

A STUDY ON CROSS-LAYER OPTIMIZATION  
FOR APPLICATION SPECIFIC WIRELESS  
SENSOR NETWORKS

BY

TAYSEER AHMED YOUSEF AL-KHDOUR

A Dissertation Presented to the  
DEANSHIP OF GRADUATE STUDIES

**KING FAHD UNIVERSITY OF PETROLEUM & MINERALS**

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the  
Requirements for the Degree of

**DOCTOR OF PHILOSOPHY**

In

COMPUTER SCIENCE AND ENGINEERING

June 2009

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS  
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This dissertation, written by TAYSEER AHMED YOUSEF AL-KHDOUR under the direction of his dissertation advisor and approved by his dissertation committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of DOCTOR OF PHILOSOPHY IN Computer Science and Engineering

Dissertation Committee



Dr. Uthman Baroudi  
Dissertation Advisor



Prof. Shokri Selim  
Dissertation Co-Advisor



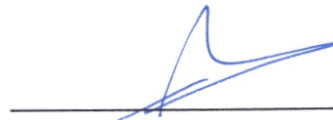
Prof. Mayez Al-Mouhamed  
Member



Dr. Mohammad Al-Suwaiyel  
Member



Department Chairman



Dean of Graduate Studies

14/3/09  
Date



## **Dedication**

*To my mother*

*To my wife*

*To my kids Laith and Noor*

*To My Brothers and Sister*

## ACKNOWLEDGMENT

All praise be to ALLAH for his limitless help and guidance. Peace and Blessings of ALLAH be upon his messenger Mohammad.

Acknowledgement is due to the King Fahd University of Petroleum and Minerals for supporting this research. Moreover, this research is supported by King Abdulaziz City for Science and Technology (KACST).

I wish to express my sincere gratitude and deepest appreciation to my advisor, Dr. Uthman Baroudi for his substantial inspiration and technical as well as philosophical guidance. Also, I would like to thank Prof. Shokri Selim for his helpful advices and cooperation. I would like to thank Prof. Mayez Almuhamad, Dr. Mohammad Alsuyail, and Dr. Tarek Sheltami for serving on my dissertation committee, and for spending their valuable time reading and correcting this dissertation.

The love and concern of every member of my small and large family will always be remembered. I will never forget the huge support of my wife. My feeling of responsibility toward every member of my family is one of the most important motivations for me in every step of my academic career. Lastly, I wish to thank all my real friends for their sincere concern and support.

# Table of contents

DEDICATION.....	iii
ACKNOWLEDGMENT.....	iv
TABLE OF CONTENTS.....	v
LIST OF TABLES.....	xi
LIST OF FIGURES.....	xiii
ABSTRACT.....	xx
ARABIC ABSTRACT.....	xxi
Chapter One : INTRODUCTION.....	1
1.1    DISTINGUISHED FEATURES OF WSN:.....	3
1.1.1    THE DEPLOYMENT OF SENSOR NODES:.....	4
1.1.2    THE RESOURCES CONSTRAINTS.....	4
1.1.3    THE QUALITY OF SERVICE (QOS) REQUIREMENT.....	5
1.1.4    DATA TRAFFIC MODELS.....	5
1.2    MOTIVATION:.....	6
1.3    PROBLEM STATEMENT.....	7
1.4    DESCRIPTION OF THE WORK.....	9
1.4.1    METHODOLOGY:.....	10
1.4.2    RESEARCH CONTRIBUTIONS:.....	11
1.5    PUBLICATIONS.....	14
1.6    DISSERTATION ORGANIZATION.....	15

Chapter Two : LITERATURE REVIEW .....	17
2.1 MAC PROTOCOLS FOR WSN .....	17
2.1.1 POWER AWARE MULTI-ACCESS PROTOCOL WITH SIGNALING FOR AD HOC NETWORKS (PAMAS) PROTOCOL.....	18
2.1.2 S-MAC PROTOCOL.....	21
2.1.3 A TRAFFIC AWARE, ENERGY EFFICIENT MAC PROTOCOL FOR WIRELESS SENSOR NETWORKS (TEEM).....	26
2.1.4 MEDIUM ACCES CONTROL WITH A DYNAMIC DUTY CYCLE FOR SENSOR NETWORK (DSMAC) .....	27
2.1.5 TIMEOUT-MAC (T-MAC).....	29
2.1.6 GANGS PROTOCOL.....	33
2.2 ROUTING PROTOCOLS FOR WSN.....	35
2.2.1 DATA-CENTRIC PROTOCOLS.....	36
2.2.2 HIERARCHICAL PROTOCOLS .....	42
2.2.3 LOCATION-BASED PROTOCOLS: .....	59
2.2.4 QOS-AWARE PROTOCOLS .....	66
2.3 CROSS LAYER DESIGN IN WSN.....	73
2.4 INTEGER LINEAR PROGRAMMIN IN ILP .....	82
Chapter Three THE GENERALIZED ENERGY-EFFICIENT TIME-BASED COMMUNICATION PROTOCOL FOR WIRELESS SENSOR NETWORKS.....	84
3.1 SHORTCOMINGS OF EAD AND LEACH.....	84
3.2 GET DESCRIPTION.....	86
3.2.1 SELECTING THE GATEWAYS.....	88

3.2.2	BUILDING THE TREE: .....	92
3.2.3	BUILDING THE SCHEDULE: .....	99
3.2.4	DATA TRANSMISSION PHASE: .....	103
3.3	ENERGY-EFFICIENT DISTRIBUTED SCHEDULE-BASED COMMUNICATION PROTOCOL FOR WSN (EEDS) .....	104
3.4	A GENERALIZED ENERGY-AWARE DATA CENTRIC ROUTING FOR WIRELESS SENSOR NETWORK (EAD <sub>GENERAL</sub> ) .....	105
3.5	CONCLUSION .....	106
Chapter Four : SIMULATION SETUP .....		108
4.1	INTRODUCTION .....	108
4.2	DESCRIPTION OF SIMULATOR .....	109
4.3	NETWORK TOPOLOGIES .....	111
4.4	ENERGY MODEL .....	112
4.5	SIMULATION ASSUMPTIONS .....	114
4.6	COLLECTING SIMULATION RESULTS .....	115
Chapter Five PERFORMANCE EVALUATION .....		120
5.1	INTRODUCTION .....	120
5.2	PERFORMANCE EVALUATION OF GET .....	121
5.2.1	A COMPARISON BETWEEN GET AND EAD:- .....	121
5.2.2	A COMPARISON BETWEEN GET AND LEACH.....	129
5.3	PERFORMANCE EVALUATION OF EEDS .....	135
5.3.1	A COMPARISON BETWEEN EEDS AND EAD .....	135
5.3.2	A COMPARISON BETWEEN EEDS AND LEACH .....	140

5.4	PERFORMANCE EVALUATION OF EAD <sub>General</sub> .....	145
5.5	CONCLUSION .....	148
Chapter Six : PERFORMANCE EVALUATION OF GET AND EEDS PROTOCOLS FOR DIFFERENT NETWORK CONFIGURATIONS AND APPLICATIONS .....		
		150
6.1	AN ENTROPY-BASED THROUGHPUT METRIC FOR WSN ROUTING PROTOCOLS .....	151
6.2	PERFORMANCE EVALUATION OF EAD, AND LEACH ROUTING PROTOCOLS USING AN ENTROPY-BASED THROUGHPUT METRIC .....	153
6.3	THE EFFECT OF NETWORK TOPOLOGY ON PERFORMANCE OF ROUTING PROTOCOLS FOR WIRELESS SENSOR NETWORKS .....	155
6.3.1	THE EFFECT OF NETWORK TOPOLOGY ON PERFORMANCE OF EAD .....	157
6.3.2	THE EFFECT OF NETWORK TOPOLOGY ON PERFORMANCE OF EEDS :- .....	163
6.3.3	THE EFFECT OF NETWORK TOPOLOGY ON PERFORMANCE OF GET:- .....	167
6.3.4	THE EFFECT OF NETWORK TOPOLOGY ON PERFORMANCE OF LEACH:- .....	172
6.4	THE EFFECT OF OVERHEAD PHASES IN THE PERFORMANCE OF EEDS AND GET .....	176
6.5	PERFORMANCE EVALUATION OF THE EAD, EEDS AND GET UNDER DIFFERENT NETWORK SIZE .....	178



6.6	PERFORMANCE EVALUATION OF EAD, EEDS, AND GET WITH DIFFERENT APPLICATIONS .....	182
6.6.1	PERFORMANCE EVALUATION OF THE EAD, EEDS AND GET ASSUMING DETERMINISTIC INTER-ARRIVAL TIME BETWEEN EVENTS.	183
6.6.2	PERFORMANCE EVALUATION OF THE EAD, EEDS AND GET ASSUMING RANDOM INTER-ARRIVAL TIME BETWEEN EVENTS.....	188
6.7	PERFORMANCE EVALUATION OF THE EEDS AND GET WITH MORE OPTIMAL SCHEDULE .....	191
6.8	CONCLUSION.....	195
Chapter Seven INTEGER LINEAR PROGRAMIN FORMULATION .....		198
7.1	INTRODUCTION .....	198
7.2	ILP MODEL FORMULATION .....	199
7.2.1	ILP CONSTRAINTS.....	200
7.2.2	ILP COST FUNCTIONS.....	208
7.3	SOLVING ILP MODEL.....	211
7.3.1	DELAY: LOWER AND UPPER BOUNDS FOR DELAY .....	212
7.3.2	UPPER BOUND OF NUMBER OF LINKS IN SUB-NETWORK WITH THREE NODES .....	215
7.3.3	NO LINK BETWEEN TWO NODES WHICH DISTANCE BETWEEN THEM GREATER THAN TRANSMISSION RANGE. ....	215
7.4	PERFORMANCE EVALUATION USING ILP.....	216
7.4.1	A COMPARISON BETWEEN ILP MODEL SOLUTION RESULTS AND SIMULATION RESULTS .....	218

7.4.2	COMPARING THE RESULTS OF DIFFERENT COST FUNCTIONS .	224
7.4.3	COMPARING THE RESULTS OF SOLVING ILP MODEL USING THE FIRST COST FUNCTION WITH DIFFERENT NETWORK SIZES: .....	231
Chapter Eight :	CONCLUSION .....	235
Chapter Nine :	FUTURE WORK .....	239
REFERENCES.....		240
Vitae .....		250

# List of Table

Table 2-1 : Classification of Routing Protocols based on the Applications .....	72
Table 2-2 : Summary of 81Cross layer Protocols for WSN .....	81
Table 3-1: Regions of the Network.....	95
Table 4-1: Simulation Parameters.....	114
Table 5-1: A comparison between GET and EAD in terms of throughput .....	127
Table 5-2: Statistics for Tree Hight .....	128
Table 5-3: A comparison between GET and LEACH in terms of throughput .....	135
Table 5-4: The Aggregated Throughput in EEDS compared with EAD protocol.....	137
Table 5-5: The Aggregated Throughput in EEDS compared with LEACH.....	143
Table 5-6: A summary of the improvement in EEDS compared with EAD and LEACH .....	144
Table 6-1 : Statistics of overhead time in EEDS .....	177
Table 6-2: Statistics of overhead time in EEDS .....	178
Table 6-3 : A summary of improvement in EEDS compared with EAD, GET (1 Data Transmission period) .....	181
Table 6-4: A summary of improvement in EEDS compared with EAD, GET (5 Data Transmission periods).....	181
Table 6-5: A summary of improvement in EEDS compared with EAD, GET (5 Data Transmission periods).....	181

Table 7-1: Lower bounds on Delay and $(t_1)$ .....	214
Table 7-2: Experiments parameters .....	217
Table 7-3: Delay for Different cost Functions .....	231

# List of Figures

Figure 1-1 Components of Sensor Nodes .....	1
Figure 1-2: A wireless Sensor Network.....	2
Figure 2-1: S-MAC: Neighboring nodes A and B have different schedules. They synchronize with nodes C and D respectively .....	23
Figure 2-2: Timing relationship between a receiver and different senders, CS stands for carrier sense .....	24
Figure 2-3: Neighboring nodes which adopt different duty cycles can still communicate with old schedule .....	28
Figure 2-4: listening sleeping periods in both S-MAC and T-MAC .....	29
Figure 2-5: The early sleeping problem. Node D goes to sleep before C can send RTS to it. ....	31
Figure 2-6: The future-request-to-send packet exchange keeps Node D awake. ....	31
Figure 2-7: Taking priority upon receiving RTS .....	32
Figure 2-8: Time frame for cluster head/Node .....	34
Figure 2-9: Time line showing LEACH operation .....	43
Figure 2-10: Superframe Structure in ALPR .....	48
Figure 2-11: A single round of the BMA protocol .....	50
Figure 2-12 : A Proposed cross-layer optimization framework.....	79

Figure 3-1: Time frame for GET protocol .....	87
Figure 3-2: A Sample Network with its tiers .....	89
Figure 3-3: The state machine for the node in selecting gateway phase .....	91
Figure 3-4: the state machine of the sink in the selecting gateway phase .....	92
Figure 3-5: The state machine of a node in building tree phase .....	94
Figure 3-6: A sample Network .....	94
Figure 3-7: A Partial Tree-1 .....	96
Figure 3-8: A Partial Tree-2 .....	97
Figure 3-9: A complete tree-1 .....	98
Figure 3-10: A complete tree-II .....	99
Figure 3-11 : Pseudo code for building schedule.....	101
Figure 3-12: The nodes of the network with their transmission time TRR, TRT.....	102
Figure 3-13: A schedule of the example Network .....	103
Figure 3-14: Time Frame for the EED.....	104
Figure 3-15: Time frame for the EAD <sub>General</sub> Protocol. ....	106
Figure 4-1: Block diagram of simulator.....	109
Figure 4-2: The Grid topology .....	111
Figure 4-3: A snap shot of an example of output results file.....	117
Figure 4-4: Example of Intervals used in average calculation.....	118
Figure 4-5: A snap shot of an example of an average output file .....	119
Figure 5-1: Number of Live Nodes vs. Time, (GET, EAD).....	122
Figure 5-2: Total Consumed Energy vs. Time, (GET, EAD).....	124
Figure 5-3: Average Consumed Energy per a single packet, (GET, EAD).....	126

Figure 5-4: The Cumulative Distribution Function of the residual energy in the nodes at the end of network lifetime (10 Data Transmission Period).....	127
Figure 5-5: Number of Live Nodes vs. Time, (LEACH, and GET) .....	130
Figure 5-6: Total Consumed energy vs. Time, (LEACH, GET) .....	132
Figure 5-7: Average Consumed Energy per a single packet , (GET,LEACH).....	134
Figure 5-8: Number of live Nodes vs. Time, (EEDS, EAD) .....	136
Figure 5-9: Total consumed Energy vs. Time , (EEDS, EAD).....	138
Figure 5-10: Average Consumed Energy per a single packet, (EEDS, EAD).....	139
Figure 5-11: Number of Live Nodes vs. Time, (EEDS, LEACH).....	141
Figure 5-12: Total Consumed Energy vs. time, (EEDS, LEACH).....	143
Figure 5-13: Average Consumed Energy per a single packet, (EEDS, LEACH).....	144
Figure 5-14: Number of Live Nodes vs. Time, (EAD, EAD <sub>General</sub> ) .....	145
Figure 5-15 : Total Consumed Energy vs. Time (EAD, EAD <sub>General</sub> ) .....	146
Figure 5-16: Throughput vs. Time, (EAD, EAD <sub>General</sub> ) .....	148
Figure 6-1: A network with different Clusters.....	152
Figure 6-2: A Comparison between EAD and LEACH using absolute throughput .....	154
Figure 6-3: A Comparison between EAD and LEACH using Entropy .....	154
Figure 6-4 : Grid topology .....	156
Figure 6-5: Random Topology.....	157
Figure 6-6: Number of Live Nodes vs. time for EAD, Grid (Random Topologies.....	158
Figure 6-7: Number of Gateway Nodes vs. time for EAD, Grid and Random Topologies .....	159

Figure 6-8: Percentage of covered area vs. time for EAD, Grid and Random Topologies .....	160
Figure 6-9: Throughput vs. time for EAD, Grid and Random Topologies.....	161
Figure 6-10: Information Throughput vs. time for EAD, Grid and Random Topologies .....	162
Figure 6-11: Number of Live Nodes vs. time for EEDS, Grid and Random Topologies	164
Figure 6-12: Number of Gateways vs. time for EEDS, Grid and Random Topologies .	165
Figure 6-13: Percentage of Covered area vs. time for EEDS, Grid and Random Topologies.....	165
Figure 6-14: Throughput vs. time for EEDS, Grid and Random Topologies.....	166
Figure 6-15: Information Throughput vs. time for EEDS, Grid and Random Topologies .....	167
Figure 6-16: Number of Live Nodes vs. Time for GET, random and Grid Topologies.	168
Figure 6-17: Number of Gateways vs. Time for GET, random and Grid Topologies....	169
Figure 6-18: Throughput vs. Time for GET, random and Grid Topologies.....	170
Figure 6-19: Information Throughput vs. Time for GET, random and Grid Topologies	170
Figure 6-20: Total Consumed Energy vs. Time for GET, random and Grid Topologies	171
Figure 6-21: Percentage of covered area vs. Time for GET, random and Grid Topologies .....	172
Figure 6-22: Number of Live Nodes vs. Time for LEACH, random and Grid Topologies .....	173
Figure 6-23: Throughput vs. Time for LEACH, random and Grid Topologies .....	174



Figure 6-24: Total Consumed energy vs. Time for LEACH, random and Grid Topologies .....	174
Figure 6-25: Information Throughput vs. Time for LEACH, random and Grid Topologies .....	175
Figure 6-26: Percentage of Covered area vs. Time for LEACH, random and Grid Topologies.....	175
Figure 6-27: A comparison between no adaptive and different number of periods .....	177
Figure 6-28: Network Lifetime vs. Number of Nodes (10 Data transmission period)...	179
Figure 6-29: Throughput vs. Number of Nodes (10 Data transmission period).....	180
Figure 6-30: Covered Area vs. Number of Nodes (10 Data transmission period) .....	180
Figure 6-31: Delay vs. Interval Time between Events, Fixed, (10 Data Transmission Periods) .....	183
Figure 6-32: Delay for processed events case-(1).....	184
Figure 6-33: Delay for processed events case (2).....	184
Figure 6-34: Delay for processed events case (3).....	185
Figure 6-35: Delay for processed events case (4).....	185
Figure 6-36: Network lifetime vs. Inter-Arrival Time (10 Data Transmission Periods) .....	186
Figure 6-37: Throughput (Packets/sec.) vs. Inter-Arrival Time (10 Data Transmission Periods) .....	187
Figure 6-38: Percentage of covered Area vs. Inter-Arrival Time (10 Data Transmission Periods) .....	188

Figure 6-39: Delay vs. Mean of Interval Time between Events, (10 Data Transmission Periods).....	189
Figure 6-40: Network Lifetime vs. Mean of Interval Time between Events, (10 Data Transmission Periods).....	190
Figure 6-41: Throughput (Packets/sec) vs. Mean of Interval Time between Events, (10 Data Transmission Periods).....	190
Figure 6-42: Percentage of Detected Events vs. Mean of Interval Time between Events, (10 Data Transmission Periods).....	191
Figure 6-43: Pseudo Code of the Optimal building Scheduling algorithm .....	192
Figure 6-44: Delay vs. Mean of Inter-arrival Time between Events, (10 Data Transmission Periods) with optimal schedule .....	193
Figure 6-45: Network Lifetime vs. Mean of Inter-arrival Time between Events, (10 Data Transmission Periods) with optimal schedule .....	194
Figure 6-46: Throughput (packets/sec) vs. Mean of Inter-arrival Time between Events, (10 Data Transmission Periods) with optimal schedule .....	194
Figure 6-47: % of Detected events vs. Mean of Inter-arrival Time between Events, (10 Data Transmission Periods) with optimal schedule.....	195
Figure 7-1: degenerate and one Level Trees.....	212
Figure 7-2: Possible trees in a network with Three Nodes.....	213
Figure 7-3: Possible trees in a network with four nodes.....	214
Figure 7-4: Interaction between Visual Basic and LINGO .....	216
Figure 7-5: Number of Live nodes vs. Time, Number of nodes=10, Area=50x50 .....	219
Figure 7-6: Percentage of covered Area vs. time, Number of nodes=10, Area=50x50..	220

Figure 7-7: Network Lifetime vs. Network Density, Number of nodes=10.....	221
Figure 7-8: Throughput vs. Network Density, Number of nodes=10.....	222
Figure 7-9: Delay vs. Network Density, Number of nodes=10.....	223
Figure 7-10: Delay vs. Network Density, Number of Nodes =10; $Max E_c E_i$ , $Min t_1$ .....	224
Figure 7-11: Network lifetime vs. Density, Number of Nodes =10; $Max E_c E_i$ , $Min t_1$ ..	225
Figure 7-12: Throughput vs. Density, Number of Nodes =10; $Max E_c E_i$ , $Min D$ .....	226
Figure 7-13: Network lifetime (sec.) vs. Delay (time slots), Network Density= 0.004 node/m <sup>2</sup> .....	227
Figure 7-14: Network lifetime (sec.) vs. Delay (time slots), Different Network Density .....	228
Figure 7-15: Number of Live Nodes vs. Time, Nodes=10, Area=50x50 .....	229
Figure 7-16: Percentage of covered Area vs. Time, Nodes=10, Area=50x50.....	230
Figure 7-17: Network Lifetime vs. Density, first cost function.....	231
Figure 7-18: Throughput (packets) vs. Density, first cost function.....	233
Figure 7-19: Number of live nodes vs. time, Density=0.004.....	234
Figure 7-20: Percentage of covered area vs. time, Density=0.004 .....	234

## DISSERTATION ABSTRACT

Name: TAYSEER AHMED YOUSEF AL-KHDOUR

Title: A STUDY ON CROSS LAYER OPTIMIZATION FOR APPLICATION SPECIFIC WIRELESS SENSOR NETWORKS

Major: COMPUTER SCIENCE AND ENGINEERING

Date: JUNE 2009

In our research, we study data forwarding in WSN. The main parameters that affect the operation of data forwarding from sensor nodes to sink are identified. Based on our study of the existing protocols, we propose a framework to forward data from sensor nodes to sink. A cross layer design methodology is adopted in designing our framework. Our framework aims at maximizing network lifetime. The proposed framework is called a Generalized Energy-Efficient Time-Based communication protocol (GET). In GET, any node can communicate with the sink directly, and an energy efficient tree is constructed from all nodes toward the sink. Based on this tree, a TDMA schedule is built to forward data from all nodes to the sink. GET is validated using different network configurations and it shows good improvements compared with EAD and LEACH. Moreover, An Energy Efficient Data Communication Protocol (EEDS) which is a special case of GET is proposed. In EEDS, only the nodes that are close to sink can communicate with it directly. In addition, we generalize the Energy-Aware Data Centric Routing (EAD) such that any node can communicate with the sink directly. The new protocol is called  $EAD_{General}$ . Extensive simulation experiments show that  $EAD_{General}$  outperforms EAD. Moreover, we proposed Information-Entropy based metric to measure the throughput in WSN. In the new metric, we defined the throughput as the amount of information delivered to the sink. The proposed metric yields a better understanding of the operation of the WSN application. Finally, to explore the optimal solutions that can be produced assuming global information, we formulate EEDS with an integer linear programming (ILP) model. We proposed four cost functions for the ILP model. We used LINGO solver to solve our model. The results obtained by solving the ILP model under the first cost function are compared with the results obtained by simulation. Moreover, optimal solutions using different cost functions for different network configuration are compared.

## ملخص الرسالة

الاسم : تيسير احمد يوسف الخضور

عنوان الرسالة : دراسة التحسين الامثل باستخدام الطبقات المتداخلة لشبكات المجسات اللاسلكية الخاصة بتطبيقات

معينة

التخصص : علوم وهندسة الحاسب

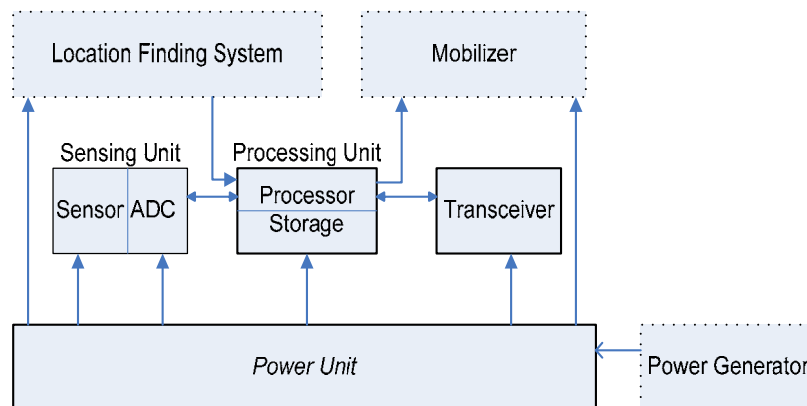
تاريخ التخرج : يونيو 2009

في هذه الرسالة تم دراسة عملية توصيل البيانات في شبكات المجسات اللاسلكية . حيث تم تحديد العوامل الرئيسية التي تؤثر في فاعلية عملية توصيل البيانات من مختلف المجسات الى المركز الرئيسي للشبكة. وبناء على دراسة البروتوكولات الموجودة حاليا و التي تتحكم في عملية توصيل البيانات فقد تم اقتراح اطار عمل عام يتحكم في توصيل البيانات . وقد تم اعتماد طريقة الطبقات المتداخلة في تصميم هذا الاطار العام. ويهدف هذا الاطار العام الى زيادة عمر شبكة المجسات اللاسلكية. وتم تسمية هذا الاطار العام بـ: البروتوكول العام المعتمد على الزمن والموفر للطاقة. في هذا الاطار العام يمكن لأي مجس الاتصال بالمركز الرئيسي مباشرة ، حيث يتم بناء شجرة لربط جميع المجسات و توصيلها بالمركز الرئيسي ، و بناء على هذه الشجرة يتم بناء جدول زمني لهذه المجسات لارسال بياناتها الخاصة. و قد تم فحص الاطار المقترح باستخدام شبكات و تطبيقات مختلفة حيث اظهر هذا الاطار العام تحسنا ملحوظا من حيث عمر الشبكة و الطاقة المستهلكة و كمية البيانات المرسله للمركز الرئيسي مقارنته مع بعض البروتوكولات الموجودة حاليا. و بالاضافة الى ذلك فقد تم في هذه الرسالة اقتراح حالة خاصة من هذا الاطار العام ، وقد سميت هذه الحالة الخاصة بـ: بروتوكول توصيل البيانات الموفر للطاقة. في هذا البروتوكول الخاص تستطيع المجسات القريبة فقط من المركز الرئيسي الاتصال مباشرة بالمركز الرئيسي ، وقد تم مقارنة هذا البروتوكول الخاص ببعض البروتوكولات الموجودة حاليا حيث اظهر كذك تحسنا ملحوظا من حيث عمر الشبكة والطاقة المستهلكة و كمية البيانات الواصلة للمركز الرئيسي. وقد تم كذلك في هذه الرسالة تعميم بروتوكول البيانات المركزية و المدرك للطاقة حيث يمكن لأي مجس في البروتوكول العام الاتصال بالمركز الرئيسي مباشرة و تم مقارنة البروتوكول العام بالبروتوكول الاصلي حيث اظهر تحسنا ملحوظا . بالاضافة الى ذلك فقد تم في هذه الرسالة اقتراح مقياس جديد لقياس كمية البيانات الواصلة للمركز الرئيسي في البروتوكولات الهيكلية لشبكات المجسات اللاسلكية ، حيث تم تعريف المقياس الجديد على انه كمية المعلومات الواصلة للمركز الرئيسي. و تكمن اهمية هذا المقياس الجديد في انه يوفر فهم اوضح للبروتوكولات الهيكلية الخاصة بتوصيل البيانات. ولاستكشاف الطول المثالية التي يمكن الحصول عليها اذا كانت جميع المعلومات الخاصة بشبكة المجسات اللاسلكية معروفة فقد تم في هذه الاطروحة اقتراح نموذج برمجة خطي عددي للبروتوكول المقترح . وتم اقتراح اربعة دوال تكلفة مختلفة لهذا النموذج الخطي ، و قد استخدم برنامج لينجو لحل هذا النموذج باستخدام الدوال المختلفة. حيث تم مقارنة نتائج الحل الناتجة عن هذا النموذج الخطي بنتائج حلول البروتوكول المقترح ، و بالاضافة الى ذلك فقد تم مقارنة نتائج الحل المختلفة الناتجة عن حل النموذج الخطي باستخدام دوال التكلفة المقترحة .

# Chapter One

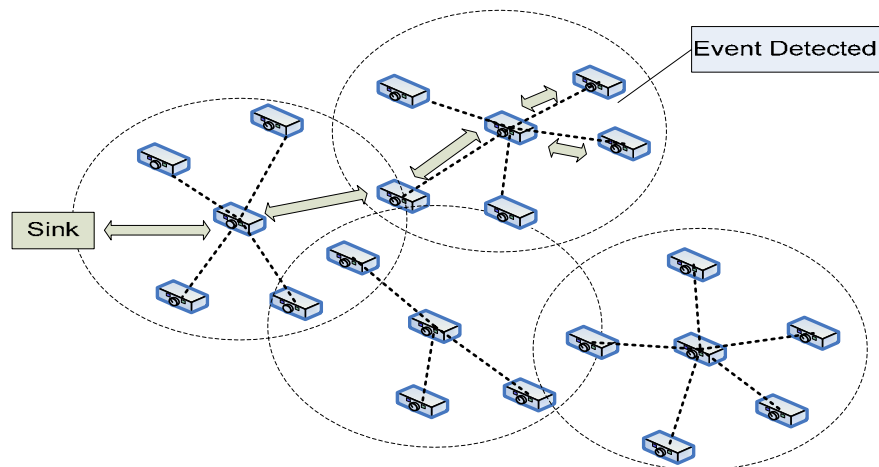
## INTRODUCTION

Recent Advances in Micro Electro Mechanical System (MEMS) has enabled the development of smart sensors. The overall architecture of sensor node consists of the processing subsystem, the sensing subsystem and the communication subsystem [1]. The sensing subsystem is responsible for collecting data from the monitored environment. The processing subsystem processes the collected data before it is transmitted by the communication subsystem. Sensor nodes have low cost, low power, and small size. Figure 1-1 shows a block diagram of a sensor node [1].



**Figure 1-1 Components of Sensor Nodes**

These advancements along with the advances in wireless technology have enabled the creation of Wireless Sensor Networks (WSN) [1]. A WSN is composed of a large number of sensor nodes that are communicating using a wireless medium. Figure 1-2 shows an example of WSN. The sensor nodes are deployed in the environment to be monitored in ad hoc structure. In WSN, there is a sink node that collects data from all sensors. Since a given node could not hear all other nodes in the WSN, WSN is considered a multi-hop network. Within WSN, sensor nodes not only perform sensing functionality but also provide forwarding service. Individual nodes work together to forward data to its final destination.



**Figure 1-2: A wireless Sensor Network**

WSN has many advantages over individual sensors. It extends the range of sensing; it covers a wider area of operation. In WSN, multiple nodes are close to each other, which increase the fault tolerance. On the other hand, the sensor nodes collaborate and combine their data to increase the accuracy of sensed data.

WSN has many potential applications in the physical world. The applications are extended over a wide range from military applications to commercial applications. In military applications WSN can be used, for example, for monitoring friendly forces, for detection and reconnaissance of Nuclear, biological and chemical attacks, and for battle damage assessment. Moreover, there are many environmental applications for WSN such as: forest fire detection, animal migration and flood detections. Tele-monitoring of human physiological data, Tracking and monitoring patients inside a hospital, and drug administration in hospitals are examples of recent health applications for WSN. Moreover, WSN is used in home applications for home automation and smart environment. Finally, WSN has a lot of other commercial applications such as: Environmental control in office buildings, Interactive museums, Managing inventory control, Vehicle tracking and detection, and Detecting and monitoring car thefts.

### 1.1 DISTINGUISHED FEATURES OF WSN:

Although a WSN is a wireless multi-hop network, it has distinguished features over the traditional multi-hop wireless networks. The easy deployment of sensor nodes, the energy constraints, and QoS requirements of WSN make WSN unique compared with traditional multi-hop ad hoc network. These features must be taken into account when designing different protocols that control the operation of WSN such as the MAC and routing protocols. In the following four subsections we discuss these features.



### 1.1.1 THE DEPLOYMENT OF SENSOR NODES:

Sensor nodes are usually deployed randomly. There is not a predefined infrastructure of the established network. Some of the nodes are very close, while other nodes may be dispersed. Existing of many nodes close to each other generates redundancy in the data collected from the environment. Therefore, it is not required to transmit all the sensed data. To reduce redundancy in the data, data aggregation can be performed at each intermediate sensor node. Data aggregation will reduce the data traffic in the network. The aggregation function depends on the running application. Examples of aggregation functions are SUM, AVG, MEAN and MAX. For example, if it is required to measure the maximum temperature in the monitored area, the MAX function will be used as an aggregation function. Unlike the traditional ad hoc network, the number of sensor nodes in WSN is very large. Hundreds or even thousands of sensor nodes are usually deployed in the monitored environment. Hence, it may not be visible to build a global addressing scheme due to the deployment of huge number of sensor nodes. However, there are recent efforts to account IP addresses for sensor network

### 1.1.2 THE RESOURCES CONSTRAINTS

Sensor node has limited resources, for example, all the sensor nodes have a limited power supply (batteries). In most WSN applications, all the sensor nodes are out of control, it is almost impossible to replace or recharge these batteries. All the control protocols of the WSN must be designed taking into account the energy constraint. These protocols must be energy efficient. Another example of limited resources in the sensor

node is the radio transceiver. The transmission range for radio transceiver of the sensor node is very limited. For example, the transmission range of MICAz from crossbow, which is a well-known wireless sensor node, is 20 *m* to 30 *m* for indoor environment, and it is less than 100*m* for the outdoor environment [2]. Hence, not all the sensor nodes in WSN can hear all other nodes. Therefore, all the sensor nodes must collaborate to transfer data from the source sensor node to the sink. The limited transmission range of the sensor node must be considered when designing routing protocols for WSN.

### 1.1.3 THE QUALITY OF SERVICE (QOS) REQUIREMENT

Different WSN applications needs different QoS requirement. For example, data latency in WSN is very critical in some applications and not so much in other applications. For example, if the status of sensed object is changing very frequently, then the data latency must be very low. Otherwise, the data collected from the environment will not be valid when it reaches the sink. In other applications, the time between two successive events may be very long. In this case, high data latency could be tolerated. On the other hand, in some WSN application, the end user does not need all the data in network. Data collected from neighboring node will be highly correlated. Hence, it is not required to send all these data to the end user. Usually, a user requires a high-level description of events being monitored in the environment.

### 1.1.4 DATA TRAFFIC MODELS

In WSN, data traffic models can be classified into: periodic, event-driven, and query based. In periodic traffic model, the sensor nodes send their measurements to the sink

once every fixed time interval. In this model, all sensor nodes must be synchronized. In event driven model, data traffic flows in the network when a specific event is detected. The events must be reported immediately to the sink when they are detected. On the other hand, in query-based model, data traffic flows from sensor to sink in response to a query generated from the sink. The sink generates a specific query then the relevant sensor node will respond to the query with the requested data. A route must be computed for the query and for the data transmission. In most applications of WSN, data traffic usually flows from multiple sources to one destination, which is not the case in the traditional ad hoc wireless network.

## 1.2 MOTIVATION:

Having discussed the distinguished characteristics of WSN, it should be clear that it is impossible to implement the routing protocols of the traditional ad hoc network directly to the WSN. Although many routing protocols are proposed for WSN, Most of them try to minimize the energy consumption. Some of the routing protocols assume unrealistic assumptions, for example, in Low-Energy Adaptive Clustering Hierarchy (LEACH) protocol [3], it is assumed that all the sensor nodes can hear each other. As we discussed above, sensor nodes are usually deployed in wide area, and the transmission range for a sensor node is very short. Therefore, in most applications, not all sensors will hear each others. Moreover, with this assumption, a node may transmit data to one of the farthest nodes which increases energy consumed in data transmission.

On the other hand, energy consumption in most protocols is not optimized according to the applications. For example, in Energy-Aware Data-Centric (EAD) routing protocol

[4], most of the sensor nodes (non-leaf nodes) stay awake for all the time, therefore they will lose more energy. They will die early. The network lifetime will be very short. For periodic applications, it is not necessary for a node to be awake all the time if the moment of data reporting is known. It is better to schedule the node to be ON at the expected time of reporting the event.

On the other hand, several routing protocols are designed without taking into account the underlying MAC protocol, which may increase the delay, for example, in the S-MAC protocol [5], a node will go into listening mode and sleeping mode alternatively. For a specific routing protocol, assuming that the next hop for a node  $x$  is the node  $y$ , and node  $y$  is currently in sleeping mode, then node  $x$  must wait until node  $y$  becomes in its listening mode, which increases the delay.

The need of more optimal energy-efficient routing protocols for WSN and the potential applications of the WSN motivate our research in data forwarding protocols for WSN.

### 1.3 PROBLEM STATEMENT

In WSN, many sensor nodes are deployed in the field to monitor a certain phenomenon. The sensor nodes must cooperate to transfer the data to the sink. Since most of sensor nodes are out of control and since they have limited power resources, the sensor nodes must consume a little amount of power during their operation. At the same time, the data must reach the sink as soon as possible. In addition to the energy consumed due to receiving and transmitting data packets, four main sources of wasted energy in the wireless node are classified. The first source is collisions which will cause retransmission

of data packet. Retransmission of data packets will consume more energy. The second source is overhearing; picking a packet intended to another node. A node will lose energy in receiving non-required data packets. The third source of energy consumption is transmission of control packets. A protocol with more control packets will increase the energy consumed by the node to transmit these packets. The final source of energy consumption is idle listening. If the sensor is in idle listening state, then the power is merely consumed for sensing the channel. The power consumed during the idle-listening state is about 50%-100% of the power consumed during transmitting or receiving. It has been shown that the idle:receive:send power consumption ratios are 1:1.05:1.4 [6]. For a control protocol to be energy-efficient, it must minimize the energy consumed due to these four sources. To minimize the energy consumed due to collision, sensor nodes must be scheduled to transmit at different time slots. Scheduling nodes for transmission is performed at MAC layer. To reduce the power consumed due to overhearing, a node must be only ON at time slots where it is expected to receive a packet which depends on the routing protocol that is performed at the network layer. To minimize the energy consumed due to transmission of control packets, the control protocol must be designed with minimum control packets. Finally, to minimize the energy consumed due to idle-listening, the node must be OFF when it has no data to transmit and when it is not expected to receive data from other nodes.

It is obvious that the four reasons of energy consumption are correlated. And they are related to MAC and Network layers. Designing two separate energy efficient protocols, MAC layer protocol and Routing layer protocol, may not always reduce the total energy consumed in all nodes, for examples:

- Designing two separate protocols will increase the number of control packets. Each protocol will have its own control packets. More energy will be consumed by the node. Therefore, integration of the two protocols in one protocol is expected to reduce the number of control packets.
- Designing a MAC protocol in which a node can be ON and OFF alternatively, similar to S-MAC protocol, will reduce the energy consumed due to idle listening. On the other hand designing a routing protocol that selects the closest node to the current node to be the next hop may decrease the transmission energy. However, applying the two protocols together may not be energy-efficient. For example, consider the following scenarios:
  - Assume that according to the routing protocol, node  $x$  is the next hop to node  $y$ . Assuming node  $x$  is currently OFF according to the MAC protocol, then node  $y$  must wait in idle listening state until node  $x$  become ON, node  $y$  will lose energy while it is waiting in idle listening state. Moreover, the delay will increase also.
  - Assume that a node is currently ON, but it is not a next hop for other nodes, it will lose energy due to overhearing.

#### 1.4 DESCRIPTION OF THE WORK

In this section, we present a brief description of our work. Firstly in the following subsection, we will present the methodology we followed in our research. Then we will present a brief description of the thesis contributions.

### 1.4.1 METHODOLOGY:

The main objective of our research is to design a framework for application-based routing protocols. To achieve this objective, we followed the following methodology:

1. We identify the main parameters that play key roles in determining the performance of a WSN. We observed that to maximize network lifetime, we have to minimize energy consumed by each node. Energy consumed by each node can be minimized by reducing time intervals in which a node is in idle listening state, reducing the transmission energy, and increasing the time interval in which a node is in OFF state.
2. According to parameters identified in step-1, we design cross layer framework to forward data from sensor nodes to the sink. In our proposal, based on the application, network and MAC layers are integrated. In our framework, to reduce energy wasted due to idle listening state, a time division multiple access (TDMA) scheme is implemented for data transmission among sensors. To reduce energy wasted due to transmission, each node in our framework will communicate with one of its neighbors. Therefore, a tree is built from all nodes towards the sink. To balance energy consumption among nodes, our framework runs in rounds, in each round a different tree will be constructed according to the residual energy of each nodes. To utilize all the energy stored in the nodes, we integrate within our framework a mechanism to select nodes that are connected directly to the sink.
3. We evaluate the performance of the proposed protocols by simulation. Performance evaluation process includes:

- a. Identifying performance evaluation metrics that can truly distinguish the best performer among several proposals. Examples of these metrics are: network lifetime, throughput, delay, energy consumption.
  - b. Using the above metrics to compare the proposed protocols with other well know existing protocols.
4. To compare the performance of our protocols with optimal solutions which are considered as theoretical solutions and are generated assuming global information, we propose an integer linear programming (ILP) model. We solve the ILP model using LINGO solver [7] for different network configurations. The results obtained by solving the ILP model are compared with the results obtained by simulation.

#### 1.4.2 RESEARCH CONTRIBUTIONS:

In our research, the main parameters that affect the operation of data forwarding from sensor nodes to sink are identified. Based on our study of the existing protocols, we proposed our own framework to forward data from sensor nodes to sink. A cross layer design methodology is adopted in designing our framework. According to the selected application, MAC and Network layers will be integrated. The proposed framework is called a Generalized Energy-Efficient Time-Based communication protocol (GET). Moreover, a special case of GET protocol which is called an Energy-Efficient Distributed Schedule-Based communication protocol (EEDS) is proposed. To compare the results obtained from simulation with optimal solutions, we proposed integer linear



programming (ILP) model for our protocols. Moreover, we proposed an entropy-based throughput metric for fairly evaluating routing protocols of WSN.

- A Generalized Energy Efficient Time Based protocol (GET): GET intends to increase network lifetime by minimizing energy consumed by each node. Energy consumed in each node is minimized by decreasing the amount of time in which a sensor node is in idle listening state. Moreover, GET intends to utilize all the initial energy stored in sensor nodes. Initial energy is fully utilized by minimizing the idle time of nodes. The time in GET is divided into rounds, each round consists of four phases; selecting gateways (nodes connected directly to sink) (SG), building tree (BT), building schedule (BS) and data transmission (DT). In GET, gateways can be changed during network lifetime. Different nodes can act as gateways. A mechanism to select gateways based on the residual energy of the nodes is proposed. The selected node will act as a gateway as long as its residual energy is greater than a threshold value  $E_{th}$ . New gateways will be selected each round. In building tree phase, an energy efficient tree from all nodes towards the sink is built. Building tree process is initiated by gateways. Since gateways differ from round to round, different trees will be constructed each round. Then, based on this tree, a TDMA schedule is built in building schedule phase. In data transmission phase, the schedule is followed to forward data from all nodes to the sink. The data transmission phase may be repeated multiple times in a single round. Performance evaluation of GET shows an improvement in terms of network lifetime, throughput and percentage of covered area compared to EAD and LEACH.

- Energy Efficient Data Communication Protocol (EEDS): EEDS is a special case of GET, only the nodes that are close to the sink will act as gateways. These nodes will be connected directly to the sink. no gateways will be selected in each round, therefore, each round in EEDS consists of three phases; building tree (BT), building schedule (BS) and data transmission (DT). Building tree process will be initiated by the sink. Although performance evaluation of EEDS shows an improvement in terms of network lifetime, throughput and percentage of covered area compared to well-known WSN protocols such as LEACH and EAD, we observed that when gateways die, the remaining nodes of the network will be isolated although they still have energy.
- A Generalized Energy-Aware Data Centric Routing For WSN: to improve the performance of energy aware data centric routing protocol (EAD), we have implemented selecting gateway mechanism within it. We called the new protocol  $EAD_{General}$ .  $EAD_{General}$  intends to increase the lifetime of the network by increasing the number of candidate gateway nodes. Extensive simulation results show that  $EAD_{General}$  protocol outperforms EAD in terms of the network lifetime for all different network configurations.
- An Entropy-Based Throughput Metric For fairly Evaluating WSN Routing Protocols: from our study of different routing protocols of WSN, we observed that in hierarchal routing protocol, a packet delivered to the sink is resulted from aggregating many raw packets. Two different packets delivered to the sink may be resulted from the aggregation of different number of raw packets. These two packets may carry different amount of information. Therefore considering

throughput as the absolute number of data packets delivered to the sink is not accurate. Hence, we proposed Information-Entropy based metric to measure the throughput. In the new metric, we define the throughput as the amount of information delivered to the sink. The proposed metric yields a better understanding of the operation of the WSN application under consideration. Moreover, it fairly compares different hierarchical routing protocols in which the clusters are formed using different techniques. We used the proposed metric to compare our proposals with different well-known routing protocols such as: EAD and LEACH.

- Integer linear programming (ILP) formulation: to explore the optimal solutions that can be produced assuming global information, we proposed an ILP model for our proposal. We used LINGO solver to solve our model. The results obtained by solving the ILP model are compared with the results obtained by simulation. Moreover, optimal solutions using different cost functions for different network configuration are generated. The performance of these different solutions is evaluated.

## 1.5 PUBLICATIONS

Based on our research, we published the following:

- Journal papers
  - T. AL-khdour, U. Baroudi “A Generalized Energy-Efficient Time-Based Communication Protocol for Wireless Sensor Networks”, Special issue of International Journal of Internet Protocols (IJIPT), Vol. 4, No. 2-2009.

- T. AL-khdour, U. Baroudi “An Energy-Efficient Distributed Schedule-Based Communication Protocol for Wireless Sensor Networks” submitted to the Arabian Journal for Science and Engineering (AJSE), accepted with minor revisions.
- Conference papers
  - T. AL-khdour, U. Baroudi, “An Entropy-Based Throughput Metric For Fairly Evaluating WSN Routing Protocols,” in the Proc. of the Fifteenth IEEE International Conference on Network Protocols, ICNP2007, China, Beijing, Oct. 16-19. pp. 342-343.
  - T. AL-khdour, U. Baroudi, “ A Generalized Energy-Aware Data Centric Routing For Wireless Sensor Network”, in the Proc. of The 2007 IEEE International Conference on Signal Processing and Communications (ICSPC 2007) , Dubai, United Arab of Emirates (UAE), Nov. 24–27.
  - T. AL-khdour, U. Baroudi, “The Effect of Network Topology on Performance of Routing Protocols for Wireless Sensor Networks” submitted to the 12-th ACM International Conference on Modeling, analysis and simulation of Wireless and Mobile Systems (MSWiM 2009).

## 1.6 DISSERTATION ORGANIZATION

In addition to introduction, this dissertation consists of 7 chapters. Chapter 2 presents literature review of MAC, Network, and cross layer protocols for WSN. In chapter 3 we describe the Generalized Energy Efficient Time-Based Protocol (GET), an Energy Efficient Distributed Schedule Based Protocol (EEDS), and a Generalized Energy-

Aware Data Centric Routing ( $EAD_{General}$ ). Our simulation setup will be presented in chapter 4. Performance Evaluation of our proposals will be discussed in chapter 5. Chapter 6 presents performance evaluation of both GET and EEDS under different network configuration and different applications. In chapter 7, we present the integer linear programming model, and we compare the results obtained by solving ILP model with the results obtained by simulation. Chapter 8 concludes the dissertation. Potential future work is presented in Chapter 9.

# Chapter Two

## LITERATURE REVIEW

In this chapter, we will survey the literature existing for MAC, routing protocols, and cross layer design protocols that are proposed for WSN. In section 2.1 we discuss some MAC protocols for WSN. The routing protocols for WSN are described in section 2.2. Finally, the cross layer design protocols are discussed in section 2.3

### 2.1 MAC PROTOCOLS FOR WSN

In designing a MAC protocol for a Wireless Sensor Network (WSN), some of the unique features of WSN must be taken into consideration. Low-power consumption must be the main goal of the protocol. The coordination and synchronization between nodes must be minimized in the protocol. The MAC protocol must be able to support a large number of nodes. It must have a high degree of scalability. The MAC protocol must take into account the limited bandwidth availability. Since sensor nodes of a WSN are deployed randomly without a predefined infrastructure, the first objective of the MAC

protocol for a WSN is the creation of the network infrastructure. The second objective is to share the medium communication between the sensor nodes [1].

IEEE 802.11 is a well-known MAC protocol for Ad hoc network [8]. In IEEE 802.11 protocol, each node will be in one of the three states, sending, idle-listening, and receiving. In the idle-listening state, the node does not do any thing except sensing the medium to check if any node sends RTS to it. Sensing the medium will consume power. The power consumed during the idle-listening state is about 50%-100% of the power consumed during transmitting or receiving. It has been shown in [6] that the idle: receive: send power consumption ratios are 1:1.05:1.4. The energy constraints in the sensor nodes make it is unpractical to apply the IEEE 802.11 protocol directly in WSN. IEEE 802.11 has a power save mode. The power save mode in IEEE 802.11 is designed for a single hop network, where all nodes can hear each other. This is not the case in WSN.

A set of MAC protocols for the WSN were proposed. Most of the existing protocols aimed to save power consumption in the sensor nodes. Since most of WSN MAC protocols aimed to reduce the power consumption we will describe firstly a Power Aware Multi-Access protocol with signaling for Ad hoc Networks (PAMAS) [9]. We explain this protocol because it is used as a base for some WSN MAC protocols.

### 2.1.1 POWER AWARE MULTI-ACCESS PROTOCOL WITH SIGNALING FOR AD HOC NETWORKS (PAMAS) PROTOCOL

PAMAS [9] is a channel access protocol that reduces the power consumption at each of the nodes in a general Ad-Hoc network. In an Ad Hoc network, if a node transmits a packet, all the close nodes will hear its transmissions even if the packet is intended to one

of them only. Overhearing the unwanted packet by these nodes will consume a power without gaining any useful data. PAMAS aims to reduce the power consumed due to receiving packets that are intended to another node. PAMAS protocol is a combination of the MACA protocol and the idea of using a separate signaling channel. In PAMAS, the RTS-CTS message exchange occurs over a signaling channel (control channel) separated from the packet transmission channel. The separate channel enables the nodes to determine when and for how long they can power themselves off. Each node in PAMAS protocol can be in one of six states:

- **Idle:** the node currently does not transmit or receive and it has not packet to send, but its RF receiver is on to be able to receive packets.
- **Await packet:** the node is waiting for data packets from a source node after it transmits Clear-To-Send (CTS) packet to that node. The node firstly receives a Request-To-Send (RTS) from the source node then it waits one time step before replying with CTS.
- **Await CTS:** the node sent RTS and it is currently waiting for the CTS.
- **Receive packet:** the node is currently receiving a packet.
- **BEB:** Binary Exponential Backoff time; the node is waiting for random backoff time.
- **Transmit packet:** the node is currently transmitting a packet.

Initially a node will be in the idle state. Upon receiving an RTS message, the node will wait for one time step, if no neighbor node is currently transmitting, i.e. it can receive data without causing collision, and then the node sends the CTS message. The node waits for a one time step to ensure that no neighbor node is waiting for the CTS



form another node. After sending CTS, the node will go to the Await packet state. If it receives a new RTS packet, the node will resend the CTS packet. If the data packets start arriving the node will transmit a busy tone over the control channel and it will go to the Receive packet state. If the node receives an RTS directed to another node over the control channel while it is in the Receive packet state, the node will transmit a busy tone over the control channel. The node that sends the RTS will be blocked when it hears the busy tone. When all the packets are received, the node will go back to the idle state. If the node is in the Await packet state, and it does not receive data packets within the expected time (round trip time to the transmitter plus some processing delay at the receiver) or it receives a noise, it will go back to the idle state.

If the node is in the idle state, and it receives packets from upper layer, and its queue size becomes greater than 0, it will send RTS to destination node and goes to Await CTS state. If it receives the CTS, it goes to Transmit packet state and it starts transmitting its data. While transmitting data the node will ignore RTS/CTS packets. When the transmission ends, the node goes back to the idle state. If the node receives RTS while it is in Await CTS state, it will send CTS and it will go to the Await packet state. If the node in the Await CTS state and it does not receive CTS within the expected time or it receives a busy tone, it will go to BEB state. It selects a random backoff time for waiting. When the selected random time is expired, the node will send RTS and go back to the Await CTS state. If the node receives RTS while it is in the BEB state, it will send CTS and go to the Await packet state.

if a node has no packet to transmit, and if one of its neighbors starts to transmit to another node, or if one of the neighboring nodes is transmitting and another one is

receiving, then the node will turn itself OFF. The node knows that one of its neighbors is transmitting by hearing data in the transmission channel. It knows that one of its neighbors is receiving by hearing the busy tone in the control channel. To determine a node's OFF duration, there will be two cases, the first case; a neighbor of the node starts transmission while the node is ON, the node determines the duration of its OFF period from the RTS duration field. The second case; its neighbor starts transmission while the node is OFF, the node will use a probe protocol to determine the OFF duration [9]. If a node wishes to transmit a packet to its neighbor while that node turns itself OFF, then the transmitter node must wait until its neighbor wakes up. This will not increase the delay since the destination node is OFF because one of its neighbors is currently transmitting so it cannot receive data correctly.

Singh et al. measure the performance of the PAMAS by simulation for different network topology: random, line, and fully connected with dense and sparse networks. They note that there is a large energy saving in PAMAS. They also derive bounds and approximations on energy saving in the different network topology.

Although there is power saving in nodes due to power off mode on nodes, using a separate channel to transmit control messages will cost some power. This is considered as a disadvantage of the PAMAS. PAMAS is extended by Wei Ye and others to propose the Sensor MAC protocol (S-MAC) [5].

### 2.1.2 S-MAC PROTOCOL

The main goal of S-MAC protocol is to reduce energy consumption while supporting good scalability and collision avoidance. Wei Ye et al extend PAMAS protocol by using

a single channel for transmitting data packets and control packets. In designing S-MAC protocol they assume the following about sensor networks and applications:

- Sensor network composed of many small nodes deployed in an Ad Hoc fashion. They take the advantage of physical proximity to simplify signal processing.
- Most communication will be between nodes as peers rather than one base station.
- The sensor nodes must be self configured.
- The sensor network is dedicated to a single application or a few collaborative applications. The focus will be on maximizing system-wide application performance. The single packet delay will be a secondary goal.
- The sensor network has the ability of in-network processing. Data aggregation can reduce the traffic in the network.
- The intended application can tolerate some latency. The monitored object has long idle periods.

The basic idea of S-MAC is to let the node sleep and listen periodically. In sleeping mode, the node turns its radio off. The listening period is fixed according to physical layer and MAC layer parameters. The complete cycle of listening and sleeping periods is called a frame. The duty cycle is defined as the ratio of the listening interval to the frame length. Neighboring nodes can be scheduled to listen and sleep at the same time. Two neighboring nodes may have different schedules if they are synchronized by different two nodes. Nodes exchange their schedule by broadcasting a SYNC packet to their

immediate neighbors. The period to send a SYNC packet is called the synchronization period. If a node wishes to transmit a packet to its neighbor it must wait until its neighbor becomes in its listening period. Figure 2-1 shows 4 neighboring nodes *A*, *B*, *C*, and *D*. Nodes *A* and *C* are synchronized together (they have the same schedule, they listen and they sleep at the same time) while nodes *B* and *D* are synchronized together.

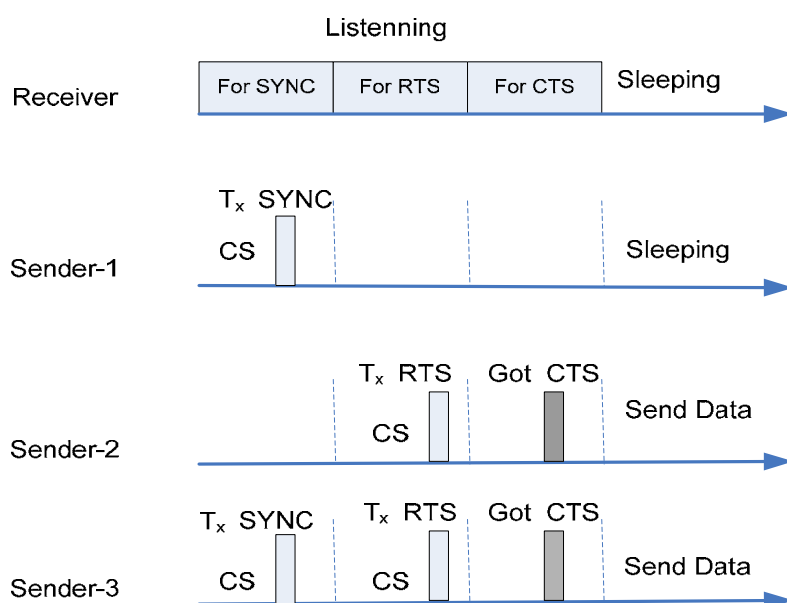


**Figure 2-1: S-MAC: Neighboring nodes A and B have different schedules. They synchronize with nodes C and D respectively**

S-MAC forms nodes into a flat, peer-to-peer topology. To choose a schedule the node firstly listens for a fixed amount of time (at least the synchronization period). If the node does not receive a schedule within the synchronization period, the node chooses its own schedule and starts to follow it, and then it announces its schedule to its neighbors by broadcasting the SYNC packet. If it hears a schedule from one of its neighbors before it chooses or announces its own schedule, it follows that schedule. If a node receives a different schedule after it announces its own schedule, then there will be two cases, in the first case, the node has not other neighbors, then it discards its own schedule and it will follow the new schedule. In the second case, the node already follows a schedule with one of its neighbors; therefore it will adopt both schedules by waking up at the listening intervals of the two schedules. To maintain the schedule, each node maintains a schedule table that stores the schedules of all its known neighbors. To prevent case two in which neighbors miss each other forever when they follow two different schedules, a periodic neighbor discovery is introduced. Each node periodically listens for the whole

synchronization period. Figure 2-2 shows the timing relationship of three possible situations.

If multiple nodes wish to talk to the same node that is in listening period, then all of them must contend for the medium. IEEE 802.11 scheme with RTS and CTS is used to avoid collision, which will save energy consumption due to the packets collision and retransmissions.



**Figure 2-2: Timing relationship between a receiver and different senders, CS stands for carrier sense**

To avoid overhearing which is one of the sources of energy consumption, each interfering node must go to sleep after it hears RTS and CTS. All immediate neighbors of both sender and receiver should sleep after they hear RTS or CTS. To reduce the delay due to sleeping, a technique called adaptive listening is integrated in S-MAC. Each node will wake up for a short period at the end of the transmission. In this way, if the node is

the next-hop node, its neighbor is able to pass the data immediately to it instead of waiting for its scheduled listening time.

To reduce energy due to control packet overhead a message passing technique is included in S-MAC. If a node wishes to transmit a long message, the long message is fragmented into fragments and the node will transmit them in burst; one RTS and one CTS are used for all the fragments. When a node sends data, it waits for ACK. The ACK is useful to solve the hidden terminal problem. Data fragment and ACK packets have a duration field. If a node wakes up or joins the network and it receives a data or ACK packet, it will go to sleep for the period in the duration field in data or ACK packet.

Synchronization among neighboring nodes is required to remedy their clock drift. Synchronization is achieved by making all nodes exchange a relative timestamps and letting the listening period is longer than clock drift.

The average latency of S-MAC without adaptive listening over  $N$  hops is

$$E[D(N)] = NT_f - T_f + t_{cs} + t_{tx} \quad (1)$$

While the average latency of S-MAC with adaptive listening over  $N$  hops is

$$E[D(N)] = \frac{NT_f}{2} - T_f + 2t_{cs} + 2t_{tx} \quad (2)$$

Where

$N$  : Number of hops.

$T_f$  : frame time (complete cycle of listening and sleep)

$t_{cs}$  : The carries sense delay

$t_{tx}$  : The transmission delay for a packet with fixed length.

A disadvantage of S-MAC is that the listening interval is fixed regardless whether the node has data to send or there are data intended to it. Suh et al proposed a Traffic Aware, Energy Efficient MAC protocol for Wireless Sensor Networks (TEEM) [10]. They extend the S-MAC protocol by reducing the listening interval.

### 2.1.3 A TRAFFIC AWARE, ENERGY EFFICIENT MAC PROTOCOL FOR WIRELESS SENSOR NETWORKS (TEEM).

The TEEM protocol is an extension to S-MAC protocol. In S-MAC protocol the listening interval is fixed while in TEEM protocol the listening interval depends on the traffic. In TEEM protocol; all nodes will turn their radio off much earlier when no data packet transfer exists. Furthermore, the transmission of a separate RTS is eliminated. In TEEM protocol; each listening interval is divided into two parts instead of three parts as in S-MAC protocol. In the first part of the listening interval, the node sends a SYNC packet when it has any data message ( $SYNC_{data}$ ). If the node has no data message, it will send a SYNC packet ( $SYNC_{nodata}$ ) in the second part of its listening interval.  $SYNC_{data}$  is combined with RTS packet to form  $SYNC_{rts}$ . If a node does not receive  $SYNC_{data}$  in the first part of its listening interval and it has no data to send it will send  $SYNC_{nodata}$  in the second part of its listening interval. If a node receives a  $SYNC_{rts}$  that is intended to another node, it will turn its radio off and goes to sleep until its successive listening interval starts. The intended receiver will send CTS in the second part of its listening interval. The performance evaluation of TEEM protocol shows that the percentage of sleeping time in TEEM is greater than the percentage of sleeping time in S-MAC. The number of control packets in TEEM protocol is less than the number of control packets in

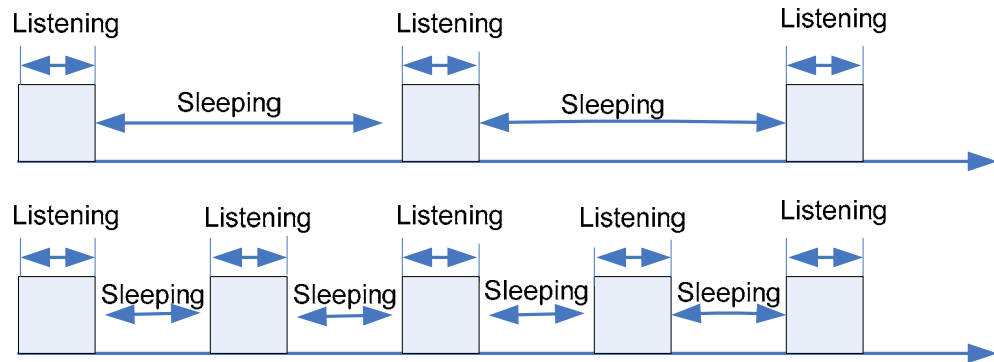
S-MAC protocol. Energy consumption in TEEM protocol is the least compared with S-MAC and IEEE 802.11. Although the power consumption is reduced in the TEEM protocol by decreasing the listening interval, the latency will increase since decreasing the listening interval depends only on the local traffic, traffic in the node itself and in the neighboring node, and does not take into account the traffic in the whole network. To take into account the delay in the whole network, Lin et al proposed a sensor medium access control protocol with a dynamic duty cycle, DSMAC [11]. DSMAC intends to achieve a good tradeoff between power consumption and latency.

#### 2.1.4 MEDIUM ACCESS CONTROL WITH A DYNAMIC DUTY CYCLE FOR SENSOR NETWORK (DSMAC)

In DSMAC, the duty cycle changes based on average delay of the data packet and the power consumption [11]. Duty cycle can be changed by changing the sleeping interval while fixing listening interval. As in S-MAC, the nodes in DSMAC form groups of peers. Each set of neighbors follow a common schedule. In DSMAC, one-hop packet latency is proposed which is the time since a packet gets into the queue until it is successfully sent out. The packet latency is recorded in the packet header and sent to the receiver. The receiver calculates the average packet latency. The average packet latency is an estimation of the current traffic. If the average packet latency is larger than a threshold delay ( $D_{max}$ ), and if the energy consumption level greater than a threshold energy ( $E_{max}$ ), then the duty cycle will be doubled by decreasing the sleeping interval such that the new frame length is half of the original frame length. Otherwise the duty cycle will be halved by doubling the sleeping interval, doubling the sleeping interval will double frame length.



The purpose of changing the duty cycle by two (or half) is to maintain the old schedule, which enables neighboring nodes to communicate using the old schedule. Figure 2-3 shows the schedule before and after doubling the duty cycle.



