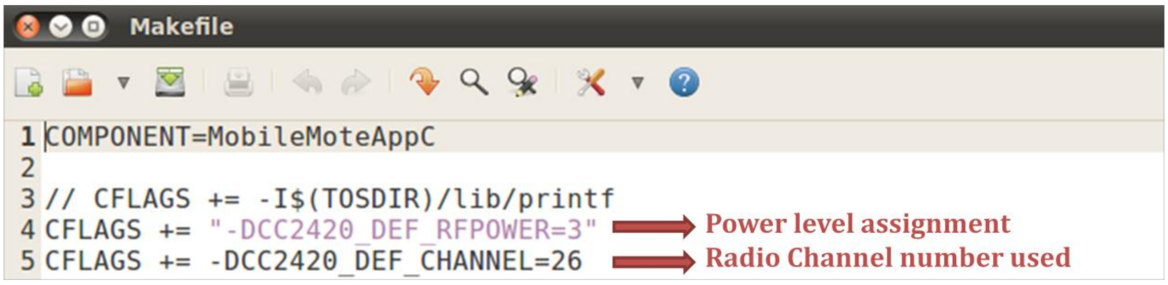
IV.3 Implementation

Smart-HOP is implemented in TinyOS 2.0.2. Since there are two major devices of MN and AP in the proposed model, there exist two algorithms developed on these sides. They had to be wisely managed in order to have a reliable interaction with each other.

IV.3.1 General Configuration

IV.3.1.1 General Settings

In our application, the general settings are stored in the *Makefile*. In principal, the *Makefile* keeps the radio settings of sensor nodes. The two major settings of each node are the transmission power level and the radio channel. In our experiments we used the minimum power level of 3=-25dBm, and the radio channel of 26.



```
COMPONENT=MobileMoteAppC

// CFLAGS += -I$(TOSDIR)/lib/printf
CFLAGS += "-DCC2420_DEF_RFPOWER=3"
CFLAGS += -DCC2420_DEF_CHANNEL=26
```

Fig IV.2: The radio settings in smart-HOP

According to CC2420 data sheet [18], the TelosB chip has 8 discrete power levels: 0, -1, -3, -5, -7, -10, -15 and -25 dBm. The corresponding values of "DCC2420_DEF_RFPOWER" are 31, 27, 23, 19, 15, 11, 7 and 3.

In 80.215.4 radio which works in 2.4 GHz bandwidth there are 15 possible channels (11-26) in which some of them (15, 20, 25, and 26) do not have interference from 802.11 which uses the same frequency[19]. It is important to consider that, the mobile nodes and the access points must be working in the same channel.

IV.3.1.2 Packets Format

In their communications, mobile nodes and access points exchange packets which have the same format.

Destination ID 16 bits	Source ID 16 bits	Status 8 bits	Sequence number 16 bits	Counte r 8 bits	Numbe r of HO 8 bits	RSSI calculated 16 bits

Fig VI.3: The packet format

1. ***Destination ID***: It contains the packet destination node's ID. In our conception, the access point nodes ID range is from 1 to 9. A mobile node ID is a multiple of 10.

2. ***Source ID***: It contains the packet source node's ID.

3. ***Status***: This field represents a flag informing if the mobile node involved in this packet is in Data Phase (*Status=1*) or in Beacon Phase (*Status=0*).

4. ***Sequence number***: This field corresponds to the sequence number of the current packet. The mobile node sends *ws* packets to the AP in each sequence, and then, increments this value by 1.

5. ***Counter***: These 8 bits value represents the packet's counter. Its value goes from 1 to *ws*. Each time the MN would send a packet from the same sequence, the MN increments this value by 1.

6. ***Number of HO***: Is stored in this field the total number of handoffs performed by the mobile node involved by this packet since its start-up.

7. ***RSSI calculated***: The AP puts here the average RSSI value assessed from the same sequence number of packets which were successfully received from the MN.

VI.3.2 Algorithm Basics

The mobile node has two main operating phases: Beacon Phase, and Data Phase.

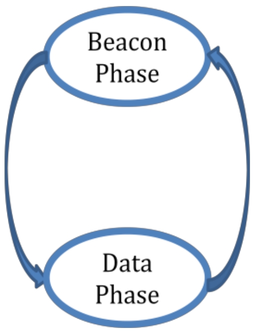


Fig IV.4: The two phases of smart-HOP

IV.3.2.1 Beacon Phase

At the start up of the experiments, the mobile node is in the Beacon Phase, this means that the MN is not connected to any access point. In this phase, the flag *Status* is set to 0. The following figure illustrates the packets exchange during this phase.

