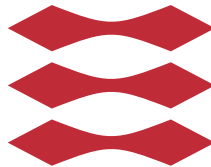


# Analysis of routing algorithms for Energy harvesting wireless sensor network

Negin Ostadabbasi

DTU



Kongens Lyngby 2013  
IMM-M.Sc.-2013-29

Technical University of Denmark  
Informatics and Mathematical Modelling  
Building 321, DK-2800 Kongens Lyngby, Denmark  
Phone +45 45253351, Fax +45 45882673  
[reception@imm.dtu.dk](mailto:reception@imm.dtu.dk)  
[www.imm.dtu.dk](http://www.imm.dtu.dk) IMM-M.Sc.-2013-29

# Abstract

---

The distribution of wireless sensor network field in modern life brings special requirements in routing protocols. The routing algorithms for being efficient in Wireless Sensor Network (**WSN**) should satisfy the features of energy consumption optimization and extension of network lifetime. A new class of **WSN** with the ability of harvesting environment power; is providing in recent decades. The objective of routing algorithms in *energy harvesting wireless sensor network* area is not to extend network's lifetime, but is to maximize the workload.

This thesis report a comprehensive survey on both energy-efficient and energy harvesting routing algorithms in **WSN** field. There are few Energy Harvesting Wireless Sensor Network (**EH-WSN**) routing algorithms that are mentioned in literature. Three of these algorithms are highlighted more in publication, therefore we choose to analyse these algorithms. In this thesis we have defined some analysis metrics and implemented a simulator which fully satisfy the requirement to test our candidate algorithms. The behavioural analysis of chosen algorithms in the case of different scenario are completely reported in this thesis.



# Preface

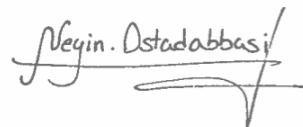
---

This thesis was prepared at the department of Informatics and Mathematical Modelling at the Technical University of Denmark in fulfilment of the requirements for acquiring an M.Sc. in Informatics.

The quality and durability of wireless sensor network improves by using energy harvesting technology. This technology works base on using environmental power instead of battery, so it gives opportunity to network to increase the system workload . To achieve this purpose an efficient routing protocol is needing. There are specific routing protocols base on energy harvesting idea.

To this end, this thesis consists of a study conducted to evaluate the functionality of some chosen **EH-WSN** routing algorithms in different simulation condition.

Lyngby 01-April-2013

A handwritten signature in black ink that reads "Negin Ostadabbasi". The signature is written in a cursive style with a horizontal line underneath the name.

Negin Ostadabbasi



# Acknowledgments

---

I would like to deliver my thesis to my parents and my sisters, that always being beside me and encourage me to improve. Maman, baba shoma behtarin hastin, asheghetoonam va har chi daram az shoma daram.

I would like to thank my boyfriend "Cisco" for encouraging me, helping and tolerating me in the last months. Without his supporting it is doubtful to finish this thesis; Ti amo 4e.

My special thanks to professor Nicola Dragoni for his excellent guidance, caring and patience. I would like to thank Xenofon Fafoutis and Alessio Di Mario for being company me in all the steps and patiently let me to improve in this topic.

Thanks to all of my friends which i spent good time with them in Italy and Denmark.





# Contents

---

<b>Abstract</b>	<b>i</b>
<b>Preface</b>	<b>iii</b>
<b>Acknowledgments</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Wireless Sensor Network . . . . .	1
1.1.1 Example of <b>WSN</b> . . . . .	2
1.1.2 Components of a WSN node . . . . .	3
1.1.3 WSN architecture . . . . .	3
1.1.4 MAC protocol in <b>WSN</b> . . . . .	5
1.1.5 Routing Protocols in <b>WSN</b> . . . . .	6
1.2 Energy Harvesting . . . . .	6
1.2.1 Components of a EH-WSN node . . . . .	7
1.2.2 MAC protocol in <b>EH-WSN</b> . . . . .	8
1.2.3 Routing Protocols in <b>EH-WSN</b> . . . . .	9
1.3 Motivation . . . . .	9
1.4 Contribution . . . . .	10
1.5 Thesis structure . . . . .	10
<b>2 Energy-Efficient Routing Protocols in WSN</b>	<b>13</b>
2.1 Routing Protocols background . . . . .	13
2.2 Energy-Efficient Routing Protocols . . . . .	14
2.3 Routing techniques in WSNs–Classification . . . . .	15
2.3.1 Network structure . . . . .	16
2.3.2 Communication model scheme . . . . .	20
2.3.3 Topology based scheme . . . . .	24
2.3.4 Reliable routing scheme . . . . .	25

<b>3</b>	<b>Routing protocols for EH-WSN</b>	<b>29</b>
3.1	EH-WSN routing algorithm overview . . . . .	29
3.2	Ford-Fulkerson Algorithm . . . . .	30
3.3	Energetic sustainability of routing algorithm for EH-WSN . . . . .	33
3.4	EH-WSN Routing algorithms . . . . .	35
3.4.1	Randomized Max-Flow ( <b>R-MF</b> ) . . . . .	36
3.4.2	Energy-opportunistic Weighted Minimum Energy ( <b>E-WME</b> )	36
3.4.3	Randomized Minimum Path Recovery Time ( <b>R-MPRT</b> )	37
3.4.4	Randomized minimum path energy ( <b>R-MPE</b> ) . . . . .	38
3.4.5	Energy Harvesting Opportunistic Routing Protocol ( <b>EHOR</b> )	39
3.4.6	Geographic Routing algorithm . . . . .	41
3.4.7	Distributed Energy Harvesting Aware Routing Algorithm ( <b>DEHAR</b> ) . . . . .	42
3.5	Evaluation and selection of routing protocols . . . . .	43
<b>4</b>	<b>Simulation</b>	<b>45</b>
4.1	Network model . . . . .	45
4.1.1	Node . . . . .	46
4.1.2	Zone . . . . .	46
4.1.3	Link . . . . .	48
4.2	Routing algorithm model . . . . .	48
4.2.1	Cost function . . . . .	49
4.2.2	Path cost . . . . .	51
4.3	Simulator Architecture . . . . .	51
4.3.1	Transceiver . . . . .	52
4.3.2	Simulation parameter . . . . .	53
4.3.3	Event flow . . . . .	55
<b>5</b>	<b>Simulation Runs</b>	<b>59</b>
5.1	Analysis metrics for simulation runs . . . . .	59
5.2	Simulation flow . . . . .	60
5.2.1	Different topology . . . . .	60
5.2.2	Different number of nodes . . . . .	65
5.2.3	Different traffic . . . . .	68
5.2.4	Different beacon rate . . . . .	75
<b>6</b>	<b>Conclusion</b>	<b>79</b>
<b>A</b>	<b>Code</b>	<b>81</b>
	<b>Acronyms</b>	<b>97</b>
	<b>Bibliography</b>	<b>99</b>

# List of Figures

---

1.1	Health-care . . . . .	2
1.2	<b>WSN</b> sensor node system architecture . . . . .	3
1.3	<b>WSN</b> layer . . . . .	4
1.4	<b>EH-WSN</b> vs Battery-powered <b>WSN</b> . . . . .	7
1.5	Components of <b>EH-WSN</b> node . . . . .	8
2.1	<b>OSI</b> levels . . . . .	14
2.2	Classification of routing protocols in <b>WSN</b> . . . . .	15
2.3	Flat routing . . . . .	16
2.4	TORA routing algorithm . . . . .	17
2.5	Cluster-based Hierarchical Model . . . . .	18
2.6	Time line showing Leach operation. . . . .	19
2.7	A three-level cluster hierarchy . . . . .	20
2.8	Directed_ Diffiusion. . . . .	21
2.9	SWE election process. . . . .	22
2.10	SPIN protocol . . . . .	23
2.11	The SPIN-BC protocol . . . . .	23
3.1	Ford-Fulkerson example . . . . .	32
4.1	Voronoi partitioning example . . . . .	47
4.2	Example of network model . . . . .	48
4.3	<b>WSN</b> Components . . . . .	52
4.4	CC2500 Transceiver . . . . .	53
4.5	Sense period . . . . .	54
5.1	Throughput of algorithms in different topologies . . . . .	61
5.2	Lifespan in different topologies with increasing number of nodes . . . . .	62

---

5.3	Example of good topology . . . . .	63
5.4	Example of bad topology . . . . .	64
5.5	Average throughput in different number of nodes . . . . .	65
5.6	E-WME algorithm . . . . .	66
5.7	Average lifespan in different number of nodes . . . . .	67
5.8	Average collision rate in different number of nodes . . . . .	68
5.9	Average throughput in different data traffic by changing A . . . . .	69
5.10	Average lifespan in different data traffic by changing A . . . . .	70
5.11	Average collision rate in different data traffic by changing A . . . . .	71
5.12	Average throughput of topology(17) in very low traffic . . . . .	72
5.13	Average lifespan of topology(17) in very low traffic . . . . .	72
5.14	Average collision-rate of topology(17) in very low traffic . . . . .	73
5.15	Average throughput in different data traffic by changing B . . . . .	74
5.16	Average lifespan in different data traffic by changing B . . . . .	74
5.17	Average collision rate in different data traffic by changing B . . . . .	75
5.18	Average throughput in different beacon rate . . . . .	76
5.19	Average collision-rate in different beacon rate . . . . .	76
5.20	Average lifespan in different beacon rate . . . . .	77

# Listings

---

4.1	Random Node placement . . . . .	46
4.2	Finding out the zone of each node . . . . .	47
4.3	Idle_Listing Cost . . . . .	49
4.4	TransmissionCost . . . . .	49
4.5	Packet energy Cost . . . . .	50
4.6	E-WME cost . . . . .	50
4.7	R-MPRT-org cost . . . . .	50
4.8	R-MPRT-mod cost . . . . .	51
4.9	Sense period . . . . .	54
4.10	Beacon period . . . . .	54
4.11	First beacon event time . . . . .	56
4.12	First data event time . . . . .	56
4.13	Harvest energy computation . . . . .	56
4.14	Available Energy computation . . . . .	56
A.1	BatchSimulation.m source code . . . . .	81
A.2	beacon.m source code . . . . .	82
A.3	colorGradient.m source code . . . . .	82
A.4	cost.m source code . . . . .	84
A.5	energy.m source code . . . . .	84
A.6	net.m source code . . . . .	85
A.7	packetenergy.m source code . . . . .	87
A.8	pathcost.m source code . . . . .	87
A.9	plotnet.m source code . . . . .	88
A.10	recursivepathcost.m source code . . . . .	89
A.11	routing.m source code . . . . .	89
A.12	sense.m source code . . . . .	90
A.13	simulator.m source code . . . . .	90



## CHAPTER 1

# Introduction

---

This chapter will explain the necessary background about the **WSN**, **EH-WSN**, and their routing protocols.

Current chapter also discuss about the idea of Medium Access Control (**MAC**) protocol in **WSN** and **EH-WSN**. Later this concepts will be used to build the simulator. The last part of this chapter point out the motivation behind this project and related work to achieve to goal.

## 1.1 Wireless Sensor Network

This section reports some general information about **WSN**, where this technology can be used, its components, architecture and routing algorithms.

The first **WSN** was designed and used in 70s, in military field during the Vietnam war. **WSN** consist of nodes, from few to several one, which work together to capture data from an environment region and send this data to a base station. These sensor nodes use to track and monitor heat, temperature, vibratory movement, etc. They are small with limited computing resources and base on a routing algorithm, they can transmit data to the user. This routing algorithm

depends on the network architecture and they can be changed. Since the sensor node have limited memory and they can be located in places which are hard to access, a wireless communication between nodes is needed. Because of this specific behavior, many routing and power management have been designed specially for **WSN**. As explained in paper [11], development of smart nodes have been researched in recent decades.

### 1.1.1 Example of WSN

There are different type of application for **WSN**:

- The most common one is *area monitoring*. In this scenario a **WSN** is distributed over a region which need to be monitored. The example of military belongs to this application.
- Another area of **WSN** usage can be *agriculture*. Many jobs can be done with **WSN**, like monitoring the gravity feed water and the pump can be controlled using wireless I/O device.
- The advancement of **WSN** gives new opportunities also in *health-care* system. In traditional method, a patient should visit a doctors in regular intervals and self-reporting experienced symptoms. But in smart home-care the **WSN** collects data in the base of physician's specification and provides continuous record to assist diagnosis. This method is also used for emergency situation and medicine reminder.

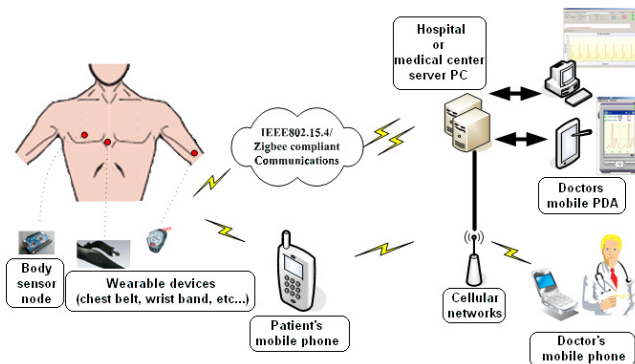


Figure 1.1: Health-care



### 1.1.2 Components of a WSN node

Each sensor node has different parts such as a radio transceiver with an internal antenna, a micro controller, an electronic circuit for interfacing with the sensors and energy source which is usually a battery or an embedded form of energy harvesting.

There are two types of **WSN**; structured and unstructured. An unstructured **WSN** contains a dense of sensor nodes which they connect to each other using an ad-hoc manner (sensor nodes randomly placed into the field.) For structured **WSN**, most of the sensors are located in a pre-planned manner.

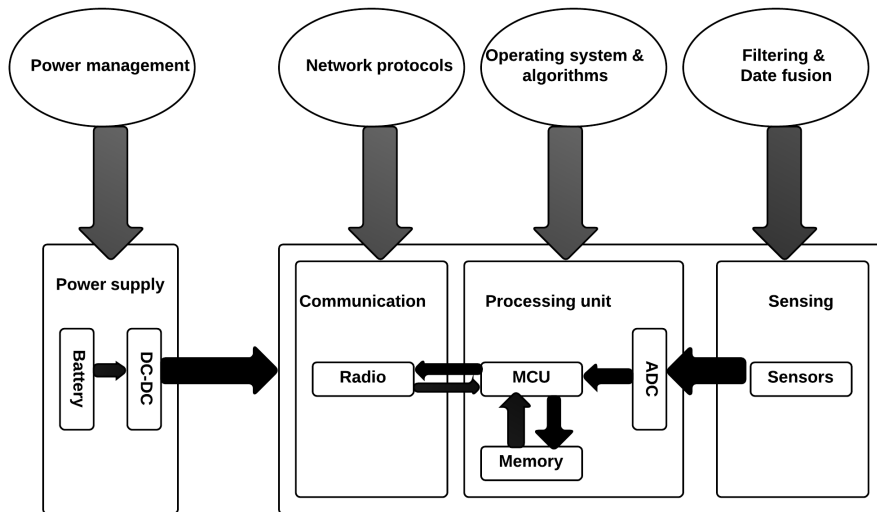
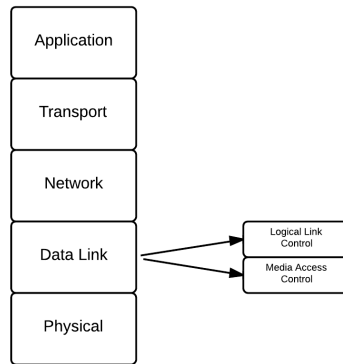


Figure 1.2: WSN sensor node system architecture

### 1.1.3 WSN architecture

The most common architecture of **WSN** is based on *OSI* Layer Model. *OSI* Layer Model is a creation defined by international organization for standards. *OSI* stands for *Open System Interconnection*. This model divides communication to seven layers. Which in **WSN** we need to analyze five layers: application layer, transport layer, data link layer and physical layer.



**Figure 1.3:** WSN layer

### 1.1.3.1 Physical Layer

This layer is the fundamental layer of network and it consists of networking hardware technologies.

This layer works as an electrical and a mechanical interface to the transmission medium. It is responsible for media and signal communication. In Open System Interconnection (**OSI**) architecture the physical layer translate the logical address that arrives from data link layer to the hardware specific operations.

### 1.1.3.2 Data Link Layer

The second layer of **OSI** Model is responsible for physical addressing and it provides functional resources for data broadcasting among networks. It also identify the errors of physical layer and tries to correct them. The other task of this layer is frame synchronization. The encoding and decoding of data into bits are the main functionality of this layer.

*OSI Data Link Layer* has two sub layers:

- *Media Access Control(MAC)* which is responsible for addressing and channel access control mechanism. It makes possible, for several nodes in a network, to communicate within a multiple access network.

The media access control is applied when one frame of data ends and the next one starts.

- *Logical Link Control(LLC)* layer is responsible for frame management and error checking. It provides multiplexing mechanisms that make it possible for several network protocols to transport over the same network medium.

### 1.1.3.3 Network Layer

**OSI Network Layer** is used for logically address the communications inside a virtual circuits, so it is used to transmit data from node to node and to determine the path that this data should follow. This layer offers routing and switching technologies. The error handling, packet sequencing, addressing and congestion control are the main functionality of Network layer. It also provides best quality of service on request of transport layer.

### 1.1.3.4 Transport Layer

The transport layer provides transparent transfer of data and providing reliable data transfer service to the upper layer. It also provides acknowledgment of the successful data transmission.

### 1.1.3.5 Application Layer

The **OSI** model define application layer as the user interface. It's responsible for displaying data and images to the user in a human-recognizable format, traffic management and provide software for different application that translate the data in an understandable form.

## 1.1.4 MAC protocol in WSN

As said before, this sub-layer of data link layer is responsible to give accessibility to nodes for communication in the network. In a **WSN**, due to the power limitation, the **MAC** protocol must allow very low duty cycle in order to guarantee the longer term sustainability of the system. Radio duty cycling introduces the problem of finding a moment where both transmitter and receiver are active so the connection can be started. Traditionally **MAC** protocol in **WSN** use synchronous or asynchronous protocols to solve the problem of connecting transmitter and receiver at the same time.

Some example of synchronous **MAC** protocols are *S-MAC*, *T-MAC* and *DSMAC* which are working on the base of synchronous clock.

In asynchronous protocol the establishment of a link between two nodes can be initiate by sender via preamble sampling like *B-MAC*, *X-MAC* and *RI-MAC* or by the receiver via beaconing.

### 1.1.5 Routing Protocols in WSN

As we described before one of the duty of network layer is the routing. This layer defines the most optimum path that packet should take from source to the destination.

Routing algorithm is a logic used to decide for each incoming packet that which output link should be chosen to transmit the data.

Routing algorithms can be classify in to two groups:

- **Static:** Routing decisions are fixed and nothing can affect on that like traffic load or network topology.
- **Dynamic:** Routing decision depends on network topology and traffic load.

In **WSN** one of the challenge is the energy consumption problem, since it is not feasible to recharge the limited battery after depletion. So when we want to choose routing algorithm for our network, we should take into account to choose the energy-efficient one.

There are several routing algorithms which support the idea of energy-efficiency, these will be explained in chapter 2.

## 1.2 Energy Harvesting

This section is explaining the meaning of Energy Harvesting Wireless Sensor Network (**EH-WSN**) and in which point is different from a normal **WSN**. It also discuss about the component of this network and how **EH-WSN** routing protocols works.

Energy Harvesting wireless sensor network(EH-WSN) is a kind of **WSN** that uses rechargeable power supply instead of using traditional battery.

In traditional **WSN**, the energy sources is limited. When the power source of node finish, it can not continue to work unless it is recharged again. So the effort was to design energy-efficient network protocols to maximize the lifetime of **WSN** by minimizing energy usage.

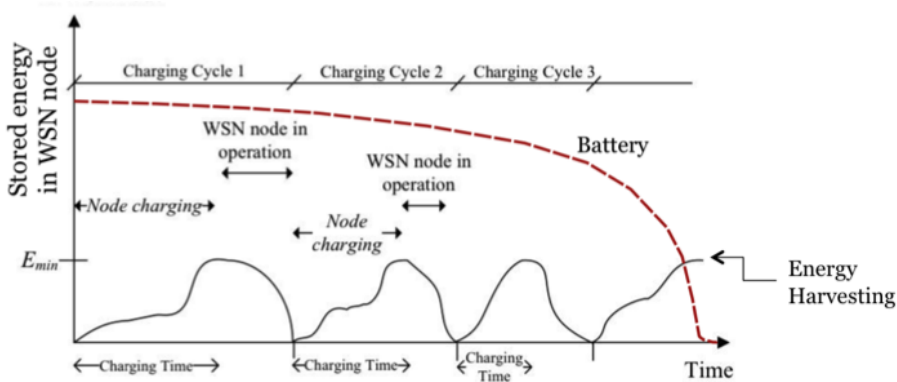


Figure 1.4: EH-WSN vs Battery-powered WSN [1]

As shown in [1] if we can have access to unlimited power, we can have infinite lifetime in network. This unlimited power can be provide by environment such as light, vibration and heat. Then stores the harvested energy in a storage device. When the device uses energy harvested instead of battery, the residual energy is no more an useful quantity to preserve. In **EH-WSN**, if the rate of harvesting power is lesser than the power used by the node, the sensor node should go to sleep to charge up.

