

## UNIVERSITI TEKNOLOGI MALAYSIA

BORANG PENGESAHAN  
LAPORAN AKHIR PENYELIDIKAN

TAJUK PROJEK :

**Transport Mechanism for Wireless Micro-Sensor Network (WSN)**Saya PROF. DR. NORSHEILA BT. FISAL  
(HURUF BESAR)Mengaku membenarkan **Laporan Akhir Penyelidikan** ini disimpan di Perpustakaan Universiti Teknologi Malaysia dengan syarat-syarat kegunaan seperti berikut :

1. Laporan Akhir Penyelidikan ini adalah hakmilik Universiti Teknologi Malaysia.
2. Perpustakaan Universiti Teknologi Malaysia dibenarkan membuat salinan untuk tujuan rujukan sahaja.
3. Perpustakaan dibenarkan membuat penjualan salinan Laporan Akhir Penyelidikan ini bagi kategori TIDAK TERHAD.
4. \* Sila tandakan ( / )

☐

SULIT

(Mengandungi maklumat yang berdarjah keselamatan atau Kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972).

☐

TERHAD

(Mengandungi maklumat TERHAD yang telah ditentukan oleh Organisasi/badan di mana penyelidikan dijalankan).

☒TIDAK  
TERHAD  
TANDATANGAN KETUA PENYELIDIKPROF. DR. NORSHEILA BT. FISAL

Nama &amp; Cop Ketua Penyelidik

Tarikh : 5 Ogos 2008

**CATATAN :** \* Jika Laporan Akhir Penyelidikan ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali sebab dan tempoh laporan ini perlu dikelaskan sebagai SULIT dan TERHAD.

**TRANSPORT MECHANISM FOR WIRELESS MICRO- SENSOR  
NETWORKS**

**PROF.DR. NORSHEILA FISAL**

**Faculty of Electrical Engineering  
Universiti Teknologi Malaysia**

**JULY 2008**

## ABSTRACT

Wireless sensor network (WSN) is a wireless ad hoc network that consists of very large number of tiny sensor nodes communicating with each other with limited power and memory constrain. WSN demands real-time routing which requires messages to be delivered within their end-to-end deadlines (packet lifetime). This report proposes a novel real-time with load distribution (RTLTD) routing protocol that provides real time data transfer and efficient distributed energy usage in WSN. The RTLTD routing protocol ensures high packet throughput with minimized packet overhead and prolongs the lifetime of WSN. The routing depends on optimal forwarding (OF) decision that takes into account of the link quality, packet delay time and the remaining power of next hop sensor nodes. RTLTD routing protocol possesses built-in security measure. The random selection of next hop node using location aided routing and multi-path forwarding contributes to built-in security measure. RTLTD routing protocol in WSN has been successfully studied and verified through simulation and real test bed implementation. The performance of RTLTD routing in WSN has been compared with the baseline real-time routing protocol. The simulation results show that RTLTD experiences less than 150 ms packet delay to forward a packet through 10 hops. It increases the delivery ratio up to 7 % and decreases power consumption down to 15% in unicast forwarding when compared to the baseline routing protocol. However, multi-path forwarding in RTLTD increases the delivery ratio up to 20%. In addition, RTLTD routing spreads out and balances the forwarding load among sensor nodes towards the destination and thus prolongs the lifetime of WSN by 16% compared to the baseline protocol. The real test bed experiences only slight differences of about 7.5% lower delivery ratio compared to the simulation. The test bed confirms that RTLTD routing protocol can be used in many WSN applications including disasters fighting, forest fire detection and volcanic eruption detection.

## ABSTRAK

Rangkaian peranti pengesan tanpa wayar (WSN) terdiri daripada sejumlah bilangan besar nod peranti pengesan yang berhubung di antara satu sama lain dengan tenaga dan ingatan yang terhad. WSN memerlukan laluan masa nyata di mana mesej di dalam rangkaian perlu dihantar dalam tempoh tamat hujung-ke-hujung (jangka hayat paket). Tesis ini mencadangkan protokol laluan masa nyata pengagihan beban (RTLDD) yang selamat, pemindahan data masa nyata dan agihan penggunaan tenaga yang efisien dalam WSN. Protokol laluan RTLDD menjamin kadar penerimaan paket yang tinggi dan *overhead* paket yang minima yang dapat memanjangkan jangka hayat WSN. Penentuan laluan bergantung kepada keputusan laluan tuju depan optima yang mengambil kira ciri-ciri kualiti rangkaian, masa lengah paket dan baki tenaga dalam nod peranti pengesan untuk hop berikutnya. Protokol laluan RTLDD mempunyai keselamatan terbina dalam. Pemilihan secara rawak bagi hop menggunakan laluan berbantuan lokasi dan laluan tuju depan berbilang. RTLDD telah berjaya diuji melalui simulasi dan implementasi pada *test bed* sebenar. Prestasi protokol laluan RTLDD juga dibandingkan dengan protokol laluan asas masa nyata. Keputusan simulasi menunjukkan bahawa RTLDD mengalami kurang dari 150 ms lengah paket untuk penghantaran melalui 10 hop. Ia meningkatkan nisbah hantaran sehingga 7% dan mengurangkan penggunaan tenaga sehingga 15% dalam penghantaran *unicast* berbanding laluan asas. Walaubagaimanapun, penghantaran laluan berbilang dalam RTLDD meningkatkan nisbah hantaran sehingga 20% berbanding penghantaran *unicast*. Tambahan pula, RTLDD menyebarkan dan mengimbangi beban penghantaran di antara nod-nod peranti pengesan sekaligus memanjangkan jangka hayat WSN sehingga 16% berbanding protokol asas. Implementasi pada *test bed* sebenar mengalami sedikit perbezaan dengan nisbah hantaran 7.5% lebih rendah berbanding simulasi. *Test bed* juga mengesahkan RTLDD boleh digunakan dalam banyak aplikasi WSN termasuklah menghadapi musibah seperti pengesanan kebakaran hutan dan pengesanan letusan gunung berapi.

## TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
	<b>DECLARATION</b>	ii
	<b>DEDICATION</b>	iii
	<b>ACKNOWLEDGEMENT</b>	iv
	<b>ABSTRACT</b>	v
	<b>ABSTRAK</b>	vi
	<b>CONTENTS</b>	vii
	<b>LIST OF TABLES</b>	xi
	<b>LIST OF FIGURES</b>	xii
	<b>LIST OF ABBREVIATION</b>	xvii
	<b>LIST OF APPENDICES</b>	xix
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Background	1
	1.2 Application of WSNs	3
	1.3 Problem Statement	4
	1.4 Objectives	5
	1.5 Scope	5
	1.6 Significance of Research Work	7
	1.7 Thesis Organization	7
<b>2</b>	<b>LITERATURE REVIEW</b>	<b>9</b>
	2.1 Introduction	9
	2.2 Architecture of WSNs	10
	2.2.1 IEEE 802.15.4 Specifications	11

2.3	Challenges in WSNs	14
2.3.1	Constrain of WSNs	15
2.3.2	Routing in WSNs	17
2.3.2.1	QoS-based Routing	19
2.3.2.2	Forwarding based Routing	27
2.3.2.3	Query Based Routing	32
2.3.2.4	Negotiation-Based Routing	32
2.3.2.5	Coherent-Based Routing	33
2.4	Summary	34
<b>3</b>	<b>RTLD SYSTEM DESIGN</b>	<b>35</b>
3.1	Introduction	35
3.2	Cross-Layer Design in RTLD	36
3.3	RTLD Design Concepts	37
3.3.1	Routing Management	39
3.3.1.1	Optimal Forwarding Determined	40
3.3.1.2	Forwarding Mechanisms	46
3.3.1.3	Routing Problem Handler	49
3.3.2	Location Management	51
3.3.2.1	Location Determination	52
3.3.3	Neighbourhood Management	57
3.3.3.1	Neighbour discovery	58
3.3.4	Power Management	61
3.4	Built-in Security in RTLD	62
3.5	Network Model and Parameters	63
3.6	Summary	66
<b>4</b>	<b>SIMULATION OF RTLD ROUTING PROTOCOL</b>	<b>68</b>
4.1	Introduction	68
4.2	Simulation Tools	69
4.2.1	IEEE 802.15.4 in NS-2	69
4.3	Development of RTLD Routing Algorithm	70
4.3.1	Development of Routing Management	72
4.3.1.1	Forwarding Metrics Calculation	72

4.3.1.2	Development Routing Problem Handler	84
4.2.1.4	Development Forwarding Mechanisms	88
4.3.2	Development of Location Management	92
4.3.3	Development of Neighbourhood Management	94
4.3.3.1	Neighbour Discovery Function	95
4.3.4	Development of Power Management	96
4.4	Simulation Analysis of RTLD Routing Protocol	98
4.4.1	Effect of End-to-end Delay	98
4.4.2	Impact of Varying Network Load	100
4.4.2.1	Effect of Poisson Traffic	102
4.4.3	Load Distribution Using RTLD Routing	104
4.4.4	Prolonging WSN Lifetime	107
4.4.5	Comparison between Geocast and Geodirection-cast Forwarding	109
4.5	Summary	111
<b>5</b>	<b>IMPLEMENTATION RTLD WSN TEST BED</b>	<b>113</b>
5.1	Introduction	113
5.2	Development of WSN Test Bed	114
5.2.1	Hardware Components	114
5.2.2	Software Components for Configuration and Programming	117
5.2.3	Development RTLD Routing Protocol in Test bed	119
5.2.3.1	Execution of RTLD Routing Protocol in TOSSIM	121
5.2.3.2	Configuration and Programming Sensor Node	125
5.3	Experimental Results of RTLD	127
5.3.1	Test Bed Network	128
5.3.2	Results of RTLD Routing Protocol in Test Bed	129
5.3.3	Geodirection-cast Forwarding in Test Bed	131
5.4	Monitoring Ambient Temperature in WSN	133
5.5	Summary	136

<b>6</b>	<b>CONCLUSION</b>	<b>137</b>
6.1	Introduction	137
6.2	Future Work	140
	<b>REFERENCES</b>	<b>142</b>
	Appendices A - D	<b>152-210</b>



## LIST OF TABLES

<b>TABLE NO.</b>	<b>TITLE</b>	<b>PAGE</b>
2.1	Physical layer description in IEEE 802.15.4	11
2.2	Summary of QoS-based routing protocols	24
2.3	Summary of forwarding based routing protocols	30
3.1	Network Parameters	64
4.1	Trials of all points in A	74
4.2	Comparison WSN lifetime between RTLD and baseline routing protocols	107
5.1	Coding size of RTLD and enhanced RTLD .	128

## LIST OF FIGURES

<b>FIGURE NO.</b>	<b>TITLE</b>	<b>PAGE</b>
2.1	WSNs architecture [13]	10
2.2	Operating Frequency Bands in IEEE 802.15.4 [19]	12
2.3	IEEE 802.15.4 operational modes [19]	13
2.4	The superframe structure without GTSs [19]	13
2.5	The superframe structure with GTSs 19]	14
2.6	Routing based on protocol operation in WSNs	18
3.1	Cross-layer concepts in RTLD	36
3.2	Functional components of RTLD	37
3.3	State machine diagram of RTLD	38
3.4	Routing management functional module	39
3.5	State machine diagram of routing management	40
3.6	PRR vs. Distance	44
3.7	IEEE 802.11b and IEEE 802.15.4 Channel Selection [23]	45
3.8	Flow chart diagram of optimal forwarding policy in unicast forwarding	47
3.9	Unicast forwarding based on Quadrant	48
3.10	Geodirection-cast forwarding based on Quadrant	49
3.11	Fast recovery of routing hole problem	50
3.12	Feedback mechanism in routing problem handler	51
3.13	Network Coordinate System	53
3.14	Flow chart of drawing NCS	53
3.15	Position detection of node c according to NCS	54

3.16	Flow chart of one-hop neighbour position determination	56
3.17	Neighbour table format	57
3.18	State machine diagram for neighbourhood management	58
3.19	Neighbourhood discovery	59
3.20	Replies of neighbour discovery RTR packet	60
3.21	State machine diagram of power management	61
3.22	Network simulation grid	65
3.23	RTLD packet header format, a) data packet and b) control packet	65
4.1	NS-2 simulator for IEEE 802.15.4 [7]	70
4.2	RTLD routing algorithm	71
4.3	Network Topology a) low density, b) medium density, c) high density, d) high density	77
4.4	Delivery Ratio at 10 packet/s a) low density, b) medium density, c) high density d) high density with several sources and e) average	80
4.5	Power Consumption at 10 packet/s a) low density, b) medium density, c) high density d) high density with several sources and e) average	83
4.6	Feedback mechanism algorithm	84
4.7	Random Distribution Topology	86
4.8	Feedback Mechanism in RTLD	86
4.9	Performance of RTLD using feedback packet at different traffic load a) Delivery ratio; b) Normalized packet overhead; and c) Energy consumption	88
4.10	Algorithm of forwarding mechanisms	89
4.11	Performance of RTLD <sub>G</sub> and RTLD <sub>U</sub> at different packet rate a) Delivery ratio; b) Normalized packet overhead; and c) Energy consumption	91
4.12	Algorithm of location management	92
4.13	Network grid with three pre-determined nodes	93
4.14	Locations determined for sensor node in WSN	93

4.15	RTLD neighbour table	94
4.16	Neighbourhood management algorithm	95
4.17	Sending control packet algorithm	95
4.18	Receiving control packet algorithm	96
4.19	Power management algorithm	97
4.20	Comparison between RTLD and baseline routing protocols at different packet deadline	99
4.21	Comparison average end-to-end delay between RTLD and baseline routing protocols for different packet rate	100
4.22	Comparison between RTLD and baseline routing protocols at different packet rate a) delivery ratio; b) normalized packet overhead; and c) Normalized energy consumption.	102
4.23	Comparison between RTLD <sub>U</sub> and baseline routing using Poisson traffic at different interval a) delivery ratio; b) normalized packet overhead; and c) energy consumption.	104
4.24	Network simulation grid of load distribution	105
4.25	Effect of load distribution in RTLD a) traffic load and b) remaining power	106
4.26	Comparison prolonging lifetime between RTLD and baseline routing protocols at different packet rate a) delivery ratio; b) normalized packet overhead; and c) Normalized energy consumption.	109
4.27	Comparison between RTLD <sub>G</sub> and RTLD <sub>GE</sub> at different packet rate a) delivery ratio; b) normalized packet overhead; and c) energy consumption	111
5.1	MICAZ mote a) MICAZ board, b) MICAZ block diagram [85]	115
5.2	TELOSB mote a) TELOSB board, b) TELOSB block diagram [85]	115
5.3	Multipurpose sensor board [85]	116
5.4	MIB 510CA programming board a) MIB 510CA	

	board, b)MIB 510 block diagram [54]	117
5.5	Flow chart diagram of development RTLD routing protocol in test bed	120
5.6	Forwarding trip from source node 24 to the sink 0	122
5.7	Neighbourhood management source code	123
5.8	Neighbour table in TOSSIM	124
5.9	Power monitoring of sensor node in TOSSIM	125
5.10	Transceiver states of sensor nodes in TOSSIM	125
5.11	Programming sensor node	127
5.12	Network simulation grid	128
5.13	Network test bed grid	129
5.14	Performance of RTLD test bed and simulation at different packet rate: (a) Delivery ratio; and (b) Average ETE delay.	130
5.15	Packet receiving in test bed	131
5.16	Performance of geodirection-cast in the test bed and simulation at different packet rate a) Delivery ratio; and b) Average ETE delay	133
5.20	GUI of sensor nodes with on line displaying temperature	135

## LIST OF ABBREVIATIONS

ART	Adaptive Real-Time
CAP	Contention-Access Period
CBC-MAC	Cipher Block Chaining-Message Authentication Code
CBR	Constant Bit Rate
CFP	Contention-Free Period
CPU	Central Processing Unit
CSMA/CA	Carrier Sense Multiple Access / Contention Avoidance
CTS	Clear-To-Send
DSSS	Direct Sequence Spread Spectrum
DGR	Directional Geographical Routing
EDF	Earliest Deadline First
FDM	Frequency Division Multiplexing
FEC	Forward Error Correction
GCC	GNU C Compiler
GFG	Geographic-Forwarding-Geocast
GFPG	Geographic-Forwarding-Perimeter-Geocast
GNU	GNU's not Unix
GPS	Global Positioning System
GPSR	Greedy Perimeter Stateless Routing
GTS	Guaranteed Time Slot
GUI	Graphical User Interface
IP	Internet Protocol
ISM	Industrial Scientific Medical
LAN	Local Area Network
LBRS	Location-Based Resilient Security
LFSR	Linear Feedback Shift Register
LILO	Linux Loader
LR-WPAN	Low Rate Wireless Personal Area Networks
LQER	Link Quality Estimation based Routing

MAC	Medium Access Control
MANET	Mobile Ad Hoc NETWORK
MEMS	Micro-Electro-Mechanical System
MWA	Multiple Winner Algorithm
NCS	Network Coordinate System
NesC	Network Embedded System C
NS-2	Network Simulator-2
OF	Optimal Forwarding
OS	Operating System
OML	Online Maximum Lifetime
PAN	Personal Area Network
PGR	Geographic Routing Protocol
PRR	Packet Reception Rate
QoS	Quality of Service
RAP	Real-Time Protocol
RFC	Request for Comments
RLQ	Resource-aware and Link Quality
RTR	Request-To-Route
RTS	Request-To-Send
RSSI	Received Signal Strength Indication
RTPC	Real-Time Power Control
RTLD Secure	Real-Time Load Distribution
SNR	Signal-to-Noise Ratio
SHA1	Secure Hash Algorithm 1
SPIN	Sensor Protocols for Information via Negotiation
SWA	Single Winner Algorithm
Tcl	Tool command language
TCP	Transmission Control Protocol
TEA	Tiny Encryption Algorithm
TinyOS	Tiny Operating System
Tk	Toolkit
TOSSIM	Tiny Operating System Simulator
TORA	Temporally-Ordered Routing Algorithm
UDP	User Datagram Protocol

USB	Universal Serial Bus
WLAN	Wireless Local Area Network
WSAN	wireless sensor and actor networks
WSN	Wireless Sensor Network



**LIST OF APPENDICES**

<b>APPENDIX</b>	<b>TITLE</b>	<b>PAGE</b>
A	MICAZ and TELOSB Specification	152
B	Integrated RTLD in NS-2	156
C	More results for RTLD	183
D	Source code of RTLD in Real Test Bed	206

## **CHAPTER 1**

### **INTRODUCTION**

#### **1.1 Background**

Wireless networking has witnessed tremendous development in recent years and it has become one of the fastest growing telecommunication sectors. There has been an explosive growth in integration and convergence of different heterogeneous wireless networks in order to ensure effective and efficient communication. These technologies primarily include: Wireless Wide Area Networks (WWANs), Wireless Local Area Networks (WLANs), Wireless Personal Area Networks (WPANs), and the Internet. The cellular networks can be classified under the WWAN, Bluetooth and Ultra Wide Bands (UWB) classified as WPANs, and finally the WLANs and HiperLANs belong to the WLAN class [1].

The recent technological advancement in wireless communications, micro-electro-mechanical systems (MEMS), and digital electronics have led to the development of low-cost, low-power, multifunctional sensor nodes that are small in size and communicate within short distances [2]. These tiny sensor nodes consist of sensing, data processing, and communicating components. The sensor nodes can be interconnected to form a network defined as wireless sensor network (WSN).

One of the most famous initiatives consolidating the possible deployment of WSN systems was the IEEE802.15.4 which specified a physical (PHY) and a medium access control (MAC) layer dedicated for low-rate wireless personal area network (LR-WPAN). The main motivation of IEEE802.15.4 is to develop a dedicated standard, and not to rely on existing technologies like Bluetooth or WLAN, and to ensure low complexity energy efficient implementations. IEEE 802.15.4 offers simple energy efficient, and inexpensive solution to a wide variety of applications in WSNs. It supports simple one hop star network and multi-hop peer-to-peer network [3]. Wireless links under IEEE 802.15.4 can operate in three license free industrial scientific medical (ISM) frequency bands. These accommodate over air data rates of 250 kb/sec in the 2.4 GHz band, 40 kb/sec in the 915 MHz band, and 20 kb/sec in the 868 MHz. In total, 27 channels are allocated in 802.15.4, with 16 channels in the 2.4 GHz band, 10 channels in the 915 MHz band, and 1 channel in the 868 MHz band [4].

WSNs may consist of large number of sensor nodes, which are densely deployed in close proximity to the phenomenon. In WSNs, sensors gather information about the physical world and the base station or the sink node makes decision and performs appropriate actions upon the environment. This technology enables a user to effectively sense and monitor from a distance [2, 5]. The envisaged size of a single sensor node can vary from shoebox-sized nodes down to size of a grain of dust [6]. The cost of sensor nodes varies similarly, ranging from hundreds of U.S. dollars to a few cents, depending on the size of the sensor network and the complexity of individual sensor node. Size and cost constraints on sensor nodes lead to the corresponding limitations on resources such as energy, memory, computational speed and bandwidth [6].

WSNs are very data-centric, meaning that the information that has been collected about an environment must be delivered in a timely fashion to a collecting agent or base station. Since large numbers of sensor nodes are densely deployed, neighbour nodes may be very close to each other. Hence, multi-hop routing idea is suitable for WSN to enable channel reuse in different regions of WSN and overcome some of the signal propagation effects experienced in long-distance wireless communication [2, 5, 6].

The sensory data reflects the physical status of the sensing environment. Thus, the sensor data is valid only for a limited time duration, and hence needs to be delivered within such time bounds called “deadline”. WSNs demand real-time routing which means messages in the network are delivered within less than their end-to-end deadlines (packet lifetime) [5]. More importantly, different sensory data has a different deadline depending on the dynamics of the sensed environment. For example, sensory data for a fast moving target has shorter deadline than that for a slow moving target. In essence, sensor network applications require delivery of various types of sensory data through multi-hop routing with different levels of end-to-end deadline [7].

## 1.2 Application of WSNs

The development of WSNs was originally motivated by military applications such as battlefield surveillance. However, WSNs are now used in many civilian application areas. A sensor node may have different types of sensors such as seismic, magnetic, thermal, visual, infrared, acoustic and radar sensor [5]. A WSN may be able to monitor a wide variety of ambient conditions that include temperature, humidity, movement, lightning condition and pressure. It also can be used to monitor soil makeup, noise levels, the presence or absence of certain kinds of objects, mechanical stress levels on attached objects, and the current characteristics such as speed, direction, and size of an object [2].