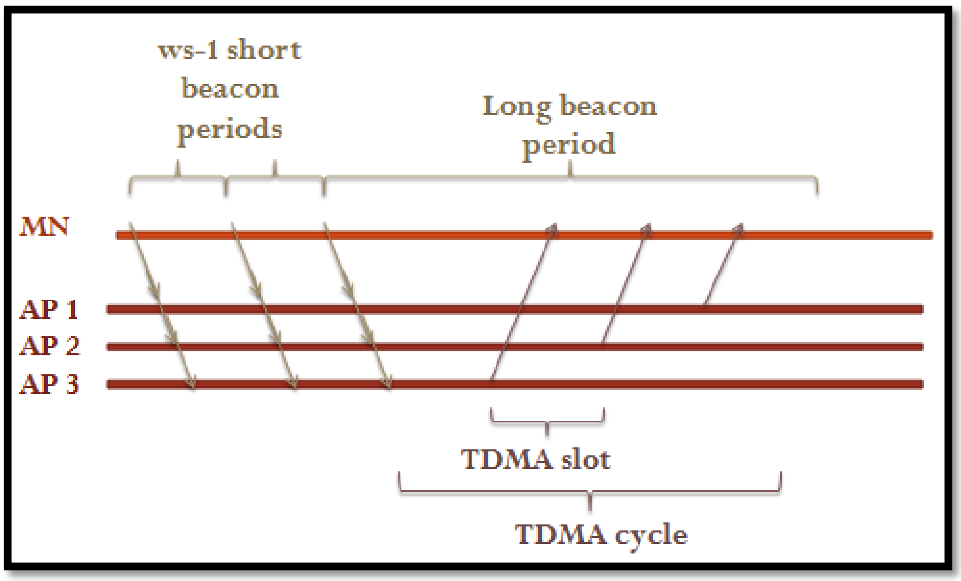


Fig IV.5: Beacon Phase beacon exchange

When the MN is in beacon phase, the mobile node broadcasts a new sequence (with a new *sequence number* value) of *ws* beacon packets (with *status*=0) to all the reachable APs each *short beacon period*. The first beacon packet of the sequence has its *counter* set to 1. This *counter* will be incremented in the next beacon packets to send until the value of *ws*.

The mobile node waits then during a *long beacon period* for the neighboring APs answers. The idea of having a longer period is to guarantee receiving all the reply packets sent from the neighboring APs.

The neighboring APs receive the broadcasted beacons. For each sequence of beacon packets, the AP calculates the average RSSI of the packets. The AP may not receive all the packets of the sequence because of the radio channel unreliability. This is a frequent problem in WSNs, this is why we introduced the variable *count*. In the AP side, we calculate the total number of the received packets corresponding to one sequence and store it in *count*. The average *RSSI calculated* for each sequence is the sum of the received sequence packets signal strength divided by the *count*'s value.

When the AP receives a packet from one MN with a new *sequence number*, a timer is started. This timer is stopped at the reception of a beacon packet from the same sequence and mobile node. If the AP receives a beacon packet with *counter* equal to *ws* or if the timer is fired, the average *RSSI* corresponding to this sequence is calculated. Here we can notice that the timer's aim is to solve the problem of the last sequence's packet loss.

The AP then waits for its time slot to send its reply packet in which the average *RSSI* is included to the requesting MN. For this, it has to wait for a period, called *waiting period*, equal to its ID multiplied by the time slot duration.

The MN receives the answers from the APs during the *long beacon period* which has to be longer than the *TDMA cycle* knowing that the TDMA cycle maximum value is:

$$\text{TDMA\_CYCLE} = \text{MAX\_AP\_ID} \times \text{TDMA\_SLOT}$$

If the stability parameter *m* is equal to 1, after gathering these replies, the MN selects the APs which send an *average RSSI* greater than T_2. If there is more than one AP that is satisfying this condition, the MN chooses the AP which has the greatest average *RSSI* and gets connected to this one and switches its *status* to the Data Phase.

If the stability parameter is greater than 1, this process must be repeated until we get *m* number of consecutive high quality link which is defined to be greater than T_2. The mobile node will then get attached to this one.

If the mobile node does not get any reply packet, or receives packets with an average *RSSI* less than T_2, it increments the *sequence number* and the described process is repeated.

IV.3.2.2 Data Phase

In Data Phase (*status* = 1), the MN is attached to one AP and the MN exchanges data packet with the selected AP only. The packets exchange in Data Phase is schematized is the following figure.

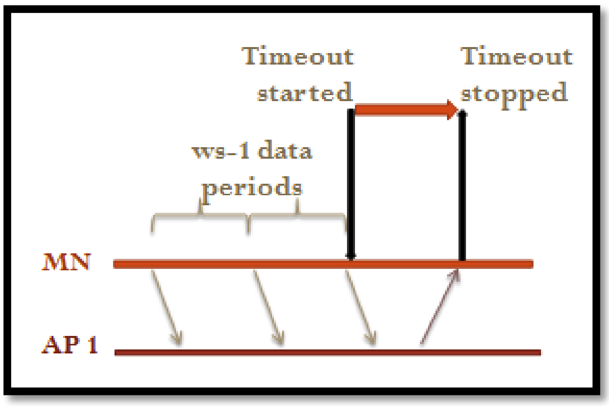


Fig VI.6: Data Phase packet exchange

The mechanism is quite similar to the beacon phase one. The difference here is that the data packets are not broadcasted but sent to the selected AP every *data period*. The *data period* is longer than the *beacon period*. The minimum period of data period is every 100ms to support getting all replies in a TDMA fashion. The AP replies with a packet in which it included the average *RSSI* calculated. If the average *RSSI* is less than T_1, then the MN will switch to the Beacon Phase.

After sending *ws* number of data packets, the mobile node starts a timeout. This timeout stops when the reception of the serving AP event occurs. If the MN does not receive any answer from its serving AP, the timeout will be fired and the MN will switch to the Beacon Phase again.

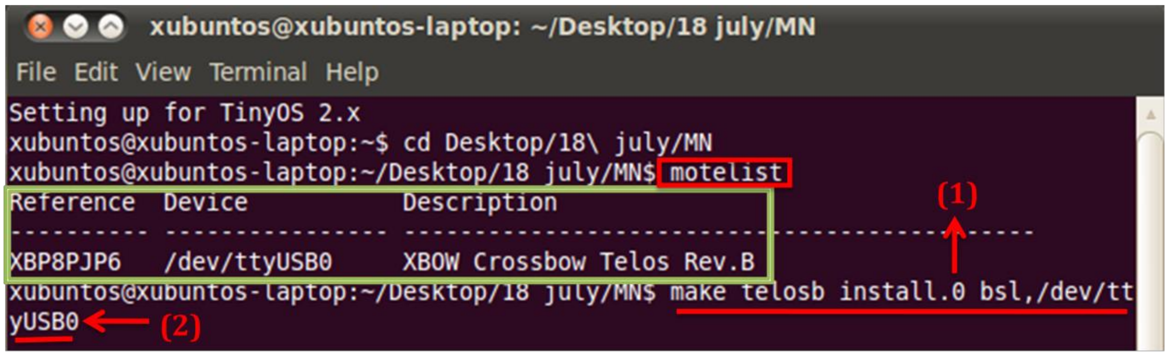
IV.3.3 Executing the TinyOS code

IV.3.3.1 Installation

The code folder is composed of a *makefile*, a header file, and the Application main code files. In the following figure, we see the steps to follow for checking the mote and installing the code.