**Figure 2-3: Neighboring nodes which adopt different duty cycles can still communicate with old schedule**

It is shown analytically in [11] that the average delay in the case of one hop in DSMAC is less than the average delay in S-MAC. It is shown also that the average delay in the case of multiple hops with adaptive listening is:

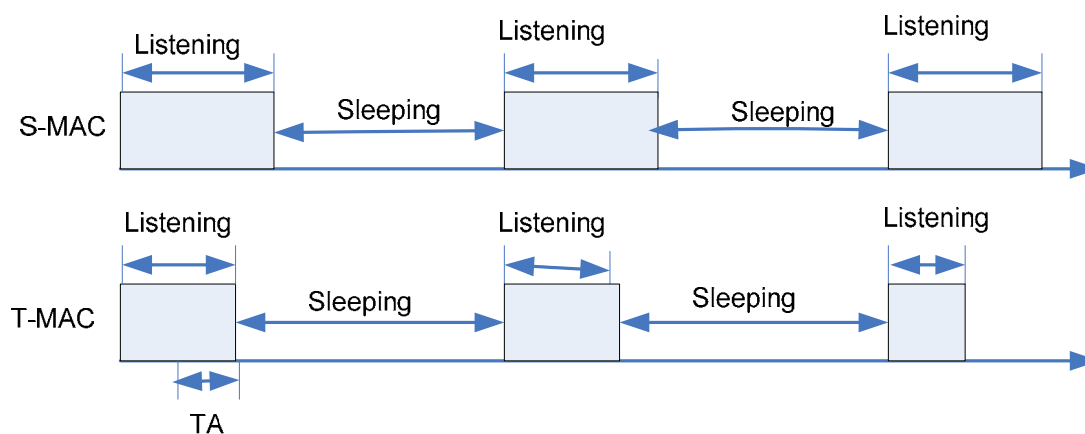
$$E[D(N)] = \frac{NT_f}{8} - \frac{T_f}{8} + 2t_{cs} + 2t_{tx} \quad (3)$$

Comparing equation 3 with equation 2 we note that the delay in DSMAC is less than the delay in S-MAC.

In S-MAC protocol, the listening and sleeping intervals are fixed. Dam et al propose a Timeout-MAC (T-MAC) protocol [12]. In T-MAC, the listening interval of the node may end earlier in some situations.

### 2.1.5 TIMEOUT-MAC (T-MAC)

In T-MAC protocol, the node will keep listening and transmitting as long as it is in an active period. An active period ends when no activation event has occurred for a specific time  $TA$ . An activation event may be firing of a periodic frame timer, reception of any data on the radio, sensing of communication on the radio, end-of-transmission of a node's own data packet or acknowledgement, or the knowledge that a data exchange of a neighbor has ended. A comparison of sleeping and listening periods of the S-MAC and T-MAC is shown in Figure 2-4.



**Figure 2-4: listening sleeping periods in both S-MAC and T-MAC**

Communications between nodes in T-MAC is performed using RTS/CTS mechanism. The node that wishes to transmit data must send an RTS and wait for the CTS. If it does not receive CTS within the  $TA$  period the node will go to sleep. The node does not receive CTS in three cases; the receiver has not received the RTS, the receiver receives RTS but it is prohibited from replying, or the receiver is sleeping. It is accepted and

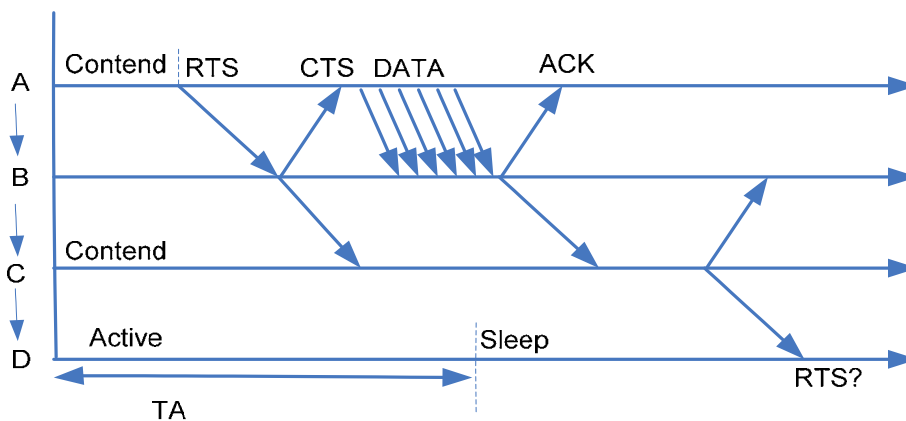
recommended for the node to go to sleeping in the third case. But it is not an optimal decision to go to sleeping in the first two cases. To take into account all the three cases; when the node does not receive CTS to the first RTS it will resend another RTS and if it does not receive a response to the second RTS then it will go to sleeping. Sending two RTS packets without getting a CTS indicates that the receiver cannot reply now so it is convenient for the sender to go to sleeping.

TA must be long enough to receive at least the start of the CTS packet. So TA must be

$$TA > C + R + T \quad (4)$$

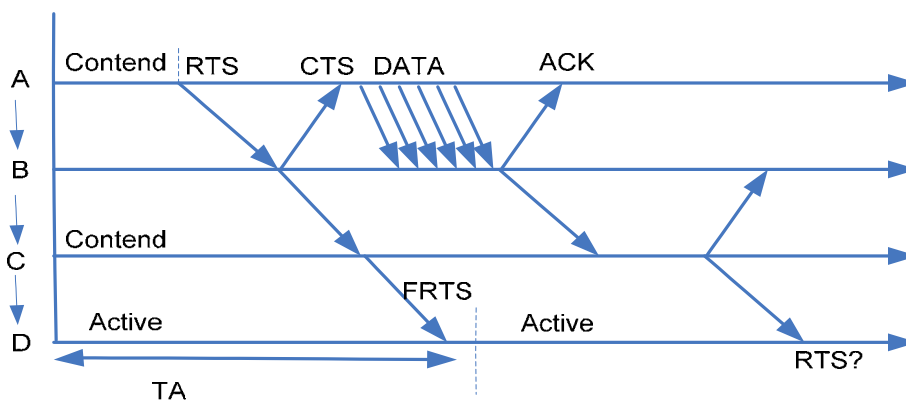
Where  $C$  is the length of contention interval,  $R$  is the length of an RTS packet, and  $T$  is the turnaround time. Overhearing avoidance is achieved by the same technique used in S-MAC protocol.

One problem of the T-MAC is the early sleeping problem, which occurs in case of asymmetric communication where there are four consecutive nodes:  $A$ ,  $B$ ,  $C$ , and  $D$  as shown in Figure 2-5. Node  $A$  sends data packet to  $B$ , the final destination of this packet is  $C$ , at the same time  $C$  wishes to send data to node  $D$  but it cannot transmit data since a collision will occur at node  $B$  with the transmission from  $A$  to  $B$ , so node  $C$  will go to sleeping. Moreover, node  $D$  will go to sleeping. Later when node  $B$  wishes to forward the data to node  $C$ , it will find that node  $C$  is sleeping which will make node  $B$  to go to sleeping and transmit its data later which will increase the delay and decrease the throughput. Two solutions are proposed: future request-to-send and taking priority on full buffers [12].



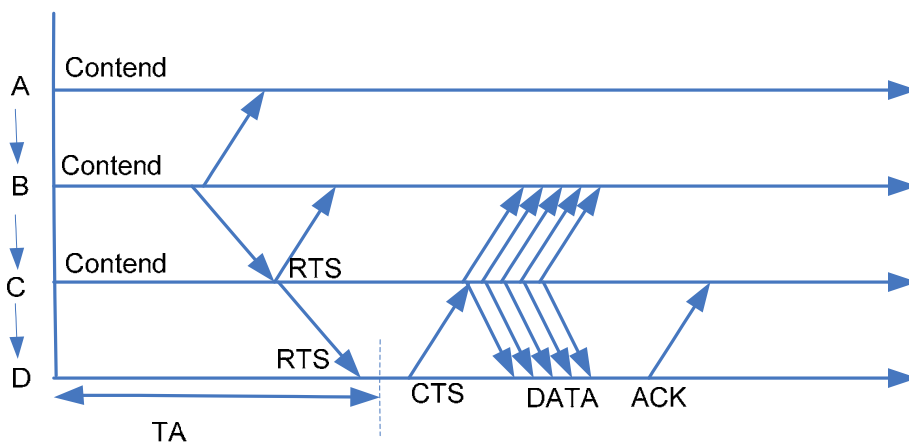
**Figure 2-5: The early sleeping problem. Node D goes to sleep before C can send RTS to it.**

The main idea of the future request-to-send technique is to let another node know that we still have message for it but we are prohibited from using the medium [12]. If a node overhears a CTS packet destined for another node it may send a future-request-to-send (FRTS) packet to that node, for example, node C sends FRTS to node D as shown in Figure 2-6. A node that receives an FRTS packet will know that it will be the future target for an RTS packet and must be awake up by that time.



**Figure 2-6: The future-request-to-send packet exchange keeps Node D awake.**

The second technique for the early sleeping problem is called the full-buffer priority. In this technique, when a node's transmit/routing buffers are almost full, it will prefer sending to receiving. When this node receives an RTS packet destined for it, it sends its own RTS packet to another node instead of replying with a CTS packet. In the previous example, when node *C* receives an RTS packet from node *B* it will send an RTS to node *D* instead of replying with CTS to node *B* as shown in Figure 2-7.



**Figure 2-7: Taking priority upon receiving RTS**

Comparing DSMAC with T-MAC protocol, we note that the purpose of the DSMAC protocol is to decrease the delay by decreasing the sleeping interval between the listening intervals. This may increase the power consumption. On the other hand in TEEM protocol the main objective is decrease the power consumption by making the node sleep earlier and maintaining the frame length fixed. The delay will not decrease in the TEEM protocol

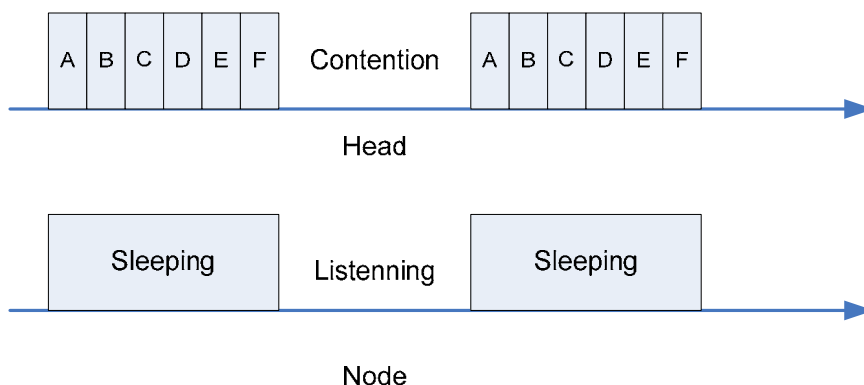
There are some applications, in which most of the traffic in the nodes is a forwarding traffic. For these network models, Biaz et al propose a MAC protocol (GANGS) in which the nodes are organized into clusters [13].

### 2.1.6 GANGS PROTOCOL

Nodes in GANGS are organized into clusters [13]. Each cluster has a head, and the heads form the backbone of the sensor network. The communication between nodes within a cluster is contention based while the communication between heads is TDMA based. The frame is divided into multiple contention free TDMA slots and one contention slot as shown in Figure 2-8. Number of TDMA slots depends on the number of neighboring clusters heads. The radios of all normal nodes will be turned OFF through TDMA slots while the radios of all heads are turned ON through the entire frame.

Establishing the cluster consists of three stages: local maximum stage, inter-cluster stage and reconfiguration stage. In the local maximum stage, the nodes communicate with their neighbors and exchange their energy information. The node that has the local maximum energy claims that it is the head and sends this claim to its neighbors. In the Inter-cluster phase, new heads are added to construct the backbone. Any node that it is not a head may be in the range of one head and accepts it as a head, in the range of multiple heads and it needs to choose one of them, or it is not in the range of any head. If it is in the range of multiple heads and if it has a maximum energy, then it will be the new head, otherwise the node will select the head with the maximum power. If it is not in the range of any head, then it sends a message to a node with local maximum power to demand head service. The node with local maximum power will be the new head. Since

the head consumes more energy, eventually it will no longer have the maximum energy and reconfiguration must be performed to select new heads.



**Figure 2-8: Time frame for cluster head/Node**

As any TDMA based protocol, synchronization between the cluster heads is needed. To arrange the TDMA schedule each head knows the number of its neighbors, each head randomly choose a number in the range  $[1, \text{number of neighbors}+1]$ . Each head sends the chosen number to its neighbors. If the chosen number is the same, the head with less number of neighbors will change its schedule. All the nodes will synchronize themselves with the head to which they belong to it.

We observe that most of the MAC protocols aim at minimizing energy consumption. Energy consumption is minimized by minimizing the amount of time in which a node is in the idle listening state.

In the following section, we will describe routing protocols for WSN. A classification of the routing protocols will be presented.

## 2.2 ROUTING PROTOCOLS FOR WSN

WSN has distinguished characteristics over traditional wireless network that makes routing in WSN is very challenging. First; it is not possible to build a global addressing scheme due to the deployment of huge number of sensor nodes, therefore the classical IP-based routing protocols cannot be applied to sensor networks. Second, Most applications of the sensor networks require the data flow from multiple sources to a particular sink. Third, the generated data has significant traffic redundancy in it. Furthermore, sensor nodes have limited power resource and processing capacity. These differences lead to propose many unique routing protocols for WSN. The routing protocols can be classified as data centric, hierarchical, or location based [14]. Data-centric protocols are query-based and depend on naming of desired data. Hierarchical protocols aim at clustering the nodes so that cluster heads can do some aggregation and reduction of data to reduce energy. Location based protocols utilize the position information to relay data to the desired region rather than the whole network.

Flooding is a classical mechanism to relay data in sensor network without using any routing protocol. In flooding, each sensor node receives a data packet; it will broadcast data to all its neighbors [15]. Eventually the data packet will reach its destination. To reduce the data traffic in the network, gossiping is implemented in which a receiving node sends packet to randomly selected neighbors. In flooding and gossiping, a lot of energy is wasted due to unnecessary transmissions. In addition to energy loss, flooding and gossiping have many drawbacks such as implosion where duplicated message sent to the same node, and overlap where many nodes sense the same region and send similar packets to the same neighbors [16].



In the following subsection, we will survey some of the existing protocols for each category.

### 2.2.1 DATA-CENTRIC PROTOCOLS

In data-centric routing protocol, the sink sends queries to specific regions and the sensor nodes located in the selected region will send the corresponding data to the sink [14]. To specify the properties of the requested data, attribute-based naming is usually used. Many data centric routing protocols are proposed such as: SPIN [16], Directed Diffusion [17], Energy-aware routing [18], Rumor routing [19], , COUGAR [20], and ACQUIRE [21].

**Sensor Protocols For Information Via Negotiation (SPIN):** In SPIN protocol, the data is named using high-level descriptor or meta-data [16]. When a node receives data, it will advertise the meta-data to all its neighbors by broadcasting an ADV message. The neighbors who do not have the advertised data, and interested in the data can reply with a REQ message to request it. In SPIN protocol, since a sensor node will request the desired data only, the problems of flooding such as data redundancy and resource blindness will be solved. In addition to reducing the data redundancy in SPIN, the energy consumption is reduced. However, SPIN protocol is not a reliable routing protocol. For example if the nodes that are interested in the data are far away from the source node and the intermediate nodes are not interested in that data, then the intermediate nodes will not request the data advertised by the source node. Therefore, the data will not arrive the

interested node. The SPIN protocol is not suitable for applications that require reliable delivery of data packets such as intrusion detection.

**Directed Diffusion:** In Directed Diffusion, a naming scheme for the data is used; attribute-value pairs for the data are used [17]. The sensor nodes are queried on demand using attribute-value pairs. To create a query, an interest is defined using a list of attribute-value pairs such as name of objects, interval, duration and geographical area. The interest is broadcasted by the sink. Each node receives the interest will cache it along with the reply link to a neighbor from which the interest is received. The reply link which is called a gradient is characterized by data rate, duration and expiration time. To establish the path between the sink and source, each node will compare the attribute of received data with the values in the cached interest. Using the gradients, the receiving node will specify the outgoing link. Path repairs are possible in Directed Diffusion, when a path between a source and sink fails, a new path should be identified. Multiple paths are identified in advances so that when a path fails one of the alternative paths is chosen without any cost of searching for another path. Directed Diffusion has many advantages; since all communication is neighbor-to-neighbor there is no need for addressing mechanism. Using caching will reduce processing delay. Moreover, Direct Diffusion is energy efficient since the transmission is on demand and there is no need for maintaining global network topology. On the other hand, directed diffusion can not be applied to all sensor network-application since it is based on query-driven data delivery model. It can not be used for applications that require continues data delivery such as environmental monitoring. In addition, the data naming scheme used in Directed Diffusion is application dependent, it must be defined in advance.

**Energy Aware Routing:** Shah et al. proposed to use a set of sub-optimal paths occasionally to increase the lifetime of a WSN [17]. The paths are chosen by a means of probability function. The probability function depends on the energy consumption. Instead of using the minimum path energy all the time, one of the multiple paths is used with a certain probability. In the proposed protocol, it is assumed that each node is addressable through a class-based addressing that includes the location and type of the node. The proposed protocol consists of three phases: setup phase, data communication phase, and route maintenance phase.

In setup phase, routes are found and forwarding tables are created. The total energy cost is calculated in each node. The destination node initiates the connection by flooding the network in the direction of the source node. It sets the cost field to 0 before sending the request. Each intermediate node forwards the request only to the neighbors that are closer to the source node than itself and farther away from the destination node. Upon receiving the request, the energy metric for the neighbor that sent the request is computed and is added to the total cost of the path. Paths that have very high cost are discarded and not added to the forwarding table. Each node assigns a probability to each of its neighbors in the forwarding table. The assigned probability is inversely proportional to the cost of the path. Each node will have a number of neighbors through which it can route packets to the destination. Each node will then calculate the average cost of reaching the destination using its neighbors. The average cost is set to the cost field in the request packet and sent to the source node.

In the Data Communication phase, each node sends the data packet to one of its neighbors, which is selected randomly. The probability to select a neighbor equals to the probability of the neighbor in the forwarding table that is assigned at the setup phase.

In the route maintenance phase, localized flooding is performed infrequently to keep all the paths alive.

**Rumor Routing:** Rumor Routing is another variation of the Directed Diffusion [19]. It is based on a query-driven data delivery model. In Rumor Routing, Instead of querying the entire network as in Directed Diffusion, the queries are routed only to the nodes that have observed a particular event. In Rumor Routing protocol, each node maintains a list of neighbors and events table with forwarding information to all the events it knows. When a node senses an event, it adds it to its event table with a distance of zero to the event, and it generates an agent. An agent is a long-lived packet that travels the network in order to propagate information about local events to all the nodes. The agent contains an events table similar to the table in the nodes. Any node may generate a query for an event; if the node has a route to the event, it will forward the query using this route. If it does not, it will forward the query in a random direction. This continues until the query TTL expires, or until the query reaches a node that has observed the target event. If the node that originated the query determines that the query did not reach a destination it can retransmit or flood the query.

**COUGAR:** In COUGAR protocol, the network is viewed as a huge distributed database system [20]. Declarative queries are used to abstract query processing from the network layer functions. A new query layer between the network and application layers is proposed to support this abstraction. Architecture for the database systems is proposed

where nodes select a leader node to perform aggregation and transmit the data to the sink. The sink generates a query plan, which specifies the necessary information about the data flow, and in-network computation for the incoming query and send it to the relevant nodes.

**Active Query Forwarding In Sensor Networks (ACQUIRE)**: In ACQUIRE protocol, the sensor network is viewed as a distributed database that well suited for complex queries that consist of several sub queries [21]. The sink broadcast the query to the network. Each node receiving the query tries to respond partially by using its pre-cached information and forward it to another sensor. If the information stored in the cache is not up-to-date, the node gathers information from its neighbors within a look-ahead of  $d$  hops. Once the query is resolved completely, it is sent back either through the reverse path or through shortest path to the sink.

**O(1)-Reception Routing Protocol**: Bachir et al. proposes a technique that enables the best route selection based on exactly one message reception. It is called O(1)-reception [22]. In O(1)-reception, each node delays forwarding of routing messages (RREQs) for an interval inversely proportional to its residual energy. This energy-delay mapping technique makes it possible to enhance an existing min-delay routing protocol into an energy-aware routing that maximizes the lifetime of sensor networks. They also identify comparative elements that help to perform a thorough posteriori comparison of the mapping functions in terms of the route selection precision. The O(1)-reception routing enhances the basic diffusion routing scheme by delaying the interests forwarding for an interval inversely proportional to the residual energy: nodes compute a forwarding delay based on their residual energy and defer the forwarding of interest messages for this

period of time. As maximum lifetime routing should combine the min and the max–min metrics, in the energy-delay mapping function, nodes with high residual-energy forward interests without delay to make diffusion equivalent to the min energy routing, and nodes with low residual-energy delay forwarding of interests for a time interval to make diffusion equivalent to the max–min residual energy routing.

**Energy-Balancing Multipath Routing (EMPR):** The basic idea of EMBR is that the base station finds multipath to the source of the data and selects one of them for data transmission [23]. The base station dynamically updates the available energy of each node along the path based on the amount of packets being sent and received. The base station then uses the updated energy condition to periodically select a new path from multiple paths. The base station takes the role of the server and all sensor nodes work as clients. Base station does every thing from querying specific sensing data, broadcasting control packets, routing path selection and maintenance to work as the interface to the outside networks. Sensor nodes are only responsible for sensing data and forwarding packets to the base station. Topology construction is initiated by the base station at any time. The base station broadcasts Neighbor Discovery (ND) packet to the whole network. Upon receiving this packet, every node records the address of the last hop from which it receives and stores it in the neighbors list in ascending order of receiving time. The node changes the source address of the packet to itself. Then it broadcasts the packet. If the new packet is already received the node drops the ND packet and does not rebroadcast. After the completion of Neighbors discovery, the base station broadcasts Neighbors Collection (NC) packet to collect information of each node's neighbors. Upon receiving the NC packet, the node replies a NCR (Neighbors Collection Reply) packet by flooding.

The base station now has a vision of the topology of the networks through the neighbor's information of all nodes. After the topology construction, the base station constructs a weighted directed graph. The weight of each edge is the available energy of the head node. In the data transmission phase, the base station broadcasts enquiry (DE) for sensing data with specific features. Then the sensor nodes satisfying an enquiry will reply with Data Enquiry Reply (DER) packet. On the other hand, the sensor node does not satisfy the enquiry will rebroadcast DE. The base station calculates the shortest path to the desired node in the weighted node.

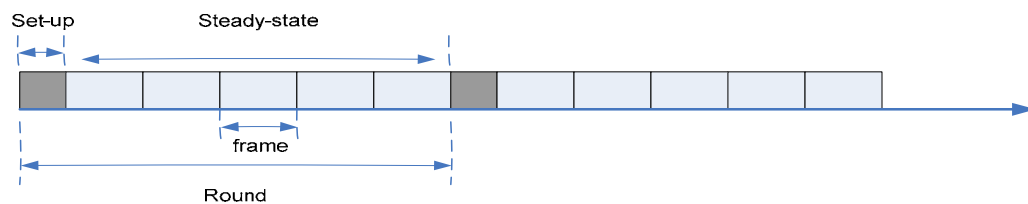
### 2.2.2 HIERARCHICAL PROTOCOLS

In hierarchical routing protocols, clusters are formed. For each cluster, a head node is assigned dynamically, a set of nodes will attach the head node, and the head nodes can communicate with the sink either directly or through upper level of heads. Data aggregation is usually performed at each head. Many hierarchical routing protocols are proposed such as LEACH [3], EAD [4], An Adaptive Low Power Reservation Based Mac Protocol (ALPR) [25], TinyDB [26], Hierarchical-PEGASIS [27], TEEN [28], and APTEEN [29].

**Low-Energy Adaptive Clustering Hierarchy (LEACH):** Heinzelman et al propose a Low-Energy Adaptive Clustering Hierarchy protocol (LEACH). LEACH is application-specific protocol architecture for wireless micro sensor network [3]. In LEACH protocol the nodes organize themselves into clusters. In designing the LEACH protocol, it is assumed that all the nodes in the network can transmit with enough power

to reach the base station (BS) of the network and each node has sufficient computational power to support different MAC protocols and perform signal processing functions. Regarding the network model it is assumed that the network consists of nodes that always have data to send to the end user and the nodes which are located close to each other have correlated data.

In LEACH protocol, the nodes organize themselves into local clusters. One of the nodes is identified as a cluster head and all other nodes in the cluster send their data to the cluster head. The cluster head is responsible for processing the data received from the nodes and transmit the resulted data to the base station. Since the cluster head performs data processing and transmission, it will consume more power than normal nodes. The cluster head must be changed through the system life time. Each node must take its turn to act as a cluster head. Operation of LEACH protocol is divided into rounds. Each round begins with a set-up phase followed by a steady-state phase as shown in Figure 2-9. In set-up phase the clusters are formed and the cluster head is elected. In the steady state phase the nodes will transmit their data.



**Figure 2-9: Time line showing LEACH operation**

The algorithm to select a cluster head is a distributed algorithm. Each node makes autonomous decision to be a cluster head. During each round, there are  $k$  clusters so



there must be  $k$  heads. At round  $r+1$  which starts at time  $t$ , each node selects itself to be a cluster head with probability  $P_i(t)$ .  $P_i(t)$  is chosen such that the expected value of the cluster head must be  $k$ . To ensure that all nodes will act as cluster head equal number of times, each node must be a cluster head once in  $N/k$  rounds. If  $C_i(t)$  is an indicator function that determines whether a node  $i$  has been a cluster head in the most recent  $(r \bmod \frac{N}{k})$  rounds, then the probability that the node is a cluster head will be:

$$p_i(t) = \begin{cases} \frac{k}{N - k(r \bmod \frac{N}{k})} & C_i(t) = 1 \\ 0 & C_i(t) = 0 \end{cases} \quad (5)$$

$C_i(t)=1$  indicates that a node  $i$  has not been a cluster head in the most recent  $(r \bmod \frac{N}{k})$  rounds, and  $C_i(t)=0$  indicates that a node  $i$  has been a cluster head in the most recent  $(r \bmod \frac{N}{k})$  rounds. The probability given in equation 5 is a good estimation for the power. All nodes that have not been assigned as a cluster head in the last  $(r \bmod \frac{N}{k})$  rounds ( $C_i(t)=1$ ) will have more energy than the other nodes and so it is more likely to be selected as cluster heads. In [3] a new probability is proposed to take into account the energy in each node

$$p_i(t) = \min \left\{ \frac{E_i(t)}{E_{total}(t)}, 1 \right\} \quad (6)$$

Where  $E_i(t)$  is the current energy of node  $i$  and  $E_{total}(t)$  is the summation of the current energy at each node. To calculate the probability using equation 6 each node must know the power of all other nodes. All nodes must broadcast its energy level to all other

nodes this can be performed directly for the neighboring nodes and using routing protocol for the non reachable nodes. Broadcasting the energy information will consume additional energy.

$N$  and  $k$  are parameters that are programmed into the nodes. However,  $k$  is a function of the number of nodes  $N$  distributed throughout an a region of dimension  $M$  by  $M$ .  $k$  can be calculated by a distributed algorithm, each node sends a hello message to all neighbors within a predetermined number of hops. Each node counts the number of hello messages received ( $N$ ) then  $k$  calculated based on  $N$ . this algorithm will cost additional energy but it is useful for networks with changing topology.

After identifying the clusters heads, each node must determine the cluster to which it belongs. Each cluster head broadcasts advertisement message containing the head's id using non-persistent CSMA scheme. Each node determines its cluster by selecting the head whose advertise signal is the strongest signal. This head is the most closest head to the node. The node will transmit a joint request message to the chosen cluster head using CSMA. Upon receiving all the joint request messages the cluster head sets up the TDMA schedule and transmit this schedule to the nodes in the cluster. Each node will turn OFF its radio all the time slots except their assigned slots. This will end up the set-up phase and start the steady state phase.

The steady state phase is divided into frames; each node sends its data to the cluster head once per frame during its assigned slot. All nodes must be synchronized and start their set-up phase at the same time. This can be done by transmitting a synchronization pulse by the base station to all nodes. To reduce energy dissipation each non head node use power control to set the least amount of energy in the transmitted

signal to the base station based on the received strength of the cluster head advertisement. When a cluster head receives the data from all nodes, it performs data aggregation and the resultant data will be sent to the base station. Processing the data locally within the cluster reduces the data to be sent to the base station; therefore the consumed energy will be reduced. This is an advantage of the LEACH protocol. To reduce inter-cluster interference, each cluster communicates using direct sequence spread spectrum DSSS. Each cluster uses a unique spreading code.

The distributed cluster formulation algorithm does not offer guarantee about placement and number of cluster head nodes. An alternative algorithm is a central cluster formation; base station (BS) cluster formation. The central cluster formation produces better clusters by dispersing the cluster head nodes throughout the network. In the central algorithm, each node sends information about its current location and its energy level to the BS. The BS computes the average energy level. Any node has energy level less than the average cannot be a cluster head, other nodes can be cluster heads. The BS uses simulated annealing to find the cluster heads. The solution must minimize the amount of energy for non-cluster head and find  $k$  the optimal number of clusters  $k_{opt}$ . When the cluster heads and associated clusters are found the BS broadcasts a message that contains the cluster head ID for each node.

Heinzelman *et al* propose a formula to find the optimum number of clusters that minimize the total consumed energy [3].

$$k_{opt} = \frac{\sqrt{N}}{\sqrt{2\pi}} \sqrt{\frac{\epsilon_{fs}}{\epsilon_{mp}}} \frac{M}{d_{toBS}^2} \quad (7)$$

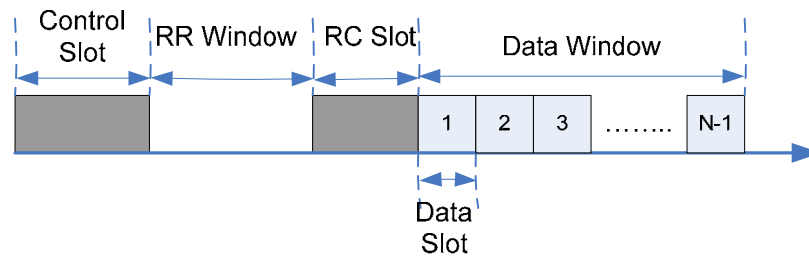
Where  $\varepsilon_{fs}$  and  $\varepsilon_{mp}$  are amplifying energy parameters which depend on the distance. The frame size in LEACH is fixed regardless of the active nodes in the cluster since it is assumed that all nodes have data to send. This is not the real case all the time, sometimes some of the nodes are active and other nodes are inactive.

### **An Adaptive Low Power Reservation Based Mac Protocol (ALPR MAC):**

Although ALRP protocol [25] is named as MAC protocol, we consider it as a routing protocol since the routes for the data to reach the sink is identified in it. ALRP MAC likes the LEACH algorithm in which both are based on cluster-hierarchical network organization and the communication in each cluster is based on TDMA-like frame structure. The difference between two protocols is that ALPR adapts the TDMA frame size based on active nodes to maintain high channel utilization [25].

- **Cluster formation and cluster head identification:** to form the cluster and to identify the cluster head, each node upon power ON waits for a random period before broadcasting a claim to become a cluster-head. The first node capture the medium will be the cluster head in the neighbor. All nodes that hear the broadcast before they transmit their claim will accept the first node as a cluster head. If a node receives more than one claim, it will select the node whose signal is the strongest. The job of cluster head in ALRP is similar to the job of cluster head in the LEACH protocol; maintaining the TDMA schedule and maintaining the synchronizations among all nodes. We note that the probability to be a cluster head depends only on the probability to capture the medium, which is unfair. A node may be a cluster head twice while another node is not identified as cluster head. This can be considered as a disadvantage of this protocol.

- **Channel structure:** the shared channel is divided into superframes. The boundaries between superframes are maintained by beacon signals that are transmitted by the cluster head. Each superframe is divided into four parts as shown in Figure 2-10



**Figure 2-10: Superframe Structure in ALPR**

- **A short control slot:** it is used by the cluster head to broadcast the control information such as the length of next superframe and the request for a new cluster-head.
- **Reservation Request (RR) window:** an unslotted contention based window. In this window, all the nodes that have data to transmit will send reservation request (RR) packet to the cluster head. All the nodes will contend the medium to send their RR packets. The RR packet contains the identities of the source and the intended receiver. To avoid collision, a non-persistent CSMA scheme is used. Due to the constraint in the RR window size the backoff time must be in the interval  $[0, RRwindowtime]$ . If a node fails to send RR packet in the current RR window, it will transmit it in the next window. If the cluster head successfully receives the RR packet, it will reserve a data slot for the source to transmit a data

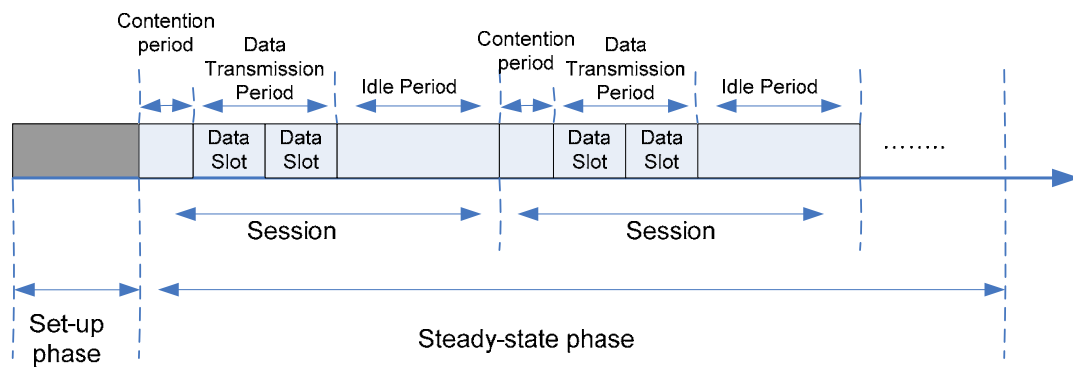
packet in data window. During RR window the cluster head and the nodes which want to transmit RR packet must be awake up

- **Short Reservation Confirmation (RC) window:** the cluster head will send the Reservation Confirmation (*RC*) packet that contains data transmission schedule of all nodes whose *RR* packets were successfully received during RR window. All nodes will be awake up during this window.
- **A slotted data window:** all nodes will be awake up in their assigned slots. The receiving nodes will be awake up also during their assigned slot. Other nodes will not be awake up; they will be in sleeping mode.

The superframe size is adapted based on the traffic intensity. The traffic intensity increases due to the addition of new nodes to the network or due to the increasing in the activity of a specific node. Increasing the traffic intensity will increase the number of RR packets, which increase the number of failure transmissions of RR packets. The number of failure transmissions of RR packets can be used as approximation for the traffic intensity. If the number of failure transmissions of RR packets increases to be larger than a threshold value, the superframe size must be increased. Otherwise, the superframe size must be decreased. To calculate the number of failure transmissions of RR packets each node counts the number of failure in transmissions of its RR packet. This count is sent with the RR packet that is successfully transmitted at the end. The cluster head calculates the average number of failure transmission of RR packet and decide whether to increase or to decrease the superframe size. However, this approach is not accurate. It is possible that a node fails to transmit RR packets many times and cannot successfully transmit any

RR packet. These numbers of failure transmissions will not be considered in calculating the average number of failure transmissions.

Other disadvantages may rise. Firstly, the superframe size is too long. If a node fails to transmit an RR packet in the current RR window, it has to wait for the next RR window, which increases the delay. Secondly, although the number of slots in TDMA schedule will be equal to the number of active nodes only, TDMA schedule will be shorter but there will be additional slots in RR window. It is possible for a node to try to send RR packet many times which will consume more energy.



**Figure 2-11: A single round of the BMA protocol**

**A Bit-Map-Assisted (BMA) MAC:** Another extension of the LEACH protocol is proposed by Li and Lazarou [30]. They proposed a bit-map-assisted (BMA) MAC protocol for large-scale cluster based WSNs. BMA is intended for event-driven applications where the sensor nodes transmit data to the cluster only if significant events are observed. The main idea is to reduce energy wastes due to idle listening and collision while keeping good low latency. The operation of BMA is divided into rounds as in

LEACH. Each round consists of cluster set-up phase and steady-state phase as shown in Figure 2-11. The set-up phase is identical to the setup phase in the LEACH protocol, where clusters heads are identified and clusters are formed. The steady state phase is divided into sessions with fixed durations. Each session consists of contention period, data transmission period and idle period.

For  $N$  nodes in the cluster, the contention period consists of  $N$  slots, and the transmission period is variable and less than  $N$ . the data transmission period plus the idle period is fixed. In the contention period all nodes keeps their radio ON. Each node is assigned a specific time slot and transmits 1-bit control message if it has data to send (source node), otherwise the scheduled slot remains empty. The cluster head knows the nodes that want to transmit. It will construct the transmission schedule and broadcasts it to the source nodes. In data transmission period, each source node turns its radio ON and sends its data to the cluster head during its allocated time slot. The node keeps its radio OFF during all over the remaining time. All non-source nodes turn their radios OFF all over the time. Li and Lazarou introduced an analytical model for the average system energy consumed during each round and an analytical model for the average time required for a packet to be transmitted by a source node and received by the cluster-head for TDMA, E-TDMA, and BMA. The results show that BMA is superior for the cases of low and medium traffic load, relatively few sensor nodes per cluster, and relatively large data packet sizes.

**Energy-Aware Data-Centric Routing Algorithm (EAD):** Boukerche *et al* proposes an Energy-Aware Data-Centric Routing Algorithm (EAD) [4]. EAD protocol is designed for event driven application. In EAD protocol, a tree rooted at the base station



is constructed. The tree consists of leaf and non-leaf nodes. A non-leaf node is a node that has at least one child. On the other hand, a leaf node is a node that has no child. All the leaf nodes of the tree will turn their radio OFF most of the time. On the other hand, all the non-leaf nodes will turn their radio ON all the time. When an event occurs, the leaf nodes will collect the related data and turn its radio ON to transmit the data to its parent. When a non-leaf node receives data from all its children, it will aggregate the data and send it to its parent. All the nodes use CSMA/CA for transmitting the data. Since the radio of the non-leaf sensor nodes will always be ON, they will lose much power than the leaf nodes. The tree will be reconstructed from time to time. Boukerche *et al* proposes an energy aware algorithm to build the tree. One of the disadvantages of EAD protocol is that the non-leaf nodes will be awake all the time even though there are not events to detect. This makes EAD unsuitable for applications with periodic data traffic.

To build a tree rooted at the sink, the sink initiates the process of building the tree. Building the tree is performed by broadcasting control messages. Each control message consists of four fields: *type*, *level*, *parent*, *power*. For the sender node  $v$ ,  $type_v$  represents its status; 0: undefined; 1: leaf node; 2: non-leaf node.  $level_v$  refers to the number of hops from  $v$  to the sink.  $parent_v$  is the next hop of  $v$  in the path to the sink;  $power_v$  is the residual power  $E_v$ . Initially each node has status 0. The sink broadcasts  $msg(2,0, NULL, \infty)$ . When a node  $v$  receives  $msg(2, level_u, parent_u, E_u)$  from node  $u$ , it becomes a leaf node, sense the channel until it is idle, then waits for  $T_2^v$  time, if the channel is still idle,  $v$  broadcasts  $msg(1, level_u + 1, u, E_v)$ . If  $v$  receives  $msg(1, level_u, parent_u, E_u)$  from  $u$ , it senses the channel until it is idle, waits for  $T_1^v$  if the channel is still idle,  $v$  broadcasts  $msg(2, level_u + 1, u, E_v)$ . And it becomes non-leaf node. If node

$v$  receives more than one message from different nodes before it broadcasts its message, it will select the node with larger energy as its parent. If both nodes have the same energy, it will select one of them randomly. The waiting node will go back to sensing state, if another node occupies the common channel before it times out. If a node  $v$  with status 1 receives  $msg(2, level_w, v, E_w)$  from node  $w$  indicating that  $v$  is its parent,  $v$  broadcasts  $msg(2, level_v, parent_v, E_v)$  immediately after the channel is idle. The process will continue until each node becomes leaf or non-leaf node. A sensor with status 2 becomes a leaf node if it detects that it has no children. Both  $T_1^v$  and  $T_2^v$  are chosen such that no two neighboring broadcasts are scheduled at the same time. On the other hand, to force the neighboring sensors with higher energy to broadcast earlier than those nodes with a lower residual power, both  $T_1^v$  and  $T_2^v$  must be monotonically decreasing functions of  $E_v$ . [4] chooses  $T_1^v = 2 * t_0 + \frac{c}{E_v}$  and  $T_2^v = t_0 + \frac{c}{E_v}$  where  $t_0$  is the upper bound of the propagation time between any pair of neighboring sensors and  $c > 0$  is an adjusting constant.

**TinyDB:** Another alternative in the same direction is the work presented in [26]. A distributed query processor for smart sensor devices (TinyDB) is proposed. In TinyDB, to disseminate queries and collecting results, a routing tree rooted at the base station is built. The routing tree is formed by forwarding a routing request (a query in TinyDB) from every node in the network. The root broadcasts a request, and then all children that hear this request will process it, and then it forwards the request to their children, and so on, until the entire network has heard the request. Each node picks a parent node that is one level closer to the root. This parent will be responsible for

forwarding the node's query results to the base station. To limit the scope of queries, a Semantic Rooting Tree (SRT) is built. This tree is built based on the routing tree. If a node knows that none of its children currently satisfies the query, it will not forward the query down the routing tree. Therefore, each node must have information about child attribute values.

#### **Power-Efficient Gathering In Sensor Information System (PEGASIS):**

PEGASIS protocol is an improvement of the LEACH protocol, instead of forming clusters, PEGASIS forms chains from sensor nodes [33]. Each node will transmit and receive from a neighbor. One node of the chain will transmit directly to the sink. Data will move from node to node until it reaches the sink. At each intermediate node, data aggregation is performed. The chain is constructed in a greedy way.

Hierarchical-PEGASIS [27] is an extension to PEGASIS. It is proposed to decrease the delay of packets delivered to the base station. It also proposes a solution for the data gathering problem by considering the (*energy by delay*) metric. To reduce delay, simultaneous transmissions of data packets are performed. Two approaches are used; CDMA or allowing the spatially separated nodes to transmit at the same time. In CDMA approach, the chain is constructed as a tree. Each node in a particular level will transmit data to the node in the upper level of the tree. The parallel data transmission will reduce the packet delay significantly. In the second approach, a three-level tree is constructed. Then simultaneous transmissions are scheduled carefully to reduce the interference effects.

#### **Threshold Sensitive Energy Efficient Sensor Network Protocol (TEEN):**

TEEN can be considered as data centric and hierarchical protocol [28]. It is proposed for

event driven applications such as sudden changes in the temperature. In TEEN protocol, clusters are formed by the base station. Cluster head can communicate with the sink directly or through another cluster head. After cluster formulation, the cluster head broadcasts two thresholds to the nodes, hard and soft threshold for the sensed attribute. The hard threshold is the minimum value of the attribute at which the sensor node will turn its transmitter ON and it will transmit the corresponding data to the cluster head. The node will transmit only when the sensed attribute is in the range of interest, which reduces the number of transmissions. Furthermore, when the sensed value is near the hard threshold, the sensor node will transmit data only if the attribute value change by an amount equal to or greater than the soft threshold, which will further reduce the number of the transmission. If the attribute value does not reach the hard threshold then the node will not transmit at all, therefore the TEEN is not suitable for applications with periodic data traffic. For this kind of applications, Adaptive TEEN (APTEEN) is proposed [29]. In APTEEN, after clusters formulation, the cluster head will broadcast the attributes, the threshold values and the transmission schedule to the all nodes.

**Unequal Cluster Based Routing (UCR):** In UCR protocol, clusters with different size are constructed [34]. Cluster heads closer to the sink will have smaller cluster sizes than those farther from the sink. Thus they can preserve some energy for the inter-cluster data forwarding. A greedy geographic and energy-aware routing protocol is designed for the inter cluster communication which considers the tradeoff between the energy cost of relay paths and the residual energy of relay nodes. The UCR protocol consists of two parts: an Energy-Efficient Unequal Clustering algorithm called EEUC and an intercluster greedy geographic and energy-aware routing protocol. Initially, the

base station broadcasts a beacon signal to all sensors at a fixed power level. Based on the received signal strength, each sensor node can compute the approximate distance to the base station. It not only helps nodes to select the proper power level to communicate with the base station, but also helps us to produce clusters of unequal sizes. In EEUC algorithm, heads will be identified randomly. As in LEACH protocol, the task of being a cluster head is rotated among sensors in each round to distribute the energy consumption across the network. After cluster heads have been selected, each cluster head broadcasts a CH\_ADV\_MSG across the network field. Each ordinary node chooses its closest cluster head, the head with the largest received signal strength, and then informs it by sending a JOIN\_CLUSTER\_MSG. After forming clusters, data will be transmitted from the cluster heads to the base station. Each cluster head first aggregates the data from its cluster members, and then sends the packet to the base station via a multi-hop path through other intermediate cluster heads. Before selecting the next hop node, each cluster head broadcasts a short beacon message across the network at a fixed power which consists of its node ID, residual energy, and distance to the base station. A threshold TD\_MAX in the multi-hop routing protocol is proposed. If a node's distance to the base station is smaller than TD\_MAX, it transmits its data to the base station directly; otherwise, it is better to find a relay node that can forward its data to the base station.

**Self-Organizing Protocol:** Subramanian et al proposed architectural and infrastructural components to build sensor applications [35]. In the proposed architecture, the sensor nodes can be either stationary or mobile node, they sense the environment, and they forward data to a set of nodes that act as routers. Routers are stationary nodes, and they form the backbone of the network. To be a part of network, a node must be able to

reach the router. A routing architecture that requires addressing of sensor node has been proposed. Sensors are identified through the address of the router node that it is connected to. The protocol for self-organizing the router nodes and creating the routing tables consists of four phases; discovery, Organization, Maintenance, and Self-reorganizing. In the discovery phase, the nodes in the neighborhood of each sensor are discovered. In the Organization phase, groups are formed and merged to form a hierarchy. Each node assigned an address based on its position in the hierarchy. Routing tables and energy levels of nodes are updated in the Maintenance phase. In the Self-Reorganizing phase, group reorganization is performed. The proposed protocol utilizes the router nodes to keep all the sensors connected by forming a dominating set.

**Energy-Aware Routing For Cluster-Based Sensor Networks:** Younis et al. proposed a hierarchical routing algorithm based on a three-tier architecture [36]. In the proposed protocol, sensors are grouped into clusters. The cluster heads (gateways) are less energy constrained than normal sensors. It is assumed that cluster heads knows the location of the sensor nodes. Gateways maintain the states of the sensors and sets up multi-hop routes for collecting sensors data. Each gateway informs each node within its clusters the time slots in which it can transmit and in which it have to listen to other nodes transmission. The sensor nodes in the cluster can be in one of four states: sensing only, relaying only, sensing-relaying and inactive. In sensing state, the sensor node senses the environment and generates the corresponding data. In the relaying only state, the node does not sense the environment but it forwards data from other active nodes. In sensing-relaying state, the node not only senses the environment but also forwards the data from other active nodes. In inactive state, the node neither senses the environment nor

forwards data. The link cost is defined as the energy consumption to transmit data between two nodes, the delay optimization and the other performance cost. A least-cost path is found between sensor nodes and the gateway. The gateway monitors the available energy level at every sensor that is active. Rerouting is triggered by an application-related event requiring different set of sensors to probe the environment or the depletion of the battery of an active node.

**Base-Station Controlled Dynamic Clustering Protocol (BCDCP):**

Muruganathan et al. proposes a clustering-based routing protocol called Base Station Controlled Dynamic Clustering protocol (BCDCP) [37]. In BCDCP, the base station sets up clusters and routing paths, performs randomized rotation of cluster heads, and carries other energy intensive tasks. The key ideas in BCDCP are: formulation of balanced clusters where each cluster head serves an approximately equal number of member nodes, uniform placement of cluster heads throughout the entire sensor field, and the utilization of cluster-head-to-cluster-head(CH-to-CH) routing to transfer the data to the base station. Class-based addressing of the form <Location ID, Node Type ID> is used in BCDCP. The Location ID identifies the location of a node. It is assumed that the base station keeps up-to-date information on the location of all the nodes in the network. A Node Type ID describes the functionality of the sensor such as seismic sensing, and thermal sensing. BCDCP operates in two major phases: setup and data communication. In setup phase, clusters are formed, clusters' heads are selected, CH-to-CH routing paths are formed, and schedule is created for each cluster. During each setup phase, the base station receives information on the current energy status from all the nodes in the network. Based on this information, the base station computes the average energy level

and then chooses a set of nodes, denoted  $S$ , whose energy levels are above the average value. Cluster heads for the current round will be chosen from the set  $S$ . To identify the cluster heads from the set and to form clusters, iterative cluster splitting algorithm is used. This simple algorithm first splits the network into two sub-clusters, and proceeds further by splitting the sub-clusters into smaller clusters. The base station repeats the cluster splitting process until the desired number of clusters is attained. Once the clusters and the cluster head nodes have been identified, the base station chooses the lowest-energy routing path and forwards this information to the sensor nodes along with the details on cluster groupings and selected cluster heads. The routing paths are selected by connecting all the cluster head nodes using the minimum spanning tree approach that minimizes the energy consumption and then a head is randomly selected to transmit data to the base station. The last step in this phase is building a TDMA Schedule for each cluster. In the data communication phase, Data gathering, Data fusion, and Data routing are performed using the TDMA schedule created in setup phase.

### 2.2.3 LOCATION-BASED PROTOCOLS:

Information Location can be utilized to forward data with minimum energy consumption. If the region to be monitored is known, the query can be forwarded to that region. Many location-based routing protocols for WSN were proposed. In the successive subsections, we will survey many of these protocols.

**Geographic Adaptive Fidelity (GAF):** GAF is energy-aware location-based routing protocol designed for mobile ad hoc protocols, but it can be applicable to sensor



networks [38]. In GAF a virtual grid for the monitored area is formed. Each node uses its GPS-indicated location to associate itself with a point in the virtual grid. Nodes associated with same point in the grid are equivalent. Some of them can be in the sleeping state to save energy while others will be in active state. Therefore, the network lifetime will increase. To balance load among nodes, equivalent nodes change their state from active to sleeping in turn. Three states are defined in GAF, discovery, sleep, and active. In the discovery state a node will determine its neighbors. While it is in sleep state, a node will turn OFF its radio. The active node will participate in data routing. A node will be in each state for a particular time period which is application dependent. On the other hand, determining which nodes that will be in sleep state is application dependent. GAF is implemented for non-mobility (GAF-basic) and mobility (GAF-mobility adaptation) of nodes. To keep the network connected, a representative node must be always active for each region on its virtual grid.

**Minimum Energy Communication Network (MECN):** In MECN protocol, low power GPS is utilized to find a minimum power topology for stationary nodes including the sink [39]. For each node, a relay region is identified. The relay region consists of the neighboring nodes where transmitting through those nodes is more energy efficient than direct transmission. The enclosure of a node  $i$  is the union of all relay regions that node  $i$  can reach. The protocol has two phases; in the first phase, the enclosure graph is constructed. The enclosure graph consists of all enclosures of each transmit node, and it contains globally minimum energy links which will be found in the second phase.

**Geographic And Energy Aware Routing (GEAR) :** In GEAR protocol, energy aware and geographical-informed neighbor selection heuristic is used to route packets

towards the destination region [40]. The key idea is to restrict the number of interests in Directed Diffusion to certain regions rather than sending interest to the whole network. Each node keeps an estimated cost and a learned cost of reaching the destination through its neighbors. The estimated cost is a combination of residual energy and distance to destination. The learned cost is a refinement of the estimated cost. A hole exists in the network when a node does not have any closer neighbor to the target region. With no holes in the network, the estimated cost is equal to the learned cost. When a packet reaches the destination, the learned cost is propagated one hop back so that route setup for next packet will be adjusted. The GEAR protocol consists of two phases; in the first phase, the packets are forwarded towards the target region, when a node receives a packet, it checks its neighbors to see if there is a neighbor that is closer to the target region. The closest neighbor to the target region is selected as the next hop. When all neighbors are further than node itself, a hole exists, one of them will be selected based on the learned cost function. This selection will be updated according to the convergence of the learned cost. In the second phase, packets will be forwarded within the region; the packets are forwarded in the region by either recursive geographic forwarding or restricted flooding.

**The Greedy Other Adaptive Face Routing (GOAFR):** GOAFR is a geometric ad-hoc routing algorithm combining greedy and face routing. The greedy algorithm of GOAFR always picks the closest neighbor to destination to be the next hop [41]. However, it can stuck at some local minimum, no neighbors closer than the current node. Other face routing is a variant of Face Routing (FR) [42]. Other Face Routing utilizes the face structure of planer graphs such that the message is routed from node to node by

traversing a series of face boundaries. The aim is to find the best node on the boundary; the closest node to the destination. It was shown that GOAFR algorithm can achieve both worst-case optimality and average-case efficiency.

**SPAN:** In SPAN protocol, some nodes are selected as coordinators based on their positions [43]. The coordinators form a network backbone that is used to forward messages. A node should become a coordinator if two neighbors of a non-coordinator node can not reach each other directly or via one or two coordinators.

### **A Mesh-Based Routing Protocol For Wireless Ad-Hoc Sensor Network**

**(MBR):** In MBR protocol, the area of the sensor network is portioned into regions; mesh topology [44]. The nodes can communicate to their neighbor nodes through virtual channels. Forming the mesh topology is performed in three phases. In the first phase, the base node for zoning is selected. Two setup sensors are determined. One of them is located at the largest diameter and in the boundary of the area and the second sensor is located on the boundary of other orthogonal diameter of the region. In phase two, the network is divided into regions. In phase three, each sensor nodes is assigned ID. Each sensor will be known with two features: its region coordinate (X,Y) and its ID. To transmit data between source nodes and sink a path is reserved between them firstly. To reserve a path, the source node sends a reserve message, called RAP, to the sensors in its target (X,Y). Upon receiving the RAP message, each node generates a priority number and returns it to the source node using ACK message. Sensors have higher energy will have higher priority. The source sensor will select sensors to form the path among the sensors that sends ACK message. Then data will be sent based on the path determined. After transmitting data, path must be released. This is done by sending a CRP message.

**Energy-Efficient Geographic Multicast Routing:** Sanchez et al. proposes a novel energy-efficient multicast routing protocol called GMREE [45]. It aims to preserve energy and network bandwidth. GMREE protocol builds multicast trees based on a greedy algorithm using local information. GMREE protocol is based in the concept of cost over progress metric and it is specially designed to minimize the total energy used by the multicast tree. The cost is defined as the energy needed to reach the furthest neighbor in the selected set of relays plus the energy that such amount of nodes will need to process the message. GMREE incorporates a relay selection function which selects nodes from a node's neighborhood taking into account not only the minimization of the energy but also the number of relays selected. Nodes only select relays based on a locally built and energy-efficient underlying graph reduction such as Gabriel graph, enclosure graph or a local shortest path tree. Thus, the topology of the resulting multicast trees really takes advantage of the benefit of sending a single message to multiple destinations through the relays which provide best energy paths.

**Energy-Aware Geographic Routing For Sensor Networks With Randomly Shifted Anchors:** Anchor-based geographic routing aims at finding a small number of intermediate nodes acting as anchors so that the path length (i.e. number of hops) between the source and destination can be reduced. However, some nodes (e.g., nodes near the boundary of the network) tend to be used as anchors repeatedly by multiple flows. As a result, their energy drains quickly and the lifetime of the network is reduced. Moreover, the intermediate nodes between source and destination change very little once the anchor list is set. This also contributes to the quick depletion of the energy for some nodes. To overcome these shortcomings, Zhao et al. introduces a random shift to the

location of each anchor in the routing process [46]. Each new packet will then be routed to a different anchor determined by the location of the original anchor plus the random shift. Because the shift is generated randomly, different packets will likely be routed through a different list of anchors. This allows more nodes to be involved in the routing process and the energy consumption is better distributed among nodes in the network.

**Projection Distance-Based Anchor Protocol (PDA):** Zhao et al. proposed a Projection Distance-based Anchor scheme, which is called PDA, to obtain the anchor list based on the projection distance of nodes in detouring mode [47]. The projection is with respect to the virtual line linking the source and destination nodes. To obtain an anchor list adaptively, the first packet of a burst is routed from the source to the destination using an existing geographic routing algorithm. During the routing of the first packet, an anchor list is built. After the first packet is delivered, the anchor list is sent back to the source from the destination, and the list is embedded into subsequently packets. A packet is then routed from the source to the first anchor node, then to the second anchor node, and so on, until it reaches the destination.

**On Optimal Geographic Routing in Wireless Networks with Holes and Non-Uniform Traffic:** Subramanian et al. propose a randomized geographic routing scheme that can achieve a throughput capacity of  $\Theta(1/\sqrt{n})$  (within a poly-logarithmic factor) even in networks with routing holes [48]. They show that the proposed scheme is throughput optimal (up to a poly-logarithmic factor) while preserving the inherent advantages of geographic routing. They also show that the routing delay incurred by the proposed scheme is within a poly-logarithmic factor of the optimal throughput-delay

trade-off curve. On the other hand, Subramanian et al. construct a geographic forwarding based routing scheme that can support wide variations in the traffic requirements as much as  $\Theta(1)$  rates for some nodes, while supporting  $\Theta(1/\sqrt{n})$  for others. They show that the above two schemes can be combined to support non-uniform traffic demands in networks with holes.

The randomized algorithm takes as an input the number of nodes in the network, the packet to be sent, as well as the number of holes. Considering the first packet in all the source nodes, The source node for every traffic flow creates  $R\log(n)$  copies of its packet to send. It chooses  $R\log(n)$  independent and uniformly distributed points from the unit region and sets the NEXT-DEST field in the packet to the randomly generated location in each of these copies. The  $R\log(n)$  packets are routed from the source in a greedy geographic manner to the location in NEXTDEST. Upon receiving a packet, a node checks if it is the NEXTDEST location. If it is not the NEXT-DEST location, it searches within its neighboring nodes for the node that is closest to the NEXT-DEST location, and forwards the packet to that node. If none of its neighbor nodes is closer to the NEXT-DEST than itself, the node drops the packet. If it is the NEXT-DEST location, it checks whether it is the final destination or not. If it is the final destination, then the packet is received. Otherwise, If the final destination is one hop away from the current node, the node forwards the packet greedily to the final destination. If the final destination is more than one hop a way from the current node, the current node makes  $R\log(n)$  copies of the packet and again generates uniform and randomly chosen locations for the NEXT-DEST in each of the packet copies, and forwards them greedily.

### 2.2.4 QoS-AWARE PROTOCOLS

QoS-aware protocols consider end-to-end QoS requirement while setting up the paths in the sensor network. Many QoS-aware routing protocols for WSN were proposed. In the successive subsections, we will survey many of these protocols.

**Maximum Lifetime Energy Routing:** Chang et al presents a routing protocol for sensor networks based on a network flow approach [49]. The protocol aims to maximize the network lifetime by defining link cost as a function of node remaining energy and the required transmission energy using that link. Finding traffic distribution is a possible solution to the routing problem. The solution to this problem maximizes the network lifetime. Two maximum residual energy path algorithms were proposed to find the best link metric for the maximization problem. The two algorithms differ in their definition of link costs and the incorporation of nodes' residual energy. The link costs that are used in the two algorithms are:

$$c_{ij} = \frac{1}{E_i - e_{ij}} \quad \text{and} \quad c_{ij} = \frac{e_{ij}}{E_i}$$

where :  $E_i$  is the residual energy at node  $i$

$e_{ij}$  is the energy consumed when a packet transmitted over link  $i$ - $j$ .

The least cost paths to destination are found using Bellman-Ford shortest path algorithm. The least cost path is the path whose residual energy is largest among all paths.

**Maximum Lifetime Data Gathering:** Kalpakis et al. models the data routes setup in sensor network as the maximum lifetime data-gathering problem [50] . A polynomial time algorithm to solve this problem is proposed. The data-gathering

schedule specifies for each round how to get and route data to sink. For each round, a schedule has one tree rooted at the sink and spans all the nodes of the network. The network lifetime depends on the duration for which the schedule remains valid. The Maximum Lifetime Data Aggregation (MLDA) protocol is proposed to set up maximum lifetime routes taking into account data aggregation. If a schedule "S" with "T" rounds is considered, it induces a flow network  $G$ . The flow network with maximum lifetime subject to the energy constraints of sensor nodes is called an optimal admissible flow network. A schedule will be constructed by using this admissible flow network. For application with no data aggregation such as video sensors, a new scenario is presented, which is called Maximum Lifetime Data Routing (MLDR). It is modeled as a network flow problem with energy constraints on sensors.

**Minimum Cost Forwarding:** The objective of Minimum Cost Forwarding protocol is to find the minimum cost path in a large sensor network [51]. The cost function for the protocols captures the effect of delay, throughput and energy consumption from any node to the sink. The protocol consists of two phases; setup phase and data transmission phase. In setup phase, starting from the sink, the cost value on all nodes is set up. The sink set its cost as zero and broadcast a message for all its neighbors. Upon receiving the message, each neighbor of the sink will set its cost as the cost of the link to sink and it broadcast its cost, and so on. Every node adjusts its cost value by adding the cost of the node it received the message from and the cost of the link. Cost adjustment is done using a back-off based algorithm. The forward of messages is deferred to allow the message with minimum cost to arrive. Therefore, optimal cost for all nodes to the sink is found. In the second phase, the source node broadcasts the data to its



neighbor. Upon receiving the broadcast message, the node adds its transmission cost to sink to the cost of packet, then the node checks the remaining cost in the packet. If it is not sufficient to reach the sink, the packet is dropped. Otherwise, the node forwards the packet to its neighbors.

**Sequential Assignment Routing (SAR):** SAR is a table driven multi-path protocol aiming to achieve energy efficiency and fault tolerant [1]. In the SAR protocol, trees rooted at one-hop neighbors of the sink is created by taking QoS metric, energy resources on each path and priority level of each packet into account. By using created trees, multiple paths from sink to sensors are formed. One of these paths is selected according to energy resources and QoS in the path. Failure recovery is done by enforcing routing table consistency between upstream and downstream nodes on each path. Any local failure causes an automatic path restoration procedure locally.

**Energy-Aware QoS Routing Protocol:** Akkaya and Younis extend the routing approach in [36]. The proposed protocol finds a least cost and energy efficient path that meets certain end-to-end delay [52]. The link cost function captures the nodes' energy transmission energy, error rate and other communication parameters. To support both best effort and real-time traffic at the same time, a class-based queuing model is employed. The proposed protocol finds a list of least cost paths by using an extended version of Dijkstra's algorithm and selects a path from the list that meets the end-to-end delay requirement.

**Bimodal Power-Aware Routing Protocol (BIPAR):** Morcos et al. proposes BImodal Power-Aware Routing Protocol (BIPAR) [53] . BIPAR has two modes of

operation; min-power and max-power routing. Min-power routing is a routing scheme that delivers packets over the minimum-power path from the source to the destination. The other mode is max power routing. Max-power routing uses more power to route packets and it favors paths of physically longer hops to those of shorter hops. The operation of BIPAR has two phases: cost establishment phase and data forwarding phase. In cost Establishment Phase, the routing status in the forwarding sensor nodes is set up. The sink sends Advertisement packet (ADV). The ADV packet is used to assign costs to each node. A node's cost is the least amount of power needed to transmit packets from this node to the sink. The ADV packet has a cost field. When the sink first broadcasts it, the ADV packet has a cost of 0. Upon receiving an ADV packet from node Y, node X sets its own cost as the sum of the cost field in the ADV packet and the amount of power needed to transmit packets from Y to X. Then, X sets the cost of the ADV to its own packet and rebroadcasts the packet. In addition, each node can utilize the ADV packet to build its list of neighbors toward the sink. This list is considered as the routing table of each node. The list of neighbors for node X contains any node that has cost less than node X. In data forwarding phase, sensor nodes sense the environment and send their measured data back to the sink. The source assigns a power budget to each data packet it sends. This budget is the total amount of power to be used to forward this packet from the source to the sink. Along with the budget, the source node sends sender's cost and consumed power so far. Upon receiving any data packet from node Y, node X compares its own cost to the cost of the sender. Node X can only rebroadcast the packet; if its cost is less than that of Y; otherwise X drops the packet. If X decides to rebroadcast the packet, it calculates the power needed to send the packet from Y to itself and update the

consumed power so far field of the packet. The latter is checked against the budget allowed for this packet. If the packet has exceeded its budget, X drops it. X then consults its neighbors' list and picks its closest neighbor to forward this packet to it. X then waits for an acknowledgement (ACK) for a predefined timeout interval. If X gets an ACK for its packet during this timeout interval, then X's job is done concerning this packet. Otherwise, X would consult its neighbors' list again, this time picking its furthest neighbor to forward the same packet to it.

**SPEED:** is a real-time communication protocol for sensor networks [54]. It provides three types of real-time communication services; real-time unicast, real-time area-multicast and real-time area-anycast. End-to-end soft real-time communication is achieved by maintaining a desired delivery speed across the sensor network through a novel combination of feedback control and non-deterministic geographic forwarding. In SPEED protocol, each node should maintain information about its neighbors. Geographic forwarding is used to find the paths. SPEED protocol strives to ensure end-to-end delay for the packets in the network such that each application can estimate the end-to-end delay for the packets. SPEED protocol consists of the following components: Neighbor beacon exchange scheme, Delay estimation scheme, The Stateless Non-deterministic Geographic Forwarding algorithm (SNGF), A Neighborhood Feedback Loop (NFL), Backpressure Rerouting, and Last mile processing. SNGF is the routing module responsible for choosing the next hop candidate that can support the desired delivery speed. NFL and Backpressure Rerouting are two modules to reduce or divert traffic when congestion occurs, so that SNGF has available candidates to choose from. The last mile process is provided to support the three communication semantics mentioned before.