Sensor nodes can be used for continuous sensing of event detection, event ID, location, and local control of actuators. For example, the physiological data about a patient can be monitored remotely by a doctor. While this is more convenient for patients, it also allows the doctor to better understand the patient’s current condition [5, 8].

WSNs can also be used to detect foreign chemical agents in the air and the water. They can help to identify the type, concentration, and location of pollutants. In

essence, WSNs will provide the end user with intelligence and a better understanding of the environment [2, 8].

### **1.3 Problem Statement**

The general research challenges for multi-hop routing in WSN arise primarily due to the large number of constraints that must be simultaneously satisfied. One of the most important constraints on sensor nodes is the power consumption requirement. Sensor nodes carry limited, generally irreplaceable power sources. WSN applications must operate for months or years without wired power supplies and battery replaced or recharged. Therefore, the power consumption must be considered while designing multi-hop routing in order to prolong the WSN lifetime [9].

Most low-power wireless networks usually have unreliable links with limited bandwidth, and their link quality can be heavily influenced by environmental factors [10, 11]. Recent empirical results obtained on the Berkeley mote platform indicate that wireless links are highly probabilistic, asymmetric, and the link quality (i.e., packet reception rate (PRR)) depends on the transmission power and the distance traveled by a packet [10, 12]. As a result, communication delays in such system are highly unpredictable. Consequently, the link quality between sensor nodes in WSN should be considered while designing multi-hop routing in order to achieve high throughput for WSN.

Real-time routing protocols designed for WSN must therefore balance real-time performance and energy efficiency.

## 1.4 Objectives

To address the above challenges in WSN, this research proposes a new routing protocol that will efficiently forward the packet from the source to the destination. The objectives of the proposed research are:

- To develop a routing protocol that will provide real-time routing for WSNs
- To prolong the lifetime of WSN nodes.
- To achieve high delivery ratio while utilizing low packet overhead and low power consumption.

A real-time with load distribution (RTLTD) routing protocol is proposed to provide high packet delivery ratio with minimum control packet overhead and efficient power consumption for routing in WSN. Packets in the proposed routing protocol should be delivered within less than their deadline time in order to satisfy real-time routing feature. The proposed routing protocol should ensure periodic selection of forwarding candidates neighbour nodes that distribute the traffic load to those neighbours in the direction of the sink. Selective optimal forwarding node may results in prolonging the WSN lifetime.

## 1.5 Scope

In order to achieve the objectives mentioned earlier, RTLTD routing protocol has been developed from concept design taking into account of the required features of having real-time routing and distributed load balancing. The development of RTLTD is divided into three technical phases that include designing of the routing protocol, simulation of the routing protocol and the test bed implementation which are explained as follows:

### i) Design of real-time routing protocol

The design of the proposed routing protocol consists of four functional modules that include location management, power management, neighbourhood

management and routing management. These functional modules will cooperate to provide real-time routing protocol with distributed load balancing in WSN. The design of state machine and flow chart diagrams have been developed for the proposed routing. The algorithm for each functional module in the proposed system has been developed and the relations between the functional modules has been studied. These include the study of the mathematical equations of link quality, packet delay, remaining power, and exhaustive optimization, probability of collision due to control packet replies and sensor node location determination. The link quality has been estimated based on MICAz and TELOSB sensor nodes.

## **ii) Simulation study of the proposed routing protocol**

The proposed routing protocol has been developed from scratch in Network simulator-2 (NS-2). NS-2 is used to simulate the RTLD routing protocol based on IEEE 802.15.4 MAC and physical layers. The simulation should reflect real access mechanism. The performance in term of delivery ratio, power consumption, and packet overhead have been studied and compared with the existing routing real-time routing protocols such as MM-SPEED [7].

## **iii) Test bed implementation of the proposed routing protocol**

TinyOS operating system and network embedded systems C (nesC) programming language will be used to develop the proposed routing source code based on MICAz and TELOSB sensor node. The TOSSIM program will be used to simulate and test the developed code before it is uploaded into the real sensor node. The test bed network used 25 sensor nodes distributed on the field to read the temperature sensory data. Performance of the test bed implementation and the simulation will be compared.

## **1.6 Significance of Research Work**

The proposed routing protocol can be used to transfer real-time data within 250 ms deadline from the source node to the base station. It can be applied in many WSN applications. For example, in a fire fighting application, appropriate actions should be taken immediately as delay may cause further damages. Similarly, the proposed routing system can be used in military application. If sensors detect a malicious node in an unauthorized area and transmit that information immediately to the security manager, then it becomes very easy to take preventive actions.

In addition, the proposed routing prolongs the lifetime of the individual sensor node and the entire WSN. Therefore, the proposed routing can be applied to monitor the habitat of rural regions where batteries replacement or recharging are expensive.

## **1.7 Thesis Organization**

This thesis consists of six chapters. Chapter 1 serves as an introduction to the thesis. It covers topics such as problem statement, objective of the research, scope of the project and the significance of the project.

Chapter 2 provides the relevant background of understanding WSNs and IEEE 802.15.4. The comparison between WSN and ad hoc network has been presented. This chapter introduces the challenges of WSN. Routing challenges on WSN is described and the relevant routing protocol related to the proposed routing has been studied and compared with RTLD routing protocol.

Chapter 3 describes the proposed system design. The flow chart diagram and state machine diagram are described in the chapter. It includes the four functions of RTLD which are routing management, neighbourhood management, location management, and power management. The optimal path equations are described. In



addition, it includes network parameter configuration and performance analysis equations.

Chapter 4 describes the simulation details of the proposed routing in WSNs. This chapter also compares the performance of RTLD routing with the existing real-time routing such as MM-SPEED.

Chapter 5 describes the hardware implementation of the proposed routing protocol in MICAz mote using TinyOS operating system, TOSSIM and nesC programming language. The proposed routing algorithm has been tested for one hop and multihop communication in WSNs. Chapter 5 also discusses the tools and software requirements to create a real test bed experiment. A graphical user interface (GUI) development and the application of the real-time in WSN are also presented in this chapter.

Finally, Chapter 6 concludes the thesis with a summary of the work that has been done, along with suggestions for future work.

## **CHAPTER 2**

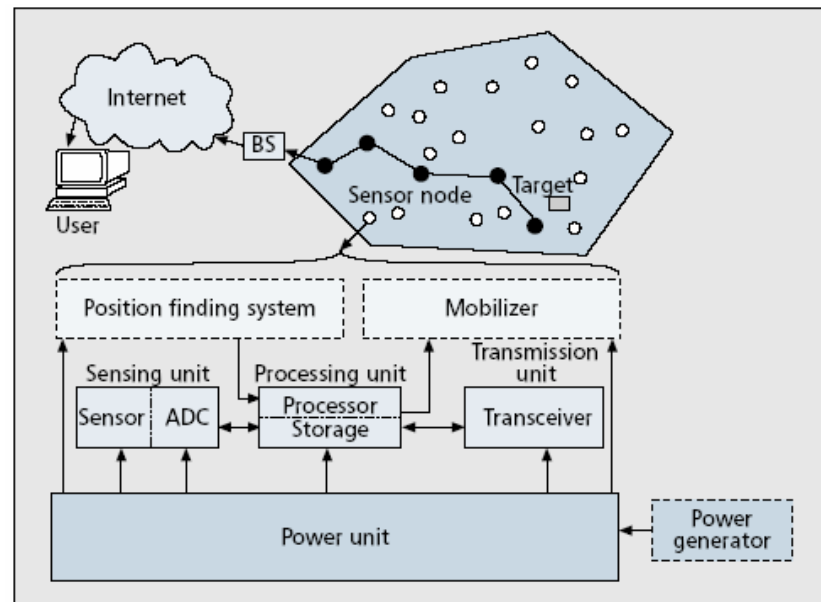
### **LITERATURE REVIEW**

#### **2.1 Introduction**

Wireless multi-hop ad hoc networking techniques constitute the basis for WSNs. An ad hoc network is a peer-to-peer wireless network, which consists of nodes that are connected to each other without infrastructure. In multi-hop routing, the nodes in a network can serve as routers and hosts, they can forward packets on behalf of the other nodes and run user applications [17]. However, the ad hoc solutions are not suitable for WSNs due to the special constraints and application requirements of sensing devices such as memory storage, power limitation and unreliable wireless communication [18]. Many new algorithms have been proposed for the routing in WSNs. These routing mechanisms have taken into consideration the inherent features of WSNs along with the application and architecture requirements. The task of finding and maintaining routes in WSNs are not trivial since energy restrictions and unexpected changes in node status (e.g., failure) cause frequent and unpredictable topological changes [8]. This chapter will present the architecture of WSN based on IEEE 802.15.4 standard. It will also present routing challenges and the related works to the proposed routing protocol in WSNs.

## 2.2 Architecture of WSNs

Figure 2.1 shows the schematic diagram of sensor node components. Each sensor node comprises of sensing, processing, transmission, mobilizer, position finding system, and power units (some of these components are optional like the mobilizer). The same figure shows the communication architecture of a WSN. Sensor nodes are usually scattered in a sensor field and sensor nodes coordinate among themselves to produce high-quality information about the physical environment. Each sensor node makes its decisions based on its mission, the current information and its knowledge of its computing, communication, and energy resources. Each of these scattered sensor nodes has the capability to collect and route data either to other sensors or back to an external base station(s). A base-station may be a fixed node or a mobile node capable of connecting the sensor network to an existing communications infrastructure or to the Internet where a user can have access to the reported data [8].



**Figure 2.1** WSNs architecture [13]

IEEE 802.15.4 protocol specifies a global standard on physical and MAC layers for low data rate, low power, low cost and short range that make IEEE 802.15.4 suitable for WSNs [6, 7, 19].

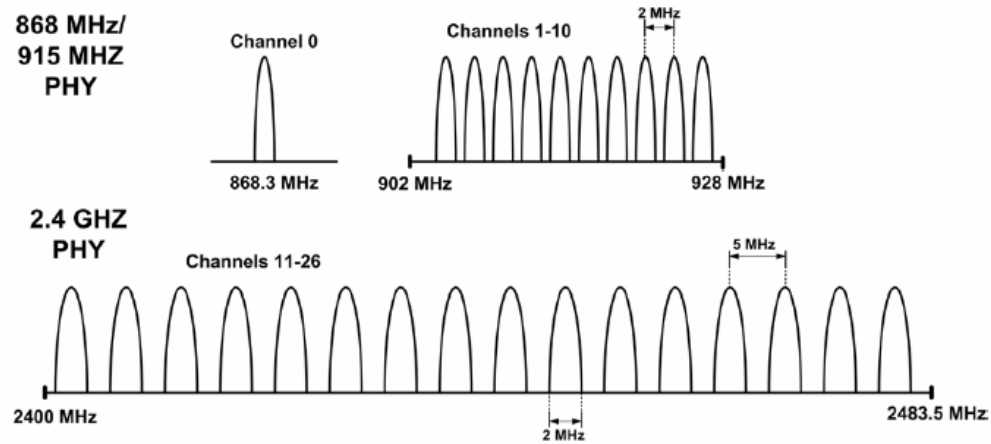
### 2.2.1 IEEE 802.15.4 Specifications

IEEE 802.15.4 is a new standard uniquely designed for LR-WPANs. It offers three operational frequency bands: 2.4 GHz, 915 MHz and 868 MHz as depicted in Table 2.1 and Figure 2.2 [20]. There is a single channel between 868 and 868.6 MHz, 10 channels between 902 and 928 MHz, and 16 channels between 2.4 and 2.4835 GHz. The data rates are 250 kbps at 2.4 GHz, 40 kbps at 915 MHz and 20 kbps at 868 MHz. Lower frequencies are more suitable for longer transmission ranges due to lower propagation losses. However, high data rate transmission provides higher throughput, lower latency and lower duty cycles. All these frequency bands are based on the Direct Sequence Spread Spectrum (DSSS) spreading technique.

The MAC sub-layer of the IEEE 802.15.4 protocol provides an interface between the physical layer and the higher layer protocols of LR-WPANs. It has many common features with the MAC sub-layer of the IEEE 802.11 protocol, such as the use of Carrier Sense Multiple Access / Contention Avoidance (CSMA/CA) as a channel access protocol, the support of contention-free and contention-based periods. However, this new standard does not include the request-to-send (RTS) and clear-to-send (CTS) mechanism as presented in IEEE 802.11. RTS and CTS packets increase the overhead packets sent in IEEE 802.11 and this is not applicable to IEEE 802.15.4 [19].

**Table 2.1:** Physical layer description in IEEE 802.15.4 [19]

Property	Range
Raw data rate	868 MHz: 20 kb/s; 915 MHz: 40 kb/s; 2.4 GHz: 250 kb/s
Range	10 - 20 meters
Latency	Down to 15 ms
Channels	868MHz: 1 channel; 915 MHz: 10 channels; 2.4 Ghz: 16 channels
Frequency band	Two PHYs: 868 MHz/915 MHz and 2.4 GHz
Addressing	Short 16-bit or 64-bit IEEE
Channel access	CSMA-CA and slotted CSMA-CA
Temperature	Industrial temperature range -40 to +85 C



**Figure 2.2** Operating Frequency Bands in IEEE 802.15.4 [19]

The RTS/CTS overhead proves to be useful when traffic load is high, but obviously too expensive for low data rate applications which IEEE 802.15.4 is defined for [7]. Figure 2.3 presents a structure of the IEEE 802.15.4 operational modes. The MAC protocol supports two operational modes that may be selected by the coordinator as explained below:

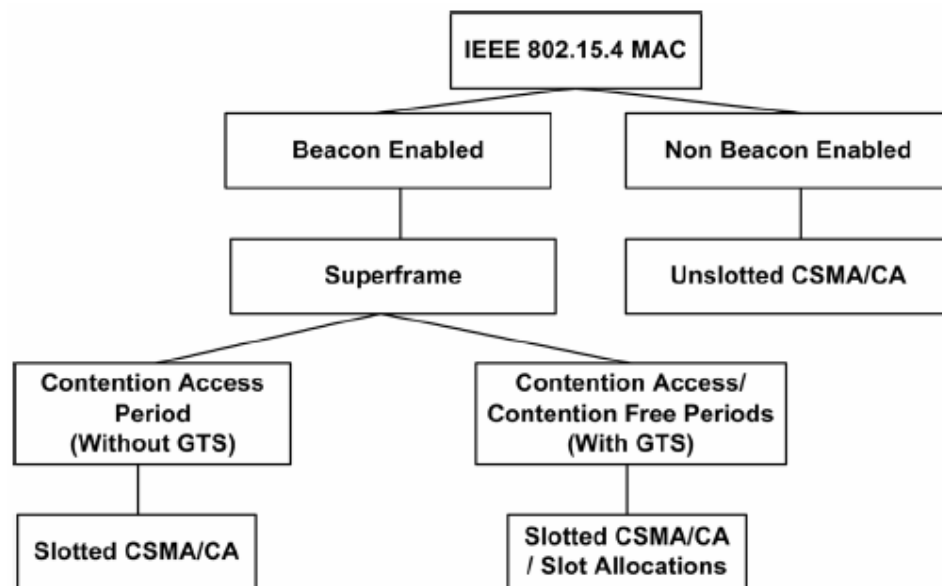
i) **Beacon-enabled mode**

In this operation mode, beacons are periodically generated by the coordinator to synchronize attached devices and to identify the Personal Area Network (PAN) coordinator. A beacon frame is the first part of a super-frame, which embeds all data frames exchanged between the nodes and the PAN coordinator [7, 19, 20]. A data transmission between nodes is also allowed during the super-frame duration. The format of the super-frame is defined by the PAN coordinator and transmitted to other devices inside every beacon frame, which is broadcasted periodically by the PAN coordinator. The super-frame is divided into 16 equally sized slots and is followed by a predefined inactive period. The super-frame lies within beacon interval, which is bounded by two consecutive beacon frames as shown in Figure 2.4 and Figure 2.5. Either a super-frame consists of Contention-Access Period (CAP) or CAP and Contention-Free Period (CFP) as follows:

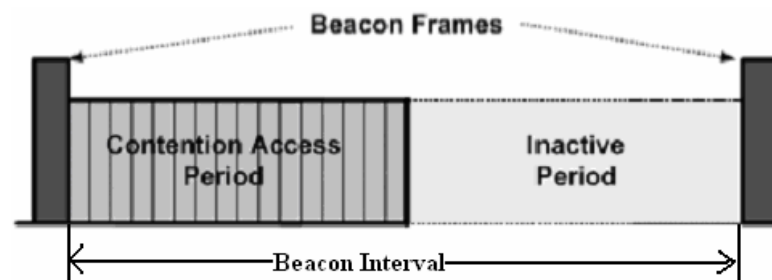
- If communications are restricted to the CAP (defined in the beacon, issued by the PAN Coordinator) a device wishing to communicate must compete with

other devices using a slotted CSMA/CA mechanism. All transmissions must be finished before the end of the super-frame, i.e., before the beginning of the inactive period (if exists) [7, 19, 20]

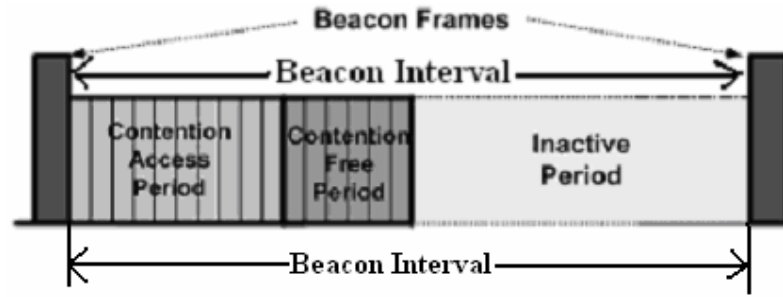
- If guaranteed QoS such as low latency is to be supported, then CFP is defined. The PAN coordinator may allocate up to seven Guaranteed Time Slots (GTSs) and each GTS may occupy more than one time slot. With this super-frame configuration, all contention-based communication must end before the start of the CFP, and a node transmitting a GTS must ensure that its transmission will be completed before the start of the next GTS (or the end of the CFP). According to the standard, the GTS is used only for communications between a PAN coordinator and a device [19, 20].



**Figure 2.3** IEEE 802.15.4 operational modes [19]



**Figure 2.4** The super-frame structure without GTSs [19]



**Figure 2.5** The super-frame structure with GTs [19]

## ii) Non Beacon-enabled mode

In non beacon-enabled mode, the devices can simply send their data by using unslotted CSMA/CA. The super-frame structure is not used in this mode. Unslotted CSMA/CA means that each time a device wishes to transmit data frames or MAC commands, it shall wait for a random period. If a channel is idle following the random backoff, the device shall transmit its data. If the channel is busy following the random backoff, the device shall wait for another random period before trying to access the channel again [7, 19, 20]. Acknowledgment frame is sent without using a CSMA-CA mechanism. Hence, non beacon-enabled mode is used in the proposed research to avoid synchronization problems due to increasing delay, increasing number of sensor nodes and inaccurate reference clock.

## 2.3 Challenges in WSNs

One of the main design goals of WSNs is to carry out data communication while trying to prolong the lifetime of the network and prevent connectivity degradation by employing aggressive energy management techniques [8]. WSNs are influenced by many challenging issues such as node deployment, data processing and routing, fault tolerance, data aggregation and connectivity [8, 25]. These challenges arise primarily due to the large number of constraints such as energy, memory, computational speed and bandwidth [4].

### 2.3.1 Constrain of WSNs

A WSN is a special network, which has many constraints compared to a traditional computer network. Due to the constraints which are discussed below, it is difficult to employ the existing wireless approaches directly to WSNs [26].

#### ♦ Very Limited Resources

All wireless approaches require a certain amount of resources for the implementation such as data memory, code space, and energy to power the sensor. However, currently these resources are very limited in tiny WSNs.

##### i) **Limited Memory and Storage Space**

A sensor is a tiny device with only a small amount of memory and storage space for coding. In order to build an effective routing mechanism, it is necessary to limit the code size of the routing algorithms. For example, one common sensor of MICA2 has an 8-bit, 7.37 MHz CPU with only 4KB SRAM, 128KB program memory, and 512K flash storage [27]. With such limitation, the software built for the sensor must also be quite small. The total code space of TinyOS is approximately 4KB [28], and the core scheduler occupies only 178 bytes. Therefore, the code size of the routing must also be small.

##### ii) **Power Limitation**

Energy is the biggest constraint to WSN capabilities. We assume that once sensor nodes are deployed in a sensor network, they cannot be easily replaced (high operating cost) or recharged (high cost of sensors) [26]. Therefore, the battery charge taken with them to the field must be conserved to extend the life of the individual sensor node and the entire sensor network.



## ◆ **Unreliable Communication**

Certainly, unreliable communication is another threat to sensor node in WSNs. Unreliable communication influences by many reasons that include

### **i) Unreliable Transfer**

Normally the packet-based routing of the sensor network is connectionless and thus inherently unreliable. Packets may get damaged due to channel errors or dropped at highly congested nodes. Furthermore, the unreliable wireless communication channel also results in damaged packets. Higher channel error rate also forces the software developer to devote resources to error handling [4, 26]. More importantly, if the protocol lacks the appropriate error handling it is possible to lose critical packets such as security packets. This may include, for example, a cryptographic key.

### **ii) Conflicts**

Even if the channel is reliable, the communication may still be unreliable. This is due to the broadcast nature of the WSNs. If packets meet in the middle of transfer, conflicts will occur and the transfer itself will fail. In a crowded (high-density) sensor network, this can be a major problem [4, 26].

### **iii) Latency**

The multi-hop routing, network congestion and node processing can lead to greater latency in the network, thus making it difficult to achieve synchronization among sensor nodes [4, 26]. In security scenario, the synchronization issues can be critical to sensor security where the security mechanism relies on critical event reports and cryptographic key distribution.