### 1.2.1 Components of a EH-WSN node

Each **EH-WSN** node uses one or more energy harvesting devices to harvest environmental energy. The **EH-WSN** is composed by different components as it is shown in Figure 1.5.

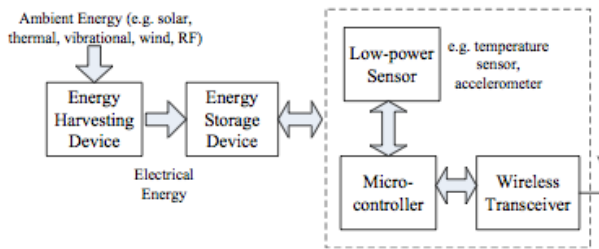


Figure 1.5: Components of **EH-WSN** node [2]

As we know, the energy needed for sensors should be electrical. So the first step before start to work is to convert environmental energy to electrical one. Inside the node there are different components for sensing the area, collecting data and processing the data. So the main different part of this kind of nodes, in compare of **WSN** nodes, is the power part. Once the stored energy reach to a certain amount, the power supply for micro controller and transceiver will start to work. They will continue to work until they have energy, as soon as the energy finish, they stops and energy storage device start to work to save energy again.

The other fact should be consider about **EH-WSN** node is that the available energy can change in different nodes. So every node has its own harvesting rate.

### 1.2.2 MAC protocol in EH-WSN

As explained in subsection 1.1.4 there are *synchronous* and *asynchronous* **MAC** protocol. In **EH-WSN** area using the synchronous **MAC** protocol is not possible, because by definition of **EH-WSN**, different nodes have different harvesting energy so they can not have an equal duty cycle required by synchronous protocol. Within the asynchronous approach the receiver-initiated methodology has proven to be more energy efficient compared to the sender-initiate [12].

In receiver-initiated **MAC** protocol, receiver periodically wakes up and sense the channel, if the channel is free, transmits the beacon. After sending the beacon, receivers, with predetermined period, listen to the channel. At the same time, when the transmitter has some data to send, it goes into an active state and listen for receiving the beacon. After receiving the beacon the sender transmit the data and waits for another beacon which is the acknowledges of the data reception. If the receiver doesn't receive any data it goes into sleep state.

The receiver-initiated significantly reduce the amount of time which channel is busy, so it let other nodes to connect to each other. As a result the throughput of the network increase. Another good point of using receiver-imitated is the efficient collision detection, because the channel access is controlled mainly by the receiver.

As a conclusion in **EH-WSN** area using the asynchronous MAC protocol is energy efficient.

### 1.2.3 Routing Protocols in EH-WSN

We talked about Routing Protocols in **WSN**, now routing algorithms for **EH-WSN** network are presented, it is better to know that different nodes harvest different amount of energy. So their duty cycle is not the same.

In this kind of network the goal is maximizing throughput since the power of network can be replenish. But at the same time we should notice that environmental power is limited. So we need a routing algorithm aware of environmental condition. One solution is to combine the replenishment rate in to cost metric and compute the routes. This idea will be used later in some part of my thesis.

As it also discussed in [1], since the harvesting rate of nodes is different, predicting the wake-up time of nodes is impossible, it is challenging to be sure that the next-hope is awake. If the next-hope is not awake the best solution is to use broadcasting and opportunistic forwarding, which means finding another volunteer to forward the packet to them. There are some specific Routing algorithms base on the idea of **EH-WSN** which will discuss in chapter 3.

## 1.3 Motivation

Wireless Sensor Network has been designed to monitor physical or environmental condition and many research works has been done on this topic. But power supply in this model of network makes problem. Because of using battery as a power ,the probability that network die will increase. Therefore should try to not waste energy,in the aim of increasing network's life time.

The idea of **EH-WSN** is proposed for solving the problem of power in **WSN**. In this theory, network harvest power from environment and use this power instead of battery. The aim in **EH-WSN** network is not keeping network alive

longer, but because there are enough energy in this theory the goal changes to maximizing the workload.

The purpose of this thesis is following in some steps: first of all review literature about the group of energy-aware routing algorithms in **WSN** and more deeper be involved in the algorithms that are specified to support energy harvesting technology. In the next step should choose some candidate routing algorithms from the category of **EH-WSN** routing algorithms base on their evidence in literature. After selecting the candidate we have to design and implement a simulator to simulate the given algorithms in different scenario. With the help of some analysis metrics, at the end, we should highlight the behaviour of our candidate in different simulation condition.

## 1.4 Contribution

- A survey of the energy-aware **WSN** routing algorithms in literature.
- Looking for routing algorithms that can support **EH-WSN** technologies.
- Selection of three **EH-WSN** Routing algorithms which are supported more in literature.
- Defining analysis metrics.
- Writing codes to implement simulator by using Matlab program.
- Simulation of network for the chosen routing algorithms.
- Behavioral analysis of the results from the simulation scenarios .

## 1.5 Thesis structure

1. **Introduction**: It provides a general information about **WSN**, **EH-WSN** and their related topic. At the end discuss about the goal of this thesis.
2. **Energy-Efficiency Routing Protocols in WSN**: Explain Energy-Efficient Routing protocols in **WSN** and classify them in to different groups.
3. **Routing Protocols for EH-WSN**: Discuss about Routing protocols can support **EH-WSN** technology and choose some of them that has more evidence in literature.



- 
4. **Simulator architecture:** Implementing simulator base on feature of candidate algorithms and analysis metrics.
  5. **Simulation Runs:** Reports the simulation flow to analyze the behavior of the chosen metrics respect to the different routing algorithms and simulation scenario.
  6. **Conclusion:** Conclude the project with summary of all contribution.



## CHAPTER 2

# Energy-Efficient Routing Protocols in WSN

---

This chapter investigates into Energy-Efficient routing protocols in Wireless Sensor Network (**WSN**). The first section discuss about what is the main idea behind a routing algorithm. The following sections describe the goal of Energy-Efficient Routing protocols and the policies for selecting a routing algorithm.

In this chapter energy-efficient routing protocols are classified by four aspects: *Network Structure*, *Communication Model*, *Topology Based* and *Reliable Routing*. *Network Structure* can be *flat* or *hierarchical*, the second category divide the **WSN** into *Query-based*, *Coherent and non-coherent* and *Negotiation-based* communication model. The *Topology-based* point of view can be categorize to *Location-based* and *Mobile agent-based*. The last classification can be divided into *Multipath-based* and *QoS-based*.

## 2.1 Routing Protocols background

As described before, **OSI** model is divided into different layers and it is used as reference for **WSN** architecture. Routing protocols are defined in the third

layer called Network layer, this layer is used for logical addressing, it also offers routing technologies.

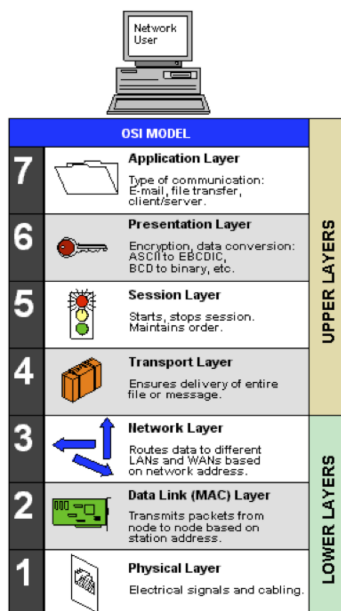


Figure 2.1: OSI levels

The error handling, packet sequencing and addressing are the main functionality of Network layer. In this layer packets are sent from source to destination. To reach destination, routing protocol selects optimal path through the series of interconnected nodes.

## 2.2 Energy-Efficient Routing Protocols

In WSN all the nodes have power source which provide energy to participate in the network. In the basic WSN the power source is batteries. As we know, the amount of energy that can be stored by battery is limited and can not be recharge. So we should try to not waste this energy and use it in an optimized way. The aim of routing protocols is to use the battery energy in an efficient way to increase the network lifetime. Therefore target of the propose routing algorithm should be energy consumption minimization and network lifetime

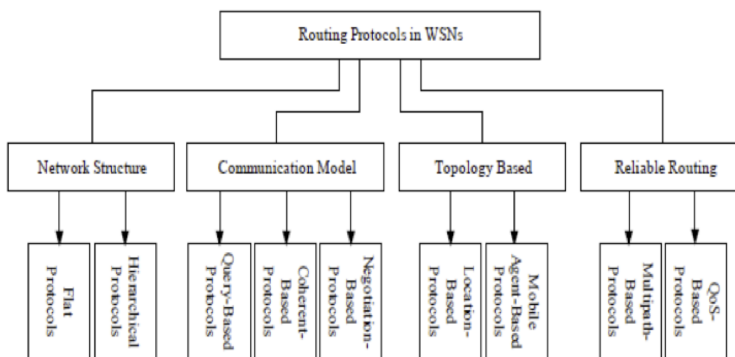
maximization.

For evaluating the performance of routing protocol there are some concept related to energy efficiency that also discuss in [13].

- **Energy per packet** : is the energy needed to transmit a packet.
- **Network lifetime**: there is no special definition about using this term. The most common one considers the time when a certain function of the networks node is dead. From previous talking we knew that maximizing the network lifetime means prolong the battery lifetime of nodes. The common idea for achieving this goal is to use the shortest path.
- **Low energy consumption**: More a routing algorithm consumes less energy; more it counts as the better routing protocol. Note that this concept doesn't support the goal of energy harvesting network.
- **Idle listening**: a sensor node in this situation does not send or receive packet but it can consume a considerable amount of energy.

## 2.3 Routing techniques in WSNs–Classification

Several routing algorithms have been developed to solve the problem of **WSN** power. All these routing algorithms take into account the inherent feature of **WSN** and the architecture requirement.



**Figure 2.2:** Classification of routing protocols in WSN [3]

### 2.3.1 Network structure

#### 2.3.1.1 Flat Networks Routing Protocols

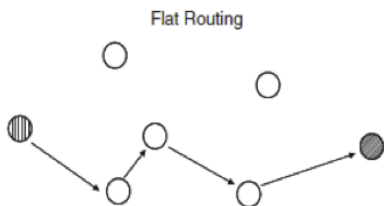


Figure 2.3: Flat routing

In flat network all the nodes are equal and the routing algorithms are divided into some categories: Pro-active protocols and Re-active protocols [3] , [14].

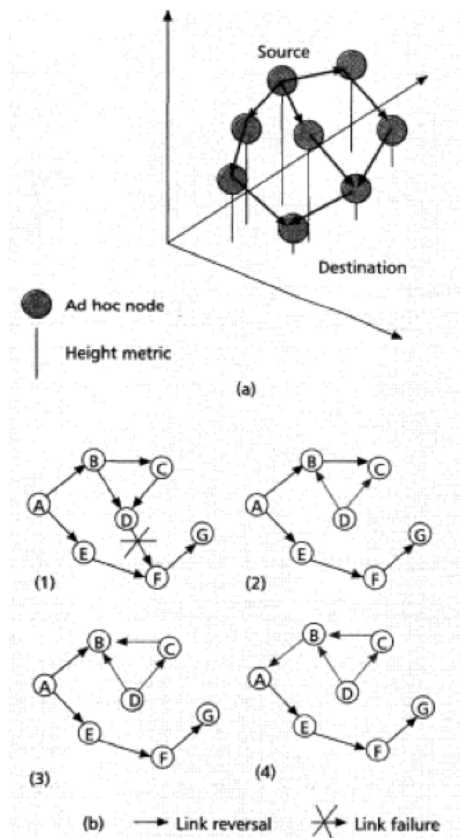
- **Pro-active routing protocol:** As described in [15], in this category path to destination is computed before the request and each node has its own routing table. This protocol is also sensitive to any network change and it sends update through the wireless network, but keeping the information up-to-date needs extra battery power which is limited in a **WSN**. There are many routing protocols with this method such as *DSDV*, *WRP*, *GSR*, *STAR*, *DREAM* and *TBRPF* [16].

Here is explaining the Topology Dissemination Based on Reverse-Path Forwarding (**TBRPF**) protocol as an example of pro-active routing protocol [17], this protocol transmits only the differences between the new network and the previous one. So the routing table becomes more up-to-date. Each node consists of many information, such as: a list of neighbor nodes, a table consist of all link-state and a list of children. This protocol works using the concept of reverse-path forwarding, it means that when something changes, the node that is responsible for this change sends the information in the reverse direction along the spanning tree formed by the minimum hop path. This method is suitable for the environment with fixed number of nodes.

- **Re-active routing protocols:** In this method there is no routing table, the process of finding route start when there is a request [18], this can produce some delay in the **WSN** since the requested path is not available and have to be found. The main idea for making this algorithm was to

reduce the overhead of networks in the opposite of Pro-active, because it use the information just for the active route.

One of the examples of this protocol is Temporarily Ordered Routing algorithm (**TORA**).



**Figure 2.4:** TORA algorithm a) Route creation (showing link direction assignment); b) Route maintenance (showing the link reversal phenomenon) [4]

This algorithm has been proposed to work in dynamic mobile network environment. In this protocol each node knows its own height and also the height of the neighbors which are directly connected to it. Each node for assigning the height, consider the location toward the destination. The algorithm consists of 3 steps:

- Route creation
- Route maintenance

- Route erasure

In the first and second phases, node uses a height metric to establish a graph toward destination. Each link is upstream or downstream, it depends to the height of neighbors which is greater or smaller than it's own height. We need nodes maintenance, because the graph can be changed in the case of mobility of the node, so the **TORA** has to re-establish a graph toward the destination. As describe in Figure 2.4 when a upstream link fail there is possibility to reverse the direction of one or more links in order to find a path.

As a comparison between Pro-active and Re-active:

1. Pro-active require all the routing information, but re-active need less amount of routing information and then less energy consumption for the sensor node.
2. Pro-active waste bandwidth and energy to periodically or when the topology change send updates but in re-active there is no need.

### 2.3.1.2 Hierarchical Network Routing protocols

Here all the nodes are in cluster and in the opposite of flat network, they are not peers. The advantage of using this method is to reduce the size of routing tables and as a result reduce the overhead.

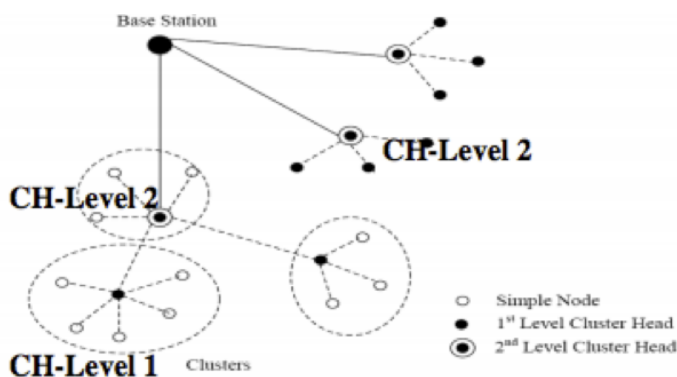


Figure 2.5: Cluster-based Hierarchical Model [5]

- Low-energy adaptive clustering hierarchy (**LEACH**) [6]: it uses cluster to minimize the communication cost and increase the network lifetime by just using a small dissipation of energy of the system. Each cluster has a



cluster head that is responsible to distributing the energy load between the nodes in the network. The operation of **LEACH** contains two process:

1. **Setup phase:** clusters are organized, cluster head become clear by using stochastic algorithm. If a node becomes a cluster head for one time, it can not become again for  $P$  rounds.  $P$  shows the desired percentage of cluster head, so the probability for each node to become cluster head in each turn is  $1/P$ . Changing the position of cluster head leads to balanced energy consumption for all the nodes and makes the life of network longer.
2. **The steady state phase:** in this part, data will be sent to base station. Each node, that was not chosen as a cluster head, selects the closest cluster head and join to that cluster. Now the cluster head make a schedule for each node in that cluster to transmit it's data.



**Figure 2.6:** Time line showing Leach operation. Adaptive clusters are formed during the setup phase and data transfer occurs during the steady state phase [6]

- *Low-energy adaptive clustering hierarchy centralized (LEACH-C):* In this algorithm base station receives information about the location and energy level of each node in the network. With this information BS find a pre-determined number of cluster head and configure the network in that clusters. It has some good points in compare of **LEACH** that:
  1. In **LEACH** the number of cluster head will be changed in any round due to the lack of global coordination among nodes but in **LEACH-C** is predetermined to an optimal value.
  2. Base station in **LEACH-C** produces the cluster in base of global knowledge of the network which is to require less energy for transmitting data.
- *Power-efficient gathering in sensor information system (PEGASIS):* It is a chain based protocol [19]. chains consist of a group of nodes that are close to each other and can also make a path to the base station. BS defines how the nodes stay together to make chain and then broadcast it to all the nodes. In any chain just one node has been selected for transmitting to BS instead of multiple nodes. For achieving the goal of extending network

life time, all the nodes communicate, just with the neighbors and take a turn in communicating with base station.

- **Sleep/Wake scheduling protocols:** This protocol is described in [7]. It saves energy by setting the node in sleep mode during idle times and wake up right before message transmission/reception. The important part in this protocol is to synchronize sender and receiver. This synchronization is achieved immediately after exchanging synchronization message. In this protocol the same as other in hierarchical category, there are many clusters, but the important issue is that the cluster members can be cluster head in other cluster. The Figure 2.7 shows an example, node C is the cluster member of A but at the same time is the cluster head of F.

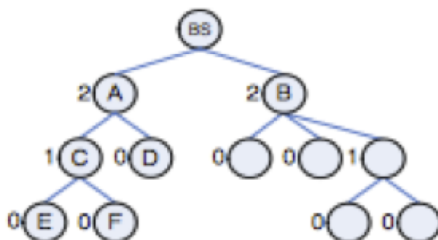


Figure 2.7: A three-level cluster hierarchy [7]

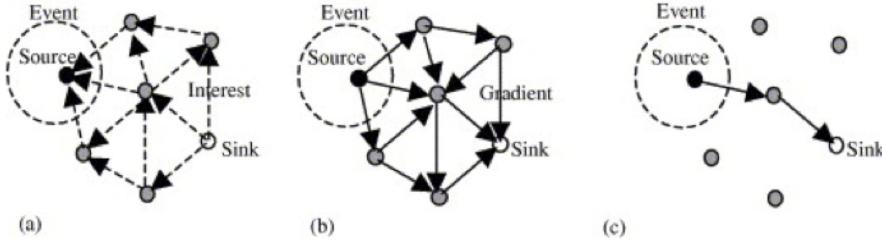
## 2.3.2 Communication model scheme

### 2.3.2.1 Query-based routing protocols

In this group of routing algorithm as describe in [20], Destination node send it's request through the network and then any node which has this data, send it back to the destination.

- *Direct Diffusion (DD):* This idea is based on diffusing data through network by using a naming method. The main reason of using this technique is to not waste energy in some unnecessary operations and try save energy. As it shows in Figure 2.8 It consist of some steps [8]:first of all quarry of interests are defined by arranging tasks which are named using a list of attribute-value,these values can be the type of data,the interval of transmission data and so on. Then interest broadcast from a sink through all

the neighbors. Each node cache received interest and compare it with received data. In the case of matching interest with received data, by using gradient and reinforcement choose one path among multiple path from source to destination. Gradient is a reply link to a neighbor that received interest.



**Figure 2.8:** Directed\_Diffusion a) Interest propagation. (b) Initial gradients setup. (c) Data delivery along reinforced path. [8]

### 2.3.2.2 Coherent and non-coherent-based routing protocols

In WSN all the data processing should be held in node level. The nodes try to process the data within the sensor network. The routing mechanism is explained in [9] and divided to this group:

- *Coherent data processing-based routing*: The nodes will minimally pre-process the raw acquired data such as time-stamp and then will send this to the aggregation.
- *Non coherent data processing-based routing* [21]: In this algorithm nodes process some part of data then send it to central node which has the responsibility to do the rest of processing. This theory consists of three phases. At first we should find the goal and try to collect all the information related to that. In second step we should choose all the nodes that participate in the function and then we should inform about it to all neighbours. At the end we should find the central node to process the information.

#### 1. *Single winner algorithm (SWE)* [9]:

Here a single aggregator node will be choose for complex processing. It is named as a central node. It will be selected base on the energy reserve. For selecting CN, all the nodes broadcast a message and introduce themselves as a central node. Then nodes that receive

these messages compare the propose CN with themselves and send the result of comparison as a second batch of message to all neighbors. Each of these message present a better CN. Otherwise the message will be discard.