Delay estimation is the mechanism by which a node determines whether or not congestion has occurred. And beacon exchange provides geographic location of the neighbors so that SNGF can do geographic based routing.

In general, WSN applications can be classified into three categories: Query based, Event driven, and periodic. In query based applications, the send forward a query to the nodes. Then, the corresponding nodes will response with the desired data. In event driven applications, the sensor nodes will forward data to sink when a specific event is detected. In the periodic applications, the nodes forward data to sink periodically, every fixed time interval. We classify the routing protocols according to these categories of application. The classification is shown in Table 2-1.

**Table 2-1 : Classification of Routing Protocols based on the Applications**

Protocol	Application		
	Query Based	Event Driven	Periodic
SPIN	√		
Directed Diffusion	√		
Shah et al.			√
Rumor Routing	√		
CADR	√		
COUGAR	√		
ACQUIRE	√		
GBR	√		
O(1)-Reception Routing Protocol		√	
EMPR	√		
LEACH		√	
EAD		√	
TinyDB	√		
PEGASIS		√	
TEEN		√	
APTEEN			√
UCR		√	
BCDCP		√	
GAF		√	√
MECN		√	
GEAR	√		
GOAFR		√	
MBR		√	
GMREE		√	
Zhao et al. Randomly Shifted Anchors:		√	
Chang et al			√
Kalpakis et al.		√	√
Minimum Cost Forwarding		√	
SAR		√	√
Energy-Aware QoS Routing Protocol			√
BIPAR		√	
SPEED	√		

### 2.3 CROSS LAYER DESIGN IN WSN

Many researchers studied the necessity and possibility of taking advantages of cross layer design to improve the power efficiency and system throughput of WSN.

Ahmed Safwat et al proposed Optimal Cross-Layer Designs for Energy-efficient Wireless Ad hoc and Sensor Networks [55]. They proposed Energy-Constrained Path Selection (ECPS) scheme and Energy-Efficient Load Assignment (E2LA). ECPS utilizes cross-layer interactions between the network layer and MAC sublayer. The main objective of the ECPS is to maximize the probability of sending a packet to its destination in at most  $n$  transmissions. To achieve this objective, ECPS employs probabilistic dynamic programming (PDP) techniques assigning a unit reward if the favorable event (reaching the destination in  $n$  or less transmissions) occurs, and assigns no reward otherwise. Maximizing the expected reward is equivalent to maximizing the probability that the packet reaches the destination in at most  $n$  transmissions. It is found that the probability of success at an intermediate node  $i$  right before the  $t^{th}$  transmission  $f_t(i)$  to be :

$$f_t(i) = \begin{cases} 1 & i = D \\ \max_j \sum_k p_k f_{t+k}(j) & otherwise \end{cases} \quad (10)$$

Where

D : Destination node

j : The next hop towards the destination D

Any energy-aware route that contains D and the distance between D and the source node is less or equal to  $n$  can be used as input to ECPS. The MAC sub-layer provides the network layer with the information pertaining to successfully receiving CTS or an ACK

frame, or failure to receive one. Then ECPS chooses the route that will minimize the probability of error

The objective of the E2LA scheme is to distribute the routing load among a set  $Z$  of Energy-aware routes. Packets are allotted to routes based on their willing to save energy. Similar to ECPS, E2LA employs probabilistic dynamic programming techniques and utilize cross-layer interactions between the network and MAC layers. At the MAC layer, each node computes the probability of successfully transmitting packets in  $\alpha$  attempt. E2LA assign loads according to four distinct reward schemes [55].

Parvathinathan Venkitasubrananiam et al propose a novel distribution medium access control scheme called opportunistic ALOHA (O-ALOHA) for reachback in sensor network with mobile agent [56]. The proposed scheme based on the principle of cross layer design that integrates physical layer characteristics with medium access control. In the O-ALOHA scheme, each sensor node transmits its information with a probability that is a function of its channel state (propagation channel gain). This function called transmission control is then designed assuming that orthogonal CDMA is employed to transmit information. In designing the O-ALOHA scheme they consider a network with  $n$  sensors communicate with a mobile agent over a common channel. It is assumed that all the sensor nodes have data to transmit when the mobile agent is in the vicinity of the network. Time is slotted into intervals with length equal to the time required to transmit a packet. The network is assumed to operate in time division duplex (TDD) mode. At the beginning of each slot, the collection agent transmits a beacon. The beacon is used by each sensor to estimate the propagation channel gain from the collection agent to itself which is the same as the channel gain from the sensor to the collection agent. It is

assumed that the channel estimation is perfect. The propagation channel gain from sensor  $i$  to the collection agent during slot  $t$  which is denoted as  $\gamma_i^{(t)}$  is modeled as:

$$\gamma_i^{(t)} = \frac{P_T R_u^2}{r_i^2 + d^2} \quad (11)$$

where

$R_u^2$  : is Rayleigh Distribution

$P_T$  : The transmission power of each sensor.

$r_i$  : radial distance of sensor  $i$

$d$  : distance from collecting agent and sensor node.

During the data transmission period, each sensor transmits its information with a probability  $s(\gamma_i^{(t)})$  where  $s(\cdot)$  is a function that maps the channel state to a probability. Two transmission controls are proposed to map from the channel gain to the probability; Location independent transmission control (LIT) and Location aware transmission control (LAT). In LIT, the decision to transmit a packet is made by observing channel state  $\gamma$  alone, while in LAT, every sensor makes an estimate of its radial distance and the decision to transmit is a function of both the channel state  $\gamma$  and the location of sensor.

Mihail L. proposed a deterministic schedule based energy conservation scheme [57]. In the proposed approach, time synchronized sensors form on-off schedules that enable the sensors to be awake only when necessary. The energy conservation is achieved by making the sensor node go to sleeping mode. The proposed approach is suitable for periodic applications only, where data are generated periodically at deterministic time. The proposed approach requires the cooperation of both the routing and MAC layers. The



on-off schedule is built according to the route determined by routing protocol. The proposed approach consists of two phases; the setup and reconfiguration phase and the steady state phase. In the setup and reconfiguration phase, a route is selected from the node originating the flow to the base station then the schedules are setup along the chosen route. In the steady phase, the nodes use the schedule established in the setup and configuration phase to forward the data to the base station. In this phase, there will be three types of actions at each node; Sample action which is taking data sample from environment, Transmit action to transmit data, and Receive action to receive data. The actions at each node along with the time when each action will take place are stored in the schedule table of each node. The node can be awake at the time of each action and go to sleep otherwise.

Li-Chun Wang and Chung-Wei Wang proposed Cross layer Design of Clustering architecture for wireless Sensor Networks. The proposed scheme is called Power On With Elected Rotation (POWER) [58]. The objective of the POWER is to determine the optimal number of clusters from the cross-layer aspects of power saving and coverage performance simultaneously. The basic concept of the POWER is to select a representation sensor node in each cluster to transmit the sensing information in the coverage area of the sensor node. The representative sensor node in a cluster is selected in rotation among all the sensor nodes in each cluster. In the POWER scheme, the scheduling procedure is rotated many rounds. In each round, there are two phases; the construction table phase (CTP), to construct the rotation table and the rotational representative phase (RRP) to transmit data. In CTP, all sensor nodes employ the MAC protocol and the first sensor node accessing the channel become the initiator node, then

the initiator node detects other neighboring node and forms the cluster. RRP starts after constructing the rotation table. RRP is divided into many sRPs (Sub-Rotated Period). In each sRP, one node will be a representative node and all other nodes in the cluster will be in sleeping mode.

Rick W. Ha et al proposes a cross-layer sleep-scheduling-based organization approach, called Sense-Sleep Trees (SS-trees) [59]. The proposed approach aims to harmonize the various engineering issues, and it provides a method of increasing monitoring coverage and operational lifetime of mesh-based WSNs engaged in wide-area surveillance applications. An iterative algorithm is suggested to determine the feasible SS-tree structure. All the SS trees are rooted at the sink. Based on the computed SS-trees, optimal sleep schedules and traffic engineering measures can be devised to balance sensing requirements, network communication constraints, and energy efficiency. For channel access a simple single-channel CSMA MAC with implicit acknowledgements (IACKs) is selected. In SS-trees approach, the WSN's life cycle goes through many stages. After the initial deployment of nodes, the WSN will enter the network initialization stage, in which the sink gathers network connectivity information from sensor nodes, compute the SS-trees, and disseminate the sleep schedules to every sensor node. Then the WSN will enter the operation stage, in which the nodes will alternate between Active and sleep stages. During long periods when sensing services are not needed the entire WSN will enter the Hibernation mode to conserve energy. The SS-trees must be computed with minimizing number of shared nodes (nodes belonging to multiple SS-trees), minimizing co-SS tree neighbors of each node, and minimizing the cost of forwarding messages between the data sink and each node. Rick W. Ha et al proposes a

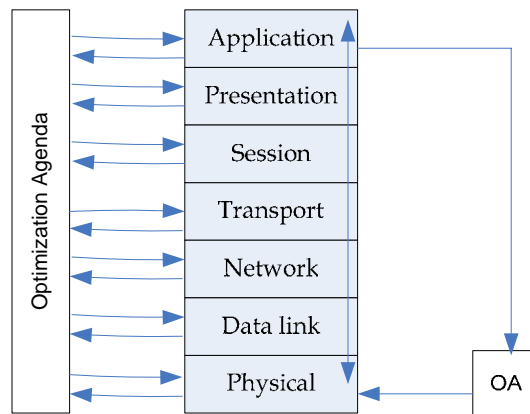
greedy algorithm to compute the SS-trees. The proposed algorithm follows a greedy depth-first approach that constructs the SS-trees from the bottom up on a branch-by-branch basis. After computing the SS-trees, an optimal sleep schedule that maximizes energy efficiency must be determined. The length of the active and sleep period will increase the data delay. The proposed SS-Tree design streamlines the routing procedures by restricting individual sensor nodes to only maintain local connectivity information of its immediate 1-hop neighbors.

Shuguang Cui et al, emphasize that the energy efficiency must be supported across all layers of the protocol stack through a cross-layer design [60]. [59] They analyze energy-efficient joint routing, scheduling, and link adaptation strategies that maximize the network lifetime. They propose variable-length TDMA schemes where the slot length is optimally assigned according to the routing requirement while minimizing the energy consumption across the network. They show that the optimization problems can be transferred into or approximated by convex problems that can be solved using known techniques. They show that link adaptation be able to further improve the energy efficiency when jointly designed with MAC and routing. In addition to reduce energy consumption, Link adaptation may reduce transmission time in relay nodes by using higher constellation sizes such as the extra circuit energy consumption is reduced.

Weilian Su and Tat L. Lim propose a cross layer design and optimization framework, and the concept of using an optimization agent (OA) to provide the exchange and control of information between the various protocol layers to improve performance in wireless sensor network [61]. The architecture of the proposed framework, as shown in Figure 2-12, which is redrawn from [61], consists of a proposed optimization agent (OA) which

facilitates interaction between various protocol layers by serving as a database where essential information such as node identification number, hop count, energy level, and link status are maintained.

Weilian Su and Tat L. conduct the performance measurements to study the effects of interference and transmission range for a group of wireless sensors. The results of their performance measurements help to facilitate the design and development of the OA. The OA can be used to trigger an increase in transmit power to overcome the effects of mobility or channel impairments due to fading when it detects a degradation due in BER. Alternatively, it can reduce the transmit power to conserve energy to prolong its lifetime operations in the absence of mobility or channel fading. The OA can also be used to provide QoS provisioning for different types of traffic. This can be done by tagging different priority traffic with different transmit power levels.



**Figure 2-12 : A Proposed cross-layer optimization framework**

Changsu Suli et al propose an energy efficient cross-layer MAC protocol for WSN. It is named MAC-CROSS [62]. In the proposed protocol, the routing information at the

network layer is utilized for the MAC layer such that it can maximize sleep duration of each node. In MAC-CROSS protocol the nodes are categorized into three types: Communicating Parties (CP) which refers to any node currently participating in the actual data transmission, Upcoming Communicating Parties (UP) which refers to any node to be involved in the actual data transmission, and Third Parties (TP) which refers to any node are not included on a routing path. The UP nodes are asked to wake up while other TP nodes can remain in their sleep modes. The RTS/CTS control frames are modified in the MAC-CROSS protocol. The modification is needed to inform a node that its state is changed to UP or TP in the corresponding listen/sleep period. A new field; `Final_destination_Addr`, is added to the RTS. On the other hand, a new field; `UP_Addr` is added to the CTS and it informs which node is UP to its neighbors. When a node B receives an RTS from another node A including the final destination address of the sink, B's routing agent refers to the routing table for getting the UP (node C) and informs back to its own MAC. The MAC agent of Node B then transmits CTS packet including the UP information. After receiving the CTS packets from node B, C changes its state to UP and another neighbor nodes change their states to TP and will go to sleep.

Table 2-2 shows summary of cross-layer design protocols for WSN.

**Table 2-2 : Summary of Cross layer Protocols for WSN**

Protocol	Layers	Approach	Evaluation method	Application	Network Topology	Cross layer Objective	Performance metrics
ECPS	MAC, Network	Mathematical Model: probabilistic dynamic programming	Experiment		Random (Static)	Maximization of probability of sending packet to its D at n transmission	Energy
E2LA	MAC, Network	Mathematical Model: probabilistic dynamic programming	Experiment		Random (Static)	Minimize Energy :- Multiple simultaneous routes Load distribution	Energy
MAC CROSS	MAC, Network	Heuristic	Simulation Hardware Implementation (MICAZ)		Random (Static)	Maximize Sleep Duration	Energy
O-Aloha	Physical, MAC	Heuristic	Simulation	SENMA	Random	Maximize throughput	Throughput
POWER	Physical, MAC, Network	Heuristic			Uniform (Static)	Optimize number of cluster	Energy
Weilian Su	ALL layers	Framework (optimization Agent)	Experimental (MICAZ)		Random	Optimize performance of WSN	Link Quality Packet Received
Shunguang Cui	Network, MAC, Link layer	Modeling as optimization problem	Analytical		Random	Maximize network lifetime	Network lifetime
Sense-Sleep Trees (SS-Trees)	MAC, Network	Heuristic	Simulation	Surveillance	Mesh-based	Maximizing Network lifetime, and monitoring coverage	Network lifetime Energy consumed
Game Theoretic Approach	Application, Physical	Game Theory	Analytical		Random	Minimize total distortion	Distortion coverage
In Yeup Kong	Physical, MAC, Network	Mathematical	Analytical		Random	Maximize Network lifetime	
Cross Layer Scheduling	MAC, Network	Heuristic	Simulation	Periodic	Random	Maximize network lifetime	Network lifetime
Cross Layer design for cluster formulation	MAC, Physical, Network	Heuristic	Simulation	Periodic	Uniform distribution	Maximize network lifetime	Network lifetime

## 2.4 INTEGER LINEAR PROGRAMMIN IN ILP

To explore optimal solution of any problem, two techniques are usually used; heuristic searching or Integer linear programming (ILP). ILP is used to formulate some WSN problems. In this section, we will present some of the problems that are formulated using ILP.

Chamam [66] address the problem of maximizing sensor networks lifetime under area coverage constraint. They propose a scheduling mechanism that calculates, for every time slot of the network operating period, an optimal covering subset of sensors that will be activated while all other sensors will go on Sleep. This mechanisms aim at balancing energy dissipation over sensors, thus maximizing network lifetime. They model this problem as an Integer Linear Programming (ILP) problem, which resolved using ILOG CPLEX and they show that the obtained solutions provide for more balanced energy consumption when they increase a balancing exponent  $\lambda$  that increases network lifetime. We also propose a greedy heuristic that could be implemented to tackle the exponentially increasing processing time of CPLEX.

Friderikos et al [67] propose a family of mathematical programs for both the uncapacitate and capacitated joint gateway selection and routing (U/C-GSR) problem in wireless mesh networks. They formulate the problem using the shortest path cost matrix (SPM) and prove that it gives the optimal solution when applied to the uncapacitated case but can lead to an arbitrary large optimality gap in the capacitated case. Furthermore, an augmented mathematical program is developed where link capacities are allowed to take values from a discrete set depending on the link distance. In this case, the multi-rate

capabilities of WMNs (via, for example, adaptive modulation and coding) can be modeled. Evidence from numerical investigations shows that using the SPM formulation realistic network sizes of WMNs can be solved.

Raja and baljai [68] presents a formulation to the Capacitated Minimal Directed Tree Problem. The formulation is amenable to several relaxation procedures. He proves that the proposed formulation is loop free. His objective is to minimize the total capacity at all links of the tree given the maximum capacity at each link is known.



# Chapter Three

## THE GENERALIZED ENERGY-EFFICIENT TIME-BASED COMMUNICATION PROTOCOL FOR WIRELESS SENSOR NETWORKS

In this chapter we shall discuss in details our general framework for energy-efficient time-based routing. In section 3.1, we discuss the shortcomings of EAD and LEACH protocols, and how we overcome these shortcomings in our framework. Then in section 3.2, we explain our framework (GET). A special case of GET which is called Energy Efficient Distributed Schedule-based (EEDS) protocol is discussed in section 3.3. Section 3.4 presents a generalization of EAD protocol.

### 3.1 SHORTCOMINGS OF EAD AND LEACH

As we mentioned in section 0, In EAD protocol, a tree is built from all nodes toward the sink. All the nodes of the network will communicate with the sink through few nodes that are close to sink. These nodes which are connected directly to sink are called gateway. A random scheme such as CSMA is used to forward data from nodes to the

sink. Therefore, all non-leaf nodes such as gateways must be ON through the whole data transmission phase. Since the gateways are at the upper level of tree, they will spend more time waiting for data packet to come from nodes at the lower level of the tree. Therefore, the gateways will waste a lot of energy while they are waiting. Hence they will die early. When the gateways die, all the remaining nodes will not be able to communicate with the sink. The remaining nodes will be considered as isolated nodes, although they still have enough energy.

On the other hand, in LEACH protocol, clusters are constructed to forward data from all nodes to sink. Few nodes will be selected as heads. All the remaining nodes will communicate with the sink through these heads. Each node will select the closest head as its parent. TDMA scheme will be used to forward data within the cluster. And random scheme such as CSMA is used to forward data from head to sink. Although each node selects the closest head to be its parent, the closest head of a node may not be close to it, the head may be located at long distance from a node. Therefore, a node will consume more energy in transmitting data for long distance. Moreover, since few heads are identified and all the nodes will be connected to these heads, number of children for each head will be high. The heads will consume more energy due to receiving data packets from their children.

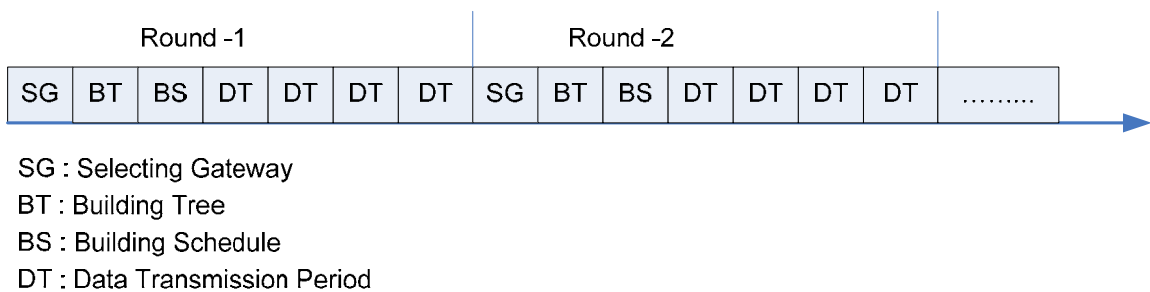
In designing our framework (GET), we try to overcome these shortcomings. GET differs from LEACH protocol, in which a tree is built from all nodes toward the sink instead of building multiple clusters with different heads connected to the sink. Therefore, each node in GET will transmit its data to its closest neighbor; its transmission distance will be shorter and little energy will be consumed during transmission. Although

a routing tree is built in GET as done in EAD protocol, the tree in GET differs from the tree built in EAD protocol in which any node on the network can be a gateway. While in EAD protocol, only the nodes that are close to sink will be gateways. A mechanism to select the gateway based on the residual energy of nodes is proposed. On the other hand, under our proposal, a node will join a tree only if it has sufficient energy that enables it to work for the whole round. This constraint is necessary to ensure the validity of the schedule for the whole round. Moreover, in GET, an efficient TDMA schedule is built in a distributed manner. The non-leaf node will be ON for its assigned time slots only instead of being ON for the whole data transmission period as in EAD and LEACH protocols. Therefore, energy consumption is optimized and concerned nodes will be ON only when it is necessary. In other words, our protocol works toward integrating energy-aware routing tree protocol and a distributed TDMA scheduler for a longer lifetime sensor network while maintaining high data throughput.

### 3.2 GET DESCRIPTION

In designing our protocol (GET), we assume that each node has the ability to transmit its data for a long distance, i.e. its transmission can reach the sink. This can be considered as a realistic assumption as MICA2 (from crossbow 2008) has transmission range more than 150 m. Each node has power control capability such that the transmission energy depends on the distance to the destination node. Such sensors are available in the market [64]. When a node sends data to its nearest neighbor, the transmission energy will be small compared to the transmission energy required to transmit data to the sink. We assume that each sensor node has multi-channel transceiver so it can use different

frequencies for transmitting and receiving, this assumption is a realistic one as several new sensor hardware implementations such as MICA2, and IMOTE2 from Crossbow support multi-channel transceivers [64]. Moreover, we assumed that all nodes are synchronized. This assumption is widely used in literature [25]. Regarding the application of the network, we assume that the event that is being monitored is periodic, so data transmission from sensor nodes to the sink will start at specific time, and it will be repeated periodically. We assume also that all the nodes that are located close to each other have correlated data. Hence, data aggregation will be used and it will reduce data redundancy.



**Figure 3-1: Time frame for GET protocol**

In GET protocol, time is divided into rounds. Each round consists of four phases: Selecting the Gateways (SG), Building the Tree (BT), Building the Schedule (BS), and Data Transmission (DT) as shown in Figure 3-1. Note that the phase size which depicted in Figure 3-1 is not to scale. In the first phase, gateways are selected; the gateway is a node that communicates directly with the sink. In the second phase, a tree rooted at the sink is built. The tree consists of leaf and non-leaf nodes. Leaf nodes sense the monitored area and transmit the corresponding data to its parent. On the other hand, the non-leaf nodes also sense the surrounding environment and they act as intermediate nodes to

transmit data from lower level to upper level of the tree. A non-leaf node will consume more energy than leaf node. Based on this tree, a TDMA schedule is built in a distributed manner in phase-3. Finally, in the fourth phase, data is transmitted from sensor nodes to the sink following the schedule prepared in phase-3. Data transmission period represents the time needed to forward all data packets in a single round. Data transmission period may be repeated many times in a single round as shown in Figure 3-1. Figure 3-1 shows two rounds; in each round, selecting gateway phase, building tree phase, building schedule phase and four data transmission periods are shown. The number of data transmission periods in a single round depends on the application. With a small number of data transmission periods in a single round, the tree will be rebuilt very frequently. Therefore, energy consumption will be distributed among all nodes, while the throughput will be lower because no data packets will be delivered to the sink in the building tree phase. On the other hand, with large number of data transmission periods in a single round, the same tree will be utilized for longer time, more data packets will be delivered to the sink, but energy consumption will not be distributed fairly among nodes.

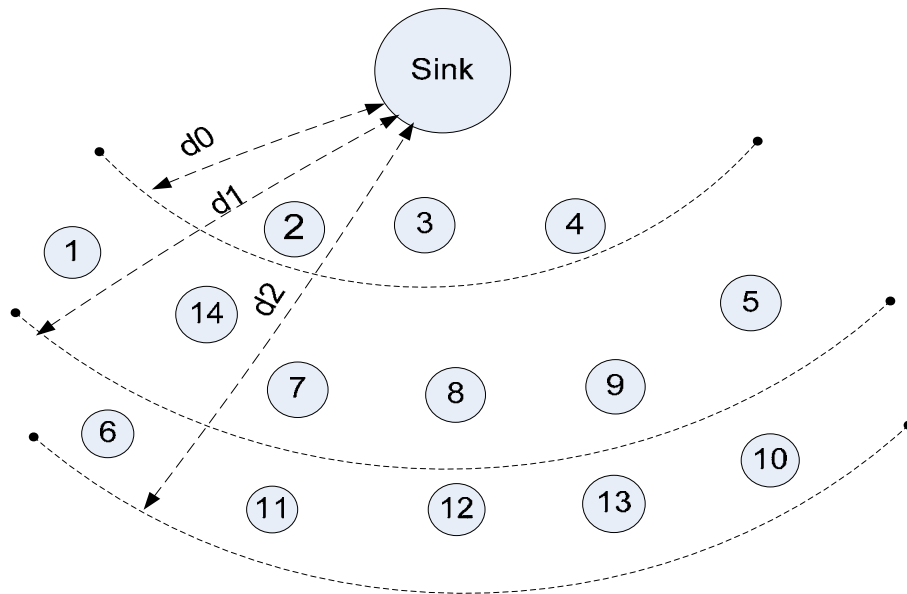
The following subsection explains in details each phase.

### 3.2.1 SELECTING THE GATEWAYS

In this phase, gateway nodes are selected. It is assumed that the network is virtually divided into tiers. Figure 3-2 shows an example of a network and its associated tiers. Each tier includes all nodes that can hear a signal transmitted with a specific energy from the sink. For example,  $tier_0$  includes all nodes that can hear the signal transmitted from sink with transmission energy equals to  $E_0$ .  $Tier_1$  includes all nodes that can hear

the signal transmitted from sink with transmission energy equals to  $E_1$ , where  $E_1 > E_0$  and so on.

Initially, the nodes of  $tier_0$  will be considered as potential candidate gateways. Based on their energy level, some of these nodes will advertise themselves as gateways. They will act as gateways until their residual energy drop below a threshold value  $E_{th}$ . Then new gateways will be selected from the nodes of  $tier_1$ . The selected nodes will act as gateways until their residual energy drop below  $E_{th}$  and so on. When all tiers are considered and no more nodes can be selected as gateways based on the current  $E_{th}$ , a new cycle will start, in this cycle new gateways will be selected from  $tier_0$  using smaller value of  $E_{th}$  and so on. The rationale behind this approach is to ensure energy consumption balance among all sensors and at the same time ensure maximum coverage.



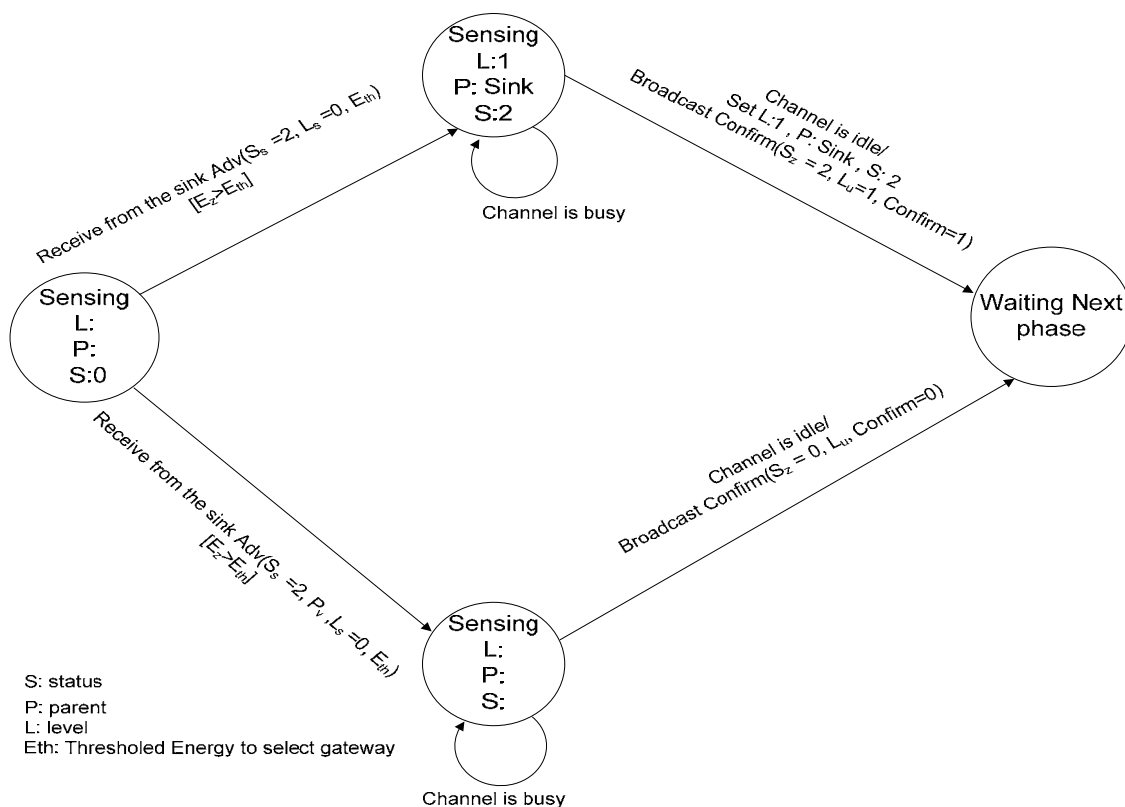
**Figure 3-2: A Sample Network with its tiers**

To select the gateways, the sink broadcasts an *ADV* message. The *ADV* message contains a field for  $E_{th}$ . Initially *ADV* message is broadcasted with energy  $E_0$  such that it

reaches the nodes of  $tier_0$  only. When a node receives the *ADV* message, it compares its residual energy with  $E_{th}$ , and then it responds with a *JOIN* message. A *JOIN* message contains a confirmation field. *Confirmation* is set to 1, if the node's residual energy is greater than  $E_{th}$ , i.e. the node can be a gateway and it selects the sink as its parent, otherwise *confirmation* is set to 0. After the node sends its *JOIN* message, it will act as gateway in the current round. Assuming a reliable channel, it does not need a confirmation from the sink to be a gateway. All nodes that send *JOIN* message with *confirmation field*=1 will be considered as gateways. If the sink receives *JOIN* messages from all nodes in the target tier and the *confirmation field* =0 in all the received *JOIN* messages, then no node from the target tier can be a gateway, since we assume that all nodes can reach the sink, the sink will broadcast a new *ADV* message with higher transmission energy  $E_1$  using the same  $E_{th}$  to select a gateway from the next tier. The nodes of the next tier will respond with *JOIN* messages according to their energy. The process will continue until all tiers are considered and no node has energy greater than  $E_{th}$ ; no node can be a gateway. A new cycle will start from  $tier_0$  with new  $E_{th}$ ,  $E_{th}(new)=eE_{th}(current)$ , where  $0 < e < 1$ . Following the same procedure as above, new gateway nodes will be selected from  $tier_0$ . For each cycle, a fixed  $E_{th}$  will be used, and at the beginning of each new cycle,  $E_{th}$  will be reduced by the factor  $e$ . The sink and sensor nodes will exchange messages using the CSMA mechanism. The node has to be ON until it receives the *ADV* message from the sink and then it sends the *JOIN* message. Since the node does not need confirmation from the sink, it will go to sleep immediately after sending the *JOIN* message.

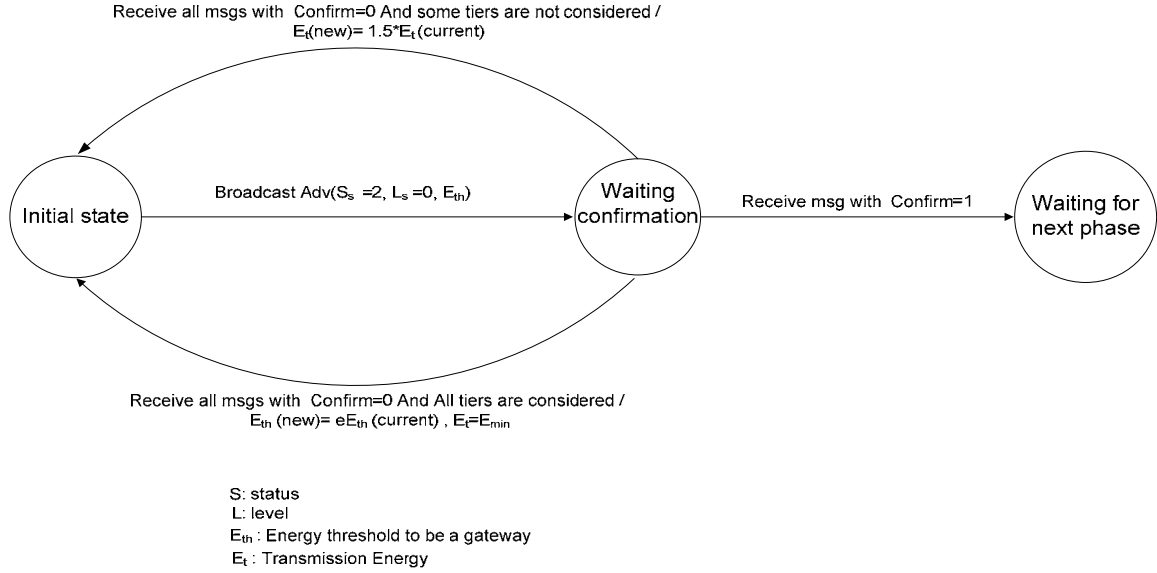
After selecting the gateways, the next phase will start to build the tree. The gateway nodes will initiate the process of building the tree. One question may be raised, when will the next phase start? in our simulation, we assume that all nodes will know the time at which the current phase is over and gateways are selected. In practice, a maximum time limit can be set. When this time limit is over, the next phase will start.

Figure 3-3 and Figure 3-4 show the state machine of a node and the sink respectively in the selecting gateway phase.



**Figure 3-3: The state machine for the node in selecting gateway phase**





**Figure 3-4: the state machine of the sink in the selecting gateway phase**

### 3.2.2 BUILDING THE TREE:

To build a tree rooted at the sink, we employ a modified version of the algorithm proposed in [4]. In the modified algorithm, the gateway nodes will initiate the process of building the tree. Building the tree is performed by broadcasting control messages. Each control message consists of four fields: *type*, *level*, *parent*, *energy*. For the sender node  $v$ ,  $type_v$  represents its status; 0: undefined; 1: leaf node; 2: non-leaf node.  $level_v$  refers to the number of hops from  $v$  to the sink.  $parent_v$  is the next hop of  $v$  in the path to the sink;  $energy_v$  is the residual energy  $E_v$ . Initially each node has status 0. The sink broadcasts  $msg(2, 0, NULL, \infty)$ . When a node  $v$  receives  $msg(2, level_u, parent_u, E_u)$  from node  $u$ , it becomes a leaf node, it senses the channel until it is idle, then waits for  $T_2^v$  time, if the channel is still idle,  $v$  broadcasts  $msg(1, level_u + 1, u, E_v)$ . If  $v$  receives  $msg(1, level_u, parent_u, E_u)$  from  $u$ , it senses the channel until it is idle, waits for  $T_1^v$  if the channel is still idle,  $v$  broadcasts  $msg(2, level_u + 1, u, E_v)$ . Then it becomes non-leaf node. If node

$v$  receives more than one message from different nodes before it broadcasts its message, it will select the node with larger energy as its parent. If both nodes have the same energy, it will select one of them randomly. The waiting node will go back to sensing state, if another node occupies the common channel before it times out. If a node  $v$  with status 1 receives  $msg(2, level_w, v, E_w)$  from node  $w$  indicating that  $v$  is its parent,  $v$  broadcasts  $msg(2, level_v, parent_v, E_v)$  immediately after the channel is idle. The process will continue until each node becomes leaf or non-leaf node. A sensor with status 2 becomes a leaf node if it detects that it has no children. Both  $T_1^v$  and  $T_2^v$  are chosen such that no two neighboring broadcasts are scheduled at the same time. On the other hand, to force the neighboring sensors with higher energy to broadcast earlier than those nodes with a lower residual power, both  $T_1^v$  and  $T_2^v$  must be monotonically decreasing functions of  $E_v$ . [4] chooses  $T_1^v = 2 * t_0 + \frac{c}{E_v}$  and  $T_2^v = t_0 + \frac{c}{E_v}$  where  $t_0$  is the upper bound of the propagation time between any pair of neighboring sensors and  $c > 0$  is an adjusting constant. Figure 3-5 shows the state machine of a node in building tree phase.

Let us show how the tree is built for network shown in Figure 3-6 assuming that nodes 2, 3 and 4 are identified themselves as gateways. In Figure 3-6 a set of nodes enclosed by dot line represent one region. We assume that each node in a specific region will hear only the transmission of all nodes in the same region. There are eight regions in the network. The regions and associated nodes are shown in Table 3-1. Each node is labeled with its id and two numbers represent its level and status. Initially, the status of each node is 0 while the level of each node is undefined.

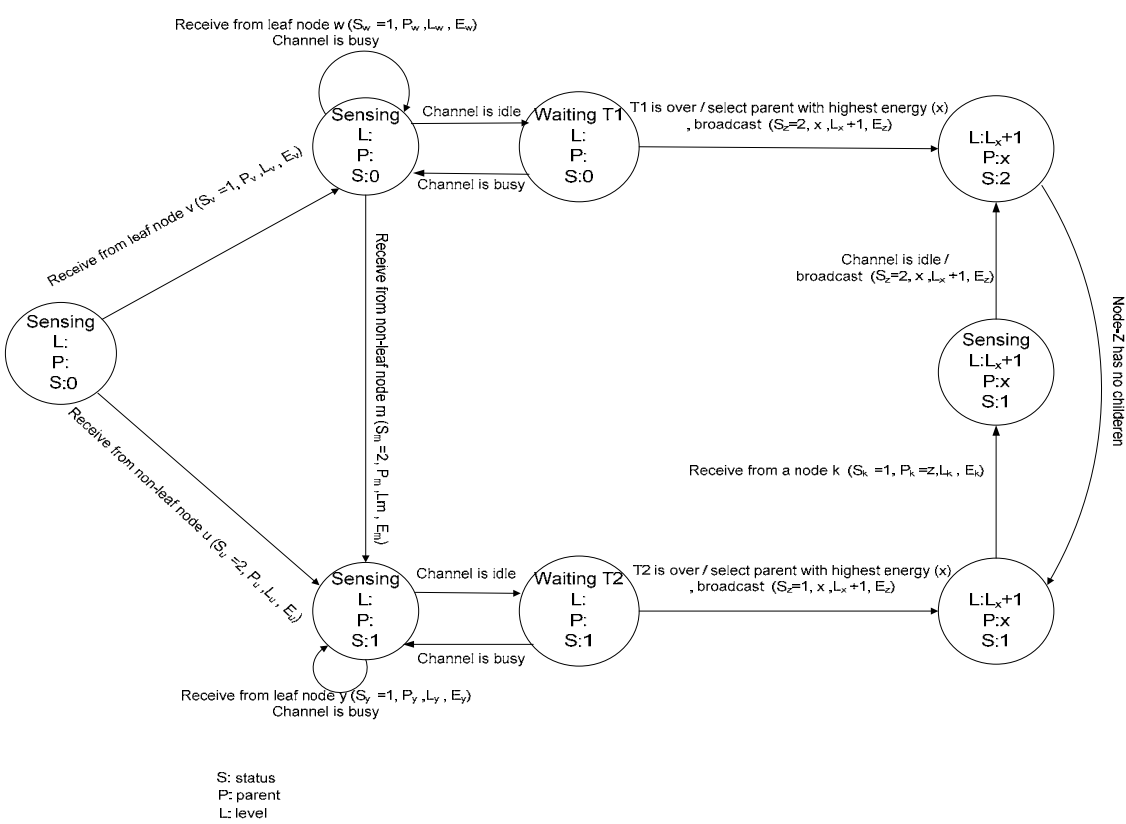


Figure 3-5: The state machine of a node in building tree phase

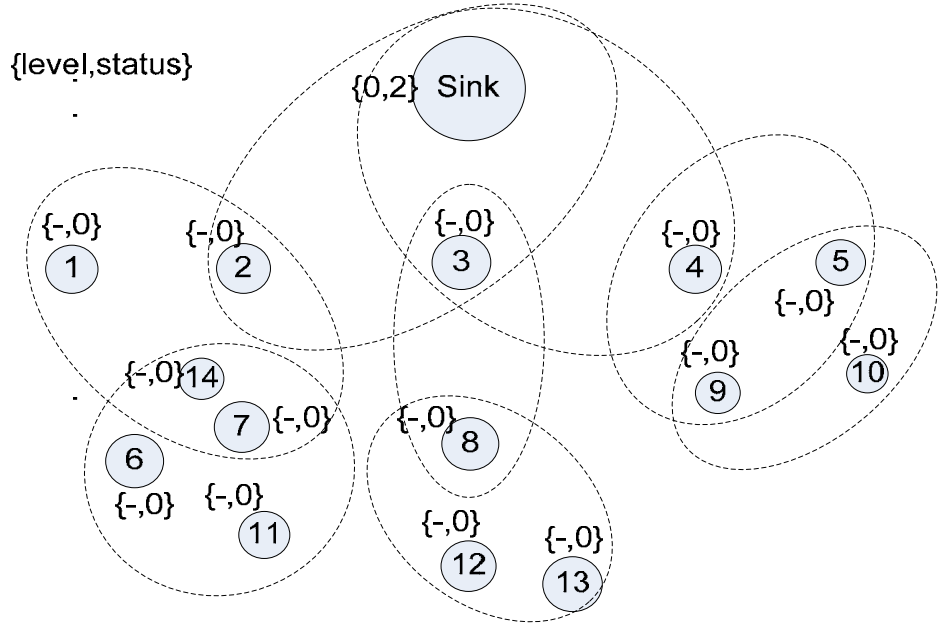
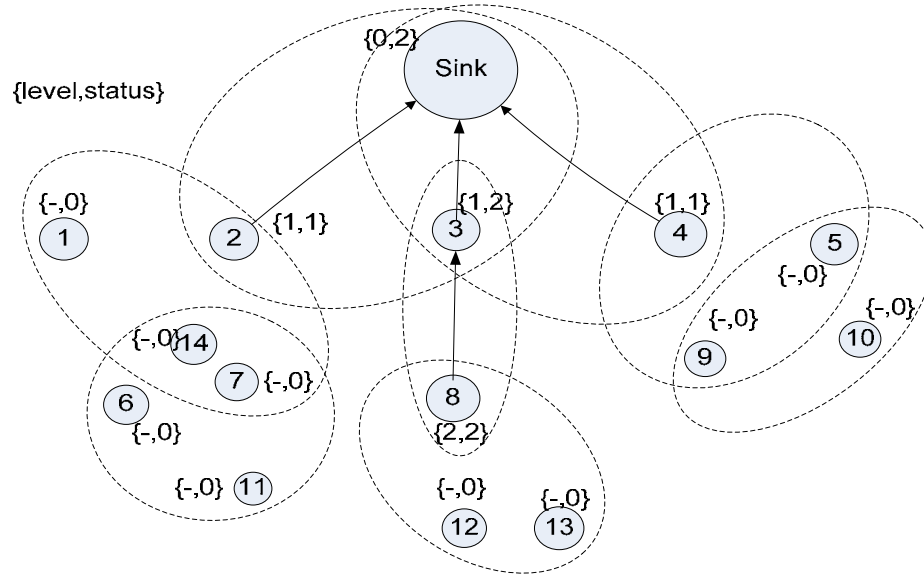


Figure 3-6: A sample Network

**Table 3-1: Regions of the Network**

<i>Region</i>		
<i>R1</i>	Sink,2,3	Sink, and 3 has more Energy than 2
<i>R2</i>	Sink,3,4	Sink, and 3 has more Energy than 4
<i>R3</i>	1,2,7,14	2 then 7
<i>R4</i>	3,8	3
<i>R5</i>	4,5,9	4
<i>R6</i>	6,7,11,14	7
<i>R7</i>	8,12,13	8
<i>R8</i>	5,9,10	9

Since Nodes: 2, 3, and 4 identify themselves to be gateways in the previous phase, their status are set to 1 and they are ready to initiate building tree process by broadcasting the control message. According to the building tree algorithm and because node 3 has the maximum energy among nodes 2, 3, 4 it has the least waiting time and it will broadcast the message  $msg(1,1,0,E_3)$  before the other nodes. Nodes: 2, 4 and 8 will receive this message. Nodes: 2, and 4 now have two messages, one from the sink and the other from node-3. They will select the sink as their parent since it has more energy than node 3. So when their waiting time is over and the channel is idle, nodes: 2 and 4 will broadcast  $msg(1,1,0,E_2)$  and  $msg(1,1,0,E_4)$  respectively, one of them will broadcast its message before the other based on their energy. On the other hand, node-8 will select node-3 as its parent. It will broadcast  $msg(2,2,3,E_8)$ . When node-3 receives this message, it knows that one of the nodes select it as its parent. It will change its status to 2, then when the channel becomes idle, it immediately, without waiting, broadcasts  $msg(2,1,0,E_3)$ . At this stage, a partial tree is built as shown in Figure 3-7.

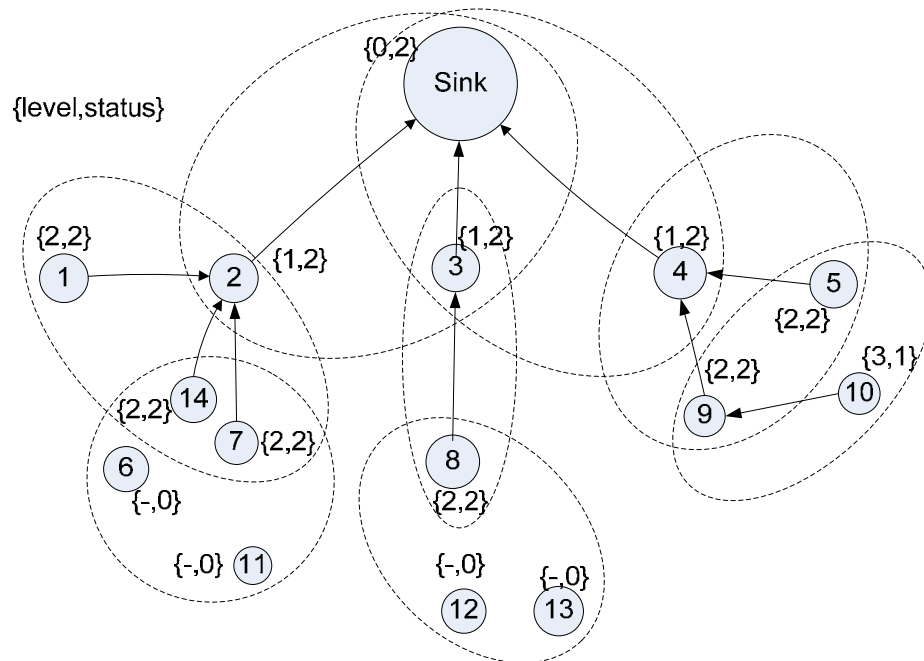


**Figure 3-7: A Partial Tree-1**

Nodes: 1, 14, and 7 will hear the message broadcasted by node-2. The waiting time for node-7 will be the least one since it has the maximum energy. It will broadcast  $msg(2,2,2,E_7)$  before nodes 1 and 14. Nodes 6, 11 and 14 will receive it. Now node-14 received two messages, the first message from node-2 and the second one from node-7. Since node-2 has more energy than node-7, node-14 will select node-2 as its parent. When waiting time of node-14 is over and the channel become idle node-14 will broadcast  $msg(2,2,2,E_{14})$ . Node-1 will also broadcast  $msg(2,2,2,E_1)$  when its waiting time is over and the channel is idle. When node-2 receives the message broadcasted by node-7, it will change its status to 2, then when the channel becomes idle node-2 immediately broadcasts  $msg(2,1,0,E_2)$ .

On the other hand, nodes 5 and 9 will hear the message broadcasted by node-4. Since node-9 has more energy than node-5, it will broadcast its  $msg(2,2,4,E_9)$  before node-5. When node-4 receives this message, it will change its status to 2, then when the channel becomes idle node-4 immediately broadcasts  $msg(2,1,0,E_4)$ . Nodes 5 and 10 will

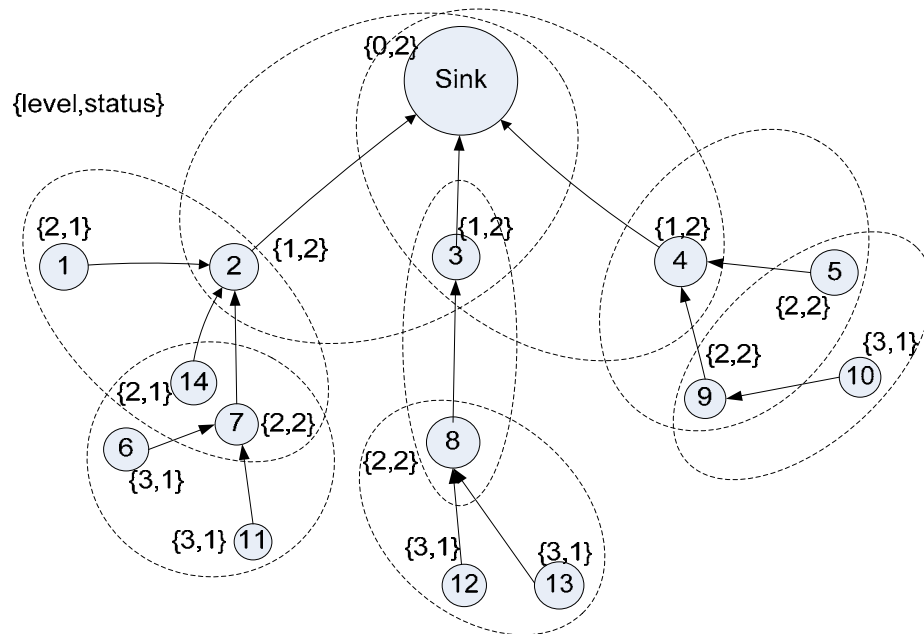
receive the messages broadcasted by node 9. Now node 5 has two messages, one from node-4 and the other from node-9. Since node-4 has more energy than node-9, node-5 will select node-4 as its parent and it will broadcast  $msg(2,2,4,E_5)$ . Node-10 will select node-9 as its parent, and it will broadcast  $msg(1,3,9,E_{10})$ . The tree now becomes as shown in Figure 3-8.



**Figure 3-8: A Partial Tree-2**

When nodes 6 and 11 receive the message broadcasted by node-7, they will change their status to 1. They will select node-7 as their parent even though they receive another message from node-14 since node-7 has more energy than node-14. When their waiting time is over and the channel is idle, they will broadcast  $msg(1,3,7,E_6)$  and  $msg(1,3,7,E_{11})$  respectively. If node-6 broadcast before node-11, then node-11 will have three messages; from node 14, node-7 and node-6. But it will select node 7 as parent because it has the maximum energy.

By the same way, nodes 12 and 13 will receive the message broadcasted by node-8. They will change their status to 1. They will broadcast  $msg(1,2,8,E_{12})$  and  $msg(1,2,8,E_{13})$  respectively. One of them will broadcast before the other based on their energy. If node-12 will broadcast firstly, then node-13 will have two messages; one from node-12 and the other from node-8. However, it selects node-8 as its parent since it has more energy. Now the status of Nodes 1, 14 and 5 is 2 and they have no children. Therefore, they will change their status to 1. The tree now becomes as shown in Figure 3-9. Nodes 2, 3 and 4 in Figure 3-9 are connected directly to the sink and they are considered as gateway nodes.



**Figure 3-9: A complete tree-1**

If gateways are selected from tier<sub>1</sub>, a different tree will be built as shown in Figure 3-10

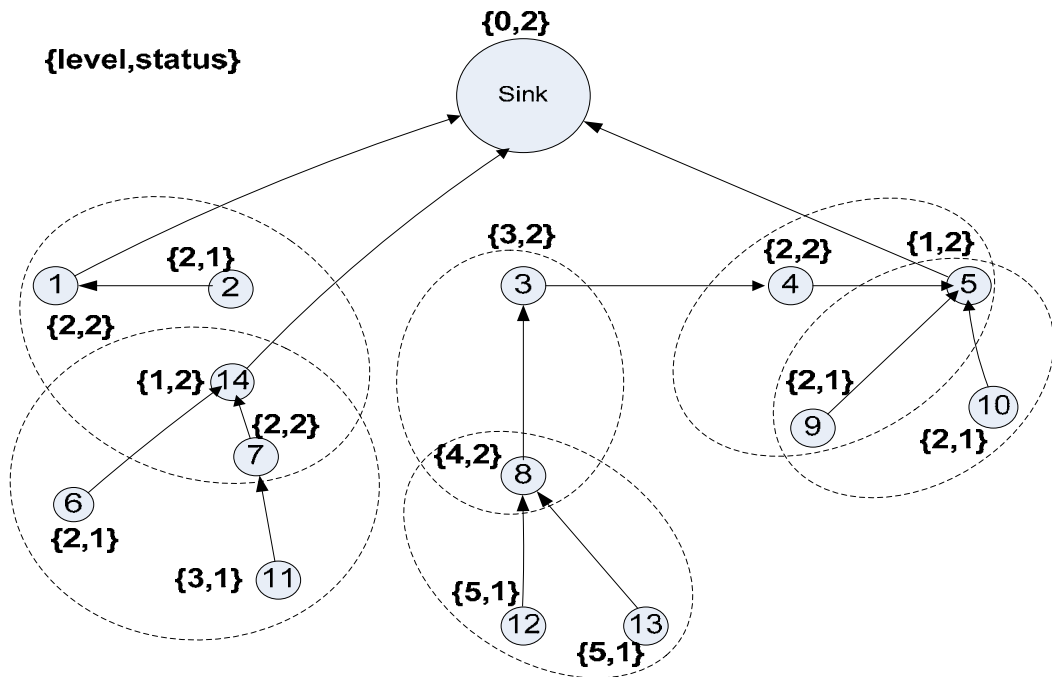


Figure 3-10: A complete tree-II

### 3.2.3 BUILDING THE SCHEDULE:

The essence of this phase is to build a TDMA schedule for data transmission in a distributed manner. The schedule will be built assuming that in the data transmission period, all nodes connected to the sink through the same gateway will use the same frequency to transmit their data. Therefore, any two nodes that are connected to the sink through two different gateways will be able to transmit simultaneously. Assuming that we have enough multiple channels, when a node is selected as a gateway, it will pick a channel randomly. After building the tree, the process of building the schedule is triggered. For each node, we identify two time constants: Time Ready to Receive ( $TRR$ ) and Time Ready to Transmit ( $TRT$ ). For a node  $v$ ,  $TRR_v$  represents the time slot when the node is ready to receive from its children. While  $TRT_v$  represents the time slot when a



node can transmit to its parent. The period  $[TRR_v, TRT_v + I]$  represents the time period at which the node must be wake up and its transceiver will be ON. Assuming  $t_0$  represents the time slot at which the periodic sensing event occurred and the data is already collected from the monitored environment. For the leaf node,  $TRT_v = t_0$  while  $TRR_v$  is not valid since it does not have children. On the other hand, for non-leaf node  $v$ :

$$\begin{aligned} TRR_v &= \text{Max}(TRT_i) \quad i = 1, 2, 3, \dots, n_v^c \\ TRT_v &= TRR_v + n_v^c T_t \end{aligned} \quad (1)$$

Where  $i$  represent an index for the children of  $v$  node,  $n_v^c$  represents the count of  $v$ 's children, and it represents the time needed to transmit one data packet. We select Max function so the parent node will be ON only when all its children are ready to transmit. Hence, the parent will be ON for one shot to receive from all its children, which is better than going from ON to OFF many times. Going from ON to OFF will consume more energy [3]. Although some nodes will be ready to transmit very early, there data will not be needed because we assume that the data coming from all children are correlated and it will be aggregated into one packet. The time for data aggregation is neglected. When data are received from all children, the parent will aggregate data then it will transmit the aggregated data to the next node.

Initially, each leaf node will transmit its  $TRT$  value to its parent. When a parent receives all values from all its children, it calculates its  $TRR$  and  $TRT$  using (1) and builds the schedule for its children. Then it transmits its  $TRT$  to its parent and broadcast the schedule to its children. The process will continue until each node receives its assigned slot from its parent. Both leaf and non-leaf nodes use CSMA/CA protocol to exchange data ( $TRT$  and the Schedule). The pseudo code for building schedule algorithm is shown in Figure 3-11

```

For leaf node j
    Transmit  $TRT_j$  to its parent
For non-leaf node j
    Receive  $TRT_i$  from all j's children
     $S_j = \{ i : i \text{ is children for } j \}$ 
    Calculate  $TRR_j$  (Eq#1)
     $T_s = TRR_j$  //  $T_s$  is current empty time slot
    While ( $S_j \neq \emptyset$ )
    {
        Select node  $w$  from  $S_j$  with minimum  $TRT$ 
         $T_w = T_s$  // node  $w$  is scheduled to transmit at  $T_w$ 
         $S_j = S_j - \{w\}$ 
         $T_s = T_s + T_t$  //  $T_t$  is time to transmit one data packet
    }
    Calculate  $TRT_j$ 
    Transmit  $TRT_j$  to the parent

```

**Figure 3-11 : Pseudo code for building schedule**

For example, to build the schedule for the tree shown in Figure 3-9, all the leaf nodes  $\{1,5,6,10,11,12,13,14\}$  will identify their  $(TRR, TRT)$  as  $(-, t_0)$  and they transmit  $t_0$  to their parents. Since a non-leaf node such as: 7, 8 and 9 receives  $t_0$  from all its leaf children,  $TRR$  for all these nodes will be  $t_0$ .  $TRT$  for nodes 7 and 8 will be  $t_0 + 2T_t$  since each node has two children. While  $TRT$  for node 9 will be  $t_0 + T_t$  since it has one child only. Node 7 will build a schedule for its children. For example, node 11 will be scheduled to transmit at time  $t_0$  and node 6 will be scheduled to transmit at time  $t_0 + T_t$ . on the other hand, nodes 8 and 9 will also build the schedule for their children. Nodes 12 and 13 will be scheduled to transmit at  $t_0$ , and  $t_0 + T_t$ , respectively, and Node 10 will be scheduled to transmit at  $t_0$ . Node 7 will transmit its  $TRT$ ,  $t_0 + 2T_t$ , to its parent (node 2). Node 2 receives  $t_0$ ,  $t_0$ ,  $t_0 + 2T_t$  from nodes 1, 14, 7 respectively. Its  $TRR$  will be  $t_0 + 2T_t$  and its  $TRT$  will be  $t_0 + 5T_t$ . Node-2 will build the schedule for its children such that nodes 1, 14, and 7 will be scheduled to transmit at  $t_0 + 4T_t$ ,  $t_0 + 3T_t$  and  $t_0 + 2T_t$  respectively. Node 8 will transmit its  $TRT$ ,  $t_0 + 2T_t$ , to its parent (node 3). Since node-3 has only one child, its  $TRR$  will be  $t_0 + 2T_t$  and its  $TRT$

will be  $t_0+3T_t$ . Node 8 will be scheduled to transmit at  $t_0+2T_t$ . Node 9 will transmit its  $TRT$ ,  $t_0+T_t$ , to its parent (node 4). Node 4 receives  $t_0, t_0+T_t$  from nodes 5, and 9 respectively. Its  $TRR$  will be  $t_0+T_t$ . Since Node 4 has two children, its  $TRT$  will be  $t_0+3T_t$ . Node 4 will build the schedule for its children such that node 9 will transmit at  $t_0+T_t$  while node 5 will transmit at  $t_0+2T_t$ . Nodes 2, 3 and 4 will transmit to sink their  $TRT$ :  $t_0+5T_t, t_0+3T_t$  and  $t_0+3T_t$  respectively. The  $TRR$  of the sink will be  $t_0+5T_t$ . and it builds the schedule for its children such that node 2 will transmit at  $t_0+5T_t$ , while node 3 will transmit at  $t_0+6T_t$ , and node 4 will transmit at  $t_0+7T_t$ . Figure 3-12 shows  $TRR$ ,  $TRT$ , and the scheduled time ( $T_s$ ) for each node of network of Figure 3-9.

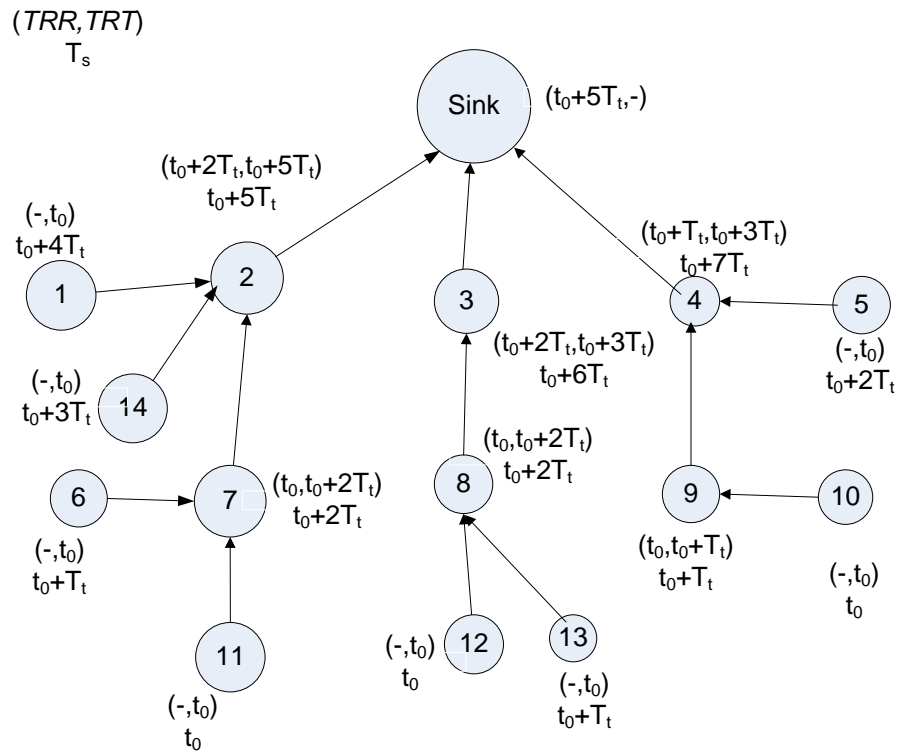


Figure 3-12: The nodes of the network with their transmission time  $TRR$ ,  $TRT$

<i>Sink</i>						R	R	R	
<i>Node 2</i>			R	R	R	T			
<i>Node 1</i>					T				
<i>Node 14</i>				T					
<i>Node 7</i>	R	R	T						
<i>Node 6</i>		T							
<i>Node 11</i>	T								
<i>Node 3</i>			R				T		
<i>Node 8</i>	R	R	T						
<i>Node 12</i>	T								
<i>Node 13</i>		T							
<i>Node 4</i>		R	R					T	
<i>Node 5</i>			T						
<i>Node 9</i>	R	T							
<i>Node 10</i>	T								
	$t_0$	$t_0+T_t$	$t_0+2T_t$	$t_0+3T_t$	$t_0+4T_t$	$t_0+5T_t$	$t_0+6T_t$	$t_0+7T_t$	$t_0+8T_t$

**Figure 3-13: A schedule of the example Network**

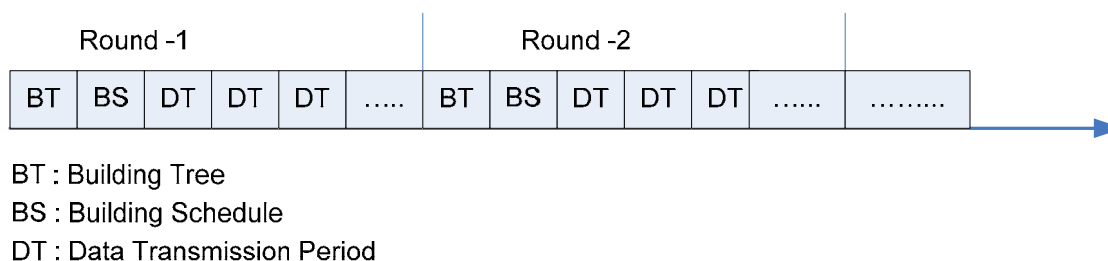
### 3.2.4 DATA TRANSMISSION PHASE:

At this phase, data packets will be forwarded from all nodes toward the sink, To avoid interference among transmissions of different nodes, each gateway and its associated nodes will use different frequency from other gateways. Each node will be ON only at their assigned slots. The leaf nodes will be ON only for one slot; to transmit data to its parent. On the other hand, the non-leaf node will be ON during the slots when its children transmit and during its assigned slot to transmit to its parent. The number of slots when the non-leaf node is ON is equal to the number of its children in addition to one slot for transmission to its parent. The data transmission phase can be repeated many times (periods) for the same schedule but each node must have sufficient energy to stay alive

during all data transmission periods. Energy required for a node to stay alive for a given number of transmission periods is calculated taking into account the number of nodes in the node's proximity. Figure 3-13 shows a timing diagram for each node of the network shown in Figure 3-9. For each node, a time slot labeled by  $R$  represents time slot at which a node receives data, while a time slot labeled by  $T$  represents a time slot at which a node transmits data.

### 3.3 ENERGY-EFFICIENT DISTRIBUTED SCHEDULE-BASED COMMUNICATION PROTOCOL FOR WSN (EEDS)

EEDS protocol is a special case of GET protocol. In EEDS, the gateways are the nodes that are close to the sink. The nodes those are located within  $tire_0$ . Since the gateways are the nodes that are close to sink, and since each node will select a parent among one of its closest neighbors, sensor nodes will not be required to transmit for long distance. Hence, the transmission range for each node is short. Each node can hear only the transmission of the nodes that are close to it (i.e. a node cannot hear the transmission of all the nodes in the network). Therefore, EEDS protocol can be implemented with sensors that have short transmission range.