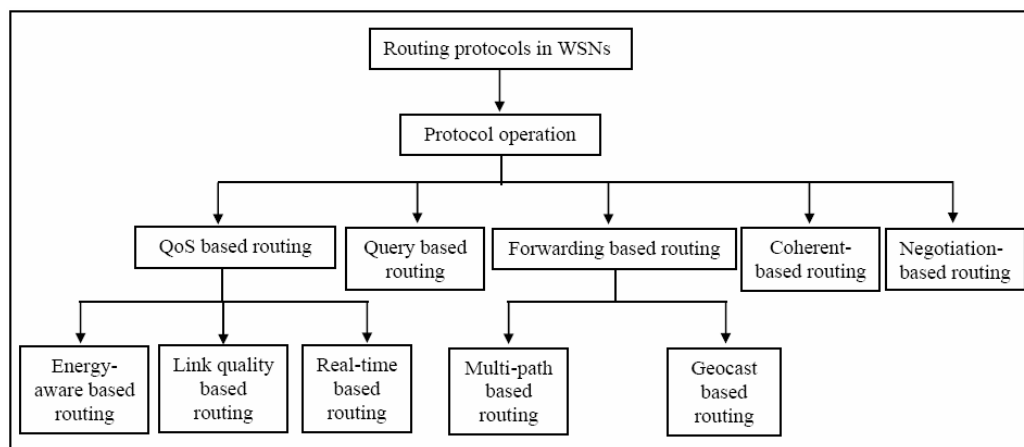
In short, WSN constrains mainly affect data routing between sensor nodes [4, 8, 26].

### 2.3.2 Routing in WSNs

Routing in WSNs is very challenging due to the above mentioned constraints that distinguish these networks from other wireless networks like mobile ad hoc networks or cellular networks [8]. First, due to the relatively large number of sensor nodes, it is not possible to build a global addressing scheme for the deployment of a large number of sensor nodes as the overhead of ID maintenance is high [2, 8]. Furthermore, sensor nodes that are deployed in an ad hoc manner need to be self-organized, as the ad hoc deployment of these nodes requires the system to form connections and cope with the ensuing load distribution. Second, in contrast to typical communication networks, almost all applications of sensor networks require the flow of sensed data from multiple sources to a particular base station. However, this does not prevent the flow of data to be in other forms (e.g., multicast or peer to peer). Third, sensor nodes are tightly constrained in terms of energy, processing, and storage capacities. Thus, they require careful resource management. Fourth, in most application scenarios, nodes in WSNs are generally stationary after deployment except for maybe a few mobile nodes. Nodes in other traditional wireless networks are free to move, which results in unpredictable and frequent topological changes. Fifth, sensor networks are application-specific (i.e., design requirements of a sensor network change with application). For example, the challenging problem of low-latency precision tactical surveillance is different from that of a periodic weather monitoring task. Sixth, position awareness of sensor nodes is important since data collection is normally based on the location. Currently, it is not feasible to use Global Positioning System (GPS) hardware for this purpose. Methods based on triangulation allow sensor nodes to approximate their position using radio strength from a few known points [29]. It is found in [29] that algorithms based on triangulation or multilateration can work quite well under conditions where only very few nodes know their positions a priori. Finally, data collected by many sensors in WSNs is typically based on common phenomena, so there is a high probability that this data has some redundancy. Such redundancy needs to be exploited by the routing protocols to improve energy and bandwidth utilization.

In general, routing in WSNs can be classified into four groups that depend on network structure, protocol operation, determined routing and cooperative routing [8]. Routing based on network structure can be divided into flat-based routing, hierarchical-based routing, and location-based routing. In flat-based routing, all nodes are typically assigned equal roles or functionality. In hierarchical-based routing, nodes will play different roles in the network. In location-based routing, sensor nodes positions are exploited to route data in the network. Furthermore, these protocols can be classified into forwarding based, query-based, negotiation-based, QoS-based, and coherent-based routing techniques depending on the protocol operation. In addition to the above, routing protocols can be classified into three categories proactive, reactive, and hybrid, depending on how the source finds a route to the destination. In proactive protocols, all routes are computed before they are really needed, while in reactive protocols, routes are computed on demand. Hybrid protocols use a combination of these two ideas. In cooperative routing, nodes send data to a central node where data can be aggregated and may be subject to further processing, hence reducing route cost in terms of energy.

In order to study the work related to the proposed routing in this thesis, a classification according to protocol operation (routing criteria) is explained. Figure 2.6 shows the routing based operation in WSNs.



**Figure 2.6** Routing based on protocol operation in WSNs

### 2.3.2.1 QoS based Routing

In QoS based routing protocols, the network has to balance between energy consumption and data quality. In particular, the network has to satisfy certain QoS metrics (delay, energy, bandwidth, etc.) when delivering data to the sink. The QoS based routing can be classified according to routing decision into real-time based routing, link quality based routing and energy-aware routing as shown in Figure 2.6. It should be noted that some of the routing protocols may fall below one or more of the above routing categories.

#### ♦ Real-time Based Routing

A comprehensive review of the challenges and the state of the art of real-time communication in sensor networks can be found in [30]. Chenyang Lu et al develop real-time architecture and protocols (RAP) based on velocity [31]. RAP provides service differentiation in the timeliness domain by velocity-monotonic classification of packets [31]. Based on packet deadline and destination, its required velocity is calculated and its priority is determined in the velocity-monotonic order so that a high velocity packet can be delivered earlier than a low velocity one. Similarly, SPEED is a stateless protocol for real-time communication in WSN. It bounds the end-to-end communication delay by enforcing a uniform communication speed in every hop in the network through a novel combination of feedback control and non-deterministic QoS aware geographic forwarding [9]. MM-SPEED is an extension to SPEED protocol [5]. It was designed to support multiple communication speeds and provides differentiated reliability. Scheduling Messages with Deadlines focuses on the problem of providing timeline guarantees for multi-hop transmissions in a real-time robotic sensor application [32]. In such application, each message is associated with a deadline and may need to traverse multiple hops from the source to the destination. Message deadlines are derived from the validity of the accompanying sensor data and the start time of the consuming task at the destination. The authors propose heuristics for online scheduling of messages with deadline constraints as follow: schedules messages based on their per-hop timeliness constraints, carefully

exploit spatial reuse of the wireless channel and explicitly avoid collisions to reduce deadline misses.

A routing protocol called Real-Time Power Control (RTPC) uses velocity with the most energy efficient forwarding choice as the metrics for selecting forwarding node [33]. A key feature of RTPC is its ability to send the data while adapting to the power of transmission. However, RTPC, RAP, SPEED, MM-SPEED and Scheduling Messages with Deadlines routing protocols depend on the velocity which is not sufficient to provide high throughput in wireless communication. The best link quality usually provides low packet loss and energy efficient [10]. On the other hand, RTPC uses minimum hop count as a metric to provide energy efficient forwarding. However, the minimum hop count affects the delivery ratio [34].

By exploiting the periodic nature of sensor network traffic, Caccamo et al [35] realize collision-free real-time scheduling as follows; frequency division multiplexing (FDM) is used among adjacent cells to allow for concurrent communications in different cells. Implicit earliest deadline first (EDF) scheduling is used inside each cell. There is a router located in the centre area of each cell. Router nodes are equipped with two transceivers so they can transmit and receive at the same time using two different frequency channels. The sensors first exchange the data amongst themselves, perform some computation and then send across the results to the router, which then forwards the data to the next hop. It does not make sense to send across the raw data because it will increase the network traffic significantly. However, this scheme suffers some drawbacks: tight clock synchronization may be required, nodes may rely on a centralized base station, and node failures may waste bandwidth due to fixed reservations.

H. Peng et al [36] propose an adaptive real-time routing scheme (ARP). This scheme provides different real-time levels for different applications and dynamically adjusts the transmission rate of data packets during the end-to-end transmission period. However, ARP does not consider link quality which is important in an unreliable communication in WSNs. In addition, ARP uses the minimum hop as a primary metric in the packet forwarding. Similar to RTPC, the minimum hop count affects the delivery ratio [34].

### ♦ **Link Quality Based Routing**

A routing protocol based on link quality is proposed by D. De Couto et al [37]. The expected transmission count metric (ETX) is developed as a metric to select forwarding node in [37]. ETX finds the path with the minimum expected number of transmissions (including retransmissions) required to deliver a packet all the way to its destination. This metric predicts the number of retransmissions required using per-link measurements of packet loss ratio in both directions of each wireless link. However, ETX does not consider the remaining power and real-time forwarding parameters. The real-time constrain such as end-to-end deadline is important in real-time applications.

Probabilistic Geographic Routing protocol (PGR) is a decentralized energy-aware routing protocol for wireless ad hoc and sensor networks [38]. PGR uses geographical location along with residual energy and link reliability information to make routing decisions. Instead of deterministically choosing the next hop, PGR assigns probabilities to the potential candidate for next hop nodes. The probability assigned to each node is a multiplication function of its residual energy with the corresponding link reliability estimation. However, PGR is not designed for real-time communication but it attempts to minimize the number of retransmissions to save energy, and increase the overall lifetime of the network. In addition, the probability mechanism based on multiplication of residual energy and link reliability only does not provide the optimal forwarding as will be explained later in this thesis.

V.C Gungor et al [39] propose resource-aware and link quality (RLQ) based routing metric for wireless sensor and actor networks (WSANs). The RLQ routing metric is a combined link cost metric, which is based on both energy efficiency and link quality statistics. Based on extensive empirical measurements and test-bed experiments, the authors also found that a strong correlation between the average LQI measurements and packet reception rates exists.

P. Jiang et al [40] propose a link quality estimation based routing protocol

(LQER) to meet the high reliability of transmitting data in water environment. It considers both energy efficiency and link qualities when the route is selected, which makes routing data more reliable and decreases the probability of retransmission, thus saves the energy and prolongs the lifetime of the whole network. Simulation results show that LQER can meet the requirements of energy efficiency, reliability and scalability for water environment monitoring in wetlands.

However, RLQ and LQER do not consider remaining power and packet deadline parameters that are important for real-time load distribution routing.

#### ♦ **Energy-aware Routing**

A. Mahapatra et al [41] develop QoS and energy aware routing for real-time traffic in WSNs. They propose energy aware dual-path routing scheme for real-time traffic, which balances node energy utilization to increase the network lifetime, takes network congestion into account to reduce the routing delay across the network and increases the reliability of the packets reaching the destination by introducing minimal data redundancy. The authors also introduce an adaptive prioritized MAC to provide a differentiated service model for real-time packets. However, QoS and energy aware routing do not consider the link quality and load distribution in the WSNs. If the packet is real-time packet, it is always forwarded to the nearest neighbour. This means that the network lifetime will be decreased.

Energy-aware QoS routing protocol for WSNs is proposed by Akkaya and Younis [42]. Real-time traffic is generated by imaging sensors. The proposed protocol finds the least cost and energy efficient path that meets certain end-to-end delay during the connection. The link cost used is a function that captures the nodes' energy reserve, transmission energy, error rate and other communication parameters. Moreover, throughput for non-real-time data is maximized by adjusting the service rate for both real-time and non-real-time data at sensor nodes. Simulation results show that the proposed protocol consistently performs well with respect to QoS and energy metrics. However, the packet deadline, load distribution and network lifetime are not considered which affect the total performance of WSN.

J.H. Chang, and L. Tassiulas [43] formulate the routing problem as maximizing the network lifetime. They use a shortest cost path routing whose link cost is a combination of transmission and reception energy consumption and the residual energy levels at the two end nodes. However, the authors do not take into account the power consumption due to control packet overhead to which is not a reasonable assumption. In addition, packet deadline is not considered in this routing protocol.

Joongseok et al [44] propose the online maximum lifetime (OML) heuristic to maximize lifetime. They use Dijkstra's shortest path algorithm [45] to reduce sensor energy to a level lower than the energy needed to transmit to its closest neighbour. However, if the forwarding mechanism attempts to maximize per-hop reliability by forwarding only to close neighbours with good links, it may cover only a small geographic distance at each hop. This will eventually result in greater energy expenditure and end-to-end delay due to the need for more transmission hops for each packet to reach the destination [46].

Table 2.2 summarizes QoS-based routing in WSN.



**Table 2.2** Summary of QoS-based routing protocols

<b>Title of the study</b>	<b>Authors</b>	<b>Feature</b>	<b>Limitation</b>
<b>Real-time power control in wireless sensor networks (RTPC)</b>	By O. Chipara et al (IWQoS 2006, June 2006 )	It uses velocity with the most energy efficient forwarding choice as the metrics for selecting next hop. It has ability to send the data while adapting to the power of transmission.	<ul style="list-style-type: none"> <li>• It uses minimum hop as a metric to provide energy efficient forwarding. However, the minimum hop affects the delivery ratio due to unreliable link quality.</li> <li>• Power consumption is not taken into account.</li> </ul>
<b>Probabilistic qos guarantee in reliability and timeliness domains in wireless sensor networks (MM-SPEED)</b>	By E. Felemban et al ( IEEE Conference of the Computer and Communications Societies, 2005 )	It was designed to support multiple communication speeds, multi-path forwarding and to provide service differentiation and probabilistic QoS guarantees in timeliness and reliability domains.	<ul style="list-style-type: none"> <li>• It decreases WSN lifetime due to load distribution is not studied.</li> <li>• Power consumption is not taken into account.</li> </ul>
<b>Research on Wireless Sensor Networks Routing Protocol for Wetland Water Environment Monitoring (LQER)</b>	By P. Jiang et al (ICICIC'06 China, 2006 )	It proposes a link quality estimation based routing protocol (LQER) to meet the high reliability of transmitting data in water environment.	<ul style="list-style-type: none"> <li>• Packet deadline is not considered.</li> <li>• Link quality is based on network layer which waste time and power</li> </ul>
<b>High-throughput Path Metric for Multi-hop Wireless Routing</b>	By D. De Couto et al ( MOBICOM conference, Sep 14-19, 2003 )	It measures packet loss ratio in both directions of each wireless link. The expected transmission count metric (ETX) finds paths with the minimum expected number of transmissions.	<ul style="list-style-type: none"> <li>• It decreases WSN lifetime due to load distribution is not studied.</li> <li>• Packet deadline is not considered</li> <li>• Link quality is based on network layer which waste time and power</li> </ul>
<b>QoS and energy aware routing for real-time traffic in wireless sensor networks</b>	By A. Mahapatra et al. ( Computer Communications,	They proposes energy aware dual-path routing scheme for real-time traffic, which balances node energy utilization to	<ul style="list-style-type: none"> <li>• It increases power consumption because packet always forwarded to the nearest neighbour. It maximizes</li> </ul>

	Elsevier Journal, February 2006 )	increase the network lifetime, takes network congestion into account to reduce the routing delay across the network and increases the reliability of the packets reaching the destination by introducing minimal data redundancy.	number of hop between the source and destination that increases end to end delay.
<b>Maximum lifetime routing in wireless sensor networks</b>	By J.H. Chang, and L. Tassiulas ( IEEE Journal, Aug. 2004 )	They use a shortest cost path routing whose link cost is a combination of receiving energy consumption and the residual energy levels at the two end nodes.	<ul style="list-style-type: none"> <li>• Power consumption due to control packet overhead is not studied.</li> <li>• Routing based on shortest path is unreliable due to link quality and delay is unpredictable.</li> <li>• Packet deadline is not considered.</li> </ul>
<b>SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks</b>	By John Stankovic et al ( IEEE Journal, Jan 2003 )	It enforces a uniform communication speed in every hop in the network.	<ul style="list-style-type: none"> <li>• One packet speed is used.</li> <li>• It decreases WSN lifetime due to load distribution is not studied.</li> <li>• Link quality is not studied</li> </ul>
<b>RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks</b>	By Chenyang Lu et al ( IEEE conference RTAS'02, 2002 )	RAP prioritizes real-time traffic through a novel velocity monotonic scheduling scheme which considers both a packet's deadline and distance to the destination.	<ul style="list-style-type: none"> <li>• It decreases WSN lifetime due to load distribution is not studied.</li> <li>• Link quality and hole routing problems are not studied.</li> </ul>
<b>An online heuristic for maximum lifetime routing in wireless sensor networks</b>	By J. Park and S. Sahni. (Computers IEEE Journal, Aug. 2006 )	They use shortest path algorithm to reduce sensor energy to a level below that needed to transmit to its closest neighbour.	<ul style="list-style-type: none"> <li>• Routing based on shortest path is unreliable due to link quality and delay is unpredictable.</li> </ul>
<b>Probabilistic Geographic Routing (PGR) in Ad Hoc and Sensor Networks</b>	By T. Roosta ( IWWAN workshop, London, UK, May 2005 )	PGR assigns probabilities to the candidate next hop nodes as a function of its residual energy and the corresponding link reliability estimation.	<ul style="list-style-type: none"> <li>• Packet deadline is not considered</li> <li>• Link quality is based on network layer which waste time and power</li> </ul>

<b>Scheduling Messages with Deadlines in Multi-hop Real-time Sensor Networks</b>	By Huan Li et al ( RTAS 2005, San Francisco, California, March 7 - 10, 2005 )	In this research, each message is associated with a deadline and may need to traverse multiple hops from the source to the destination.	<ul style="list-style-type: none"><li>• It decreases WSN lifetime due to load distribution is not studied.</li></ul>
--	--	---	--

### 2.3.2.2 Forwarding Based Routing

The forwarding based routing is divided into three types of forwarding; unicast forwarding, multi-path forwarding and geocast forwarding. The unicast forwarding was explained previously in QoS based routing. We will explain the multi-path and geocast based routing in the following section.

#### ♦ Multi-path Based Routing

The multi-path routing protocols use multiple paths rather than a single path in order to enhance the network performance. The fault tolerance of a protocol is measured by the probability that an alternate path exists between a source and a destination when the primary path fails. This can be increased by maintaining multiple paths between the source and the destination at the expense of an increased energy consumption and traffic generation. These alternate paths are kept alive by sending periodic messages. Hence, network reliability can be increased at the expense of increased overhead of maintaining the alternate paths.

M. Chen et al addressed the problem of real-time video streaming over a bandwidth and energy constrained WSN from a small number of video-sensor nodes to a sink by combining forward error correction (FEC) coding with a multi-path routing scheme called directional geographical routing (DGR) [3]. DGR constructs multiple disjointed paths for a video-sensor node to transmit parallel FEC-protected H.26L real-time video streams over a bandwidth-limited, unreliable networking environment. The multiple routing in DGR uses shortest path forwarding with greedy forwarding to forward the data packet to the sink. However, the shortest path forwarding creates routing hole problem due to energy expenditure in each sensor node in the shortest path. The multi-path routing in DGR causes data redundancy and more energy expenditure due to forwarding the packet to all neighbours of the source node. DGR does not study the effect of packet deadline and control packet overhead in WSN.

S. Dulman et al [47] propose a multi-path routing that was used to enhance the reliability of WSNs. The proposed scheme is useful for delivering data in unreliable environments. It is known that network reliability can be increased by providing several paths from source to destination and by sending the same packet on each path. However, using this technique, traffic will increase significantly. Hence, there is a tradeoff between the amount of traffic and the reliability of the network. This tradeoff is studied in [47] using a redundancy function that is dependent on the multi-path degree and on failing probabilities of the available paths. The idea is to split the original data packet into sub packets and then send each sub packet through one of the available multi-paths. According to their algorithm, it has been found that for a given maximum node failure probability, higher multi-path degree than a certain optimal value will increase the total probability of failure. S. Dulman et al [47] experience high end-to-end delay due to packet fragmentation.

The proposed multi-path forwarding in RTLD routing protocol is based on directional forwarding that selects the paths based on quadrant. The directional forwarding saves power usage, reduces packet flooding and minimizes collision.

#### ♦ **Geocast Based Routing**

In global flooding, the sender broadcasts the packet to its neighbours. Each neighbour receives the packet; it broadcasts it to its neighbour. This mechanism will continue until all reachable nodes in the geocast region nodes receive the packet. It is simple but has a very high overhead and is not scalable to large and limited networks such as WSNs.

Greedy Perimeter Stateless Routing (GPSR) [48] is a geographic routing protocol for wireless networks that works in two modes; greedy mode and perimeter mode. In greedy mode, each node forwards the packet to the neighbour closest to the destination. When greedy forwarding is not possible, the packet switches to perimeter mode. Perimeter routing (face routing) is used to route around dead-ends until nodes closer to the destination are found.

Ko and Vaidya [49] proposed geocasting algorithms to reduce the overhead, compared to global flooding, by restricting the forwarding zone for geocast packets. Nodes within the forwarding zone forward the geocast packet by broadcasting it to their neighbours and nodes outside the forwarding zone discard it. Each node has a localization mechanism to detect its location and to decide when it receives a packet and whether it is in the forwarding zone or not.

GeoTORA [50] integrates local flooding with Temporally-Ordered Routing Algorithm (TORA [51]) which is a non-location based routing. It uses a TORA routing protocol to unicast the delivery packet to the region and then floods the packet within the region.

K. Seada and A. Helmy [52] proposed two protocols for geocast: Geographic-Forwarding-Geocast (GFG) and Geographic-Forwarding-Perimeter-Geocast (GFPG). In the GFG, GPSR is used by nodes outside the region to guarantee the forwarding of the packet to the region. Nodes inside the region broadcast the packet to flood the region. GFPG uses a mix of geocast and perimeter routing to guarantee the delivery of the geocast packet to all nodes in the region. Although the algorithm solves the region gap problem in sparse networks, it causes unnecessary overhead in dense networks.

The proposed geodirectional-cast in RTLD is a location based routing which is more scalable than non-location based ad-hoc routing protocols and more suitable for sensor networks. Location based routing has several advantages: nodes require only information from their direct neighbours so discovery floods and state propagation are not required. Moreover, it has lower overhead and faster response to dynamics. In addition, due to the forwarding algorithm in [52] which uses single forwarding path to send the data packet toward the destination in geographic region, the delivery ratio is not guaranteed. The geodirectional-cast uses multi-path forwarding toward the destination, which provides more guaranteed delivery ratio and fault tolerance than [52].

Table 2.3 summarizes geocast routing in WSN.