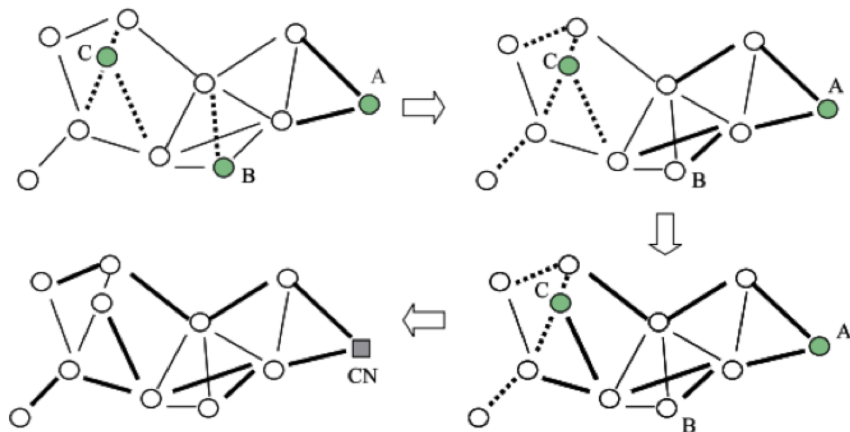


Figure 2.9: SWE election process [9]

2. *Multiple winner algorithm (MWE)*: This algorithm is the extension of previous algorithm. All the nodes send their data to the aggregator, but this process use a lot of energy, the way to reduce this energy is to limit the number of node that can send data to central aggregator. Also, instead of choosing one node as a central aggregator there is the possibility for each node to keep a record of up to  $n$  nodes as a candidate.

### 2.3.2.3 Negotiation-based routing protocols

Negotiation-based routing protocols or *sensor protocols for information via negotiation (SPIN)* is a data-centric algorithm and spread information in network in an energy constrained way. It relies on two ideas [10]:

1. Sensor nodes should communicate about the data that they have or data they will obtain, to operate efficiently and save energy.
2. Nodes should have the ability to adapt to their energy resources variation to increase the operating lifetime.

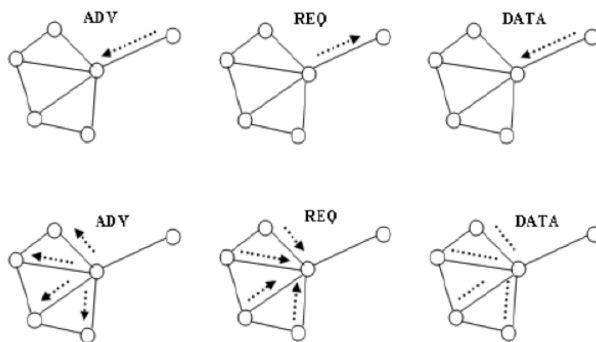


Figure 2.10: SPIN protocol, *ADV*, *REQ* and *DATA* packet [10]

The main idea as explain in [22] and in Figure 2.10 is that if a node has some data, it advertise it by sending *ADV* packet to other nodes and if nodes are interested in this data they send *REQ* packet and then received the actual *DATA*. This algorithm is based on the idea that each node should have information just about single-hop neighbors, so all the changes related to the topology will be local. No guarantees of data delivering is the main issue of this algorithm.

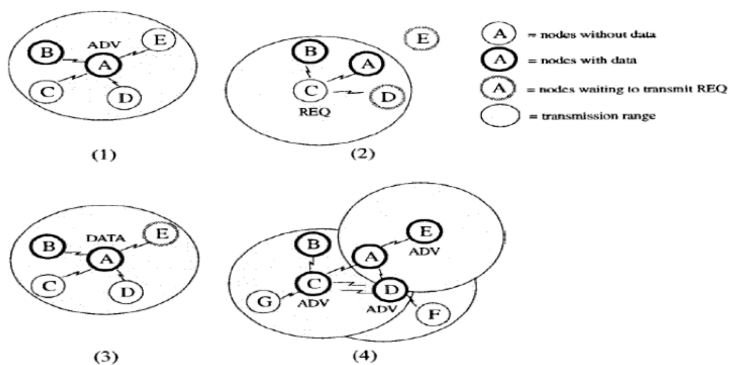


Figure 2.11: The SPIN-BC protocol. Node A starts by advertising its data to all of its neighbors(1). Node C responds by broadcasting a request, specifying A as the originator of the advertisement(2), and suppressing the request from D. After receiving the requested data(3), E’s request is also suppressed, and C, D, and E send advertisements out to their neighbors for the data that they receive from A(4) [22]

- ***SPIN-BC***:

This protocol design [22] for networks where nodes use a single shared channel to communicate. In this solution if there are more than one node that is interested in *ADV*, just one of them start to send *REQ*, this helps to save energy. Later, when node start to send *DATA*, it broadcasts *DATA* once, so it will be received by all the nodes that was interested in it.

### 2.3.3 Topology based scheme

#### 2.3.3.1 Location-based routing protocols

In this category the main idea is physical distance. Physical distance and distribution of nodes in area can effect on network performance. This protocol is based on two principal ideas: First all the nodes have some information about their neighbors, second the source of packet has information about the position of destination. In this algorithm nodes should send a *HELLO* message periodically to let neighbors find their position. One of the interesting thing about this protocol is that it works without any routing tables.

- *Geographic and energy aware routing (GEAR)* [23]: This protocol use energy aware and geographically heuristic routing to find some information about neighbors in order to send a packet to destination.
  1. When there is a closer neighbors to the destination, this protocol choose the next-hop from a group of neighbors which is located closer to the destination.
  2. When all the neighbors are far away from the destination, the next-hop should choose in a way to minimize the cost value of this gap which completely discuss in [23].
- *Implicit geographic forwarding (IGF)*: Here instead of using static routing table and independent of knowing information about topology; the next-hop will choose online in real-time. This method is really valuable in dynamic sensor network. It omits costly communication because it doesn't need to save all the information of neighbors for routing.
- *Minimum energy relay routing (MERR)*: It is base on the idea that distance between two nodes is the key parameter to consider and can effect on energy consumption of whole path. Thus in **MERR** each nodes search for the closest nodes within its maximum transmission range. As soon as it finds the node it adjust transmission power to the lowest, so

the radio signal has power to reach the specific node not more. The main advantage of this protocol is that it minimizes energy consumption by choosing the nearest node in routing path.

### 2.3.3.2 Mobile agent-based protocols

Each sensor has limited memory to save all the programs and run all the application. So we can use mobile agent to migrate code among the nodes of the network as an information processing. So it makes a network more flexible. In [24] design issues of mobile agent in **WSN** are divided in to different group.

1. **Architecture:** It is based on the topology of network
  2. **Itinerary planning:** It selects a group of nodes that can be visited by mobile-agent.
  3. **Middleware system design:** It fills the gap between operating system and high-level component and to facilitate the development of application.
  4. **Agent-cooperation:** It can work as a single processing unit or distributed collection of components.
- *Multi-agent based Itinerary Planning (MIP)*[25]: The goal of this algorithm is to dynamically define a group of source node that can be visited by mobile agents. For achieving this goal the following phases are needed:
    1. Find the visiting central location (**VCL**) in any agent. This location is computed in function of source density.
    2. Determine the source nodes that can be visited by a specific agent in a specific area centered at **VCL**.
    3. Determine the source-visiting sequence.
    4. If still there are some nodes that doesn't support in any agent,the next time for finding **VCL** we should also care about these nodes.

### 2.3.4 Reliable routing scheme

#### 2.3.4.1 Multipath-based routing protocols

This algorithm is multi-path routing. Using multi-path routing is good to obtain load balancing and it is also more flexible to route failure. Also by finding

reliable path it ensure reliable communication. Actually it takes advantage of lower routing overhead in compare of single-path routing protocols [26].

- *Routing on-demand acyclic multipath (ROAM)*: As explain in [27] it is an on-demand routing algorithm that is responsible for gathering multiple loop-free path to destination. Each router has a distance table and routing table. Distance table is a matrix with distance information of two neighbors. Routing table organize as a column vector, for any destination, containing the distance to the destination. It may happen that router get a data for delivering to a destination without having entry about that destination in routing table, in this case the router has to start diffusing search. It will explore from source; node by node all the network till find a router that has an entry about destination. After finding an entry the source can obtain the distance to destination. If there is no route to that destination, it will be counted as unreachable.
- *Label-based multipath routing (LMR)*: Here we broadcast the control message in network [28]. By passing the message from a path, label are assigned to that path. In a working path, when a node choose a link, this label message broadcast to their neighbors. Label message count as an integer term label. This number is increased by one after passing from each working node and then broadcast a new label message to neighbors. All the nodes should be inform about this number. When a node receives two or more label message, take the smaller number and will forward it. If it receives two equal label message, it will forward the one which receive first. Each node should record all the label that has been seen and from where they are coming from.
- *Gradient broadcast (GRAB)*: The main idea is to deal with the unreliable nodes and fallible wireless links by robust data delivery [29]. The sink broadcasts an advertisement packet with cost zero. The initial cost of each node is  $\infty$ . When a node receives the advertisement packet, it add the link cost between itself and sender to the sender's advertise cost. Then compare this cost and previously recorded cost and set the new cost as the smaller of these two. This process is reiterated in whole the network. The algorithm chooses the path by using the information from multiple nodes that has effort in deliver data, without dependency of any of them.

#### 2.3.4.2 QoS-based routing protocols:

In this theory network has to be balance between energy consumption and data quality. The main goal is about throughput and average response time.



- *Sequential assignment routing (SAR)*: This routing algorithm concern about requirements while it searches for a path. So at the same time should think about three factors:energy resources,priority level of each packet and QoS metrics on each path such as delay,energy and bandwidth. To avoid single path failure, it use multi-path approach. The main idea here is to minimize the average of QoS during lifetime of network.
- *SPEED protocol*: It's a routing protocol that provides real-time end-to-end guarantee. It cares about desired delivery speed in network. So the advantage of this method is to performs better in terms of end-to-end delay but it doesn't support energy consumption policy.



# Routing protocols for EH-WSN

---

In this chapter we are going to describe some routing protocols for **EH-WSN**. As we discussed in previous chapter, in **WSN** each node has a finite energy storage element such as batteries. So the goal of efficient routing algorithm is to minimize the energy consumption and maximize the lifetime of network.

The recent advances in ambient energy harvesting technologies allow a sensor node to get power from environment instead of using batteries. Each node has energy harvesting device which converts ambient energy into electrical energy. In following chapter we will discuss about the goal of **EH-WSN** routing algorithm and the basic algorithm for finding a max flow in network. Section 3.3 describes sustainability in a network and how to compute the maximum energetically sustainable workload. The last section describes seven routing algorithms which are suitable for **EH-WSN** with all their details.

## 3.1 EH-WSN routing algorithm overview

In this kind of routing algorithm the goal is not maximizing the lifetime of network, but is maximizing the workload in the energy-harvesting network. At

the same time we should consider the possibility that environmental power sources changes over time, so we should define a dynamic routing algorithm that adapt itself with the variation of environmental conditions. In general, a node consumes energy in three stages: sampling, processing and relaying packets. Routing algorithm in most of the time is responsible for packet relaying. For example in the same area if there are some nodes equal to each other, the routing algorithm should choose the energy optimize path.

As explained before our goal is to maximize the workload in the energy-harvesting network and at the same time we know that routing affects on workload of the node, so to reach our goal, we should choose a good routing algorithm.

### 3.2 Ford-Fulkerson Algorithm

This section explains the *Ford-Fulkerson* algorithm. This algorithm is a basic algorithm that later will be extend to evaluate the sustainability of two considered energy harvesting routing algorithms such as **(R-MF)** and **(R-MPRT)**.

The aim of Ford-Fulkerson algorithm is to compute the maximum flow in network. It associate costs with network links and try to route packet from source to destination along path with the minimum cost.

There are several paths from source to destination and each of them has a bottleneck edge. Bottleneck edge is the edge with minimum capacity on that path. The maximum flow unit that we can send through the path is equal to minimum capacity of that path. By passing from each path, should try to saturate at least one edge. Saturate means use all the capacity of that edge. The algorithm uses residual graph which shows remain capacity in each edge after each algorithm iteration.

The following points are important for the algorithm:

- Equation 3.1 means that the flow from each edge cannot be more than the capacity of that edge.

$$\forall (u, v) \in E f(u, v) \leq c(u, v) \quad (3.1)$$

- Equation 3.2 means that in all the nodes except source and destination the amount of incoming flow it is equal to outgoing flow.

$$\forall u \in V > u \neq s \text{ and } u \neq t \Rightarrow \sum_{w \in V} f(u, w) = 0 \quad (3.2)$$

Now as explanation of the residual capacity we can say that:

$$c_f(u, v) = c(u, v) - f(u, v) \quad (3.3)$$

It means that new capacity is equal to old capacity minus the flow that is passing in that direction.

**Algorithm** *Ford-Fulkerson*

**Input:** Graph  $G$  with capacity  $c$ , source node  $s$  and destination  $d$ .

**Output:** Flow  $f$  from  $s$  to  $d$  which is the maximum flow.

- $f(u, v) \leftarrow 0$  for all edges  $(u, v) \in p$  In first step we assume the flow of all edges as zero.
- While there is a path from  $s$  to  $d$  such that  $c_f(u, v) > 0$  for all edges  $(u, v) \in p$ :
  1. Find  $c_f(p) = \min_{(u, v) \in p} c_f(u, v)$
  2. For each edge  $(u, v) \in p$ 

$$f(u, v) \leftarrow f(u, v) + c_f(p)$$

Here we explain the algorithm by using an example:

- The *Figure 3.1a* shows the original graph by showing capacity in any edge, which is the maximum flow you can send from that edge.
- In first step in *Figure 3.1b* we assume that all the edges has the flow at zero and we try to show the residual capacity in another edges.
- **Iteration 1:** In *Figure 3.1c* we have to choose one path in the graph, for first step we choose the path  $[s, 1, d]$  and by take a look at *Figure 3.1b* we choose the minimum capacity in that path which is five. Then count the flow and capacity through formula as explained before.
- **Iteration 2:** In second iteration in *Figure 3.1d* choose the path  $[s, 2, d]$ . The minimum capacity which can be find in *Figure 3.1c* is three.

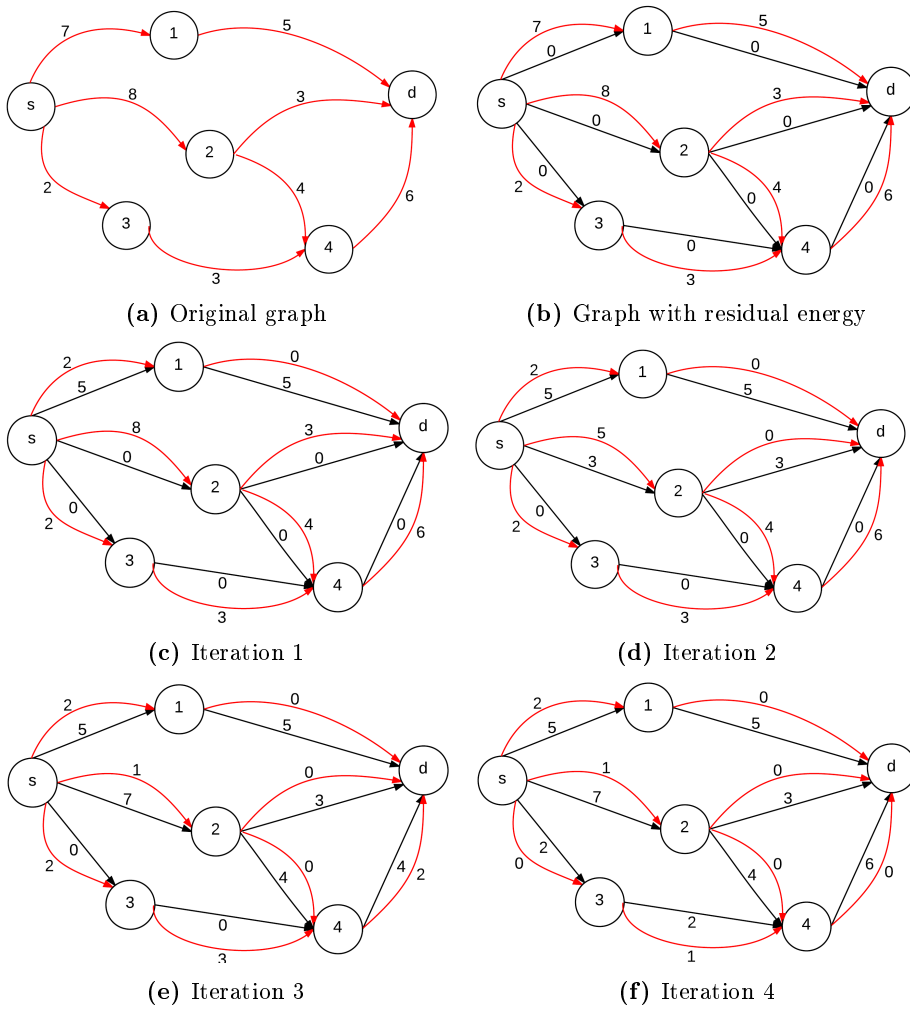


Figure 3.1: Ford-Fulkerson example

- **Iteration 3:** In this step we choose the path which pass from node two and four as [s,2,4,d]. As shows in *Figure 3.1e* The minimum capacity is 4.
- **Iteration 4:** In *Figure 3.1f* By choosing the path [s,3,4,d]and minimum capacity of two , we have already passed from all the path from source to destination.
- In this step for counting maximum flow, we have to sum all the minimum capacity that we found in previous steps. So as a result:

$$5 + 3 + 4 + 2 = 14$$

### 3.3 Energetic sustainability of routing algorithm for EH-WSN

As already hinted in overview the aim of **EH-WSN** routing algorithm is to increase the workload. But for some algorithms of this category such as **R-MF** and **R-MPRT**, the concept of sustainability is also considered [30]. Sustainability means that each node should not consume more energy than the energy it can harvest over a period of time.

In this section we try to collect all the information presents in literature about energetic sustainability of routing algorithms in **EH-WSN**, discuss about how to compute the maximum energy sustainable workload and how to evaluate the sustainability of algorithms.

- **EH-WSN model**

1. **Power model:** The energetic sustainable routing algorithm should be aware about the concept of packet energy ( $pE$ ), which is the amount of energy used by each node to process a packet. It should be also aware of available power ( $AE$ ), which show the amount of available power in each node.

Packet energy contains all the needed energy for producing, processing and directing packet in a path. Both producing and processing , can be constant value [31], but transmitting energy is related to the distance between source and destination. In fact there is a quadratic relation between distance and the amount of power consumption  $pE = pE0 + PE1.d^2$ , as explained in [32].

Consider that packet energy is not the only power used in **WSN**. There is also power usage in idle time, wait to receive events or listening for incoming packets.

In this kind of routing algorithm since we are attracted in sustainability of the workload, we just consider such a node with positive available power. It means that a group of nodes spends power in idle state lesser than the power that they get from environment.

2. **Network model:** We define **EH-WSN** as a graph  $G = (V, E)$  where  $V$  is the set of nodes (vertices) and  $E$  is the wireless link (edges). Take into account that there is an edge between two nodes if they can send a packet in that channel. Each node is commented with available power.
  3. **Workload model:** We use sensor network in case of monitoring environment or detect an event. In any event, the data that will be sent can be continuous or random any time that an event occurs. In our project we care about monitor environment and collect information. All sensors sense the area periodically and send the data that has been collected to the base station.
- **Energetic Sustainability:** To evaluate the optimal routing algorithm we should examine the Maximum Energetically Sustainable Workload (**MESW**) of a given routing algorithm, which represent a workload that can be energetically sustainable in a given routing algorithm. It is also need to examine the Optimum MESW ( $MESW^{opt}$ ) that express the **MESW** of the best routing algorithm.

To evaluate how much the packet energy can be sustained with the environmental power we define recovery time ( $T_e$ ). This is the time required by each node to harvest an amount of energy used to receive and transmit a packet through a path. As a result the following relation exists between environmental power ( $P_n$ ) and packet energy (Packet energy ( $pE_e$ )).

$$T_e = \frac{pE_e}{P_n} \quad (3.4)$$

Another main quantity is Channel capacity ( $C_e$ ) [30], which is the maximum packet rate across an edge. The channel capacity has an inverse relation with recovery time. It means as much as the time for receiving energy increase, the maximum rate decrease.

$$C_e = \frac{1}{T_e} \quad (3.5)$$

The flow  $F_e$  from edge  $e$ , as explained in Ford-Fulkerson algorithm, is limited by energetically sustainable channel capacity so:

$$F_e \leq C_e = \frac{1}{T_e} = \frac{P_n}{pE_e} \quad (3.6)$$



This networks with constrained edges to a given channel capacity are called flow network. For finding a maximum flow in this network we use Ford-Fulkerson algorithm that described before. At the same time we shouldn't forget the fact that all the edges belong to one node should share the same amount of power (the available power in source node). So when we want to define the channel capacity it is not enough to just think about the maximum packet rate across each edge, we should also consider another concept which is called *node-constrained flow network* [30], it point out that the flow from any edge is not limited just with capacity of that edge, it is also limited by the overall budget power of that node.

$$\sum_{e \text{ exiting from } n} F_e p E_e \leq P_n \quad (3.7)$$

This concept is not used in Ford-Fulkerson method. Because as we explained, this algorithm is edge-constrained and doesn't take an account the idea of node-constrained.