First we check if the mote is correctly plugged into our USB port by using the command motelist. Once we have the confirmation, we can install our code in the specific port which the terminal shows.

The Linux command needed for this is in the figure below. We need to indicate the mote's ID (1) and the USB port to which our mote is connected (2).

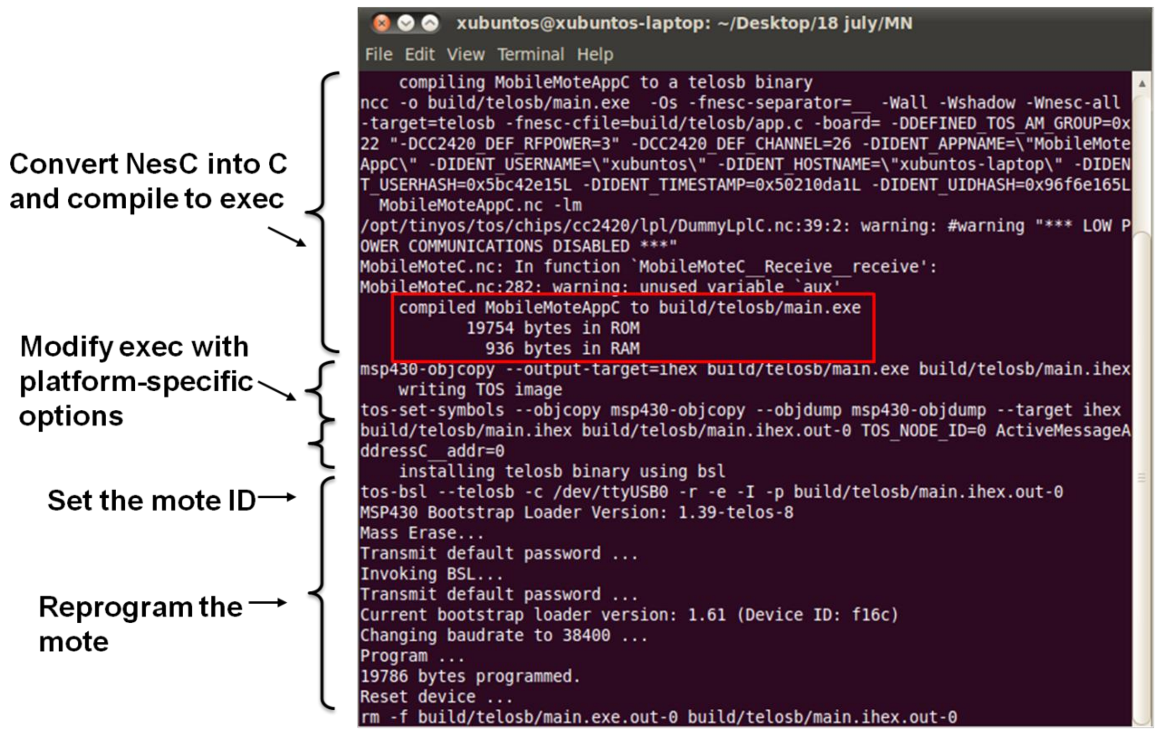

xubuntos@xubuntos-laptop: ~/Desktop/18 july/MN
File Edit View Terminal Help
(1)
(2)

```
Setting up for TinyOS 2.x
xubuntos@xubuntos-laptop:~$ cd Desktop/18\ july/MN
xubuntos@xubuntos-laptop:~/Desktop/18 july/MN$ motelist
Reference  Device           Description
---------- ---------------- ---------------------------------------------
XBP8PJP6   /dev/ttyUSB0     XBOW Crossbow Telos Rev.B
xubuntos@xubuntos-laptop:~/Desktop/18 july/MN$ make telosb install.0 bsl,/dev/tt
yUSB0
```

Fig IV.7: List of available motes and installing command

IV.3.3.2 Compiling

By running the installation command, the code will be executed. First, the nesC code is converted to a common C code. Then the platform specifications are added according to the settings in the *makefile*. After that the mote ID is set according to the command line. Finally, the executable file is generated and compiled on the sensor hardware.