**Figure 3-14: Time Frame for the EED**

Since any node close to the sink can act as a gateway, it is not required to select a gateway. Therefore, each round in EEDS consists of three phases only; Building the tree (BT), Building the schedule (BS), and Data Transmission (DT) as shown in Figure 3-14. In the first phase, as in GET protocol, a tree rooted at the sink is built. In EEDS, the building tree process will be initiated by the sink not by the gateways as in GET. The other two phases are similar to the corresponding phases in GET.

### 3.4 A GENERALIZED ENERGY-AWARE DATA CENTRIC ROUTING FOR WIRELESS SENSOR NETWORK (EAD<sub>GENERAL</sub>)

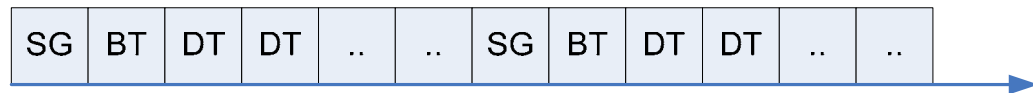
We use the selected gateway algorithm discussed in section 3.2.1 to generalize the Energy Aware Data centric routing Protocol (EAD) [4]. A detailed discussion of EAD protocol is discussed section 0. We call the new protocol EAD<sub>General</sub>. The proposed protocol intends to increase the lifetime of the network by increasing the number of candidate gateway nodes. The building tree protocol is generalized such that not only the nodes that are close to the sink can be connected directly to sink, but any node in the network can also be connected directly to the sink.

To generalize EAD protocol, we assume that each node has the ability to transmit its data for a long distance, i.e. its transmission can reach the sink. Each node has power control capability such that the transmission energy depends on the distance to the destination node. When a node sends data to its nearest neighbor, the transmission energy will be small compared with the transmission energy required to transmit data to the sink.

In our proposed protocol (EAD<sub>General</sub>), a new phase; Selecting Gateways (SG), is added. In this extra phase, gateways nodes, nodes that will communicate directly with the

sink, will be selected autonomously.  $EAD_{General}$  works in rounds, each round consists of three phases, selecting gateway phase, building tree phase, and data transmission phase.

The time frame for  $EAD_{General}$  is shown in Figure 3-15



SG: Selecting Gateway

BT: Building Tree

DT: Data Transmission

**Figure 3-15: Time frame for the  $EAD_{General}$  Protocol.**

### 3.5 CONCLUSION

In this chapter, we described in details our general framework (GET), EEDS protocol which is a special case of GET, and  $EAD_{General}$  which is a generalization of the EAD.

GET is intended for applications with periodic data traffic. In GET, a tree rooted at the sink is built to deliver data packets from different sensors to the sink. The time is divided into rounds. Each round consists of four phases. In the first phase, the gateway nodes are selected. In the second phase, starting from the selected gateway nodes, an energy-aware tree is built, and then a TDMA schedule is constructed in a distributed manner in the third phase. In the fourth phase, according to the TDMA schedule that is built in phase 3; data packets are forwarded from different nodes toward the sink. At the beginning of each round, new gateway nodes will be selected, the tree will be rebuilt and

a new TDMA schedule will be constructed. The data transmission phase can be repeated several times within a single round using the same tree and schedule.

EEDS is a special case of GET. In EEDS, only the nodes that are close to sink will act as gateways. All the nodes that hear the signal transmitted by the sink will be gateways; therefore, selecting gateways mechanism is not needed. And hence, each round in EEDS consists of three phases only: building tree, building TDMA schedule, and data transmission phase.

$EAD_{General}$  is a generalization of EAD protocol such that any node in the network can be a gateway. The electing gateway algorithm that is implemented in GET is integrated within EAD. Therefore, each round of  $EAD_{General}$  consists of three phase, selecting gateways, building tree, and building schedule.

In the next chapters we shall evaluate the performance of these protocols, and compare them with EAD and LEACH. The simulation setup that is used in the performance evaluation will be presented in the next chapter.



# Chapter Four

## SIMULATION SETUP

### 4.1 INTRODUCTION

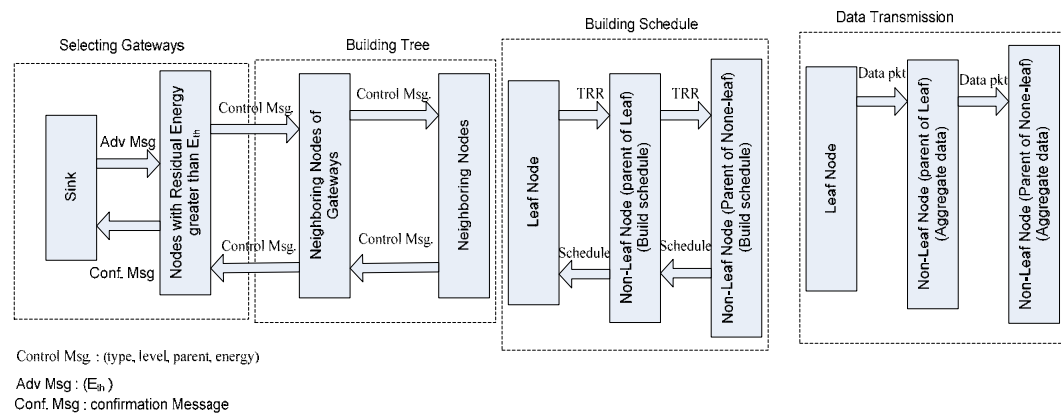
In the previous chapter, we describe GET, EEDS, and EAD<sub>General</sub> protocols in details. In this chapter, we shall present the simulation setup that is assumed when we evaluate the performance of our protocols.

To evaluate the performance of the proposed protocols, a simulator using C-language was built. In this chapter, we describe our simulation setup. Section 4.2 presents a description of the simulator. A description of the network topologies and how they are generated are described in section 4.3. In section 4.4, we present the energy model that is used in our simulation. Simulation assumptions will be discussed in 4.5. in section 4.6, we describe how the measures of the simulation are collected.

## 4.2 DESCRIPTION OF SIMULATOR

Our simulator tracks the actual behavior of each node in all phases as explained in the previous chapter. Figure 4-1 shows a block diagram of simulator. In our simulator, each node is represented by a structure which has different data fields. Some of data fields in the structure of the node:

- Node id
- Status of a node : transmitting, receiving, idle-listening, waiting, OFF
- List of neighbors
- List of interference neighbors
- List of children
- Current parent
- Residual energy
- X and Y coordinates of the node
- Flag to indicate weather a node has a message to transmit



**Figure 4-1: Block diagram of simulator**

The simulator is time-driven simulator. In selecting gateway, building tree, and building schedule phases, the simulation time is incremented by a time step equal to 0.01 milliseconds. At each time step of the simulation, each node is checked to determine its next status. For example, if a node is currently transmitting a packet, then the remaining time that is needed to finish transmitting the packet is decremented by the time step. If the remaining time reaches zero, the node status changed to finish transmitting. On the other hand, if the node intends to transmit a packet, it checks the channel. If the channel is idle, the node can transmit, otherwise, the node implements the binary back off algorithm to determine the waiting time. The node picks a random number in the interval  $1..2^m$  where  $m$  is back off round. Then the node waits for a time interval according to the picked random number. A node checks the channel status by checking the status of node's neighbors. If one of them is transmitting, then the channel is not idle. If none of its neighbors is transmitting, then the node can transmit. On the other hand, if the node is currently waiting, then the waiting time will be decremented by the time step. If the waiting time of the node reaches zero, then its status will change to a new status (intends to transmit). and so on. These are examples of operations within our simulator. To simulate all aspects of nodes behavior, we implement a lot of other operations such as collecting the signal from the channel.

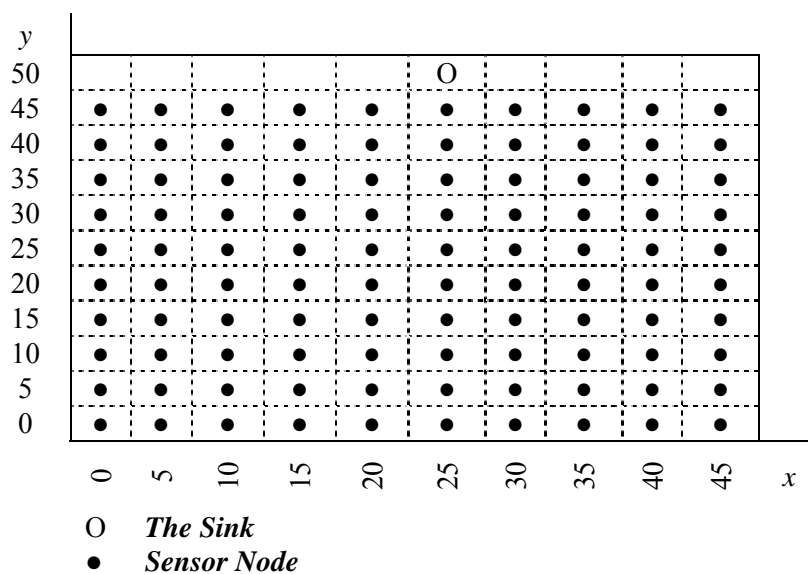
In the data transmission phase, the simulation time is incremented by the length of TDMA schedule.

Randomness is used extensively in our simulator. For example, each node will pick a random number when it implements the back off algorithm. Moreover, if multiple nodes

have a packet to transmit and all of them are ready to transmit, then one of them will be randomly selected to transmit. To guarantee a random behavior for our simulator for each configuration, our simulator will be run with a different seed.

### 4.3 NETWORK TOPOLOGIES

We evaluate the performance of our protocols using grid and random topologies. One example of the grid topologies we used is shown in Figure 4-2. In this topology, 100 sensors distributed in an area of  $50 \times 50 \text{ m}^2$ . The nodes are uniformly distributed between  $(x=0, y=0)$  and  $(x=45, y=45)$ . The sink is located at location  $(x=25, y=50)$  as shown in Figure 4-2.



**Figure 4-2: The Grid topology**

For random topology, 30 different networks are considered. To generate 30 different random network topologies, we write a program that generates the  $x$  and  $y$  coordinates of

each node for each topology.  $x$  and  $y$  coordinates are uniform random variables within the interval  $[0, Limit_x]$  and  $[0, Limit_y]$  respectively.  $Limit_x$  and  $Limit_y$  represent the length and width of the monitored area. To generate a different topology each time our program runs, it must be run with a different seed.

#### 4.4 ENERGY MODEL

We use a power control model in which the energy consumed during transmission depends on the transmission distance [3]. The energy consumed ( $E_{Tx}$ ) during transmission of  $k$  bits for a distance of  $d$  meters, and the energy consumed ( $E_{Rx}$ ) to receive  $k$  bits are calculated by:

$$\begin{aligned} E_{Tx} &= kE_{elec} + kE_{amp}d^2 \\ E_{Rx} &= kE_{elec} \end{aligned} \tag{1}$$

where  $E_{elec}$  represents the electronics energy and it depends on several factors such as the digital coding, modulation, filtering, and spreading of the signal. On the other hand,  $E_{amp}$  represents the amplifier energy. In our experiments, we assume  $E_{elec}=50\text{nJ/bit}$  and  $E_{amp}=100\text{pJ/bit/m}^2$ , which are the same values used in [3]. The initial energy stored in each node is 2 J. Since we are interested in comparing the different protocols in terms of the energy consumed in transmission and receiving states, we neglect the energy consumed in sensing the environment as it will be the same under all protocols. Furthermore, the node will be considered a dead node, and it will not participate in the coming round, if its energy becomes less than a threshold value ( $E_{threshold}$ ). For example, for the grid network shown in Figure 4-2,  $E_{threshold}$  can be calculated as follows: In the worst case, a node will be a non-leaf node and it will have eight children

in the coming round. The minimum energy needed for this node to be able to participate in the coming round is  $E_{threshold_{min}}$ .  $E_{threshold_{min}}$  is calculated using (1). For a single transmission period, the node will receive a maximum of eight data packets and transmit one data packet. Then,  $E_{threshold_{min}}$  can be computed as follows:

$$E_{threshold_{min}} = k * E_{elec} + kE_{amp}d^2 + 8kE_{elec} \quad (2)$$

Assuming the size of data packets is 100 bytes, and the maximum distance is  $5\sqrt{2}$  m, and using the values of  $E_{elec}$ ,  $E_{amp}$  as in [3],  $E_{threshold_{min}}$  for a single data transmission period will be 376  $\mu$ J. Taking into account the energy needed for selecting gateway, building the tree and TDMA schedule, we found empirically from extensive simulation results that 400  $\mu$ J is a good estimate for  $E_{threshold_{min}}$  for a single data transmission period. We mean by good estimate that if the node has a residual energy equal to this  $E_{threshold_{min}}$ , then this energy will be sufficient for the node to participate for a round with a single data transmission period, i.e The node will stay alive to the end of the round. Now, for five and ten Data Transmission periods,  $E_{threshold_{min}}$  will approximately be 2000, and 4000  $\mu$ J respectively. This assumption is a pessimistic one as it simply multiplies the energy needed by one cycle of one data transmission period by 5 or 10 respectively. Of course, in reality it will be less as the first three phases of the process are only done once. On the other hand, we assume the isolated nodes as died nodes. Isolated nodes are the nodes that did not receive a broadcast message to join the tree.

#### 4.5 SIMULATION ASSUMPTIONS

In our simulator, we assume that the control message length is 48 bytes while the data message length is 100 bytes [4]. In addition, we assume perfect aggregation in which a set of data packets are aggregated into one packet. Since we are interested in comparing the different protocols in terms of the energy consumed in transmission and receiving states, we assume that the physical channel is reliable and there is no message loss. The sensor antenna is Omni directional and the nodes are distributed in an open space area where radio coverage is expected to be circular. The circular radio coverage assumption is widely used in literature (see e.g. [3][4]). Regarding the application, we assume that our application is periodic, and the nodes always have data transmit. For GET protocol we assume that the initial  $E_{th}$  which is used in selecting gateway to be 1 J. Moreover,  $E_{th}$  is reduced every cycle by a factor of 0.5 ( $e=0.5$ ). A summary of simulation parameters is shown in Table 4-1

**Table 4-1: Simulation Parameters**

PARAMETER	VALUE
Monitored area dimension	50 * 50 m <sup>2</sup>
Maximum communication distance between two nodes ( $d$ )	$\sqrt{2} * 5$ m
Electronics Energy ( $E_{elec}$ )	50nJ/bit
Amplifier Energy ( $E_{amp}$ )	100pJ/bit/m <sup>2</sup>
Initial Energy in Each Node	2J
Control Packet size	40 bytes
Data Packet size	100 bytes
$E_{th}$ (Single Data Transmission Period)	$4 * 10^{-4}$ J
$E_{th}$ (initial)	1 J
$e$	0.5

## 4.6 COLLECTING SIMULATION RESULTS

For each configuration, we collect a set of measures such as:

- Number of live nodes: we consider a node as a live node if it has enough energy to participate in the successive round and it can communicate with the sink. In other words, the live node is a node that is not isolated and it has energy greater than  $E_{threshold_{min}}$  which is calculated above in section 4.4.
- Throughput: we define throughput as the number of packets delivered to the sink; a definition which is widely used in the context of WSN [25] [4]. The number of packets delivered to the sink in each transmission period is the number of gateways. the number of packets delivered to the sink in each round is the number of gateways multiplied by the number of data transmission periods in the round.
- Total consumed energy: is calculated by summing the energy consumed by all nodes in each phase. This includes the energy consumed by each node due to receiving, transmitting and idle listening.
- Percentage of covered area: is defined as the percentage of the area that is monitored by a live node. To calculate the percentage of covered area, the monitored area is divided into small grids. Each grid's dimension is  $0.20 \times 0.20$  meter. Each grid is considered to be covered if it is within the sensing range of a live node. To calculate the percentage of covered area, the number of covered grids is counted and divided by the total number of grids.



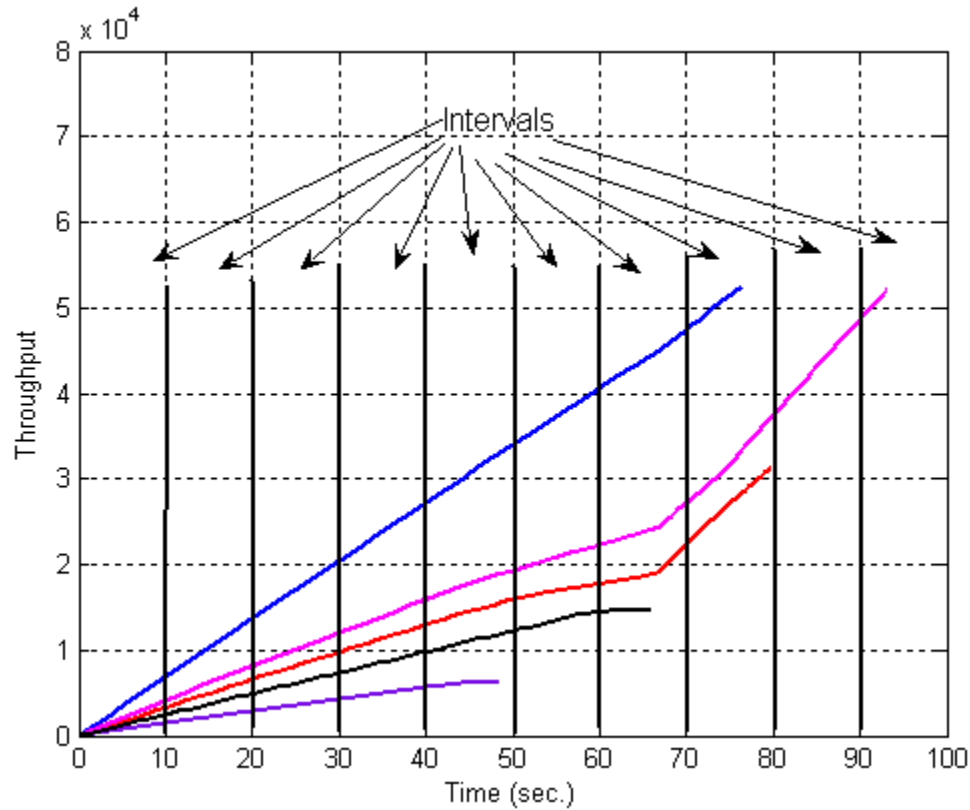
- Delay: is defined as the time interval since an event occurred until the time at which the data packets of that event reach the sink.
- Percentage of detected events: is defined as the number of detected events divided by the total events that occurred during the simulation time. The number of detected events is the total number of events minus the number of missed events. An event is considered to be missed if it occurs while the network is busy with forwarding data packet of the previous event.

Simulation measures are collected every a specific time interval. These results accompanied with the current simulation time are recorded in an output file. As we mentioned above, due to the randomness operation of the nodes, we run our simulator 30 different runs, each run with different seed. For a grid topology, 30 runs are conducted for the same network topology. For the random topology, each run will be conducted using different network topology. For each run, a single output file is generated. A snapshot of an example of output results file is shown in Figure 4-3.

Row	Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8	Col 9
0.	1.95700	100	70	8	0	0.431735	44.901552	0.912789	7
1.	796020	100	630	8	0	4.021400	383.736598	0.912789	7
3.	394770	100	1190	9	0	7.564445	696.885312	0.912789	7
4.	983510	100	1750	9	0	11.106482	1021.886703	0.912789	7
6.	595670	100	2310	10	0	14.709799	1332.702471	0.912789	7
8.	186940	100	2870	13	0	18.271865	1671.925412	0.912789	7
9.	799270	100	3430	11	0	21.813374	1991.553940	0.912789	7
11.	413310	100	3990	10	0	25.379059	2335.789091	0.912789	7
13.	075690	100	4550	9	0	29.036252	2648.902737	0.912789	7
14.	747130	100	5110	10	0	32.734674	2960.010690	0.912789	7
16.	388140	100	5670	11	0	36.327534	3282.668270	0.912789	7
18.	087190	100	6230	11	0	40.020069	3569.113421	0.912789	7
19.	798750	100	6790	12	0	43.696442	3898.362694	0.912789	7
21.	465320	100	7350	10	0	47.311216	4217.291204	0.912789	7
23.	083110	100	7910	9	0	50.940274	4535.618205	0.912789	7
24.	716050	100	8470	9	0	54.564985	4862.201450	0.912789	7
26.	371330	100	9030	10	0	58.217964	5200.316163	0.912789	7
28.	052860	100	9590	9	0	61.923676	5500.853018	0.912789	7
29.	693000	100	10150	10	0	65.589164	5800.761261	0.912789	7
31.	358380	100	10710	11	0	69.232815	6100.036933	0.912789	7
33.	016600	100	11270	8	0	72.865551	6423.773657	0.912789	7
34.	680530	100	11830	11	0	76.496728	6756.044948	0.912789	7
36.	353240	100	12390	13	0	80.142452	7052.675390	0.912789	7
38.	025800	100	12950	12	0	83.836310	7382.153217	0.912789	7
39.	747490	100	13510	11	0	87.622906	7726.666867	0.912789	7
41.	428550	100	14070	11	0	91.296035	8026.909046	0.912789	7
43.	100860	100	14630	9	0	94.971144	8358.362196	0.912789	7
44.	739230	100	15190	9	0	98.554325	8681.998013	0.912789	7
46.	395160	100	15750	13	0	102.197020	8994.151168	0.912789	7
48.	069090	100	16310	9	0	105.928681	9337.562689	0.912789	7
49.	716240	100	16870	11	0	109.575856	9686.981865	0.912789	7
51.	382760	100	17430	11	0	113.229021	9964.453369	0.912789	7
52.	999400	100	17990	11	0	116.780221	10280.884123	0.912789	7
54.	715940	100	18550	13	0	120.527957	10559.107008	0.912789	7
56.	541020	99	19110	15	0	124.545069	10798.681343	0.912789	7
58.	364170	99	19590	13	0	128.525896	11037.425260	0.912618	6
60.	170970	99	20070	14	0	132.445012	11290.398105	0.912618	6
62.	018700	99	20550	13	0	136.422390	11539.060548	0.912618	6
63.	915440	99	21030	15	0	140.447509	11802.273257	0.912618	6
65.	764840	99	21510	10	0	144.460227	12054.652761	0.912618	6
67.	706660	99	21990	13	0	148.667063	12278.978415	0.912618	6
69.	999800	95	22440	18	0	153.353339	12449.543685	0.901706	4

**Figure 4-3: A snap shot of an example of output results file**

The results obtained by the 30 runs are averaged. The average results are stored in a single output file. A snap shot of an example of an average output file is shown in Figure 4-3. To calculate the average, we divide the simulation time into intervals, and then all the measures within a specific interval are averaged into a single value. For example, Figure 4-4 shows the throughput versus time for 5 different runs. We consider 5 runs in this example for illustration. As we mentioned above in real calculations we consider 30 different runs. In Figure 4-4, the simulation time is divided into 10 intervals. Each interval's length is 10 seconds. All the throughput values within each interval are averaged into one value.

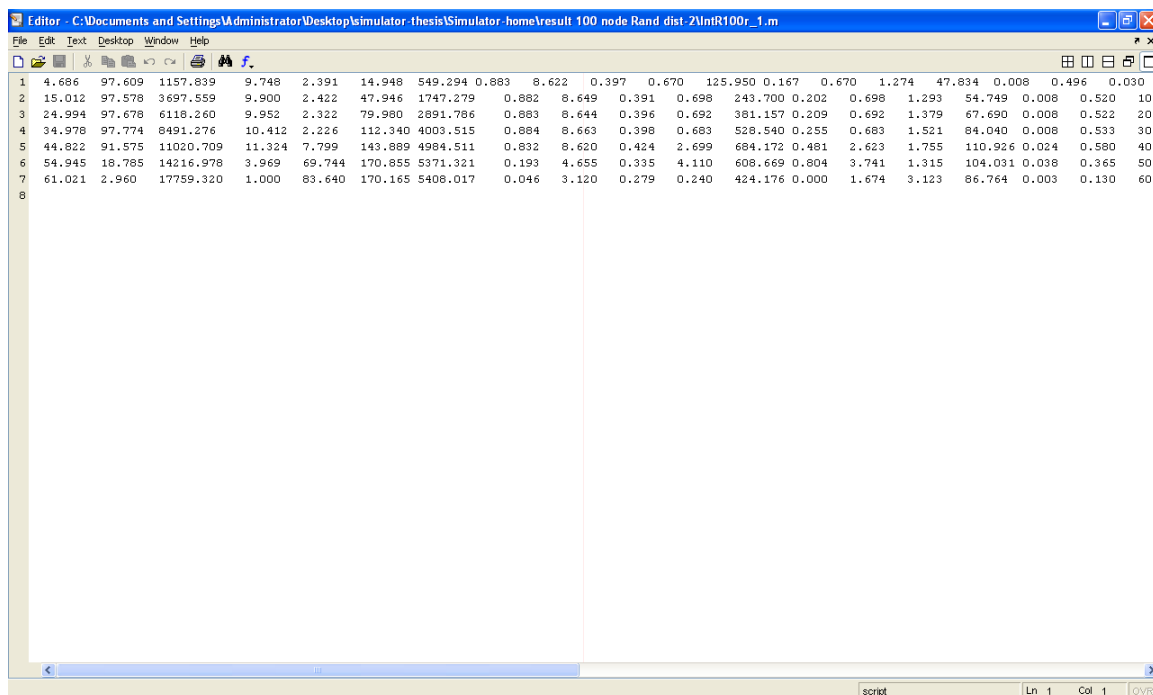


**Figure 4-4: Example of Intervals used in average calculation**

We calculate the confidence interval for the average. Confidence interval is calculated by

$$\text{Confid. Interval} = \left[ \bar{x} - Z_{\frac{\alpha}{2}} * \frac{S}{\sqrt{n}}, \bar{x} + Z_{\frac{\alpha}{2}} * \frac{S}{\sqrt{n}} \right] \quad (3)$$

Where  $\bar{x}$  and  $S$  are the mean and the standard deviation of the random samples of size  $n$ .  $Z_{\frac{\alpha}{2}}$  is the  $Z$  value with  $\nu=n-1$  degrees of freedom, and  $\alpha$  equal to (*confidence level -1*). In our calculations, we consider the confidence level to be 95%, therefore using normal distribution tables,  $Z_{\frac{\alpha}{2}}$  is 1.96. With considering confidence level of 0.95 the error bound of the obtained results range from 2% to 3%.



The image shows a screenshot of a text editor window titled "Editor - C:\Documents and Settings\Administrator\Desktop\simulator-thesis\Simulator-home\result 100 node Rand dist-2\IntR100r\_1.m". The window contains a table of numerical data with 8 rows and 20 columns. The data is as follows:

1	4.686	97.609	1157.839	9.748	2.391	14.948	549.294	0.883	8.622	0.397	0.670	125.950	0.167	0.670	1.274	47.834	0.008	0.496	0.030	
2	15.012	97.578	3697.559	9.900	2.422	47.946	1747.279	0.882	8.649	0.391	0.698	243.700	0.202	0.698	1.293	54.749	0.008	0.520	10.	
3	24.994	97.678	6118.260	9.952	2.322	79.980	2891.786	0.883	8.644	0.396	0.692	381.157	0.209	0.692	1.379	67.690	0.008	0.522	20.	
4	34.978	97.774	8491.276	10.412	2.226	112.340	4003.515	0.884	8.663	0.398	0.683	528.540	0.255	0.683	1.521	84.040	0.008	0.533	30.	
5	44.822	91.575	11020.709	11.324	7.799	143.889	4984.511	0.832	8.620	0.424	2.699	684.172	0.481	2.623	1.755	110.926	0.024	0.580	40.	
6	54.945	18.785	14216.978	3.969	69.744	170.855	5371.321	0.193	4.655	0.335	4.110	608.669	0.804	3.741	1.315	104.031	0.038	0.365	50.	
7	61.021	2.960	17759.320	1.000	83.640	170.165	5408.017	0.046	3.120	0.279	0.240	424.176	0.000	1.674	3.123	86.764	0.003	0.130	60.	
8																				

**Figure 4-5: A snap shot of an example of an average output file**

# Chapter Five

## PERFORMANCE EVALUATION

### 5.1 INTRODUCTION

In the last chapter, we presented the simulation setup that is used in performance evaluation of our proposals. In this chapter, we will discuss the performance evaluation of the GET, EEDS, and EAD<sub>General</sub>. A comparison between these protocols and the well-known protocols EAD and LEACH will be presented.

We compare all protocols in terms of network lifetime, throughput and total consumed energy. Network lifetime is defined as the time at which tree can be built from nodes toward sink. While throughput is defined as the total number of data packets delivered to the sink. We investigate the performance of the three protocols with different data Transmission periods (1, 5 , 10) in a single round. We assume the network topology shown in Figure 4-2, simulation parameters presented in Table 4-1 are assumed.

The performance evaluation of GET, EEDS, and EAD<sub>General</sub> will be discussed in sections: 5.2 , 5.3, and 5.4 respectively. Finally, we conclude this chapter with section 5.5

## 5.2 PERFORMANCE EVALUATION OF GET

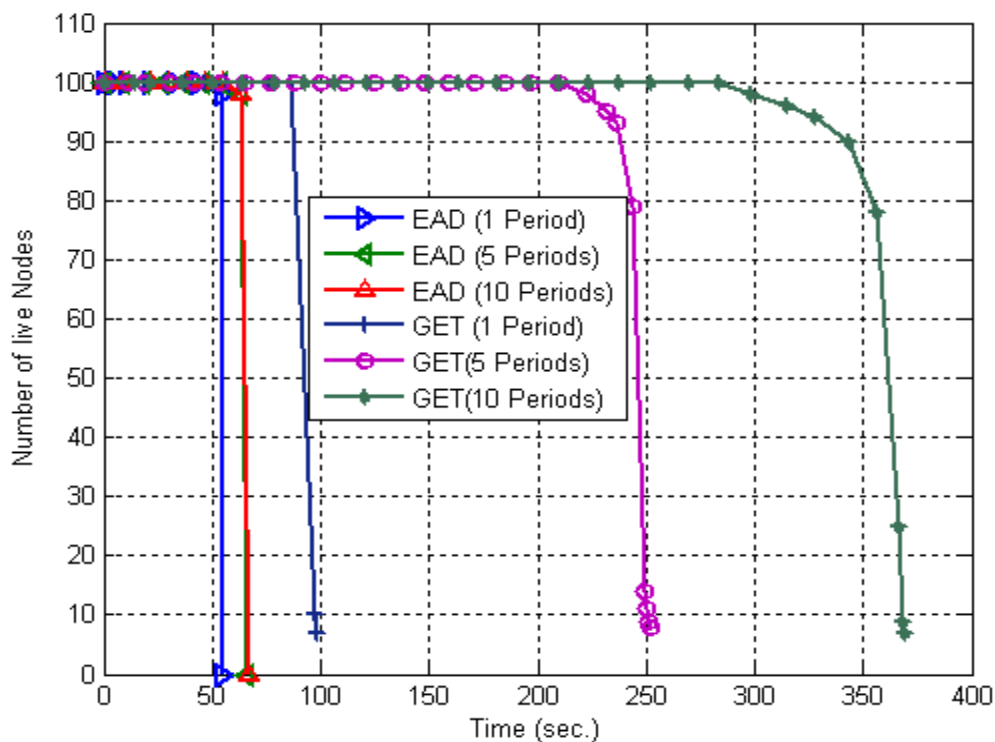
In this section, we compare the performance of GET with the performance of both LEACH and EAD. A comparison between GET and EAD will be presented in section 5.2.1. Then in section 5.2.2, we present a comparison between EEDS and LEACH.

### 5.2.1 A COMPARISON BETWEEN GET AND EAD:-

GET is similar to EAD in the sense a tree rooted at the sink is used to forward data from sensor nodes to the sink. However, it differs from EAD in building the tree. In GET, building the tree is more general. Any node can act as a gateway node. Another important difference between the two protocols is in the data transmission phase; in EAD, CSMA mechanism is used for data transmission, while in GET, a TDMA schedule is used for the data transmission. In this section, we will discuss the performance evaluation of GET and EAD.

Figure 5-1 shows a comparison between GET and EAD in terms of number of live nodes for different data transmission period. We can observe that GET outperforms EAD in all cases. For example, with 1 data transmission period scenario, the network lifetime for GET is about 97.9 seconds, while it is about 55 seconds in EAD. The network lifetime is improved in GET by 78%. On the other hand, increasing the number of data transmission periods for a single tree and for the corresponding schedule will increase the network lifetime. This is very intuitive since selecting gateways, building the tree, and building the schedule will consume energy. Using the same tree for multiple consecutive data transmission periods will save more energy. When considering 10 data transmission

periods for every round, the network lifetime in GET is about 370 seconds, while it is about 66.5 seconds in EAD. In other words, GET shows three folds increase in network lifetime compared to 1 data transmission scenario, while only 18% improvement has been achieved under EAD approach. It is very important to note that these network lifetime figures are functions of the data-sensing period that is considered very small in our simulation experiments. For example, considering the 10-data transmission periods example, the data-sensing period is 32.80 msec.



**Figure 5-1: Number of Live Nodes vs. Time, (GET, EAD)**

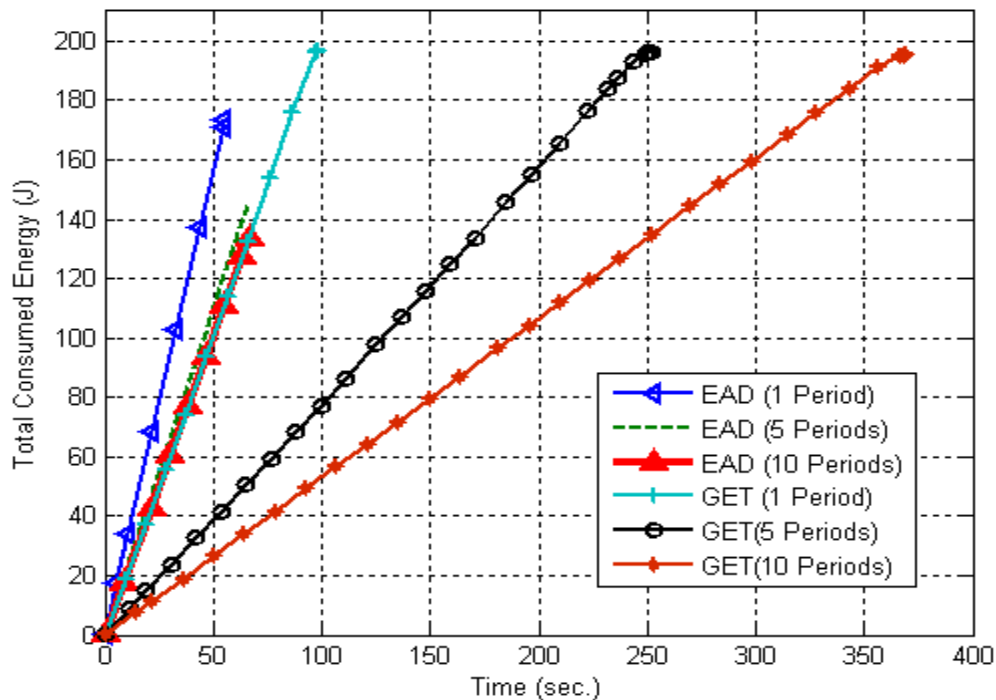
In both protocols, the number of live nodes is stable around 100 then starts going down. This occurs when the gateway nodes (nodes that communicate directly with the sink) start to die. In EAD, since the gateway nodes will wait until it receives all the data packets from all leaf nodes, they will transmit their data packets to the sink at the end of the data transmission period, so they have to be ON for the whole data transmission

period. This will consume a lot of energy; therefore, they will die very early. At the same time, the other nodes will be awake up until they send their data packet then they will go to sleep. They will consume less energy compared with gateways. On the other hand, on GET, the gateway nodes will sleep for most of the time except during their scheduled time to receive and transmit data. They will be ON at the end of data transmission phase. Their energy consumption will be smaller compared with energy consumption in EAD, which will increase their lifetime. This is the case for all nodes; gateway and non-gateway nodes. Moreover, we can observe from Figure 5-1 that the number of live nodes decreases faster in EAD while it decreases smoothly in GET. In EAD, the closest nodes to the sink limit the number of gateway nodes. When these nodes die most of the remaining nodes will be isolated since they will not be able to communicate with the sink. The gateway nodes in the EAD will die approximately at the same time since they all act as gateway nodes for the whole network lifetime. Therefore, the number of live nodes that can communicate with the sink will decrease very fast in EAD. On the other hand, the number of gateway nodes in GET is not limited by those nodes which are close to sink; any node can be a gateway. When the closest nodes to the sink die, since we assume all nodes can communicate with the sink directly, other nodes on the network will be selected as gateways. Therefore, there will not be isolated nodes. The remaining nodes will be able to communicate with the sink until they consume all their energy. Therefore, the number of live nodes in GET will decrease smoothly compared with EAD as shown in Figure 5-1

Figure 5-2 shows a comparison between GET and EAD in terms of the total consumed energy for different data transmission period. We observe that at any time



instant, the total consumed energy by EAD is greater than the total consumed energy by GET. For example, with 5 data transmission periods, the total consumed energy in GET at  $t=50$  seconds is 38.5 J, while it is about 109 J in EAD. The consumed energy is dropped by 65%. The consumed energy in EAD is higher compared with GET, since a CSMA mechanism is used for data transmission in EAD; the sensor node has to be ON for all the data transmission phase. On the other hand, the sensor node in GET will be only ON for receiving and transmitting slots, which will reduce the consumed energy in each node considerably.



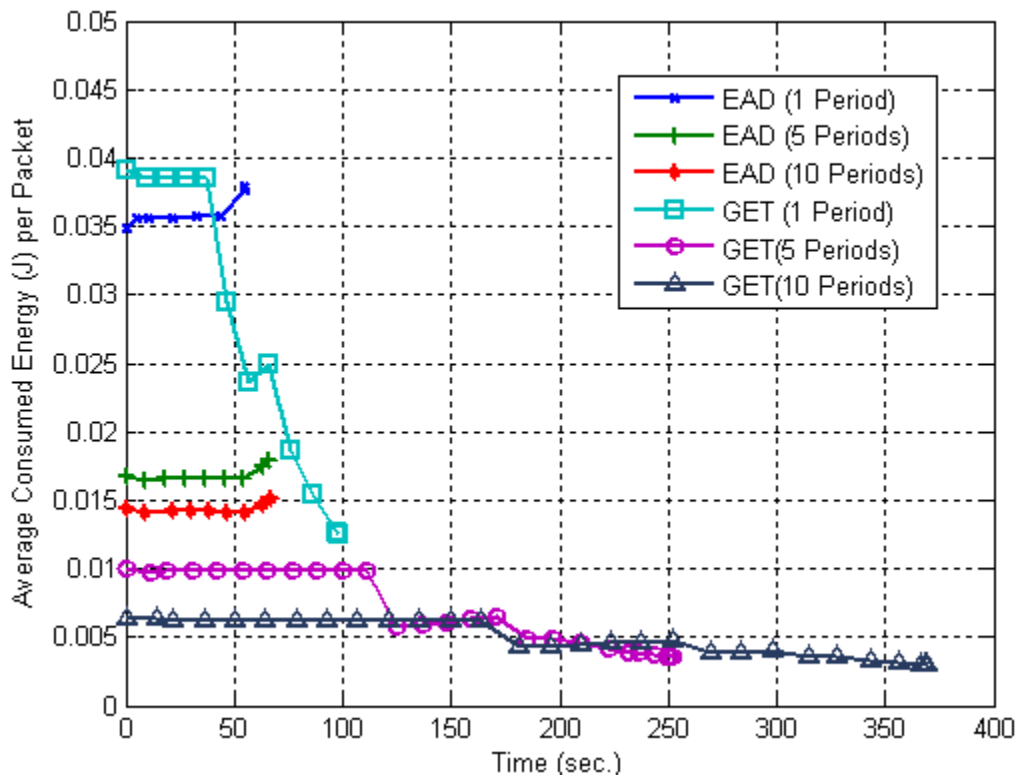
**Figure 5-2: Total Consumed Energy vs. Time, (GET, EAD)**

For the entire simulation time, the total consumed energy in EAD is less than 180 J for all data transmission periods. Since we have 100 sensor nodes and each node has initially 2 J. The total initial energy in the whole network is 200 J. There is more than 20

J of non-consumed energy in the network. Hence, 10% of the initial energy in the network is not utilized. This non-consume energy is due to isolated nodes in the network. This is another way where EAD shows ineffectiveness in utilizing energy. On the other hand, the total consumed energy in GET reaches approximately 200 J for all data transmission periods. The overall initial energy in the network is consumed, i.e. all the nodes are fully utilized. As we discussed above, since any node in the network can be a gateway in GET, there will not be isolated nodes in the network. A node will continue working until its whole energy is depleted.

Figure 5-3 shows a comparison between GET and EAD in terms of the average consumed energy per a packet. We observe that under one data transmission period, EAD shows better performance than GET. On the other hand, under five and ten data transmission periods, GET is better than EAD. For both protocols, under one data transmission period, the tree is utilized for once only. There is one setup phase in EAD and three setup phases in GET. In other words, the energy consumed in setup phases in GET is greater than in EAD. Therefore, in GET, the energy consumed in setup phases is large compared with energy saved in the data transmission phase. While in EAD, the energy consumed in setup phases is comparable to energy consumed in data transmission phase. Therefore, EAD outperforms GET under one data transmission periods. On the other hand, under five and ten data transmission periods, in EAD, the same tree is utilized multiple times, and in GET, the tree and the schedule are utilized many times. Since the energy consumed by each node in GET in data transmission phase is less than in EAD, the saved energy in GET is large compared with the overhead energy, and the saved energy in EAD is small compared with the overhead energy. Therefore GET outperforms

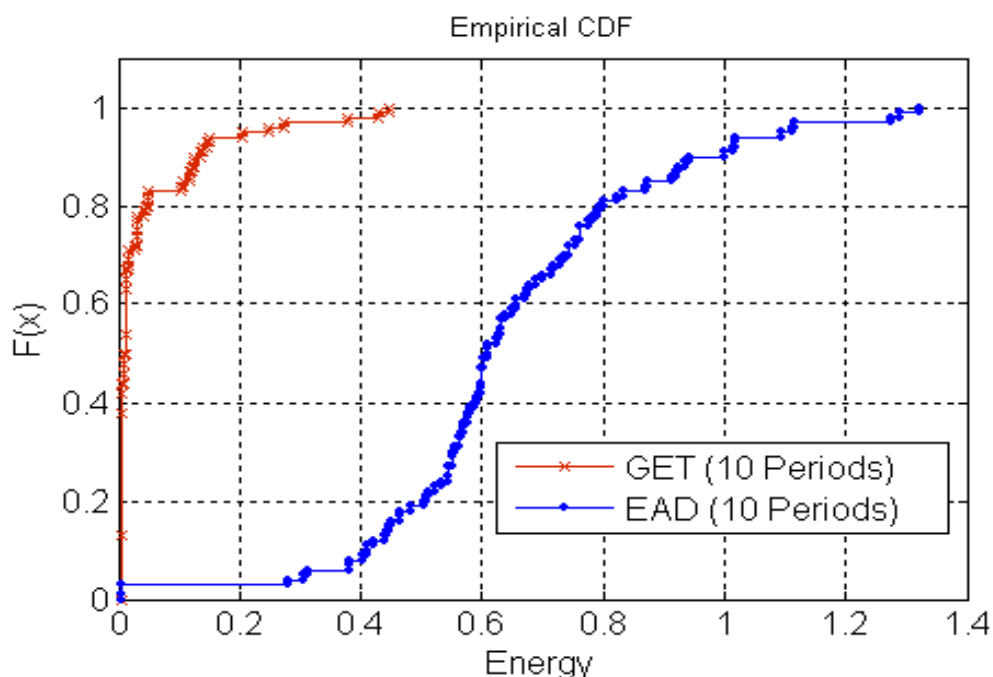
EAD in terms of average consumed energy per packet under five and ten data transmission periods.



**Figure 5-3: Average Consumed Energy per a single packet, (GET, EAD)**

**Statistical Analysis For The Consumed Energy:** To quantify the statistical characteristics of the energy consumption at the end of network lifetime, we compute the distribution of the residual energy in all nodes at the end of simulation for each protocol. Figure 5-4 shows the distribution of the residual energy in all nodes at the end of the simulation for GET and EAD for 10 data transmission periods. We observed that most of the nodes in GET consumed all of their initial energy, while in EAD; few nodes consumed all of their initial energy. For example, the residual energy in 50% of the nodes in GET is about zero J. While in EAD, only about 4% of the nodes has approximately 0 J

residual energy, i.e., 96% of the nodes still have unconsumed energy. These nodes are isolated and cannot reach the sink since the gateway nodes are died. Moreover we observed from Figure 5-4 that in EAD, about 60% of the nodes still have more than 0.6 J, i.e., 60% of the nodes still have more than 30% of their initial energy, on the other hand, in GET, only 4% of the nodes still have more than 0.4 J.



**Figure 5-4: The Cumulative Distribution Function of the residual energy in the nodes at the end of network lifetime (10 Data Transmission Period)**

**Table 5-1: A comparison between GET and EAD in terms of throughput**

	GET	EAD	Improvement in GET
1 Data Transmission period	16834	4723	256.4%
5 Data Transmission periods	54950	8085	579.6%
10 Data Transmission periods	65100	8810	638.9%

Increasing network life time in GET increases the total number of data packets delivered to sink (throughput). Table 5-1 shows a comparison between GET and EAD in terms of throughput. It can be clearly noticed that GET outperforms EAD for all data

transmission periods. For example, for 5 data transmission period the total throughput is improved by 579.6% (from 8085 to 54950), while for 10 data transmission periods it is improved by about 638.9% (from 8810 to 65100 packets).

This huge improvement in the total throughput can be attributed into two reasons. First, the increase in network lifetime will ensure more influx of data packets, which can be considered a true enhancement to the overall throughput. Second, with GET, the sink might receive data packets from larger number of the gateways compared to EAD.

To investigate this issue, we shall study the structure of the constructed trees. The larger number of gateways in the network will decrease the height (i.e. depth) of the tree. Decreasing the tree's height will decrease the aggregation ratio (N:1) and eventually increase the data redundancy in the packets delivered to the sink. For example, in a tree with height 10, 10 data packets will be aggregated into 1 data packet. On the other hand, in a tree with height 13, 13 data packets will be aggregated into 1 data packet. Table 5-2 shows some statistics for tree height in the network. We can notice that for all data transmission periods, the mean of trees height in GET is less than the mean in EAD. For example, in the GET with 5 data transmission periods the mean of tree height is 11.78 while it is 13.87 in the EAD. In the GET, in average, 11.78 data packets will be aggregated into 1 data packets. While 13.87 data packets will be delivered into 1 data packets in EAD.

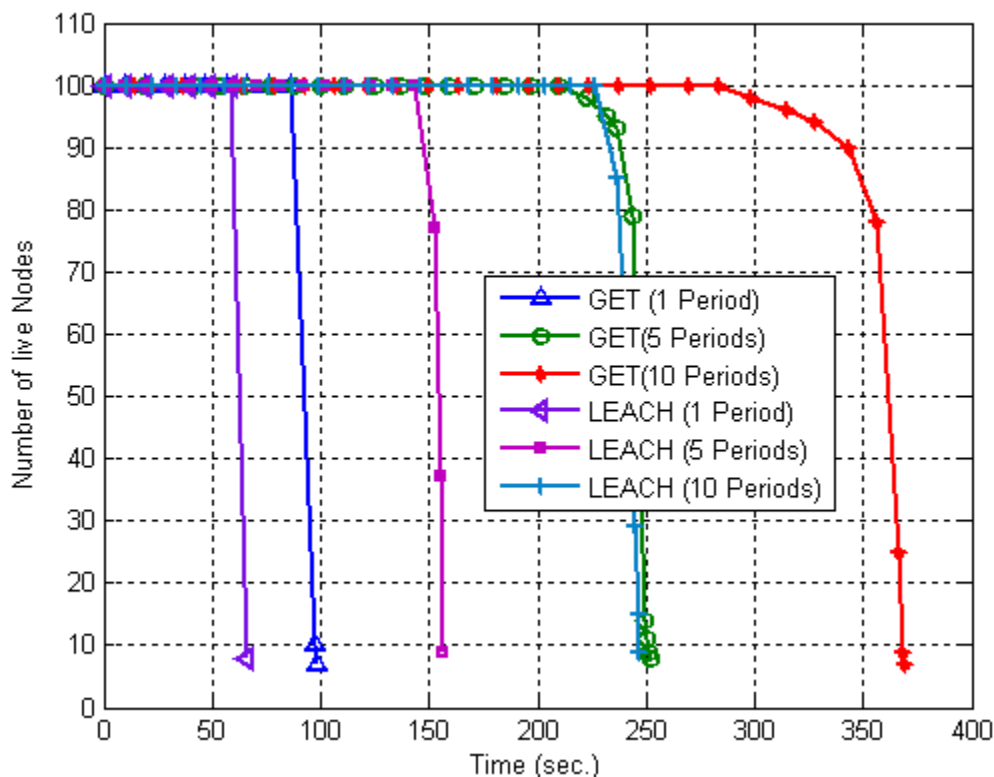
**Table 5-2: Statistics for Tree Hight**

	Minimum	Maximum	Mean	Median	Std	Range
EAD(1 Period)	11	16	13.36	13	1.567	5
EAD(5 Periods)	12	17	13.87	13	1.457	5
EAD(10 Periods)	12	16	13.31	13	1.401	4
GET(1 Period)	3	16	11.3	13	4.269	13
GET(5 Periods)	4	18	11.78	13	3.194	14
GET (10 Periods)	4	17	12.65	13	2.496	13

### 5.2.2 A COMPARISON BETWEEN GET AND LEACH

Now we turn our discussion toward the comparison between GET and LEACH. GET likes LEACH in which any node can communicate directly with the sink, and the data transmission within the cluster is scheduled-based. However, there are two major differences in GET. First, in GET, a structured tree is connecting nodes to the sink, while in LEACH; clusters are constructed such that nodes communicate with sink via heads. Second, the data transmission between gateways and sink is performed using TDMA access mechanism rather than CSMA mechanism. In addition, the gateway adaptively controls its transmission power to reach nearest active node. For LEACH, based on [3], we compute the optimal number clusters for our network configuration and we found it to be 4.

Figure 5-5 shows a comparison between GET and LEACH in terms of number of live nodes. We observe that GET outperforms LEACH for all scenarios. For instance, consider the scenario of 10 data transmission periods, the network lifetime for GET is 370 seconds while it is about 247 seconds for LEACH. An improvement by 49.8% is achieved in GET. Increasing network lifetime in GET is due to minimizing energy consumption in each sensor node.



**Figure 5-5: Number of Live Nodes vs. Time, (LEACH, and GET)**

To compare the energy consumed by a node in LEACH and GET, we note that in LEACH there are two types of nodes; head and non-head nodes. While in GET, there are three types of nodes: leaf, non-leaf, and gateway node. We have to compare the energy consumed by all nodes in transmitting, receiving and idle-listening state. As discussed above, in building the tree phase of GET, the nodes transmit their signals with minimum energy. Therefore, a parent of any node in GET will be one of its closest neighbors. On the other hand, in LEACH, each node can hear all other nodes in the network. Therefore, a parent of a node can be any node in the network.

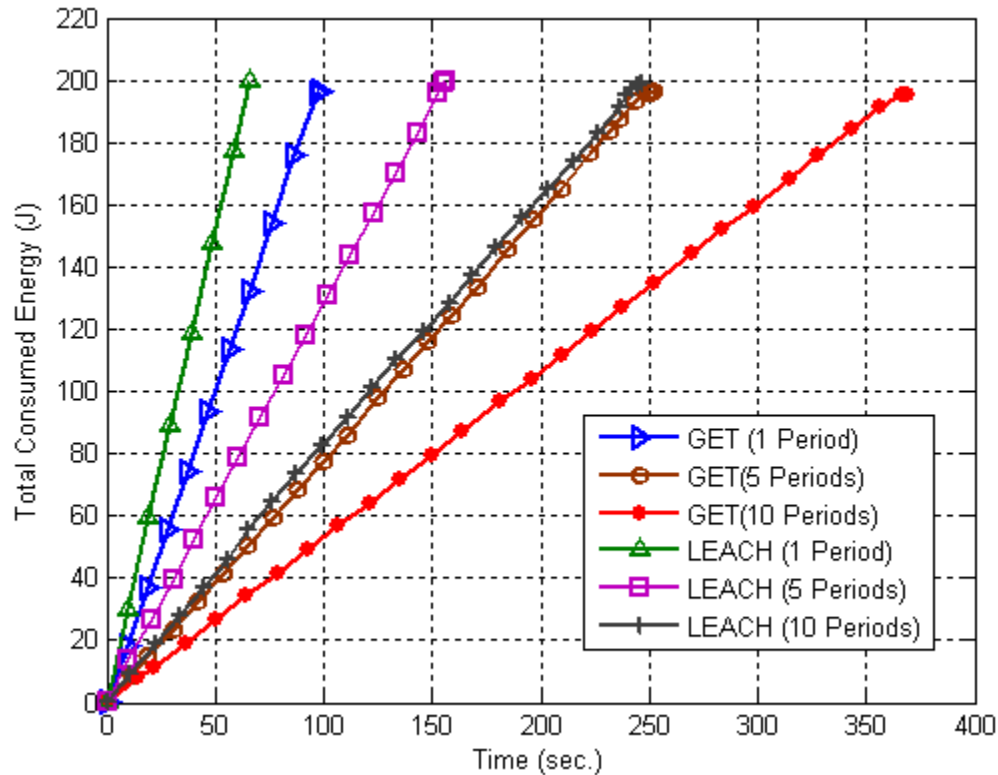
To compare the energy consumed by leaf node in GET and non-head node in LEACH, we note that both types of nodes will not lose energy due to idle listening or due

to receiving data. They will not lose energy due to receiving data since they are not parent nodes, and they will not receive data from other nodes. In addition, they will not lose energy due to idle listening since a TDMA transmission scheme is implemented in each cluster in LEACH and in the tree of GET. A node will be ON at its scheduled time only. Regarding the energy consumed in transmitting, a leaf node in GET will consume less energy than a non-head node in LEACH, because in GET the parent of a leaf node is always one of the nearest neighbors of the node. Therefore, the transmission distance will be very short, and the transmission energy will be smaller. While in LEACH, the parent of non-head node may be any head in the network, since it is assumed that all nodes will hear each other. As any node in the network can advertise itself as a head, and since heads are selected randomly, in some cases, all heads may be far away from a node. Although the parent of a node is the closest head, it may not be one of the nearest neighbors. Consequently, the transmission distance will be longer, and the transmission energy must be greater. Therefore, in data transmission phase, a non-head node in LEACH will lose more energy compared with a leaf node in GET.

To compare the energy consumed by a non-leaf node in GET and by a head node in LEACH, we note that non-leaf nodes in GET will not lose energy due to idle listening since they are scheduled to receive and transmit data at specific time slots. They will be ON at these time slots only, and they will be OFF for other time slots. On the other hand, a head in LEACH will use CSMA to transmit data to the sink. It has to be ON all the time and sense the channel to get its turn for transmission. Therefore, it will lose energy due to idle listening. Regarding the energy consumed during data transmission, a parent of non-leaf node in GET will always be one of the closest neighbors, while the parent of a head



in LEACH will be the sink, which is far away from the head. Therefore, a head node in LEACH will transmit data for longer distance compared with a non-leaf node in GET. LEACH will transmit data for longer distance compared with a non-leaf node in GET. Further, a head node will consume larger energy during transmission than the non-leaf node.

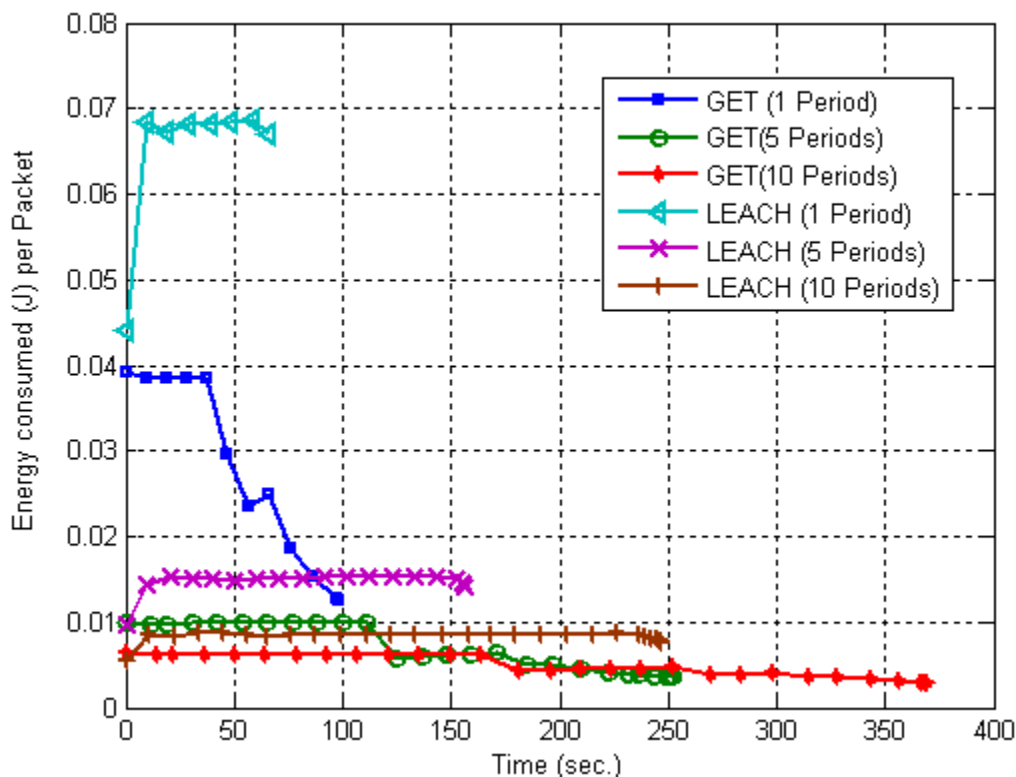


**Figure 5-6: Total Consumed energy vs. Time, (LEACH, GET)**

Regarding the consumed energy during receiving mode, a non-leaf node in GET will consume smaller receiving energy compared with a head node in LEACH. A non-leaf node in GET will have smaller number of children compared to the number of children associated with a head in LEACH. As a result, the total consumed energy by a non-leaf node in GET is less than the total consumed energy by a head in LEACH. Finally, considering the energy consumed by a gateway in GET and a head in LEACH,

we note that both nodes will transmit their data to the sink that might be far away from them. In average, both of them will consume comparable amount of energy in transmitting data. On the other hand, a gateway in GET will consume less energy in receiving data because it has less number of children compared with a head in LEACH. In GET, only the closest neighbors of a gateway can hear its transmission, therefore only the closest neighbor can be a child for a gateway. On the other hand, in LEACH, it is assumed that all nodes can hear the transmission of other nodes, any node in the network can be a child of a specific head. In addition, comparing with the head in LEACH, a gateway in GET will consume less energy due to lacking of overhearing and idle-listening modes. Gateway nodes in GET will be ON only in their time slots to receive and transmit data, while heads in LEACH will be ON all the time.

It should be clear from the above discussion that any node in GET will consume less energy compared with a node in LEACH. Therefore, the total consumed energy in GET will be smaller compared with LEACH as shown in Figure 5-6. For example, with 10 data transmission periods, at  $t=150$  seconds, the total consumed energy in LEACH is about 121 J while it is about 80 J in GET. The consumed energy is dropped by about 30%. In both protocols, the overall consumed energy by the end of the network lifetime is about 200 J, which is the initial energy in the network. Therefore, Energy utilization in both protocols is very efficient in using the whole energy. Nevertheless, the question that will be raised is: which protocol achieves higher throughput and longer lifetime for the same initial energy?



**Figure 5-7: Average Consumed Energy per a single packet , (GET,LEACH)**

Figure 5-7 shows the average consumed energy per a single packet for GET and LEACH protocols. We observe that under one data transmission period, the average consumed energy per a single packet in LEACH is less than in GET. For a single data transmission period, there is more overhead energy in GET protocol. In GET protocol, with single data transmission phase, there are three setup phases in which no data packets are delivered to the sink. The energy consumed in these phases is considered as overhead energy. The overhead energy is more than the energy saved in data transmission phase. On the other hand, in LEACH protocol, in addition to the single data transmission phase, there is a single phase in which no data packets are delivered to the sink. In LEACH, the energy saved in the data transmission phase is more than the overhead energy. For five

and ten data transmission periods, the average consumed energy per a packet in GET is less than in LEACH. In GET protocol, an overhead energy is consumed in setup phases. On the other hand, energy is saved in data transmission phase. The energy saved in the data transmission phase is more than the energy consumed in setup phases.

Finally, Table 5-3 shows a comparison between GET and LEACH in terms of the number of data packets delivered to the sink. For 5 data transmission periods, the throughput in the GET is improved by 291% (from 14045 to 54950 packets). Increasing the network lifetime will increase the number of data packets delivered to the sink.

**Table 5-3: A comparison between GET and LEACH in terms of throughput**

	GET	LEACH	Improvement in GET
1 Data Transmission period	16834	3407	394.1%
5 Data Transmission periods	54950	14045	291%
10 Data Transmission periods	65100	26640	144%

### 5.3 PERFORMANCE EVALUATION OF EEDS

In this section, we compare the performance of EEDS with the performance of both LEACH and EAD. A comparison between EEDS and EAD will be presented in section 5.3.1. Then in section 5.3.2, we present a comparison between EEDS and LEACH.

#### 5.3.1 A COMPARISON BETWEEN EEDS AND EAD

Figure 5-8 shows the number of live nodes per time for both EAD and EEDS for different transmission period. We note that EEDS outperforms EAD in terms of the network lifetime. The time for the first node to die in EEDS is higher than the time for the first node to die in EAD. In EEDS, assuming one data transmission period, the time

for the first node to die is about 87 seconds, while it is about 50 seconds when applying EAD. An improvement of about 74% is achieved. On the other hand, increasing the number of data transmission period for a single tree and for the corresponding schedule will improve the time for the first node to die. This is very intuitive since building the tree and building the schedule will consume energy. Using the tree for multiple data transmission period will save energy. When using 10 consecutive data transmission periods, the time for the first node to die in EEDS is about 278 seconds, while it is about 60 second in EAD.

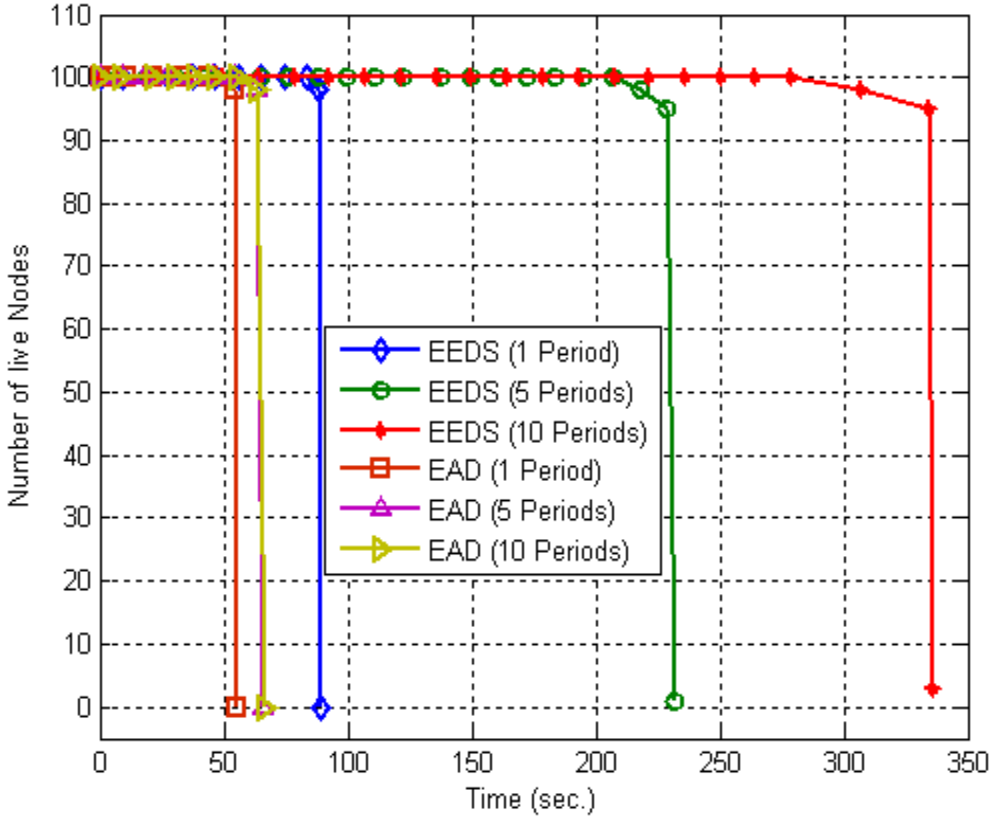


Figure 5-8: Number of live Nodes vs. Time, (EEDS, EAD)

It is very important to note that these network lifetime figures are functions of the data-sensing period that is considered very small in our simulation experiments. For

example, considering the 10-data transmission periods example, the data-sensing period is 32.80 msec.