We want to evaluate how much a **MESW** of a given routing algorithm is similar to the optimal one  $MESW^{opt}$ , in this way we can have a comparison metrics between the proposed routing algorithm. These quantities can be computed as follow:

1. **Computing  $MESW^{opt}$ :** To evaluate  $MESW^{opt}$  we have to extend Ford-Fulkerson method, because this algorithm give us the maximum flow in the network without considering the idea of node-constrained. So we have to use the extension version of Ford-Fulkerson which explain in [33].
2. **Computing **MESW**:** The authors of [30] try to explain an algorithm for estimating the **MESW** in a routing algorithm. The base of this algorithm is on using the minimum residual power. It is the difference between the available environmental power at node  $n$  and the power usage to sustain the workload.

The authors in [30] try to evaluate two routing algorithms(**R-MF** and **R-MPRT**) by using the concept of **MESW** and simulation. Their result say that these algorithms show good sustainability.

### 3.4 EH-WSN Routing algorithms

This section presenting a literature survey on routing algorithms, which are suitable for Energy Harvesting Wireless Sensor Network (**EH-WSN**), are reported.

### 3.4.1 Randomized Max-Flow (R-MF)

This algorithm is one of the energetic sustainable routing algorithm for **EH-WSN**.

Each edge is assigned with:

$$Capacity = \frac{\text{harvesting rate of the transmitter}}{\text{packet energy}} \quad (3.8)$$

This routing algorithm is based on using an off-line routing table, stored in each node that shows the node links used for packet transmission. The probability to use the edge  $i$  in node  $n$  is proportional to the max flow from that edge. For finding the node-constrained maximum flow in each edge we have to use the extension version of Ford-Fulkerson method.

### 3.4.2 Energy-opportunistic Weighted Minimum Energy (E-WME)

This routing algorithm provides energy aware framework with energy replenishment. It tries to use the important part of two following routing algorithm. Algorithm '*ME*' trying to achieve a minimum energy routing and algorithm '*max-min*', trying to choose a node with high residual energy. As we know most of the **EH-WSN** protocols for routing use the residual energy, meanwhile **E-WME** uses both residual energy and the replenishment rate of the transmitter.

As shown in [34], choosing the shortest path with taking into account the cost function is a good option. This cost function is the combination of residual energy and the cost of replenishment rate.

In this algorithm, resources are allowed to refill energy storage and each node just have information about short-term energy replenishment [35]. Replenishment can happen at different rates. There is possibility to have constant replenishment rate or flexible rate, but here we describe the algorithm in case of constant replenishment rate (in time) for each node. We should also consider that there is possibility that different node has different replenishment rate.

The algorithm works really simple, we assign a cost to each edge, which is based on residual energy and replenishment rate. Then find a shortest path with respect to this metric. For each edge this metric has been modelled as [36]:

$$c_u = \frac{E_{M,u}}{(p_u + \epsilon) \log(\mu)} \cdot (\mu^{\lambda_u} - 1) \cdot e \quad (3.9)$$

Where  $p_u$  is energy replenishment rate,  $E_{M,u}$  is battery capacity,  $\lambda_u$  is the power depletion,  $e$  is the energy needed to send a packet to neighbor nodes,  $\mu$  and  $\epsilon$  are constant.

The  $\lambda_u$  shows the power depletion index and it can be defined as:

$$\lambda_u = \frac{E_{M,u} - E_{C,u}}{E_{M,u}} \quad (3.10)$$

That  $E_{C,u}$  is the available energy exactly before processing the packet.

After defining the cost for each link we should find the minimum path cost to send data. We assume a path cost for sink is equal to zero. Every time the node  $n$  receive a beacon from node  $i$ ; add the cost link between node  $n$  and node  $i$  to the cost function of node  $i$  and then compare this result with the path cost to the sink that already have. Then choose the minimum one as a path cost from that node to the sink.

The algorithm gives results in a good performance because:

1. As we mentioned, it is a mix up the idea of minimum energy routing and residual energy. As an example, if in the routing path, we have two parallel link which receive and transmit with the same residual energies, we can choose the link with the minimum energy. In other hand if we have two nodes with equal link energy cost, we choose the node that has the larger residual power.
2. In a network if the refine rate is different between nodes, the algorithm tries to choose nodes with faster energy renewable in the path.

### 3.4.3 Randomized Minimum Path Recovery Time (R-MPRT)

This algorithm has two versions. We refer to the original one as *R-MPRT-org* and to the modified algorithm as *R-MPRT-mod*.

#### 3.4.3.1 R-MPRT-org

This algorithm is really similar to the **E-WME** discussed before. But with a simpler cost function.

Choosing a route at each node is based on energetic sustainability information. The idea is the same as **E-WME**, choosing a shortest path with considering the cost function. We can define a cost function to each edge as:

$$cost = \frac{\text{Packet energy}}{\text{Harvesting rate of the transmitter}} \quad (3.11)$$

This cost function is the inverse of cost function in **R-MF**, so the probability to send a packet in the path is inverse proportional to the corresponding path recovery time. Because cost function is the same as recovery time here. As explained before recovery time is the time needed to harvest energy needed for packet transmission.

The algorithm requires local knowledge of the network, because for sending the packet to other nodes, it should know about their cost function and choose the minimum one for sending data. The responsibility of sending the information of each node to the local neighbors is within the beacon transmission.

For each path to the sink should compute the cost function of that path. At the end should choose the shortest path from each node to the sink which explained in 3.4.2 how to do it.

Routing table in this theory is dynamic because it depends on the harvesting rate of each node.

### 3.4.3.2 R-MPRT-mod

The author found that this modified version performs much better when it uses available energy of the transmitter instead of using the harvesting rate. The author do not support this claim by providing any evidence.

$$cost = \frac{\text{Packet energy}}{\text{Available energy at the transmitter}} \quad (3.12)$$

## 3.4.4 Randomized minimum path energy (R-MPE)

This algorithm works base on the minimum energy required to reach the sink [30]. Path energy information (*E<sub>path</sub>*) is sent downward within a message and stored in the local routing table of each node.

This information propagation starts from the base station which has *E<sub>path</sub>* equal to zero. When the message reach a node *n*, from edge *i*, it updates the packet

energy required by that edge in the routing table :

$$E_{path_i} = E_{path} + pE_i \quad (3.13)$$

The probability of sending packet from an edge is inversely related to the corresponding path energy.

### 3.4.5 Energy Harvesting Opportunistic Routing Protocol (EHOR)

This is the opportunistic routing protocol for multi-hop *WSN-HEAP* (powered by ambient energy harvesting). This algorithm uses opportunistic retransmission, it means that in the routing if one node fail we can use another nodes for transmit data packet. Another challenge of this method is to choose optimal forwarder. As we know predicting the next awake nodes is difficult, since the rate of charging is dependent on environmental factor. So it works in the base of partitions nodes into regions and then gives the priority to them for transmission. This priority is based on the proximity to the sink and residual energy.

#### 3.4.5.1 Traffic and energy characterization

In *WSN-HEAP* there is the energy-harvesting device that converts ambient energy to electrical energy. Also there is energy storage device, which it stores the energy, has been harvested from the environmental. When enough energy is harvested the transmitter starts to work and continuously broadcast data packet till the energy finish. Then it will turn off. This process repeats again in the next cycle.

#### 3.4.5.2 Node classification:

- **Relay node:** It uses to forward data packets from the source to the destination. When it receives any data packet, it would buffer the data packet and schedule it for transmission. Relay node can be three different phases: charging state, receive state and transmit state. At first it is in charge mode, after storing enough energy, if it has data packet to send, it start to transmit

- **Source node::** It works the same as relay node, but it doesn't have the phase of transmitting data after receive it. It will send its own data in the transmit time. Every data packet has a unique *ID*
- **Sink::** This node is connect to the main power and collects all the data in network.

### 3.4.5.3 Regioning in EHOR

For sending a packet from each node to the sink should know the forwarder nodes. A node is in the forwarding stage of the sender if its distance to the sink be less than the distance of sender to sink.

This algorithm divide the possible set of forwarding neighbors into several region. The number of region depends on the available candidate for forwarding and average energy harvesting rate.

A node is sending data by itself or has duty of forwarding data. If a node receive data from other source node and distinguish that is not within the forwarding region of sender it will not forward data otherwise it tries to find the other region to transmit.

Each region has an *ID* which calculate by knowing the distance of the sender to that region as explain complete in [2]. As much as the region be closer to the sink in compare of other regions it has lower *ID*. Node with lower region *ID* has more priority for forwarding the packet.

### 3.4.5.4 Performance

The performance of the **EHOR** will increase if we choose the priority of the region based on the amount of energy left in the node, in addition to the distance from the sender. Nodes further away from the sender have more priority for forwarding data but the sender should have enough energy to send data far away. There is a factor  $\beta$  which shows the forwarding priority base on residual energy and distance to the sink.

### 3.4.5.5 Routing behavior

As we said, there are three possible phases for each node such as charge phase, transmitting or receiving phase. In a network with  $n$  region, any node has  $n-1$  receiving time slots for each region and 1 transmission time slot. When node charges with enough energy, it is ready to participate in receiving packet. If it receives a packet in any of its time slots, it can distinguish the region which packet came from. Then if the region is further from the sink than its own region, it caches the data and wait till reach its own transmitting time slot. In transmitting time, it choose the next region with considering their priority for forwarding, in base of distance to sink and have enough energy. Then it will start to transmit.

## 3.4.6 Geographic Routing algorithm

In these three algorithms we should consider that each node knows its own location, by using the localization algorithm or by programming into nodes [37]. These algorithms are broadcast-based geographic routing algorithm.

### 3.4.6.1 Geographic Routing (GR)

This algorithm is also called with different name such as *directional geometric* and *location-based* [38]. The algorithm is based on two concepts: First all the nodes have information about geographic position of themselves and their neighbors. Second when source start to send a packet, it knows about the position of destination.

If a node which is more near to the sink than the sender receive a packet in its receive time;buffering the data and then at the end of receive time, if the channel is free, it will transmit the packet.

### 3.4.6.2 Geographic Routing with Duplicate Detection (GR-DD)

As it explain in [39], this algorithm is the extension version of **GR**, the difference is that, when a node receive a packet from a node which is located further from sink than it, it will check whether it received the same packet before or not. If it received the packet before, the packet is discarded to reduce unnecessary power consumption.

### 3.4.6.3 Geographic Routing with Duplicate Detection and Retransmission (GR-DD-RT)

The procedure of this algorithm is the same as **GR-DD**, it means that it works with utilizing the duplicate packet. But there is one part more in compare of **GR-DD**, in fact if a node is fully charge and it is in transmitting time, but there is no more new packet then it starts to resent the previously sent packet.

### 3.4.6.4 Comparison

From simulation of these algorithms, we can find that increasing the number of node in network has a direct effect on delivery ratio in **GR-DD**. Since the number of nodes in transmission range of each node increase, the number of node, which can receive a packet from sender, will increase. But in **GR-DD-RT** it's not the same. In this algorithm each node re-transmit the packet as many time as possible, so the energy consumption and the probability of collision goes high. As a result the delivery ratio decrease.

About the throughput of the network ( $T$ ) (the rate of data packet received by the sink including the duplicated packet) **GR-DD-RT** performs the best. Because all the nodes at the end of receive time try to transmit packet, it does not matter if they transmit a new one or re-transmit the old packet.

But throughput is not a good metric in this theory, because the sink doesn't accept duplicate one. So we should define goodput that shows the rate of unique packet receive by sink. In comparing goodput we can say that **GR-DD-RT**, in case of node density less than a given threshold, performs better than **GR-DD**. In efficiency, means the ratio of goodput to throughput (the probability of receiving unique packet by sink) the **GR-DD** algorithm is more powerful.

### 3.4.7 Distributed Energy Harvesting Aware Routing Algorithm (DEHAR)

As we know the goal of **WSN** is to have a long lifetime or continuous operation. To reach to this goal, it is important to find an optimize routing algorithm to harvest energy in the network. Here we present an adaptive routing algorithm, which can find optimize path from source to destination base on the information which it has from neighbor nodes [40].



All the nodes have information about the energy level of themselves and their neighbors. So it is possible to count the *local penalty* for each node which is inverse proportional to the energy level. We capture the global information in all nodes as *distributed penalty*. This distributed energy shows the cost (energy wise) of sending packet from that node to destination. At the end these local penalties and distributed penalties combine together and count as total penalty which is known to the node.

For routing a packet toward base station this algorithm consider the penalty of each node and then count the shortest path. Should take into account that shortest path means the path that consume energy less than other path.

### 3.5 Evaluation and selection of routing protocols

In the previous section we analyzed some protocols from literature. They are based on energy harvesting paradigm and their goal is to increasing the workload ,but some of them like **R-MPRT** in addition of increasing workload, also support the idea of sustainability. At the same time we found in literature there are more information about another algorithm which is **E-WME**. Our purpose in this thesis is to select some good routing algorithms and do some experiment on them. For choosing these algorithms we rely on literature. Literature is more focused on three routing algorithms and there are more information about them. So for doing our experiment on Routing protocols in **EH-WSN** we chose **E-WME**,**R-MPRT-mod** and **R-MPRT-org** algorithms as our candidate.



# Simulation

---

In previous chapter we chose three algorithms(**E-WME**,**R-MPRT-org**,**R-MPRT-mod**) as candidates to do analysis. Testing in real condition is not feasible so we have to use simulation to virtually model the networks and do experiment on them. We chose to use *Matlab* as simulation environment because it offers all the required tools and utility so we can focus our effort only on the simulation.

This chapter explores the *Network model*,*Routing model* and the *simulator architecture* has been designed for the analysis.

## 4.1 Network model

This section explores how network is modelled and implemented. The main goal is to implementing a random network which has some specific properties. A topology can be declared with the *net(num,a,range,nzone,seed)* command, where:

- *num*: Defines the number of nodes that are present in the network.
- *a*: Defines the area  $A \times A$  of our network.

- *range*: Defines transmission range for the node.
- *nzone*: Is used for defining the number of zone.
- *seed*: Each seed gives us a specific topology.

All of this information will be chosen by user.

### 4.1.1 Node

Network compose of different nodes. The number of nodes can be define by user, but the location should be choose randomly. This random place is produced by using a Matlab function which is called '*randi*'. By using this function we can generate random  $X$  and  $Y$  values between 0 and  $a$  which indicates the position of the node. These two lines show how to use *randi* function to create the random nodes.

```

1 Node(i).X = randi([0 , a] ,1)
2 Node(i).Y = randi([0 , a] ,1)

```

**Listing 4.1:** Random Node placement

Note that for the sink, defined as Node(1), the position is fixed to (0,0) for all the random topology.

### 4.1.2 Zone

Whole network area can be divided to different regions which are called *zone*. Zones groups a bunch of nodes and gives them some common properties, for example harvesting rate. The number of zones will be define by user and it is defined as a properties of the node. How to divide the whole area into different zones such a way that no overlap occurs was challenging. Finally by using the '*voronoi*' function in Matlab, the problem has been solved.

To work with '*voronoi*' function first we should produce some random seed for each zone. Each region seed has a random  $X$  and  $Y$  value that can be produced by function '*randi*'. Then *voronoi* decomposes the space around each seed(i) into a region of influence  $R(i)$ . The way that it divide the region is simple, it takes into account that all the points in one zone are more near to the seed of their zone than the seed of other zones.

The Figure 4.1 shows an example of how *voronoi* function by giving random seeds, divide the area into nine zones. Then by using the functions in Listing 4.2, we can find out which node belong to which zone.

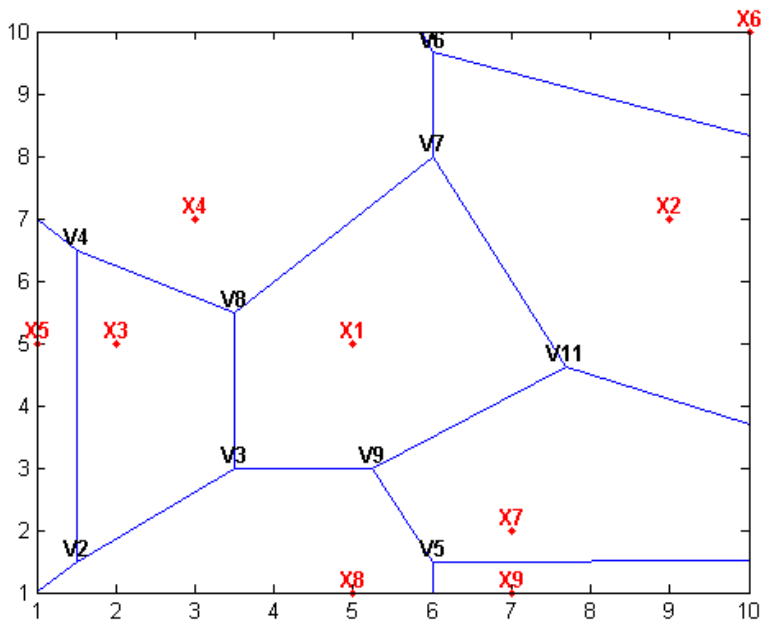
```

1 dt = DelaunayTri(X(:),Y(:))
2 pid = nearestNeighbors(dt,[X,Y])

```

**Listing 4.2:** Finding out the zone of each node

For example Node(8,2) belongs to zone 7.



**Figure 4.1:** Voronoi partitioning example

After plotting our network with different zones, we found out for some zones have no border, to address this issue we had to put some limits for our zones, this can be seen in the Listing A.6.

### 4.1.3 Link

A *link* is used to connect two nodes. If the distance between two nodes is less than the transmission range then we can create a link between them. Each link in our model is a bidirectional link, this is implemented with two one-directional link. Each node has at least one *inLink* and one *outLink*. Each link has its own properties such as *Cost*, *TXNode* and *RXNode*. Note that *Cost* properties is explained in routing section. For each link *TXNode* and *RXNode* show which node is transmitter and which node is receiver of that specific link.

Figure 4.2 shows a basic network model with some *Nodes*, *Links* and *Zones* and the relation between these elements.

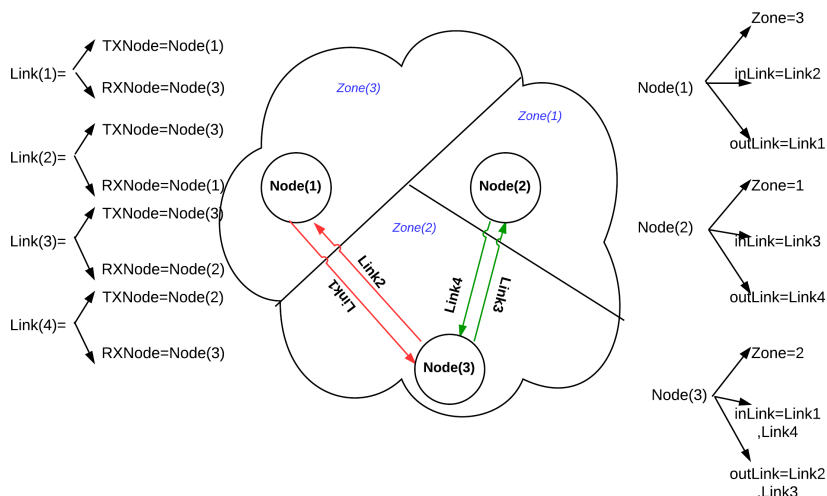


Figure 4.2: Example of network model

## 4.2 Routing algorithm model

We intend to analyze how routing protocols behave in different working condition. For modeling our routing algorithms first we defined *Cost* to each link. Then by using some algorithms, found the shortest path cost from each node to the sink. *Cost* is different in the three candidate routing algorithms.

### 4.2.1 Cost function

For computing *cost* function in algorithms, should have some knowledge about *packet energy*, *available energy* and *harvesting rate*.

#### 4.2.1.1 Harvesting rate

Harvesting rate of each node shows the rate of harvested energy for that node, this varies for different nodes in function of the zone to which they belong. Each zone in network has random value of harvesting rate. So all the nodes are located in one zone has the same harvesting rate. We assumed that harvesting rate for sink is infinitive.

#### 4.2.1.2 Packet energy

Packet energy defines the energy required for sending data to node  $n$ . As we explained in subsection 1.2.2, in asynchronous **MAC** protocol which is suitable for **EH-WSN**, a node with some data to transmit always wait for a beacon to start the transmission toward the beacon source.