**Table 2.3** Summary of forwarding based routing protocols

<b>Title of the study</b>	<b>Authors</b>	<b>Feature</b>	<b>Limitation</b>
<b>Directional Geographical Routing for Real-Time Video Communications in Wireless Sensor Networks</b>	By M. Chen, V.C.M. Leung, S. Mao, Y. Yuan. (Elsevier Computer Communications Journal, Volume 30, Issue 17, November 2007)	DGR constructs multiple disjointed paths for a video-sensor node to transmit parallel real-time video streams over a WSN. The multiple routing in DGR uses shortest path forwarding with greedy forwarding to forward the data packet to the sink	<ul style="list-style-type: none"> <li>• The shortest path forwarding creates routing hole problem due to energy consumed in each sensor node in the shortest path which will decrease WSN lifetime.</li> <li>• It does not study the effect of packet deadline, and control packet overhead in WSN.</li> </ul>
<b>Efficient and Robust Geocasting Protocols for Sensor Networks.</b>	By Karim Seada and Ahmed Helmy. In Elsevier Computer Communications Journal, Special Issue on Dependable Wireless Sensor Networks, 2005.	It proposed two protocols for geocast: GFG and GFPG. In the GFG, GPSR is used by nodes outside the region to guarantee the forwarding of the packet to the region. Nodes inside the region broadcast the packet to flood the region.	<ul style="list-style-type: none"> <li>• Power consumption is high because packet is flooded.</li> <li>• It does not study the effect of packet deadline, and control packet overhead in WSN.</li> </ul>
<b>Trade-Off between Traffic Overhead and Reliability in Multi-path Routing for Wireless Sensor Networks</b>	By S. Dulman, T. Nieberg, J. Wu, P. Havinga. ( WCNC Workshop, New Orleans, Louisiana, USA, March 2003)	The idea is to split the original data packet into sub packets and then send each sub packet through one of the available multi-paths. It has been found that even if some of these sub packets were lost, the original message can still be reconstructed.	<ul style="list-style-type: none"> <li>• It experiences high delay due to packet fragmentation.</li> <li>• It does not study the effect of packet deadline, and control packet overhead in WSN.</li> </ul>
<b>GPSR: greedy perimeter stateless routing for wireless networks</b>	By B. Karp, and H. Kung ( MOBICOM 2000, Boston, Massachusetts. 6-	It works in two modes: greedy mode; each node forwards the packet to the neighbour closest to the destination.	<ul style="list-style-type: none"> <li>• Power consumption is high because packet is flooded.</li> <li>• It does not study the effect of</li> </ul>

	11 August, 2000.)	Perimeter mode is used to route around dead-ends until closer nodes to the destination are found.	packet deadline, and control packet overhead in WSN.
<b>Flooding-based geocasting protocols for mobile ad hoc networks</b>	By Ko and Vaidy ( MONET Journal, ACM/Baltzer 2002.)	It restricts the forwarding zone for geocast packets. Nodes within the forwarding zone forward the geocast packet by broadcasting it to their neighbours and nodes outside the forwarding zone discard it.	<ul style="list-style-type: none"> <li>• Power consumption is high because packet is flooded.</li> <li>• It does not study the effect of packet deadline, and control packet overhead in WSN.</li> </ul>
<b>Anycasting-based protocol for geocast service in mobile ad hoc networks</b>	By Y. Ko (Computer Networks, Elsevier North-Holland Journal, Volume 41 , Issue 6 2003. Pages: 743 – 760. )	It integrates TORA with local flooding. It uses a TORA routing protocol to unicast the delivery packet to the region and then floods the packet within the region.	<ul style="list-style-type: none"> <li>• Power consumption is high because packet is flooded.</li> <li>• It does not study the effect of packet deadline, and control packet overhead in WSN.</li> </ul>



### **2.3.2.3 Query Based Routing**

In this kind of routing, the destination nodes propagate a query for data sensing task through the network. If the sensor node matches the query and has sensory data, it will send the data to the destination node that initiates the query. Usually these queries are described in natural language, or in high-level query languages.

Directed diffusion [53] is an example of this type of routing. In directed diffusion, the sink node sends out interest messages to sensors. As the interest message is propagated throughout WSN, the gradients from the source back to the sink are set up. When the source has data for the interest, the source sends the data along the interests gradient path.

The direct rumor routing protocol [54] uses geographical information to help the traditional rumor routing increases the delivery ratio and decreases the power consumption. It routes the event agents and the query agents in straight lines centered at the source point and the sink point respectively. When a node senses an event, it creates a number of event agents and propagates them into the network along some linear paths towards the sink.

### **2.3.2.4 Negotiation-Based Routing**

In this routing, communication decisions depend on the resources availability. The negotiation-based routing in WSNs suppresses duplicate information and prevents redundant data from being sent to the next sensor or the sink by conducting a series of negotiation messages before the real data transmission begins.

The protocol in [55] is an example of negotiation-based routing protocols. The authors present a family of adaptive protocols, called SPIN (Sensor Protocols for Information via Negotiation). Nodes running a SPIN communication protocol name their data using high-level data descriptors, called meta-data. They use meta-data negotiations to eliminate the transmission of redundant data throughout the network. In addition, SPIN nodes can base their communication decisions both upon

application-specific knowledge of the data and upon knowledge of the resources that are available to them.

### **2.3.2.5 Coherent-Based Routing**

In general, sensor nodes will cooperate with each other to process different data flooded in the network area. Two examples of data processing techniques in WSNs are coherent and non-coherent [56]. In non-coherent data processing routing, nodes will locally process the raw data before sending it to other nodes for further processing. The nodes that perform further processing are called aggregators. In coherent routing, the data is forwarded to aggregators after least processing tasks like time stamping and duplicate suppression.

In [56], single and multiple winner algorithms were proposed for non-coherent and coherent processing respectively. In the single winner algorithm (SWE), a single aggregator node is elected for complex processing. The election of a node is based on the energy reserves and computational capability of that node. By the end of the SWE process, a minimum-hop spanning tree will completely cover the network. In the multiple winner algorithm (MWE), a simple extension to SWE is proposed. When all nodes are sources and send their data to the central aggregator node, a large amount of energy will be consumed. Hence, this process has a high cost. One way to lower the energy cost is to limit the number of sources that can send data to the central aggregator node. Instead of keeping a record of only the best candidate node (master aggregator node), each node will keep a record of up to  $n$  nodes of those candidates. At the end of the MWE process, each sensor in the network has a set of minimum-energy paths to each source node. After that, SWE is used to find the node that yields the minimum energy consumption. This node can then serve as the central node for coherent processing. In general, the MWE process has longer delay, higher overhead, and lower scalability than SWE for non-coherent processing networks.

## 2.4 Summary

This chapter presented an overview of WSN challenges. The MAC and physical layers are based on IEEE 802.15.4 which is designed for low rate communication such as WSN. This chapter concludes that the real-time routing design in WSN are not easy works due to the numerous constraints in WSN such as memory storage, power limitation and unreliable wireless communication. The aforementioned limitations should be considered when real-time routing is designed. In RTLD, the optimal value based on the weighting of velocity, PRR and remaining power mechanism are used to select forwarding node. RTLD can be adapted for two types of communication; geodirectional-cast and unicast forwarding. RTLD routing scheme possesses built-in security. The following chapters will elaborate the system design concepts of RTLD routing protocol for WSNs.

## **CHAPTER 3**

### **RTLD SYSTEM DESIGN**

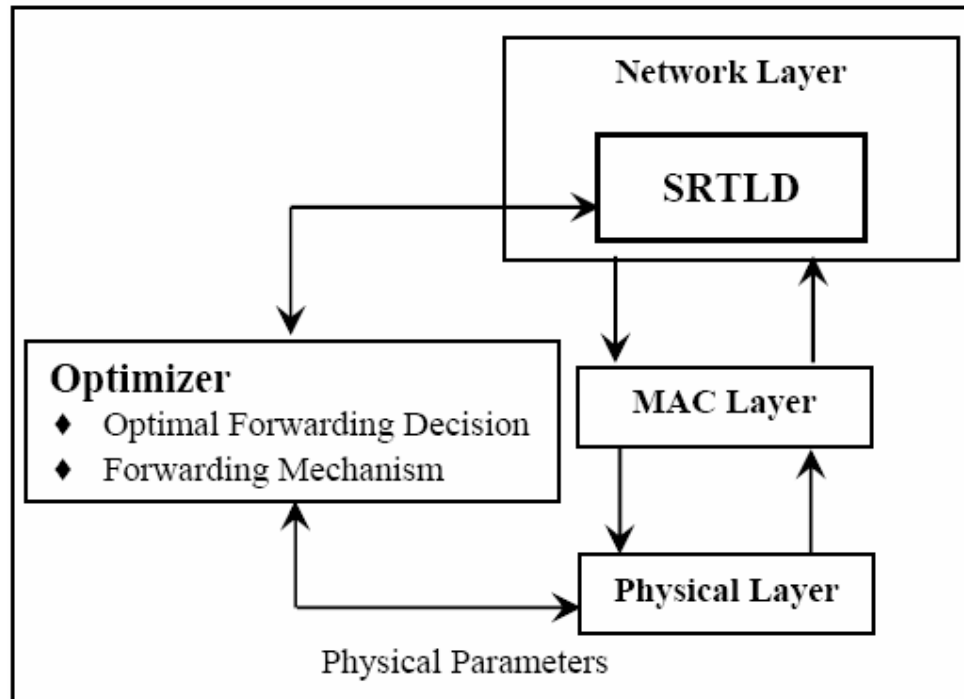
#### **3.1 Introduction**

A real-time communication in WSN is important in monitoring disasters such as fire detection and flooding. Different sensory data has a different time deadline depending on the dynamics of the sensed environment. Moreover, WSN applications must operate for months or years without battery replaced or recharged. Further, most real-time communication did not address security, so it is easy for an adversary to attack WSNs. The proposed routing protocol is designed to solve the above problems while achieving high performance in term of delivery ratio, packet overhead, and power consumption.

In this chapter, the design concepts of RTLD will be explained in detail. RTLD consists of four functional modules that include location management, power management, neighbour management, and routing management. These functions cooperate and coordinate with each other to provide secure real-time routing protocol that ensure high delivery ratio for real-time packet delivery and longer WSN lifetime. The security in the proposed RTLD is further enhanced by including security enhancement mechanism to overcome the selective forwarding attack and HELL flooding attack. The following section will elaborate the design concepts of RTLD with security enhancement.

### 3.2 Cross-Layer Design in RTLD

In order to achieve high gains in the overall performance of WSN, cross-layer interaction is used in the design of RTLD. The concept of cross-layer design is about sharing of information among two or more layers for adaptation purposes and to increase the inter-layer interactions [36, 41, 74]. The proposed system uses interaction between physical layer and network layer in order to select the next hop forwarding as shown in Figure 3.1. The network process at the network layer optimizes the optimal forwarding decision based on the physical parameters translated as forwarding metrics. The physical parameters are the signal strength, remaining power and timestamp. The forwarding metrics is used to determine the next hop communication. The optimization of the routing mechanism is done using exhaustive search techniques. The features of the forwarding metrics are explained in details in the next section. The forwarding metrics are requested only during neighbour discovery and network initialization as explained in section 3.3.3.1.

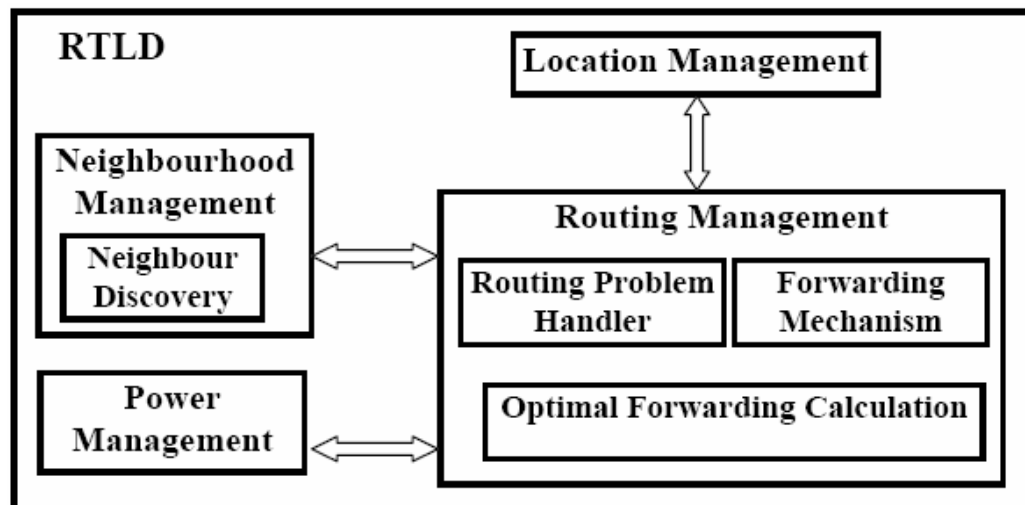


**Figure 3.1** Cross-layer concepts in RTLD

### 3.3 RTLD design Concepts

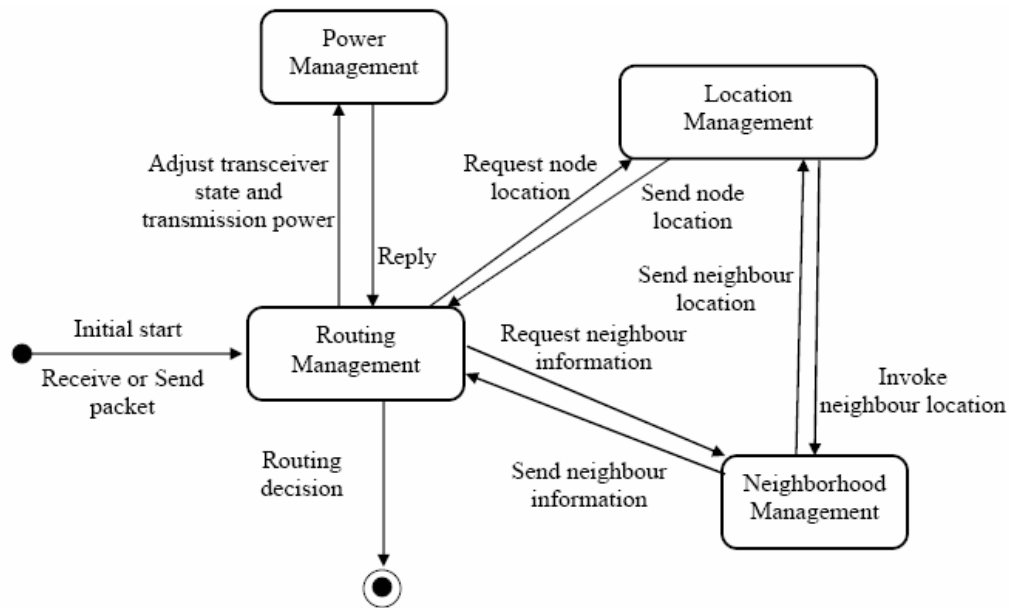
In order to develop real-time routing in WSN, the packet deadline (velocity) is utilized in the forwarding calculation. The wireless link quality at the physical layer is studied to predict the communication between sensors. In addition, the remaining power is estimated to spread all traffic load distribution during path forwarding to the destination.

In Figure 3.2, RTLD consists of four functional modules that include location management, routing management, power management and neighbourhood management. The location management in each sensor node calculates its location based on the distance to three pre-determined neighbour nodes. The power management determines the state of transceiver power and the transmission power of the sensor node. The neighbourhood management discovers a subset of forwarding candidate nodes and maintains a neighbour table of the forwarding candidate nodes. The routing management computes the optimal forwarding choice, makes forwarding decision and implements routing problem handler. A new type of forwarding mechanism in WSN called geodirectional-cast forwarding based on quadrant is proposed. Geodirectional-cast forwarding combines geocast with directional forwarding to forward the data packet through multiple paths to the destination.



**Figure 3.2** Functional components of RTLD

Routing management is the main module in RTLD. It cooperates with the other modules and coordinates routing process in order to carry out the RTLD routing protocol. Figure 3.3 shows the state machine diagram of RTLD. In this figure, the routing management sends request to the location management to invoke sensor node location. Then, the location management sends request to the neighbourhood management to reveal three pre-determined neighbour nodes. The neighbourhood management invokes neighbour discovery if its neighbour table is unable to meet the request.. The location management calculates the sensor node location and sends it to the routing management. The sensor node location is defined in request-to-route (RTR) control packet, and reply RTR control packet. The location information will be used by the geodirectional-cast mechanism to forward data packets to the destination. Whenever the routing management sends packet to its neighbour, it instructs the power management to change the transceiver state from idle to transmit or from receive to transmit and adjust the power level of the transceiver to optimum power usage. Finally, routing management forwards the data packet based on the forwarding mechanism to the selected neighbour

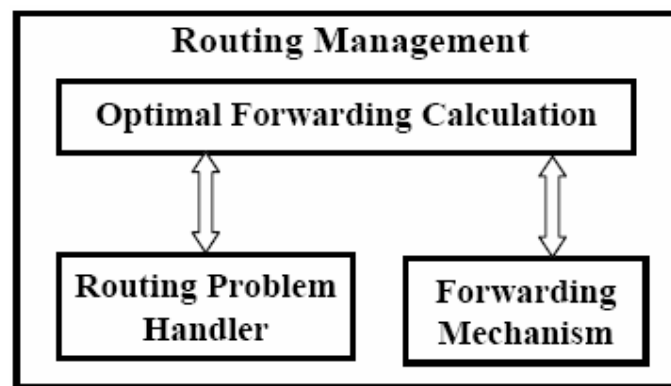


**Figure 3.3** State machine diagram of RTLD

### 3.3.1 Routing Management

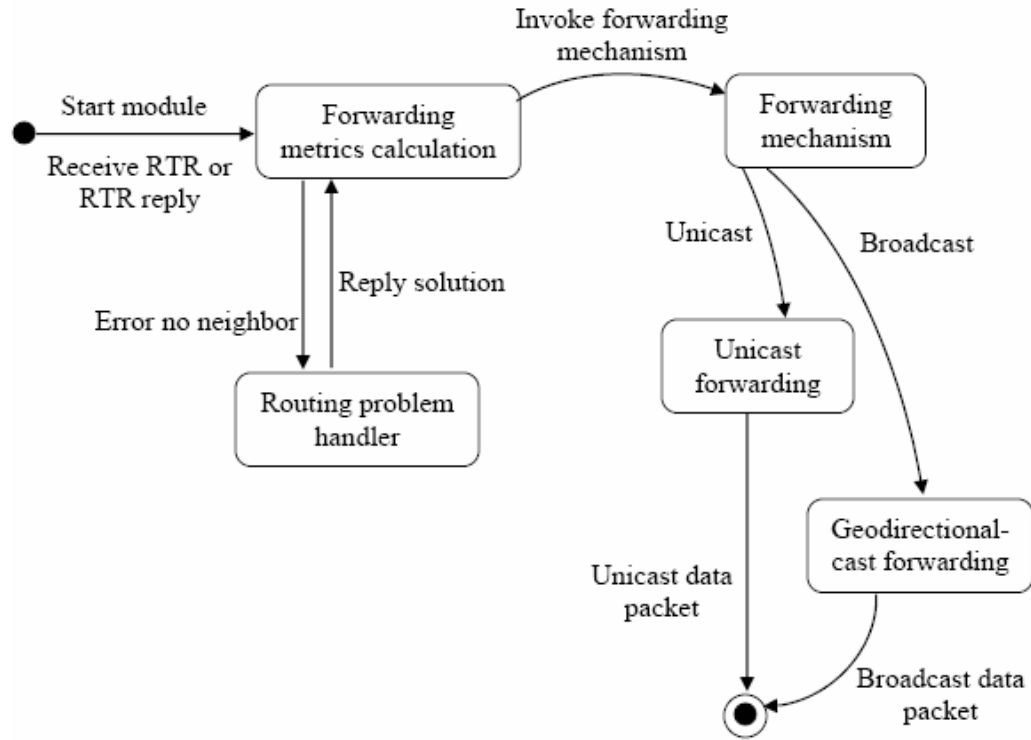
The routing management consists of three sub functional processes; forwarding metrics calculation, forwarding mechanism and routing problem handler as shown in Figure 3.4. In this figure, optimal forwarding calculation is used to calculate next hop based on the forwarding metrics that include packet reception rate (PRR), packet velocity and remaining power. The routing problem handler is used to solve the routing hole problem due to hidden sensor nodes in WSN. Unicast and geodirectional-cast are the mechanisms used to select the way to forward data.

Figure 3.5 shows the state machine diagram of the routing management. When the sensor node receives RTR or reply RTR packets from its neighbour, it will calculate the forwarding metrics for each packet. The forwarding metrics are calculated from the physical parameters which include signal strength, remaining power and timestamp identified in the RTR packets. The routing management stores the replies in the neighbour table that located in the neighbourhood management. Finally, the routing management selects the best progress neighbour and the forwarding mechanism either unicast or geodirectional-cast. In the case of the source node does not receive any RTR or RTR reply packets, the routing problem handler is invoked to deal with neighbour hidden problem.



**Figure 3.4** Routing management functional module





**Figure 3.5** State machine diagram of routing management

### 3.3.1.1 Optimal Forwarding Determined

In order to carry out the optimal forwarding calculation, the routing management calculates three parameters, which are maximum packet velocity; link quality and remaining power (remaining battery) for every one hop neighbours. Eventually, the router management will forward a data packet to the one-hop neighbour that has an optimal forwarding. The optimal forwarding (OF) is computed as follows:

$$OF = \max (\lambda_1 * PRR + \lambda_2 * V_{batt} / V_{mbatt} + \lambda_3 * V / V_m)$$

where  $\lambda_1 + \lambda_2 + \lambda_3 = 1$  (3-1)

where  $V_{mbatt}$  is the maximum battery voltage for sensor nodes and is equal to 3.6 volts [8].  $V_m$  is the maximum velocity of the RF signal that is equal to the speed of light. The determination of PRR, battery voltage ( $V_{batt}$ ) and packet velocity ( $V$ ) is elaborated in the following section. The values of  $\lambda_1$ ,  $\lambda_2$  and  $\lambda_3$  are estimated by exhaustive search using NS-2 simulation as will be explained in detail in chapter 4.

### i) Determination of Packet Velocity

The total delay to one hop neighbour ( $N$ ) from the source ( $S$ ) can be calculated as follows:

$$Delay(S, N) = T_c + T_t + T_p + T_q + T_b + T_s = \frac{Round\_trip\_time}{2} \quad (3-2)$$

Where,  $T_c$  is the time it takes for  $S$  to obtain the wireless channel with carrier sense delay and backoff delay.  $T_t$  is the time to transmit the packet that is determined by channel bandwidth, packet length and the adopted coding scheme.  $T_p$  is the propagation delay that can be determined by the signal propagation speed and the distance between  $S$  and  $N$ . In sensor networks, the distances between sensor nodes are normally very small, and the propagation delay can normally be ignored.  $T_q$  is the processing delay which depends on network data processing algorithms to process the packet before forwarding it to the next hop.  $T_b$  is the queuing delay, which depends on the traffic load. In a heavy traffic case, queuing delay becomes a dominant factor.  $T_s$  is sleep delay which is caused by nodes periodic sleeping. When  $S$  gets a packet to transmit, it must wait until  $N$  wakes up. Equation (3-2) shows that the delay between two pair of nodes varies since the  $T_c$  and  $T_b$  delays differ for all nodes.