Convert NesC into C and compile to exec
Modify exec with platform-specific options
Set the mote ID
Reprogram the mote
xubuntos@xubuntos-laptop: ~/Desktop/18 july/MN
File Edit View Terminal Help

```
    compiling MobileMoteAppC to a telosb binary
ncc -o build/telosb/main.exe  -Os -fnesc-separator=__ -Wall -Wshadow -Wnesc-all
-target=telosb -fnesc-cfile=build/telosb/app.c -board= -DDEFINED_TOS_AM_GROUP=0x
22 "-DCC2420_DEF_RFPOWER=3" -DCC2420_DEF_CHANNEL=26 -DIDENT_APPNAME=\"MobileMote
AppC\" -DIDENT_USERNAME=\"xubuntos\" -DIDENT_HOSTNAME=\"xubuntos-laptop\" -DIDEN
T_USERHASH=0x5bc42e15L -DIDENT_TIMESTAMP=0x50210da1L -DIDENT_UIDHASH=0x96f6e165L
  MobileMoteAppC.nc -lm
/opt/tinyos/tos/chips/cc2420/lpl/DummyLplC.nc:39:2: warning: #warning "*** LOW P
OWER COMMUNICATIONS DISABLED ***"
MobileMoteC.nc: In function `MobileMoteC__Receive__receive':
MobileMoteC.nc:282: warning: unused variable `aux'
    compiled MobileMoteAppC to build/telosb/main.exe
           19754 bytes in ROM
             936 bytes in RAM
msp430-objcopy --output-target=ihex build/telosb/main.exe build/telosb/main.ihex
    writing TOS image
tos-set-symbols --objcopy msp430-objcopy --objdump msp430-objdump --target ihex
build/telosb/main.ihex build/telosb/main.ihex.out-0 TOS_NODE_ID=0 ActiveMessageA
ddressC__addr=0
    installing telosb binary using bsl
tos-bsl --telosb -c /dev/ttyUSB0 -r -e -I -p build/telosb/main.ihex.out-0
MSP430 Bootstrap Loader Version: 1.39-telos-8
Mass Erase...
Transmit default password ...
Invoking BSL...
Transmit default password ...
Current bootstrap loader version: 1.61 (Device ID: f16c)
Changing baudrate to 38400 ...
Program ...
19786 bytes programmed.
Reset device ...
rm -f build/telosb/main.exe.out-0 build/telosb/main.ihex.out-0
```

Fig IV.8: Installing TinyOS code on a mote

The memory footprint of the MN and the AP are shown in the following table.

	ROM	**RAM**
Mobile Node	19754 bytes	936 bytes
Access Point	18720 bytes	810 bytes

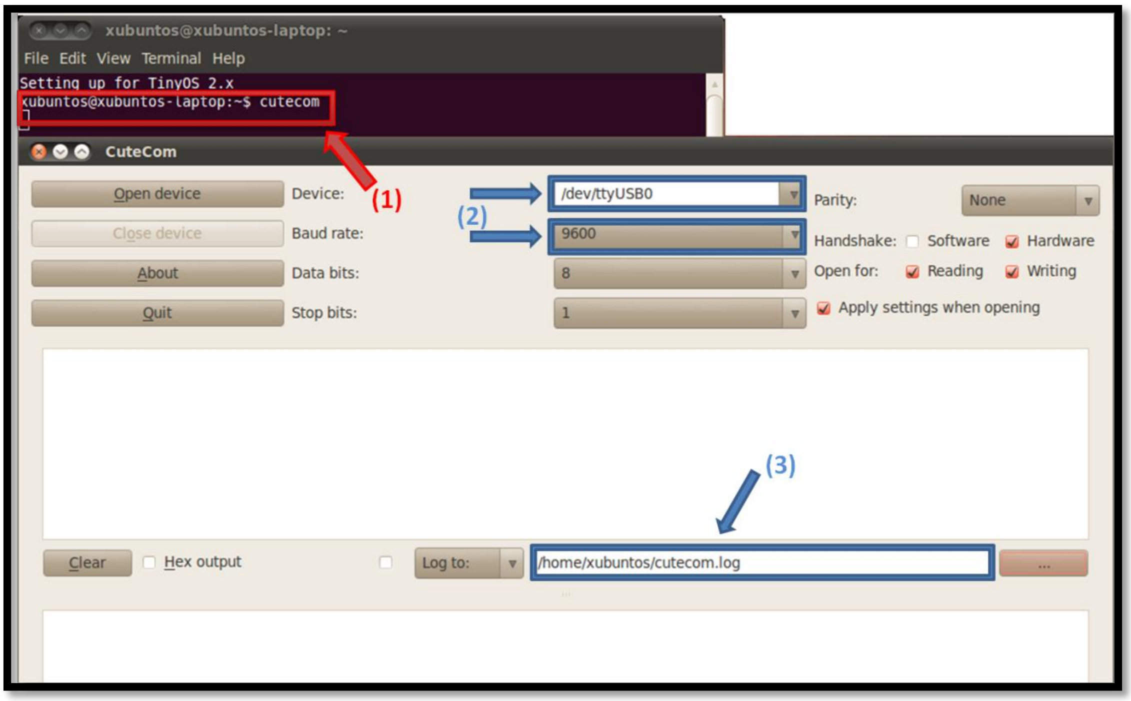
Tab IV.1: Memory footprint of MN and AP

VI.4 Implementation Results

In order to monitor our algorithm progress, we can use many tools to get online/offline log.

IV.4.1 CuteCom/GTKTerm/MiniCom

CuteCom is one of the graphical serial terminals, it is a free software distributed under the GNU General Public License Version 2 [20]. Cutecom allow us to follow the progress of the algorithm when the mote is plugged into a PC through the USB port. As shown in the figure below, to run Cutecom the Linux command is simply cutecom (1). Then the USB port must be specified with the right data rate (2). The file where we want to store the results is defined also in (3).



```
Setting up for TinyOS 2.x
xubuntos@xubuntos-laptop:~$ cutecom
```

Device: /dev/ttyUSB0
Baud rate: 9600
Data bits: 8
Stop bits: 1
Parity: None
Handshake: ☐ Software ✓ Hardware
Open for: ✓ Reading ✓ Writing
✓ Apply settings when opening
☐ Log to: /home/xubuntos/cutecom.log

Fig IV.9: The CuteCom window

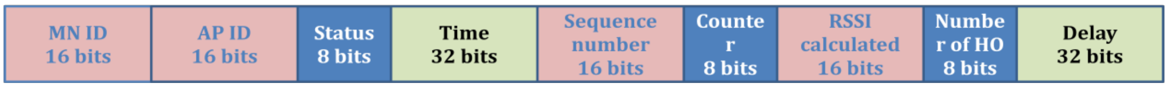
CuteCom will gather the information send by the command printfUART() like showed in the example below.

```
printfUART("RSSI2[%u]=%i\n", src, RSSI2[src]);
```

IV.4.2 Serial port logging/UART

There is another way to follow the progress of our algorithm, which is UART. A Universal Asynchronous Receiver/Transmitter, or simply UART, is a asynchronous receiver/transmitter, a hardware that translates data between parallel and serial forms.