Node needs to wait for receiving beacon, this waiting time is called "Idle-Listening". For sure in this period nodes need some energy to survive. This energy can be calculate by using the following statement:

$$IdleListening = (Node(n).BeaconPeriod / 2) * Prx$$

**Listing 4.3:** Idle\_Listing Cost

Another factor which affects the packet energy is *TransmissionCost* which is the energy needed to send a packet, this quantity can be computed by

$$TransmissionCost = (L/R) * Ptx$$

**Listing 4.4:** TransmissionCost

$Prx$  and  $Ptx$  are properties of our transceiver which are explained in section 4.3. So the total formula for counting *packet energy* is:

```
1 PE = TransmissionCost + IdleListening
```

**Listing 4.5:** Packet energy Cost

#### 4.2.1.3 Available energy

This quantity shows the amount of energy that each node has before processing the packet. This parameter is simulation related and will be explained better in Section 4.3

#### 4.2.1.4 Cost computation

The formula for counting cost function is different in three algorithms.

- **Cost in E-WME algorithm:** The cost for this algorithm compute in this way:

```
1 cost = (cap / ((Net.Node(Net.Link(i).TxNode).HR + epsilon) *
    log(mu))) * (mu . ^ ((cap - Net.Node(Net.Link(i).TxNode).AE) /
    cap) - 1) * (Net.Node(Net.Link(i).RxNode).PE);
```

**Listing 4.6:** E-WME cost

This cost function is written by using some metrics such as: harvesting rate of transmitter, battery capacity, available energy, packet energy,  $\mu$  and  $\epsilon$ , the last two are constant.  $\mu \gg 1$  is acceptable and  $\epsilon$  is small positive.

- **Cost in R-MPRT-org algorithm:** In this algorithm we define the cost function by using the packet energy and harvesting rate of transmitter.

```
1 cost = (Net.Node(Net.Link(i).RxNode).PE) / (Net.Node(Net.Link(i).TxNode).HR)
```

**Listing 4.7:** R-MPRT-org cost

- **Cost in R-MPRT-mod algorithm:** This algorithm is the modified version of previous algorithm. Instead of using harvesting rate of transmitter, it uses available energy of transmitter.



```

1 cost = (Net.Node(Net.Link(i).RxNode).PE) / (Net.Node(Net.Link
      (i).TxNode).AE)

```

**Listing 4.8:** R-MPRT-mod cost

### 4.2.2 Path cost

After setting *Cost* to each link, we can compute the minimum path cost from each node to sink by using a recursive path cost function. In this function assumed that path cost of sink is zero, whenever a node receive beacon from other node recompute the minimum path cost to the sink. The complete explanation is in the section 3.4.2.

## 4.3 Simulator Architecture

In previous steps we described a way to model a network and defined how to compute path cost from each node to the sink. This section reports simulator architecture used in this thesis work and the transceiver features that we used in our simulation.

The simulator should be configurable in such a way to analyze some metrics which will be use later for experimental purpose, these metrics are:

- *Throughput(Packets per second)*
- *Collision rate*
- *Lifespan(Seconds)*

Simulator works base on the concept of *event driven*. For each simulation gives a set of random event . This event can be mainly three types:

- *Data forwarding*
- *Beacon transmission*
- *Data transmission*

In simulator function first we initialize the network and variables After that we initialize the simulation parameter like; condition when the simulation should stop and when the routing path should be recompute.

Simulator stops to work in two cases: if it reach to the maximum time of simulation or one of the node die. We configure that node die when the available energy of a node goes below of zero.

To update the routing path,update threshold has been defined so whenever the available energy of battery goes below of this threshold the routing algorithm recompute the path.

### 4.3.1 Transceiver

Each node in **WSN** has different component, Micro controller unit (**MCU**), a *storage unit*, Analog to digital converter (**ADC**), *power manage unit* and Radio-Frequency (**RF**). These component are shown in Figure 4.3.

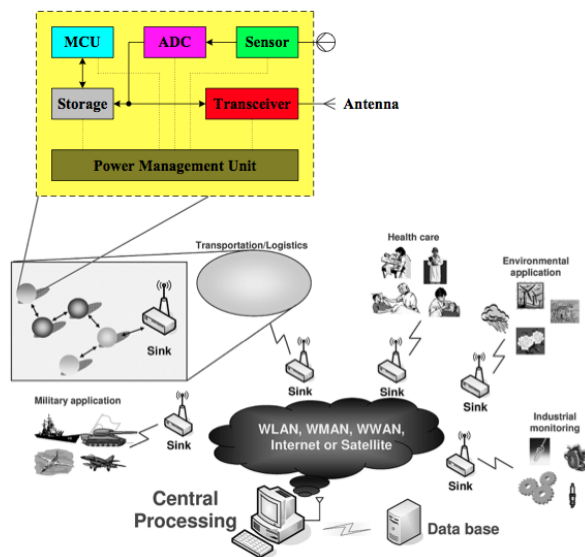


Figure 4.3: WSN Components

The important part of **WSN** node is the *RF Transceiver*. The reason of using transceiver is to realize the wireless communication among the nodes.

In our simulation we suppose to use the **CC2500** (Figure 4.4) which is a 2.4GHZ transceiver. It is designed for very low power wireless application. It has different features such as Radio frequency performance, Analog and Digital controls and low power design. The simulator will use some parameter of this transceiver such as :

- $R = 500 * 1024$  Kbps
- $P_{tx} = 0.0538$  watt
- $P_{rx} = 0.0425$  watt



Figure 4.4: CC2500 Transceiver

### 4.3.2 Simulation parameter

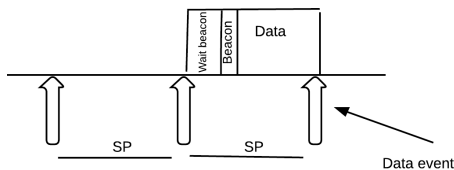
To create different simulation configuration the simulator defines different parameters that are listed below.

As explained before the simulation is event driven. Each of these event has their own firing time, this time shows the time that event should start. To calculate this event time we use simulation time and the concept of sense period and beacon period.

#### 4.3.2.1 Sense Period

Node sends data event after each sense period. Sense period includes the period that node should wait for the beacon, receive beacon, send data and then go back

to sleep.



**Figure 4.5:** Sense period

Sense period controls how much traffic we generate for every  $mW$  of power harvested. This period for each node is inversely proportional to the harvesting rate of that node. As much as the node have more harvesting energy, the period of packet generation and transmission decrease.

Sense period formula in our assumption as it is shown here, is composed of two variables:  $B$  shows how often transmit data for each  $mW$  of power harvested;  $A$  defines an offset to the sense period of each node:

$$Sp = A + (B/Node(n).HR)$$

**Listing 4.9:** Sense period

Note that  $A$  is the parameter that affects all the network, instead  $B$  is related to each node, so changing  $A$  will affect the total amount of produced data in the network meanwhile changing  $B$  will affect only the produced data for each node <sup>i</sup>.

#### 4.3.2.2 Beacon Period

It shows how often a node should send beacon. It is made by the parameter  $C$  over the harvesting rate. For changing the beacon rate of each node the only possibility is to changing parameter  $C$ .

$$BP = C / (Node(n).HR*1000)$$

**Listing 4.10:** Beacon period

<sup>i</sup>HR is group related variable.

### 4.3.2.3 Others parameters

Other parameter for the simulation are:

- **Num**:number of nodes
- **Range**:Node transmission range which in our simulator is 400 meters.
- **a**:Field area which in our test is 1000 meters.
- **nzone**:Define the number of zone which is 20.
- **SP**:Sense period for each node is used to show the traffic of the network and it composed of variable  $A, B$  and harvesting rate of the node .
- **BP**:It shows the frequent of sending beacon in network and it composed of variable  $C$  and harvesting rate of each node.
- $\epsilon$ : This constant use in calculating cost function of **E-WME** routing algorithm with the value of two.
- $\mu$ : Which is a constant in cost function of **E-WME** routing algorithm with the value of three.
- **Ptx**:Transmit power of transceiver with the value of 0.0538 watt.
- **Prx**:Receive mode power is 0.0425 watt.
- **R**:Maximum data rate up to  $500 * 1024$  Kbps in transceiver.
- **B**:The length of beacon can be  $8*8$  bits.
- **L**:Length of data is defined to  $16*8$  bits.
- **Battery capacity**:Is 1.476 Joules.
- **Seed**:is a number used to initialize a pseudo random number generator. Any seed gives us a new topology.

### 4.3.3 Event flow

Using the concepts of Beacon period and Sense period is making possible to define the event timing. Beacon event time for the first generation of event is:

```
1 Event.Time = Net.Node(i).BeaconPeriod*rand
```

**Listing 4.11:** First beacon event time

for data event:

```
1 Event.Time = Net.Node(i).SensePeriod*rand
```

**Listing 4.12:** First data event time

For each node, both the event time is a random value related to Sense period and Beacon period. All the nodes make their events and put them in the waiting list. The priority of this queue is event time, any event that has smaller event time start first.

Every time the event happens, a new event of the same type is added to the queue with new random event time refer to the current simulation time, beacon period and sense period. The process of adding new event to the queue give us possibility to always have some event to simulate.

In every *Beacon event*, simulator compute the time passed from the last event till current one in the same node. This harvest time is used to compute the harvest energy in that node in this way:

```
1 HarvestEnergy = (Net.Node(i).HR * value)/1000
```

**Listing 4.13:** Harvest energy computation

At the end this harvest energy over the total battery capacity add to the current battery level to compute the available energy in the energy storage device.

```
1 Net.Node(i).AE = min(1, Net.Node(i).AE + HarvestEnergy /  
    BatteryCapacity)
```

**Listing 4.14:** Available Energy computation

Note that maximum value of available energy is one. Available energy shows if we have enough harvested energy for continue or not. Every time that a new beacon event arrives or data wants to transmit, the available energy is computed, if this value is less than zero the simulation stops because the node is died.

If the available energy support beacon event, the node that generated the beacon event check how many nodes use this beacon source node as next-hop. If this value is equal to one it means that only one node wants to forward data through the "beacon's node", so we have a normal transmission. Instead, if this value is greater than one, multiple nodes want to use it as next hop so we have a data collision.

If the receiver of data is not the sink the simulation put an event to the queue to forward this data toward the sink. As we know next hop can be calculate from recursive path cost function.

When *Data event* arrives it says which time the data generated and ready to be transmitted and, it highlights which node is the originator of this data. Every time that a *Data event* is captured the available energy for the event node will be check, if it is less than the update threshold the path cost is recomputed.

The simulation infrastructure which is proposed, as can be seen in next chapter , fully address all the needing on a complete and a reliable testing for our routing algorithm candidates.





# Simulation Runs

---

Currently three routing algorithms are chosen for analysis. Since the idea of **EH-WSN** is new, the experiments has been planned step by step. After each step, the data are analyzed to plan the next step. Therefore each step gave a new idea on how to continue with experiment.

## 5.1 Analysis metrics for simulation runs

The simulator was prepared to show some analysis metrics. These metrics can be classified in three groups.

- **Throughput:** Which present the number of packets received by the sink per second.
- **Lifespan:** This metric present the life time of network.
- **Collision rate:** It expresses how many transmitted packet collide over all the transmission.

The analysis parameters are totally dependent to the configuration of the network, therefore by changing the network configurations we can see how these metrics will change in each of the three algorithms.

Our interest is to figure out how parameters like beacon rate, different data traffic and number of nodes affect the analysis metrics. The data traffic can be changed tuning the sense period. Sense period is the total period that each node wait for the beacon, receive the beacon and send data(see section 4.3.2.1).

Beacon period defines how frequent a beacon is sent (see Section 4.3.2.2).

## 5.2 Simulation flow

In the following section is reported how we tuned all the simulation parameters in different scenarios to analysis the performance of analysis metrics. In the following experiments some parameters are set as a default such as:

a(meter)	Range(meter)	nZone	$\epsilon$	$\mu$	Ptx(watt)
1000	400	20	2	3	0.0538
Rrx(watt)	R(Kbps)	B(bits)	L(bits)	Battery Capacity (J)	•
0.0425	500*1024	8*8	16*8	1.476	•

**Table 5.1:** Constant parameters

### 5.2.1 Different topology

This subsection explores some experiments on different topologies to analysis the performance of analysis metrics and take some logic conclusion for further experiments.

#### 5.2.1.1 Topology experiments

In this part we want to analyze *Throughput* and *Lifespan* metrics of algorithms in different topologies.

1. **Throughput analysis in different topologies:** In this experiment we focused our attention on the comparison of three routing algorithms, trying to find out which algorithm gives the highest average packet per second. We run the simulation with fix configuration but in different topologies.

In this experiment we used the constant parameters of simulator which is reported in table 5.1 and other variables which are mentioned here:

- Num = 100
- C = 600
- A = 0
- B = 20

We took the results of different topologies such as [30,40,1,10,20,42,56,17].

As you can see in Figure 5.1 is not easy to find which algorithm has the highest throughput because there is no stable trend in different topologies. For instance in topology (42) *R-MPRT-mod* algorithm has the highest throughput, but in the same condition in another topology like (40) *R-MPRT-org* has the highest packet rate.

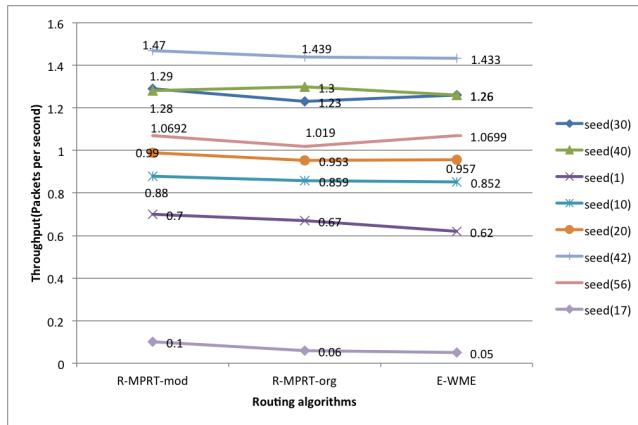


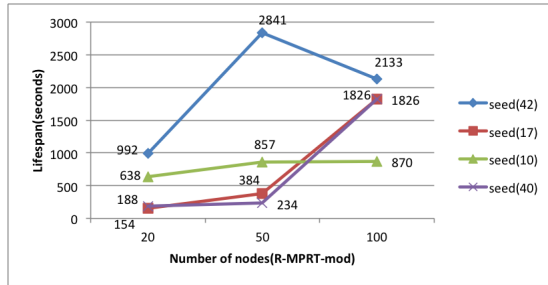
Figure 5.1: Throughput of algorithms in different topologies

2. **Lifespan analysis in different topologies:** Here we tried to figure out how increasing the number of node could be affect on the lifespan of network. We did the experiment with different topologies, such as [40,10,42,17], with this simulation parameters:

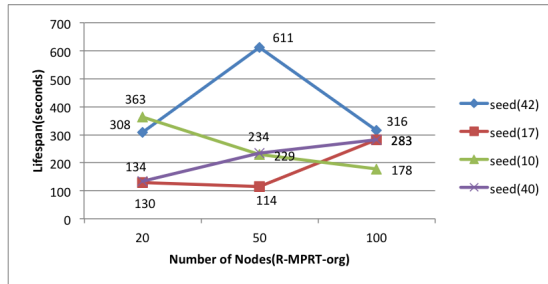
- Num = [20,50,100]
- C = 600

- $A = 0$
- $B = 20$

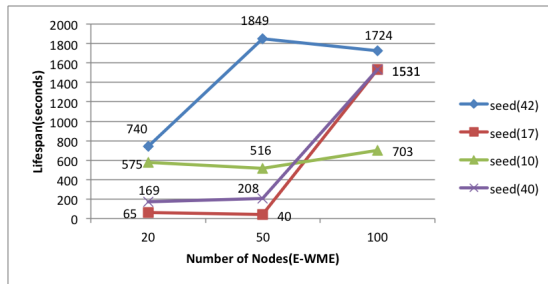
As it is shown in Figure 5.2; In all candidate routing algorithms with increasing the number of node in different topologies, the end time of simulation doesn't follow a constant trend.



(a) R-MPRT-mod algorithm



(b) R-MPRT-org algorithm



(c) E-WME algorithm

Figure 5.2: Lifespan in different topologies with increasing number of nodes

## 5.2.1.2 Topology classification

From the previous experiments we can see that analysis metrics do not have the same behaviour in every topology. More in depth, from the first experiment we have seen that topology(17) has a significant lower throughput respect to the average of other topologies, so we have decided to categorize these topologies into two groups:

- **Good topology:** Out of 50 topologies that we tried, topologies like (40) was the most common case as a good topology. In this kind of topology there are some area with low energy and some area with high energy as it is clear from the color of region in Figure 5.3. The algorithms can find a path through red area to reach to the sink. So by passing the routing from red area the network will be alive more.

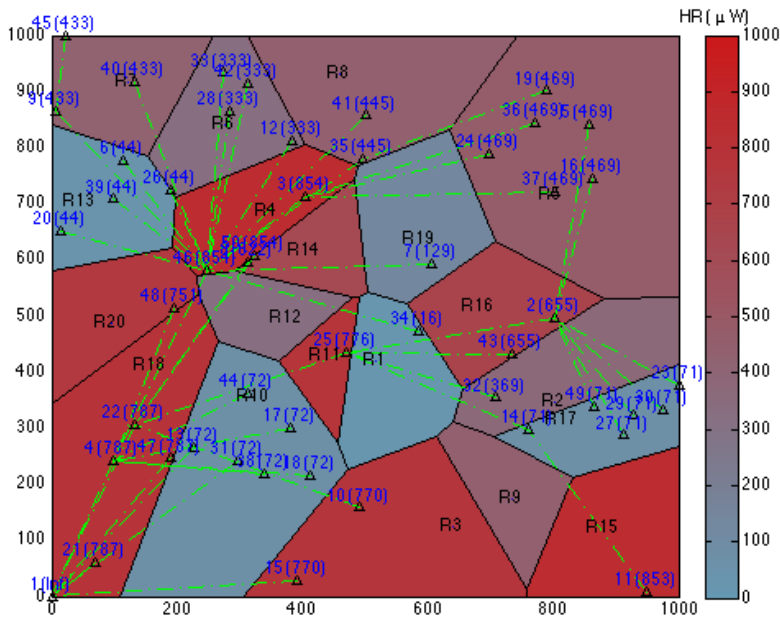


Figure 5.3: Example of good topology

- **Bad topology:** Topology (17) is one of the example of bad topology. As we can see in Figure 5.4 there is a big region close to the sink that has low energy and the only way for all the nodes reach to the sink is to

pass from that region. This interface (Node(10)) because of having low harvesting rate, transmits beacon in a lower frequency. So other nodes who are waiting for this beacon, should wait a lot for receiving this beacon. Eventually their energy will be finish in this waiting period and network will be die soon.

It's obvious that, as much as node is close to the sink needs more energy. In topology(17) the area with low energy to the sink rise the problem.

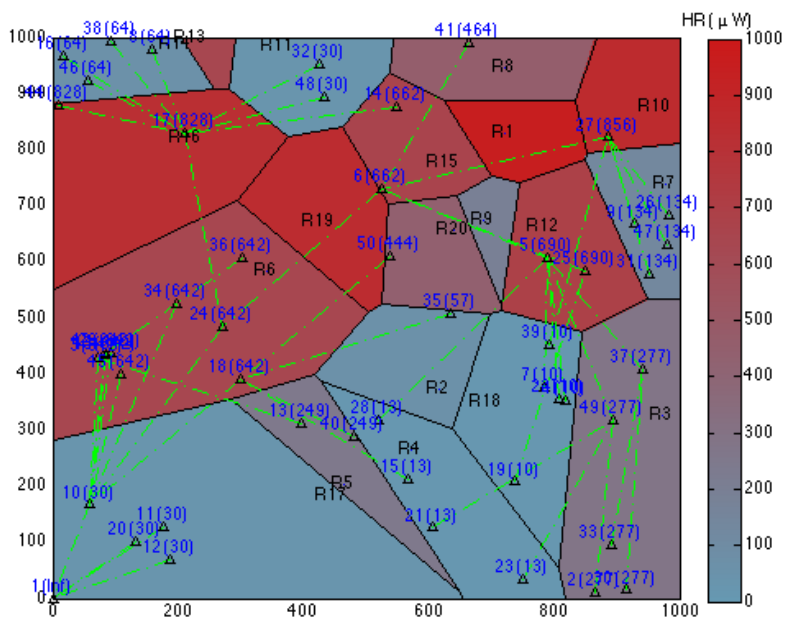


Figure 5.4: Example of bad topology

### 5.2.1.3 Results

For further analysis we have decided to concentrate our effort on good topologies and to make the results more reliable, compute the metrics using the average of different good topologies.

### 5.2.2 Different number of nodes

The purpose of this experiment is to compare three algorithms in the case of increasing number of nodes. We took the following simulation parameters to our test in addition to some parameters which are fixed :

- Num =[50,60,70,80,90,100]
- C = 600
- A = 0
- B =20

Note that we did this experiment with using 50 different topologies for each number of node. We didn't do experiment with less than 50 nodes because there is a high chance that network be disconnected.