It is interesting to note that the routing management is independent of synchronization timing. The non-synchronization is solved by inserting the transmission time in the header of request to route (RTR) packet. When receiving node  $N$  replies to sensor node  $S$ , it inserts the RTR transmission time in its reply. Once  $S$  receives the reply, it subtracts the transmission time from the arrival time to calculate the round trip time. The maximum packet velocity ( $V$ ) between a pair of nodes is calculated as follows;

$$V = \frac{d(S, N)}{Delay(S, N)} \quad (3-3)$$

where  $d(S, N)$  is the one-hop distance between source node  $S$  and destination node  $N$ . In this study, this distance is assumed to be fixed. However, if the sensor node is mobile, the distances can be calculated from the signal strength as shown in [75, 76]. If the velocity is high, the packet has a high probability to arrive before deadline and thus ensure real-time communication.

## ii) Determination of Link Quality

The wireless medium does not guarantee reliable data transfer. The link quality of the wireless medium determines the performance of WSN. In designing RTLD routing protocol, the link quality is considered in order to improve the delivery ratio and energy efficiency [10]. This section will elaborate the mathematical calculation of the link quality. It should be noted that the link quality is measured based on PRR to reflect the diverse link qualities within the transmission range. PRR is approximated as the probability of successfully receiving a packet between two neighbour nodes [34, 46]. If PRR is high that means the link quality is high and vice versa. In this work, the physical layer is based on the IEEE 802.15.4/Zigbee RF transceiver that has a frequency of 2.4 GHz with O-QPSK modulation. It is based on a chip rate  $R_c$  of 2000 kc/s, a bit rate  $R_b$  of 250 kb/s and a codebook of  $M=16$  symbols. The PRR in routing management uses the link layer model derived in [20, 34, 77]. Conversion from SNR to bit noise density  $E_b/N_0$  assumes matched filtering and half-sine pulse shaping as shown bellow:

$$\frac{E_b}{N_0} = \frac{0.625 R_c}{R_b} SNR = \frac{0.625 \times 2000000}{250000} SNR = 5.0 SNR \quad (3-4)$$

Conversion from  $E_b/N_0$  to symbol noise density  $E_s/N_0$  is

$$\frac{E_s}{N_0} = \log_2(M) \frac{E_b}{N_0} = 4 \frac{E_b}{N_0} \quad (3-5)$$

Symbol error rate  $P_s$  is computed in [78] as

$$P_s = \frac{1}{M} \sum_{j=2}^M (-1)^j \binom{M}{j} \exp\left(\frac{E_s}{N_0} \left(\frac{1}{j} - 1\right)\right) \quad (3-6)$$

Finally, conversion from  $P_s$  to bit error rate ( $P_b$ ) is given as

$$P_b = P_s \left( \frac{M/2}{M-1} \right) \quad (3-7)$$

By substitution equations (3-5) and (3-6) in equation (3-7)  $P_b$  becomes

$$P_b = \left[ \frac{1}{M} \sum_{j=2}^M (-1)^j \binom{M}{j} \exp\left(\frac{4E_b}{N_0} \left(\frac{1}{j} - 1\right)\right) \right] \left( \frac{M/2}{M-1} \right) \quad (3-8)$$

The PRR is calculated from  $P_b$  as;

$$PRR = (1 - P_b)^m \quad (3-9)$$

By substitution equation (3-8) into equation (3-9), PRR becomes

$$P_b = \left( 1 - \left[ \frac{1}{M} \sum_{j=2}^M (-1)^j \binom{M}{j} \exp \left( \frac{4E_b}{N_0} \left( \frac{1}{j} - 1 \right) \right) \right] \binom{M/2}{M-1} \right)^m \quad (3-10)$$

Where  $m$  is the frame length in bits and  $M=16$ . Since the average frame length for IEEE 802.15.4 is 22 bytes [20],  $m$  is 176 bits. From equation (3-4) and (3-10), PRR is determined by

$$PRR = \left[ 1 - \left( \frac{8}{15} \right) \left( \frac{1}{16} \right) \sum_{j=2}^{16} (-1)^j \binom{16}{j} \exp \left( 20SNR \left( \frac{1}{j} - 1 \right) \right) \right]^m \quad (3-11)$$

SNR is calculated in [78, 79] as

$$SNR = P_t - PL(d) - S_r \quad (3-12)$$

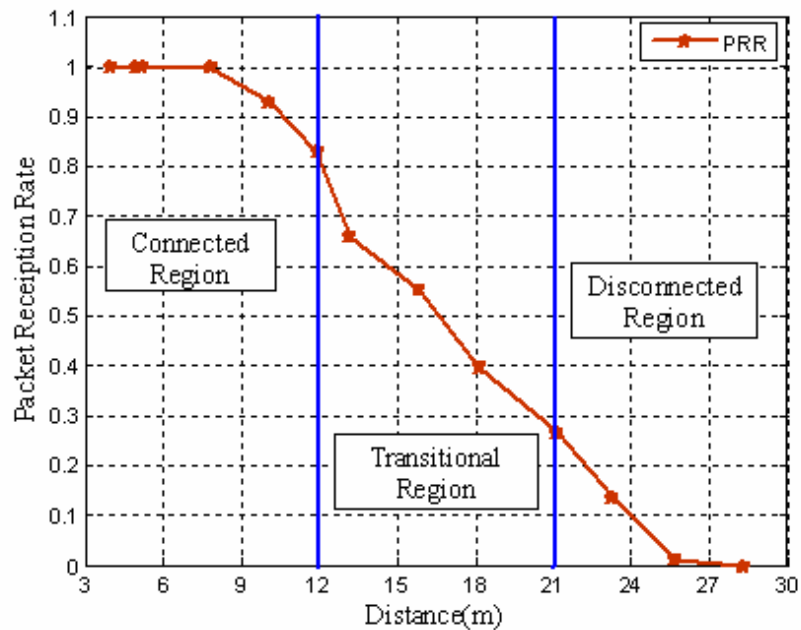
where  $P_t$  is the transmitted power in dBm (maximum is 0 dBm for MICAZ) and  $S_r$  is the receiver's sensitivity in dBm (-95 dBm in MICAZ) [80].  $PL(d)$  is the path loss model which can be calculated as in [78]

$$PL(d) = PL(d_0) + 10\beta \log\left(\frac{d}{d_0}\right) + X_\sigma \quad (3-13)$$

where  $d$  is the transmitter-receiver distance,  $d_0$  is the reference distance,  $\beta$  is the path loss exponent (rate at which signal decays) which depends on the specific propagation environment. For example,  $\beta$  equals to 2 in free space and will have larger value in the presence of obstructions. This work estimates the value of  $\beta$  to be between 2.4 and 2.8 as calculated in [76].  $X_\sigma$  is a zero-mean Gaussian distributed random variable in (dB) with standard deviation  $\sigma$  (shadowing effects in dB).

The PRR for IEEE 802.15.4/Zigbee was simulated in NS-2 simulator and the results are as shown in Figure 3.6. The figure shows the effect of PRR as the distance is increased. Each point in this figure is the average of ten PRR values with the same distance. The PRR reaches to disconnected region when the distance is more than 21 meters because the signal strength is very low. The optimal forwarding choice is generally in the transitional region

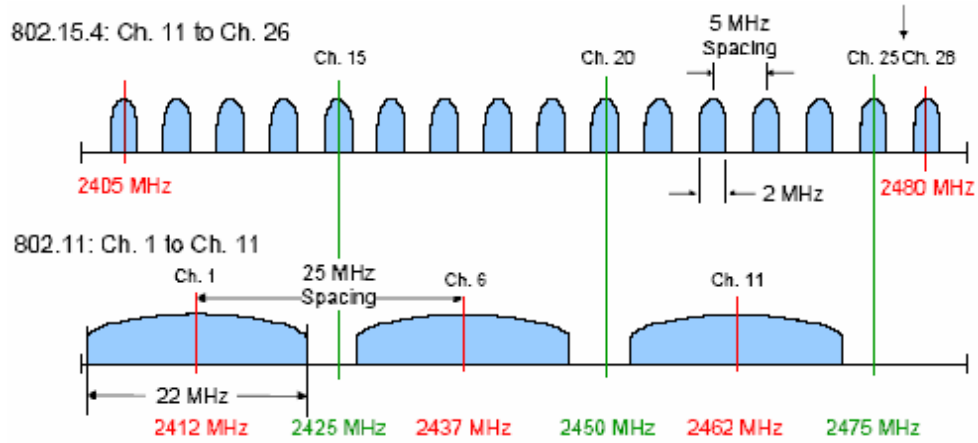
[46]. The main reason is the fact that geographic forwarding scheme attempts to minimize the number of hops by maximizing the geographic distance covered at each hop (as in greedy forwarding). This is likely to incur significant energy expenditure due to retransmission on the unreliable long weak links which wastes up to 80% of communication energy [10]. On the other hand, if the forwarding mechanism attempts to maximize per-hop reliability by forwarding only to close neighbours with good links, it may cover only a small geographic distance at each hop. Also, this will result in greater energy expenditure due to the need for more transmission hops for each packet to reach the destination [46]. Therefore, the proposed forwarding mechanism allows the data packet to be forwarded to sensor nodes in the connected and transitional regions.



**Figure 3.6** PRR vs. Distance

The relationship between the IEEE 802.11b (non-overlapping sets) and the IEEE 802.15.4 channels at the 2.4 GHz is illustrated in Figure 3.7. In order to prevent the interference between the IEEE 802.15.4 and the IEEE 802.11b, IEEE 802.15.4 recommends the use of channels that fall in the guard bands between two adjacent of IEEE 802.11b channels or above these channels. Figure 3.7 shows 2 channels fall in the guard bands between two adjacent IEEE 802.11b channels and 2 channels fall above. If IEEE 802.15.4 network operates on one of these channels, the interference between IEEE 802.15.4 and IEEE 802.11b will be minimized. However,

if there will be more IEEE 802.15.4 networks, these four channels are not enough [23].



**Figure 3.7** IEEE 802.11b and IEEE 802.15.4 Channel Selection [23]

### iii) Determination of Remaining Power

A routing mechanism that takes into account the battery in sensor nodes can assist in load distribution among sensor nodes, which may eventually prolong the WSN lifetime. In order to compute the remaining power in the battery of a sensor node, MICAZ has an accurate internal voltage reference that can be used to measure battery voltage ( $V_{batt}$ ). Since the eight-channel ADC on the microcontroller of MICAZ (ATMega128L) uses the battery voltage as a full scale reference, the ADC full scale voltage value changes as the battery voltage changes. In order to track the battery voltage, the precision voltage reference (band gap reference) is monitored to determine the ADC full-scale (ADC\_FS) voltage span which corresponds to  $V_{batt}$  [27]. The battery voltage is computed as follows:

$$V_{batt} = \frac{V_{ref} * ADC\_FS}{ADC\_Count} \quad (3-14)$$

ADC\_FS equals 1024 while  $V_{ref}$  (internal voltage reference) equals 1.223 volts and ADC\_Count is the ADC measurement data at internal voltage reference.

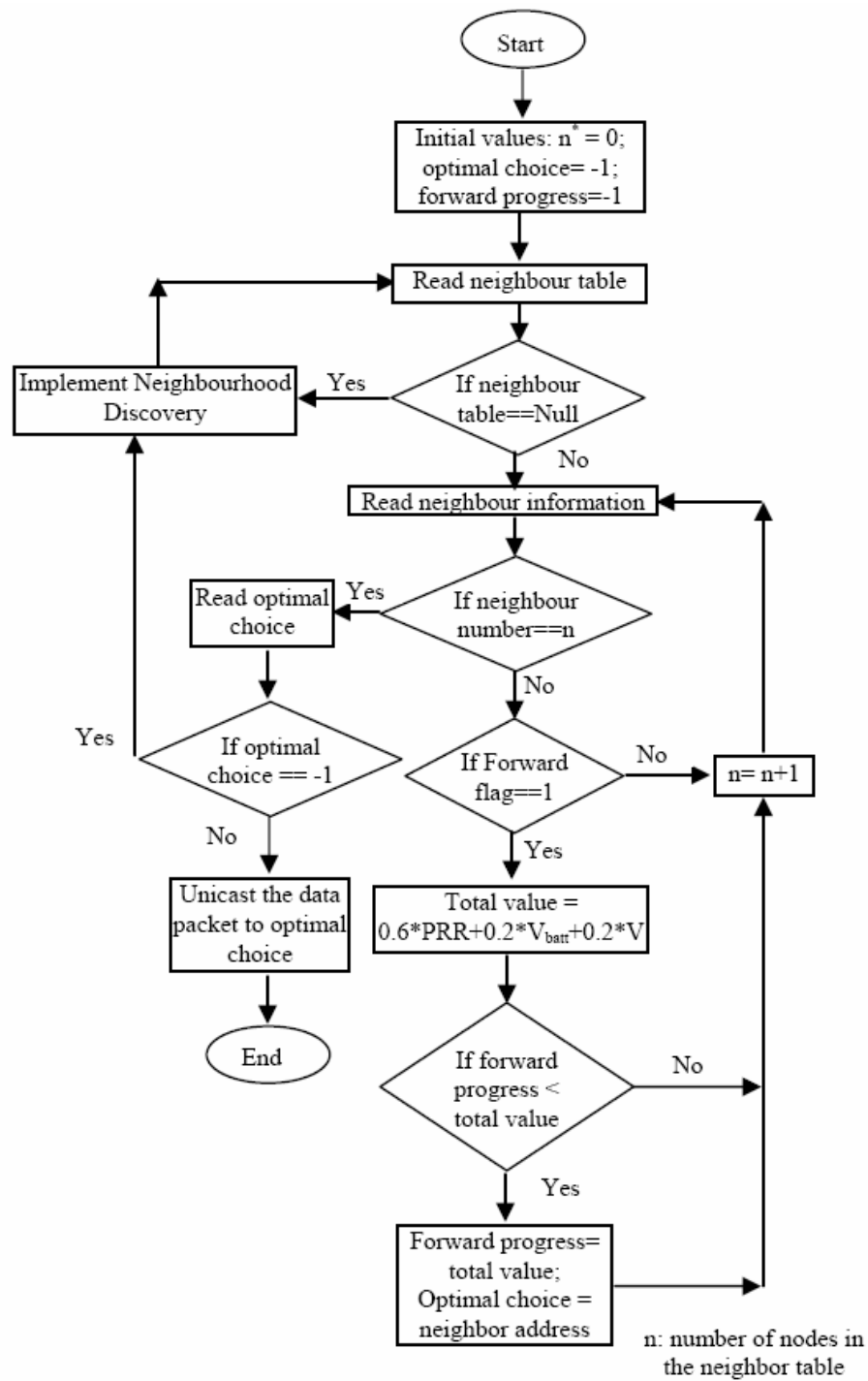
### 3.3.1.2 Forwarding Mechanisms

The routing management proposes two different types of forwarding in RTLD: unicast forwarding and geodirectional-cast forwarding towards the destination based on quadrant.

#### i) Unicast Forwarding Mechanism

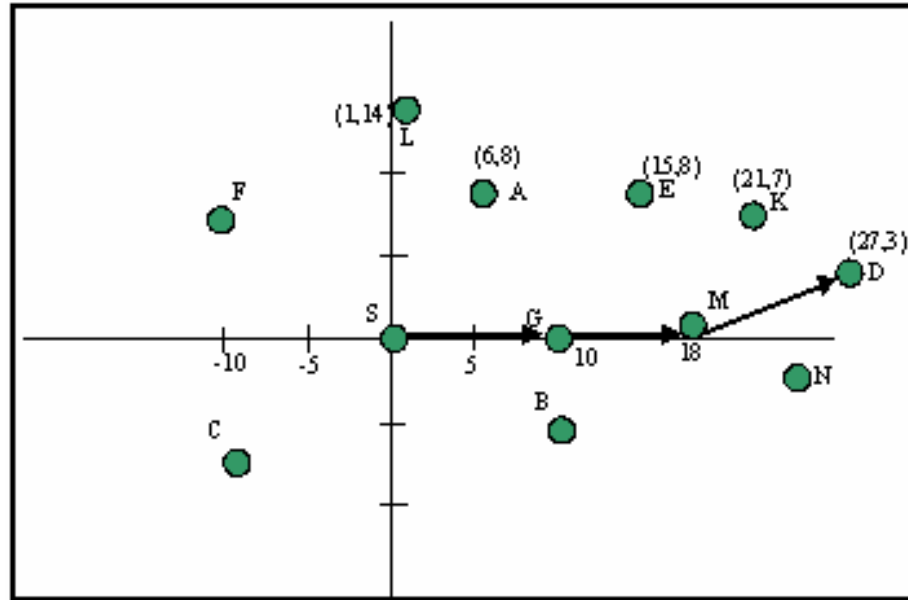
Figure 3.8 shows the flow chart diagram of optimal forwarding algorithm with unicast forwarding. In unicast forwarding, the source node checks for the forward flag of each neighbour in the neighbour table. If the forward flag is 1, the source node will check the optimal forwarding metrics and compute forwarding progress as in equation (3-1). This procedure continues until the optimal forwarding choice is obtained. If there are no nodes in the direction to the destination, the source node will invoke the neighbour discovery. Once the optimal forwarding choice is obtained, the data packet will be unicast to the selected node. The selected forwarding node will then select the next forwarding node if the destination is not one of its neighbours. This procedure continues until the destination is one of the selected node's neighbours. At this instance, the data packet will be unicast directly to the destination.

Figure 3.9 shows an example of unicast forwarding of 12 nodes in a global coordinate system based on quadrant system. In this figure, S checks the forward flag. The forward flag is one if S considers the optimal choice to be in the same quadrant as D and the distance between the optimal choice and D is less than the distance between S and D. The forward flag for nodes B, C, F and N is zero because S does not consider them to be in the first quadrant as D. The forward flag for node L is zero because the distance between L and D is greater than the distance between S and D. On the other hand, the forward flag for nodes A and G is one, therefore S selects the optimal from A and G based on equation (3-16). This procedure continues until the data packet is delivered to D.



**Figure 3.8** Flow chart diagram of optimal forwarding policy in unicast forwarding





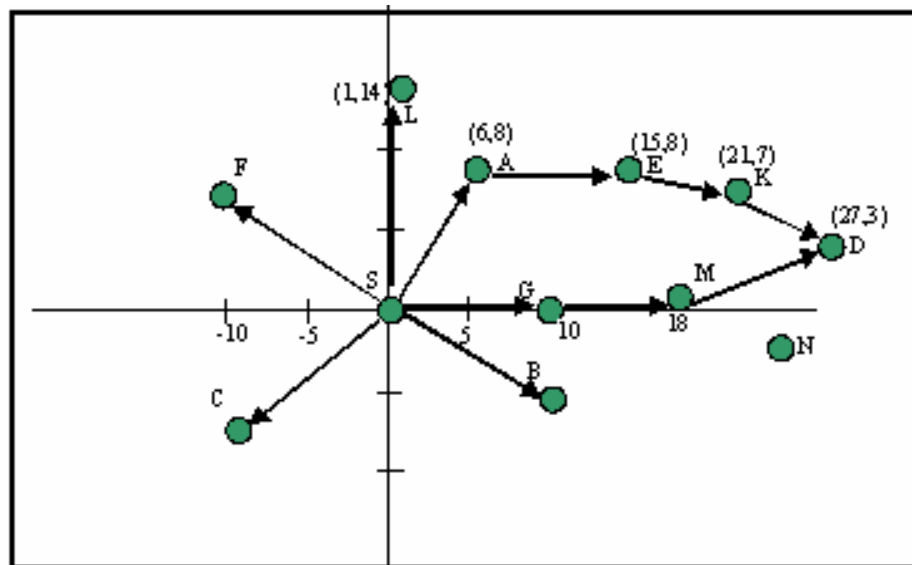
**Figure 3.9** Unicast forwarding based on quadrant

## ii) Geodirectional-cast Forwarding Mechanism

Directional forwarding is defined as forwarding to the next nodes that have the best progress towards the destination. In geodirectional-cast forwarding, if a node wants to forward a data packet to a specific destination in a specific geographical location, it will broadcast the packet in the first hop to all neighbours. Then the selected neighbouring node will use unicast forwarding to forward the packet towards the destination. Therefore, if the neighbouring nodes are in the same quadrant as the destination and if the distance to the destination is less than the distance from source to destination, nodes will forward the packet using unicast forwarding. Otherwise, the packet will be ignored. Since nodes have information of its neighbours, it will not only forward but also select a neighbour that has the optimal forwarding progress towards the destination. If the destination receives multiple copies of the same packet, it will accept the first packet delivered and ignore the others.

The proposed mechanism is a modification of the work done on Q-DIR [81]. In Q-DIR, all forwarding nodes broadcast the packet without knowing the distance. However, in the proposed mechanism, source node only broadcasts the packet to one hop neighbour using transmission power control. This modification of Q-DIR will save power usage, reduce packet flooding and minimize collision.

Figure 3.10 shows an example of geodirectional-cast forwarding of 12 nodes in a global coordinate system based on quadrant system. In this figure, S broadcasts the data packet to its neighbours. S considers D to be in the first quadrant. Nodes B, C, F, and N ignore the forwarding request because they are not in the same quadrant as D. Node L also ignores the forwarding request because its distance to D is greater than the distance between S and D. On the other hand, nodes A and G are in the first quadrant as D and the distance between them and D is less than the distance between S and D. Hence A and G will participate and forward the data packet to E and M respectively. It is interesting to note that nodes A and G will use unicast forwarding to forward the data packet to E and M rather than broadcast.



**Figure 3.10** Geodirectional-cast forwarding based on Quadrant

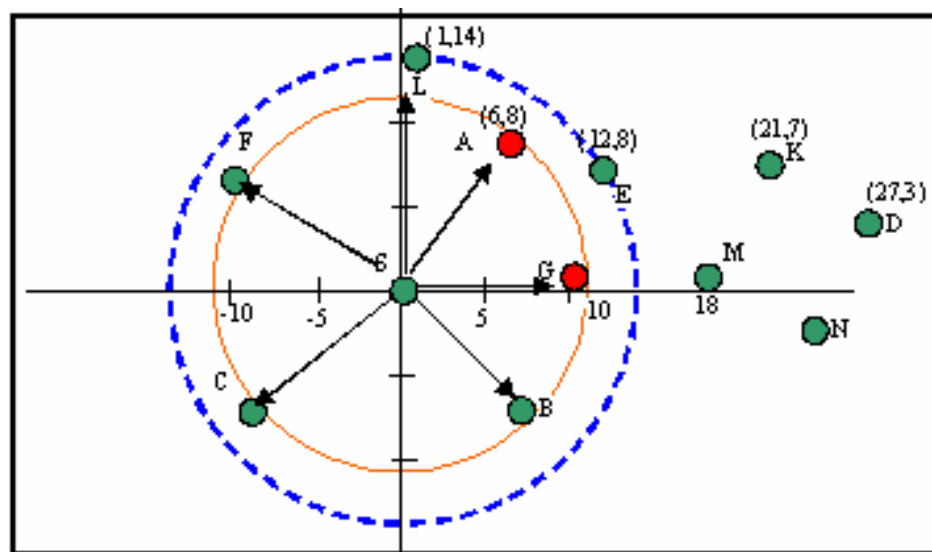
The forwarding policy may fail to find a forwarding node when there is no neighbour node currently in the direction of destination. The routing management recovers from these failures by using routing problem handler as described in the following section.