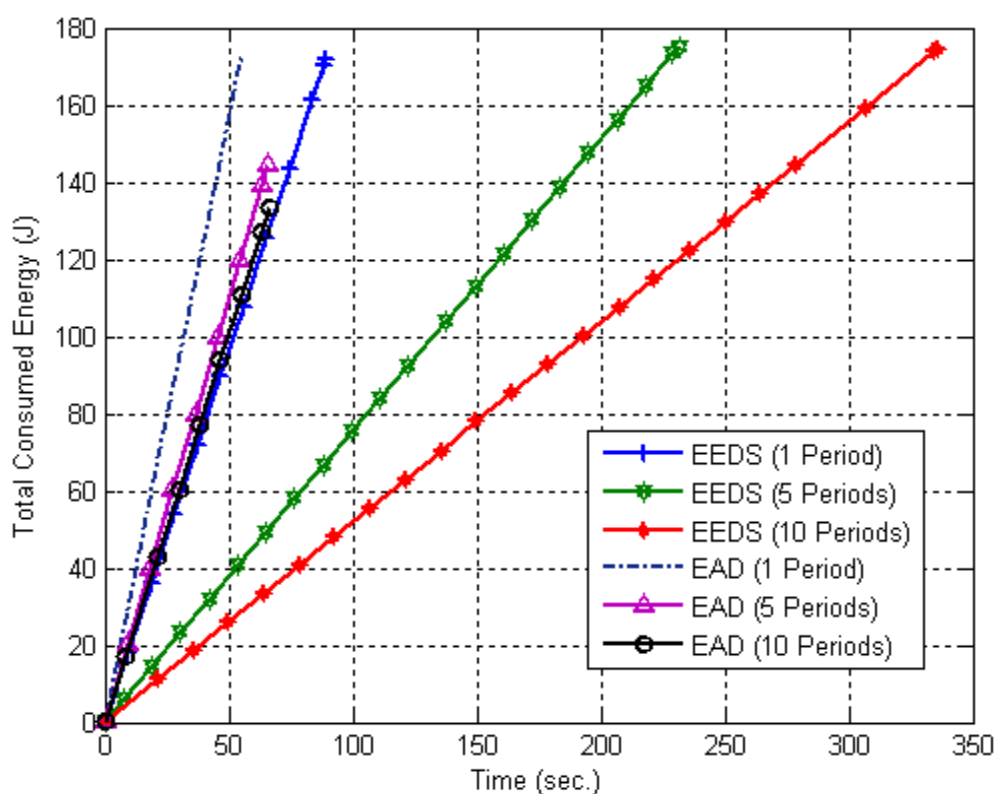
In EAD, since the gateway nodes will wait until it receives all the data packets from the leaf nodes, they will transmit their data packets to the sink at the end of the data transmission period, so their radios transceivers have to be ON for the whole data transmission period. This will consume a lot of energy; therefore, they will die very early. Since the application considered here is periodic, other nodes will be ON until they transmit their data packet then they will go to sleep until the successive event occurs. The gateway nodes will die earlier than other nodes. On the other hand, on EEDS, the nodes will sleep for most of the time except during their scheduled time slots to receive and transmit. Their energy consumption will be smaller, compared with energy consumption in gateway nodes in EAD, which will increase their lifetime. This is the case for all nodes; gateway and non-gateway nodes. In both protocols, when the gateway nodes die, most of the nodes will be isolated since they will not be able to communicate with the sink. As mentioned earlier, the nodes that act as gateways in both protocols are limited to the closest existing nodes to the sink. These nodes act as gateways for the whole network lifetime and they die approximately at the same time. Therefore, in both protocols, the number of live nodes is stable around 100 then goes down very fast.

**Table 5-4: The Aggregated Throughput in EEDS compared with EAD protocol**

Data Transmission Period	EEDS	EAD	Improvement
1	4569	4561	0.17%
5	18180	8085	124.9%
10	28350	8810	221.8%

The improvement in the network lifetime in EEDS will improve the number of data packets delivered to the sink. Table 5-4 shows the number of data packets

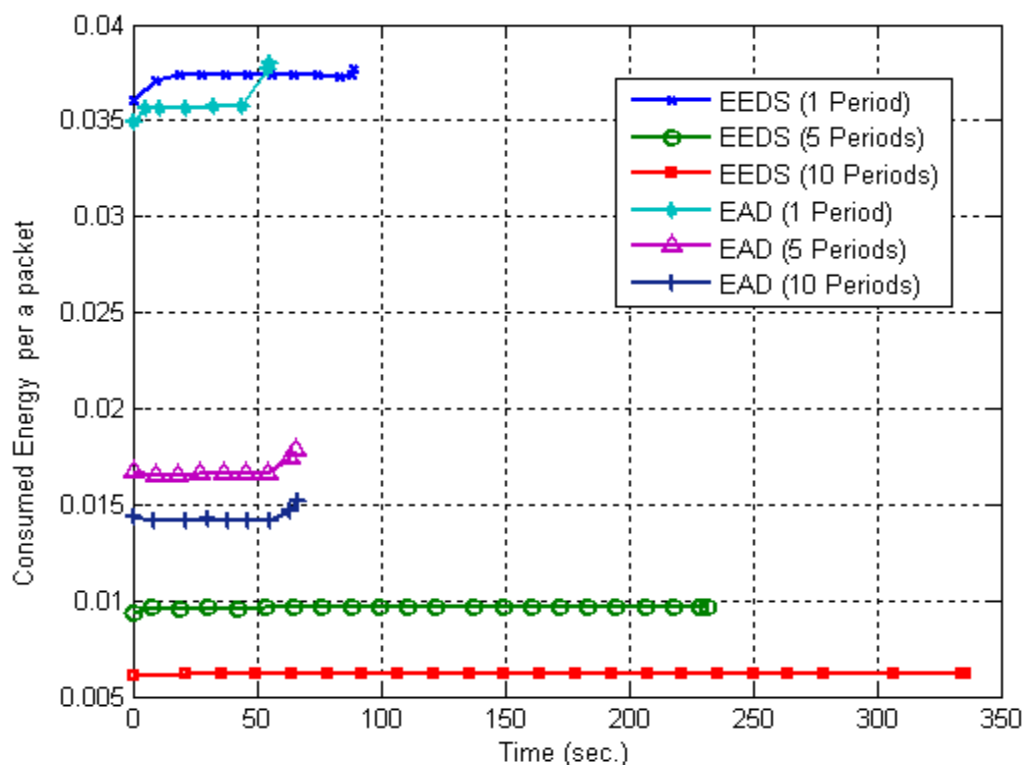
(throughput) that are delivered to the sink for both *EAD* and *EEDS*. Since the network lifetime in *EEDS* is longer than *EAD*, the sink will be able to receive more data packets from gateways for longer time. Therefore, the aggregated throughput in *EEDS* is higher than the aggregated throughput in *EAD*. For example, with five data transmission periods, the aggregated throughput in *EEDS* is improved by about 124.9%. These results show the efficient of *EEDS* in saving energy of WSN.



**Figure 5-9: Total consumed Energy vs. Time , (EEDS, EAD)**

Figure 5-9 shows the total consumed energy versus time in the network for both *EAD* and *EEDS*. It is clear that the total consumed energy in *EEDS* is much less than the energy consumed in *EAD*. For example, at  $t=50$  seconds, assuming one data transmission period, the total consumed energy in *EEDS* is about 100.4 J while it is about 154.5 in

EAD. We have less energy consumption by 34.3%. Assuming five and ten data transmission periods, we have less energy consumption by 65.6% and 74.5 % respectively. Since the gateways in EEDS stay alive for longer time, the remaining nodes will be able to communicate with the sink for longer time. More energy will be utilized from these nodes before they become isolated nodes. Therefore, the overall total consumed energy in EEDS is increased with an improvement in the throughput as shown above.



**Figure 5-10: Average Consumed Energy per a single packet, (EEDS, EAD)**

Figure 5-10 shows a comparison between EEDS and EAD in terms of the average consumed energy per a packet. We observe that under one data transmission period, EAD outperforms EEDS, while under five and ten data transmission periods, EEDS

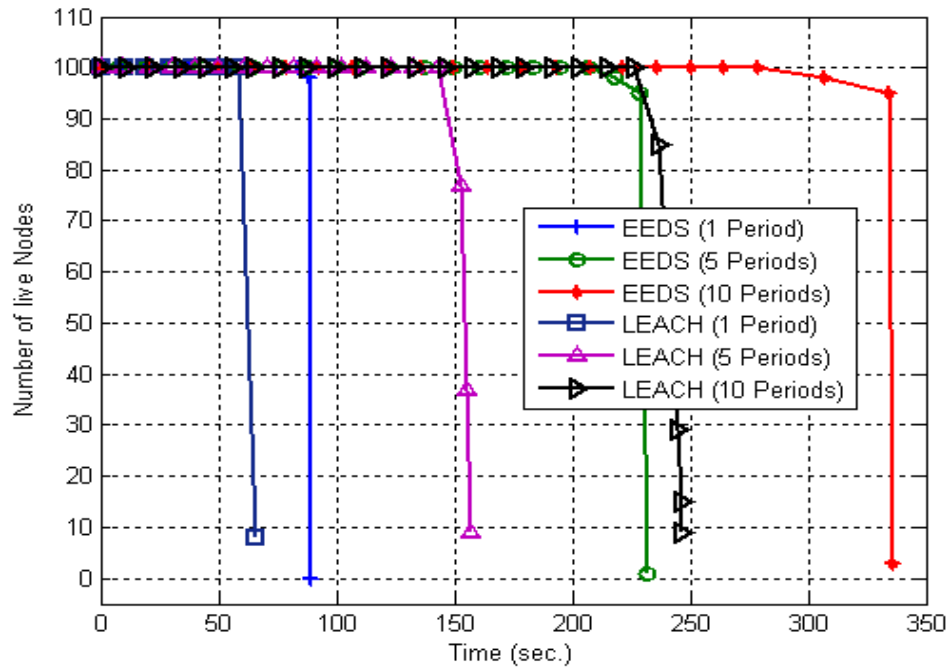


outperforms EAD. The justification is similar to the justification of the comparison of GET and EAD.

### 5.3.2 A COMPARISON BETWEEN EEDS AND LEACH

EEDS differs from LEACH in which a tree is connected between nodes and the sink rather than sets of clusters connected to the sink via heads. In addition, the data transmission between gateways and sink is performed using TDMA schedule rather than CSMA mechanism. In this subsection, we compare the performance of EEDS with the LEACH. For LEACH, we use the optimal number of clusters as in [3] for our network configuration, which is 4.

Figure 5-11 shows a comparison between EEDS and LEACH for the number of live nodes as time progresses. We observed that EEDS outperforms LEACH for all data transmission periods, for 10 data transmission periods, the network lifetime for EEDS is 330.7 seconds while it is about 246.7 seconds for LEACH. An improvement by 34.0% is achieved in EEDS. The improvement in the network lifetime is due to the saving in energy in each node in EEDS

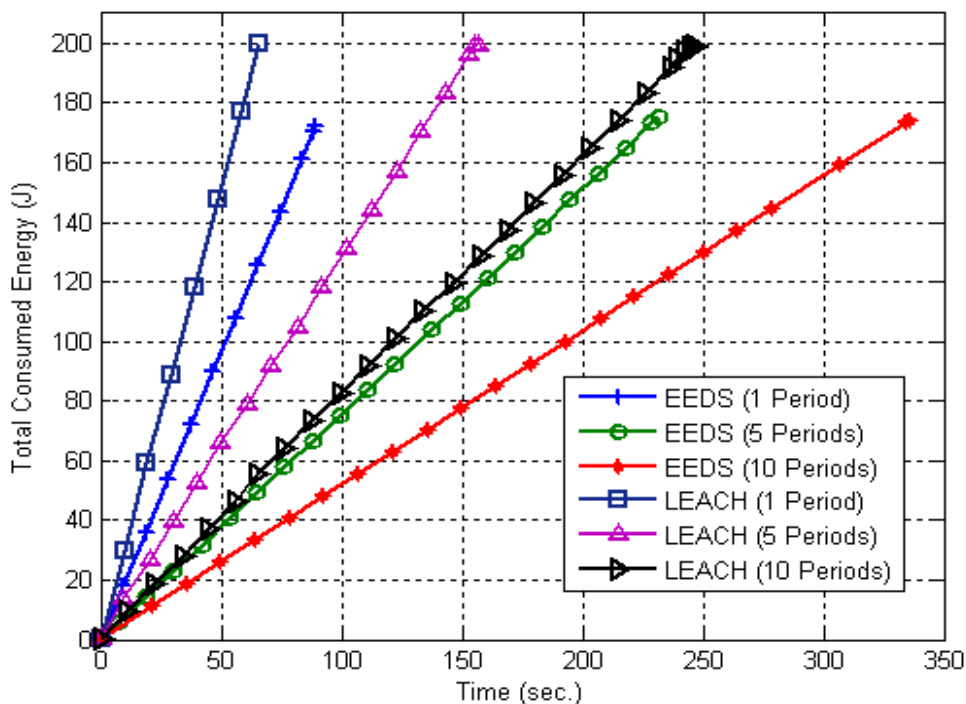


**Figure 5-11: Number of Live Nodes vs. Time, (EEDS, LEACH)**

To compare the energy consumed by a node in LEACH and a node in EEDS, we note that in both protocols there are two types of nodes; head and non-head nodes in LEACH, leaf, and non-leaf nodes in EEDS. The energy consumed by leaf node in proposed protocol is less than the energy consumed by the non-head node in LEACH, because in EEDS it is assumed that the parent of a leaf node is always one of the nearest neighbors of the node. Therefore, the transmission distance will be very short. i.e. its transmission energy will be smaller. While in LEACH, the parent of non-head node may be any head in the network, since it is assumed that all nodes will hear each other. Although the parent of a node is the closest head but it may not be one of the nearest neighbors. So its transmission energy will be larger compared with the node in EEDS. On the other hand, the head in LEACH will consume more transmission energy than non-leaf node in EEDS, since the head node uses CSMA while non-leaf node is scheduled to

transmit at a particular time slot. The head has to be ON and sense the channel to get its turns for transmission for the whole data transmission period. While non-leaf node will be ON only at its assigned time slot and it will be OFF for the remaining data transmission period. In addition, the head will transmit to the sink, which may be far away from it, while non-leaf node will transmit to its parent that is very close to it. Regarding the consumed receiving energy, the non-leaf node in EEDS will consume smaller energy than the head node in the LEACH, since the non-leaf node will have smaller number of children compared with the number of children of the head node. In the LEACH, any node can join a head, while in EEDS only the nearest nodes to the non-leaf node will join it. Taking into account the consumed energy during transmission and receiving, the total consumed energy in EEDS will be smaller compared with the LEACH as shown in Figure 5-12. For example, with 10 data transmission periods, at  $t=50$  seconds, the total consumed energy in LEACH is about 47.6 J while it is about 26.12 J in EEDS. The consumed energy is improved by about 45.1%. Furthermore, in EEDS, the overall consumed energy is less than 200 J, the initial energy in the network, while in the LEACH, it is 200 J for fewer throughputs.

The improvement in the network lifetime will improve the number of data packets delivered to the sink. Table 5-5 shows a comparison between EEDS and LEACH in terms of the number of data packets delivered to the sink. For example, for 5 data transmission period, the total number of packets delivered to the sink is improved by 29.4%.

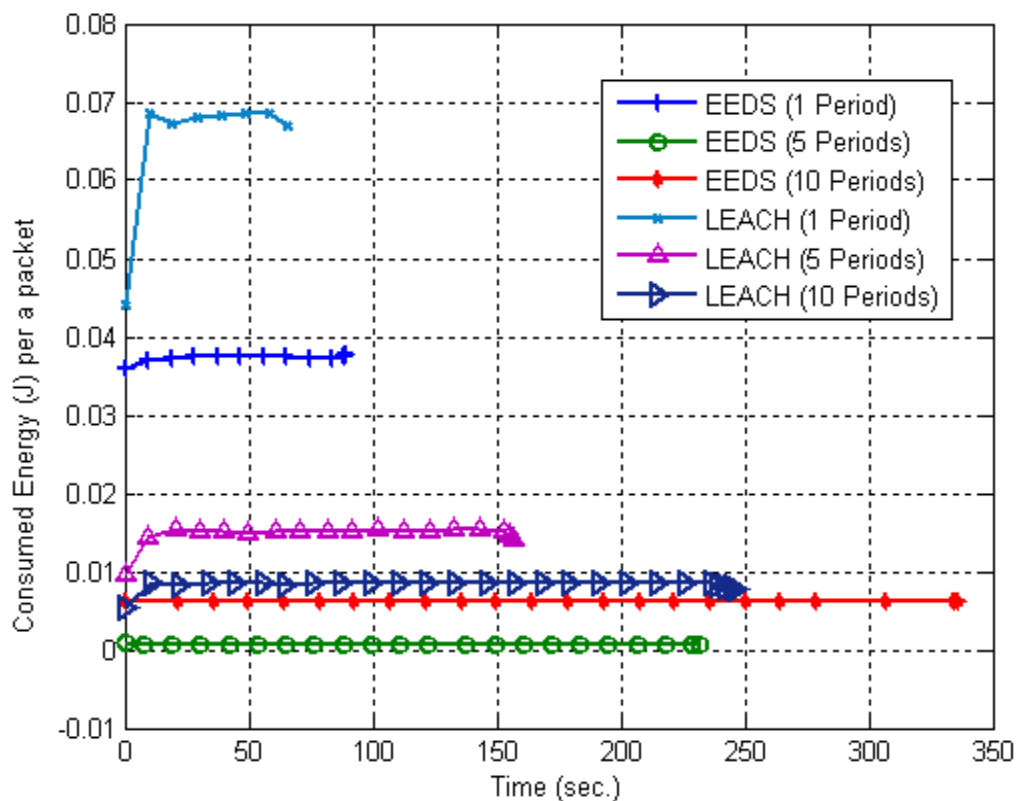


**Figure 5-12: Total Consumed Energy vs. time, (EEDS, LEACH)**

**Table 5-5: The Aggregated Throughput in EEDS compared with LEACH**

Data Transmission Period	EEDS	LEACH	Improvement
1	4569	2984	53.1%
5	18180	14045	29.4%
10	28350	25840	9.6%

Figure 5-13 shows a comparison between EEDS and LEACH in terms of the average consumed energy per a packet. We observe that under one data transmission period, LEACH outperforms EEDS, while under five and ten data transmission periods, EEDS outperforms LEACH. The justification is similar to the justification of the comparison of GET and LEACH.



**Figure 5-13: Average Consumed Energy per a single packet, (EEDS, LEACH)**

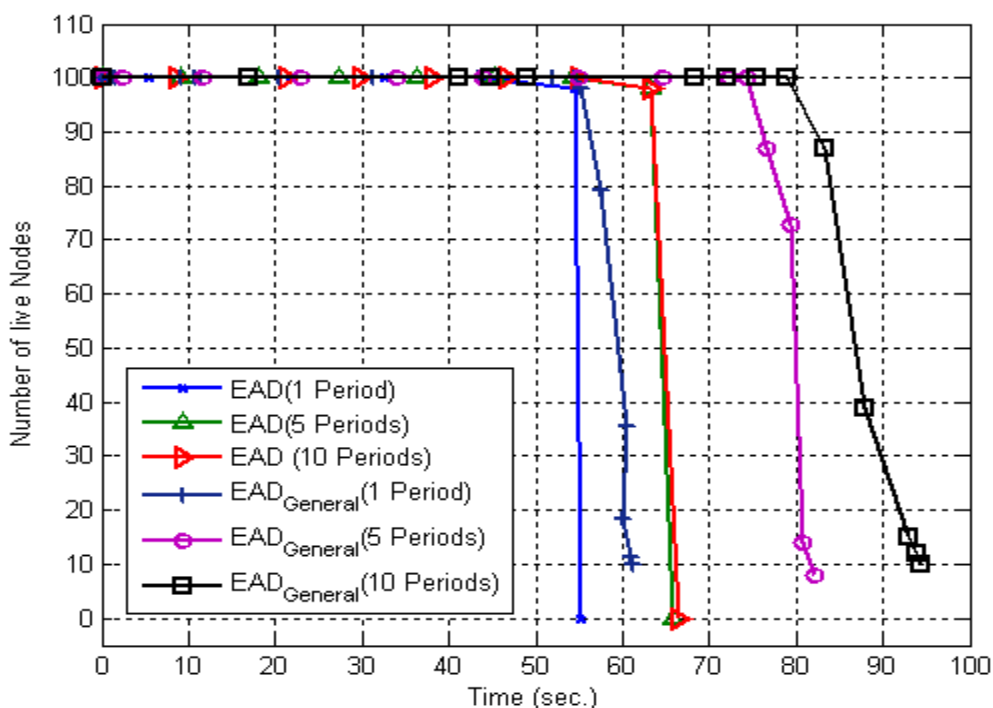
Table 5-6 shows a summary of the improvement in EEDS compared with EAD and LEACH for all metrics (network lifetime, throughput, energy consumed) using different data transmission periods.

**Table 5-6: A summary of the improvement in EEDS compared with EAD and LEACH**

	One Data Transmission		Five Data Transmission		Ten Data Transmission	
	EAD	LEACH	EAD	LEACH	EAD	LEACH
Network Lifetime	61.9%	35.5%	251.4%	47.8%	405.4%	36.0%
Energy Consumed (t=50 s)	39.1%	36.5%	65.6%	42.6%	74.5%	37.7%
Throughput	0.17%	53.1%	124.9%	29.4%	221.8%	9.6%

#### 5.4 PERFORMANCE EVALUATION OF $EAD_{General}$

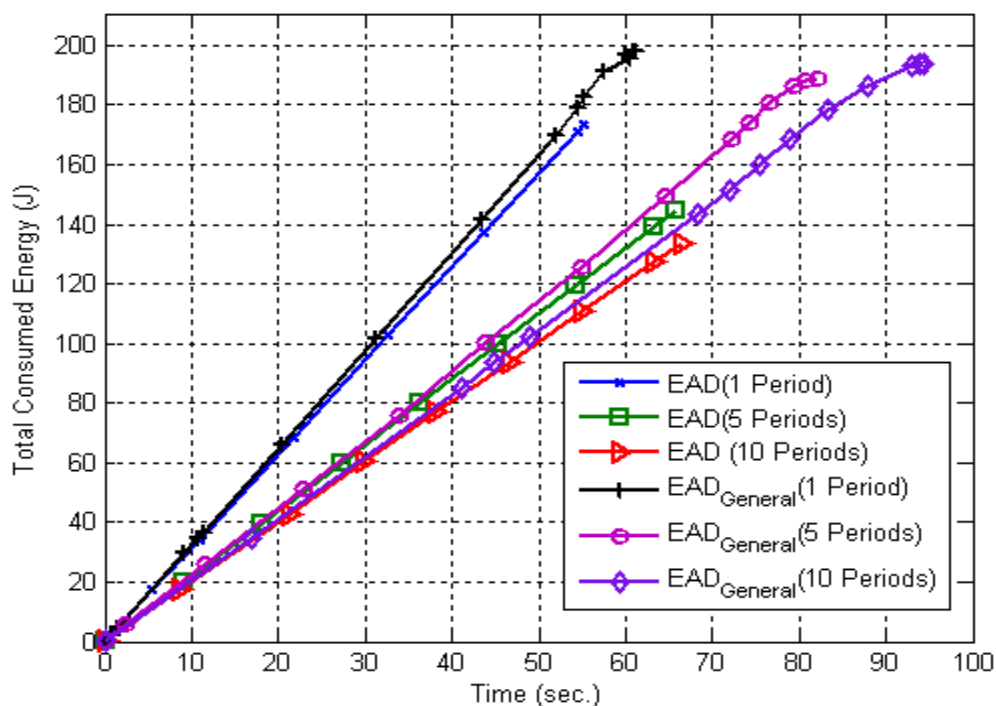
In this section we compare the performance of  $EAD_{General}$  with EAD. Figure 5-14 to Figure 5-16 show a comparison between the results obtained when implementing the original (EAD) and the proposed general EAD ( $EAD_{General}$ ). Figure 5-14 shows that  $EAD_{General}$  outperforms EAD in terms of the network lifetime for all data periods. For example, in 10 data periods, the network lifetime when implementing  $EAD_{General}$  is about 97 seconds, while it is only 67 seconds when EAD is implemented. An improvement by 44.8% is achieved. The improvement is due to utilizing the isolated nodes by increasing the candidate gateways.



**Figure 5-14: Number of Live Nodes vs. Time, (EAD,  $EAD_{General}$ )**

In both protocols, the number of live nodes is stable around 100 then goes down. It goes down very fast in the EAD, while it goes down gracefully in  $EAD_{General}$ . Further, in

the EAD, only the nodes close to the sink will act as gateways for the entire network lifetime. Since the gateway nodes will wait until it receives all the data packets from all leaf nodes, they will transmit their data packets to the sink at the end of the data transmission period, so they have to turn their radio ON for the whole data transmission period. This will consume a lot of energy; therefore, they will die very early. When the gateway nodes die, the rest of the nodes will be isolated since they will not be able to communicate with the sink although they may still have unutilized energy. This can be observed in Figure 5-15 which shows a comparison between the total consumed energy when implementing EAD, and the total consumed energy when implementing EAD<sub>General</sub>.



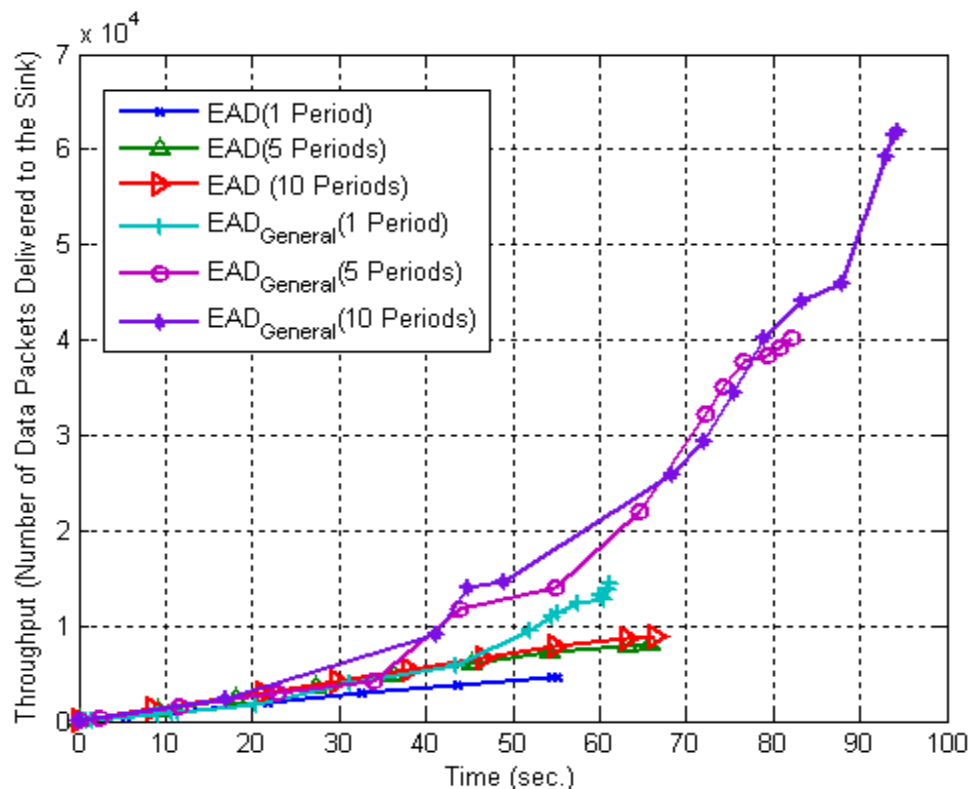
**Figure 5-15 : Total Consumed Energy vs. Time (EAD, EAD<sub>General</sub>)**

Since we have 100 nodes in the network, and initially each node has 2J, then the total initial energy in the network is 200J. From Figure 5-15, we observed that the total

consumed energy in the EAD with different data transmission period does not exceed 180J. There is still more than 20J as non-consumed energy. This non-consumed energy is stored in the isolated nodes. It is considered a wasted energy. In EAD, the gateway nodes will die approximately at the same time so the number of live nodes goes down very fast. On the other hand, in the  $EAD_{General}$ , any node in the network can be a gateway. When the nodes close to sink die, any other node in the network can be a gateway, therefore the number of isolated nodes will be decreased compared with EAD. Consequently, the number of live nodes will decrease more smoothly in  $EAD_{General}$  compared with EAD. Since any node can be a gateway in  $EAD_{General}$ , most of the nodes will be able to communicate with the sink and will stay active in the network until all its energy is utilized. This can be illustrated by Figure 5-15 where the total consumed energy in  $EAD_{General}$  is about 200 J, which is very close to the total initial energy in the network. The energy utilization is more efficient in  $EAD_{General}$  compared with EAD.

On the other hand, due to the increase in the network lifetime a significant improvement in the throughput is achieved as shown in Figure 5-16. For example, the total data packets delivered to the sink when implementing  $EAD_{General}$  for 10 data periods is about 62000 packets, while it is about 10000 packets in EAD. An improvement by 520% is achieved. The significant improvement in throughput can be attributed to increasing the number of gateways. In the  $EAD_{General}$ , there will be more gateways than in the EAD. Increasing the number of gateways will increase the data packets delivered to the sink. Although number of data packets delivered to the sink is improved in  $EAD_{General}$ , we must take into account that these data packets are more correlated compared with the data packets in the EAD.





**Figure 5-16: Throughput vs. Time, (EAD, EAD<sub>General</sub>)**

## 5.5 CONCLUSION

In this chapter, we evaluate the performance GET, EEDS, and EAD<sub>General</sub> assuming grid topology. The protocols are validated using different network configurations and compared with EAD and LEACH in terms of network lifetime, throughput and energy consumption.

Compared to EAD, GET has improved the network lifetime from 78% to 237%. Furthermore, the energy consumption is reduced by 65% and the throughput is significantly improved. When compared to LEACH, GET has shown outstanding

improvement in network lifetime, throughput, and consumed energy. The network lifetime is improved by about 50%, while the energy consumption is reduced by 51%; on the other hand, the throughput is extensively improved by GET by more than 291% in some cases.

Compared with EAD the network lifetime in EEDS is improved by at least 67.3%. On the other hand, the energy consumption is also decreased by 34.4% to 74.5% and the aggregate throughput is significantly improved. On the other hand, compared with LEACH, the network lifetime in EEDS is improved by at least 33.3%., the energy consumption is also decreased by more than 33.3%, while the throughput is improved by 8.6 %-48.7%.

Finally,  $EAD_{General}$  is examined against the EAD using simulation. It shows significant improvements in terms of larger network lifetime, less consumption energy and higher throughput.

In the next chapter, we will evaluate the performance of GET, EEDS assuming different network topologies, grid and random. Moreover, we will evaluate the performance of the protocols assuming applications with different inter-arrival time between events.

# Chapter Six

## PERFORMANCE EVALUATION OF GET AND EEDS UNDER DIFFERENT NETWORK CONFIGURATIONS AND APPLICATIONS

In the last chapter, we evaluated the performance of EEDS and GET assuming grid network topology, which lend itself to many applications some fields such as environment monitoring, and agriculture. We assumed that all nodes always have data to transmit.

In this chapter, we shall investigate the sensitivity of sensor deployment method on the performance of the proposed routing protocol. In addition, the effect of network size on the performance of the proposed protocol will be studied. Moreover, we will discuss the performance of proposed protocols for different applications. We characterize applications based on the inter-arrival time between events.

In addition, in the last chapter, we identified throughput as the number of data packets delivered to sink. In hierarchal protocols such as EEDS, GET, EAD and LEACH, each data packet delivered to sink is aggregated from many raw data packets. Two different packets may be aggregated from different number of raw packets; therefore it is

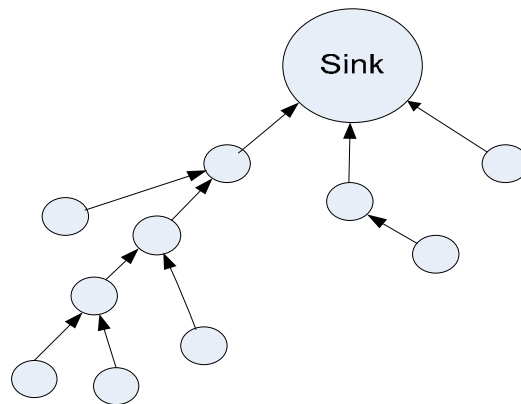
unfair to consider that the two packets are identical. In this chapter, we propose an entropy-based throughput metric to measure the throughput of the hierarchal routing protocols for WSN. In the proposed metric, the information delivered to the sink is calculated instead of the number of data packets delivered to the sink. The information in the packet delivered to sink is measured base on the number of raw packets which the packet is aggregated from. This metric will lead to fair comparison and evaluation of different routing protocols as well as more informative decision by the sink. We use the proposed metric to compare the performance of our proposed protocols, GET, and EEDS to LEACH and EAD.

This chapter is organized as follows; section 6.1 presents the entropy-based throughput metric for WSN. Performance evaluation of EAD and LEACH will be presented in section 6.2. The effect of network topology on the performance of EEDS, GET, EAD and LEACH is discussed in 6.3. Section 6.4 discusses the performance evaluation for EEDS, GET, EAD and GET assuming different network sizes. The performance of EEDS, GET, and EAD assuming applications with different inter-arrival time is discussed in 6.6. Section 6.7 presents a modified building schedule algorithm. Finally, the performance of EEDS, GET and EAD will be discussed in this section also.

## 6.1 AN ENTROPY-BASED THROUGHPUT METRIC FOR WSN ROUTING PROTOCOLS

In the hierarchal routing protocols for WSN, clusters are usually formed. In each cluster, a set of nodes are connected to a head as shown in Figure 6-1. The head collects the data packets from the nodes and aggregates them into one packet. The resulted

packets are then transmitted to the sink. In each routing protocol, different techniques are used to form the clusters. The number of nodes in each cluster is different. To calculate the throughput, it is not accurate to consider that all packets delivered to the sink have the same amount of information. For example, in Figure 6-1 the packet delivered through the most left branch is aggregated from 7 distinct packets while the one delivered through the most right branch is resulted from just one packet.



**Figure 6-1: A network with different Clusters**

To distinguish between these two packets, we can measure the information entropy in each delivered packet. In general, for a discrete random variable  $X$  that takes values from  $x$ , the Shannon entropy of  $X$ ,  $H(X)$  is defined by

$$H(x) = - \sum_{x \in X} p(x) \log_2 p(x) \quad (1)$$

To apply the above concept to WSN, we will consider the packet that is aggregated from more "raw" packets (i.e. original sensed information) to be more informative.

Therefore, for a cluster  $x$  composed of  $C$  nodes, we assumed that  $p(x) = \frac{1}{C}$ .

The Information delivered to the sink can then be defined as

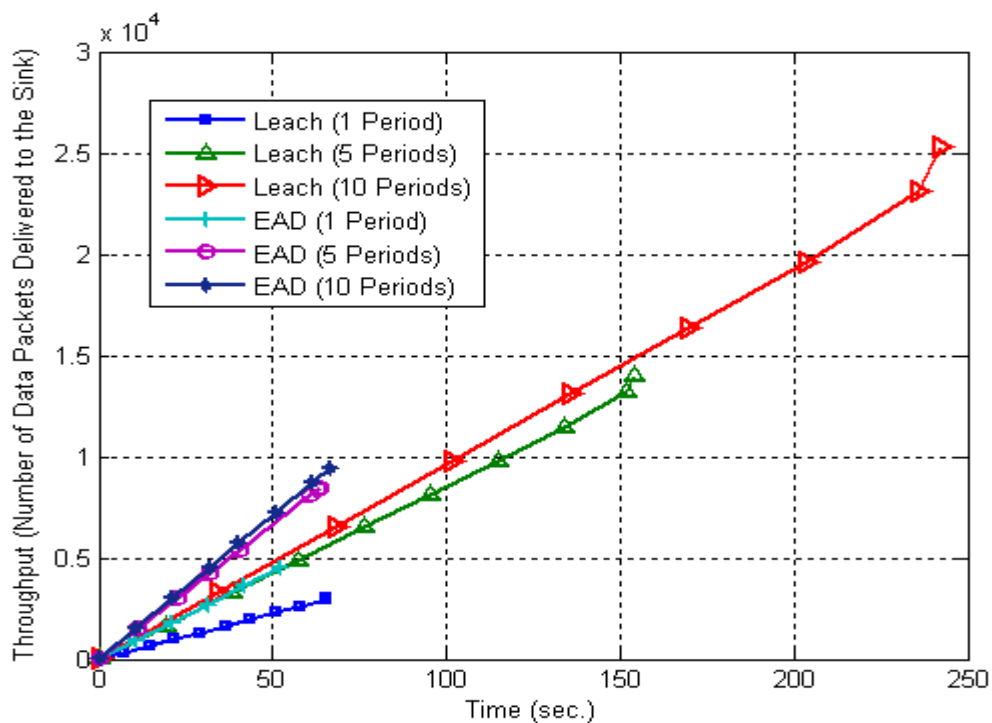
$$I = \sum_{x \in X} -\frac{1}{C_x} \log_2 \frac{1}{C_x} \quad (2)$$

where  $X$  is the set of clusters and  $C_x$  is the number of nodes in cluster  $x$ .

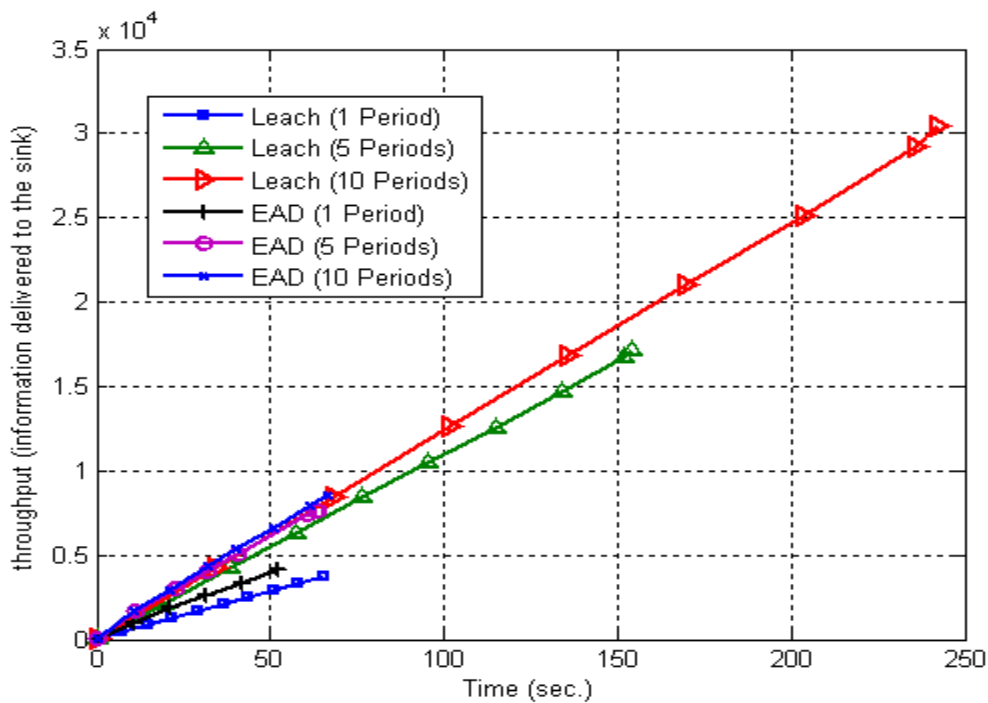
## 6.2 PERFORMANCE EVALUATION OF EAD, AND LEACH ROUTING PROTOCOLS USING AN ENTROPY-BASED THROUGHPUT METRIC

We use the proposed metric to compare between two well-known data forwarding protocols EAD and LEACH. We select these two protocols because they use a different mechanism to build the clusters. In EAD, multiple trees rooted at the sink are built. Each tree can be considered as a separate cluster. In LEACH, heads are identified in each round; each node will attach itself to one of the heads. The number of heads is not fixed in all rounds.

We compare the two protocols assuming a grid network shown in Figure 4-2. The simulation parameters shown in Table 4-1 are reused. We assume different number of data transmission period (1, 5 and 10) in each round. We measure the absolute number of data packets delivered to the sink and the information delivered to the sink according to (2). Figure 6-2 and Figure 6-3 show the comparison between the two protocols using the absolute throughput, and the information delivered to the sink, respectively. It is interesting to notice that for 1 data transmission period, assuming the absolute throughput, EAD is improved by 51%, while using the proposed information-based throughput metric EAD is improved by only 8.6%. For 5 data transmissions, the improvement in LEACH using absolute throughput is 39.4% while it is 55.5% using information based throughput.



**Figure 6-2: A Comparison between EAD and LEACH using absolute throughput**



**Figure 6-3: A Comparison between EAD and LEACH using Entropy**

### 6.3 THE EFFECT OF NETWORK TOPOLOGY ON PERFORMANCE OF ROUTING PROTOCOLS FOR WIRELESS SENSOR NETWORKS

In some applications, sensor nodes are deployed randomly on the monitored area. For example, hundreds of sensors are thrown by plane in a forest for fire detection and monitoring. Therefore, it is required to investigate the WSN protocols using random topology and large number of nodes. In the previous chapters, we investigated our proposals assuming grid topology only. Moreover, we assume that 100 nodes are deployed in the monitored area.

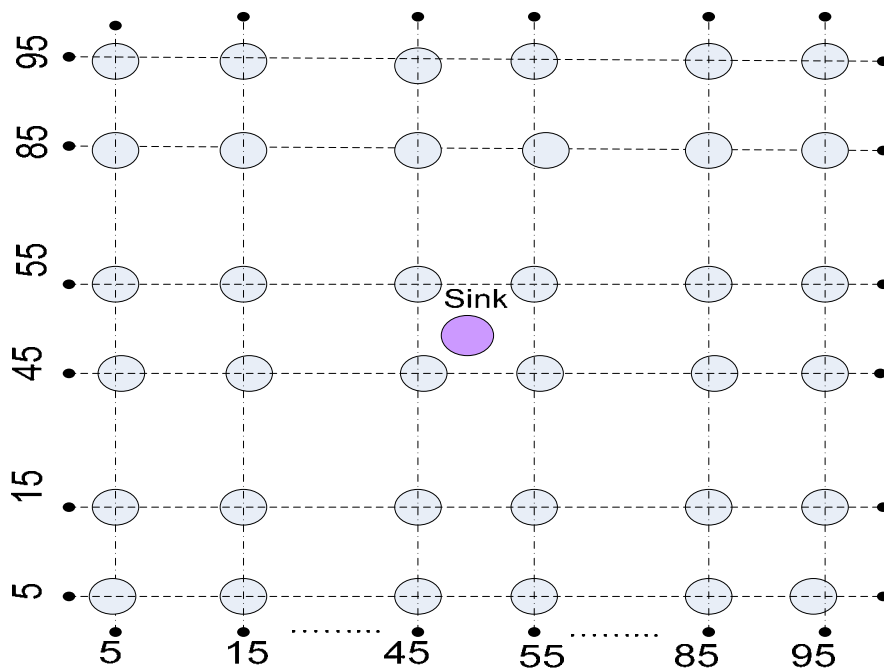
In this section, we will study the performance of different routing protocols with different network topologies, grid and random. We will evaluate the performance of EAD, LEACH, EEDS and GET. For each protocol, we will measure network lifetime, throughput, energy consumption, and percentage of covered area for the grid and random topologies. A comparison of the performance of each protocol for the grid and random topologies will be presented.

We investigate the performance of the protocols with two different network topologies; random and grid. In both topologies, 100 sensors distributed in an area of  $100 \times 100$  m<sup>2</sup>. The simulation parameters shown in Table 4-1 are reused here. The energy model presented in section 4.4 is reused here also. Moreover, we assume the sensing range of the sensor to be 10 meters.

Since the sensing range of a sensor node is assumed to be 10 meter, it is useless to position sensors on the boundary of our monitored area. Therefore, since we assume that the monitored area dimension is  $100 \times 100$  m<sup>2</sup>, in both topology, the sensor nodes will be deployed between (x=5 ,y=5) and (x=95, y=95). In the grid topology, The nodes are



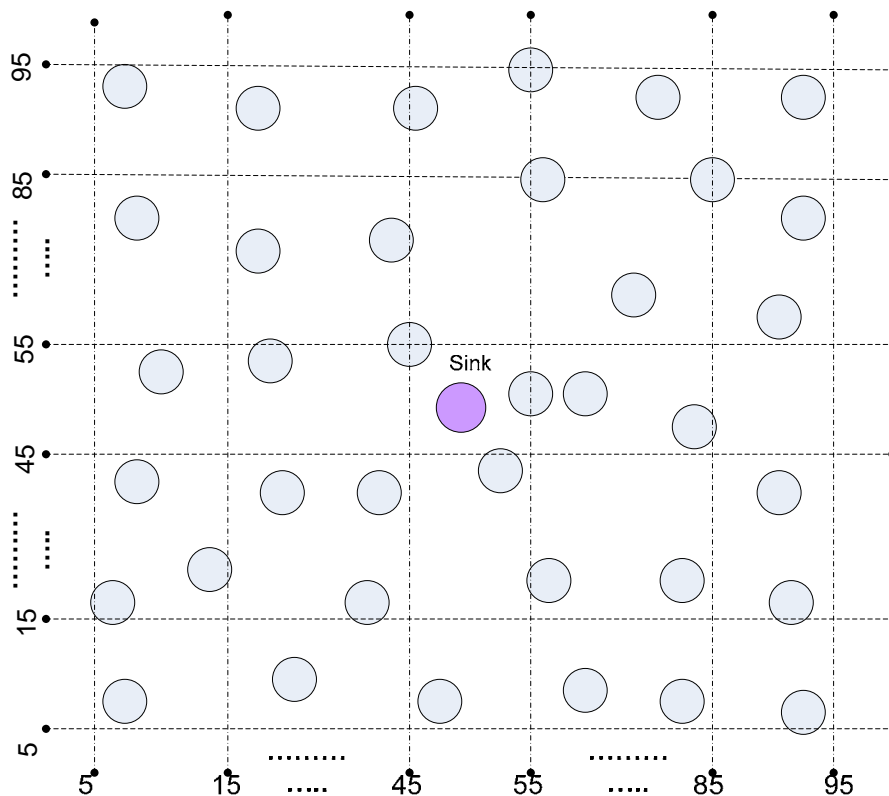
deployed in rows and columns as shown in Figure 6-4, The distance between each pair of nodes is either 10 or  $10\sqrt{2}$  m. Since the transmission range for each node is 15 m, each inner node will have eight neighbors. On the other hand, for random topology, nodes are randomly deployed between  $(x=5, y=5)$  and  $(x=95, y=95)$  as shown in Figure 6-5. The x-coordinate and y-coordinate for each node are randomly generated in the interval [5, 95]. The x and y coordinates are uniform random variables in the interval [5, 95]. In both topologies, sink is positioned at location  $(x=50, y=50)$ .



**Figure 6-4 : Grid topology**

For each configuration, we measure the network lifetime, throughput, total consumed energy, information-based throughput, percentage of covered area, and the number of gateway. To calculate the percentage of covered area, we assume that the sensing range for each node is 10 m. we divide the monitored area into small square grids, 0.20 by 0.20

m. It is assumed that a grid is covered, if there is at least one sensor that is far away from the grid by less than the sensing range (e.g. 10 meter).

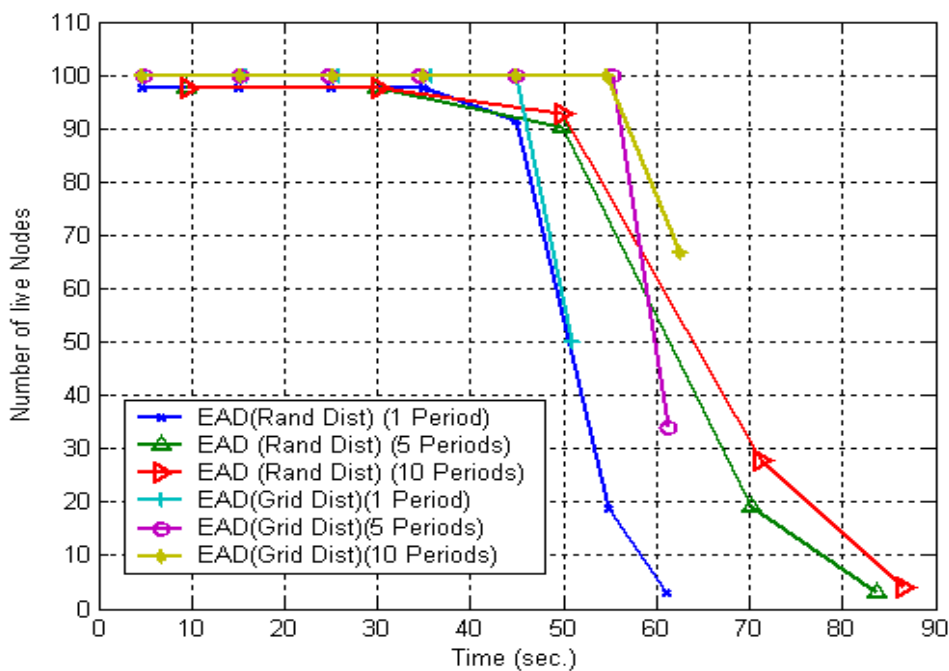


**Figure 6-5: Random Topology**

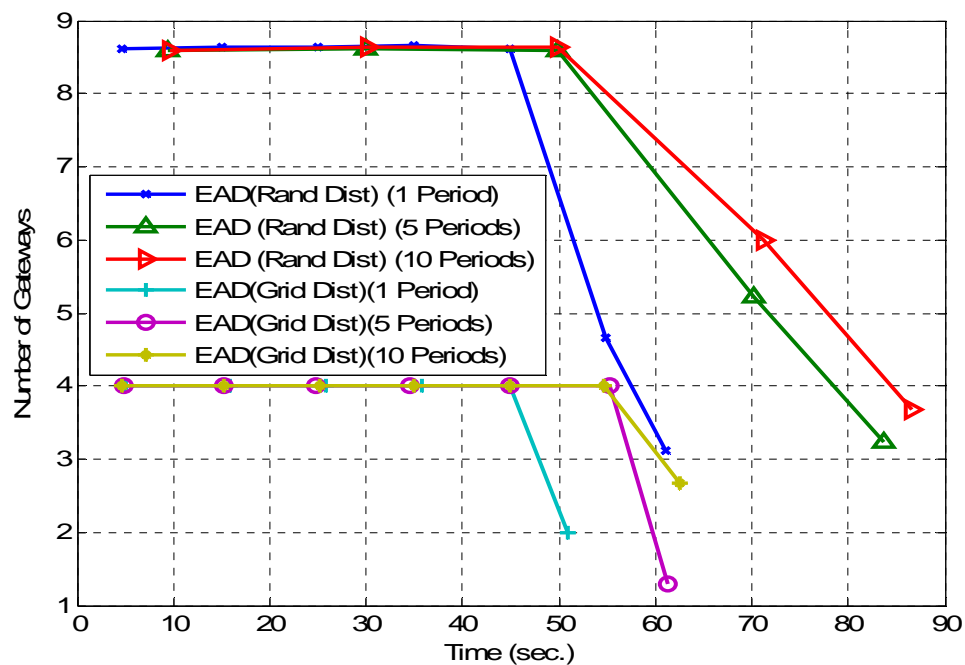
### 6.3.1 THE EFFECT OF NETWORK TOPOLOGY ON PERFORMANCE OF EAD

Figure 6-6 shows the number of live nodes vs. time for both grid and random topologies under EAD. We observe that the network lifetime for random topology is larger than the network lifetime for grid topology. For 10 data transmission periods, the network lifetime for grid topology is about 65 sec. while it is 95 sec. with random topology. An improvement by 46% is achieved. On the other hand, in random topology,

the time at which the first node dies is less than the time at which the first node dies in grid topology. In grid topology, the number of live node decreases very fast, while it decreases very smoothly in random topology. This can be explained as follows. The network is considered alive as long as a tree can be built from all nodes towards the sink. In other words, it will be alive as long as there are live nodes that are close to the sink (Gateways). In the random topology, the number of gateways is larger than the number of gateways in the grid topology. Although some gateways start to die early in the random topology, it will take longer time for all gateways to die. Therefore, the number of live nodes in the random topology will decrease gracefully as shown in Figure 6-6. On the other hand, it will take less time for all gateways to die in a grid topology.



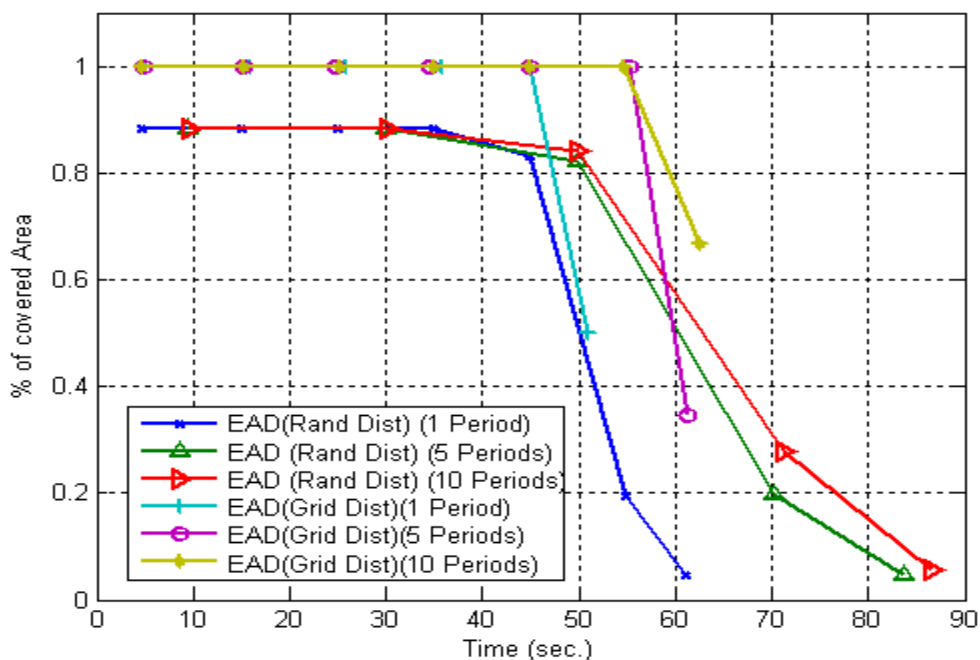
**Figure 6-6: Number of Live Nodes vs. time for EAD, Grid (Random Topologies)**



**Figure 6-7: Number of Gateway Nodes vs. time for EAD, Grid and Random Topologies**

Number of gateways in each topology can be calculated by counting number of nodes that are located within the transmission range of the sink. Since the sink is located at location (50,50) and since the transmission range for all nodes is 15m, in the grid topology, only the nodes located at locations: (45,45), (55,45), (45,55) and (55,55) will be able to communicate with the sink. The number of gateways in grid topology is only 4. On the other hand, in the random topology, the number of gateways is random. Since the nodes are distributed uniformly in the monitored area, the average number of gateways will be the average number of nodes located inside a circle centered at location of the sink, the average number of nodes located inside a circle with radius 15 m. The area of this circle is  $706.6 \text{ m}^2$ , since 100 nodes will be distributed in the whole monitored area ( $90 \times 90 \text{ m}$ )  $8100 \text{ m}^2$ , then the average number of nodes located in this circle will be 8.70

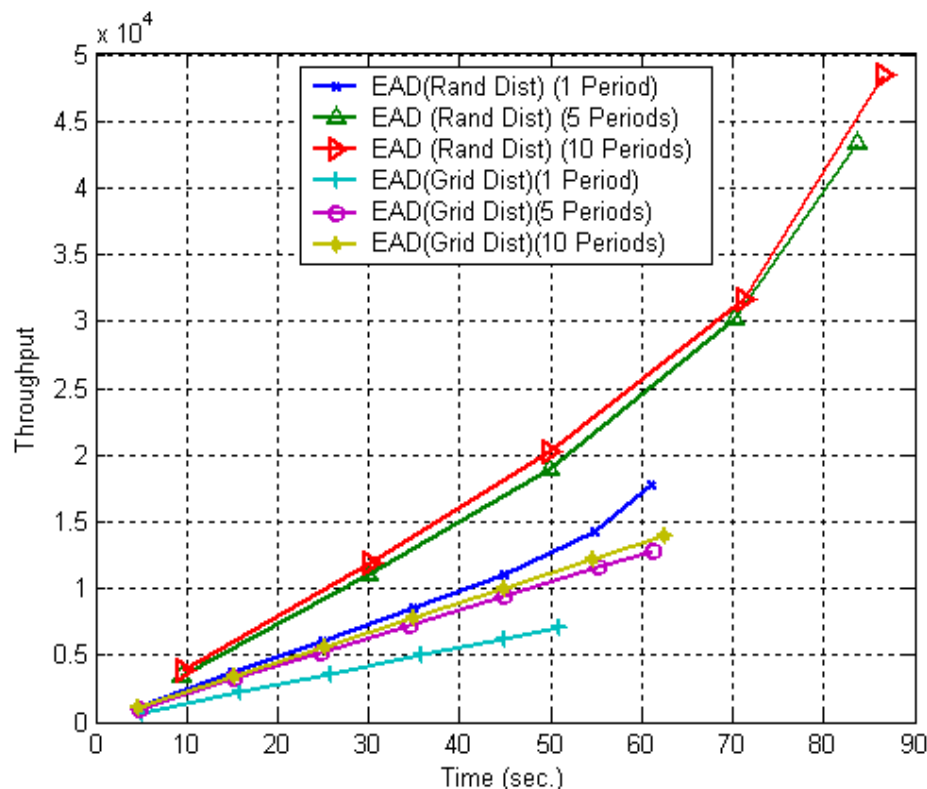
(i.e. The average number of gateways in random topology is 8.70). Figure 6-7 shows the average number of gateways in both grid and random topology as obtained by simulation. We can observe that the number of gateways for grid topology is 4 then starts decreasing, while it is about 8.7 for random topology and starts decreasing; these simulation results match the values calculated analytically



**Figure 6-8: Percentage of covered area vs. time for EAD, Grid and Random Topologies**

Figure 6-8 shows percentage of covered area vs. time for both grid and random topology. The percentage of covered area in grid topology is 100% then it starts decreasing, while it is less than 100% in random topology. This is very intuitive, in grid topology, the distance between nodes is 10 meter and the sensing range is 10 meter, then at the beginning of network lifetime when all nodes are alive all the area will be covered. When some nodes start to die, the percentage of covered area will start decreasing. On the other hand, in the random topology, the nodes are randomly distributed; the nodes are

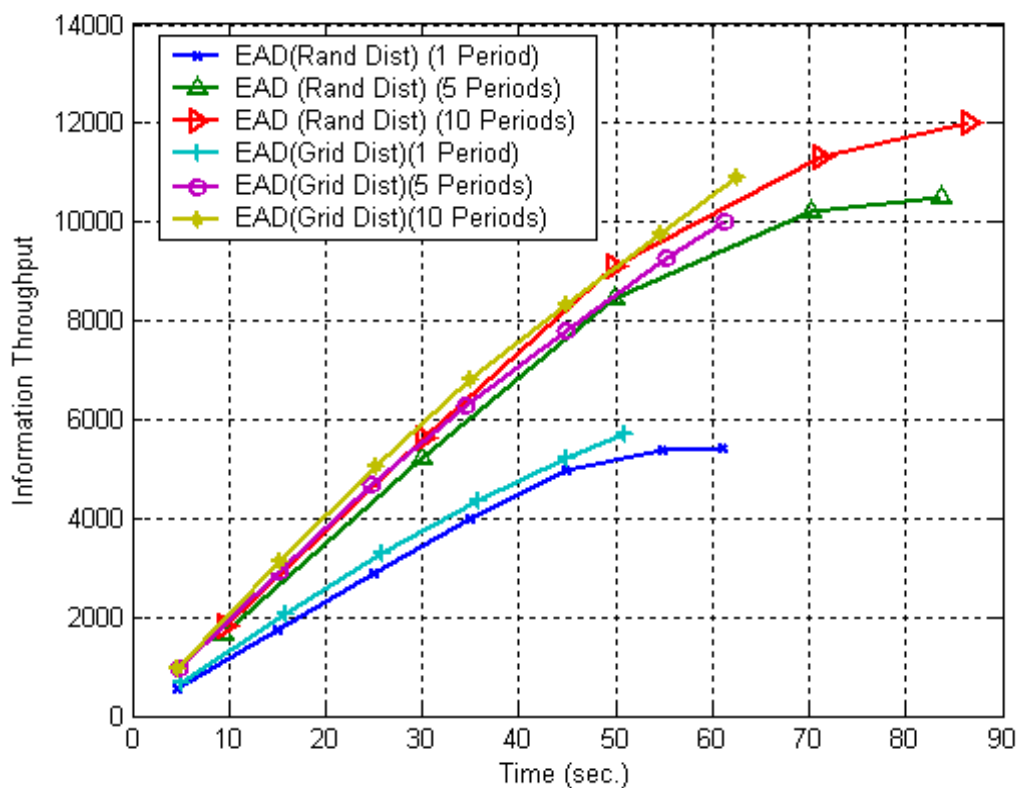
not well organized. Some small grids of the monitored area will not be covered. Even when all the nodes are alive, there are still some regions that are not covered.



**Figure 6-9: Throughput vs. time for EAD, Grid and Random Topologies**

Regarding the total number of packets delivered to the sink (i.e. throughput), we more packets will be delivered when random topology is chosen as illustrated in Figure 6-9. This result is attributed to the large number of gateways in the random topology. For the grid topology, the number of gateways is 4. Therefore, four packets will eventually be delivered to sink in each data transmission period. Meanwhile, in average, 8.7 packets will be delivered to sink in random topology. For 1 data transmission period, the total number of packet delivered to sink in grid topology is 7500, while it is 17500 in random topology. Throughput is improved by 133%. Since the number of gateways in random

topology is larger than number of gateways in grid topology, then the number of nodes associated with each gateway in random topology will be smaller. Therefore, each packet delivered to the sink in random topology will be aggregated out of fewer number of raw data packet compared with packets delivered to sink in grid topology. however, It will have less information. In random topology, we have greater number of packets with less information in each packet compared with the information in each packet delivered to sink in grid topology.



**Figure 6-10: Information Throughput vs. time for EAD, Grid and Random Topologies**

Therefore, the WSN architect should pay careful when he deals with these data diffused packets. Information throughput delivered to sink in both random and grid topology is shown in Figure 6-10. We observed that the difference between information

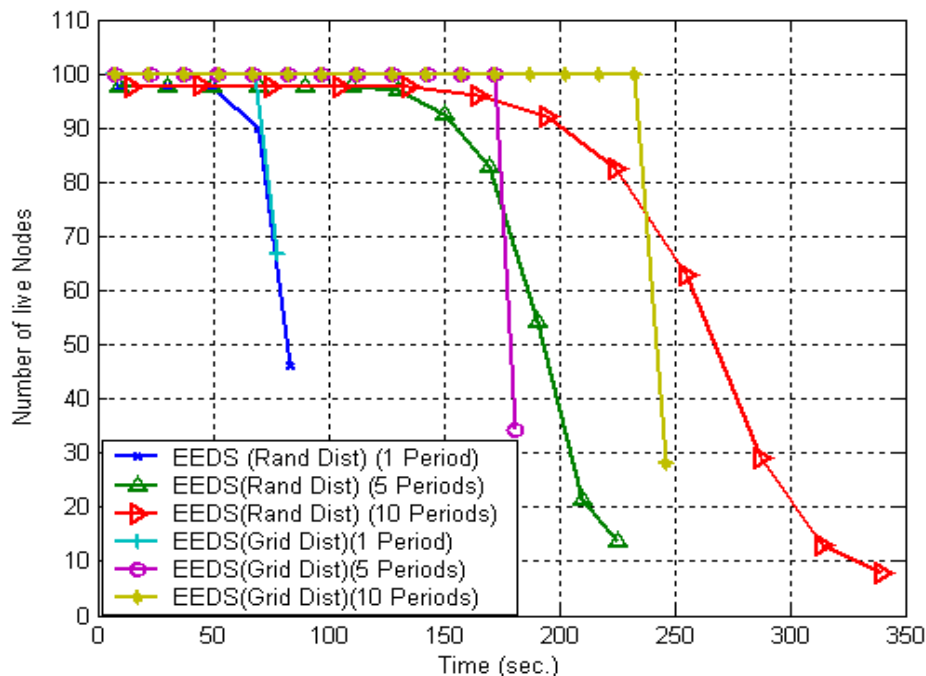
throughputs in both topologies is not very high as in the throughput. For example, with 5 data transmission periods, the information throughput in grid topology is 10000 while it is about 11000 in random topology. The improvement is about 10% only, while the improvement in throughput was 133%. The 10% improvement in information throughput is attributed to improvement in network lifetime. On the other hand, the 133% improvement in throughput is due to improvement in network lifetime and due to increasing in the number of gateways.

### 6.3.2 THE EFFECT OF NETWORK TOPOLOGY ON PERFORMANCE OF EEDS :-

Figure 6-11 shows the number of live nodes vs. time for EEDS for both grid and random topologies. We can observe from Figure 6-6 and Figure 6-11 that the network lifetime of EEDS is greater than the network lifetime in EAD. Figure 6-12 shows number of gateways in EEDS number of gateways in random topology decreases smoothly in random topology, while it decreases very fast in grid topology. as we explained in EAD, number of gateways in random topology is greater than in grid topology. Therefore, it will take longer time for all gateways to die in random topology. We observe that number of gateways and number of live nodes decrease smoothly in EEDS. On the other hand, the number of gateways and number of live nodes in EAD decrease very fast. In EAD, random scheme is used to forward data; all gateways will be awake for the whole data transmission period, therefore most of gateways will die almost at the same time. On the other hand, TDMA scheme is used for data transmission in EEDS. Therefore, the gateways will be ON for their assigned time slots only. The number of time slots for each



gateway depends on number of children which is different for gateways. Hence, gateways will die at different time.