#### 5.2.2.1 Results

By doing this experiment we got some results that explained separately with their reasons.

- **Throughput :**

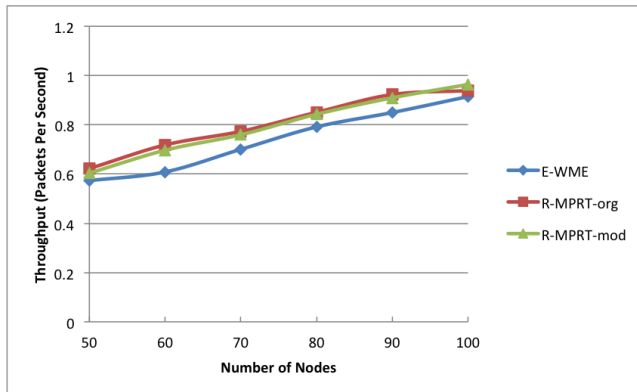


Figure 5.5: Average throughput in different number of nodes

The Figure 5.5 shows the average throughput of three algorithms in the case of different number of nodes. All three algorithms have increasing trend in their throughput. The reason is obvious because by increasing the number of nodes the probability to have more data packet will be increased.

Another point which is clear is that the **E-WME** algorithm has lower packet rate in compare of other algorithms. To find the reason, we should put our attention on the graph topology of this algorithm. In all the graph topologies for **E-WME** algorithm, connection assumes star shape which means many nodes use one node as their interface in the way toward the sink.

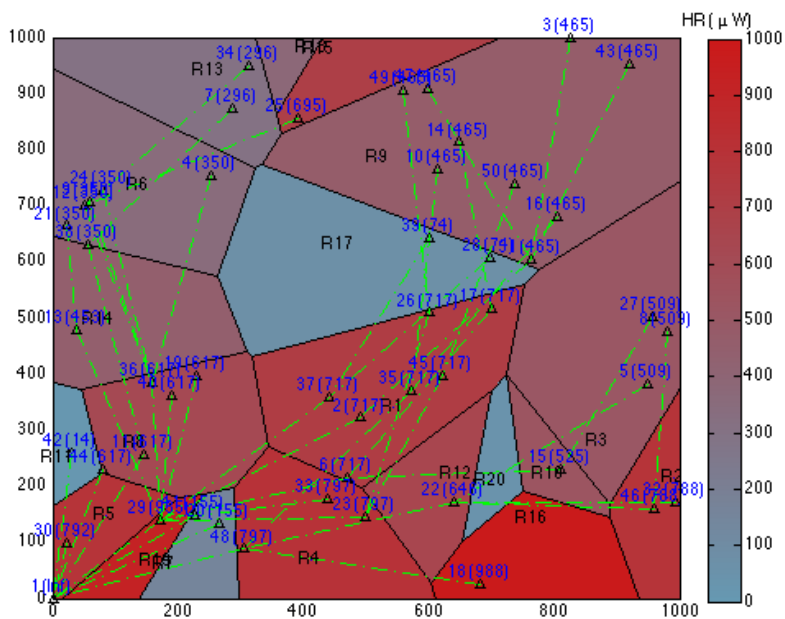
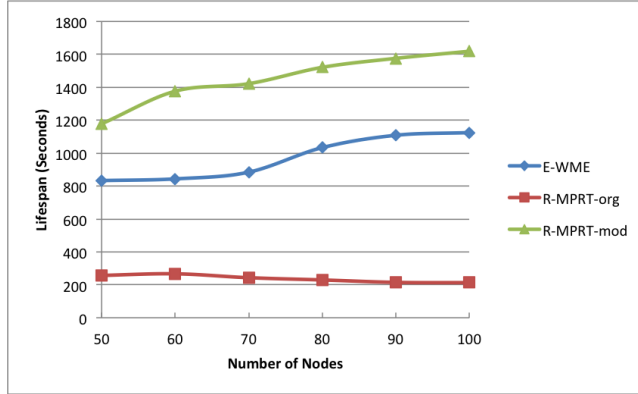


Figure 5.6: E-WME algorithm

In Figure 5.6 node 29 have to send the data received from many nodes to the sink. So we guessed that this star shape is the reason because the **E-WME** algorithm has lower throughput than other algorithms. Because with sending data from different nodes to one node the collision rate increase and as a result the packet rate will be decrease.

- Lifespan:





**Figure 5.7:** Average lifespan in different number of nodes

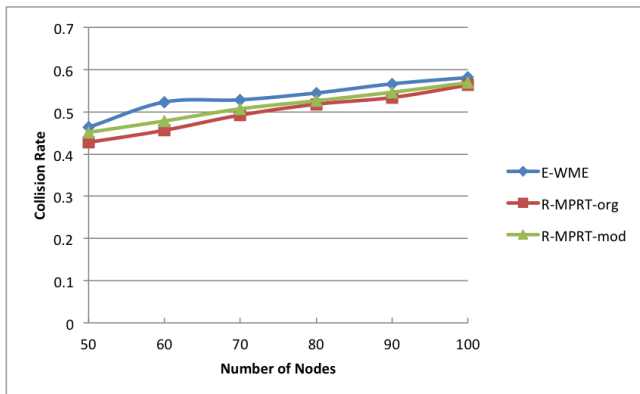
From the Figure 5.7 we figured out that the network which use *R-MPRT-mod* algorithm has a bigger average lifetime in compare of *R-MPRT-org*. The reason of this behaviour can be explained in this way: The two algorithms basically are the same, their difference is in the way they define cost for each link. *R-MPRT-org* algorithm use harvesting rate concept. So when the recursive path cost function wants to choose the minimum path cost from each node to the sink, it will choose the node with biggest harvesting rate as forwarder. In our simulator we have configured that whenever the available energy of a nodes, which is in data event, goes lower than threshold the routing path will be update. But because the *R-MPRT-org* algorithm just care about harvesting rate and harvesting rate of each node is constant, the next path will be the same as previous path. In this way after finishing the harvested energy of one of the node, the network will die.

Meanwhile *R-MPRT-mod* algorithm uses the available energy of each node for defining the cost function. As soon as the available energy of data event node become lower than a simulation threshold, the path cost function choose another path, choosing the node with more available energy as a forwarder. So the node with low available energy have time to replenish energy. Therefore the life time of network with *R-MPRT-mod* algorithm is more than the network with *R-MPRT-org* algorithm.

Another interesting point is that the average lifespan of *R-MPRT-mod* and **E-WME** algorithms are improved by increasing the number of nodes. When we enhance the number of nodes it means to add more battery, more available energy and more routing option to the network. These two algorithms take into account the routing options, as much as this options increase the possibility that network stay alive will increase.

The lifespan for *R-MPRT-org* algorithm is more or less is constant because with increasing the number of node, actually the route option is increasing but this algorithm always use one routing path. As a result the lifespan more or less doesn't change a lot.

- **Collision Rate:**



**Figure 5.8:** Average collision rate in different number of nodes

About collision with respect to Figure 5.8 we concluded that with increasing the number of node in network, there are more nodes that wants to send data. So as a result there are more chance that two nodes decide to send data to one destination, and the average collision rate will increase.

### 5.2.3 Different traffic

Node sends data event every sense period, more information on how this parameter has been modelled can be found in 4.3.2.1.

The goal in this experiment is to analyze the behavior of algorithms on different traffic level, by changing  $A$  and  $B$ .

#### 5.2.3.1 Changing $A$ for good topologies

In the first experiment about sense period we choose  $B$  as a constant and tried with different value of  $A$ . In this way we are able to change the data traffic in whole the network.

So we configured our simulation parameters for the first experiment with  $A$  in this way:

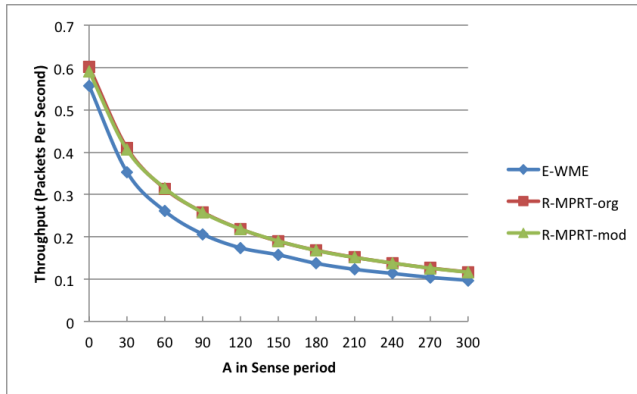
- Num =50
- $C = 600$
- $A = [0,30,60,90,120,150,180,210,240,270,300]$
- $B = 20$

Taking into account that this experiment has done with taking 100 different topologies.

### 5.2.3.1.1 Results

By increasing the  $A$  we understood the following points:

- **Throughput:**

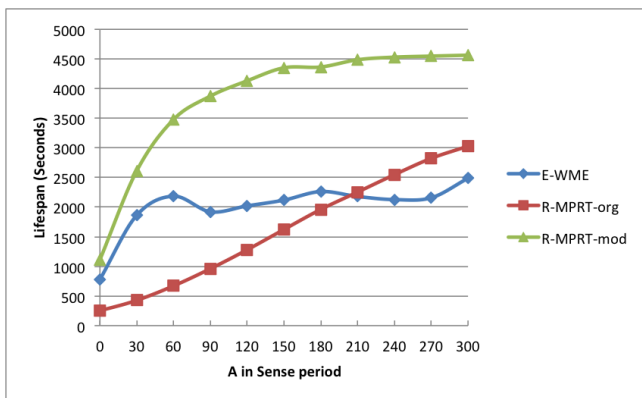


**Figure 5.9:** Average throughput in different data traffic by changing  $A$

As it is clear in Figure 5.9 by increasing the  $A$  the sense period increase, so the average probability of sending packet decrease.  $A$  is lower bounded to zero, with this value the network produces the highest traffic at network level. Note that if we want to increase the traffic furthermore we should modify  $B$  but this is evaluated in section 5.2.3.3.

Like the previous experiment, **E-WME** algorithm has lower throughput than other algorithms. The reason completely explained in *Throughput* part of section 5.2.2.1.

- **Lifespan:**



**Figure 5.10:** Average lifespan in different data traffic by changing A

By looking at Figure 5.10 we realized that in the case of decreasing traffic, *R-MPRT-mod* algorithm has bigger average lifetime in compare of other algorithms because of using available energy in its cost function. The further explanation is available in the *Lifespan* part of section 5.2.2.1.

With focusing on the behavior of *R-MPRT-org* and **E-WME** algorithms we understood that in a high traffic **E-WME** algorithm has longer average lifetime than *R-MPRT-org*. This trend will change by going to the area with low traffic. In low traffic network *R-MPRT-org* will be alive more than the **E-WME** algorithm.

- **Collision Rate:**

The Figure 5.11 shows that by decreasing the traffic the average collision rate will decrease. In **E-WME** the average collision rate is more than other algorithms, guess because of star shape paths.

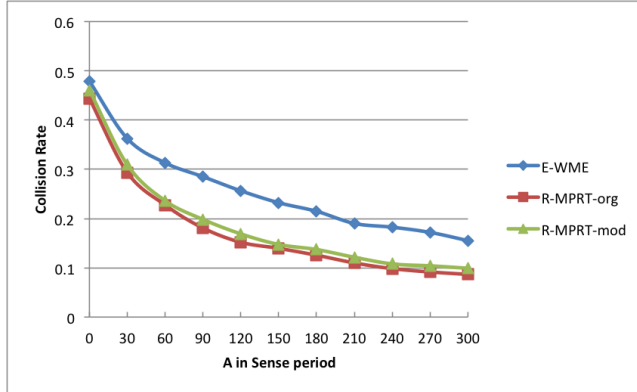


Figure 5.11: Average collision rate in different data traffic by changing A

### 5.2.3.2 Changing A for bad topology

As we proved in topology classification of 5.2.1.2 , topology(17) is one of the example of bad cases which will die fast.

In this part we put our effort to understand if decreasing the traffic, will help bad topology to live more or not.

The simulation parameters for this experiment are:

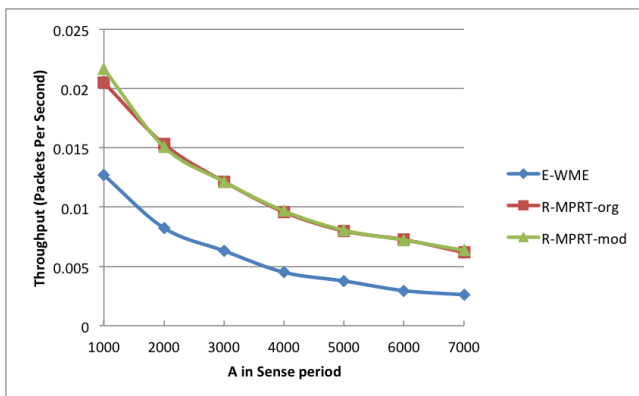
- Num =50
- C = 600
- A = [1000,2000,3000,4000,5000,6000,7000]
- B =20

We made 100 simulations on topology(17).

#### 5.2.3.2.1 Results

By doing the experiment on topology(17) in a very low traffic we found out some points:

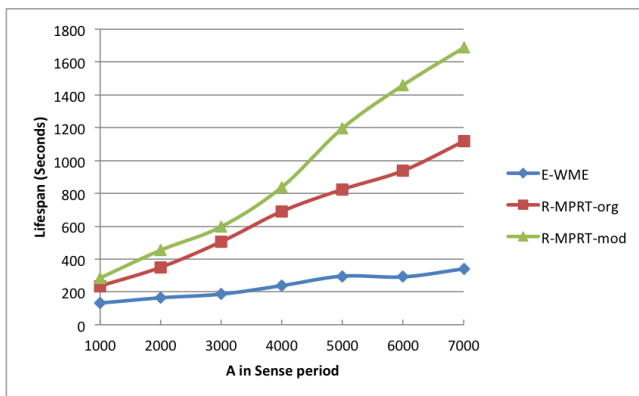
- **Throughput:**



**Figure 5.12:** Average throughput of topology(17) in very low traffic

With too much decreasing the traffic the rate of receiving packets in sink is decreasing slowly.

- **Lifespan:**

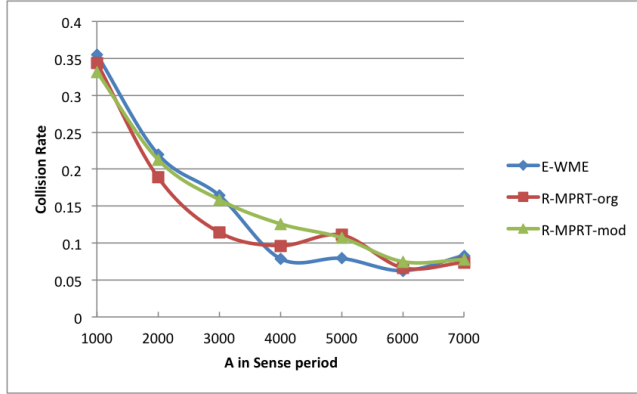


**Figure 5.13:** Average lifespan of topology(17) in very low traffic

In bad topology network dies fast so for solving this problem decided to decrease the traffic and from Figure 5.13 it seems in a really low traffic network can work and stay alive.

In good topologies as explained in lifespan part of 5.2.3.1.1, in low traffic *R-MPRT-org* has bigger average lifetime in compare of **E-WME**. This behavior in a really low traffic of bad topology works significantly better.

- **Collision Rate:**



**Figure 5.14:** Average collision-rate of topology(17) in very low traffic

Decreasing the traffic let networks to have less collision rate.

### 5.2.3.3 Changing B

In previous test we wanted to analyze how changing the sense period for all the network impacts on the routing algorithms behavior, in this experiment we want to analyze the algorithms behavior, respect to changing the sense period for each group of nodes. This can be done by changing the value of  $B$  in the Sense Period formula. In this experiment  $A$  will be fixed to zero and  $B$  will change.

The configuration of our network is :

- Num =50
- C = 600
- A =0
- B =[5,10,15,20,25,30,50,100,150,200,250,300]

The number of topologies used in this experiment is 50.

5.2.3.3.1 Results

As it was simple to imagine the change of  $B$  has more or less the same results of changing  $A$ , so anytime that  $A$  and  $B$  are increased the total traffic on the network decreases. For sake of completeness a categorized result analysis is reported:

- **Throughput:**

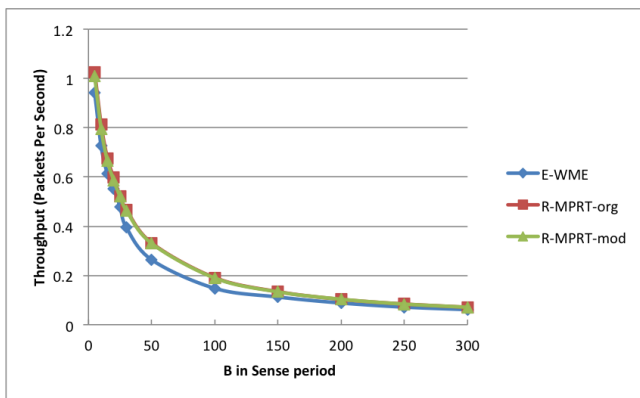


Figure 5.15: Average throughput in different data traffic by changing B

With decreasing the traffic the packet rate is decrease for all algorithms.

- **Lifespan:**

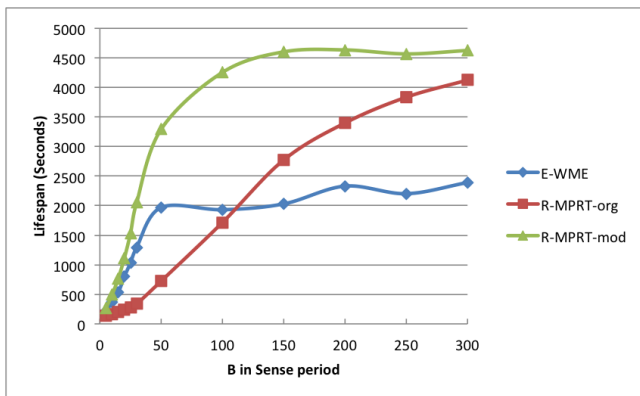
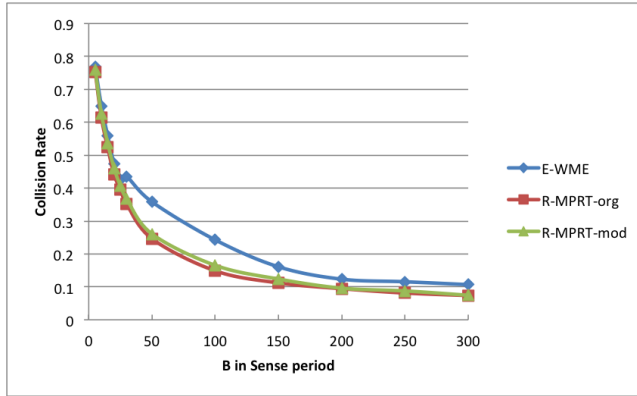


Figure 5.16: Average lifespan in different data traffic by changing B



As much as the sense period increase, the number of sending packet decrease. So the network have more energy to be alive. As it is clear from Figure 5.16 in low traffic *R-MPRT-org* algorithms works better than **E-WME** in the sense of lifetime. In high traffic **E-WME** can be alive more than *R-MPRT-org* algorithm.

- **Collision Rate:**



**Figure 5.17:** Average collision rate in different data traffic by changing B

It's obvious that less traffic bring less collision.

### 5.2.4 Different beacon rate

In this step of our test, the goal is to verify the behavior of algorithms in the case of different beacon rate. Beacon rate shows the number of beacon which is transmitted per second.

By changing the parameter  $C$  in beacon period formula; the beacon period can be controlled.

$$BP = C / (\text{Node}(n).HR * 1000)$$

In this experiment our assumption was:

- Num = 50
- $C = [1, 10, 20, 30, 40, 50, 60, 80, 100, 120, 140, 160, 200, 400, 600, 800, 1000, 2000]$

- A =0
- B =20

We took the average result of 50 topologies.