In our mobile node code, we build the message to send through UART to our PC like showed in the figure below. This message contains the information gathered by the mobile mote each time it receives a packet from an access point. We connect the MN to a PC to collect all the data of the experiment at the MN side only.



MN ID 16 bits	AP ID 16 bits	Status 8 bits	Time 32 bits	Sequence number 16 bits	Counter 8 bits	RSSI calculated 16 bits	Number of HO 8 bits	Delay 32 bits

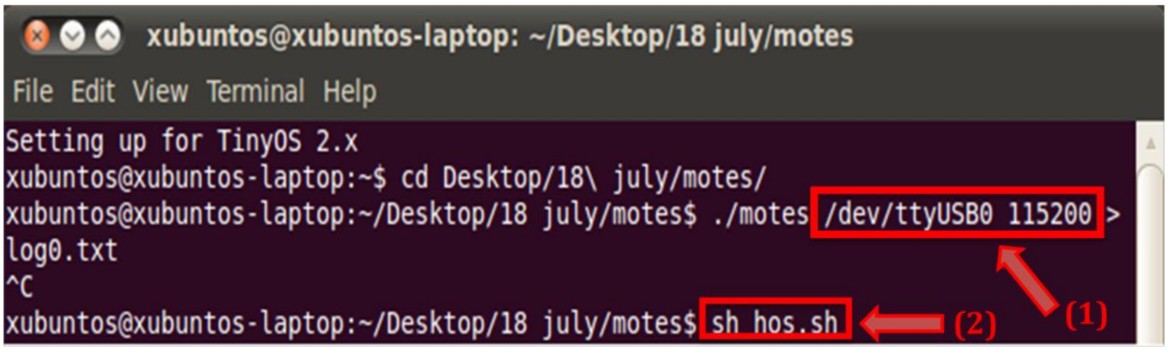
Fig IV.10: the packet payload sent through serial port

1. ***MN ID***: Mobile node's ID.
2. ***AP ID***: AP node's ID.
3. ***Status***: Status of the MN.
4. ***Time***: packet reception time.
5. ***Sequence number***: sequence number of the received packet from the AP.
6. ***Counter***: number of packets successfully received by the AP of this sequence number.
7. ***RSSI calculated***: RSSI average calculated by the AP and sent back to the MN.
8. ***Number of HO***: total number of HOs of the MN since it started up.
9. ***Delay***: HO delay of the last performed one (in milliseconds).

To send this message through UART, we have to call this task:

```
call UartSend.send[80](AM_BROADCAST_ADDR, &aux, sizeof(NeighMsg2)
```

We collect the received messages in the file called log0.txt by using the commands in the next figure (1). After that we execute a bash file in order to get the suitable information from the packet payload and generate the required handoff performances (2). You can find more details about this file in Appendix 3.


xubuntos@xubuntos-laptop: ~/Desktop/18 july/motes
File Edit View Terminal Help
(1)
(2)

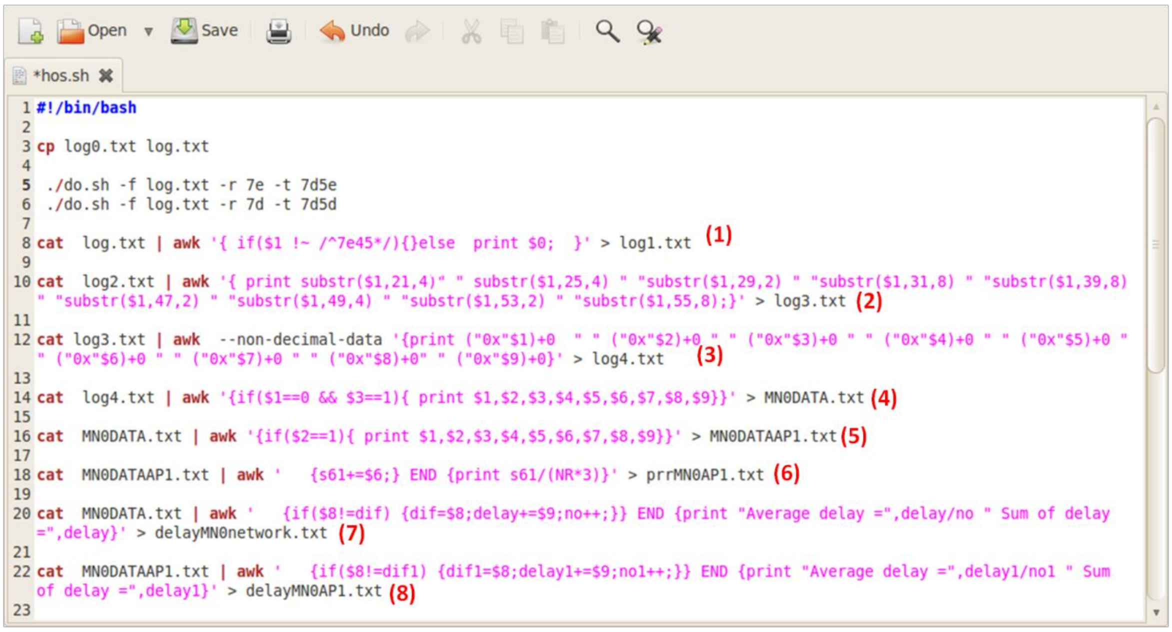
```
Setting up for TinyOS 2.x
xubuntos@xubuntos-laptop:~$ cd Desktop/18\ july/motes/
xubuntos@xubuntos-laptop:~/Desktop/18 july/motes$ ./motes /dev/ttyUSB0 115200 >
log0.txt
^C
xubuntos@xubuntos-laptop:~/Desktop/18 july/motes$ sh hos.sh
```