### 3.3.1.3 Routing Problem handler

A known problem with geographic forwarding is the fact that it may fail to find a route in the presence of network holes even with neighbour discovery. Such holes may appear due to voids in node deployment or subsequent node failures over the lifetime of the network. Routing management in RTLD solves this issue by introducing routing problem

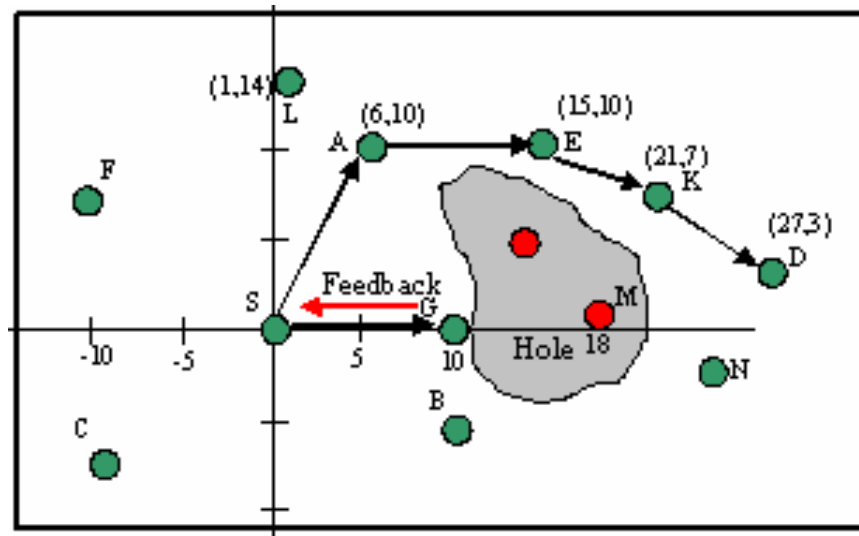
handler which has two recovery methods; fast recovery using power adaptation and slow recovery using feedback control packet.

The fast recovery is applied if the diameter of the hole is smaller than the transmission range at the maximum power. The routing problem handler then will inform neighbour discovery to identify a maximum transmission power that is sufficient to transmit the packet across the hole as shown in Figure 3.11. In this figure, if nodes A and G are failures due to some problems such as diminishing energy of sensor node or due to unreliable connection, S will use maximum transmission power (0 dBm in IEEE 802.15.4) to send RTR. Therefore, node E will receive RTR from S and will reply using maximum transmission power. Hence, node E is OF node. If the fast recovery can not avoid routing hole problem, the slow recovery is applied.



**Figure 3.11** Fast recovery of routing hole problem

In the slow recovery, candidate OF node will send feedback packet to its parent. The feedback packet will inform the sensor node parent to stop sending data packet toward OF sensor node. When the parent received feedback control packet, it will calculate OF again for all candidates as depicted in Figure 3.12. In this case, node G has a hole routing problem. Therefore, node G sends feedback to node S that will select node A as OF.



**Figure 3.12** Feedback mechanism in routing problem handler

### 3.3.2 Location Management

The proposed location management determines localized information of sensor nodes. It assumes that all sensor nodes are in a fixed position. It also assumes that the sink node is at the origin (0,0) and at least two of its neighbours are location aware. The location management is used to determine the sensor node location in a grid of WSNs. It is assumed that each node has a location aware mechanism such as in [75, 76] to obtain its location in the WSNs area. The location mechanism uses at least three signal strength measurements extracted from RTR packets broadcasted by pre-determined nodes at various intervals. Each pre-determined node broadcasts RTR packet and inserts its location in the packet header. The distance of the unknown node from the pre-determined nodes is determined from the signal strength received based on a propagation path loss model of the environment. If the distance and location of these pre-determined nodes are known, unknown nodes can triangulate their coordinates as explained in [75, 76]. The developed location management will not require additional hardware such as GPS since it uses the existing wireless communication hardware. The location determination of an unknown node is explained in the next section.

### 3.3.2.1 Location Determination

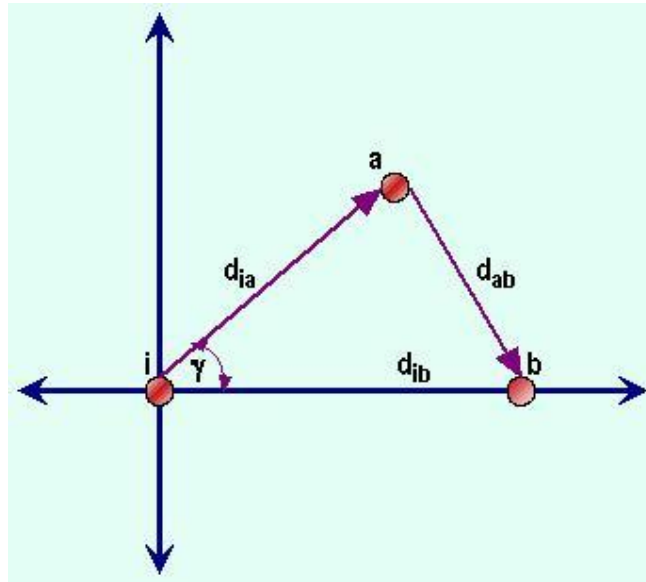
Let  $W$  be the set of all the nodes in the network.  $\forall i \in W$ ,  $K_i$  is defined as the set of one-hop neighbours of  $i$  as shown in Figure 3.13. Likewise,  $\forall i \in W$ ,  $D_i$  is defined as a set of distances between  $i$  and each node  $j \in K_i$ . The neighbours can be detected by using RTR which are sending from the sensor nodes. After an absence of a certain number of successive beacons, it will be concluded that the node is no longer a neighbour. The position of the sink (node  $i$ ) is in the centre of Network Coordinate System (NCS) that is  $(0, 0)$ . Two one-hop neighbouring nodes chosen are  $a$ , and  $b$ , where  $a, b \in K_i$ . These two nodes will assist in adjustments of NCS and then from these nodes and the sink, the position of any node one-hop away from the sink can be determined by triangulation method. The distance between  $a$  and  $b$  is denoted as  $d_{ab}$ . Note that  $d_{ab}$  is larger than zero and nodes  $i$ ,  $a$ , and  $b$  must not lie on the same line. The NCS defines such that node  $b$  must lie on the positive x-axis of the coordinate system and node  $a$  has positive  $a_y$  and  $a_x$  components as shown in Figure 3.13. Thus, the coordinate systems of nodes  $i$ ,  $a$ ,  $b$  are:

$$\begin{aligned} i_x &= 0, i_y = 0 \\ b_x &= d_{ib}, b_y = 0 \\ a_x &= d_{ia} \cos \gamma, a_y = d_{ia} \sin \gamma \end{aligned} \quad (3-15)$$

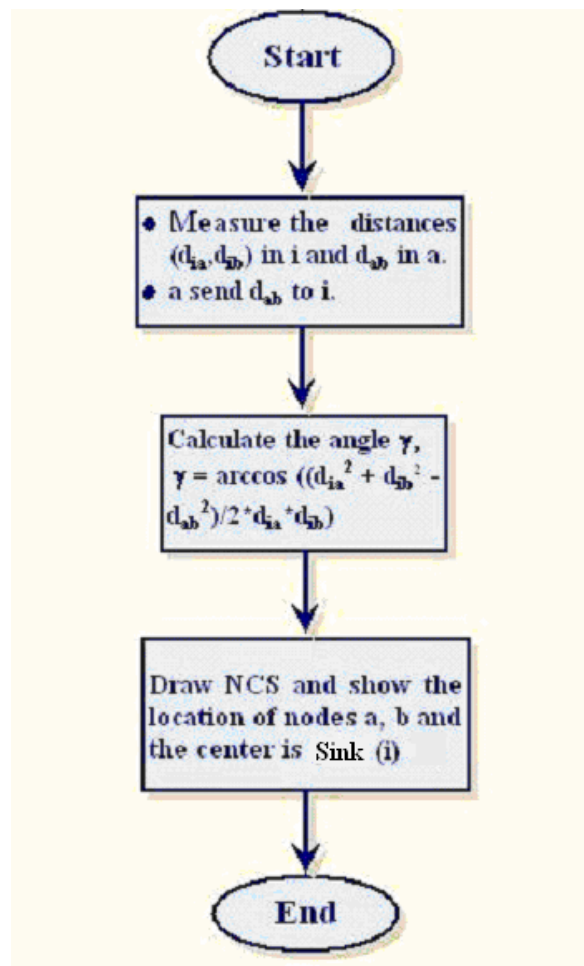
Where  $\gamma$  is the angle  $\sphericalangle(bia)$  and is obtained by using a cosines rule for triangles as:

$$\gamma = \arccos((d_{ia}^2 + d_{ib}^2 - d_{ab}^2) / 2d_{ia}d_{ib}) \quad (3-16)$$

The flow chart of NCS is illustrated in Figure 3.14 which shows that nodes  $a$  and  $b$  are initial nodes. The positions of  $a$  and  $b$  are known and the angle  $\sphericalangle bia$  can be determined.



**Figure 3.13** Network Coordinate System



**Figure 3.14** Flow chart of drawing NCS

Location of node  $c$  such that  $c \in K_i$  and  $c \neq a$  or  $b$  as shown in Figure 3.15 can be determined by collecting signal strength and calculating the distances  $d_{ac}$ ,  $d_{bc}$  and  $d_{ic}$ . Therefore,  $c_x$  is obtained from the following equation:

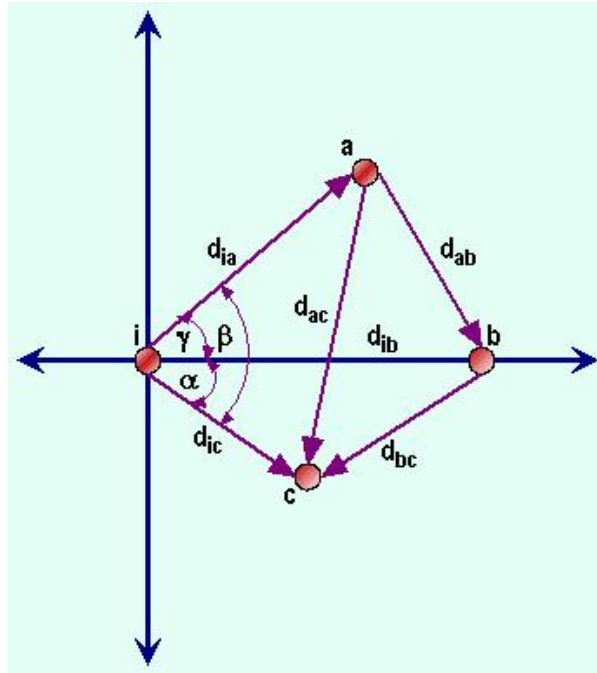
$$c_x = d_{ic} \cos \alpha \quad (3-17)$$

Equation (3-17) will always give a positive angle for  $\gamma$ . To determine if  $c_y$  is positive or negative, Equation (3-18) will be used:

$$c_y = \begin{cases} d_{ic} \sin \alpha & \text{if } (\beta - \alpha + \gamma) = 0 \\ -d_{ic} \sin \alpha & \text{if } (\beta - \alpha + \gamma) \neq 0 \end{cases} \quad (3-18)$$

where  $\alpha$  is the angle  $\angle bic$ , and  $\beta$  is the angle  $\angle cia$ . The purpose of this exercise is to find out on which side of the x-axis node  $c$  is located. In practice,  $\beta - \alpha + \gamma$  will never be exactly equal to zero and will always give a negative value of  $c_y$  due to the errors in distance measurements. Hence, equation (3-18) was modified to become:

$$c_y = \begin{cases} d_{ic} \sin \alpha & \text{if } (\beta - \alpha + \gamma) \leq \text{error} \\ -d_{ic} \sin \alpha & \text{if } (\beta - \alpha + \gamma) > \text{error} \end{cases} \quad (3-19)$$



**Figure 3.15** Position detection of node  $c$  according to NCS

We obtain the values of  $\alpha$  and  $\beta$  by using the cosine rule as follows:

$$\alpha = \arccos((d_{ic}^2 + d_{ib}^2 - d_{bc}^2) / 2d_{ic}d_{ib}) \quad (3-20)$$

$$\beta = \arccos((d_{ia}^2 + d_{ic}^2 - d_{ac}^2) / 2d_{ia}d_{ic})$$

The angles  $\alpha$ ,  $\beta$  and  $\gamma$  are placed within the triangles  $(bic)$ ,  $(cia)$  and  $(bia)$ , respectively and thus we observe just their absolute values but not their directions.

The position of any node such as  $k \in K_i$ ,  $k \neq b, a$ , which is not neighbour of nodes  $b$  and  $a$ , can be computed by using the positions of the node  $i$  and at least two other nodes for which the positions are known and distance from node  $k$  to these nodes is known. Figure 3.16 shows the flow chart of one hop detection in node  $c$ . In this figure, the input to node  $c$  is the signal strength of all neighbours  $a$ ,  $b$  and  $i$ . The distances  $d_{ac}$ ,  $d_{bc}$  and  $d_{ic}$  were determined using location management algorithm. The angles  $\alpha$ ,  $\beta$  and  $\gamma$  were calculated and equation (3-19) were used to compute position which considers the error in distance measurements.

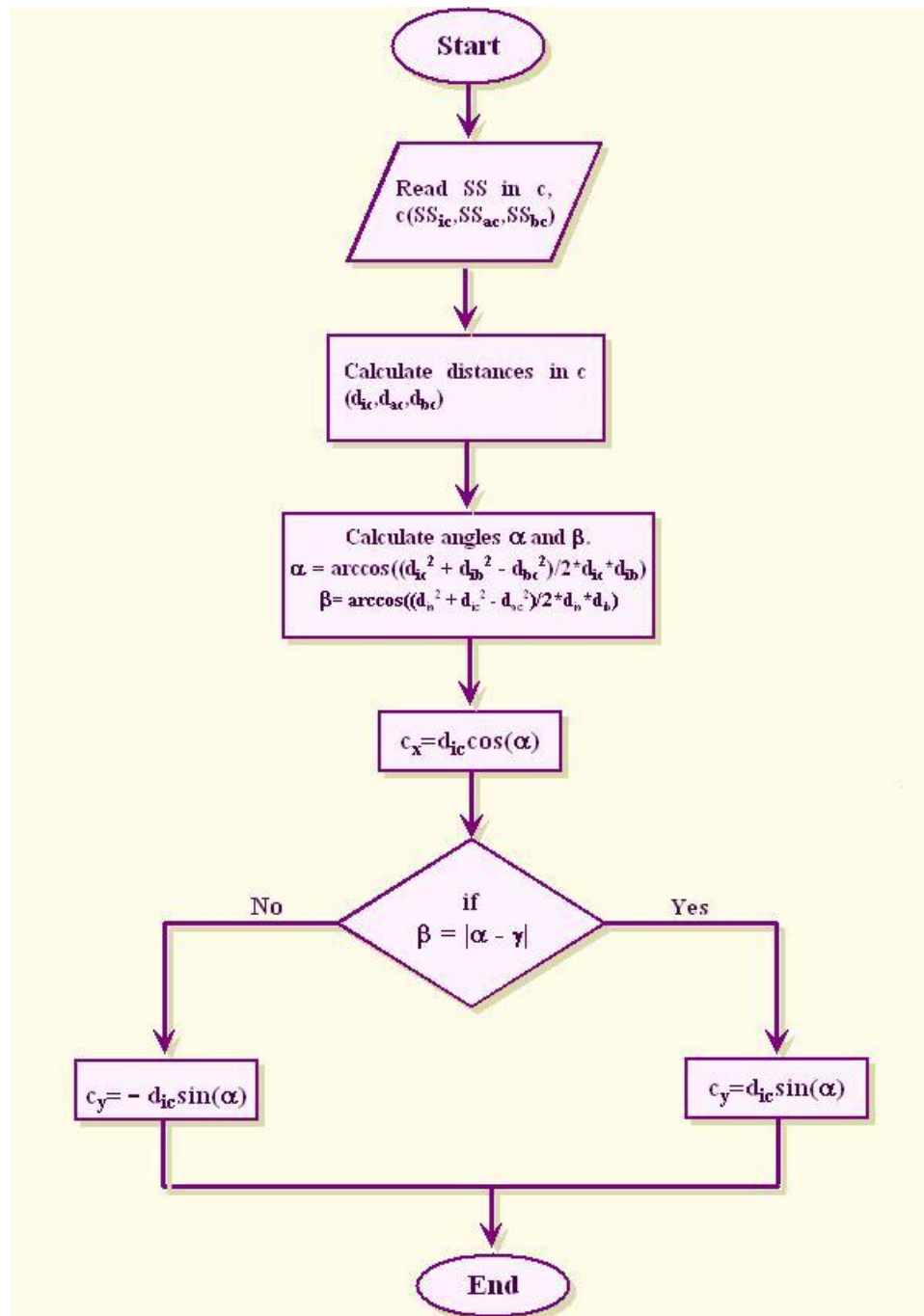
It is interesting to note that equations (3-17) and (3-19) are only valid for the neighbours of the sink. In order to calculate the location of sensor node from any position, equations (3-24) and (3-25) are applied as follows:

$$c_x = \bar{i}_x + d_{ic} \cos \alpha \quad (3-21)$$

$$c_y = \begin{cases} \bar{i}_y + d_{ic} \sin \alpha & \text{if } (\beta - \alpha + \gamma) \leq \text{error} \\ \bar{i}_y - d_{ic} \sin \alpha & \text{if } (\beta - \alpha + \gamma) > \text{error} \end{cases} \quad (3-22)$$

Where  $\bar{i}$  is the new NCS.





**Figure 3.16** Flow chart of one-hop neighbour position determination

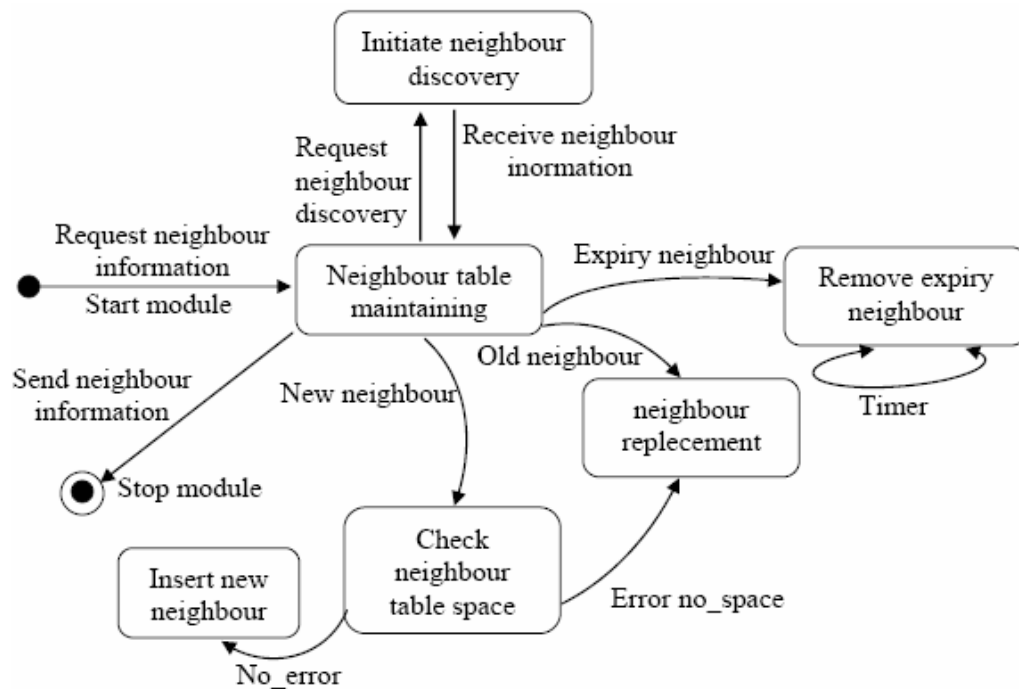
### 3.3.3 Neighbourhood Management

The design goal of the neighbourhood manager is to discover a subset of forwarding candidate nodes and to maintain a neighbour table of the forwarding candidate nodes. Due to limited memory and large number of neighbours, the neighbour table is limited to a small set of forwarding candidates that are most useful in meeting the one-hop end-to-end delay with the optimal PRR and remaining power. The neighbour table format contains node ID, remaining power, one-hop end-to-end delay, PRR, forward flag, location information and expiry time as shown in Figure 3.17. The proposed system manages up to a maximum store of 16 sensor nodes information in the neighbour table.

Figure 3.18 shows the state machine diagram of the functional process of the neighbourhood management. In this figure, if the neighbourhood management receives new neighbour from route management, it will check the neighbour table. If the node ID already exists, the neighbourhood procedure will update the information of the existing neighbour such as PRR, remaining power, end-to-end delay and expiry time. If the ID of new neighbour does not exist and the number of nodes in the neighbour table is less than 16 nodes, the neighbourhood management will add the new neighbour at the end of the neighbour table. Finally, if the neighbour table is full, the neighbourhood management will compare the forwarding progress metrics of all nodes in neighbour table with the new node. If the new node has a higher progress than any node in the neighbour table, the neighbourhood management will replace the lower forwarding progress node. Otherwise, the neighbourhood management will drop the new neighbour information.