### 5.2.4.1 Results

Considering different beacon rate had some affected in our metrics such as:

- **Throughput and Collision Rate:**

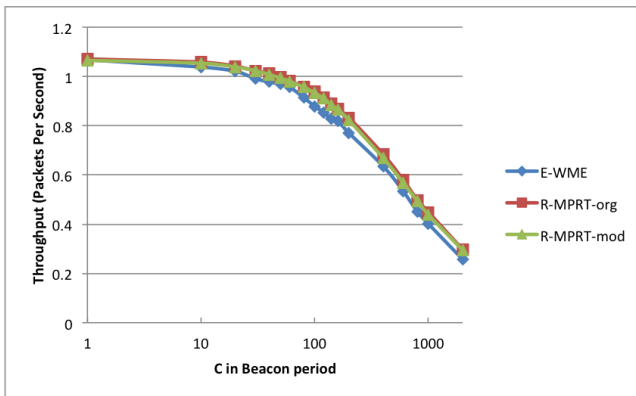


Figure 5.18: Average throughput in different beacon rate

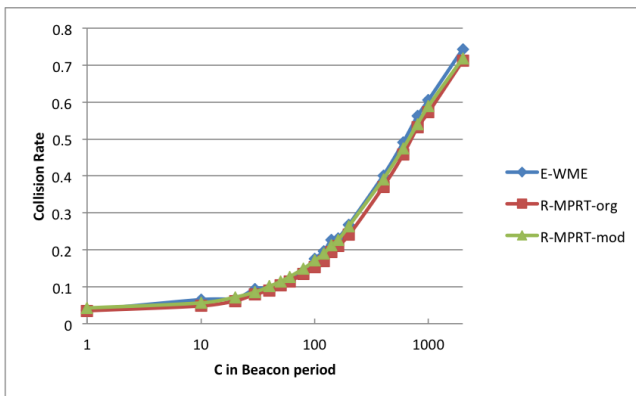
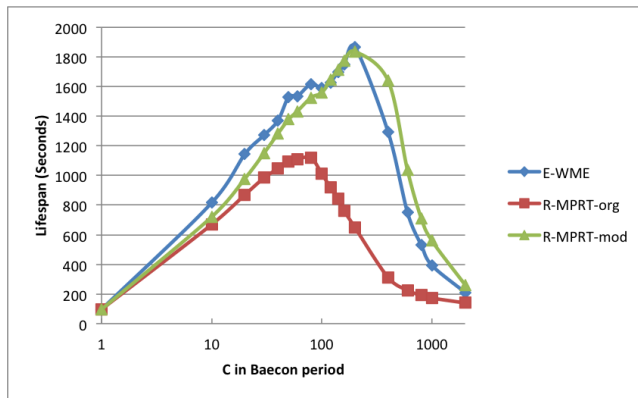


Figure 5.19: Average collision-rate in different beacon rate

Increasing the parameter  $C$  makes the process of sending beacon slower. So the probability that two or more data wants to response to one beacon increase. As a result the average collision rate will increase and because many packet collide in the way the average number of packet which are received by sink decrease.

The trend of  $R$ -MPRT-mod and  $R$ -MPRT-org is more or less equal. **E-WME** has lower average throughput and higher average collision rate in compare of other algorithms.

- **Lifespan:**



**Figure 5.20:** Average lifespan in different beacon rate

In this experiment the goal is to analyze the average lifespan behavior in fix traffic but with different beacon rate. As it explain in [41] in the scenario of having long beacon period the nodes which are waiting for beacon should wait more;therefor wasting more energy in the idle listening. So the energy consumption will increase and then the lifespan of network decrease.

In the other hand,by decreasing the beacon period we have more frequent beacon,the network consume a lot energy because of sending beacon,so again the network die fast.

In the middle we have the best configuration of beacon period;the network doesn't waste too much energy for receiving or sending beacon. Therefore we have the longest lifespan.



# Conclusion

---

Routing in sensor network is a new area of research. This thesis explored a comprehensive survey on the energy constraint routing techniques in wireless sensor network that are present in the literature. The investigation brought to the idea of having different classification of routing algorithms in **WSN**.

By doing survey on the energy-efficient routing algorithms category; it has been clear that common aim in these techniques is to extend the life time of the sensor network without compromising on data delivery. These routing algorithms have been classified in *flat, hierarchical, query-based, coherent, non-coherent based, negotiation based, location based, mobile agent-based, multi path-based and QoS-based*.

In another category that take into account the energy harvesting idea; the main goal for routing algorithms is to maximize the workload in the energy-harvesting network.

This thesis was more involved in the concept of routing algorithms for **EH-WSN**. By looking in literature it was understandable that the number of available routing algorithms in **EH-WSN** is limited in compare to **WSN** because it is a new topic. A total of seven **EH-WSN** protocols was found in literature such as *:R-MF, R-MPRT, R-MPE, EHOR, DEHAR, Geographic routing algorithm*. Among all these algorithms there were two algorithms (*R-MF, R-MPRT*) which

have been also evaluated in the area of energy sustainability. Choosing some candidates for further experiment was challenging. Finally three algorithms (*R-mprt-mod*, *R-MPRT-org*, *E-WME*) were chosen for further analysis because they are the one with the higher trend in researches and publications.

The design and implementation of simulator fully satisfied the requirement for correct simulation of the candidate algorithms.

Different analysis metrics could be evaluated by means of different scenario simulation. For instance in the scenario of different number of node with increasing the number of nodes the throughput and collision rate increase. The lifespan of *R-MPRT-mod* and **E-WME** algorithms increase meanwhile this trend for *R-MPRT-org* is some how constant. In different traffic scenario; as much as the traffic increase the throughput and collision rate will decrease; but the life time in all algorithms improve except the **E-WME** algorithm that has weird behavior. Decreasing the beacon rate in the constant data traffic have direct proportional effect on packet rate but the trend of collision rate is in the opposite and with decreasing the beacon rate, will increase. The life time in this scenario shows different behavior with different amount of beacon rate and the trend is not monotonic.

Further work should be done to improve the analysis for bad topology, maybe looking for some scenario in which the algorithm behave in a correct manner. Some other work can be focused on a quantitative analysis of proposed routing algorithms; to find out the reason of fuzzy behavior of some routing algorithms in some scenario. Understanding these behaviors and the reason of them can be the starting point for modifying algorithm toward the optimum case.

## APPENDIX A

# Code

---

```
1 function BatchSimulation()  
2  
3  
4 Cs = [1,10,20,30,40,50,60,80,100,120,140,160,200];  
5 As = [0];  
6 Bs = [20];  
7 rAlgos = {'R-MPRT-mod'; 'R-MPRT-org'; 'E-WME'}  
8 nodes = [50];  
9 seeds = [1,2,5,10,20,30,35,40,42,56];  
10 nOfAlgos = size(rAlgos,1);  
11 rAlgos(2,1)  
12 %UNTITLED3 Summary of this function goes here  
13 % Detailed explanation goes here  
14 fprintf(1, 'Batch Simulator\n');  
15 for i=1:nOfAlgos  
16     rAlgo = rAlgos(i,1);  
17     for nNode = nodes  
18         for seed = seeds  
19             for A = As  
20                 for B = Bs  
21                     for C = Cs  
22                         fprintf(1, 'Simulate algo : %s nodes : %i  
seed : %i A: %i B: %i BP: %i\n', char(  
rAlgo), nNode, seed, A, B, C);  
23 Net = simulator(seed, nNode, char(rAlgo), A, B,  
C);  
24 filename = sprintf('net_%s_N%i_S%i_A%i_B%  
i_BP%i.mat', char(rAlgo), nNode, seed, A, B,  
C);
```

```

25         save(filename, '-struct', 'Net');
26         fprintf(1, 'Simulation results saved in %s\n', filename);
27     end
28     end
29     end
30
31     end
32     end
33 end
34 fprintf(1, 'Batch Simulator End !!!!!\n');
35 end

```

Listing A.1: BatchSimulation.m source code

```

1 function BP = beacon(Node, n, C)
2
3 C = 600;
4 BP = C / (Node(n).HR*1000);
5
6 end

```

Listing A.2: beacon.m source code

```

1 function [grad, im]=colorGradient(c1, c2, depth)
2 % COLORGRADIENT allows you to generate a gradient between 2 given
3 % colors,
4 % that can be used as colormap in your figures.
5 % USAGE:
6 %
7 % [grad, im]=getGradient(c1, c2, depth)
8 %
9 % INPUT:
10 % - c1: color vector given as Intensity or RGB color. Initial value
11 %
12 % - c2: same as c1. This is the final value of the gradient.
13 %
14 % - depth: number of colors or elements of the gradient.
15 %
16 % OUTPUT:
17 % - grad: a matrix of depth*3 elements containing colormap (or
18 % gradient).
19 % - im: a depth*20*3 RGB image that can be used to display the
20 % result.
21 %
22 % EXAMPLES:
23 % grad=colorGradient([1 0 0],[0.5 0.8 1],128);
24 % surf(peaks)
25 % colormap(grad);
26 %
27 % _____
28 % [grad, im]=colorGradient([1 0 0],[0.5 0.8 1],128);
29 % image(im); %display an image with the color gradient.

```



```

27 | % Copyright 2011. Jose Maria Garcia-Valdecasas Bernal
28 | % v:1.0 22 May 2011. Initial release.
29 |
30 | %Check input arguments.
31 | %input arguments must be 2 or 3.
32 | error(nargchk(2, 3, nargin));
33 |
34 | %If c1 or c2 is not a valid RGB vector return an error.
35 | if numel(c1)~=3
36 |     error('color c1 is not a valid RGB vector');
37 | end
38 | if numel(c2)~=3
39 |     error('color c2 is not a valid RGB vector');
40 | end
41 |
42 | if max(c1)>1&&max(c1)<=255
43 |     %warn if RGB values are given instead of Intensity values.
44 |     %Convert and
45 |     %keep procesing.
46 |     warning('color c1 is not given as intensity values. Trying to
47 |     convert ');
48 |     c1=c1./255;
49 | elseif max(c1) >255||min(c1)<0
50 |     error('C1 RGB values are not valid.')
51 | end
52 | if max(c2)>1&&max(c2)<=255
53 |     %warn if RGB values are given instead of Intensity values.
54 |     %Convert and
55 |     %keep procesing.
56 |     warning('color c2 is not given as intensity values. Trying to
57 |     convert ');
58 |     c2=c2./255;
59 | elseif max(c2) >255||min(c2)<0
60 |     error('C2 RGB values are not valid.')
61 | end
62 | %default depth is 64 colors. Just in case we did not define that
63 | %argument.
64 | if nargin < 3
65 |     depth=64;
66 | end
67 |
68 | %determine increment step for each color channel.
69 | dr=(c2(1)-c1(1))/(depth-1);
70 | dg=(c2(2)-c1(2))/(depth-1);
71 | db=(c2(3)-c1(3))/(depth-1);
72 |
73 | %initialize gradient matrix.
74 | grad=zeros(depth,3);
75 | %initialize matrix for each color. Needed for the image. Size 20*
76 | %depth.
77 | r=zeros(20,depth);
78 | g=zeros(20,depth);
79 | b=zeros(20,depth);
80 | %for each color step, increase/reduce the value of Intensity data.

```

```

76 for j=1:depth
77     grad(j,1)=c1(1)+dr*(j-1);
78     grad(j,2)=c1(2)+dg*(j-1);
79     grad(j,3)=c1(3)+db*(j-1);
80     r(:,j)=grad(j,1);
81     g(:,j)=grad(j,2);
82     b(:,j)=grad(j,3);
83 end
84
85 %merge R G B matrix and obtain our image.
86 im=cat(3,r,g,b);

```

Listing A.3: colorGradient.m source code

```

1  function cost = cost(Net,i)
2
3  cap = 1; % Battery Capacity
4  mu = 3; % Constant
5  epsilon = 2; % Constant
6
7  switch Net.Properties.RoutingAlgorithm
8      case 'E-WME'
9          cost = (cap / ((Net.Node(Net.Link(i)).TxNode).HR + epsilon) *
10                 log(mu)) * (mu .^ ((cap - Net.Node(Net.Link(i)).TxNode)
11                 .AE) / cap - 1) * (Net.Node(Net.Link(i)).RxNode).PE);
12      case 'R-MPRT-org'
13          cost = (Net.Node(Net.Link(i)).RxNode).PE / (Net.Node(Net.
14                 Link(i)).TxNode).HR);
15      case 'R-MPRT-mod'
16          cost = (Net.Node(Net.Link(i)).RxNode).PE / (Net.Node(Net.
17                 Link(i)).TxNode).AE);
18  end
19
20 end

```

Listing A.4: cost.m source code

```

1  function Net = energy(Net,i,type,value)
2
3      BatteryCapacity = 1.476; % Joules (0.1mAh*3600s*4.1V)
4      B = 8*8; % lenght of beacon Bits
5      L = 16*8; % lenght of data Bits
6      R = 500*1024; %Kbps
7      Ptx = 0.0538; % Watt
8      Prx = 0.0425; % Watt
9
10
11
12  switch type
13      case 'harvest'
14          HarvestEnergy = (Net.Node(i).HR/1000) * value;
15          Net.Node(i).AE = min(1, Net.Node(i).AE + HarvestEnergy/
16                 BatteryCapacity);
17
18  end

```

```

17     case 'beacon'
18         BeaconCost = (B/R)*Ptx + (L/R)*Prx;%Energy consume to
           transmit beacon and wait for data
19         Net.Node(i).AE = Net.Node(i).AE - BeaconCost/
           BatteryCapacity;
20     case 'data'
21         IdleListeningCost = value*Prx;% Energy consumption
           waiting for beacon
22         TransmissionCost = (L/R)*Ptx;% Energy consumption in
           transmitting data
23         Net.Node(i).AE = Net.Node(i).AE - (IdleListeningCost+
           TransmissionCost)/BatteryCapacity;
24     end
25
26
27     if Net.Node(i).AE < 0
28         Net.DeadNode = i;
29     end
30
31 end

```

Listing A.5: energy.m source code

```

1  function Net = net(num, a, range, nzone, seed)
2
3  if seed ~=0
4      s = RandStream('mcg16807', 'Seed', seed);
5      RandStream.setGlobalStream(s);
6  end
7
8  Node = struct;
9  Link = struct;
10 Zone = struct;
11
12 k = 0;
13
14
15 Node(1).X = 0;
16 Node(1).Y = 0;
17 Node(1).outLinks = [];
18 Node(1).inLinks = [];
19
20 Node(1).PE=0;
21 Node(1).AE=inf;
22 Node(1).HR=inf;
23 Node(1).PathCost = inf;
24
25 Node(1).BeaconPeriod = 0.100;
26 Node(1).SensePeriod = inf;
27 Node(1).Generate = -1;
28
29 Link = [];
30
31
32

```

```

33
34 %Generates random zones seeds
35 for b=1:nzone
36     Zone(b).X = randi([0,a],1);
37     Zone(b).Y = randi([0,a],1);
38     Zone(b).HR = rand;% random HR for each zone
39 end
40
41 % Generate random number for each node
42 for i = 2:num
43     Node(i).X= randi([0 a],1);
44     Node(i).Y= randi([0 a],1);
45
46 end
47
48
49 % Zone bounds
50 zbx = [ -a -a 2*a 2*a ];
51 zby = [ -a 2*a -a 2*a ];
52 for b=1:4
53     Zone(nzone+b).X = zbx(b);
54     Zone(nzone+b).Y = zby(b);
55     Zone(nzone+b).HR = 0;
56 end
57
58 %Partition the nodes in zones
59 X=vertcat(Zone.X);
60 Y=vertcat(Zone.Y);
61 dt = DelaunayTri(X(:),Y(:));
62
63 for i = 2:num
64     Node(i).Zone= nearestNeighbor(dt, [Node(i).X,Node(i).Y]);
65     v=Node(i).Zone;
66     Node(i).HR = Zone(v).HR;
67     Node(i).AE = 1;
68
69 end
70
71 for i = 2:num
72     Node(i).BeaconPeriod = beacon(Node,i,C);
73     Node(i).SensePeriod = sense(Node,i,A,B);
74     Node(i).PE = packetenergy(Node,i);
75     Node(i).PathCost = inf;
76     Node(i).Generate = -1;
77 end
78
79 % For each node count the distance to other node if it is less than
80 % range make link between that two nodes
81 for i = 1:num
82     for j = (i+1):num
83         d = sqrt(power(Node(i).X - Node(j).X,2)+power(Node(i).Y - Node(
84             j).Y,2));
85         if le(d,range)
86             Link = [Link, struct('TxNode', i, 'RxNode', j) struct('

```

```

86         TxNode', j, 'RxNode', i)];
87         Node(i).outLinks = [Node(i).outLinks, k+1];
88         Node(j).inLinks = [Node(j).inLinks, k+1];
89         Node(j).outLinks = [Node(j).outLinks, k+2];
90         Node(i).inLinks = [Node(i).inLinks, k+2];
91         k = k+2;
92     end
93 end
94
95 Net.Node = Node;
96 Net.Link = Link;
97 Net.Zone = Zone;
98
99 Net.Properties.dt = dt;
100 Net.Properties.NodeNum = num;
101 Net.Properties.Area = a;
102 Net.Properties.Range = range;
103 Net.Properties.ZoneNum = nzone;
104 Net.Properties.Seed = seed;
105
106 end

```

Listing A.6: net.m source code

```

1  function PE = packetenergy(Node,n)
2  % NOTE: This is the energy required to SEND data to node n
3
4      L = 16*8; % lenght of data Bits
5      R = 500*1024; %Kbps
6      Ptx = 0.0538; % Watt
7      Prx = 0.0425; % Watt
8
9      IdleListeningCost = (Node(n).BeaconPeriod / 2)*Prx;
10     TransmissionCost = (L/R)*Ptx;
11
12
13
14     PE = TransmissionCost + IdleListeningCost;
15
16 end

```

Listing A.7: packetenergy.m source code

```

1  function [cost, nexthop] = pathcost(Net,n)
2
3  if n == 1
4      cost = 0;
5      nexthop = -1;
6  else
7      minN = -1;
8      minC = inf;
9      for l = 1: size(Net.Node(n).outLinks,2)
10         tc = Net.Node(Net.Link(Net.Node(n).outLinks(l)).RxNode).
             PathCost + Net.Link(Net.Node(n).outLinks(l)).Cost;

```

```

11         if tc < minC
12             minC = tc;
13             minN = Net.Link(Net.Node(n).outLinks(1)).RxNode;
14         end
15     end
16     if minC < Net.Node(n).PathCost
17         cost = minC;
18         nexthop = minN;
19     else
20         cost = Net.Node(n).PathCost;
21         nexthop = Net.Node(n).NextHop;
22     end
23 end
24
25 end

```

Listing A.8: pathcost.m source code

```

1  function plotnet(Net)
2
3  hold off
4
5  % Plot Voronoi Diagram
6  X=vertcat(Net.Zone.X);
7  Y=vertcat(Net.Zone.Y);
8  voronoi(X(:),Y(:));
9
10 % Fill regions with color
11 hold on
12 grad=colorGradient([0.4 0.6 0.7],[0.8 0.1 0.1],101);
13 [V, R] = Net.Properties.dt.voronoiDiagram();
14 for i = 1:size(R,1)
15     if R{i}(1) ~= 1
16         XX = [];
17         YY = [];
18         for j = 1:size(R{i},2)
19             XX = [XX V(R{i})(j),1];
20             YY = [YY V(R{i})(j),2];
21         end
22         fill(XX,YY, grad(round((Net.Zone(i).HR*1000+10)/10),:));
23     end
24 end
25
26 % Plot Links
27 for i = 1:size(Net.Node,2)
28     if Net.Node(i).NextHop > 0
29         hold on
30         plot([Net.Node(i).X Net.Node(Net.Node(i).NextHop).X],[Net.
31             Node(i).Y Net.Node(Net.Node(i).NextHop).Y], '-.g');
32     end
33 end
34
35 % Plot Region id
36 hold on
37 nump = size(X,1);

```

```

37 plabels = arrayfun(@(n) {sprintf('R%d', n)}, (1:nump)');
38 Hpl = text(X(:), Y(:), plabels, 'color', 'k', 'FontWeight', 'bold',
    'HorizontalAlignment', 'center', 'BackgroundColor', 'none');
39
40
41 % Plot Nodes
42 hold on
43 X=vertcat (Net.Node.X);
44 Y=vertcat (Net.Node.Y);
45 Z=vertcat (Net.Node.HR);
46 scatter(X,Y, '^k');
47
48 % Plot Harvesting Rate
49 hold on
50 nump = size(X,1);
51 plabels = arrayfun(@(n) {sprintf('%d (%d)', n, round(Net.Node(n).HR
    *1000))}, (1:nump)');
52 Hpl = text(X(:), Y(:)+(Net.Properties.Area/50), plabels, 'color', '
    b', 'FontWeight', 'bold', 'HorizontalAlignment', 'center', '
    BackgroundColor', 'none');
53
54 % Colorbar
55 colormap(grad);
56 h = colorbar;
57 set(get(h, 'title'), 'string', 'HR (\mW)');
58 caxis([0,1000]);
59 %lcolorbar('test')
60
61 % Axis Range
62 axis([0 Net.Properties.Area 0 Net.Properties.Area]);
63
64 drawnow;
65
66 end

```

Listing A.9: plotnet.m source code

```

1 function Net = recursivepathcost (Net, n)
2
3 pc = Net.Node(n).PathCost;
4 [Net.Node(n).PathCost, Net.Node(n).NextHop] = pathcost (Net, n);
5 if Net.Node(n).PathCost < pc
6     for i = 1:size(Net.Node(n).outLinks, 2)
7         ID = Net.Link(Net.Node(n).outLinks(i)).RxNode;
8         Net = recursivepathcost (Net, ID);
9     end
10 end
11
12 end