Fig IV.11: Collect log and analyze data

IV.4.3 Getting Results

By executing *hos.sh*, we can generate many files according to our needs. To get the suitable results from the Log file, various processes can be performed to come up with the final results. The data which is collected is in hex format. It can be analysed with different programs such as bash, awk, perl, C and many others. We have selected bash as the processing time is not very important for this experiment.

The Log file may contain bad packets which are not in our favor. The bash file is represented in the figure below. We initially know the packet payload and its header format. Hence, it is simple to filter out the useless information (1). Then, the header should be removed and the main information of the packet get separated in different columns (2). The hexadecimal format is then converted to decimal for further analysis (3).

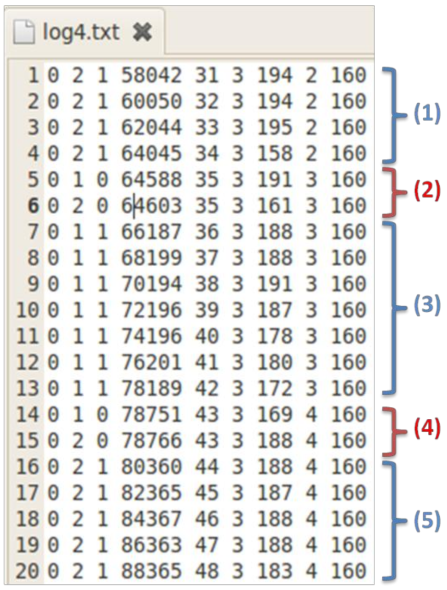


```
#!/bin/bash

cp log0.txt log.txt

 ./do.sh -f log.txt -r 7e -t 7d5e
 ./do.sh -f log.txt -r 7d -t 7d5d

cat  log.txt | awk '{ if($1 !~ /^7e45*/){}else  print $0;  }' > log1.txt (1)

cat  log2.txt | awk '{ print substr($1,21,4)" " substr($1,25,4) " "substr($1,29,2) " "substr($1,31,8) " "substr($1,39,8)
" "substr($1,47,2) " "substr($1,49,4) " "substr($1,53,2) " "substr($1,55,8);}' > log3.txt (2)

cat log3.txt | awk  --non-decimal-data '{print ("0x"$1)+0  " " ("0x"$2)+0 " " ("0x"$3)+0 " " ("0x"$4)+0 " " ("0x"$5)+0 "
" ("0x"$6)+0 " " ("0x"$7)+0 " " ("0x"$8)+0" " ("0x"$9)+0}' > log4.txt   (3)

cat  log4.txt | awk '{if($1==0 && $3==1){ print $1,$2,$3,$4,$5,$6,$7,$8,$9}}' > MN0DATA.txt (4)

cat  MN0DATA.txt | awk '{if($2==1){ print $1,$2,$3,$4,$5,$6,$7,$8,$9}}' > MN0DATAAP1.txt (5)

cat  MN0DATAAP1.txt | awk '   {s61+=$6;} END {print s61/(NR*3)}' > prrMN0AP1.txt (6)

cat  MN0DATA.txt | awk '   {if($8!=dif) {dif=$8;delay+=$9;no++;}} END {print "Average delay =",delay/no " Sum of delay
=",delay}' > delayMN0network.txt (7)

cat  MN0DATAAP1.txt | awk '   {if($8!=dif1) {dif1=$8;delay1+=$9;no1++;}} END {print "Average delay =",delay1/no1 " Sum
of delay =",delay1}' > delayMN0AP1.txt (8)

```

Fig IV.12: The bash file

Let's focus on the log4.txt file. The following figure illustrates the results obtained. Each line of the file represents one received message through UART. The columns correspond to the different fields of the received UART message format already described.

In our example, the MN which ID (column 1) is equal to 0 is attached (*status* = 1 at column 3) to the AP 2 (column 2). The *RSSI average* calculated and sent back to the MN is written in the column 7. When this value is less than our experimental T_1 (175), the mobile node switches to Beacon Phase (2). As we can see here, the mobile node receives two answer packets. The first one is from AP 1 and has an RSSI average equal to 191. The second packet is from AP2 and has an average RSSI equal to 161. The AP 1 is then selected and the MN gets attached to it (3).


log4.txt

```
0 2 1 58042 31 3 194 2 160
0 2 1 60050 32 3 194 2 160
0 2 1 62044 33 3 195 2 160
0 2 1 64045 34 3 158 2 160
0 1 0 64588 35 3 191 3 160
0 2 0 64603 35 3 161 3 160
0 1 1 66187 36 3 188 3 160
0 1 1 68199 37 3 188 3 160
0 1 1 70194 38 3 191 3 160
0 1 1 72196 39 3 187 3 160
0 1 1 74196 40 3 178 3 160
0 1 1 76201 41 3 180 3 160
0 1 1 78189 42 3 172 3 160
0 1 0 78751 43 3 169 4 160
0 2 0 78766 43 3 188 4 160
0 2 1 80360 44 3 188 4 160
0 2 1 82365 45 3 187 4 160
0 2 1 84367 46 3 188 4 160
0 2 1 86363 47 3 188 4 160
0 2 1 88365 48 3 183 4 160
```

Fig IV.13: A sample processed log file

For our performance analysis, we will need to separate the data packet from the beacon ones (4 in Fig IV.11) then split them in different files according to the serving AP they are received from (5 in Fig IV.11). These files allow us to calculate the PRR (6 in Fig IV.11), or the Packet Reception Rate in any serving AP, and the average delay of all the Handoffs performed in the whole network or according to each serving AP.