**Figure 6-11: Number of Live Nodes vs. time for EEDS, Grid and Random Topologies**

The percentage of covered area in EEDS is shown in Figure 6-13. For grid topology, as in EAD, percentage of covered area starts from 100% then decrease very fast when all gateways die. For random topology, percentage of covered area starts from less than 100%, then it starts decreasing smoothly. It decreases smoothly since number of gateways is larger; it takes longer time for all nodes to die.

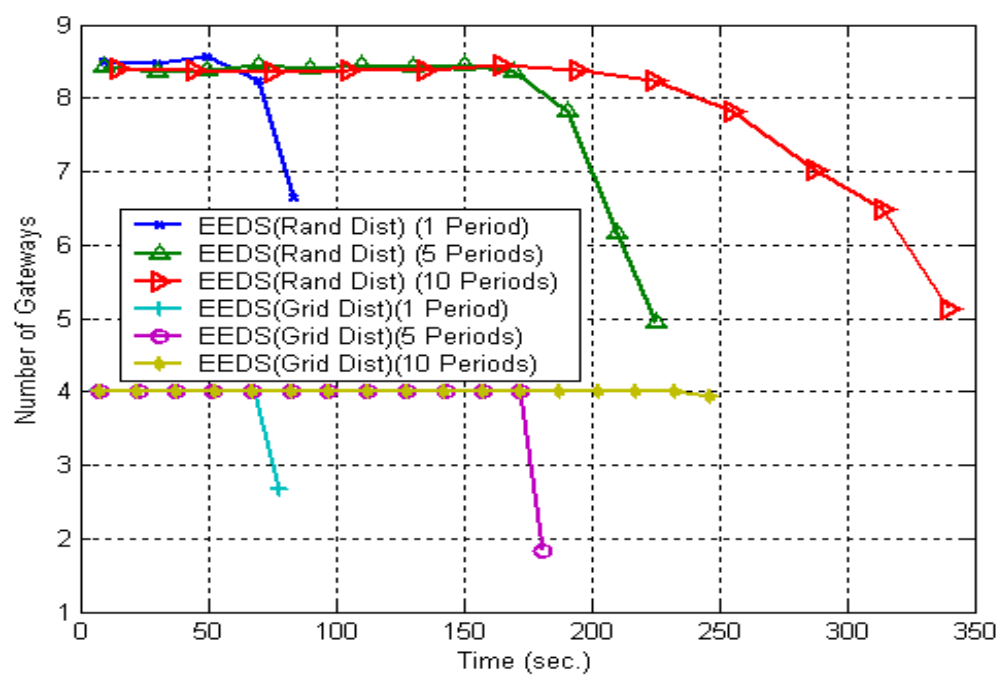


Figure 6-12: Number of Gateways vs. time for EEDS, Grid and Random Topologies

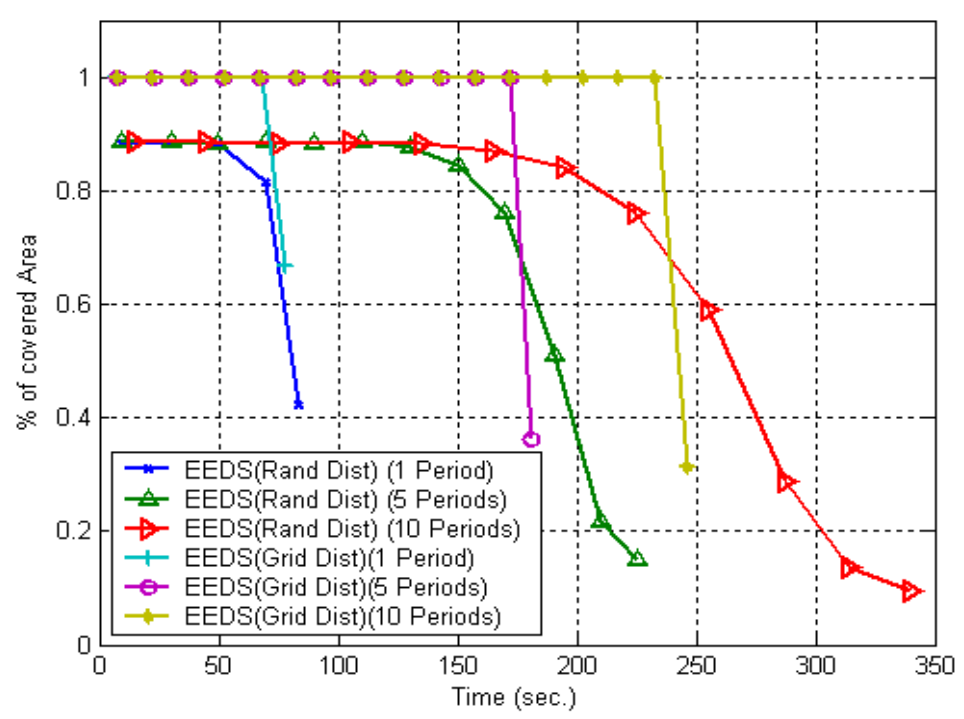


Figure 6-13: Percentage of Covered area vs. time for EEDS, Grid and Random Topologies

The throughput vs. time for EEDS is shown in Figure 6-14. We observed that the total throughput in random topology is very large compared with total throughput in grid topology. This is due to larger number of gateways in random topology compared with grid topology. Although the throughput is larger in random topology, the information throughput is approximately equal in random and grid topology as shown in Figure 6-15. Since number of gateways in random topology is higher than grid topology, the number of nodes associated with each gateway in random topology will be fewer than grid topology. Therefore, more number of packets will be delivered to sink in random topology, but each packet is aggregated from smaller number of raw packets.

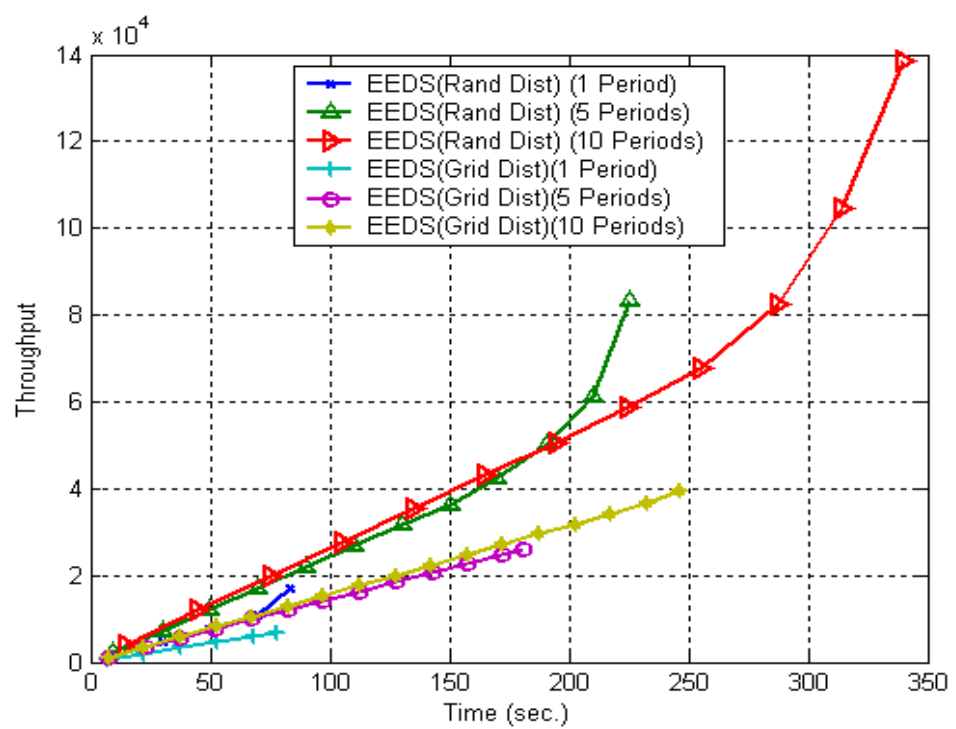
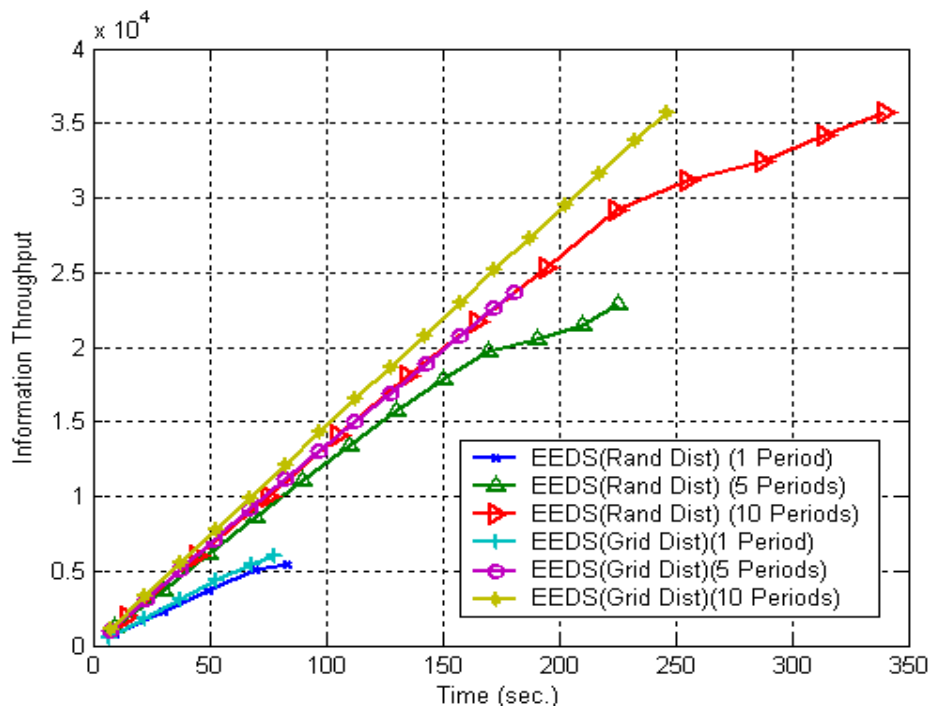


Figure 6-14: Throughput vs. time for EEDS, Grid and Random Topologies



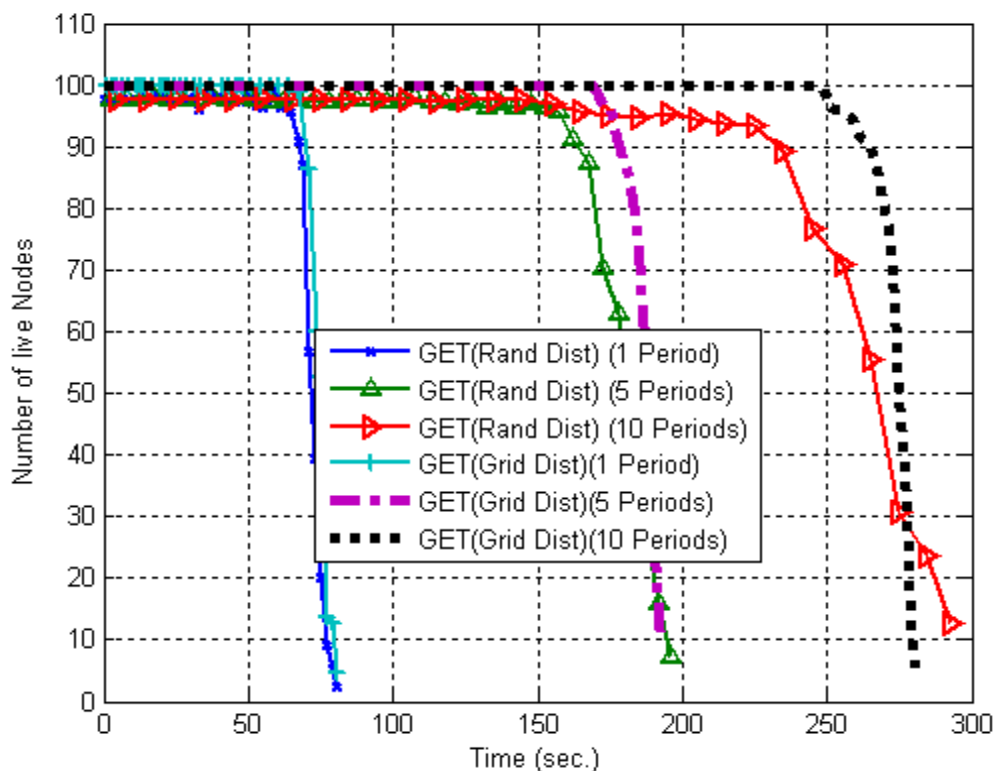
**Figure 6-15: Information Throughput vs. time for EEDS, Grid and Random Topologies**

### 6.3.3 THE EFFECT OF NETWORK TOPOLOGY ON PERFORMANCE OF GET:-

In evaluating GET, we assume that the initial value of  $E_{th}$  is 50% of the initial energy stored in a sensor node (i.e.1 J). Every cycle  $E_{th}$  will be reduced by a factor of 0.5 ( $e=0.5$ ).

Figure 6-16 shows the number of live nodes versus time for GET for random and grid topologies. We observe that GET shows better performance in grid topology compared with random topology. The main idea of GET is to select gateways from different tiers of the network. In the grid topology, nodes are well organized; therefore the tiers are setup such that each tier will have sufficient number of candidate gateways. In random topology, nodes are not well organized; some tiers may not have sufficient candidate

gateways. One possible scenario is as follows. The sink broadcasts ADV signal with a specific transmission energy. If all the nodes that hear this ADV signal have residual energy less than  $E_{th}$ . Then, none of them will advertise itself as a gateway. In this case, all the nodes are staying ON, but no gateways are selected. The nodes waste energy without selecting gateways. Therefore, the nodes will die earlier.

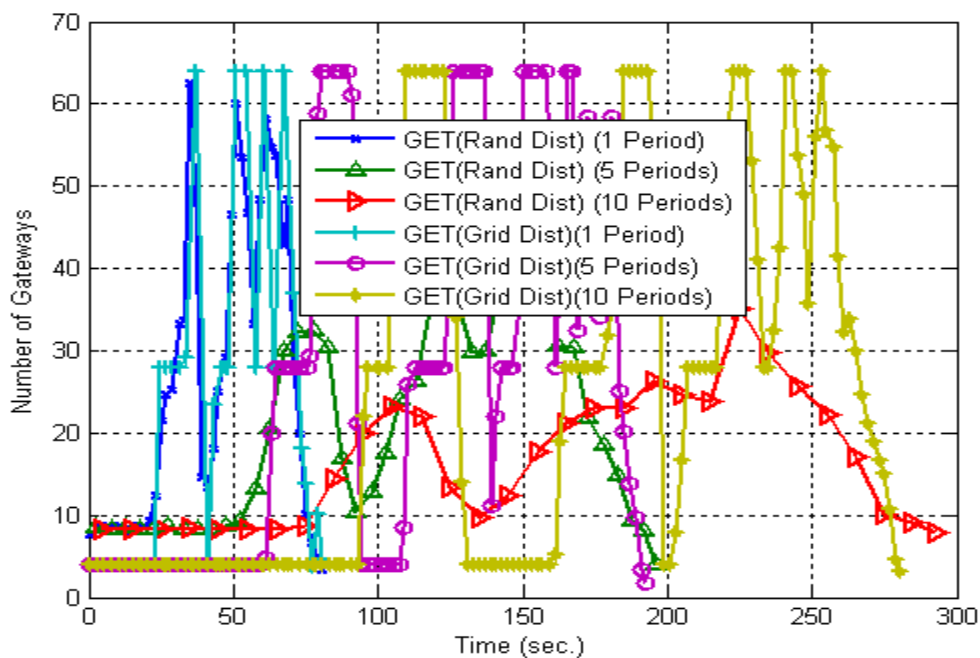


**Figure 6-16: Number of Live Nodes vs. Time for GET, random and Grid Topologies**

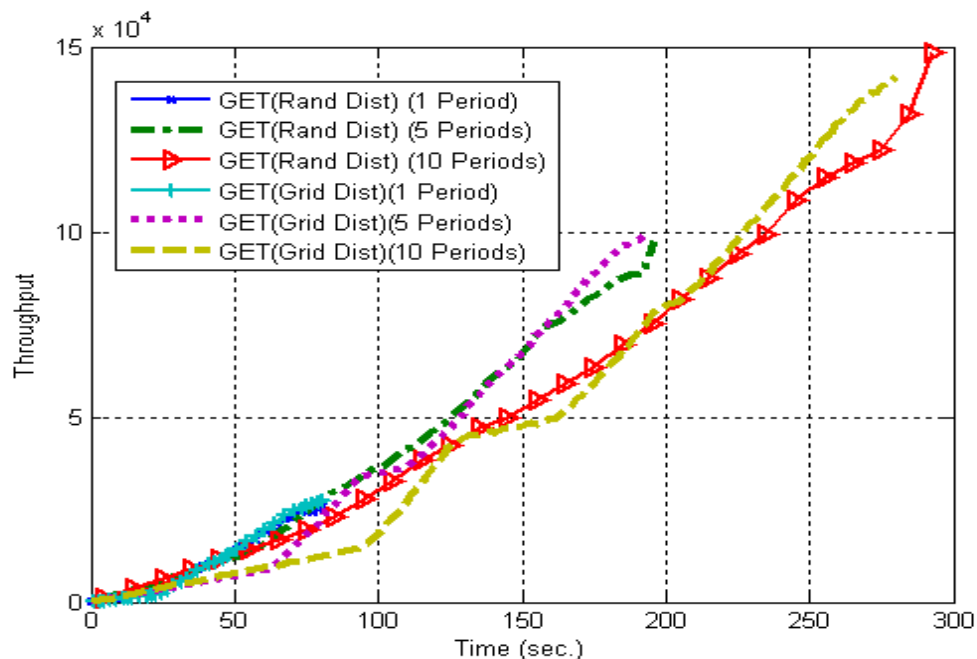
Figure 6-17 shows the number of gateways vs. time. In both random and grid topologies, the number of gateways is varied based on the current tier.

Figure 6-18 and Figure 6-19 show the throughput and information throughput in GET respectively. We observe that the total throughput in random topology is equal to the throughput in grid topology. They are equal since the average number of gateways in

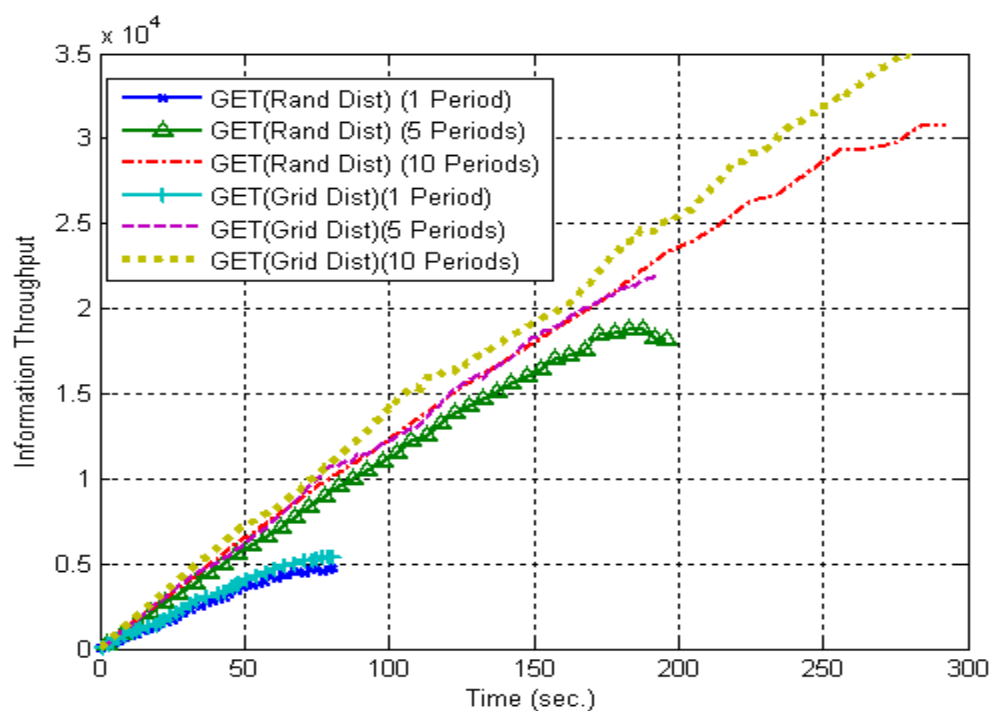
both topologies is equal. In contrary to EEDS, any node can be a gateway. In grid topology the number of gateways is not four as in EEDS. It is 4 in the first tier, and it will be larger in successive tiers. Information throughput in random and grid topologies is almost equal except for 10 data transmission periods where the grid topology shows a about 17% improvement.. The total consumed energy is shown in Figure 6-20. The total consumed energy in random topology equals to the total consumed energy in grid topology for 1 and 5 data transmission periods. For 10 data transmission periods, energy consumed in grid topology is higher than in random topology.



**Figure 6-17: Number of Gateways vs. Time for GET, random and Grid Topologies**

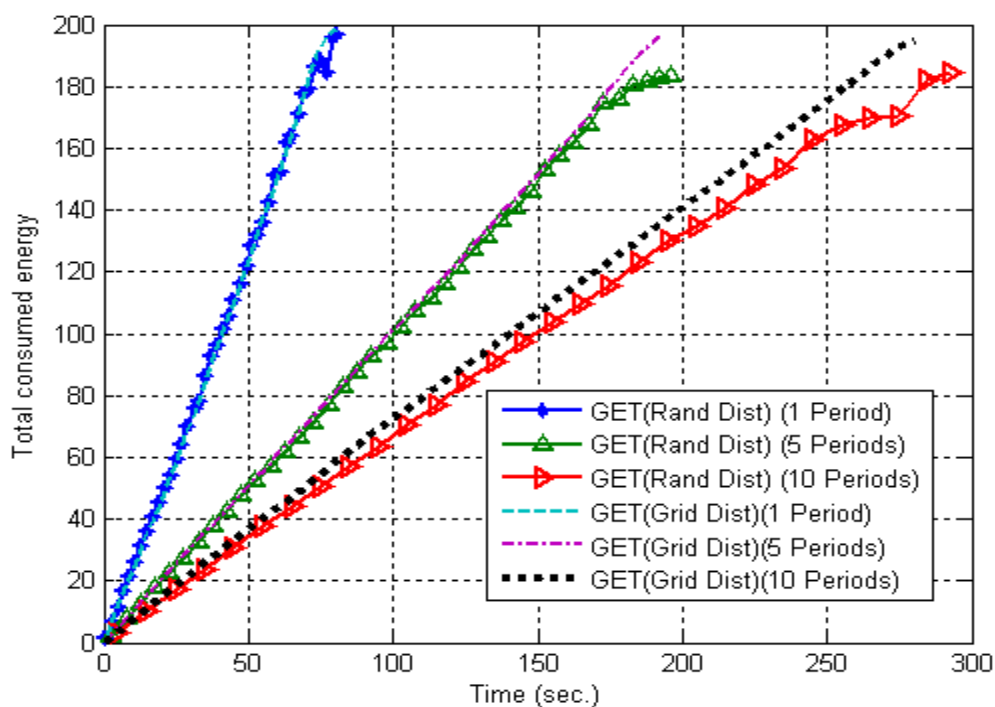


**Figure 6-18: Throughput vs. Time for GET, random and Grid Topologies**



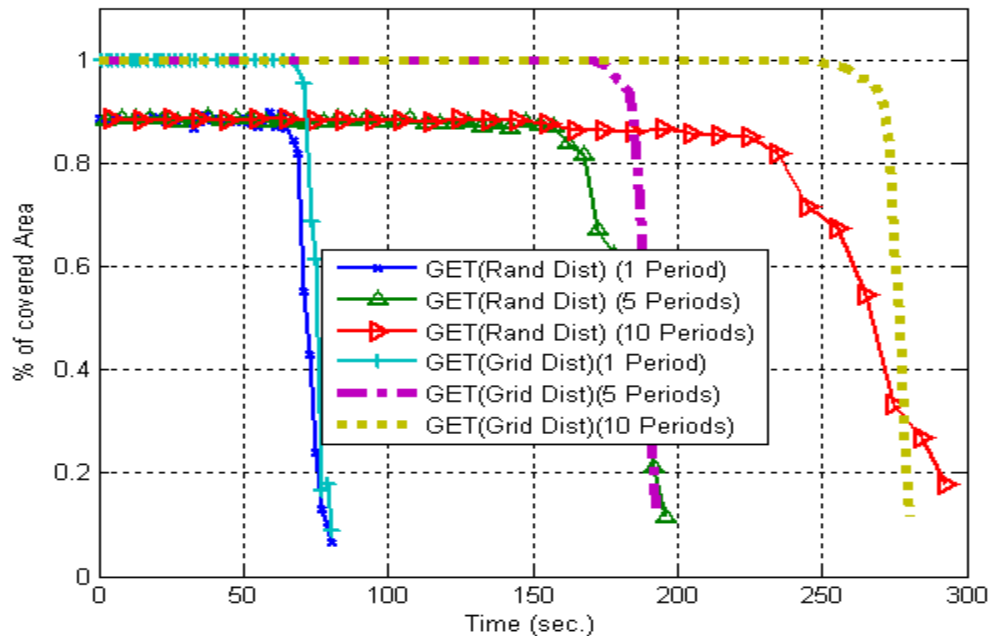
**Figure 6-19: Information Throughput vs. Time for GET, random and Grid Topologies**

The percentage of covered area in GET in random and grid topologies is shown in Figure 6-21 . The percentage of covered area in grid topology is better than in random topology. For example at  $t=240$  seconds, the percentage of covered area in random topology is 80% while it is 100% in grid topology. The percentage of covered area in grid topology is improved by 25%. The well organization of nodes in grid distribution enhances the percentage of covered area.



**Figure 6-20: Total Consumed Energy vs. Time for GET, random and Grid Topologies**





**Figure 6-21: Percentage of covered area vs. Time for GET, random and Grid Topologies**

#### 6.3.4 THE EFFECT OF NETWORK TOPOLOGY ON PERFORMANCE OF LEACH:-

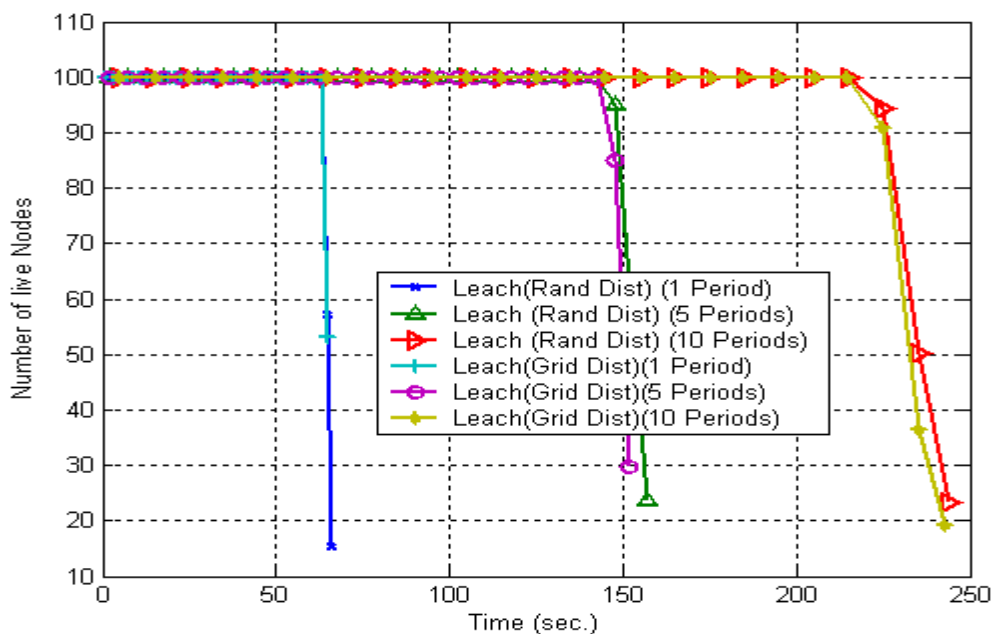
To evaluate the performance of the LEACH, the optimal number of clusters is assumed, it is calculated using (3) which is used in [3]

$$K_{opt} = \frac{\sqrt{N}}{\sqrt{2\pi}} \sqrt{\frac{E_{fs}}{E_{amp}}} \frac{M}{d_{toBS}} \quad (3)$$

where  $N$ : is number of nodes, and  $M$  is the length of the monitored area (100 m),  $E_{amp}$  and  $E_{fs}$  are the amplifying energy for short and long distances respectively. Since the sink in our configuration is placed in the middle of the monitored area and it is close to all nodes, the distance from all nodes to sink is short, therefore, we consider  $E_{amp} =$

$E_{fs}=100pJ/bit/m^2$ . And we consider  $d_{toBS}$  instead of  $d_{toBS}^2$ . The optimal number of clusters for our configuration is 4.

Figure 6-22 to Figure 6-26 show the results for LEACH for random and grid distributions. In contrary to of EAD, and EEDS, the number of live nodes, throughput, Information throughput, and total consumed energy are the same for grid and random distribution. In LEACH, it is assumed that all nodes can hear the transmission of all other nodes. Therefore, any node can join any head in the network. Therefore, the distribution of nodes does not affect the clusters. For different runs, the distribution will not affect the performance of the protocol. The percentage of covered area in grid distribution is better than the random distribution. In grid distribution, nodes are well organized and they will cover larger area.



**Figure 6-22: Number of Live Nodes vs. Time for LEACH, random and Grid Topologies**

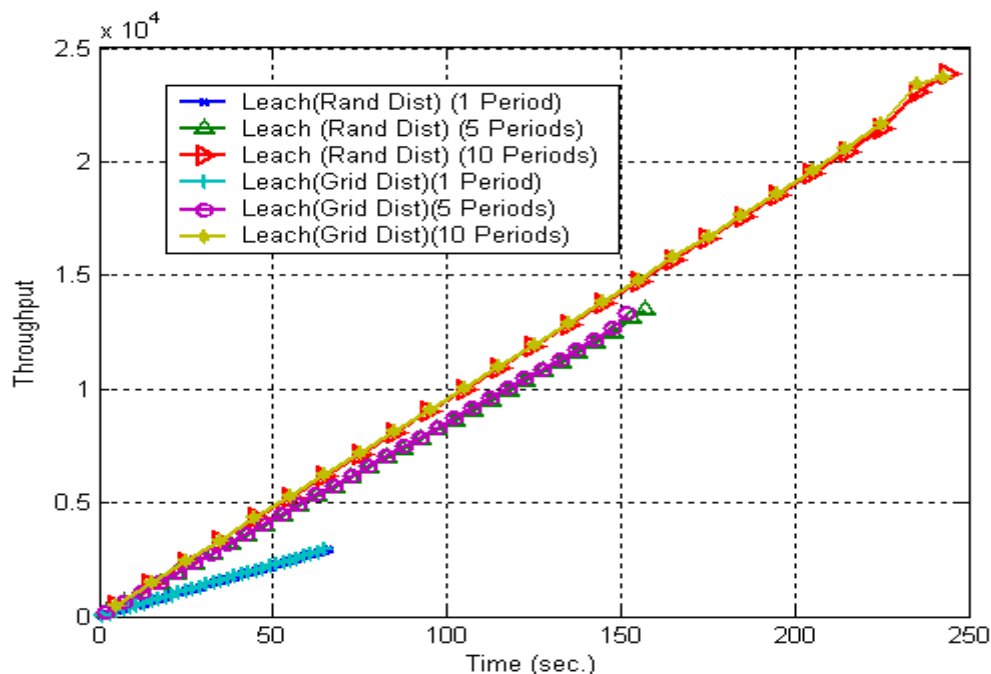


Figure 6-23: Throughput vs. Time for LEACH, random and Grid Topologies

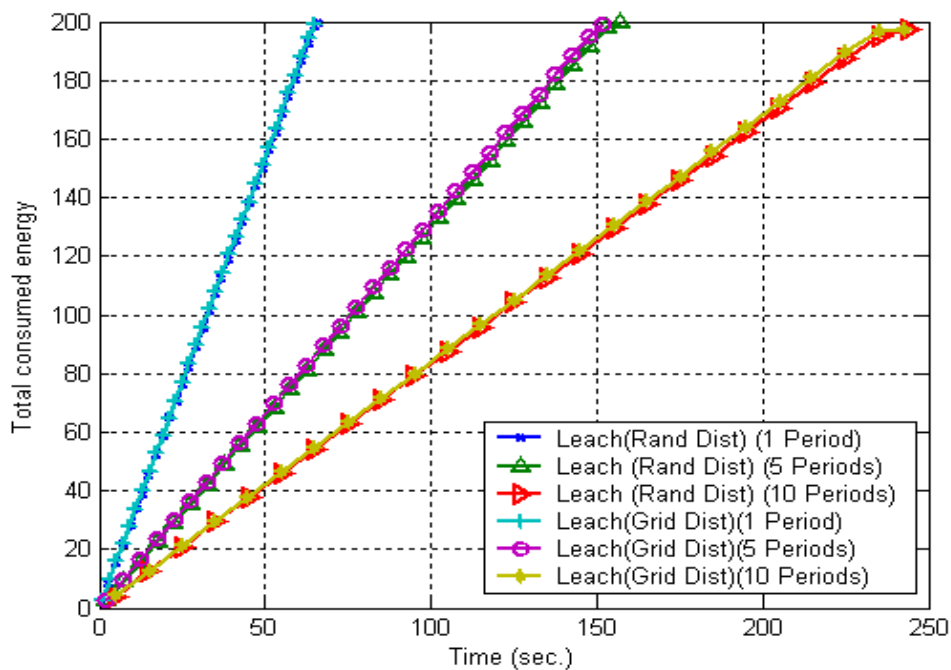
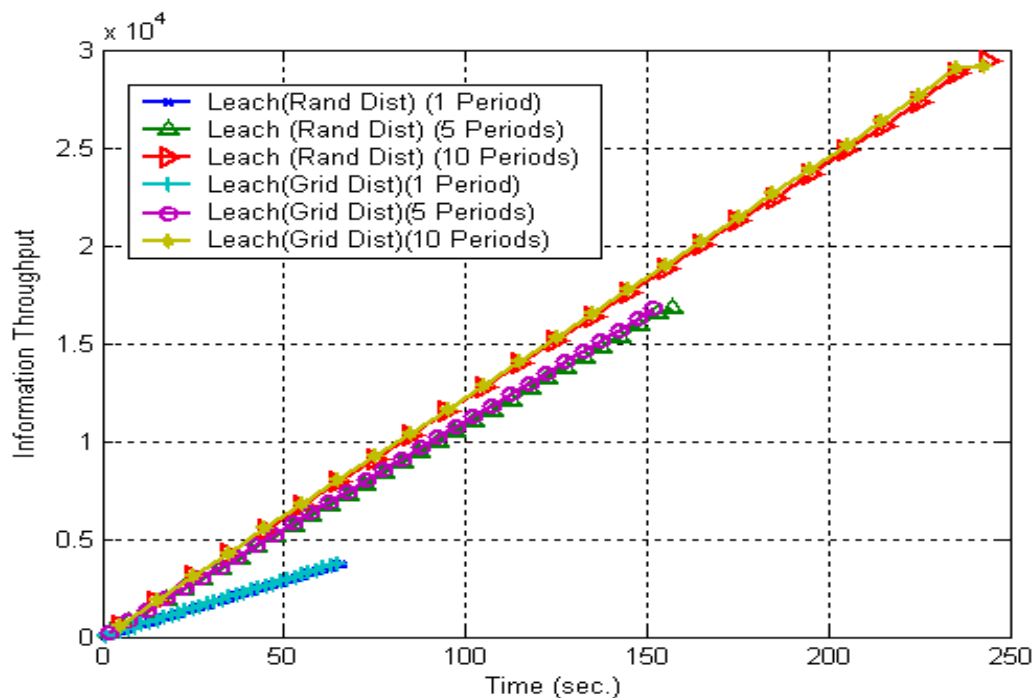
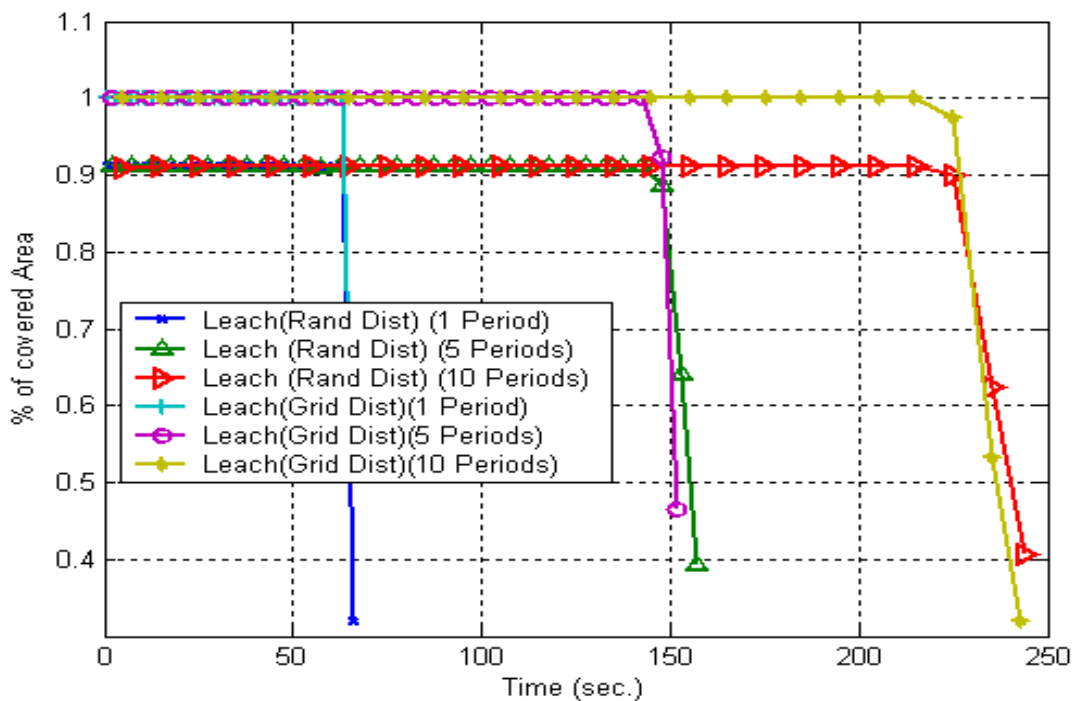


Figure 6-24: Total Consumed energy vs. Time for LEACH, random and Grid Topologies



**Figure 6-25: Information Throughput vs. Time for LEACH, random and Grid Topologies**

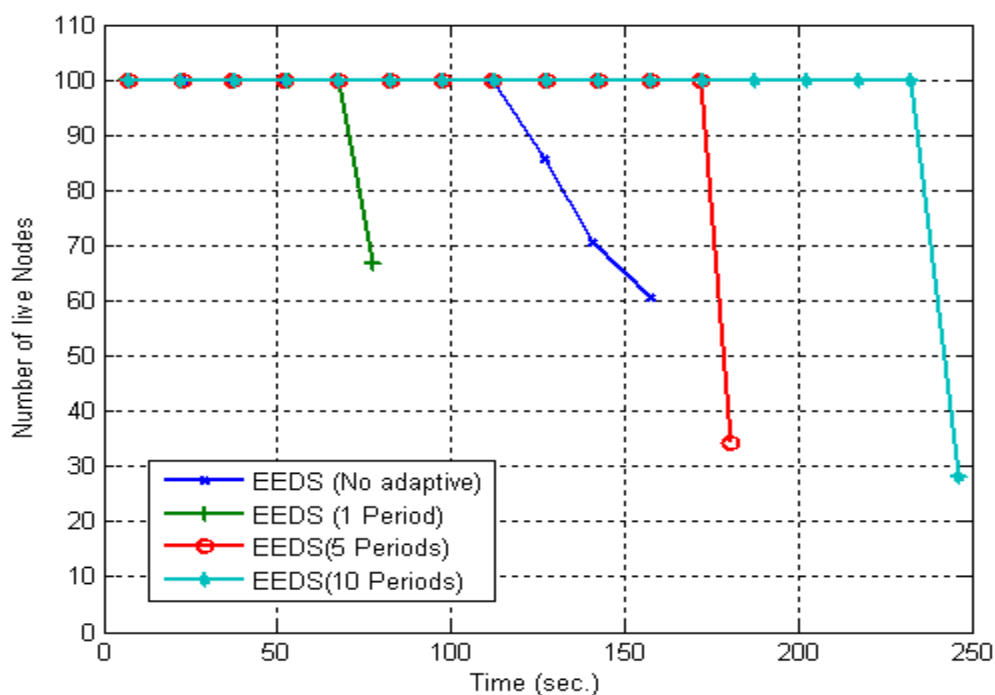


**Figure 6-26: Percentage of Covered area vs. Time for LEACH, random and Grid Topologies**

#### 6.4 THE EFFECT OF OVERHEAD PHASES IN THE PERFORMANCE OF EEDS AND GET

We observed from Figure 6-11 that increasing number of data transmission periods in a single round improves network lifetime for both grid and random topology. The network lifetime is improved with increasing number of data transmission periods because the same tree will be utilized more for larger number of data transmission periods, and hence the overhead consumed energy due to building tree and building schedule will be minimized. Therefore, to improve network lifetime, a larger number of data transmission periods must exist in a single round. At the same time, increasing number of data transmission periods in a single round will make energy utilization unfair among all nodes. Some nodes will act as non-parent nodes for longer time, and hence they will die early. In the worst case, if the single tree is utilized until the first node die (no adaptive), then the network lifetime will be minimized. Figure 6-27 shows a comparison between no adaptive case and the cases with different number of data transmission periods in a single round. We observe from Figure 6-27 that network lifetime in no adaptive case is higher than case with one data transmission period in a single round. At the same time, the network lifetime in no adaptive case is less than the case with five and ten data transmission periods in a single round. This is intuitive and can be justified as follows. In the case when a single data transmission period in a single round, a tree will be rebuilt for each data transmission period. Therefore more overhead energy will be consumed. The nodes will die early, and the network lifetime is minimal. In the no adaptive case, the overhead consumed energy will be minimized, but the same tree will be utilized longer. The same nodes will act as parents, and hence they will die early. Some nodes will be

isolated due to the die of these nodes. On the other hand, utilizing the same tree for large number of data transmission periods minimize the overhead consumed energy and changes the nodes that act as non-parents. Therefore, the nodes will take longer time to die. Table 6-1 and Table 6-2 shows a statistics of the overhead time of EEDS and GET protocols respectively.



**Figure 6-27: A comparison between no adaptive and different number of periods**

**Table 6-1 : Statistics of overhead time in EEDS**

Number of Data Transmission periods	Building Tree	Building Schedule	Data Transmission	Overhead Time	Percentage of overhead time
1	0.016233	0.010475	0.029463	0.026708	0.90649288
5	0.015898	0.009621	0.026834	0.025519	0.00013695
10	0.014837	0.008819	0.024252	0.023656	5.73705E-05

**Table 6-2: Statistics of overhead time in EEDS**

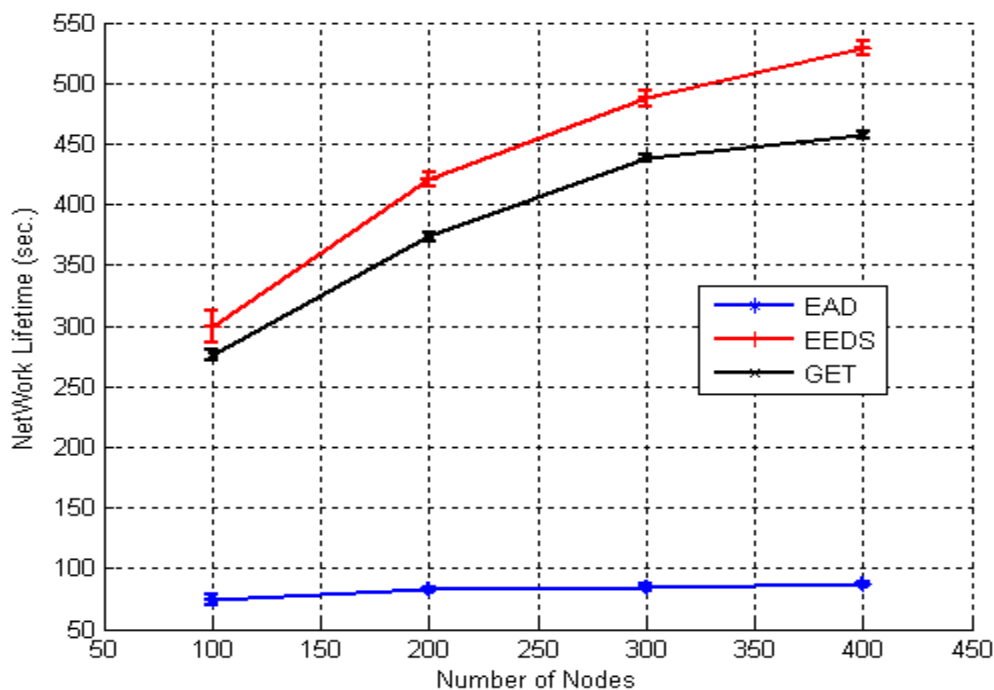
Number of Data Transmission periods	Selecting Gateway	Building Tree	Building Schedule	Data Transmission	Overhead Time	Percentage of Overhead Time
1	0.010205	0.018188	0.009564	0.034811	0.037957	1.090373732
5	0.008805	0.015531	0.009807	0.033939	0.034143	0.000231756
10	0.007341	0.015255	0.009887	0.032287	0.032483	0.000104878

### 6.5 PERFORMANCE EVALUATION OF THE EAD, EEDS AND GET UNDER DIFFERENT NETWORK SIZE

In this section, we compare the performance of the EEDS, GET and EAD under different network size. We examine the performance of the protocols assuming different number of nodes (100, 200, 300, 400) that are randomly deployed in a monitored area with dimension  $100 \times 100 \text{ m}^2$ . In each case, the sink is positioned at the center of the monitored area; location (50,50). Furthermore, we implement the protocols with different number of data transmission periods; 1, 5 and 10, Simulation parameters presented in Table 4-1 are reused here.

Figure 6-28 shows the network life time versus the number of nodes for 10 data transmission periods. We observed that increasing number of nodes improves the network life time for EEDS, GET and EAD. The improvement in network life time in EEDS and GET is significant, while the improvement is minimal in EAD. In EEDS and GET, a schedule based mechanism to forward data is implemented; increasing the number of nodes will decrease the probability of a node to be non-leaf. The number of leaf nodes will increase. They spent most of their time in sleeping state, and hence they consume little energy. They will take more time to die. On the other hand, although

increasing number of nodes in EAD will decrease the probability of a node to be non-leaf, all leaf nodes will consume more energy since a random mechanism is implemented to forward data. All nodes will stay ON to sense the channel to take their turns to transmit their data.



**Figure 6-28: Network Lifetime vs. Number of Nodes (10 Data transmission period)**

We observed from Figure 6-28 that increasing number of nodes improve network life time which improves throughput as shown in Figure 6-29. We observe in Figure 6-29 that increasing number of nodes improves throughput. Moreover, increasing number of nodes increases number of gateways which will improve also the throughput. Therefore, although network lifetime in EEDS is higher than in GET, the throughput in GET is better than in EEDS,



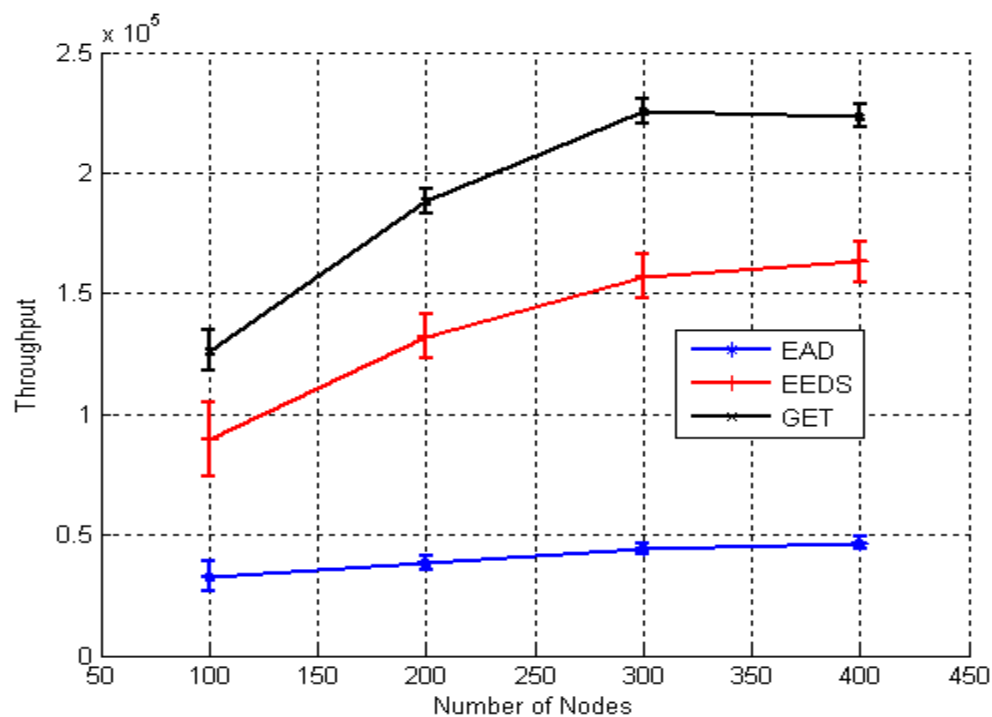


Figure 6-29: Throughput vs. Number of Nodes (10 Data transmission period)

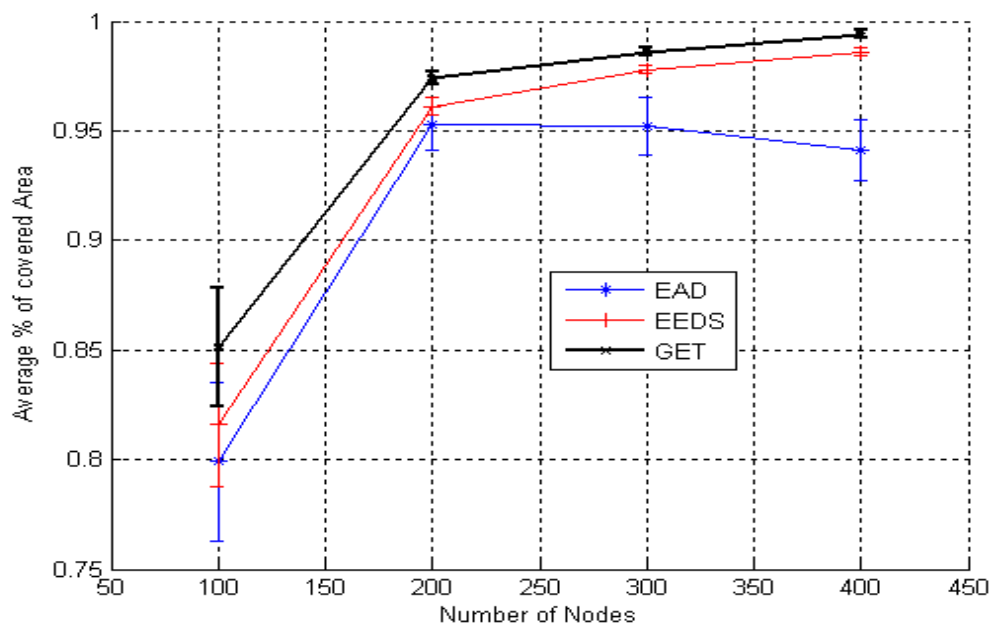


Figure 6-30: Covered Area vs. Number of Nodes (10 Data transmission period)

Increasing number of nodes distributed in the monitored area will increase network density. a specific grid will be covered by more sensors. If of one of these sensors die, the grid is still covered by other sensors. Therefore, the percentage of covered area is improved by increasing number of nodes as shown in Figure 6-30.

Table 6-3 to Table 6-5 show a summary of improvement in EEDS when compared with EAD, and GET for different number of nodes.

**Table 6-3 : A summary of improvement in EEDS compared with EAD, GET (1 Data Transmission period)**

	200 nodes		300 nodes		400 Nodes	
	EAD	GET	EAD	GET	EAD	GET
Network Lifetime	53.6%	8.5%	58.6%	9.4%	61%	9.5%
Throughput	-2.3%	-39.3%	-2.9%	-33.5%	-3.3%	-32.1%
Total Consumed Energy	-3.9%	-9%	-3.2%	-7.7%	-2.9%	-7%

**Table 6-4: A summary of improvement in EEDS compared with EAD, GET (5 Data Transmission periods)**

	200 nodes		300 nodes		400 Nodes	
	EAD	GET	EAD	GET	EAD	GET
Network Lifetime	237.4%	10%	267.4%	12%	279.3%	12.4%
Throughput	121.2%	-38.8%	121.3%	-35.3%	116%	-34.6%
Total Consumed Energy	8.6%	-5.8%	9%	-5%	7.7%	-4.7%

**Table 6-5: A summary of improvement in EEDS compared with EAD, GET (5 Data Transmission periods)**

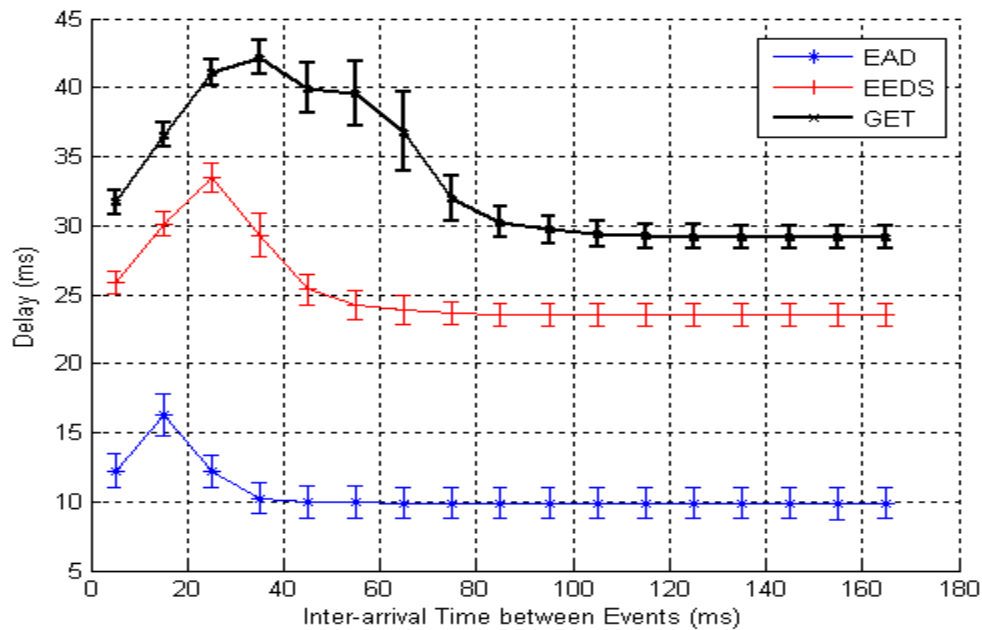
	200 nodes		300 nodes		400 Nodes	
	EAD	GET	EAD	GET	EAD	GET
Network Lifetime	409.4%	12.8%	474.3%	11.2%	506.5%	15.9%
Throughput	244.2%	-29.7%	255.5%	-30.5%	248.9%	-27.1%
Total Consumed Energy	13%	-1.4%	12.7%	-3.6%	11.2%	0.2%

## 6.6 PERFORMANCE EVALUATION OF EAD, EEDS, AND GET WITH DIFFERENT APPLICATIONS

In this section, we present the performance evaluation of EEDS, GET and EAD with different applications. We characterize the applications based on inter-arrival time between events. We assume 100 nodes distributed randomly in a monitored area with dimension:  $100 \times 100 \text{ m}^2$ . A sink is positioned in the center of the monitored area (50,50). Simulation parameters presented in Table 4-1 are reused here. The number of data transmission periods is 10. We evaluate the performance of the protocols assuming two scenarios. In the first scenario, the inter-arrival time between events is deterministic and fixed. In the second scenario the inter-arrival time between events is exponential random variable. For each configuration, we measure delay, network lifetime, percentage of detected events, and throughput. We define the delay as the time since the event occurs until the time at which the corresponding data packet reaches the sink. In other words, the delay is the sum of the time needed to forward the data packet to sink and the time interval from the occurrence of the event to the time at which the network starts forwarding packet to sink. To measure the percentage of detected events, we assume that the node can process a single event only at specific time. This is the event that occurs before the end of the current data transmission period. Therefore, all the events that occur before this event are considered as missed events. In other words, an event is missed if it occurs while the network is busy in forwarding data for the previous event.

### 6.6.1 PERFORMANCE EVALUATION OF THE EAD, EEDS AND GET ASSUMING DETERMINISTIC INTER-ARRIVAL TIME BETWEEN EVENTS

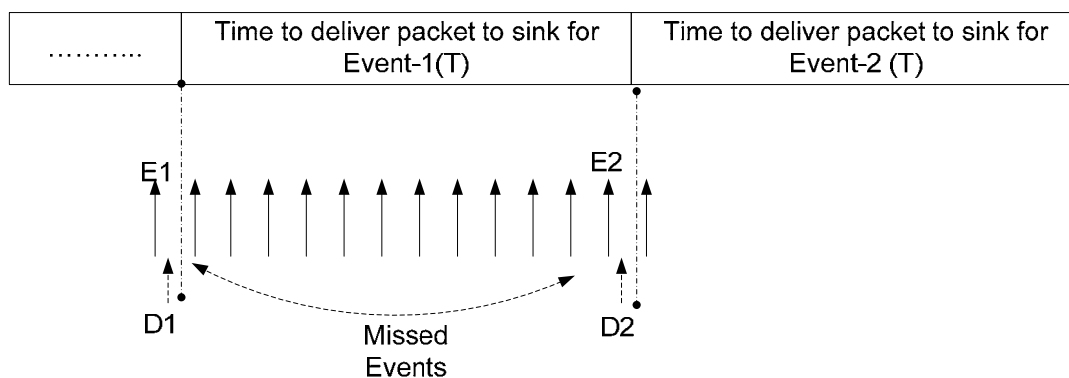
In this section, we assume that the inter-arrival time between events is deterministic and fixed for the whole network lifetime.



**Figure 6-31: Delay vs. Interval Time between Events, Fixed, (10 Data Transmission Periods)**

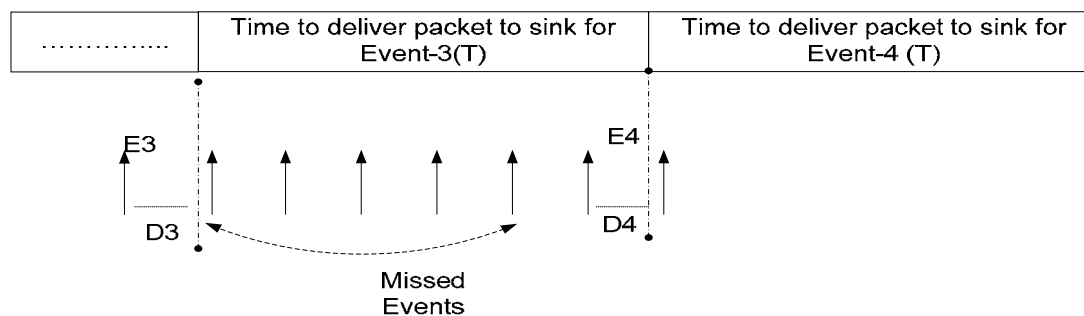
Figure 6-31 shows the delay versus inter-arrival time. We observe that for a short inter-arrival time the delay is low. Increasing inter-arrival time will increase delay until it reaches maximum. Then, further increasing for inter-arrival time will decrease delay. This can be explained as follows. Let  $D_k$  refers to the time interval between the time at which event- $k$  ( $E_k$ ) occurs to the time at which the network starts forwarding the data packet of  $E_k$  to sink, and  $T$  refers to time needed to forward packet to sink. When inter-arrival time between events is short then  $D_k$  will be short. And hence the delay will be

small. For example, Figure 6-32 shows the case where the inter arrival time between events is short. In this case, only events  $E_1$  and  $E_2$  will be processed, and the remaining events will be missed. The delay for event one ( $E_1$ ) is  $D_1+T$ , while the delay for event two ( $E_2$ ) is  $D_2+T$



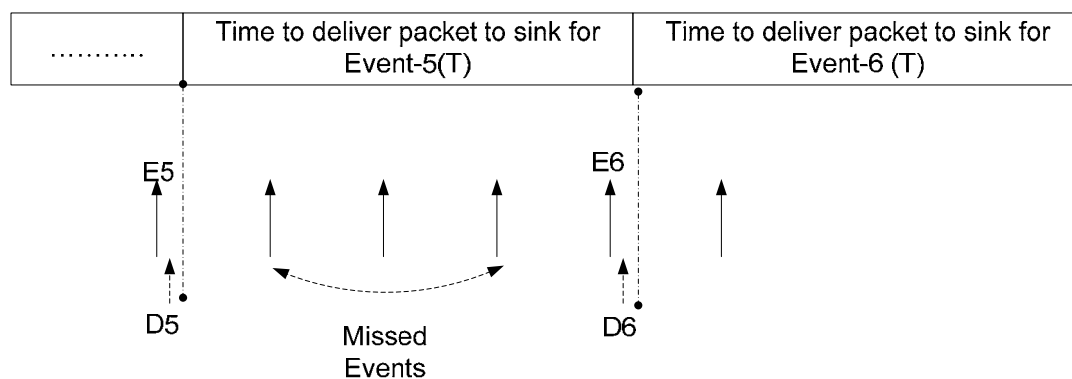
**Figure 6-32: Delay for processed events case-(1)**

Further increasing of inter-arrival time will increase delay as shown in Figure 6-33. In Figure 6-33, the inter-arrival time between events is double the inter-arrival time between events in Figure 6-32. In Figure 6-33,  $E_3$  and  $E_4$  will be processed and the remaining events will be missed. The delay for  $E_3$  is  $D_3+T$ , while it is  $D_4+T$  for  $E_4$ . We observe from Figure 6-32 and Figure 6-33 that  $D_3 > D_1$  and  $D_4 > D_2$ . Therefore, the delay for events in Figure 6-33 is higher than the delay for events in Figure 6-32



**Figure 6-33: Delay for processed events case (2)**

Further increasing of the inter-arrival time between events will minimize delay as shown in Figure 6-34. Inter-arrival time in Figure 6-34 is triple the inter-arrival time in Figure 6-32.  $E_5$  and  $E_6$  will be processed only in this case, and the remaining events will be missed. The delay for  $E_5$  is  $D_5+T$ , and the delay for  $E_6$  is  $D_6+T$ . We observe from Figure 6-33 and Figure 6-34 that  $D_5 < D_3$  and  $D_6 < D_4$ , therefore, delay in Figure 6-34 is less than delay in Figure 6-33.