```

Listing A.10: recursivepathcost.m source code

```

1 function Net = routing (seed)
2

```

```

3 N = 50; % Number of Nodes
4 a = 1000; % Field Area
5 Range = 400; % Node Transmission Range
6 Z = 20; % Number of Zones
7
8
9 % Create a random topology
10 Net = net(N,a,Range,Z,seed);
11
12 %Select Routing Algorithm
13 Net.Properties.RoutingAlgorithm = 'R-MPRT-mod';
14
15
16 % Assign Cost to each link
17 for i=1:size(Net.Link,2)
18     Net.Link(i).Cost = cost(Net,i);
19 end
20
21 % Find minimum path cost
22 Net = recursivepathcost(Net,1);
23
24 % Plot the Network
25 plotnet(Net);
26
27
28 hold off
29 end

```

Listing A.11: routing.m source code

```

1 function SP = sense(Node,n,A,B)
2 %Sense period
3 A = 0;
4 B = 20;
5 SP = A + (B / (Node(n).HR));
6
7 end

```

Listing A.12: sense.m source code

```

1 function Net = simulator(seed)
2
3
4     Net = routing(seed);
5
6     Net.Event = struct;
7     Net.Event = [];
8
9     Net.Simulation.CutTime = 5000;
10    Net.Simulation.UpdateThr = 0.5;
11    Net.Simulation.UpdateTopology = true;
12    Net.Simulation.EndFlag = false;
13    Net.DeadNode = 0;
14

```



```

15 Net.Statistics.PacketsReceived = 0;
16 Net.Statistics.EndTime = Net.Simulation.CutTime;
17 for i = 1:Net.Properties.NodeNum
18     Net.Node(i).HarvestTimeStamp = 0;
19     Net.Statistics.PacketsForwarded(i) = 0;
20     Net.Statistics.Beacons(i) = 0;
21     Net.Statistics.Packets(i) = 0;
22     Net.Statistics.Collisions(i) = 0;
23     Net = Init(Net,i);
24 end
25
26 fprintf(1,'Simulation begun.. (stops when a node dies or at
    time: %.2f)\n',Net.Simulation.CutTime);
27 fprintf(1,'Simulation running.. (time: 0)\n');
28 tic
29 while true
30     if Net.DeadNode > 0 || Net.Simulation.EndFlag == true
31         if Net.DeadNode > 0
32             Net.Statistics.EndTime = round(E.Time);
33         end
34         break;
35     end
36     [Net.Event,E] = dequeue(Net.Event);
37     if ~isempty(E)
38         Net = Handle(Net,E);
39     end
40     i=i+1;
41     if i == 10000
42         fprintf(1,'Simulation running.. (time: %.2f)\n',E.Time)
43         ;
44         if Net.Simulation.UpdateTopology == true
45             plotnet(Net);
46         end
47         i = 0;
48     end
49 end
50 toc
51 for i = 1:Net.Properties.NodeNum
52
53     Net.Statistics.CollisionRate(i) = Net.Statistics.Collisions
        (i)/(Net.Statistics.Collisions(i)+Net.Statistics.
        PacketsForwarded(i));
54
55 end
56 Net.Statistics.CollisionRateTot = sum(Net.Statistics.Collisions
    )/(sum(Net.Statistics.Collisions)+Net.Statistics.
    PacketsReceived);
57 Net.Statistics.PacketRate=Net.Statistics.PacketsReceived/Net.
    Statistics.EndTime;
58 Net.Statistics.AvgSensePeriod = 0;
59 for i = 2:nNode
60     Net.Statistics.AvgSensePeriod = Net.Statistics.
        AvgSensePeriod + (Net.Node(i).SensePeriod / nNode);
61 end

```

```

62 end
63
64
65 function Net = Init(Net,i)
66
67     Event = struct ;
68     Event.Time = Net.Node(i).BeaconPeriod*rand;
69     Event.Node = i;
70     Event.Origin = -1;
71     Event.Type = 'Beacon';
72     Net.Event = enqueue(Net.Event,Event);
73     Net.Statistics.Beacons(i) = Net.Statistics.Beacons(i) + 1;
74
75     if i~=1 % Sink doesn't sense
76         Event.Time = Net.Node(i).SensePeriod*rand;
77         Event.Node = i;
78         Event.Origin = i;
79         Event.Type = 'Data';
80         Net.Event = enqueue(Net.Event,Event);
81     end
82
83 end
84
85 function Net = Handle(Net,E)
86
87     Event = struct ;
88
89     switch E.Type
90     case 'Beacon'
91
92         %Compute the time that passed between the last event
93         and
94         %new
95         HarvestTime = E.Time - Net.Node(E.Node).
96         HarvestTimeStamp;
97         %update the time of the last event
98         Net.Node(E.Node).HarvestTimeStamp = E.Time;
99         %compute the accumulated energy in the given Harvest
100         time
101
102         Net = energy(Net,E.Node,'harvest',HarvestTime);
103         %compute the energy need for processing beacon
104         Net = energy(Net,E.Node,'beacon',0);
105
106         Rx = [];
107         %for each beacon's node check all the outLinks
108         for i=1:size(Net.Node(E.Node).outLinks,2)
109             Receiver = Net.Link(Net.Node(E.Node).outLinks(i)).
110             RxNode;
111             %Check for all outLinks which of them use beacon's
112             node as
113             %their next-hop
114             if Net.Node(Receiver).NextHop == E.Node
115                 %if the Receiver has data to send to it's next-
116                 hop
117                 %which is beacpn'node

```

```

111         if Net.Node(Receiver).Generate ~= -1
112
113             %compute the time passed between the last
114             %communication and now
115             HarvestTime = E.Time - Net.Node(Receiver).
                HarvestTimeStamp;
116             %update the time of the last communication
117             Net.Node(Receiver).HarvestTimeStamp = E.
                Time;
118             %compute the energy accumulated in harvesting
119             Net = energy(Net, Receiver, 'harvest',
                HarvestTime);
120             %compute the time lost waiting for the
                beacon
121             IdleListeningTime = E.Time - Net.Node(
                Receiver).Generate;
122             %recompute the energy needed for processing
                data
123             Net = energy(Net, Receiver, 'data',
                IdleListeningTime);
124
125             %add this node to the list of generated
                data.
126             Rx = [Rx Receiver];
127             %this receiver does not need anymore to send
                data
128             Net.Node(Receiver).Generate = -1;
129         end
130     end
131 end
132 %If there are one transmitter
133 if size(Rx,2) == 1
134     % Normal Transmission
135     if E.Node == 1 % if sink
136         Net.Statistics.PacketsReceived = Net.Statistics
                .PacketsReceived + 1;
137         Net.Statistics.PacketsForwarded(E.Node) = Net.
                Statistics.PacketsForwarded(E.Node) + 1;
138         Net.Statistics.Packets(Net.Node(Rx(1)).Origin)
                = Net.Statistics.Packets(Net.Node(Rx(1)).
                Origin) + 1;
139     else
140         Net.Statistics.PacketsForwarded(E.Node) = Net.
                Statistics.PacketsForwarded(E.Node) + 1;
141         Event.Time = E.Time;
142         Event.Node = E.Node;
143         Event.Origin = Net.Node(Rx(1)).Origin;
144         Event.Type = 'Forward-Data';
145         Net.Event = enqueue(Net.Event, Event);
146     end
147     %If two receiver want to send data to beacon's node
148     elseif size(Rx,2) > 1
149         Net.Statistics.Collisions(E.Node) = Net.Statistics.
                Collisions(E.Node) + size(Rx,2);
150 else

```

```

151         % Useless Beacon
152     end
153
154     Event.Time = E.Time + Net.Node(E.Node).BeaconPeriod + (
155         Net.Node(E.Node).BeaconPeriod/10)*randn;
156     Event.Node = E.Node;
157     Event.Origin = -1;
158     Event.Type = 'Beacon';
159     Net.Event = enqueue(Net.Event, Event);
160     Net.Statistics.Beacons(E.Node) = Net.Statistics.Beacons
161         (E.Node) + 1;
162
163     case 'Data'
164
165         %specify when a node produced a data
166         Net.Node(E.Node).Generate = E.Time;
167         %because node can produce or forward data this
168         %statement tells
169         %which is the data event origin.
170         Net.Node(E.Node).Origin = E.Origin;
171
172         %queue a new data event
173         Event.Time = E.Time + Net.Node(E.Node).SensePeriod + (
174             Net.Node(E.Node).SensePeriod/10)*randn;
175         Event.Node = E.Node;
176         Event.Origin = E.Node;
177         Event.Type = 'Data';
178         Net.Event = enqueue(Net.Event, Event);
179
180         %if the event node available energy is lesser than
181         %UpdateThr
182         if Net.Node(E.Node).AE < Net.Simulation.UpdateThr
183             %recompute the routing path
184             for i=1:size(Net.Node,2)
185                 Net.Node(i).PathCost = inf;
186             end
187             for i=1:size(Net.Link,2)
188                 Net.Link(i).Cost = cost(Net,i);
189             end
190             Net = recursivepathcost(Net,1);
191         end
192     case 'Forward-Data'
193
194         Net.Node(E.Node).Generate = E.Time;
195         Net.Node(E.Node).Origin = E.Origin;
196
197     end
198
199     if E.Time > Net.Simulation.CutTime
200         Net.Simulation.EndFlag = true;
201     end
202 end

```

```
201 function Q = enqueue(Q,E)
202     if isempty(Q)
203         Q = [E];
204     else
205         if E.Time > Q(end).Time
206             Q = [ Q E ];
207         elseif E.Time < Q(1).Time
208             Q = [ E Q ];
209         else
210             for i = size(Q,2)-1:-1:1
211                 if E.Time > Q(i).Time
212                     Q = [Q(1:i) E Q(i+1:end)];
213                     break;
214                 end
215             end
216         end
217     end
218 end
219
220 function [Q,E] = dequeue(Q)
221     if size(Q,2) > 1
222         E = Q(1);
223         Q = Q(2:end);
224     elseif size(Q,2) == 1
225         E = Q(1);
226         Q = [];
227     else
228         E = [];
229         Q = [];
230     end
231
232 end
```

Listing A.13: simulator.m source code



# Acronyms

---

<b>OSI</b>	Open System Interconnection
<b>WSN</b>	Wireless Sensor Network
<b>EH-WSN</b>	Energy Harvesting Wireless Sensor Network
<b>MESW</b>	Maximum Energetically Sustainable Workload
<b>TBRPF</b>	Topology Dissemination Based on Reverse-Path Forwarding
<b>TORA</b>	Temporarily Ordered Routing algorithm
<b>LEACH</b>	Low-energy adaptive clustering hierarchy
<b>MCU</b>	Micro controller unit
<b>ADC</b>	Analog to digital converter
<b>RF</b>	Radio-Frequency
<b>MAC</b>	Medium Access Control
<b>LEACH-C</b>	Low-energy adaptive clustering hierarchy centralized
<b>PEGASIS</b>	Power-efficient gathering in sensor information system
<b>DD</b>	Direct Diffusion
<b>SWE</b>	Single winner algorithm
<b>MWE</b>	Multiple winner algorithm

---

<b>SPIN</b>	sensor protocols for information via negotiation
<b>SPIN-BC</b>	SPIN for Broadcast Network
<b>GEAR</b>	Geographic and energy aware routing
<b>IGF</b>	Implicit geographic forwarding
<b>MERR</b>	Minimum energy relay routing
<b>MIP</b>	Multi-agent based Itinerary Planning
<b>VCL</b>	visiting central location
<b>ROAM</b>	Routing on-demand acyclic multipath
<b>LMR</b>	Label-based multipath routing
<b>GRAB</b>	Gradient broadcast
<b>SAR</b>	Sequential assignment routing
<b>R-MF</b>	Randomized Max-Flow
<b>E-WME</b>	Energy-opportunistic Weighted Minimum Energy
<b>R-MPRT</b>	Randomized Minimum Path Recovery Time
<b>R-MPE</b>	Randomized minimum path energy
<b>EHOR</b>	Energy Harvesting Opportunistic Routing Protocol
<b>GR</b>	Geographic Routing
<b>GR-DD</b>	Geographic Routing with Duplicate Detection
<b>GR-DD-RT</b>	Geographic Routing with Duplicate Detection and Retransmission
<b>DEHAR</b>	Distributed Energy Harvesting Aware Routing Algorithm
$T_e$	recovery time
$MESW^{opt}$	Optimum MESW
$pE_e$	Packet energy
$C_e$	Channel capacity



# Bibliography

---

- [1] W.K.G. Seah, Z.A. Eu, and H.P. Tan. Wireless sensor networks powered by ambient energy harvesting (wsn-heap)-survey and challenges. In *Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology, 2009. Wireless VITAE 2009. 1st International Conference on*, pages 1–5. IEEE, 2009.
- [2] Z.A. Eu, H.P. Tan, and W.K.G. Seah. Opportunistic routing in wireless sensor networks powered by ambient energy harvesting. *Computer Networks*, 54(17):2943–2966, 2010.
- [3] N. Pantazis, S. Nikolidakis, and D. Vergados. Energy-efficient routing protocols in wireless sensor networks: A survey.
- [4] E.M. Royer and C.K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *Personal Communications, IEEE*, 6(2):46–55, 1999.
- [5] S.K. Singh, MP Singh, and DK Singh. Routing protocols in wireless sensor networks—a survey. *International Journal of Computer science and engineering Survey (IJCSSES)*, 1(2):63–83, 2010.
- [6] DA Vidhate, AK Patil, and SS Pophale. Performance evaluation of low energy adaptive clustering hierarchy protocol for wireless sensor networks. In *Proceedings of the International Conference and Workshop on Emerging Trends in Technology*, pages 59–63. ACM, 2010.
- [7] Y. Wu, S. Fahmy, and N.B. Shroff. Energy efficient sleep/wake scheduling for multi-hop sensor networks: Non-convexity and approximation algo-

- rithm. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications*. IEEE, pages 1568–1576. IEEE, 2007.
- [8] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 56–67. ACM, 2000.
- [9] K. Sohrabi, J. Gao, V. Ailawadhi, and G.J. Pottie. Protocols for self-organization of a wireless sensor network. *Personal Communications, IEEE*, 7(5):16–27, 2000.
- [10] W.R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 174–185. ACM, 1999.
- [11] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer networks*, 52(12):2292–2330, 2008.
- [12] Xenofon Fafoutis and Nicola Dragoni. Analytical comparison of mac schemes for energy harvesting—wireless sensor networks. In *Networked Sensing Systems (INSS), 2012 Ninth International Conference on*, pages 1–6. IEEE, 2012.
- [13] L. Alazzawi, A. Elkateeb, et al. Performance evaluation of the wsn routing protocols scalability. *Journal of Computer Systems, Networks, and Communications*, 2008, 2009.
- [14] P. Arce, J.C. Guerri, A. Pajares, and O. L aro. Performance evaluation of video streaming over ad hoc networks using flat and hierarchical routing protocols. *Mobile Networks and Applications*, 13(3-4):324–336, 2008.
- [15] J.C. Castillo, T. Olivares, and L. Orozco-Barbosa. Routing protocols for wireless sensor networks-based network. *Albacete Research Institute of Informatics, University of Castilla-La Mancha Spain*, 2007.
- [16] M. Abolhasan, T. Wysocki, and E. Dutkiewicz. A review of routing protocols for mobile ad hoc networks. *Ad hoc networks*, 2(1):1–22, 2004.
- [17] B. Bellur and R.G. Ogier. A reliable, efficient topology broadcast protocol for dynamic networks. In *INFOCOM’99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 178–186. IEEE, 1999.
- [18] Z.J. Haas and M.R. Pearlman. The performance of query control schemes for the zone routing protocol. *IEEE/ACM Transactions on Networking (TON)*, 9(4):427–438, 2001.

- [19] J.N. Al-Karaki and A.E. Kamal. Routing techniques in wireless sensor networks: a survey. *Wireless Communications, IEEE*, 11(6):6–28, 2004.
- [20] N. Sadagopan, B. Krishnamachari, and A. Helmy. Active query forwarding in sensor networks. *Ad Hoc Networks*, 3(1):91–113, 2005.
- [21] V. Jolly and S. Latifi. Comprehensive study of routing management in wireless sensor networks-part-2. In *Proceedings of International Conference on Wireless Networks*, pages 49–62, 2006.
- [22] J. Kulik, W. Heinzelman, and H. Balakrishnan. Negotiation-based protocols for disseminating information in wireless sensor networks. *Wireless Networks*, 8(2):169–185, 2002.
- [23] Y. Yu, R. Govindan, and D. Estrin. Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks. Technical report, Citeseer, 2001.
- [24] Min Chen, Sergio Gonzalez, and Victor CM Leung. Applications and design issues for mobile agents in wireless sensor networks. *Wireless Communications, IEEE*, 14(6):20–26, 2007.
- [25] Min Chen, Sergio Gonzalez, Yan Zhang, and Victor CM Leung. Multi-agent itinerary planning for wireless sensor networks. *Quality of Service in Heterogeneous Networks*, pages 584–597, 2009.
- [26] M. Tarique, K.E. Tepe, S. Adibi, and S. Erfani. Survey of multipath routing protocols for mobile ad hoc networks. *Journal of Network and Computer Applications*, 32(6):1125–1143, 2009.
- [27] J. Raju and J.J. Garcia-Luna-Aceves. A new approach to on-demand loop-free multipath routing. In *Computer Communications and Networks, 1999. Proceedings. Eight International Conference on*, pages 522–527. IEEE, 1999.
- [28] X. Hou, D. Tipper, and J. Kabara. Label-based multipath routing (lmr) in wireless sensor networks. In *Proceedings, The International Symposium on Advanced Radio Technologies (ISART)*, 2004.
- [29] F. Ye, G. Zhong, S. Lu, and L. Zhang. Gradient broadcast: A robust data delivery protocol for large scale sensor networks. *Wireless Networks*, 11(3):285–298, 2005.
- [30] E. Lattanzi, E. Regini, A. Acquaviva, and A. Bogliolo. Energetic sustainability of routing algorithms for energy-harvesting wireless sensor networks. *Computer Communications*, 30(14):2976–2986, 2007.

- [31] J.H. Chang and L. Tassiulas. Routing for maximum system lifetime in wireless ad-hoc networks. In *PROCEEDINGS OF THE ANNUAL ALLERTON CONFERENCE ON COMMUNICATION CONTROL AND COMPUTING*, volume 37, pages 1191–1200. The University; 1998, 1999.
- [32] M. Bhardwaj and A.P. Chandrakasan. Bounding the lifetime of sensor networks via optimal role assignments. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1587–1596. IEEE, 2002.
- [33] A. Bogliolo, E. Lattanzi, and A. Acquaviva. Energetic sustainability of environmentally powered wireless sensor networks. In *Proceedings of the 3rd ACM international workshop on Performance evaluation of wireless ad hoc, sensor and ubiquitous networks*, pages 149–152. ACM, 2006.
- [34] K. Kar, Murali Kodialam, T.V. Lakshman, and L. Tassiulas. Routing for network capacity maximization in energy-constrained ad-hoc networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 1, pages 673 – 681 vol.1, march-3 april 2003.
- [35] Longbi Lin, Ness B Shroff, and R Srikant. Asymptotically optimal energy-aware routing for multihop wireless networks with renewable energy sources. *Networking, IEEE/ACM Transactions on*, 15(5):1021–1034, 2007.
- [36] David Hasenfratz, Andreas Meier, Clemens Moser, Jian-Jia Chen, and Lothar Thiele. Analysis, comparison, and optimization of routing protocols for energy harvesting wireless sensor networks. In *Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), 2010 IEEE International Conference on*, pages 19–26. IEEE, 2010.
- [37] Zhi Ang Eu, H. Tan, and W.K.-G. Seah. Routing and relay node placement in wireless sensor networks powered by ambient energy harvesting. In *Wireless Communications and Networking Conference, 2009. WCNC 2009. IEEE*, pages 1–6, April.
- [38] F. Kuhn, R. Wattenhofer, and A. Zollinger. An algorithmic approach to geographic routing in ad hoc and sensor networks. *IEEE/ACM Transactions on Networking (TON)*, 16(1):51–62, 2008.
- [39] M. Yoshida, T. Kitani, M. Bandai, T. Watanabe, P. Chou, and W.K.G. Seah. Probabilistic data collection protocols for energy harvesting sensor networks. In *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*, pages 366–373. IEEE, 2011.

- 
- [40] Mikkel Koefoed Jakobsen, Jan Madsen, and Michael R Hansen. Dehar: A distributed energy harvesting aware routing algorithm for ad-hoc multi-hop wireless sensor networks. In *World of Wireless Mobile and Multimedia Networks (WoWMoM), 2010 IEEE International Symposium on a*, pages 1–9. IEEE, 2010.
- [41] Xenofon Fafoutis and Nicola Dragoni. Adaptive media access control for energy harvesting—wireless sensor networks. In *Networked Sensing Systems (INSS), 2012 Ninth International Conference on*, pages 1–4. IEEE, 2012.