2 byte	2 byte	2 byte	2 byte	2 byte	1 bit	2 byte	2 byte
<i>Next hop</i>	<i>PRR</i>	<i>Expiry</i>	<i>Batt rem</i>	<i>EtE delay</i>	<i>Flag</i>	<i>X coordinate</i>	<i>Y coordinate</i>

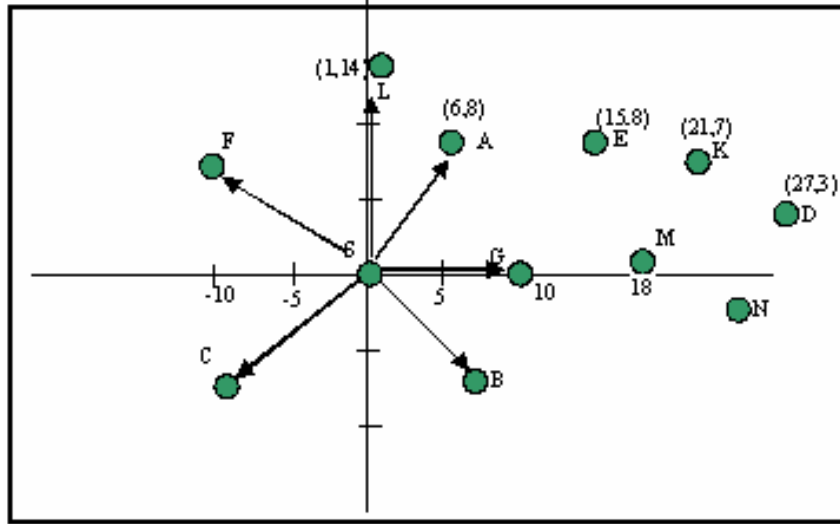
**Figure 3.17** Neighbour table format



**Figure 3.18** State machine diagram of neighbourhood management

### 3.3.3.1 Neighbour Discovery

The neighbour discovery procedure is executed in the initialization stage. The goal of the neighbour discovery is to identify a node that satisfies the forwarding condition. The neighbour discovery mechanism introduces small communication overhead. This is necessary to minimize the time it takes to discover a satisfactory neighbour. The source node invokes the neighbour discovery by broadcasting RTR packet as shown in Figure 3.19. Some neighbouring nodes will receive the RTR and send a reply. Upon receiving the replies, the neighbourhood management records the new neighbour in its neighbour table. Initially, the neighbour discovery will broadcast the RTR at the default power level. However, if the source node does not receive a reply from any node, the routing problem handler will be invoked.



**Figure 3.19** Neighbour discovery

Since the RTR is broadcasted, a large number of nodes may reply. This causes a high network contention. The common solution for this is to let the replying nodes pick a randomized delay before transmitting. A node withdraws from replying if it hears replies from other nodes. The probability of collision due to randomize delay of replies is analyzed mathematically. The aim is to formulate a probability that in a group of  $k$  nodes, at least two of them reply at the same time. The reply time can be modeled as an integer random variable, with uniform distribution between 1 and  $n$  where  $n$  is the time window of replies [82]. Then, the number of ways that we can choose  $k$  values out of  $n$  without duplication would be  $N_k$  where;

$$N_k = n.(n-1).....(n-k+1) \quad (3-23)$$

On the other hand, the number of possibilities for choosing  $k$  elements out of  $n$ , without the restriction of not having any duplicates is  $n^k$ . Thus, the probability of no collision among  $k$  replies out of  $n$  is  $P_n$  where

$$P_n = \frac{n!}{(n-k)! \cdot n^k} \quad (3-24)$$

Performing simple computations in equation (3-24), we obtain:

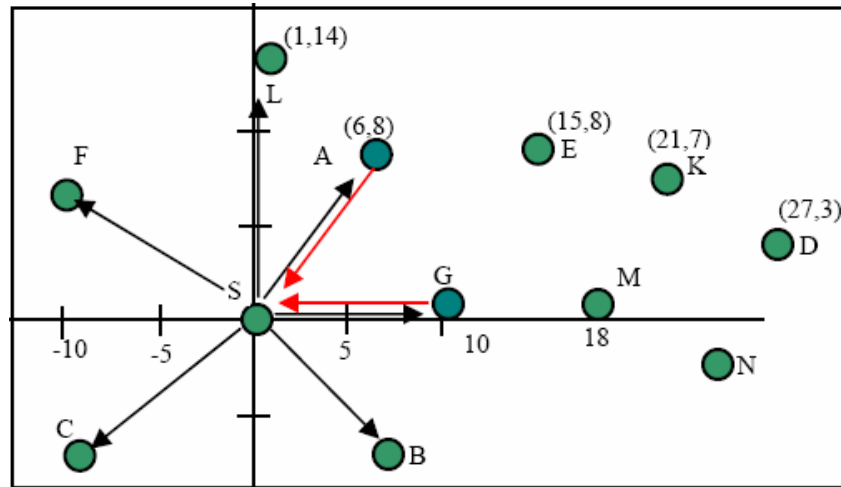
$$P_n = 1.(1-\frac{1}{n}).(1-\frac{2}{n}).....(1-\frac{k-1}{n}) \quad (3-25)$$

Therefore, the resulting expression for the probability of the collision of one or more replies is:

$$P_c = 1 - P_n = 1 - \frac{n!}{(n-k)! \cdot n^k} \quad (3-26)$$

This simple scheme clearly shows that if the time window  $n$  is small, the probability of collision is high and if the time window is large, the probability of collision is small. However, a large time window prolongs the time needed to find a viable neighbour that meets the real-time forwarding requirement. In addition, the mathematical equation shows that less number of replies  $k$  reduces the probability of collision.

The proposed neighbour discovery reduces the number of replies by restricting the set of replying nodes to include only those that may help in meeting the forwarding requirement. This means that a node replies only if it makes progress toward destination as shown in Figure 3.20. In this case, nodes A and G will reply while nodes B, C, F and L will refrain from replying, in accordance to the forwarding mechanism in routing management. Thus, the number of reply nodes are reduced and the probability of collision due to replies is minimized.

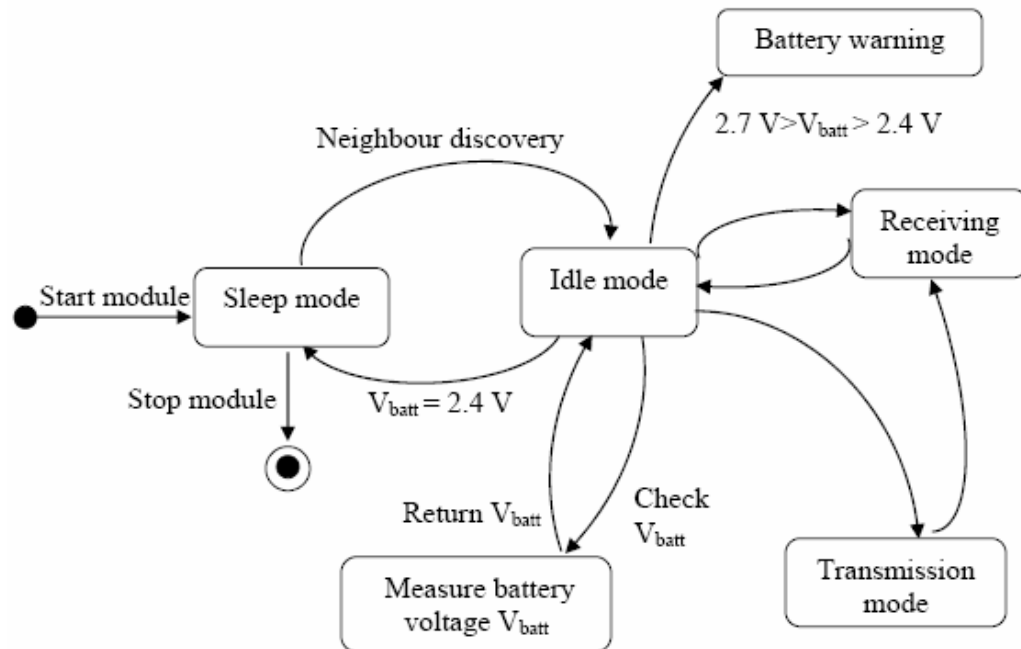


**Figure 3.20** Replies of neighbour discovery

### 3.3.4 Power Management

The main function of power management is to adjust the state of the transceiver and to select the level of transmission power of the sensor node. It focuses on minimizing the energy consumed in each sensor node between the source and the destination to increase node lifetime. To minimize the energy consumed, power management minimizes the energy wasted by idle listening and control packet overhead.

The power management has been designed to balance real-time performance with power efficiency. The transceiver component in MICAZ consumes the most energy compared to other relevant components of the MICAZ. The radio transceiver has four different states; down or sleep state, idle state, transmit state and receive state [80, 83]. The proposed power management is developed based on the state machine diagram shown in Figure 3.21. In this figure, the sensor node changes its state from sleep state to idle state then to transmit state when it implements neighbour discovery. If the transmission of RTR packet is successful, the transceiver state will change from transmit state to receive state. Otherwise, the transceiver stays in the transmit state until it finishes all trails of transmission.



**Figure 3.21** State machine diagram of power management

Figure 3.21 also shows that if the battery voltage is less than 2.4 V, the sensor node will not be able to send or receive packets and hence it will change its state to idle state and then to sleep state [27].

### 3.4 Built-in Security in RTLD

RTLD is a routing protocol that takes advantage of location based routing, multi-path forwarding and random selection of next hop. The random selection of next hop in RTLD provides some measure of security in WSN. Since the random selection of next hop depends on PRR, packet velocity and remaining power, which are totally dependent on the physical parameters. These parameters can not be changed by other sensor node and thus ensures probabilistic selection chance of next hop node.

RTLD constructs the routing topology on demand using only localized interactions and information. Because traffic is naturally routed towards the physical location of a sink, it is difficult to attract it elsewhere to create a sinkhole attack. A wormhole is most effective when used to create sinkholes or artificial links that attract traffic. Artificial links are easily detected in location based routing protocols because the neighboring nodes will notice the distance between them is well beyond normal radio range [14, 99]. Probabilistic selection in RTLD of a next hop from several acceptable neighbours can assist to overcome the problem of wormhole, sinkhole, and Sybil attacks. Hence, RTLD can be relatively secure against wormhole, sinkhole, and Sybil attacks. However, the main remaining problem is that location information advertised from neighboring nodes must be trusted. A compromised node advertising its location on a line between the targeted node and a sink will guarantee it is the destination for all forwarded packets from that node.

Even though RTLD is resistant to sinkholes, wormholes, and the Sybil attack, a compromised node has a significant probability of including itself on a data flow to launch a selective forwarding attack if it is strategically located near the source or a sink. A compromised node can also include itself on a data flow by appearing to be

the only reasonable node to forward packets to the destination in the presence of routing hole problem. Multi-path forwarding in RTLD can be used to counter these types of selective forwarding attacks. Messages routed over  $n$  paths whose nodes are completely disjoint are completely protected against selective forwarding attacks involving at most  $n$  compromised nodes and still offer some probabilistic protection whenever  $n$  nodes are compromised. In addition, RTLD allows nodes to dynamically choose a packet's next hop probabilistically from a set of possible candidates which can further reduce the chances of an adversary gaining complete control of a data flow.

Major classes of attacks that are not countered by RTLD are selective forwarding and HELLO flood attacks. Defense mechanisms that are more sophisticated are needed to provide reasonable protection against selective forwarding and HELLO flood attacks. We focus on countermeasures against these attacks by enhancing security measure in RTLD as explained in the following section.

### **3.5 Network Model and Performance Parameters**

The network model for RTLD has been developed based on Table 3.1. The network model used in this research conforms to IEEE 802.15.4 MAC and physical layers. Many-to-one traffic pattern is used which is common in WSN applications. This traffic is typical between multiple source nodes and a base station. In all simulations, each node updates its neighbour table every 180s. The neighbourhood management of the RTLD protocol is designed to maintain those nodes that have good progress towards the destination.

In the simulation work, 121 nodes are distributed in a 100m x 100m region as shown in Figure 3.26. Nodes numbered as 120, 110, 100 and 90 are the source nodes and node 0 is the base station node (sink). To increase the hop count between sources and the sink, we select the source nodes from the leftmost grid of the topology and the sink in the middle of the grid. We assume the traffic used is constant bit rate (CBR) with User Datagram Protocol (UDP). Thus, there is no retransmission for the



data packet. In this thesis, the network model uses the model mentioned above unless mentioned otherwise.

**Table 3.1** Network Parameters.

Propagation Model	Shadowing
path loss exponent	2.45
shadowing deviation (dB)	4.0
reference distance (m)	1.0
Parameter	IEEE 802.15.4
phyType	Phy/WirelessPhy/802_15_4
macType	Mac/802_15_4
Operation mode	Non Beacon (unslotted)
Ack	Yes
CSTresh_	1.10765e-11
RXThresh_	1.10765e-11
freq_	2.4e+9
Initial Energy	3.3 Joule
Power transmission	1 mW
Transport layer	UDP
Traffic	CBR

The attributes of RTLD packet header format are declared in Figure 3.27. In this figure, the data packet is used to transfer the sensory data from the sources to the sink and the control packet is used to exchange one hop information between sensor nodes. The *sourceaddr* and *seqno* fields are used to fill the source address and the sequence number of transmitter. The *originaddr* and *originseqno* fields are used to fill the original source address and sequence number of the packet. The *hopcount*, *batt\_rem* and *deadline* fields are used to store the number of hops between the source and the destination, the remaining power of transmitter and packet deadline respectively. In addition, the *timestamp* field is used to store the time of packet transmission in order to calculate one hop end-to-end delay. It is interesting to note that the *next\_hop* field is not included in the packet format because it is already defined in the IP header format.

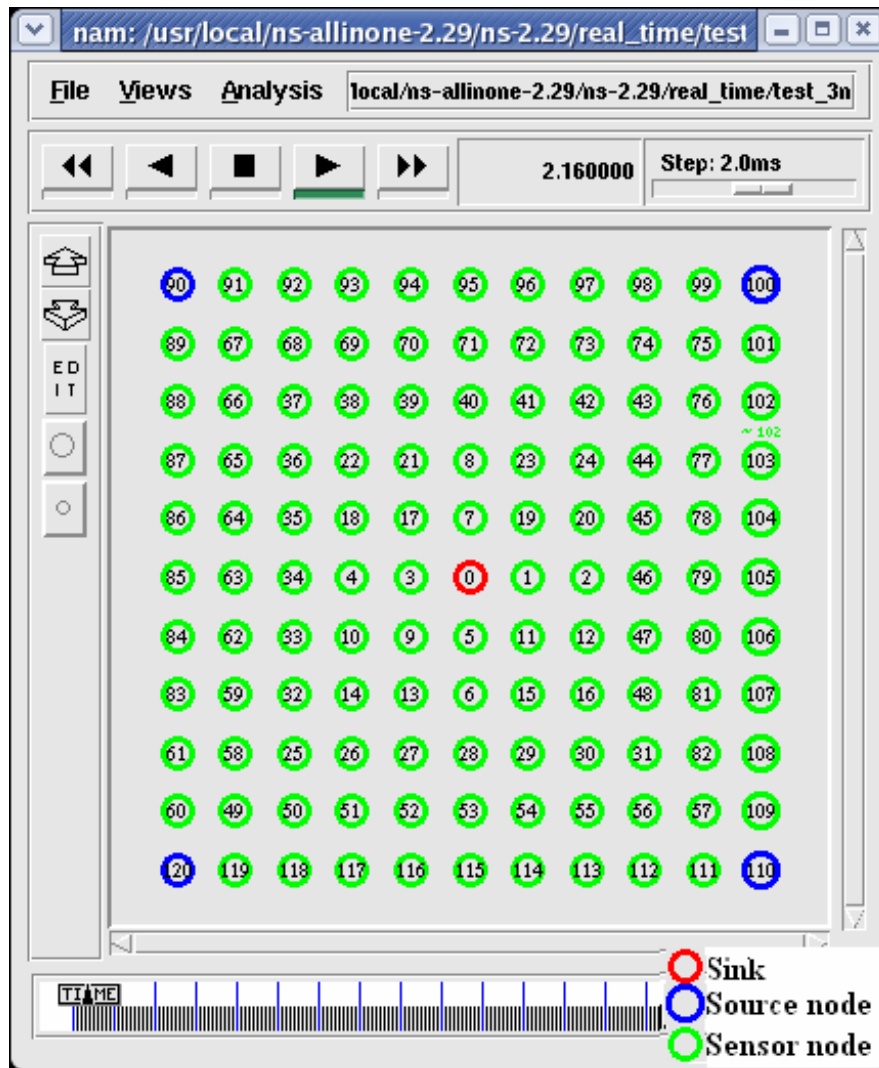


Figure 3.22 Network simulation grid

2 byte	2 byte	2 byte	2 byte	2 byte	2 byte	2 byte	15 byte
<i>sourceaddr</i>	<i>originaddr</i>	<i>seqno</i>	<i>originseqno</i>	<i>hopcount</i>	<i>batt_rem</i>	<i>deadline</i>	<i>data</i>

(a)

2 byte	2 byte	2 byte	2 byte	2 byte	2 byte
<i>sourceaddr</i>	<i>seqno</i>	<i>batt_rem</i>	<i>timestamp</i>	<i>X coordinate</i>	<i>Y coordinate</i>

(b)

Figure 3.23 RTLD packet header format, a) data packet and b) control packet

The performance of RTLD has been evaluated using equations (3-30), (3-31) and (3-32). Packet delivery ratio, normalized control packet overhead and normalized energy consumption are the metrics used to analyze the performance of RTLD. All metrics are defined with respect to the network layer. Packet delivery ratio is the ratio of packets received at the destination to the total number packets sent at the source in network layer. Normalized control packet overhead counts the number of control packets sent in the network for each data packet delivered while normalized energy consumption is the energy consumed in each sensor node for each packet delivered. The packet delivery ratio (*PDR*) is defined as.

$$PDR = \frac{PR}{PS} \quad (3-29)$$

Where *PR* denotes the packets received at the destination and *PS* denotes to the total number of data packets sent from the source. The normalized control packet overhead (*NCPO*) is defined as

$$NCPO = \frac{TCPS}{PR} \quad (3-30)$$

Where *TCPS* denotes the total number of control packets sent in the network and for each data packet received. The normalized energy consumption (*NEC*) is defined as

$$NEC = \frac{TEC}{PR} \quad (3-31)$$

Where *TEC* denotes the total energy consumed for every packet received in each sensor node during the simulation task.

### 3.6 Summary

In this chapter, the overall system design of RTLD based on cross layer design concept has been explained. The proposed RTLD routing is expected to grant end-to-end real-time communication within less than 250 ms while ensuring security against attacks such as wormhole, sinkhole Sybil, selective forwarding and HELLO flood attacks. RTLD routing protocol consists of several functions that include location management, power management, neighbourhood management, routing

management, and security management. The location management in each sensor node calculates its location based on the distance to three pre-determined neighbour nodes. The power management determines the transceiver state and the power level of transmission in the sensor node. The neighbourhood management discovers a subset of forwarding candidate nodes and maintains a neighbour table of the forwarding candidate nodes. The routing management computes the OF node based on neighbour table information. It provides forwarding decision and investigation of routing problem handler.

RTLD possesses built-in security due to random selection of OF node. The RTLD is meant to realize real-time routing feature within deadline. In addition, RTLD ensures that the load is well distributed and thus warrant longer lifetime of sensor node and WSN. In the following chapter, the simulation study of RTLD is carried out and the performance of the proposed RTLD is evaluated.

## **CHAPTER 4**

### **SIMULATION OF RTLD ROUTING PROTOCOL**

#### **4.1 Introduction**

The proposed RTLD routing has been developed and studied through a simulation process using NS-2, a discrete event simulator. The performance of RTLD on a network model has been analysed and compared with several existing WSN real-time routing protocols for traffic load. Since the WSN model studied is based on WPAN, the proposed simulation network model is tailored to match the characteristics of the MICAz mote from Crossbow [52]. In the simulation, IEEE 802.15.4 MAC and physical layer protocols were adapted and embedded in the WSN model to function similar to the MICAz motes.

This chapter elaborates the simulation development of the RTLD routing algorithm on WSN. The object oriented programming based on C/C++ and OTcl was used to create WSN model utilizing RTLD routing protocol. A new packet header for data and control packets has been developed for the network layer task in order to implement the variance functions of the proposed routing system. The analysis of the performance of RTLD was carried out and comparisons were made with some of the existing real-time routing protocols defined as baseline protocols.

## 4.2 Simulation Tools

In the simulation study, NS-2 simulator is used to develop RTLD functional modules. NS-2 is a discrete event simulator targeted at networking research. It provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks [97, 102, 103]. NS-2 uses two languages because simulator has two different kinds of things it needs to do. On one hand, detailed simulations of protocols require a systems programming language that can efficiently manipulate bytes, packet headers, and implement algorithms that run over large data sets. For these tasks run-time speed is important and turn-around time (run simulation, find bug, fix bug, recompile, re-run) is less important. On the other hand, a large part of network research involves slightly varying parameters or configurations, or quickly exploring a number of scenarios. In these cases, iteration time (change the model and re-run) is more important. Since configuration runs once (at the beginning of the simulation), run-time of this part of the task is less important. NS-2 meets both of these needs with two languages, C++ and OTcl [97, 102, 103]. C++ is fast to run but slower to change, making it suitable for detailed protocol implementation. OTcl runs much slower but can be changed very quickly (and interactively), making it ideal for simulation configuration [86]. The next section explained the simulation of IEEE 802.15.4 in NS-2.

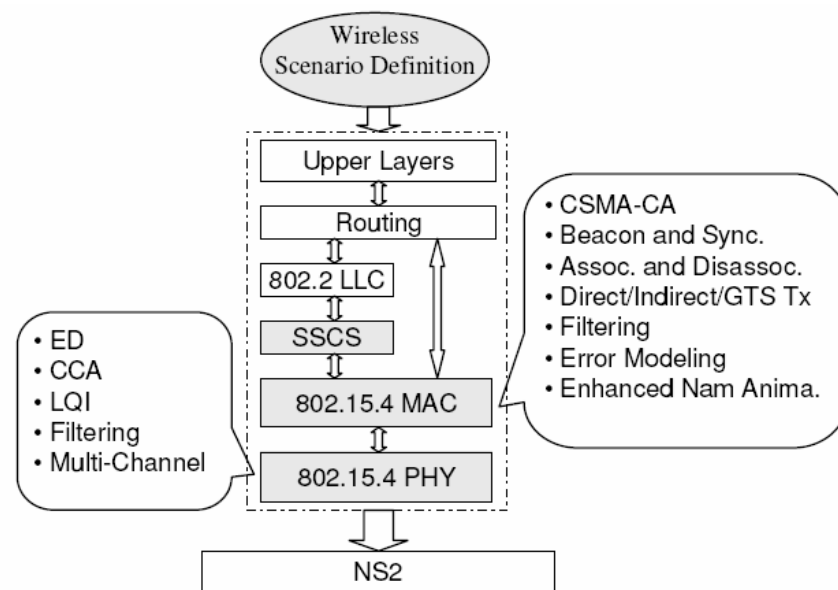
### 4.2.1 IEEE 802.15.4 in NS-2

The IEEE 802.15.4 has been simulated in NS-2. It was developed by the Joint Lab of Samsung in New York and confirmed to IEEE P802.15.4/D18 Draft [7]. Figure 4.1 outlines the IEEE 802.15.4 functional modules in NS-2 simulator, and a brief description is given below for each of the modules [22].