**Figure 6-34: Delay for processed events case (3)**

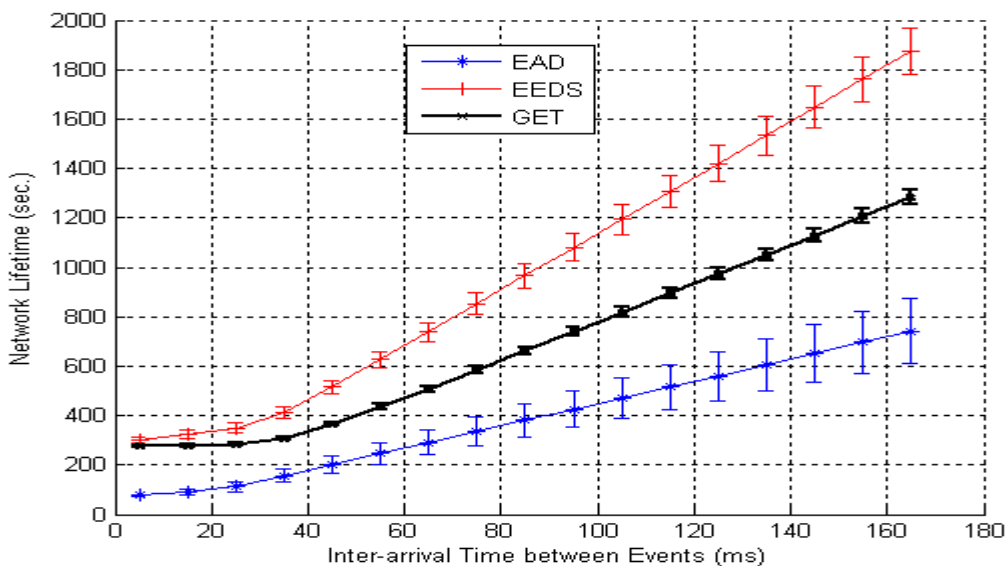


**Figure 6-35: Delay for processed events case (4)**

Increasing the inter-arrival time such that it becomes greater than the time needed to deliver the packet to sink will minimize the delay. Figure 6-35 shows this case, we observe from Figure 6-35 that the delay for events  $E_7$  and  $E_8$  is only the time needed to deliver packet to sink which is fixed. Therefore, further increasing of the inter-arrival

time will not change the delay. The delay will be stable and it is equal to the time needed to deliver packet to sink. This matches the simulation results shown in Figure 6-31

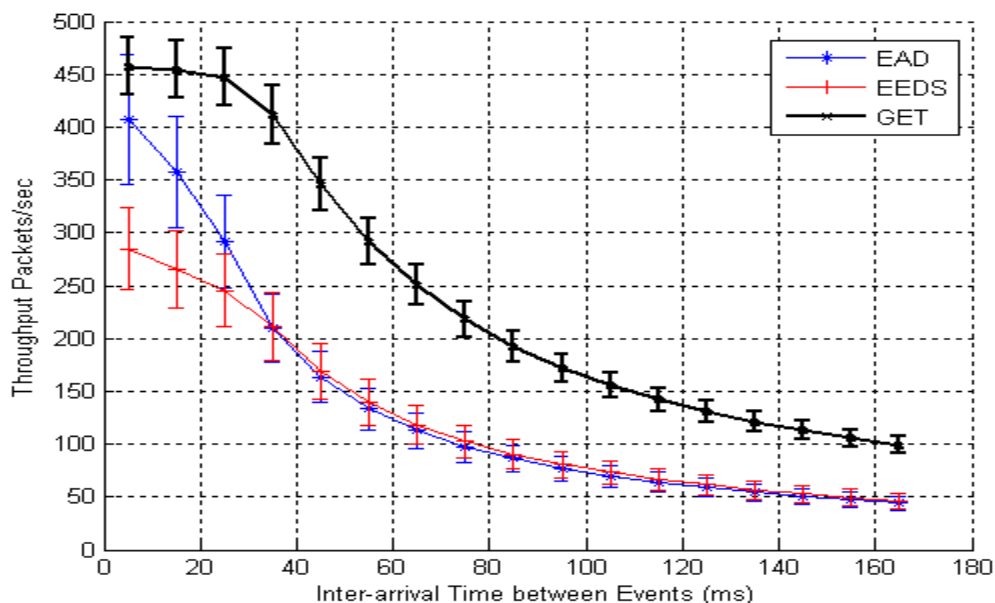
We observe from Figure 6-31 that the delay in EAD is less than the delay in both EEDS and GET. In EEDS and GET, there are extra phases such as selecting gateway and building schedule. These extra phases will increase the delay. Moreover, EEDS and GET are schedule based protocols, while EAD is random base protocols. The delay in random based protocols usually is less than delay in schedule-based protocols. In addition, in designing EEDS and GET we assumed that each branch; gateway and its associated nodes will use the same frequency. Each non-leaf node will build TDMA schedule for its children such that the TDMA frame will immediately begin after maximum  $TRT$  of its children. This will also increase the delay.



**Figure 6-36: Network lifetime vs. Inter-Arrival Time (10 Data Transmission Periods)**

Figure 6-36 shows network lifetime versus inter-arrival time between events. We observed that increasing inter-arrival time improves network lifetime. With higher inter-

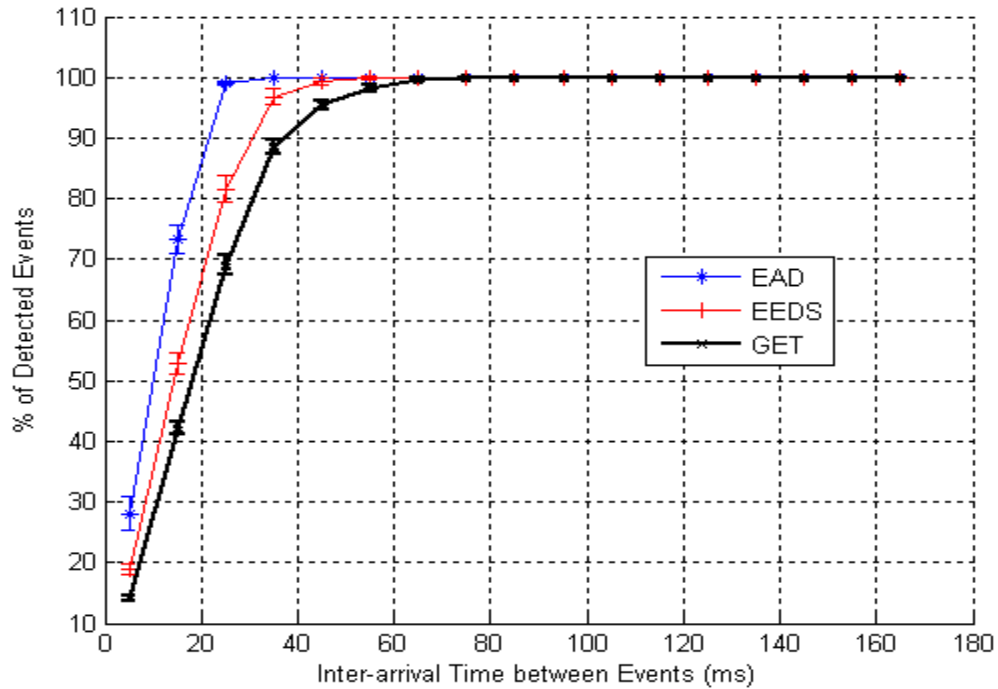
arrival time, the number of events will be fewer. Therefore, fewer data packets will be forwarded, and hence less energy will be consumed. Since fewer packets are delivered to sink, throughput decreases with high inter-arrival time as shown in Figure 6-37.



**Figure 6-37: Throughput (Packets/sec.) vs. Inter-Arrival Time (10 Data Transmission Periods)**

Figure 6-38 shows percentage of detected events versus inter-arrival time between events. We observe that when inter-arrival time between events is short the percentage of detected event is low, since more events will occurred while network is busy in forwarding data packet corresponding to previous event. These events will be missed. On the other hand, increasing of inter-arrival time will improve the percentage of detected events. The network will finish forwarding the data corresponding to the current event before the successive event occurs. This can be observed from Figure 6-32 to Figure 6-35. Number of missed events in Figure 6-32 is 11. it decreases to 6 in Figure 6-33, and it is zero in Figure 6-35.





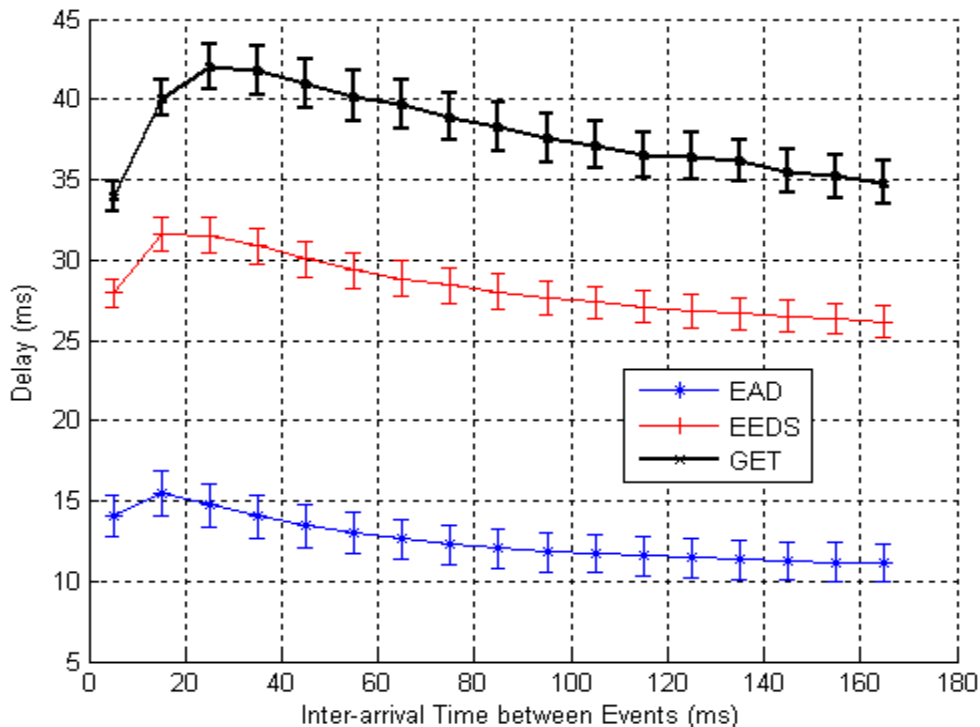
**Figure 6-38: Percentage of covered Area vs. Inter-Arrival Time (10 Data Transmission Periods)**

Moreover, since the delay in EAD is less than delay in EEDS and GET, with short inter-arrival time, the percentage of detected events in EAD is better than the percentage of detected events in EEDS and GET. In EAD, the network will finish forwarding data to sink before the successive event occurs, while in EEDS and GET a new event will occur while network is busy in forwarding data of the current event to sink.

### 6.6.2 PERFORMANCE EVALUATION OF THE EAD, EEDS AND GET ASSUMING RANDOM INTER-ARRIVAL TIME BETWEEN EVENTS

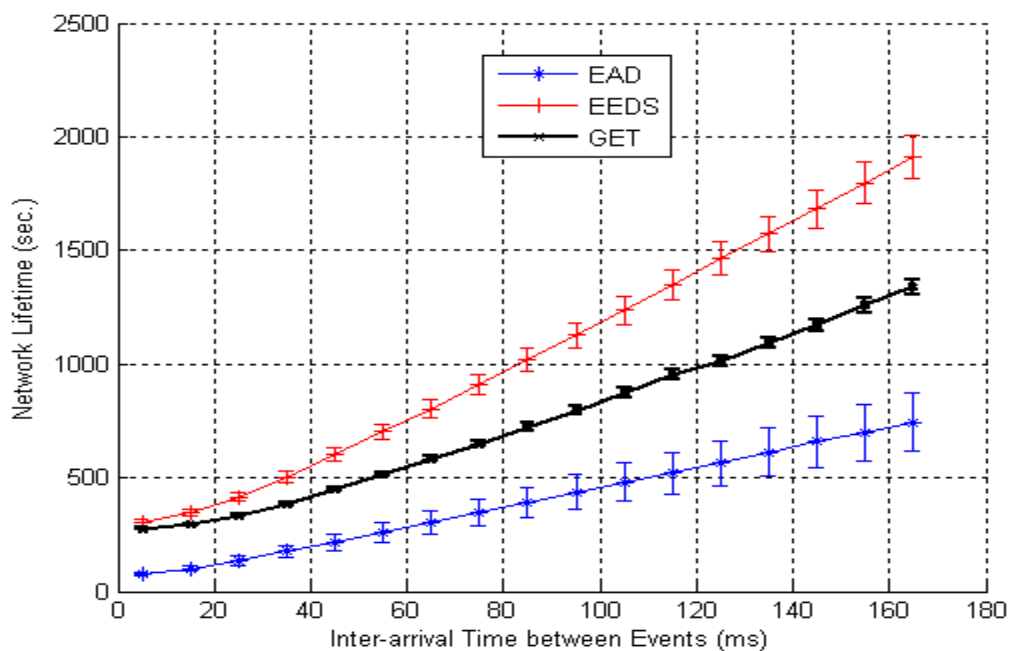
In this section, we investigate the performance of EAD, EEDS, and GET assuming random inter-arrival time between events. We assume the inter-arrival time to be exponential random variable. We investigate the performance of the protocols for

different cases. In each case, a specific mean of inter-arrival time is considered. A sequence of events is generated for each case.

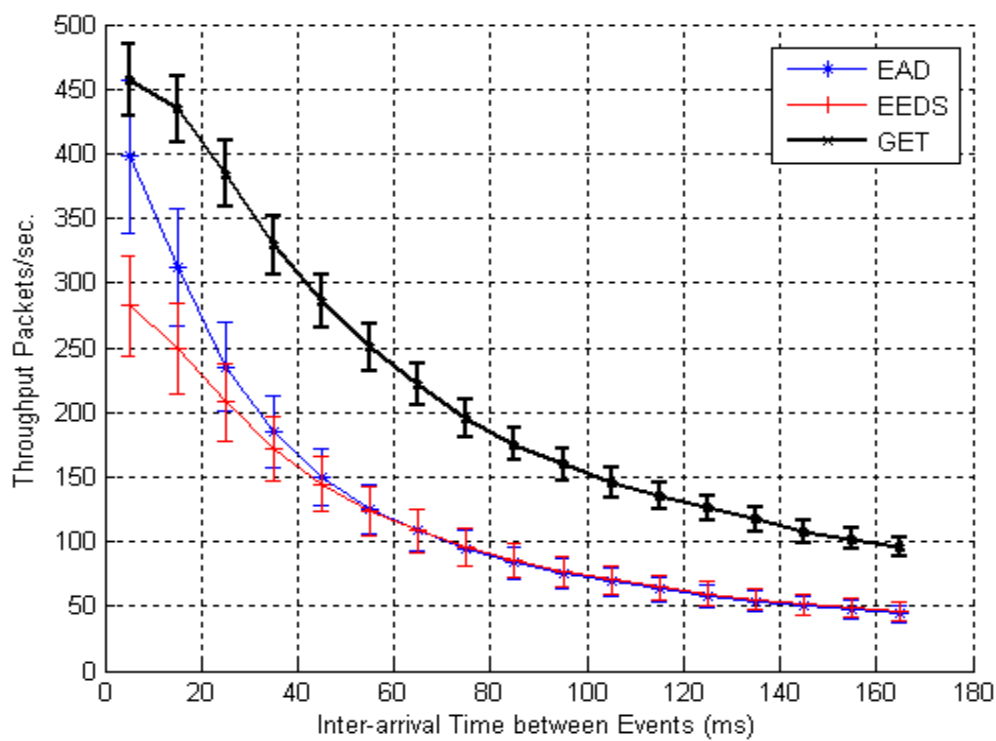


**Figure 6-39: Delay vs. Mean of Interval Time between Events, (10 Data Transmission Periods)**

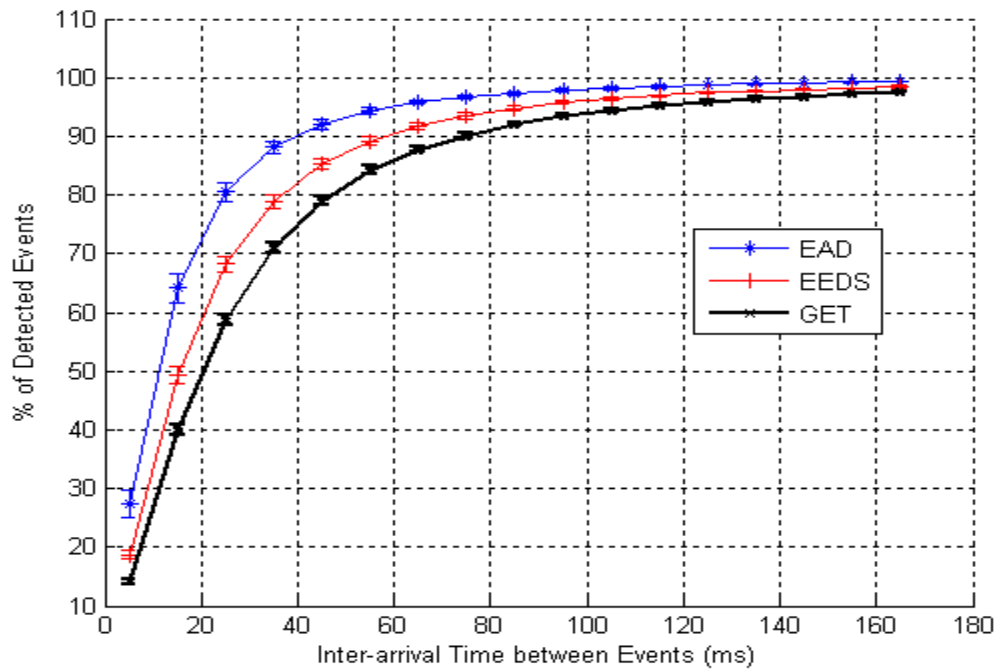
Figure 6-39 to Figure 6-42 show delay, network lifetime, throughput (Packets/sec), and percentage of detected events versus the mean of the inter-arrival time. We observed that the results obtained for random inter-arrival time shows similar behavior to the results obtained for fixed inter-arrival time.



**Figure 6-40: Network Lifetime vs. Mean of Interval Time between Events, (10 Data Transmission Periods)**



**Figure 6-41: Throughput (Packets/sec) vs. Mean of Interval Time between Events, (10 Data Transmission Periods)**



**Figure 6-42: Percentage of Detected Events vs. Mean of Interval Time between Events, (10 Data Transmission Periods)**

## 6.7 PERFORMANCE EVALUATION OF THE EEDS AND GET WITH MORE OPTIMAL SCHEDULE

We observe from Figure 6-31 and Figure 6-39 that the delay in EAD is less than the delay in EEDS and GET. The delay in EEDS and GET is higher, since the TDMA schedule built such that a non-leaf starts receiving from its children after the node with highest *TRT* is ready to transmit.

To minimize the delay in EEDS and GET, we modify the building schedule algorithm such that a node can start receiving data from some ready nodes, even though other nodes are not ready to transmit. The schedule is built such that when a parent finishes receiving from the current nodes, the other nodes will be ready to transmit. We build the schedule such that the node will be ON for one shot. In the new scheduling

algorithm, each node and its children will use different frequencies for communication. Therefore, we need more channels to forward data from nodes towards the sink. Figure 6-43 shows pseudo code for the optimal scheduling algorithm.

```

For leaf node j
    Transmit  $TRT_j$  to its parent
For non-leaf node j
    Receive  $TRT_i$  from all  $j$ 's children
     $S_j = \{ i : i \text{ is children for } j \}$ 
    Calculate  $TRR_j$  (Eq#1)
     $H = TRR_j$ 
     $L = H - T_t$ 
    Select node  $w$  from  $S_i$  with maximum TRR
     $T_w = H$ 
     $S_j = S_j - \{w\}$ 
    While ( $S_i \neq \emptyset$ )
    {
        Select node  $w$  from  $S_j$  with maximum TRR
        If ( $TRT_w \leq L$ )
             $T_w = L$ 
             $L = L - T_w$ 
        else
             $H = H + T_t$ 
             $T_w = H$  //  $T_t$  is time to transmit one data packet

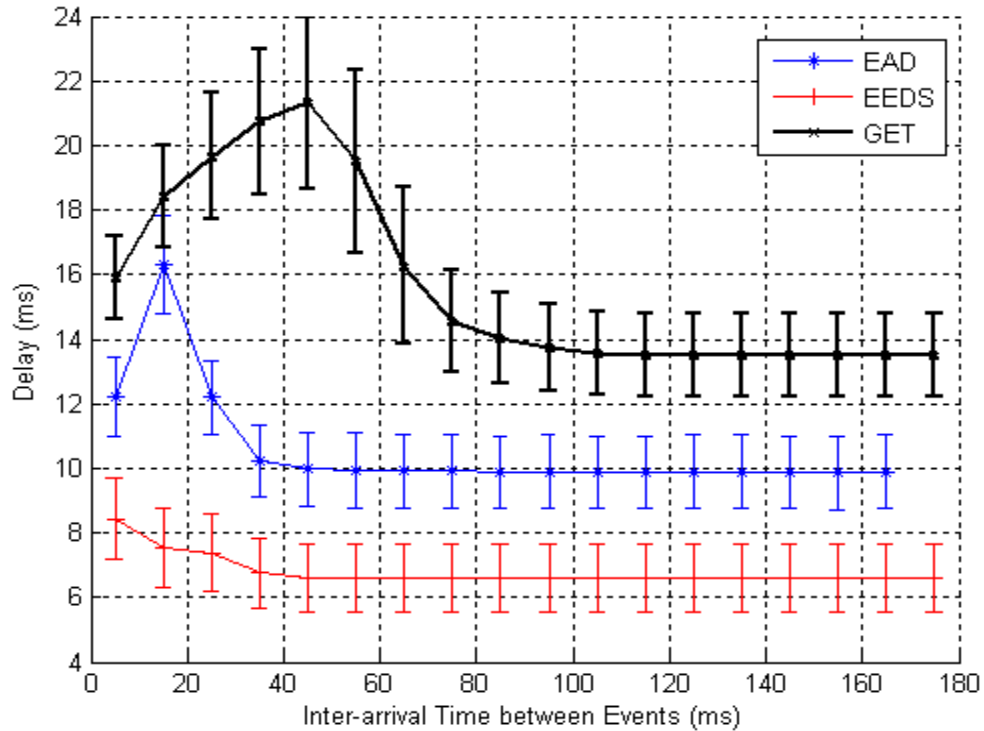
         $S_j = S_j - \{w\}$ 
    }
     $TRR_j = L$ 
     $TRT_j = H + T_t$ 
    Transmit  $TRT_j$  to the parent

```

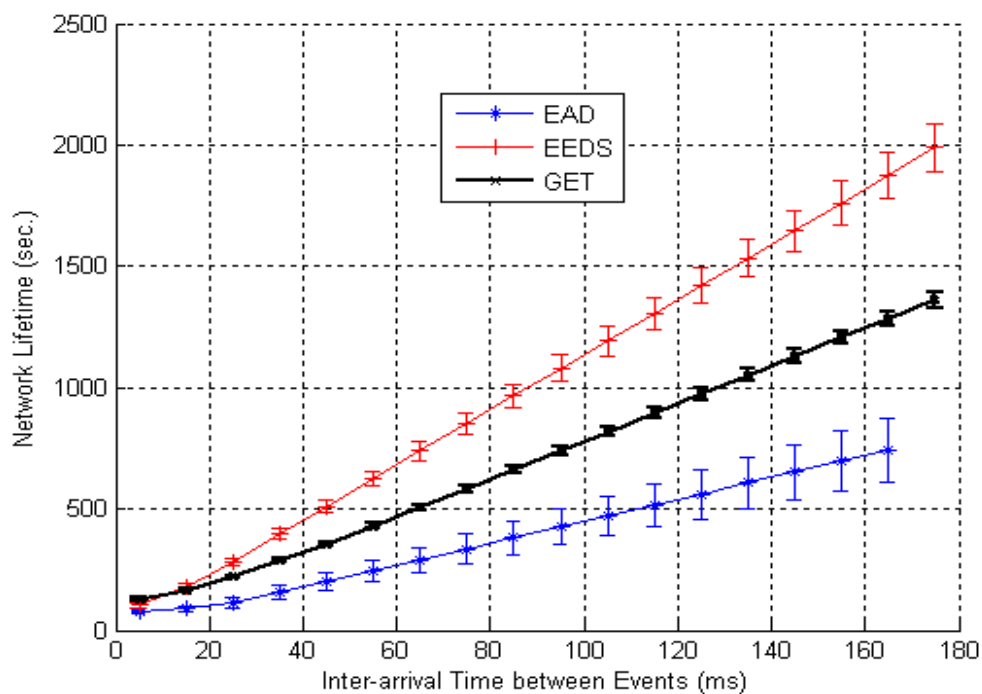
**Figure 6-43: Pseudo Code of the Optimal building Scheduling algorithm**

Figure 6-44 to Figure 6-47 show the results obtained when implementing the optimal building schedule algorithm in EEDS and GET. We assume a fixed inter-arrival time between events, and the simulation setup presented at the beginning of 6.6 is reused here. On contrary to the original scheduling algorithm, we observe that the delay in EEDS is less than the delay in EAD. For example with inter-arrival time between events equals to 20 msec, the delay in EEDS is about 7.5 msec when implementing the optimal

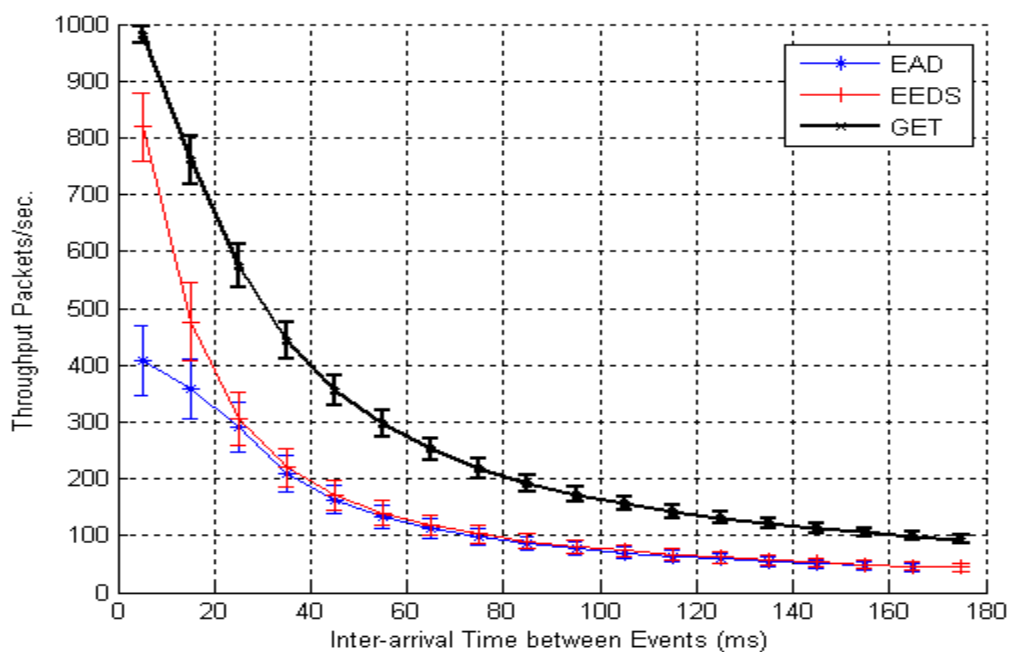
scheduling algorithm. On the other hand, it was 31 msec when implementing the original scheduling algorithm. The delay is reduced by about 75.8%. The cost for this reduction in the delay is the need of more channels and the need for a high efficient algorithm to assign different channels for the different cluster.



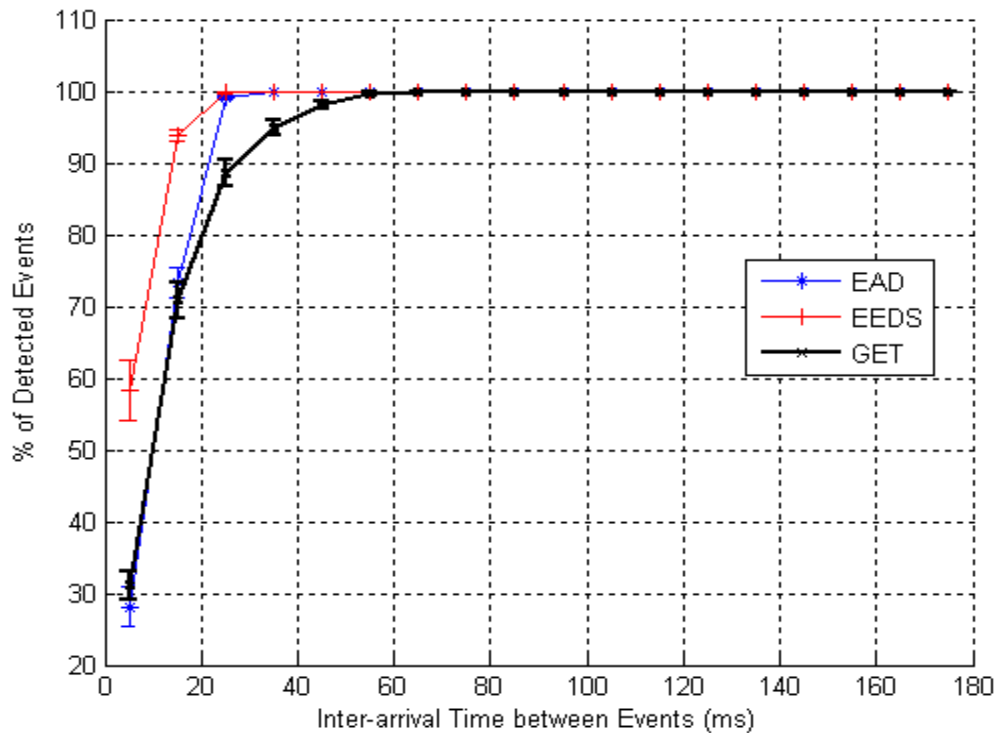
**Figure 6-44: Delay vs. Mean of Inter-arrival Time between Events, (10 Data Transmission Periods) with optimal schedule**



**Figure 6-45: Network Lifetime vs. Mean of Inter-arrival Time between Events, (10 Data Transmission Periods) with optimal schedule**



**Figure 6-46: Throughput (packets/sec) vs. Mean of Inter-arrival Time between Events, (10 Data Transmission Periods) with optimal schedule**



**Figure 6-47: % of Detected events vs. Mean of Inter-arrival Time between Events, (10 Data Transmission Periods) with optimal schedule**

## 6.8 CONCLUSION

In this chapter we discuss the performance evaluation of GET, EEDS and EAD assuming different network topology. A comparison among the different protocols is presented.

Firstly, we present a new throughput metric to investigate the performance of hierarchical protocols for WSN. The throughput in the new metric is defined as the amount of information delivered to sink. The information in each packet delivered to sink depends on the number of raw packets from which the packet is aggregated.



Moreover, we study in this chapter the effect of network topology on the performance of the different data forwarding protocols. It has been shown that EAD, EEDS and GET have behaved differently under the considered deployment methods. On the other hand, LEACH showed the same performance for both random and grid distribution which can be attributed to the fact that the number of cluster chosen in our simulation the optimal.

The effect of network size on the performance of the EEDS, GET, and EAD protocols is discussed we observed that increasing network size improve the performance of the protocols in terms of network lifetime, throughput, and percentage of covered area.

The performance of EAD, EEDS, and GET protocols assuming different inter-arrival time is discussed in this chapter. We observed that, EAD protocol outperforms EEDS and GET in terms of delay. On the other hand, EEDS and GET protocols outperform the EAD protocol in terms of network lifetime, and throughput.

A new building schedule algorithm is presents in this chapter. The new algorithm aims at minimizing delay in EEEDS and GETS. Performance evaluation shows that implementing the new building schedule algorithm within EEDS and GET minimize the delay.

So far, in the previous chapters, we discuss the performance of different protocols using simulation. It is necessary to compare the solutions obtained by our proposal with optimal solutions. Optimal solutions are solutions that are generated assuming that global information is known. Two techniques are usually used to find optimal solutions; heuristic algorithms such as simulated annealing, or integer linear programming (ILP). In our research, we adopt the ILP technique. In the next chapter, we will present the integer

linear programming (ILP) model. The model will be solved using LINGO solver assuming different cost functions and different network configuration. The results obtained by the ILP model are compared with the results obtained by simulation.

# Chapter Seven

## INTEGER LINEAR PROGRAMMING FORMULATION

### 7.1 INTRODUCTION

In the previous chapters, we evaluated the performance of our proposals using simulations. Moreover, we compared the performance of our proposals with the performance of EAD and LEACH. Usually, it is very important to check how much the results obtained by any protocol are close to optimal solutions. Optimal solutions are generated assuming that global information is known for a central agent. Usually, optimal solutions are generated by either heuristic techniques such as simulated annealing or integer linear programming (ILP) formulation. In our research, we use the ILP technique to generate the optimal solutions. We used ILP rather heuristic search because its results are more accurate. In ILP mathematical equations are solved, therefore solid results will be generated.

In this chapter, we propose an integer linear programming (ILP) model for building an optimal tree from all nodes toward the sink accompanied with a TDMA schedule for all nodes. The ILP model will be solved using LINGO solver.

We solve the model assuming different network configuration and different cost functions. The results obtained by solving the ILP model will be compared by the results obtained by simulation. Moreover, a comparison among the solutions obtained by solving the model using different cost functions will be discussed.

This chapter is organized as follows: section 7.2 presents the ILP formulation. Solving the ILP model and additional constraints to speed up the ILP solving process are presented in section 7.3. Section 7.4 presents performance evaluation of the solutions obtained by solving ILP model.

## 7.2 ILP MODEL FORMULATION

In general, each ILP model consists of cost function and a set of constraints. The cost function depends on the objective of the problem. On the other hand, ILP constraints depend on the physical characteristics of the problem. The optimal solution is the solution that satisfy the set of constraints and maximize (or minimize) the cost function.

In our problem, the solution of the ILP model is a tree and its associated TDMA schedule. Therefore, the constraints will represent the conditions that must be satisfied to build a tree and its associated TDMA schedule. In section 7.2.1, we present in details the constraints of our ILP model.

To identify our ILP cost function, we have to define our objectives. In our problem we have two main objectives: maximizing network lifetime and minimizing delay. In GET and EEDS, we maximize the network lifetime by building an energy efficient tree such that each node will select the parent with highest energy. Moreover, the network lifetime can be affected by the energy consumed by each node while transmitting data

packets which depends on the distance between each node and its parent. Therefore minimizing total consumed energy can be considered as a primary objective. The second main objective is to minimize the delay which is defined as the time needed to forward data packet from sensor node to sink. To formulate the two objectives, we propose four cost functions. The first cost function is identified such that a node with high energy will have more children. This cost function is similar to our selecting parent technique used in GET and EEDS. The second cost function will be minimizing the time to forward data packets from all nodes toward the sink (delay). To maximize the network lifetime and to minimize the delay, we propose the third cost function. The third cost function is a combination of the first and the second cost functions. To minimize the transmitting energy and assigning more children to high-energy node, we propose the fourth cost function. The fourth cost function is identified to minimize the total transmitting energy and assign more children for high-energy nodes. In section 7.2.2 we present the formulation of these cost functions

In our formulation, we assume that we have  $n$  nodes including the sink; the sink node is node number 1. The distance between each pair of nodes  $i$  and  $j$  is  $d_{ij}$ . The transmission range of each node is  $R$ . The residual energy of each node is  $E_i$

### 7.2.1 ILP CONSTRAINTS

In our problem, an energy efficient tree will be built; moreover a TDMA schedule will be built. Therefore, the constraints of our ILP model represent the constraints to build a tree and the associated TDMA schedule.

To represents a link between node  $i$  and node  $j$ , we define a binary variable:  $x_{ij}$ ,  $x_{ij}$  will be 1 if node  $j$  is a parent for node  $i$ , otherwise  $x_{ij}$  will be 0.

$$\begin{aligned} i &= 1, 2 \dots n \\ j &= 1, 2 \dots n \\ x_{ij} &\in \{1, 0\} \end{aligned} \quad (1)$$

Since we have a unidirectional tree, if node  $j$  is a parent of node  $i$ , then node  $i$  cannot be a parent of node  $j$ . Therefore,  $x_{ij} = 1$ , iff  $x_{ji} = 0$ , and vice versa. Moreover, it is possible that node  $i$  is not a parent of node  $j$  and node  $j$  is not a parent of node  $i$ , in this case both  $x_{ij}$  and  $x_{ji}$  are 0. Therefore, in general:

$$\begin{aligned} i &= 1, 2 \dots n \\ j &= 1, 2 \dots n \\ x_{ij} + x_{ji} &\leq 1 \end{aligned} \quad (2)$$

Each node (excluding the sink) has only one parent, therefore

$$\begin{aligned} i &= 2 \dots n \\ \sum_{j=1}^n x_{ij} &= 1 \end{aligned} \quad (3)$$

Since node 1 is assumed to be the sink, and it has no parent, then

$$\sum_{j=1}^n x_{1j} = 0 \quad (4)$$

All nodes will be connected to the tree. Therefore, the total number of links in the tree will be  $n-1$

$$\sum_{i=1}^n \sum_{j=1}^n x_{ij} = n-1 \quad (5)$$

Since we have a tree, and node-1 is the root of tree, there must be at least one link from any node to node-1:

$$\sum_{i=2}^n x_{i1} \geq 1 \quad (6)$$

A node can not be connected to itself; there will not be a link from a node to itself:

$$\begin{aligned} i &= 1..n \\ x_{ii} &= 0 \end{aligned} \quad (7)$$

Since nodes are usually distributed in wide area, a node can not communicate directly with all nodes. It can communicate only with nodes located within its transmission range ( $R$ ), for each pair of nodes  $i$  and  $j$ , if the distance between them ( $d_{ij}$ ) is longer than transmission range  $R$ , then  $x_{ij}$  must be zero, otherwise,  $x_{ij}$  can be zero or one :

$$\begin{aligned} i &= 1, 2 .. n \\ j &= 1, 2 .. n \\ x_{ij} &\leq \left\lfloor \frac{R}{d_{ij}} \right\rfloor \end{aligned} \quad (8)$$

Let  $EC_i$  be the energy consumed in each node  $i$  in a single data transmission phase due to receiving data packets from all its children. ( $EC_i$ ) depends on the number of children for node  $i$ . Number of children for node  $i$  ( $Num\_child_i$ ) is the sum of edges from all nodes to node  $i$ , it can be formulated by:

$$\begin{aligned} i &= 1, 2..n \\ Num\_child_i &= \sum_{j=1}^n x_{ji} \end{aligned} \quad (9)$$

As we have seen in section 4.4, the energy consumed to receive one data packet with  $k$  bits ( $E_{Rx}$ ) is calculated by :

$$E_{Rx} = kE_{elec} \quad (10)$$

Since a node will receive one data packet from each child in a single data transmission period,  $E_{Rx}$  will be the consumed energy in each node due to receiving a single data packet from one of its children, Therefore,  $EC_i$  can be computed by :

$$\begin{aligned}
 i &= 1, 2..n \\
 EC_i &= Num\_child_i * E_{Rx} \\
 &= kE_{elec} \sum_{j=1}^n x_{ji}
 \end{aligned} \tag{11}$$

Let  $ET_i$  be the energy consumed at each node  $i$  in a single data transmission phase due to transmitting a single data packet to its parent. As we see in section 4.4, the energy consumed during transmission of single data packets with  $k$  bits ( $E_{Tx}$ ) for a distance  $d$  meters can be calculated by:

$$ET_i = kE_{elec} + kE_{amp}d^2 \tag{12}$$

Therefore,  $(ET_i)$  depends on the distance ( $d_{ip_i}$ ) from a node to its parent, where  $p_i$  refers to the parent of node  $i$ . Since a node has only one parent, if node  $j$  is parent of node  $i$ , then  $x_{ij}$  will be 1 and  $x_{ik}$  will be 0, where  $k=1..n$  and  $k \neq j$ , therefore  $d_{ip_i}^2$  can be defined as

$$\begin{aligned}
 i &= 1..n \\
 d_{ip_i}^2 &= \sum_{j=1}^n x_{ij}d_{ij}^2
 \end{aligned} \tag{13}$$

Substituting (13) into (12):



$$i = 1, 2 \dots n$$

$$ET_i = kE_{elec} + kE_{amp} \sum_{j=1}^n x_{ij} d_{ij}^2 \quad (14)$$

For a node to work properly, it must have enough energy to receive from all its children and to transmit a single data packet to its parent. Therefore, the total consumed energy due receiving data packets from all children in addition to the consumed energy due to transmitting a single data packet must be less than the residual energy in the node.

$$i = 1, 2 \dots n$$

$$EC_i + ET_i \leq E_i \quad (15)$$

To formulate the data transmission schedule for all nodes, we consider a binary variable  $y$  to indicate whether there is data transmission at link  $ij$  (from node  $i$  to node  $j$ ) at a given time slot or not. For a node  $i$ ,  $y_{ijl}$  indicates whether node  $i$  is scheduled to transmit to node  $j$  at time slot  $l$  or not. If node  $i$  is scheduled to transmit to node  $j$  at time slot  $l$ , then  $y_{ijl} = 1$ . Otherwise,  $y_{ijl} = 0$ . If we have  $n$  nodes then we need at most  $n$  slots for all nodes to transmit.

$$i = 1, 2 \dots n$$

$$j = 1, 2 \dots n$$

$$l = 1, 2 \dots n$$

$$y_{ijl} \in \{1, 0\} \quad (16)$$

if  $y_{ijl} = 1$ , then the time at which node  $i$  will transmit ( $t_i$ ) will be  $l$ . in general

If there is no link between node  $i$  and node  $j$  ( $x_{ij}=0$ ), then there will no data transmission from node  $i$  to node  $j$  at any time slot;  $y_{ijl}=0$ . On the other hand, if there is a link between node  $i$  to node  $j$  ( $x_{ij}=1$ ), then there is at least a single time slot ( $k$ ) in which node  $i$  will transmit to node  $j$ .  $y_{ijk}$  will be 1 for that time slot and zeros for other time

slots. In other words, if  $x_{ij}$  is zero then  $y_{ijl}$  must be zero for any time slot  $l$ , however, if  $x_{ij}$  is one, then  $y_{ijl}$  will be 1 for a given time slot  $l$ . and it will be zero for the remaining time slots.

$$\begin{aligned}
 i &= 1, 2..n \\
 j &= 1, 2..n \\
 l &= 1, 2..n \\
 y_{ijl} &\leq x_{ij}
 \end{aligned} \tag{17}$$

Furthermore, we assumed that a node will transmit once in each data transmission period. Then

$$\begin{aligned}
 i &= 1, 2..n \\
 \sum_{j=1}^n \sum_{l=1}^n y_{ijl} &= 1
 \end{aligned} \tag{18}$$

The transmission time for node  $i$ ,  $t_i$  can be formulated by

$$\begin{aligned}
 i &= 1, 2..n \\
 t_i &= \sum_{j=1}^n \sum_{l=1}^n l y_{ijl}
 \end{aligned} \tag{19}$$

Where  $l$  points the time slot at which node  $i$  will transmit.

In our protocol, we assume that the parent node will transmit after it receives from all its children. Therefore, the transmission time ( $t_i$ ) for node  $i$  will be greater than the transmission time for node  $k$ , if node  $k$  is a child of node  $i$

$$\begin{aligned}
 i &= 1, 2..n \\
 k &= 1, 2..n \\
 t_i &\geq t_k + 1
 \end{aligned} \tag{20}$$

Substituting (19) into (20):

$$\begin{aligned}
i &= 1, 2, \dots, n \\
k &= 1, 2, \dots, n \\
\sum_{j=1}^n \sum_{l=1}^n ky_{ijl} &\geq \sum_{l=1}^n ky_{kil} + 1
\end{aligned} \tag{21}$$

At a given time slot  $l$ , A parent node  $i$  can receive from a single child only. If it receives from a child  $k$  at time slot  $l$ , then  $y_{kil}=1$ , and  $y_{jil}=0$  for all  $j \neq k$ , in general

$$\begin{aligned}
l &= 1, 2, \dots, n \\
i &= 1, 2, \dots, n \\
\sum_{j=1}^n y_{jil} &\leq 1
\end{aligned} \tag{22}$$

The following is a summary of the constraints that must be satisfied to build a tree and to build the associated data transmission schedule from all nodes towards the sink :

$$\begin{aligned}
i &= 1, 2 \dots n \\
j &= 1, 2 \dots n \\
x_{ij} &\in \{1, 0\}
\end{aligned} \tag{c-1}$$

$$\begin{aligned}
i &= 1, 2, \dots, n \\
j &= 1, 2, \dots, n \\
x_{ij} + x_{ji} &\leq 1
\end{aligned} \tag{c-2}$$

$$\begin{aligned}
i &= 2 \dots n \\
\sum_{j=1}^n x_{ij} &= 1
\end{aligned} \tag{c-3}$$

$$\sum_{j=1}^n x_{1j} = 0 \tag{c-4}$$

$$\sum_{i=1}^n \sum_{j=1}^n x_{ij} = n - 1 \tag{c-5}$$

$$\sum_{i=1}^n x_{i1} \geq 1 \quad \text{c- 6}$$

$$\begin{aligned} i &= 1, 2, \dots, n \\ x_{ii} &= 0 \end{aligned} \quad \text{c- 7}$$

$$\begin{aligned} i &= 1, 2, \dots, n \\ j &= 1, 2, \dots, n \\ x_{ij} &\leq \left\lfloor \frac{R}{d_{ij}} \right\rfloor \end{aligned} \quad \text{c- 8}$$

$$\begin{aligned} i &= 1, 2, \dots, n \\ EC_i + ET_i &\leq E_i \end{aligned} \quad \text{c- 9}$$

$$\begin{aligned} i &= 1, 2, \dots, n \\ j &= 1, 2, \dots, n \\ l &= 1, 2, \dots, n \\ y_{ijl} &\in \{1, 0\} \end{aligned} \quad \text{c- 10}$$

$$\begin{aligned} i &= 1, 2, \dots, n \\ j &= 1, 2, \dots, n \\ l &= 1, 2, \dots, n \\ y_{ijl} &\leq x_{ij} \end{aligned} \quad \text{c- 11}$$

$$\begin{aligned} i &= 1, 2, \dots, n \\ \sum_{j=1}^n \sum_{l=1}^n y_{ijl} &= 1 \end{aligned} \quad \text{c- 12}$$

$$\begin{aligned} i &= 1, 2, \dots, n \\ k &= 1, 2, \dots, n \\ \sum_{j=1}^n \sum_{l=1}^n ky_{ijl} &\geq \sum_{l=1}^n ky_{kil} + 1 \end{aligned} \quad \text{c- 13}$$

$$l = 1, 2, \dots, n$$

$$i = 1, 2, \dots, n$$

c- 14

$$\sum_{j=1}^n y_{jil} \leq 1$$

### 7.2.2 ILP COST FUNCTIONS

We have proposed four cost functions for our ILP model. The first cost function will be identified such that a node with high energy will have more children. This cost function is identical to assumption in our protocols EEDS and GET. The second cost function is minimizing the time needed to forward data from all nodes to sink. i.e minimizing the time for the sink to transmit. The third cost function is a multi objectives function. It will be a combination of the first and second cost functions. The fourth cost function is identified to assign more children for high energy nodes and minimizing total consumed energy.

**The Cost Function for Assigning More Children for High Energy Nodes:** In our protocols EEDS and GET, we assume that each node will select a parent from its neighbors; a neighbor with highest energy will be selected as a parent. Therefore our objective will be: a parent with higher energy will have more children, while a node with lower energy will have fewer children. Since GET and EEDS works in rounds, assigning more energy for high energy nodes will balance energy consumption among nodes. A node which has high number of children in the current round will lose more energy. Therefore in the next round, its energy will be small; it will not have large number of

children. Moreover, assigning more nodes for high energy nodes will reduce data traffic by aggregating large number of packets into one packet.

The energy consumed in each node  $i$  due to receiving data packets from all its children ( $EC_i$ ) can be calculated using (11). Therefore,  $EC_i$  must be maximized for high-energy nodes and it must be minimized for low energy nodes. For a node  $i$ , if we maximize  $EC_iE_i$ , then a node  $i$  with higher energy will have more children, and vice versa, Therefore, our cost function will be maximizing the summation of  $EC_iE_i$  for all nodes:

$$\max \sum_{i=1}^n EC_iE_i \quad (23)$$

**The Cost Function for Minimizing Delay:** Our second cost function will be minimizing the delay. Minimization delay is very important in WSN applications. For example, in fire detection application it is very important to deliver the event to the sink in a very short time. Since we assume that each node will transmit after it receive from all its children, and since we assume that node 1 is the root of the tree. Then, to minimize delay we have to minimize the time at which node-1 can transmit (if it will transmit). Therefore, our second cost function will be:

$$\min t_1 \quad (24)$$

**Multi-Objective Cost Function for Assigning More Children For High Energy Nodes And Minimizing Delay:** The solution obtained by using the first cost function is an energy efficient tree. The tree will be built without consideration of the delay. On the other hand, the solution obtained by using the second cost function will be

a tree that achieves minimum delay. to obtain an energy efficient tree that achieves minimum delay, we propose the third cost function. The third cost function is a multi objectives function. It is a combination of the first and the second cost functions. In general, a multi objective function (*obj*) that is a combination of two objective functions *obj*<sub>1</sub> and *obj*<sub>2</sub> can be written as

$$obj = \alpha_1 * obj_1 + \alpha_2 * obj_2 \quad (25)$$

$\alpha_1, \alpha_2$  are the desired weights for each objective.

Our third cost function will be minimizing delay and assigning more children for high energy nodes. It can be formulated as

$$\max \quad \alpha_1 \sum_{i=1}^n EC_i E_i - \alpha_2 t_1 \quad (26)$$

In our performance evaluation, we will consider different values for  $\alpha_1$  and  $\alpha_2$ .

**The Cost Function for Assigning More Children for High Energy Nodes And Minimizing Total Transmitting Energy:** The consumed energy in transmitting data packets affects the lifetime of any node. If the distance between the node and its parent is long, then the node will consume more energy in transmitting, and vice versa. Therefore, to minimize the transmitting energy, it is better for the node to select the closest neighbor as its parent. At the same time, selecting the same closest node as a parent in each round will make that node act as non-leaf node for the whole network lifetime. It loses more energy, and then it will die early. Therefore, it is better to change the nodes that act as parents in each round. This can be achieved using the first cost function. Hence to minimize the transmitting energy and to change nodes that act as non-leaf nodes in each round, we propose the fourth cost function. In the fourth cost function we aim at

assigning more children for a high energy node and minimizing total transmitting energy. The transmitting energy for each node is calculated by using Eq. (14). Then, the total transmitted energy ( $ET_{total}$ ) is

$$\begin{aligned}
 ET_{total} &= \sum_{i=2}^n ET_i \\
 &= \sum_{i=2}^n (kE_{elec} + kE_{amp} \sum_{j=1}^n x_{ij} d_{ij}^2) \\
 &= (n-1)kE_{elec} + kE_{amp} \sum_{i=2}^n \sum_{j=1}^n x_{ij} d_{ij}^2
 \end{aligned} \tag{27}$$

Our cost function will be a combination of the first cost function and minimizing the  $ET_{total}$ .

$$\max \quad \alpha_1 \sum_{i=1}^n EC_i E_i - \alpha_2 ET_{total} \tag{28}$$

which can be simplified to

$$\max \quad \alpha_1 \sum_{i=1}^n EC_i E_i - \alpha_2 \sum_{i=2}^n \sum_{j=1}^n x_{ij} d_{ij}^2 \tag{29}$$

### 7.3 SOLVING ILP MODEL

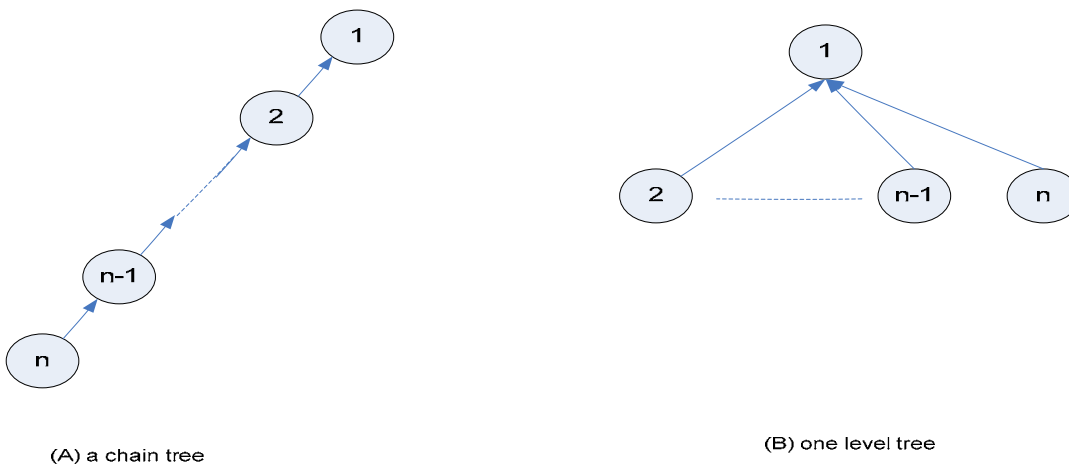
LINGO [7] solver is used to solve our model. Due to large number of constraints and variables in our ILP model, solving our model usually consume a lot of time. For example, for 30 nodes, the number of variables is 7760 and the number of constraints is 33611. The time needed to solve a model using the second cost function with the number of nodes equals to 30 is more than 80 hours.



To speed up solving our model, some common techniques in ILP formulation are used such as: adding other constraints to minimize the search region, and adding lower and upper bounds for some variables. The additional constraints must be selected such that they do not affect the solution of the model. In the following we discuss some of these constraints:

### 7.3.1 DELAY: LOWER AND UPPER BOUNDS FOR DELAY

If we have  $n$  nodes, then the upper bound of the delay is  $n-1$ . This occurred in two cases. The first case is shown in Figure 7-1-A, where a degenerate tree is built from all nodes towards the sink (node 1). The delay in the chain tree is  $n-1$ , because we assume that each node will wait until it receives a packet from its child then it sends the packet to its parent. The second case is shown in Figure 7-1-B, where a one level tree is constructed from all nodes towards the sink. In one level tree, each node needs one time slot to send its packet to sink. Since we have  $n-1$  nodes connected to sink, the minimum delay for one level tree will be  $n-1$ .

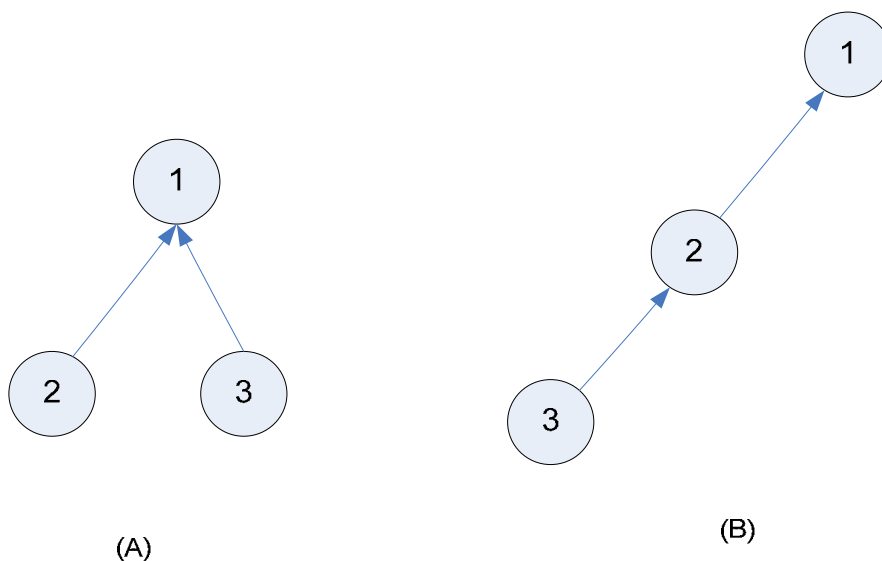


**Figure 7-1: degenerate and one Level Trees**

Since the upper bound of delay in a network of size  $n$  is  $n-1$ , the upper bound for the transmission time of the sink ( node-1) will be  $n$  . Therefore:

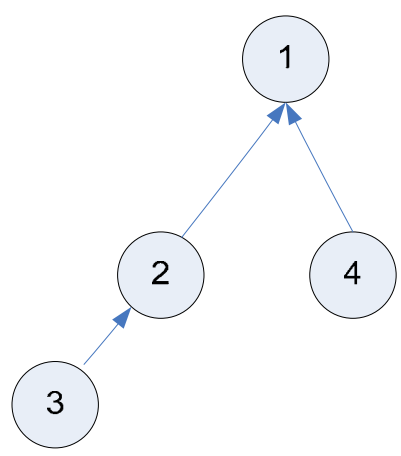
$$t_1 \leq n \quad (30)$$

To calculate the lower bound of the delay in networks with different sizes, we assume that the network is completely connected. We adopt the following technique: To calculate the lower bound of the delay in a network with size  $n$ , we start with a delay equals to the minimum delay that can be achieved in a network with size  $n-1$ . We try to build a tree that achieves this delay. If we can find such a tree, then this delay will be the lower bound of the delay for the network with size  $n$ . Otherwise, we increment the minimum delay. We try to build a tree that achieves the new delay. If we can not build such tree we increment the delay again and so on. We start with a node with size 2. The minimum delay that can be achieved in this network is one time slot. In a network with size 3 we could not build a tree that achieves delay equals to 1, but we can build a tree that achieves a delay equal to 2 as shown in Figure 7-2



**Figure 7-2: Possible trees in a network with Three Nodes**

In a network with 4 nodes, a tree with delay equals to 2 can be achieved. This tree is build by connecting the fourth node to node-1 of tree-B of Figure 7-2. Such tree is shown in Figure 7-3. one possible schedule for this tree is: node-3 and node-4 will transmit at time slot 1 while node 2 will transmit at time slot 2.



**Figure 7-3: Possible trees in a network with four nodes**

In a network with five nodes, minimum delay equals to 2 can not be achieved. Assuming the tree shown in Figure 7-3, a new node can be connected to node-1, node-3 or node-4. In all cases the minimum delay equals 3. We repeat the same procedure for networks with different sizes. We come up with the results shown in Table 7-1. Table 7-1 shows lower bound of delay and transmission time for sink ( $t_1$ ) for networks with different sizes.

**Table 7-1: Lower bounds on Delay and ( $t_1$ )**

Number of Nodes	Lower bound of Delay	Lower bound of ( $t_1$ )
2	1	2
3-4	2	3
5-8	3	4
9-16	4	5

From data shown in Table 7-1, we observed that a pattern exist between number of nodes in the tree and the minimum delay. This can be formulated as :

$$t_1 \geq \lfloor \log_2 n \rfloor + 1 \quad (31)$$

### 7.3.2 UPPER BOUND OF NUMBER OF LINKS IN SUB-NETWORK WITH THREE NODES

Since loops are not allowed in any tree, then number of links in a sub-network of three nodes must be less than or equal to 2 .

$$\begin{aligned} i &= 1..n \\ j &= 1..n \\ k &= 1..n \\ x_{ij} + x_{ik} + x_{jk} &\leq 2 \end{aligned} \quad (32)$$

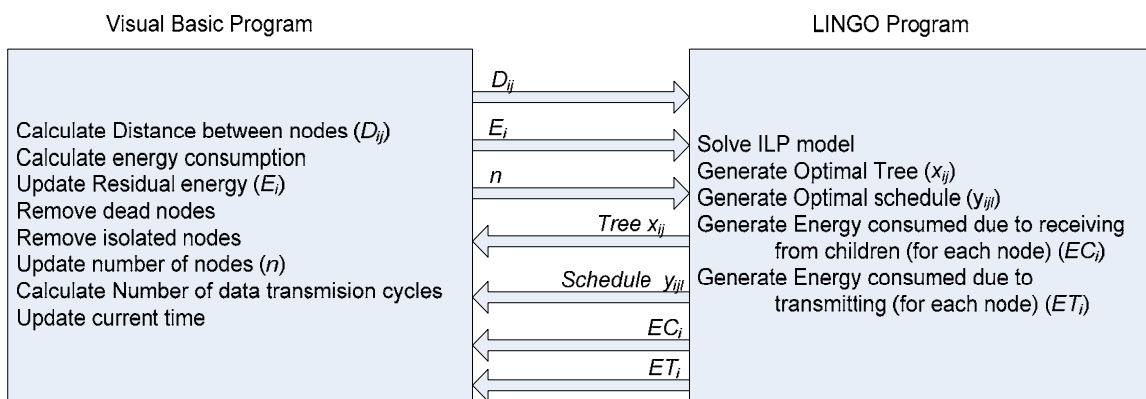
### 7.3.3 NO LINK BETWEEN TWO NODES WHICH DISTANCE BETWEEN THEM GREATER THAN TRANSMISSION RANGE.

Since a node can not communicate with nodes that are located out of its transmission range, there will be no link and no transmission will be occurred between this node and those nodes. Although this is included in constraint 8 (c-8), we observe that adding the following constraints will speed up the solution.

$$\begin{aligned} i &= 1..n \\ j &= 1..n \\ k &= 1..n \\ \text{if } d_{ij} &\geq R \\ x_{ij} &= 0 \\ y_{ijk} &= 0 \end{aligned} \quad (33)$$

## 7.4 PERFORMANCE EVALUATION USING ILP

The inputs that are needed to solve our ILP model are: number of nodes, distance between each pair of nodes and residual energy in each node. LINGO solver is a static tool. It solves the ILP model for a specific set of inputs. These inputs can not be changed during solving of the ILP model. On the other hand, our protocol works in rounds. Number of nodes and residual energy in each node vary from round to round. To generate results similar to results obtained by simulation, we have to solve our model in each round. An optimal solution must be generated according to number of nodes, residual energy in each node, and distance between each pair of nodes. To solve the model in each round with different inputs, we integrate LINGO solver with a VISUAL BASIC program. A block diagram showing the interaction between visual basic and LINGO is shown in Figure 7-4.



**Figure 7-4: Interaction between Visual Basic and LINGO**

At the beginning of each round, the visual basic program provides the LINGO solver with its input and calls it to solve the model. The LINGO solver generates the

optimal tree, the TDMA schedule, the energy consumed ( $EC_i$ ) in each node due to receiving from its children, and the energy consumed ( $ET_i$ ) in each node due to transmitting. According to  $EC_i$ , the maximum number of cycles that a tree can be utilized before the node die is calculated by visual basic program. Then the energy consumed at each node is calculated. Moreover, according to the schedule produced by LINGO solver, the time needed to forward data packets to sink is calculated. Both consumed energy and time are calculated by visual basic program taking into account the number of cycles in the round. In our experiments, we assume the tree is utilized for 1000 cycles. If 1000 cycles is greater than the maximum number of cycles that can be utilized, then we used the maximum number of cycles that can be utilized. We use 1000 cycles to minimize the time taking in solving the problem. This can be achieved by minimizing the number of times the LINGO solver called, because LINGO consumes a lot of time solving the model. Visual basic program will calculate the residual energy in each node. The dead and isolated nodes will be removed. The ILP solver will be called again with new inputs in the successive round.

**Table 7-2: Experiments parameters**

<i>PARAMETER</i>	<i>VALUE</i>
<i>Transmission Range (R)</i>	<i>15m</i>
<i>Electronics Energy (<math>E_{elec}</math>)</i>	<i>50nJ/bit</i>
<i>Amplifier Energy (<math>E_{amp}</math>)</i>	<i>100pJ/bit/m<sup>2</sup></i>
<i>Initial Energy in Each Node</i>	<i>100J</i>
<i>Initial Energy in Each Node</i>	<i>2J</i>
<i>Control Packet size</i>	<i>40 bytes</i>
<i>Data Packet size</i>	<i>100 bytes</i>

In our experiments, we assume different network configurations where 10 and 20 nodes are deployed randomly in different monitored areas. The sink is positioned at the

center of the monitored area. For each configuration, 30 different networks are tested. The results shown in this section are the average of the 30 different runs with 0.95 confidence level. We use the energy model presented in section 4.4. A summary of experiments parameters are shown in Table 7-2

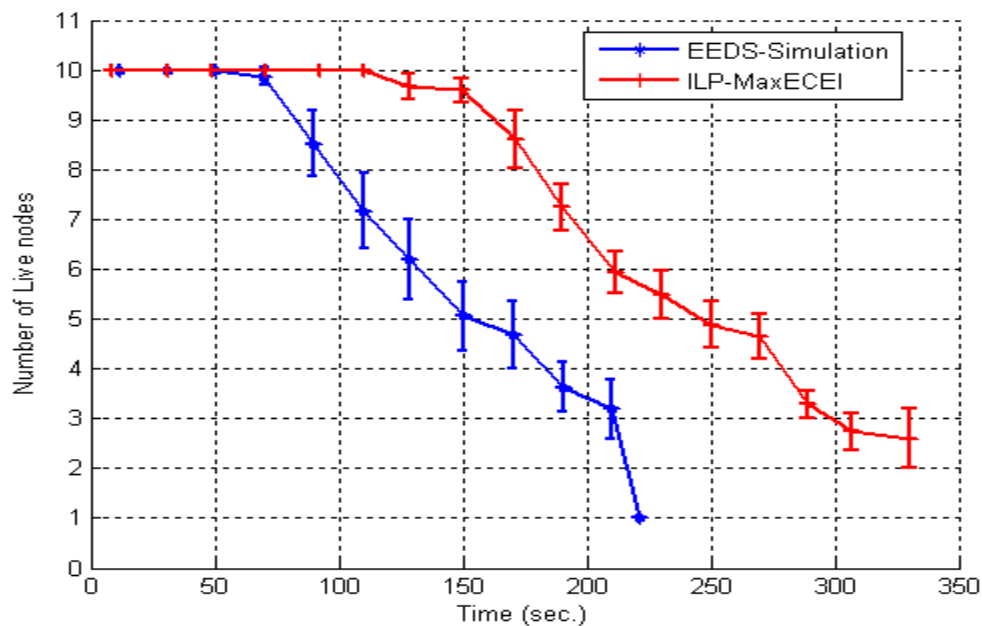
We compare the results obtained by simulation of EEDS by the results obtained by solving the ILP model using the first cost function moreover. Moreover, we compare the results obtained by solving the ILP model using the first cost function with the results obtained by solving the model using the second, third, and the fourth cost functions. Finally, we compare the solutions obtained by solving the ILP model using the first cost function for different number of nodes and different network density.

#### 7.4.1 A COMPARISON BETWEEN ILP MODEL SOLUTION RESULTS AND SIMULATION RESULTS

To compare results obtained by ILP model and results obtained by simulation, we use the first cost function ( $\max \sum_{i=1}^n EC_i E_i$ ). We compare the simulation results and ILP solution in terms of network lifetime, throughput, and percentage of covered area.