Conclusion

Smart-HOP algorithm implementation was successfully carried out. The code is dead lock free thanks to the different timers we introduced our conception. We tried in this code to consider and prevent most of the possible scenario that may occur especially regarding the link unreliability.

GENERAL CONCLUSION

My internship in CISTER represented my first work in a prestigious research center. I experienced the development of a handoff algorithm called Smart-HOP. I have contributed in enhancing this proposed protocol.

The main contributions are summarized as follows in four main steps.

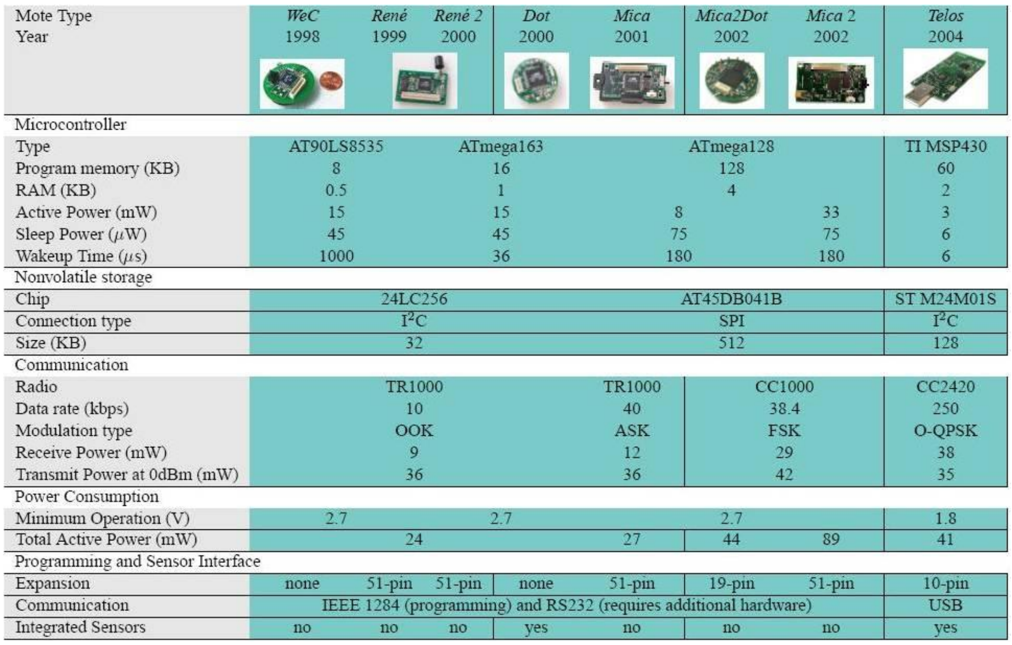
The first step was to develop an analytical model for the smart-HOP approach. The evaluation of this handoff performance is based on the average received signal strength by considering a threshold level for starting and ending a handoff. The system model consists of two access points separated by a distance while the mobile node is moving from one to another with a constant speed. The wireless channel is modeled according to WSN low power radio limitations. This approach allowed us to describe the best parameters in tuning the smart-hop to cope with the link variability.

The second step was to establish a simulation model with various window size and stability monitoring parameters. In order to get a realistic RSSI value, the white Gaussian random noise value at each sample was exploited. The simulation had been run in MATLAB for several times (100,000 times) and the data was getting from the averaged value. The simulation allowed us to vary the parameters and compare different scenarios.

Finally, we had to perform further experiments by considering more realistic scenario after getting the results of analytical and simulation model. The implementation was an extension to the Smart-HOP approach in TinyOS. The design of the algorithm was modified in order to support different scenarios with more number of mobile nodes and access points. It also considers defining of various timers to prevent failure cases and increase the protocol reliability in harsh environment.