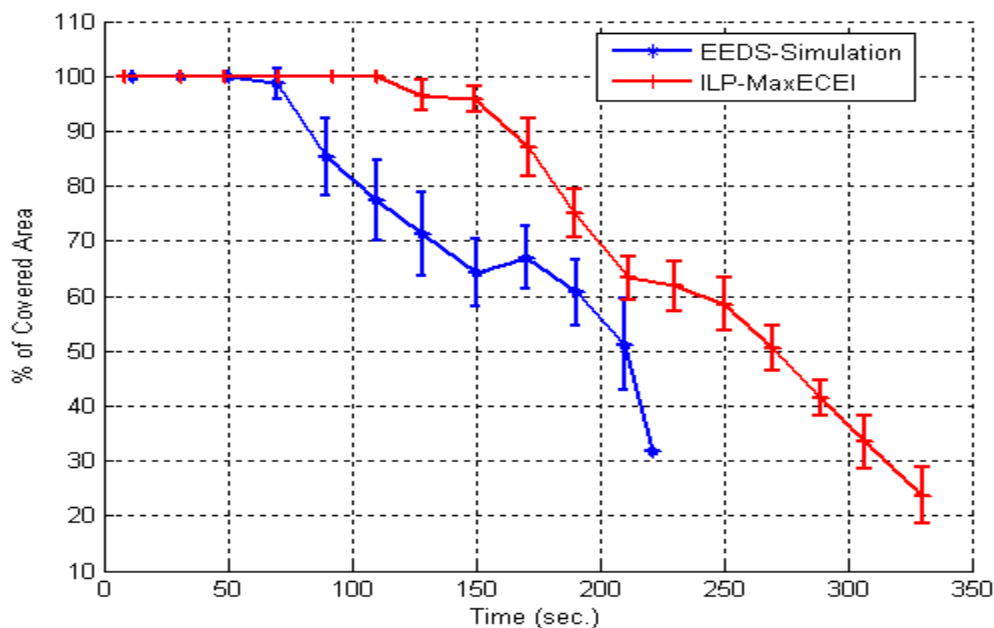
Figure 7-5 shows number of live nodes vs. time for ILP solution and simulation assuming 10 nodes are randomly distributed in an area with dimension 50x50 m<sup>2</sup>. We observed that although the two solutions shows similar behavior, the ILP solution outperforms the solution obtained by simulation. For example, the number of live nodes in ILP solution reaches 5 after 250 seconds. While the number of live nodes in solution obtained by simulation reaches 5 after 150 seconds. The ILP solution outperforms the

solution obtained by simulation by about 66%. The optimal solution is generated assuming that global knowledge of the network is known by the solver. On the other hand, in simulation, building tree process is initiated by the sink. Therefore the nodes that are closer to sink will announce themselves earlier. When a node decides to select a parent, it has local information about its neighbors that are closer to sink which they have already announced themselves. Therefore, it will select a parent among of these nodes. Some nodes may have higher energy, but they have not yet announced themselves because they are far away from the sink and they do not receive any broadcast message. They will not be considered as potential parents. Therefore some nodes may always be selected as parents, therefore they will die early.



**Figure 7-5: Number of Live nodes vs. Time, Number of nodes=10, Area=50x50**



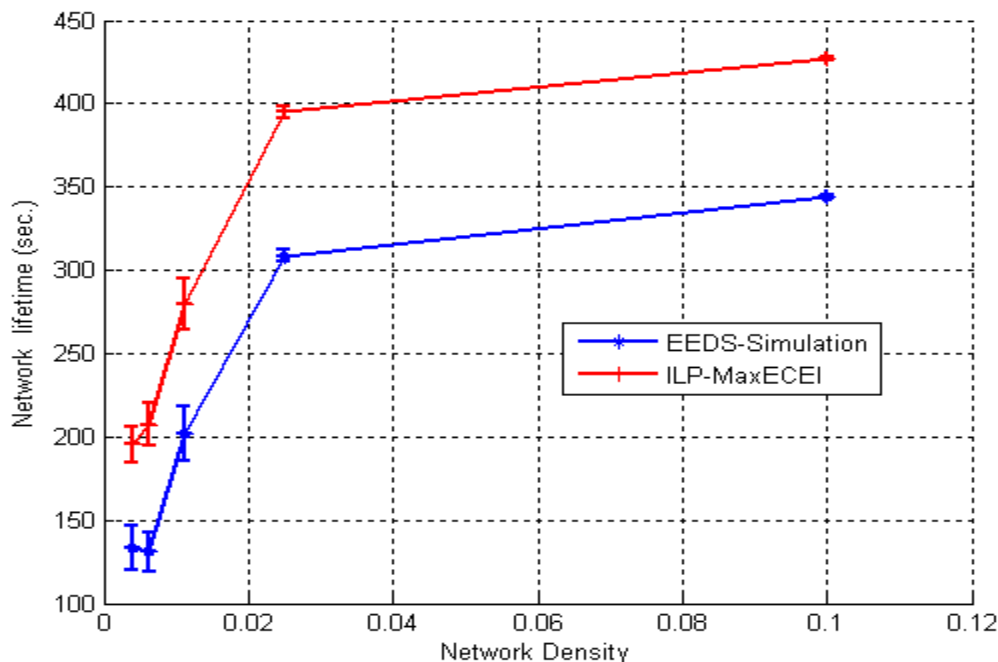


**Figure 7-6: Percentage of covered Area vs. time, Number of nodes=10, Area=50x50**

Percentage of covered area for both ILP solution and simulation is shown in Figure 7-6. We observed that the percentage of covered area in ILP solution is better than in simulation. The percentage of covered area in ILP solution becomes about 60% of the monitored area after 250 seconds, while the percentage of covered area in solution obtained by simulation reaches 60% of the monitored area after 190 seconds. The ILP solution outperforms the solution obtained by simulation by about 31.5%.

Moreover, we compare the results obtained by ILP solutions and results obtained by simulation assuming different network densities. We assume different network configurations where 10 nodes are randomly deployed in areas with different dimensions. The two solutions are compared in terms of network lifetime, total throughput, and delay. Network lifetime and total throughput are measured when the percentage of covered area

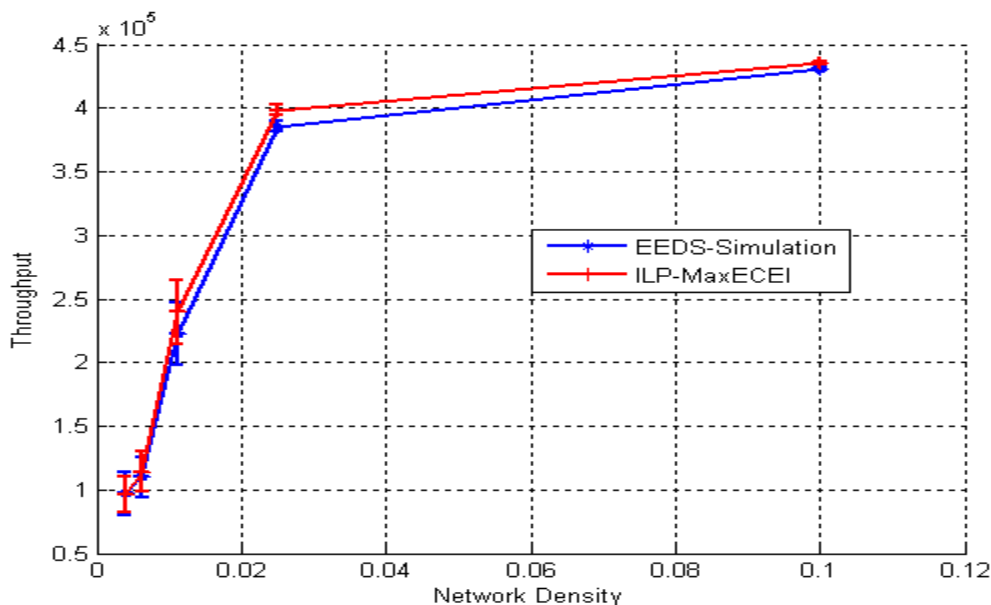
reaches 75% of the monitored area, while delay is measured as the average of the delay that achieved in the interval from the beginning of simulation until the first node die.



**Figure 7-7: Network Lifetime vs. Network Density, Number of nodes=10**

Figure 7-7 shows network lifetime versus network density for both ILP solution and simulation. We observed that the two solutions behave similarly. Increasing network density will improve network lifetime. When network density is high, the nodes will be closer and they will consume less energy when transmitting data packets. Moreover, with high network density, each node has more neighbors than network with low density. Therefore it is more likely for a node to select different neighbor each round. Energy consumption will be distributed among node's neighbor. They will take more time before they die. On the other hand, in low density networks, each node may have only one neighbor. This neighbor will be selected as a parent each round. It will consume more energy. Therefore it will die very early. We observe from Figure 7-7 that the network

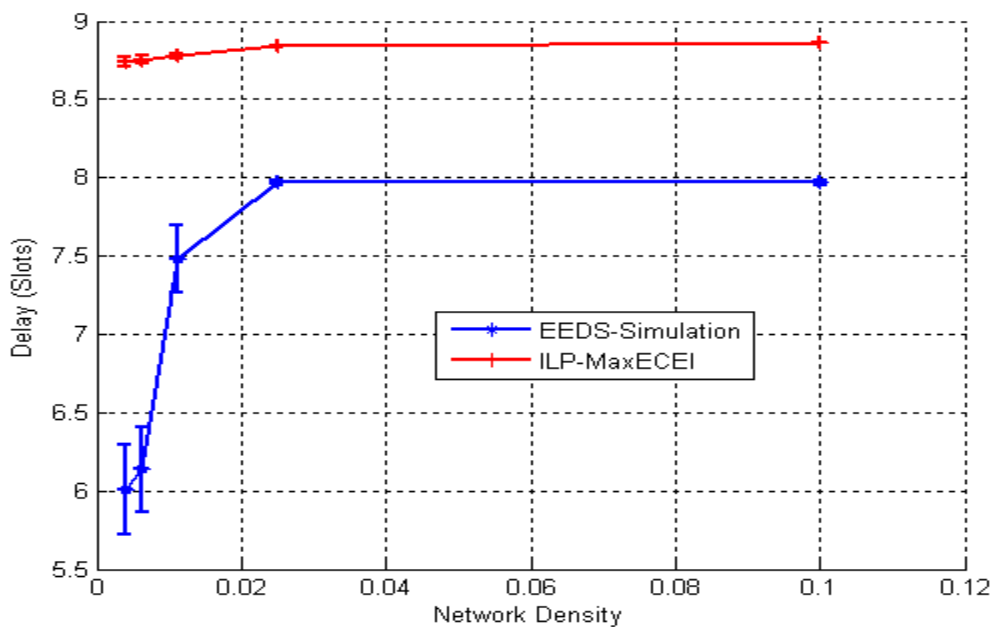
lifetime in ILP solution is higher than in simulation solution. For example, when network density is  $0.025 \text{ node/m}^2$ , the network lifetime in ILP solution is 395 second, while it is 308 seconds in solution obtained by simulation. The ILP solution outperforms the solution obtained by simulation by 28.3%.



**Figure 7-8: Throughput vs. Network Density, Number of nodes=10**

Figure 7-8 shows throughput versus network density for ILP solution and solution obtained by simulation. We observe that for both solutions throughput improves with higher network density. This can be attributed to the improvement in the network lifetime with higher network density. With higher network lifetime, more packets will be delivered to sink. Moreover, we observe from Figure 7-8 that throughput in ILP solution is improved a little bit compared with solutions obtained by simulation. For example, when network density is  $0.025 \text{ nodes/m}^2$ , the throughput achieved by ILP solution is about 398412 packets, while the throughput achieved by simulation is 385600 packets. In terms of throughput, ILP solution outperforms solution obtained by simulation by 3.3%.

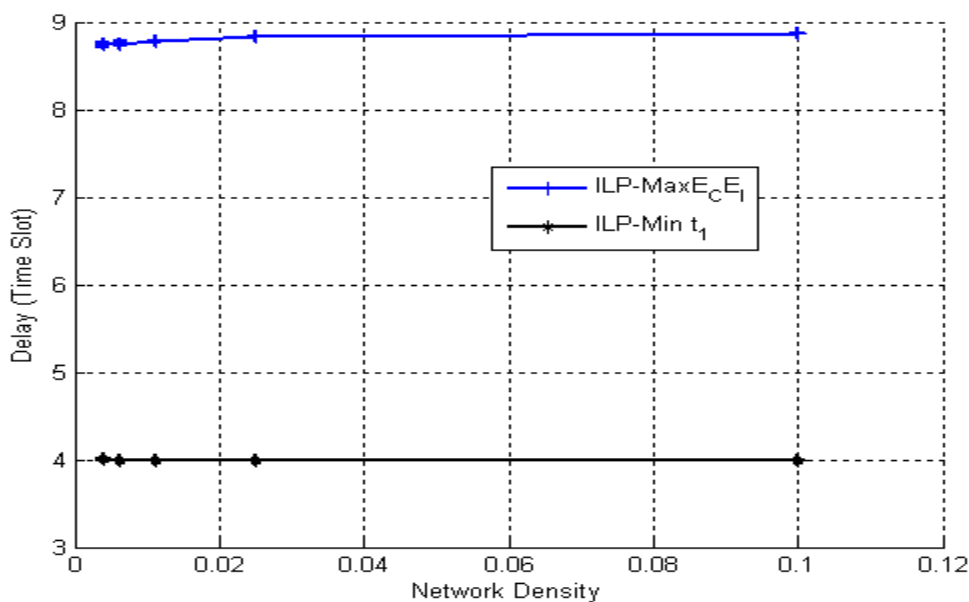
Although, network lifetime is improved in ILP solution by 28.3%, the throughput in ILP solution is improved by only 3.3%. This can be explained as follows. The ILP model is solved according to the cost function which is assigning more packets to high energy nodes, therefore the tree will be built according to this cost function. Meanwhile, the schedule will be built according to this tree. The schedule will not be optimal. The schedule may be built with empty slots. These empty slots will be counted in the network lifetime. On the other hand, these slots are not useful, since no data packets will be forwarded within these slots. On the other hand, when we design building schedule algorithm in EEDS, we try to build an optimal schedule. This is achieved by assigning contiguous time slots for each node as explained in section 3.2.3. Since the schedule obtained by ILP solution contains some empty slots, while the schedule built by simulation does not, the delay in ILP solution will be larger than in solutions obtained by simulation. This can be seen clearly in Figure 7-9.



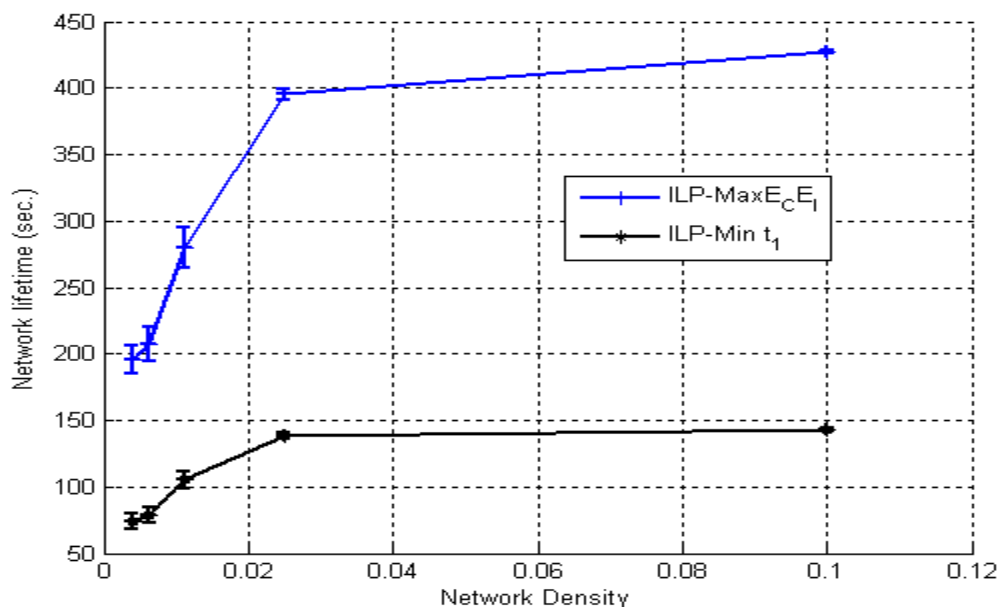
**Figure 7-9: Delay vs. Network Density, Number of nodes=10**

### 7.4.2 COMPARING THE RESULTS OF DIFFERENT COST FUNCTIONS

We observed from the last subsection that when the first function is used an energy efficient tree will be built without taking into account minimizing delay. On the other hand, our objective in the second cost function ( $Min t_1$ ) is to build a tree that achieves minimum delay. Therefore, the delay when using the second cost function will be lower than when using the first cost function as shown in Figure 7-10. As we mentioned in the previous section, the delay shown in Figure 7-10, is measured as the average of the delay that achieved in the interval from the beginning of simulation until the first node die. In other words, it is the average of the delay that achieved when number of nodes equals to 10. We observe from Figure 7-10 that the minimum delay that is achieved when using the second cost function is the lower bound of the delay when number of nodes is 10.

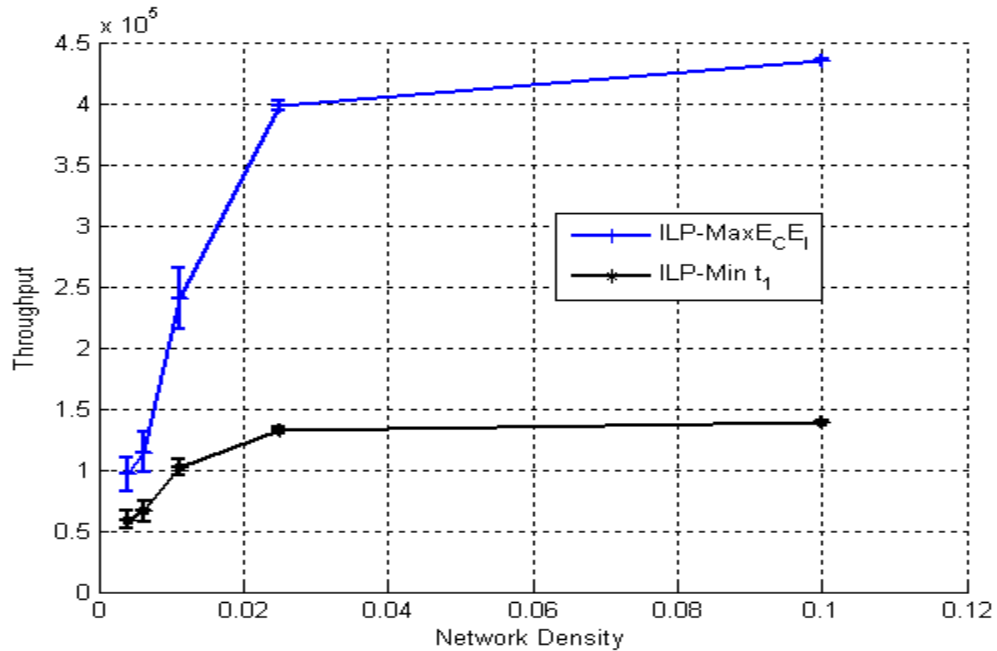


**Figure 7-10: Delay vs. Network Density, Number of Nodes =10;  $Max E_c E_b$ ,  $Min t_1$**



**Figure 7-11: Network lifetime vs. Density, Number of Nodes =10; Max  $E_c E_i$ , Min  $t_1$**

Since our objective when using the first cost function is to assign more children for high energy nodes, and since the residual energy of each node will differ from round to round, different nodes will work as parents in each round. Different trees will be built in each round. Energy consumption will be distributed fairly among different nodes. Therefore, network lifetime will be improved. However, our objective in the second cost function is to minimize delay. If no nodes die in the current round, then the number of nodes doesn't change in the successive round. Therefore, the same tree will be built each round. The same nodes will act as parents in each round. They will consume more energy, and they will die very early. Therefore, it is expected that the network lifetime when using the cost function that minimize delay will be less compared with the performance under the cost function that assign more children for high energy node. This can be shown Figure 7-11. Improvement in network lifetime will improve throughput as shown in Figure 7-12



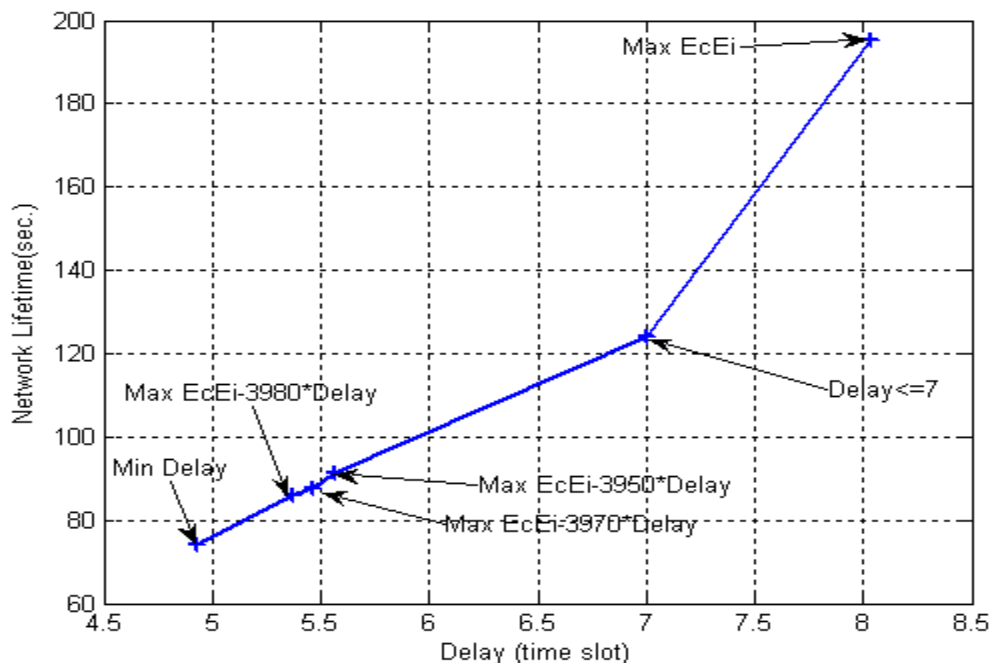
**Figure 7-12: Throughput vs. Density, Number of Nodes =10; MaxEcEi, MinD**

To achieve a high network lifetime accompanied with a reasonable delay, we use the third cost function (26):

$$\max \quad \alpha_1 \sum_{i=1}^n EC_i E_i - \alpha_2 t_1 \quad (26)$$

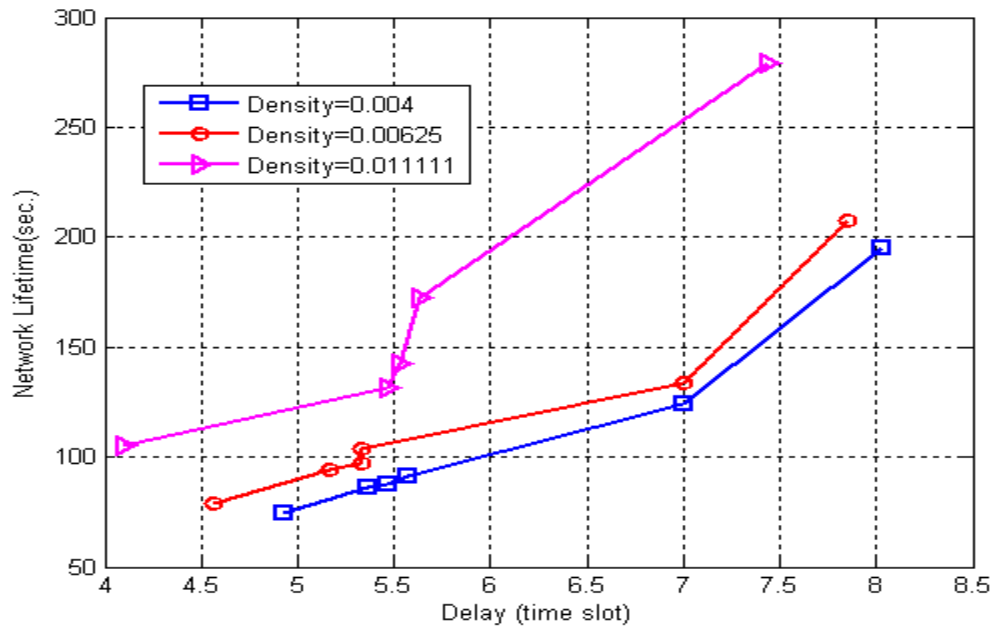
Assuming parameters shown in Table 7-2, and using (10), the energy consumed to receive one data packet is 40  $\mu J$ . This is very low value and sometimes can be approximated to 0 in LINGO solver. Therefore, we use in our experiments mill joule units. We consider the initial energy for the sink is 100000 mJ, the initial energy of the other nodes is 2000 mJ, and the consumed energy to receive one data packet is 0.04 mJ. Assuming this range of values and taking into account the number of child for the sink to be 3 in average, the first term of the third cost function (26) will be in the range of ten thousands. On the other hand, assuming we have ten nodes, the second term (time slot for the sink node to transmit) of third cost function (26) will be in the range of ten.

Therefore, we use  $\alpha_2$  to assign high weight for the second term to be comparable with the first term. We assume  $\alpha_1$  to be 1 and  $\alpha_2$  to be in the range of thousands. Figure 7-13 shows network lifetime versus delay for different values of  $\alpha_1$  and  $\alpha_2$  assuming network density equals to 0.004. If  $\alpha_1$  is 1 and  $\alpha_2$  is zero, then the first cost function is considered. On the other hand, if  $\alpha_1$  is 0 and  $\alpha_2$  is 1, then the second cost function is considered. For other values of  $\alpha_1$  and  $\alpha_2$ , the cost function is a combination of the first and second cost functions. Figure 7-13 is very useful; it can be used to know the network lifetime for a given delay. Figure 7-14 shows network lifetime versus delay for different network density. both figures have very interesting results, For the same delay, we can achieve the high lifetime if we increase the network density, however, we achieve the same network lifetime at the expense of long delay under lower density



**Figure 7-13: Network lifetime (sec.) vs. Delay (time slots), Network Density= 0.004 node/m<sup>2</sup>**



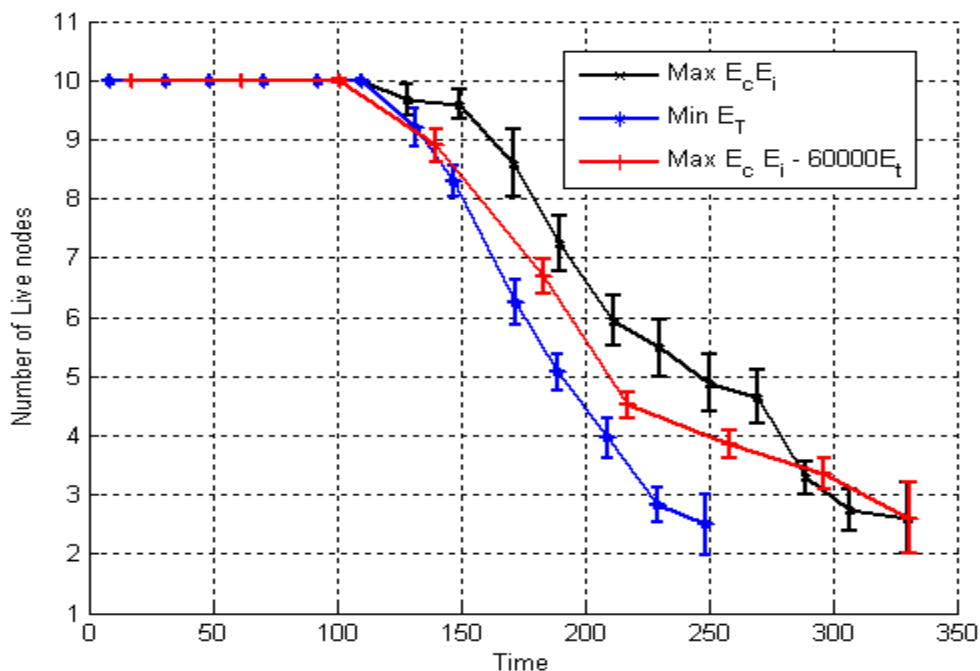


**Figure 7-14: Network lifetime (sec.) vs. Delay (time slots), Different Network Density**

Finally we compare the results obtained by solving the model using the first cost function with the results obtained by solving the model using the fourth cost function (28)

$$\max \quad \alpha_1 \sum_{i=1}^n EC_i E_i - \alpha_2 ET_{total} \quad (28)$$

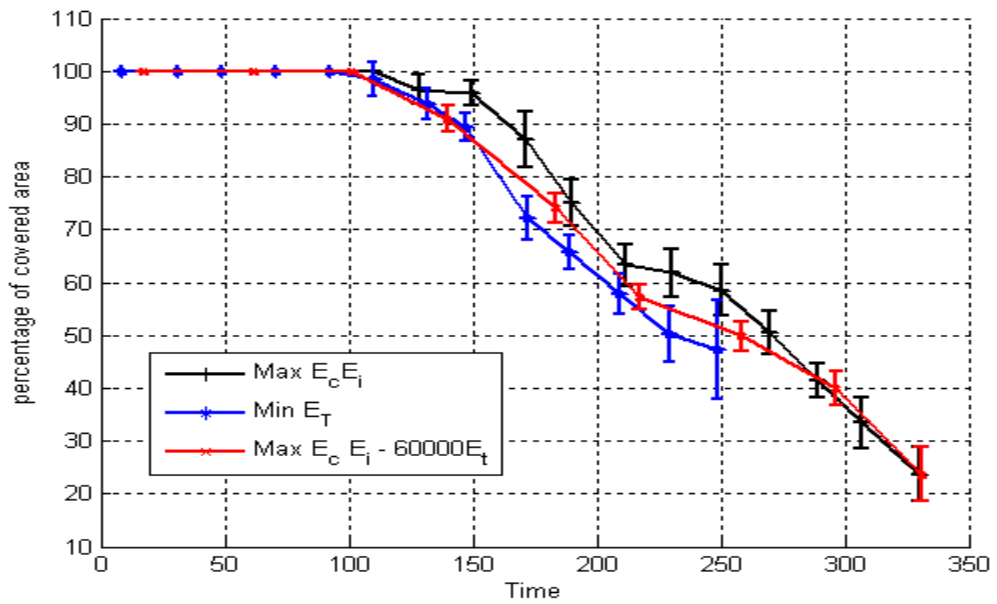
The second term represents the total consumed energy due to transmitting data packets in each node. Using parameters presented in Table 7-2 and assuming the worst case where distance between two pairs of nodes is 15 m, the energy consumed by each node will 0.098 mJ. Since we have 10 nodes, the maximum value of second term is in 0.98 mJ. On the other hand, the value of the first term is in the range of ten thousands. Therefore, to assign comparable weights for the two terms, we consider  $\alpha_1$  to be 1 and  $\alpha_2$  to be on the range of ten thousands. We consider a network configuration where 10 nodes are distributed in area with dimension with  $50 \times 50$ .



**Figure 7-15: Number of Live Nodes vs. Time, Nodes=10, Area=50x50**

Figure 7-15 shows number of live nodes vs. time when solving the ILP model using the first cost function, the second term of the fourth cost function( minimizing total transmitted energy), and the fourth cost function assuming  $\alpha_1=1$  and different values for  $\alpha_2$ . We observed that when considering the cost function to minimize the total transmitted energy, the nodes start to die earlier than other cases. When the cost function is to minimize the total transmitted energy, the tree is built such that each node will select the closest neighbor as a parent. Since nodes are fixed and they are not moving, each node will select the same parent each round assuming that parent is still alive; therefore will be some successive rounds where the same tree will be built in each round. The same nodes will work as parents in these successive rounds. Therefore, they will consume more energy than other nodes, and hence they these nodes will die early. On the other hand, when we consider the first cost function as our objective, different nodes will

work as parents in each round. Therefore nodes will take longer time to die. When we consider the fourth cost function, the tree is built such that high nodes will have more children as long as the transmitted energy will be minimized. In this case, the nodes will die later than the case when the cost function is minimizing the transmitted energy. At the same time, the nodes will die earlier than when assigning more children for high-energy nodes.



**Figure 7-16: Percentage of covered Area vs. Time, Nodes=10, Area=50x50**

Figure 7-16 shows percentage of covered area versus time for the different cases. Percentage of covered are shows a behavior similar to the number of live nodes. The delay under different cost function is shown in Table 7-3. We observe that under different cost functions, approximately the same delay is achieved. The delay shown in Table 7-3 is the average of the delay that achieved in the interval between the beginnings of simulation time until the time at which the percentage of covered area reaches 75% of the monitored area.

**Table 7-3: Delay for Different cost Functions**

Cost Function	Delay (slot)
Min $E_t$	8.167333
Max $E_{C_i}E_i$	8.3919
Max $E_{C_i}E_i - 6000E_t$	7.9566

### 7.4.3 COMPARING THE RESULTS OF SOLVING ILP MODEL USING THE FIRST COST FUNCTION WITH DIFFERENT NETWORK SIZES:

In this section we compare the results obtained by solving the model using the first cost function assuming networks with 10 and 20 nodes. We consider 10 and 20 nodes distributed randomly in an area with different dimensions. We consider only the first cost function, because solving the model using the other cost functions under 20 and 30 nodes took a lot of time. For example trying to solve the model using the second function with 30 nodes takes more than 80 hours without generating an optimal solution.

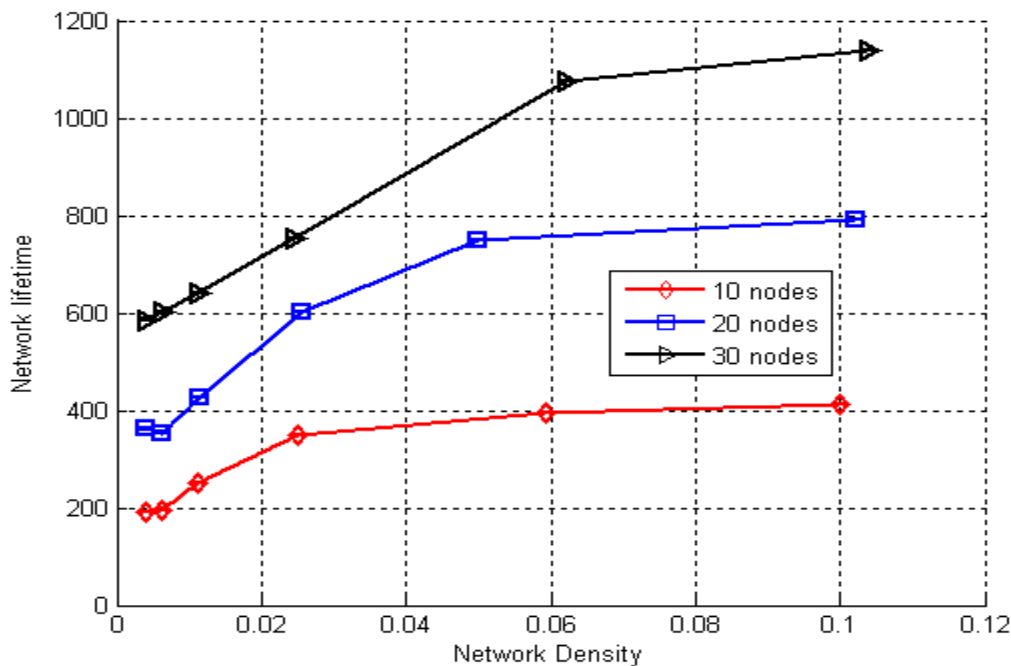
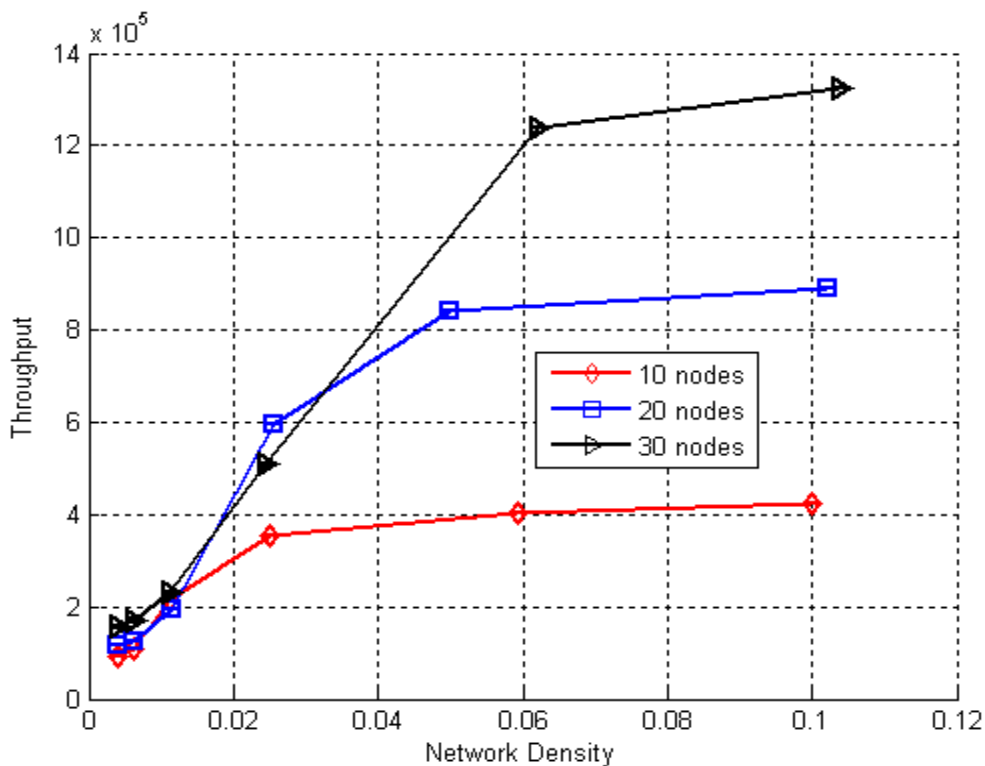
**Figure 7-17: Network Lifetime vs. Density, first cost function**

Figure 7-17 shows the network lifetime vs. network density. We can observe that for low network density, network lifetime is low. As network density increases, network lifetime improves. For high network density, network lifetime become stable. For low network density, the monitored area will be large. Therefore, distance between nodes will be longer; nodes will consume more energy in transmitting data packets, which minimize network lifetime. Increasing network density will decrease distance between nodes. Therefore, nodes will consume less energy in transmitting data packets, which improves network lifetime. For very high network density with 10 and 20 sensor nodes, the monitored area will be small. Therefore, all nodes will be close to the sink. Since the cost function here is to assign more children to high energy nodes and since the sink has the highest energy, all the nodes will select the sink as a parent. We have a tree in which all nodes are leaf for the sink. They will consume energy to transmit data packets to sink. Further increasing of network density with fixed number of nodes will not change tree. All nodes will select the sink as a parent. Therefore, network lifetime will not be changed. This is valid for 10 and 20 nodes as we observed from **Figure 7-17**. With 20 nodes, the network lifetime is larger. With larger number of nodes, the node will act as a non-leaf node for smaller portion of time. Therefore, they will consume less energy, which improves network lifetime.

Improving in network lifetime will increase number of data packets delivered to the sink as shown in Figure 7-18. We observe that while network density increased, the throughput improved. For high network density, the throughput will become stable with increasing network density. This is the case when the monitored area becomes small such that all nodes will be connected directly to the sink. The same tree will be built for

different network density. The tree will not vary from round to round, in each round, the same number of data packets delivered to the sink will be equal to the number of live nodes of the network.



**Figure 7-18: Throughput (packets) vs. Density, first cost function**

The number of live nodes vs. time when 10 and 20 nodes are distributed in a monitored area with dimension (50x50 m) is shown in Figure 7-19. We observed that: initially, the number of live nodes is constant, and then it starts decreasing. For larger number of nodes, the network lifetime is higher. The percentage of covered area vs. time is shown in Figure 7-20

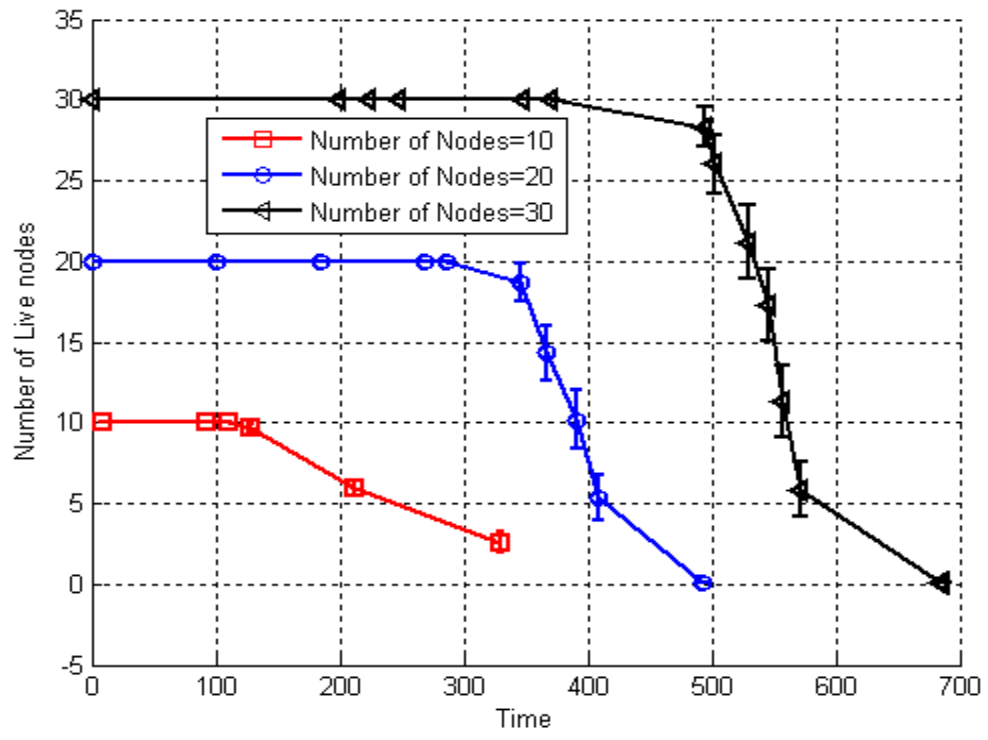


Figure 7-19: Number of live nodes vs. time, Density=0.004

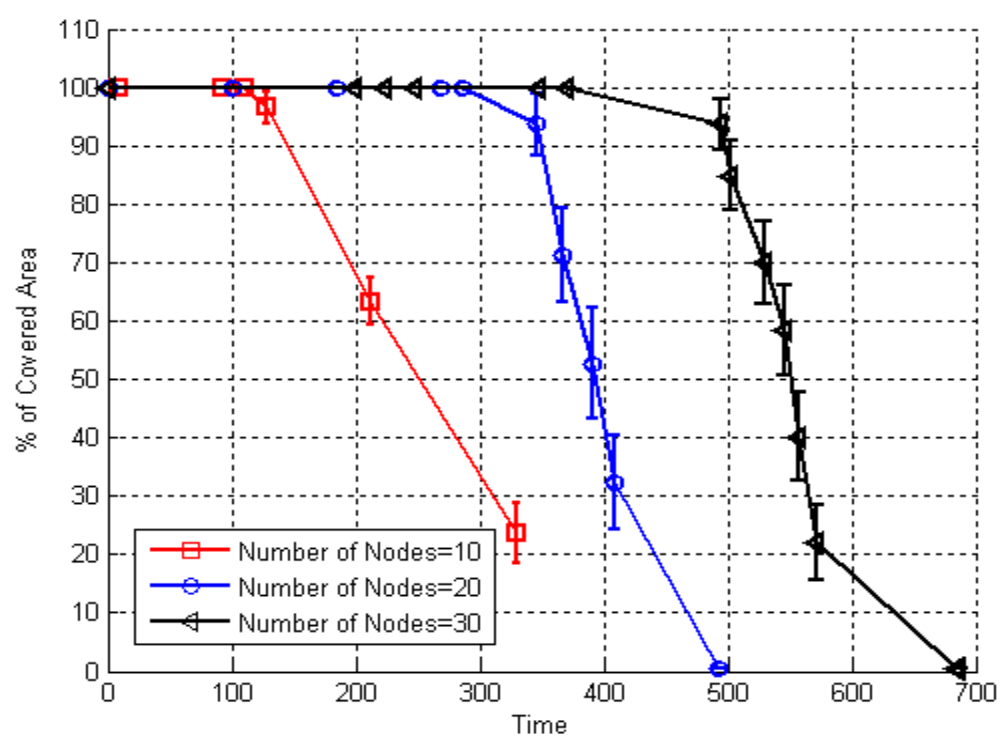


Figure 7-20: Percentage of covered area vs. time, Density=0.004

# Chapter Eight

## CONCLUSION

In our research, we studied the data forwarding in WSN. The main parameters that affect the operation of data forwarding from sensor nodes to sink are identified. Based on our study of the existing protocols, we proposed a framework to forward data from sensor nodes to sink. A cross layer design methodology is adopted in designing our framework. MAC and Network layers will be integrated. Our framework aims at maximizing network lifetime.

The proposed framework is called a Generalized Energy-Efficient Time-Based communication protocol (GET). GET intends to increase network lifetime by minimizing energy consumed by each node. Energy consumed in each node is minimized by decreasing the amount of time in which a sensor node is in idle listening state. Moreover, GET intends to utilize all the initial energy stored in sensor nodes. Initial energy is fully utilized by minimizing the isolated nodes. The time in GET is divided into rounds, each round consists of four phases; selecting gateways (nodes connected directly to sink) (SG), building tree (BT), building schedule (BS) and data transmission (DT). In GET, gateways can be changed during network lifetime. Different nodes can act as gateways. A



mechanism to select gateways based on the residual energy of the nodes was proposed. The selected node will act as a gateway as long as its residual energy is greater than a threshold value  $E_{th}$ . New gateways will be selected each round. In building tree phase, an energy efficient tree from all nodes towards the sink is built. Building tree process is initiated by gateways. Since gateways differs from round to round, different tree will be constructed each round. Then, based on this tree, a TDMA schedule is built in building schedule phase. In data transmission phase, the schedule is followed to forward data from all nodes to the sink. The data transmission phase may be repeated multiple times in a single round.

The GET is validated using different network configurations and compared with the random-based protocol (EAD) and the LEACH in terms of network lifetime, throughput and energy consumption. Compared to EAD, GET has improved the network lifetime from 78% to 237%. Furthermore, the energy consumption is reduced by 65% and the throughput is significantly improved. When compared to LEACH, GET has shown outstanding improvement in network lifetime, throughput, and consumed energy. The network lifetime is improved by about 50%, while the energy consumption is reduced by 51%; on the other hand, the throughput is improved by GET by more than 291% in some cases.

In GET, we assume that sensor nodes are able to transmit signal for long distance. Sometimes this is not valid for all sensors. Therefore we proposed the Energy Efficient Data Communication Protocol (EEDS) which is a special case of GET. In EEDS only the nodes that are close to the sink will act as gateways. Therefore, all nodes will communicate with their closest neighbors. Hence, it is not required from the sensor node

to be able to transmit for long distance. Since no gateways are selected in EEDS, each round in EEDS consists of three phases; building tree (BT), building schedule (BS) and data transmission (DT). Building tree process will be initiated by the sink. EEDS is examined (using different network configurations) and compared with the random-based protocol (EAD) and the LEACH in terms of network lifetime, throughput and energy consumption. Compared with EAD, the network lifetime in EEDS is improved by at least 67.3%. Moreover, the energy consumption is also decreased by 34.4% to 74.5% and the aggregate throughput is significantly improved. When it is compared with the LEACH, the network lifetime in EEDS is improved by at least 33.3%., the energy consumption is also dropped by more than 33.3%, while the throughput is improved between 8.6 %-48.7%.

The idea of selecting gateways is used to generalize the Generalized Energy-Aware Data Centric Routing (EAD). We called the new protocol  $EAD_{General}$ .  $EAD_{General}$  intends to increase the lifetime of the network by increasing the number of candidate gateway nodes. Extensive simulation experiments show that  $EAD_{General}$  outperforms EAD in terms of the network lifetime for all different network configurations.

Our study of different routing protocols of WSN proved that in hierarchal routing protocol, a packet delivered to the sink is resulted from aggregating many raw packets. Two different packets delivered to the sink may be resulted from the aggregation of different number of raw packets. These two packets may carry different amount of information. Therefore considering throughput as the absolute number of data packets delivered to the sink is not a precise measure. Hence, we proposed Information-Entropy based metric to measure the throughput. In the new metric, we defined the throughput as

the amount of information delivered to the sink. The proposed metric yields a better understanding of the operation of the WSN application under consideration. Moreover, it fairly compares different hierarchical routing protocols in which the clusters are formed using different techniques. We used the proposed metric to compare our proposals with different well-known routing protocols such as: EAD and LEACH.

Finally, to explore the optimal solutions that can be produced assuming global information, we formulate EEDS with an integer linear programming (ILP) model. We proposed four cost functions for the ILP model. The first cost function is defined to assign more children for high energy nodes. This cost function is similar to selecting parents in our proposals. The objective of the first cost function is to maximize the network lifetime. The second cost function was defined to minimize the time (delay) needed to forward data packets to the sink. To maximize the network lifetime accompanied with reasonable delay, we proposed the third cost function. The third cost function is a combination of the first and the second cost function. The fourth cost function is defined to assign more children to high energy and to minimize the total transmitted energy. We used LINGO solver to solve our model. The results obtained by solving the ILP model under the first cost function are compared with the results obtained by simulation. We observe that although network lifetime is improved in ILP solution by 28.3%, the throughput in ILP solution is improved by only 3.3%. Moreover, optimal solutions using different cost functions for different network configuration are generated.

# Chapter Nine

## FUTURE WORK

Although our proposals: GET and EEDS show good improvements compared with some well-known protocols such as EAD and LEACH. We think there is a chance for more improvement in GET and EEDS. For example, studying the effect of the number of gateways on the performance of GET may lead to optimal number of gateways that may maximize the network lifetime. Moreover, the selection of parents can be studied more. For example, integrating new criteria into selection parent mechanism other than residual energy of the node may improve the network lifetime. On the other, hand the optimal number of cycles that improve the network lifetime can be studied.

Regarding the ILP formulation, we think that it can be enhanced. In our formulation, an optimal solution is generated in each round. We think it is better to generate a set of optimal solutions for several rounds. Moreover, more cost functions can be defined for the ILP model.

## REFERENCES

- [1] Ian F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Personal Communications Magazine*, Volume 40, Issue 8, Aug. 2002, pp.102 - 114.
- [2] Sensor Modules , <http://www.xbow.com>, Manual data sheet for MICAz.
- [3] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “An Application-Specific Protocol Architecture for Wireless Microsensor Networks”, *IEEE On Wireless Communications Trans.*, vol. 1, No. 4, Oct. 2002, pp. 660-670.
- [4] A. Boukerche, X. Cheng, J. Linus, “A Performance Evaluation of a Novel Energy-Aware Data-Centric Routing Algorithm in Wireless Sensor Networks”, *Wireless Networks* 11, 2005, pp.619–635,
- [5] Wei Ye, John Heidemann, and Deborah Estrin, Fellow, “Medium Access Control With Coordinated Adaptive Sleeping for Wireless Sensor Networks”, *IEEE/ACM Transactions on Networking*, Vol 12, No. 3, June 2004, pp. 493-506.
- [6] Mark Stemm and Randy H Katz, “Measuring and reducing energy consumption of network interfaces in hand-held devices,” *IEICE Transactions on Communications*, vol. E80-B, no. 8, Aug. 1997, pp. 1125–1131.
- [7] Lindo systems, <http://www.lindo.com>, accessed, April 2009

- [8] The working group for WLAN standards. IEEE 802.11 standards, Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications. Technical report, IEEE, 1999.
- [9] S. Singh and C. S. Raghavendra, "PAMAS: Power aware multi-access protocol with signalling for ad hoc networks," *ACM Comput. Commun. Rev.*, vol. 28, no. 3, July 1998. pp. 5–26,
- [10] Chansu Suh, Young-Bae Ko. "A traffic Aware, Energy Efficient MAC Protocol for Wireless Sensor Networks", *IEEE International Symposium on Circuits and Systems, 2005. ISCAS 2005. Vol. 3 , 23-26 May 2005 pp.2975 - 2978*
- [11] Lin, P. ; Qiao, C. and Wang, X. "Medium Access Control With A Dynamic Duty Cycle For Sensor Networks," in *Wireless Communications and Networking Conference (WCNC), IEEE, Vol. 3, 21-25 March 2004, pp.:1534 - 1539.*
- [12] T. V. Dam and K. Langendoen, "An adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *ACM SenSys'03, November 5–7, 2003, Los Angeles, California, USA.*
- [13] Saad Biaz, Yawen Dai Barowski, "GANGS: an Energy Efficient MAC Protocol for Sensor Networks", in the *Proceedings of the 42nd annual Southeast regional conference, ACMSE'04, Huntsville, Alabama, April 2-3, 2004, pp. 82 - 87*
- [14] K. Akkaya and M. Younis, "A Survey of Routing Protocols in Wireless Sensor Networks, " in the *Elsevier Ad Hoc Network Journal, vol. 3/3, 2005, pp. 325-349.*
- [15] S. Hedetniemi and A. Liestman, "A survey of gossiping and broadcasting in communication networks," *Networks, Vol. 18, No. 4, 1988, pp. 319-349.*

- [16] W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in the Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99), Seattle, WA, August 1999. pp. 174-185.
- [17] Intanagonwiwat, C.; Govindan, R.; Estrin, D.; Heidemann, J.; Silva, F.; " Directed diffusion for wireless sensor networking ", Networking, IEEE/ACM Transactions on Volume 11, Issue 1, Feb. 2003, pp.:2 – 16.
- [18] Shah, R.C.; Rabaey, J.M.; "Energy Aware Routing for Low Energy Ad Hoc Sensor Networks", in the Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 2002), Orlando, FL, Volume 1, 17-21 March 2002 pp. :350 - 355
- [19] Braginsky , D.; Estrin, D., "Rumor Routing Algorithm for Sensor Networks," in the Proceedings of the First Workshop on Sensor Networks and Applications (WSNA'02), September 28, 2002, Atlanta, GA, pp. 22-30.
- [20] Y. Yao and J. Gehrke, "The cougar approach to in-network query processing in sensor networks," in ACM SIGMOD Record, Volume 31 , Issue 3, September 2002. pp. 9-18.
- [21] N. Sadagopan, B. Krishnamachari and A. Helmy, "The ACQUIRE mechanism for efficient querying in sensor networks", in in the Proceedings of First IEEE Internat. Workshop on Sensor Network Protocols and Applications (SNPA), in conjunction with IEEE ICC, Anchorage (May 2003) pp. 149-155.
- [22] A. Bachir, D. Barthel, M. Heusse, and A. Duda, "O(1)-Reception routing for sensor networks," Computer Communications 30 (2007), pp. 2603-2614.

- [23] Y. Chen, N. Nasser, "Energy-Balancing Multipath Routing Protocol for Wireless Sensor Networks," in the Proc. of the third International Conference on Quality of Service in Heterogeneous Wired/Wireless Network, Waterloo, Ontario, Canada, August 7-9, 2006.
- [24] Chu, M., Haussecker, H. and Zhao, F. Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks. The International Journal of High Performance Computing Applications, Vol. 16, No. 3, August 2002. pp. 293--313.
- [25] Saurabh Mishar and Asis Nasipuri, "An Adaptive Low Power Reservation Based MAC Protocol for Wireless Sensor Networks ", IEEE International Conference on Performance, Computing, and Communications , Page(s):731 – 736, 2004
- [26] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and Wei Hong, "TinyDB: An Acquisitional Query Processing System for Sensor Networks", ACM Transaction on Database Systems, Vol. 30, No 1, March 2005, pp. 122-173.
- [27] Stephanie Lindsay , C. S. Raghavendra , Krishna M. Sivalingam, Data Gathering in Sensor Networks using the Energy Delay Metric, Proceedings of the 15th International Parallel & Distributed Processing Symposium, April 23-27, 2001. pp.188,
- [28] Arati Manjeshwar , Dharma P. Agrawal, TEEN: A Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks, Proceedings of the 15th International Parallel & Distributed Processing Symposium, April 23-27, 2001. pp.189.
- [29] A. Manjeshwar and D. P. Agrawal, "APTEEN: A Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks," in



- the Proceedings of the 2nd International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile computing, Ft. Lauderdale, FL, April 2002.
- [30] Jing Li, and Georgios Y. Lazarou, "A Bit-Map-Assisted Energy-Efficient MAX Scheme for Wireless Sensor Networks" in Proc. Hawaii Int. Conf. Systems Sciences, pp. 3005–3014, Jan. 2000.
- [31] Schurgers, C.; Srivastava, M.B., "Energy efficient routing in wireless sensor networks," Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE , vol.1, no., vol.1, 2001. pp. 357-361
- [32] Wendi Rabiner Heinzelman , Anantha Chandrakasan , Hari Balakrishnan, Energy-Efficient Communication Protocol for Wireless Microsensor Networks, Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8, p.8020, January 04-07, 2000
- [33] S. Lindsey and C. S. Raghavendra, "PEGASIS: Power Efficient GATHERing in Sensor Information Systems," in the Proceedings of the IEEE Aerospace Conference, Big Sky, Montana, March 2002.
- [34] G. Chen, C.F. Li, M. Ye, and Jie Wu, "An Unequal Cluster-Based Routing Strategy in Wireless Sensor Networks ," Book chapter in Wireless Networks (JS) , April 2007. pp. 193-207.
- [35] Subramanian, L.; Katz, R.H., "An architecture for building self-configurable systems," Mobile and Ad Hoc Networking and Computing, 2000. MobiHOC. 2000 First Annual Workshop on, 2000, pp.63-73,

- [36] Younis, M.; Youssef, M.; Arisha, K., "Energy-aware routing in cluster-based sensor networks," *Modeling, Analysis and Simulation of Computer and Telecommunications Systems, 2002. MASCOTS 2002. Proceedings. 10th IEEE International Symposium on*, Fort Worth, TX, October 2002, pp. 129-136.
- [37] S. Muruganathan, D. Ma, R. Bhasin, and A. Fapojuwo, "A Centralized Energy-Efficient Routing Protocol for Wireless Sensor Networks," *IEEE Radio Communication*, March 2005, pp. S8-S13.
- [38] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad hoc routing," in the *Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'01)*, Rome, Italy, July 2001. pp. 70-84.
- [39] V. Rodoplu and T.H. Ming, "Minimum energy mobile wireless networks," *IEEE Journal of Selected Areas in Communications*, Vol. 17, No. 8, 1999, pp. 1333-1344.
- [40] Y. Yu, D. Estrin, and R. Govindan, "Geographical and Energy-Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks," *UCLA Computer Science Department Technical Report, UCLA-CSD TR-01-0023*, May 2001.
- [41] F. Kuhn, R. Wattenhofer, A. Zollinger, "Worst-Case optimal and average-case efficient geometric ad-hoc routing", *Proceedings of the 4th ACM International Conference on Mobile Computing and Networking*, , 2003, pp. 267-278.
- [42] I. Stojmenovic and X. Lin. "GEDIR: Loop-Free Location Based Routing in Wireless Networks", In the proceeding of *International Conference on Parallel and*

- Distributed Computing and Systems, Boston, MA, USA, Nov. 3-6, 1999. pp. 1025-1028.
- [43] B. Chen, K. Jamieson, H. Balakrishnan, R. Morris, "SPAN: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks", *Wireless Networks*, Vol. 8, No. 5, September 2002. pp. 481-494,
- [44] F. Lotfifar, H. Shahhoseini, "A mesh-Based Routing Protocol for Wireless Ad-Hoc Sensor Networks," in the Proc. of International Wireless Communication and Mobile Computing Conference (IWCMC'06), Vancouver, British Columbia, Canada, July 3-6, 2006. pp. 115 – 120.
- [45] J. Sanchez, P. Ruiz, and I. Stojmenovic, "Energy-efficient geographic multicast routing for Sensor and Actuator Networks," *Computer Communications* 30 (2007) pp. 2519–2531
- [46] G. Zhao, X. Liu, and M. Sun, "Energy-Aware Geographic Routing for Sensor Networks with Randomly Shifted Anchors," in the Proc. of Wireless Communications and Networking Conference WCNC 2007, 11-15 March 2007, pp. 3454-3459
- [47] G. Zhao, X. Liu, and M.-T. Sun, "Anchor based geographic routing for sensor networks using projection distance," in Proc. IEEE International Symposium on Wireless Pervasive Computing (ISWPC), San Juan, Puerto Rico, Feb. 2007. pp. 48-52.
- [48] S. Subramanian, S. Shakkottai, and P. Gupta, "On Optimal Geographic Routing in Wireless Networks with Holes and Non-Uniform Traffic," in the Proc. of 26th IEEE

- International Conference on Computer Communications. INFOCOM 2007, May 2007, pp. 1019-1027
- [49] J.-H. Chang and L. Tassiulas, "Maximum Lifetime Routing in Wireless Sensor Networks," in IEEE/ACM Transactions on Networking (TON), Volume 12 , Issue 4, August 2004, pp. 609 – 619.
- [50] K. Kalpakis, K. Dasgupta, and P. Namjoshi, "Maximum Lifetime Data Gathering and Aggregation in Wireless Sensor Networks," in the Proceedings of IEEE International Conference on Networking (NETWORKS '02), Atlanta, GA, August 2002.
- [51] M. Chu, H. Haussecker, and F. Zhao, "Scalable Information-Driven Sensor Querying and Routing for ad hoc Heterogeneous Sensor Networks," The International Journal of High Performance Computing Applications, Vol. 16, No. 3, August 2002. pp. 293—313.
- [52] Akkaya, K.; Younis, M., "An energy-aware QoS routing protocol for wireless sensor networks," Proceedings. 23rd International Conference on Distributed Computing Systems Workshops, 19-22 May 2003, pp. 710-715,
- [53] H. Morcos, I. Matta, and A. Bestavros, "BIPAR: BImodal Power-Aware Routing Protocol For Wireless Sensor Networks," in the proceeding of the First International Computer Engineering Conference (ICENCO 2004), Cairo, Egypt, December 2004.
- [54] T. He et al., "SPEED: A stateless protocol for real-time communication in sensor networks," in the Proceedings of International Conference on Distributed Computing Systems, Providence, RI, May 2003. pp. 46—55.

- [55] Safwati. A., Hassanein. H., Mouftah. H.,” Optimal Cross-Layer Designs for Energy-Efficient Wireless Ad hoc and Sensor Networks”, in the Proceedings of the IEEE International Conference of Performance, Computing, and Communications 9-11 April 2003, pp. :123 – 128
- [56] Venkitasubramaniam P., Adireddy S., Lang Tong, “Opportunistic ALOHA and cross layer design for sensor networks” , Military Communications Conference, 2003. MILCOM 2003. IEEE Volume 1, 13-16 Oct. 2003 Page(s):705 - 710
- [57] Sichitiu M.L., “Cross-Layer Scheduling for Power Efficiency in Wireless Sensor Networks” ,INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies , Volume 3, 2004, Page(s):1740 - 1750
- [58] Li-Chun Wang, Chung-Wei Wang, “A Cross-layer Design of Clustering Architecture for Wireless Sensor Networks”, in the Proceedings of the IEEE International Conference on Networking, Sensing & Control Taipei, Taiwan, March 21-23, 2004, Page(s): 547-552
- [59] Rick W. Ha, Pin-Han Ho, Sherman X. Shen: Cross-layer application-specific wireless sensor network design with single-channel CSMA MAC over sense-sleep trees. *Computer Communications* 29(17): (2006) , pp. 3425-3444
- [60] Shuguang Cui, Madan R. , Goldsmith A. , Lall S. , “Joint routing, MAC, and link layer optimization in sensor networks with energy constraints “ IEEE International Conference on Communications, ICC 2005 ,Volume 2, 16-20 May 2005 Page(s):725 - 729
- [61] Su. W., T.L. Lim , “Cross-Layer Design and Optimization for Wireless Sensor Networks,” Proceedings of the Seventh ACIS International Conference on Software

Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, SNPD June 2006 Page(s):278 – 284

- [62] Changsu Suh, Young-Bae Ko, and Dong-Min Son, “An Energy Efficient Cross-Layer MAC Protocol for Wireless Sensor Networks”, APWeb 2006, LNCS 3842, 2006. pp. 410–419,
- [63] H. Wang, K. Yao, " Entropy-based Sensor Selection Heuristic for Target Localization," in Proc of IPSN'04, California, USA, April 2004, pp. 36-45.
- [64] Sensor Modules , <http://www.xbow.com>, accessed October 2008
- [65] Shijin Dai; Lemin Li; Du Xu, "A Novel Cluster Formation Approach Based on The ILP for Wireless Sensor Networks," Communications and Networking in China, 2006. ChinaCom '06. First International Conference on , Oct. 2006. pp.1-5, 25-27
- [66] Ali Chamam, Samuel Pierre ,” Energy-Efficient State Scheduling for Maximizing Sensor Network Lifetime under Coverage Constraint”, in the Proceedings of the Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, 2007, pp. 63.
- [67] Katerina Papadaki, Vasilis Friderikos, ”Joint Routing and Gateway Selection in Wireless Mesh Networks”, Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE , March 31 2008-April 3 2008. pp.2325-2330
- [68] Raja Jothi, Balaji Raghavachari , “Approximation algorithms for the capacitated minimum spanning tree problem and its variants in network design” in ACM Transactions on Algorithms (TALG) Volume 1 , Issue 2 (October 2005) Pages: 265 - 282

**VITAE**

Tayseer AL-khdour obtained his B.Sc. degree in Electrical Engineering (Minor: Computer engineering) in 1994 from Jordan University of Science and Technology (JUST), Irbid, Jordan. He worked for a Jordanian company as a computer engineer for one year. He obtained his M.Sc. in Computer Engineering in 1998 from JUST. He joined King Faisal University (Computer Science Dept.)-Saudi Arabia for five years. He has recently defended his PhD Dissertation in Computer Science and Engineering at King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia. His research of interest includes ad-hoc networks, WSN. He has presented his work in journals and conference papers