APPENDICES

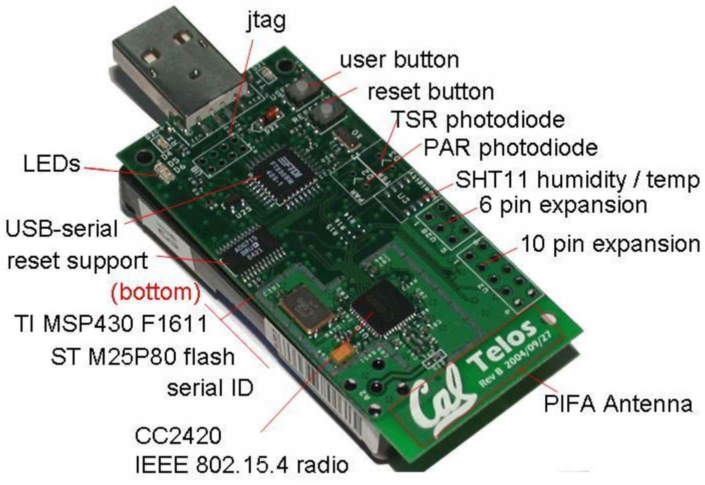
Appendix 1



Mote Type	*WeC*	*René*	*René 2*	*Dot*	*Mica*	*Mica2Dot*	*Mica 2*	*Telos*
Year	1998	1999	2000	2000	2001	2002	2002	2004
Microcontroller								
Type	AT90LS8535	AT90LS8535	ATmega163	ATmega163	ATmega128	ATmega128	ATmega128	TI MSP430
Program memory (KB)	8	8	16	16	128	128	128	60
RAM (KB)	0.5	0.5	1	1	4	4	4	2
Active Power (mW)	15	15	15	15	8	8	33	3
Sleep Power (μW)	45	45	45	45	75	75	75	6
Wakeup Time (μs)	1000	1000	36	36	180	180	180	6
Nonvolatile storage								
Chip	24LC256	24LC256	24LC256	24LC256	AT45DB041B	AT45DB041B	AT45DB041B	ST M24M01S
Connection type	I^2C	I^2C	I^2C	I^2C	SPI	SPI	SPI	I^2C
Size (KB)	32	32	32	32	512	512	512	128
Communication								
Radio	TR1000	TR1000	TR1000	TR1000	TR1000	CC1000	CC1000	CC2420
Data rate (kbps)	10	10	10	10	40	38.4	38.4	250
Modulation type	OOK	OOK	OOK	OOK	ASK	FSK	FSK	O-QPSK
Receive Power (mW)	9	9	9	9	12	29	29	38
Transmit Power at 0dBm (mW)	36	36	36	36	36	42	42	35
Power Consumption								
Minimum Operation (V)	2.7	2.7	2.7	2.7	2.7	2.7	2.7	1.8
Total Active Power (mW)	24	24	24	24	27	44	89	41
Programming and Sensor Interface								
Expansion	none	51-pin	51-pin	none	51-pin	19-pin	51-pin	10-pin
Communication	IEEE 1284 (programming) and RS232 (requires additional hardware)							USB
Integrated Sensors	no	no	no	yes	no	no	no	yes

UC Berkeley Family of Motes

Appendix 2



TelosB mote components

Appendix 3

CC2420 is true single-chip 2.4 GHz IEEE 802.15.4 compliant RF transceiver designed for low power and low voltage wireless applications. CC2420 includes a digital direct sequence spread spectrum baseband modem providing a spreading gain of 9 dB and an effective data rate of 250 kbps.

Parameters	**CC2420**
Frequency (Min)	2400MHz
Frequency (Max)	2483.5MHz
Device Type	Transceiver
Data Rate (Max) (kbps)	250
Operating Voltage (Min) (V)	2.1
Operating Voltage (Max) (V)	3.6
Programmable Output Power Ranging From (dBm)	-25 to 0

CC2420 characteristics

References

1. www.wikipedia.org.
2. John A. Stankovic, "Wireless Sensor Networks" , Department of Computer Science, University of Virginia, Charlottesville, Virginia , 2006.
3. Piergiuseppe Di Marco, "Protocol Design and Implementation for Wireless Sensor Networks", Masters' Degree Project, Stockholm, Sweden,2008.
4. Berta Carballido Villaverde, Susan Rea, and Dirk Pesch. "Inrout - a qos aware route selection algorithm for industrial wireless sensor networks. Ad Hoc Networks", 2011.
5. Hande Alemdar, Cem Ersoy, "Wireless sensor networks for healthcare: A survey", Computer Networks 54, pp. 2688–2710, 2010.
6. Hua Qin, Zi Li, Yanfei Wang, Xuejia Lu, Wensheng Zhang, and Guiling Wang. "An integrated network of roadside sensors and vehicles for driving safety: Concept, design and experiments" *IEEE PERCOM* 2010.
7. Octav Chipara, Chenyang Lu, Thomas C. Bailey, and Gruia-Catalin Roman. "Reliable clinical monitoring using wireless sensor networks: experiences in a step-down hospital unit." *SenSys*, 2010.
8. Ryo Sugihara ,"Controlled Mobility in Sensor Networks", UNIVERSITY OF CALIFORNIA, SAN DIEGO,2009.
9. Xinbing Wang, Shiguang Xie, and Xiuwen Hu, "Recursive analysis for soft handoff schemes in cdma cellular systems", *IEEE Trans*. *Wireless Communications*, 2009.
10. Ning Zhang and Jack M. Holtzman, "Analysis of Handoff Algorithms Using both Absolute and Relative Measurements" , *IEEE* , 1996.
11. Huamin Zhu , Kyung-sup Kwak, "Performance analysis of an adaptive handoff algorithm based on distance information" , Inha University, Republic of Korea, 2006.
12. Ken-Ichi Itoh, Soichi Watanabe, Jen-Shew Shih, and Takuro Sato, "Performance of Handoff Algorithm Based on Distance and RSSI Measurements" , IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, VOL. 51, NO. 6, 2002.
13. Jack M. Holtzman, "A Simple, Accurate Method to Calculate Spread-Spectrum Multiple-Access Error Probabilities" , IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. 40, NO. 3, 1992.
14. Hossein Fotouhi, Marco Zuniga, Mario Alves, Anis Koubaa, and Pedro Marron, "smart-HOP: a reliable handoff mechanism for mobile wireless sensor networks" , *EWSN* , 2012.
15. T. S. Rappaport, "Wireless Communications: Principle and Practice" New Jersey: Prentice Hall, 1996.
16. http://www.tinyos.net/tinyos-2.x/doc/.
17. "nesC: A Programming Language for Deeply Networked Systems", *UC Berkeley WEBS Project*, 2004.
18. http://inst.eecs.berkeley.edu/~cs150/Documents/CC2420.pdf.
19. Sinem Coleri Ergen ,"ZigBee/IEEE 802.15.4 Summary", 2004.
20. http://cutecom.sourceforge.net/COPYING.