- **Wireless Scenario Definition:** It selects the routing protocol; defines the network topology; and schedules events such as initializations of PAN coordinator, coordinators and devices, and starting (or stopping) applications. It defines radio-propagation model, antenna model, interface queue, traffic

pattern, link error model, link and node failures, super-frame structure in beacon enabled mode, radio transmission range, and animation configuration.

- Service Specific Convergence Sub layer (SSCS): This is the interface between 802.15.4 MAC and upper layers. It provides a way to access all the MAC primitives, but it can also serve as a wrapper of those primitives for convenient operations. It is an implementation specific module and its function should be tailored to the requirements of specific applications.
- 802.15.4 PHY: It implements all PHY primitives.
- 802.15.4 MAC: This is the main module. It implements all the MAC sub layer primitives.



**Figure 4.1** NS-2 simulator for IEEE 802.15.4 [7]

### 4.3 Development of RTLD Routing Algorithm

The overall algorithm of RTLD is shown in Figure 4.2. Initially, the location management module is invoked in order to determine the sensor node location using three pre-determined nodes extracted from the neighbour table located in the neighbourhood management module. If the neighbour table is empty, the neighbour discovery is invoked to discover one-hop neighbour nodes. Once the location is

determined, the routing management is summoned to calculate the optimal forwarding node. The routing management selects the forwarding mechanism and requests the power management to adjust power transceiver for packet transmission. Besides, the routing management replies RTR packet if the sensor node is in same direction of the sink. Appendix B shows the integration of RTLD in NS-2. The following section will elaborate each functional module in routing management.

```

RTLD Routing ()
{
  Read neighbour table;
  If neighbour table is empty OR location is not determined
    Neighbourhood management implements neighbour discovery
  else
  {
    Routing management do{
      If type of packet is not RTR then
      {
        Select optimal forwarding node;
        Select forwarding mechanism;
        Request power adjustment;
        Data Packet is sent;
      }
    }
    else
    {
      Request power adjustment;
      RTR Packet is broadcasted;
    }
  }
}

if neighbour discovery was initiated and RTR is received then
{
  Routing Management invokes forwarding metric calculation;
  Neighbourhood management is invoked;
  Send RTR reply;
}

if neighbour discovery was initiated and RTR is not received then
  Routing Management invokes routing problem handler;
}

```

**Figure 4.2** RTLD routing algorithm



### 4.3.1 Development of Routing Management

Routing management is the main important module in RTLD and it has three components: optimal forwarding calculation, forwarding mechanisms and routing problem handler.

#### 4.3.1.1 Forwarding Metrics Optimization

In order to determine the value of PRR, the propagation model is used to predict the received signal power of each packet. When a packet is received with a signal power below the receiving threshold, it is marked as error and will be dropped by the MAC layer. There are three propagation models in NS-2, which are the free space model, two-ray ground reflection model and the shadowing model. The free space model and the two-ray model predict the received power as a deterministic function of distance. They represent the communication range in an ideal circle. In reality, the received power at certain distance is a random variable due to multi-path propagation effects, which are also known as fading effects. A more general and widely-used model is called the shadowing model [87].

In this simulation study, shadowing model is adopted to envisage the signal strength. The physical layer parameters received from neighbour node include signal strength, remaining power and timestamp. PRR is calculated based on the signal strength. The packet velocity is calculated based on one-hop delay and the remaining power is calculated based on battery voltage. After that, the metrics optimization is estimated in order to select the optimal one-hop neighbour that can forward the data packet towards the sink with increasing delivery ratio and power efficiency.

An optimization finds the optimal forwarding node from all feasible solutions. In this case, an optimization problem  $V$  consists of a quadruple parameters  $(I, f, m, g)$  [88] where:

- $I$  is a set of instances  $x$
- $f(x)$  is a set of feasible solutions  $y$
- $g$  is the goal function which is either min or max, and

- Given an instance  $x$  and a feasible solution  $y$  of  $x$ ,  $m(x, y)$  denotes the measure of  $y$ .

The optimization problem is then to find for instance  $x$  in  $I$ , an optimal solution that is a feasible solution  $y$  with which can be calculated as

$$m(x, y) = g \{m(x, \bar{y}) | \bar{y} \in f(x)\} \quad (4-1)$$

Equation (4-1) is a general optimization equation and it used in the proposed forwarding metrics optimization. The quadruple parameters ( $I, f, m, g$ ) in this research represent the trial of  $\lambda 1, \lambda 2$  and  $\lambda 3$ , the performance of RTLD in specific trial, measure of OF and max goal function respectively. Exhaustive search optimization method is used to select the weightage of the three metrics  $PRR, V$ , and  $V_{batt}$  that will produce optimal performance for RTLD as follows:

$$OF = \max (\lambda 1 * PRR + \lambda 2 * V_{batt} / V_{mbatt} + \lambda 3 * V / V_m)$$

$$\text{Where } \lambda 1 + \lambda 2 + \lambda 3 = 1 \quad (4-2)$$

where  $V_{mbatt}$  is the maximum battery voltage for sensor nodes and is equal to 3.6 volts [8].  $V_m$  is the maximum velocity of the RF signal, which is equal to the speed of light ( $3.0 * 10^8$  m/s) that will cross the distance between the transmitter and the receiver. The optimal weightage values of  $\lambda 1, \lambda 2$  and  $\lambda 3$  are estimated by exhaustive search using NS-2 simulator from all the probabilities of  $\lambda 1, \lambda 2$  and  $\lambda 3$  such that  $\lambda 1 + \lambda 2 + \lambda 3 = 1$ . It is important to note that we assume each  $\lambda$  is between 0.0 and 1.0 with one digit after floating point to minimize the complexity of calculation.

In order to calculate the total outcomes of  $\lambda 1, \lambda 2$  and  $\lambda 3$ , the sample space is calculated as in [89]:

- ◆ Let  $\Omega$  be the sample space of all  $\lambda$  trials and  $A$  is the event that  $\sum_{i=1}^3 \lambda_i$  is 1.
- ◆ Let  $N(\Omega)$  is the number of points in  $\Omega$  and  $N(A)$  is the number of points in  $A$ . Then,  $N(\Omega)$  can be calculated using the fundamental of counting principle as:

$$N(\Omega) = n_1 * n_2 * n_3$$

$$\text{where } n_1, n_2, n_3 = \text{number of possible values for } \lambda_1, \lambda_2, \lambda_3 \quad (4-3)$$

Due to the number of possible values for each  $\lambda$  is 11 (from 0.0 to 1.0) and  $n_1=n_2=n_3$ ,  $N(\Omega) = 11*11*11=1331$ . To find  $N(A)$  which  $A = \{(0,0,1), (0,0.1,0.9), \dots\}$ , equation (4-5) is used as :

$$N(A) = \sum_{i=1}^k i \quad \text{Where } k \text{ is the number of possible value of } \lambda=11 \quad (4-4)$$

Hence,  $N(A)$  is  $(1+2+3+4+5+6+7+8+9+10+11)$  66. Table 4.1 shows all points in  $N(A)$ . In this table, the total outcomes are equal to 66 trials.

In order to determine the optimal trial from the 66 trials, simulation has been developed with four types of grid network topology which are used to examine the network performance.

**Table 4.1** Trials of all points in  $A$

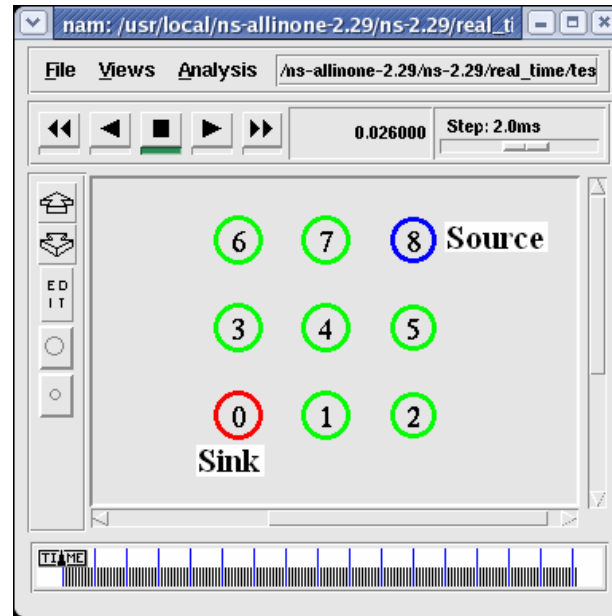
T	$\lambda_1, \lambda_2, \lambda_3$	T	$\lambda_1, \lambda_2, \lambda_3$	T1	$\lambda_1, \lambda_2, \lambda_3$	T	$\lambda_1, \lambda_2, \lambda_3$	T	$\lambda_1, \lambda_2, \lambda_3$
1	0.1,0.8,0.1	15	0.2,0.1,0.7	29	0.5,0.2,0.3	43	0.6,0.4,0.0	57	0.0,0.1,0.9
2	0.1,0.7,0.2	16	0.3,0.6,0.1	30	0.5,0.1,0.4	44	0.5,0.5,0.0	58	0.9,0.0,0.1
3	0.1,0.6,0.3	17	0.3,0.5,0.2	31	0.6,0.3,0.1	45	0.4,0.6,0.0	59	0.8,0.0,0.2
4	0.1,0.5,0.4	18	0.3,0.4,0.3	32	0.6,0.2,0.2	46	0.3,0.7,0.0	60	0.7,0.0,0.3
5	0.1,0.4,0.5	19	0.3,0.3,0.4	33	0.6,0.1,0.3	47	0.2,0.8,0.0	61	0.6,0.0,0.4
6	0.1,0.3,0.6	20	0.3,0.2,0.5	34	0.7,0.2,0.1	48	0.1,0.9,0.0	62	0.5,0.0,0.5
7	0.1,0.2,0.7	21	0.3,0.1,0.6	35	0.7,0.1,0.2	49	0.0,0.9,0.1	63	0.4,0.0,0.6
8	0.1,0.1,0.8	22	0.4,0.5,0.1	36	0.8,0.1,0.1	50	0.0,0.8,0.2	64	0.3,0.0,0.7
9	0.2,0.7,0.1	23	0.4,0.4,0.2	37	0.0,0.0,1.0	51	0.0,0.7,0.3	65	0.2,0.0,0.8
10	0.2,0.6,0.2	24	0.4,0.3,0.3	38	0.0,1.0,0.0	52	0.0,0.6,0.4	66	0.1,0.0,0.9
11	0.2,0.5,0.3	25	0.4,0.2,0.4	39	0.1,0.0,0.0	53	0.0,0.5,0.5		
12	0.2,0.4,0.4	26	0.4,0.1,0.5	40	0.9,0.1,0.0	54	0.0,0.4,0.6		
13	0.2,0.3,0.5	27	0.5,0.4,0.1	41	0.8,0.2,0.0	55	0.0,0.3,0.7		
14	0.2,0.2,0.6	28	0.5,0.3,0.2	42	0.7,0.3,0.0	56	0.0,0.2,0.8		

Figure 4.3 represents low density, medium density, high density with one traffic source and high density with several traffic sources network topologies. In each topology, simulations of 66 trials are studied. The network model and simulation parameters are similar to those in chapter 3 with different number of sensor nodes in each topology. In this simulation, the end-to-end deadline and the simulation time were fixed at 250 ms and 300 s respectively. There are three types of traffic load used in the simulation which are 1 packet/s (low traffic), 4 packet/s (medium traffic) and 10 packet/s (high traffic). The delivery ratio and the power consumption are used to measure the performance and efficiency of WSN. Analyses of all trials for each topology and the average performance are determined. Figures 4.4 and 4.5 show the performance in term of delivery ratio and power consumption of all trials at high traffic load. Trials (5, 12, 24, and 32) provide high performance as shown in Figure 4.4(e) and 4.5(e). From the extensive exhaustive search illustrated in appendix C, the maximum delivery ratio and minimum power consumption become apparent at four trials (5, 12, 24, and 32) for low, medium and high traffic load. Since the optimal performance transpires in trials (5, 12, 24, and 32), equation (3-1) can be rewritten as:

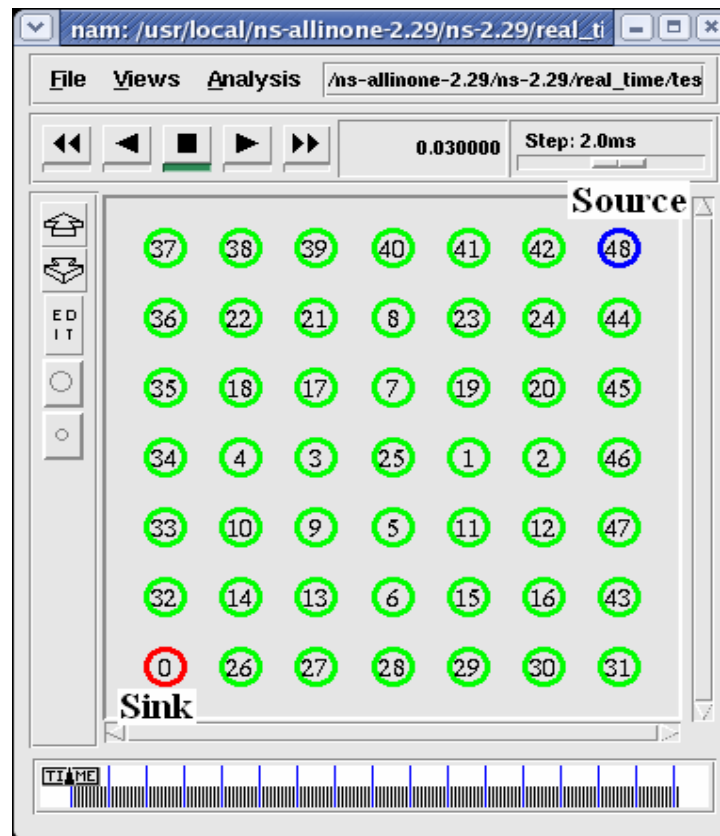
$$\begin{aligned}
 OF &= 0.1*PRR + 0.4*V_{batt}/V_{mbatt} + 0.5*V/V_m && \text{for trail 5 or} \\
 OF &= 0.2*PRR + 0.4*V_{batt}/V_{mbatt} + 0.4*V/V_m && \text{for trail 12 or} \quad (4-5) \\
 OF &= 0.4*PRR + 0.3*V_{batt}/V_{mbatt} + 0.3*V/V_m && \text{for trail 24 or} \\
 OF &= 0.6*PRR + 0.2*V_{batt}/V_{mbatt} + 0.2*V/V_m && \text{for trail 32}
 \end{aligned}$$

Since the four trials in equation (4-5) give similar result, any one of them can be chosen for routing decision in RTLD. However, trial 5 and 12 ensure lower link quality and less secure due to probabilistic of routing decision, higher load distribution and hence shorter packet delay. Trial 24 and 32 ensure higher link quality and better security, moderate load distribution and packet delay. Therefore, the preferred trial is 32 because it has the highest weightage (0.6) for PRR. The high PRR weight ensures lower packet loss, energy efficient and better security which will accomplish better WSN performance in RTLD. Thus, equation (4-6) is used for routing decision in RTLD.

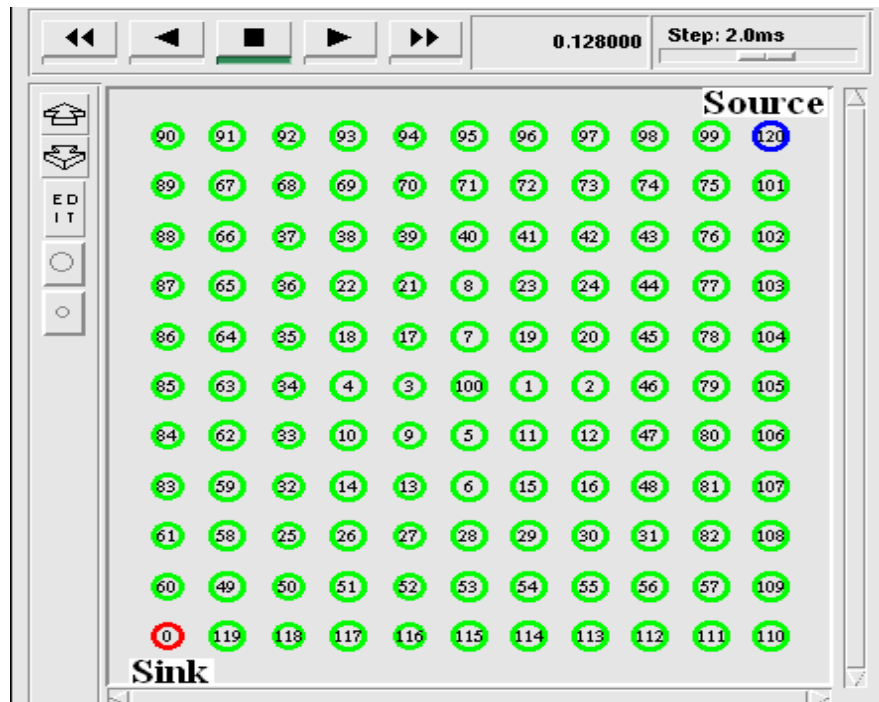
$$OF = 0.6 * PRR + 0.2 * V_{batt} / V_{mbatt} + 0.2 * V / V_m \quad (4-6)$$



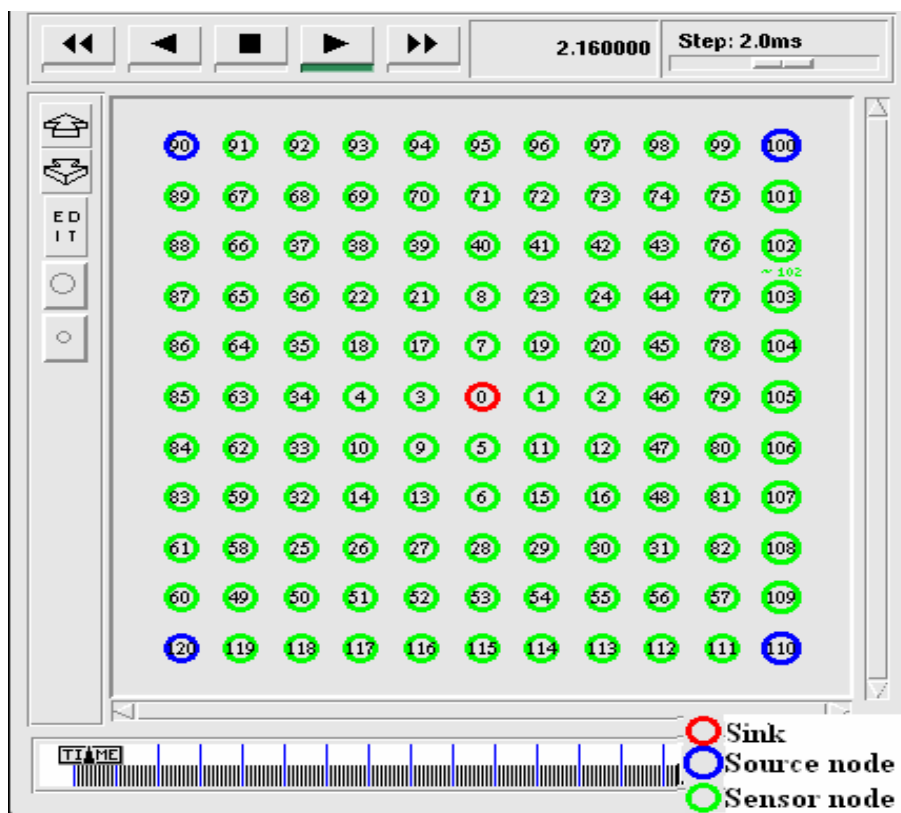
(a)



(b)

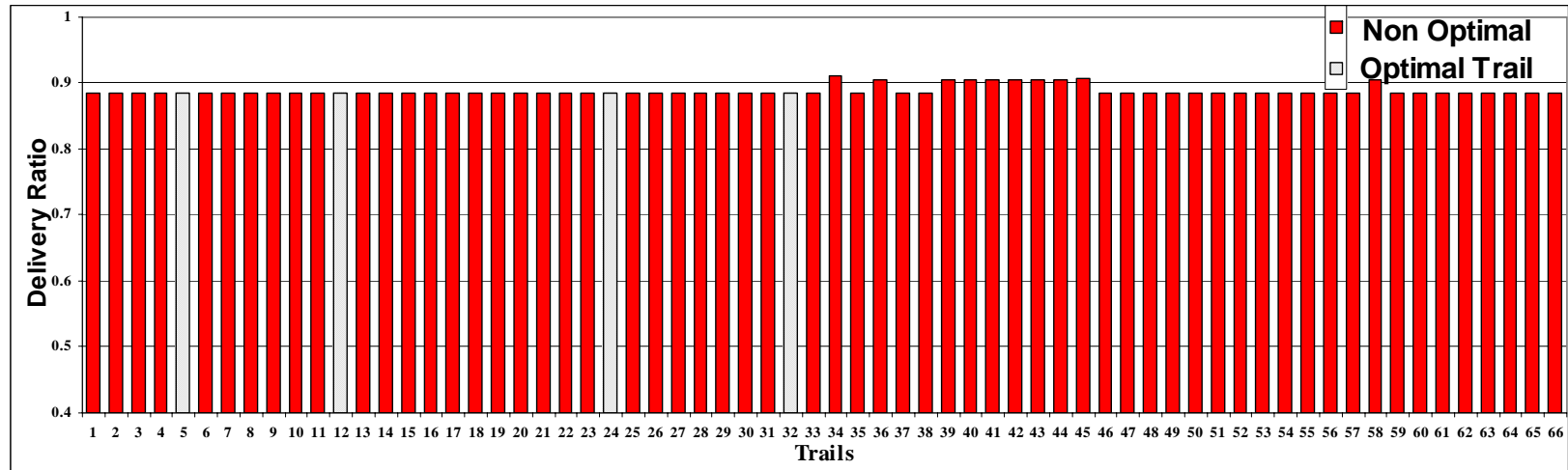


(c)

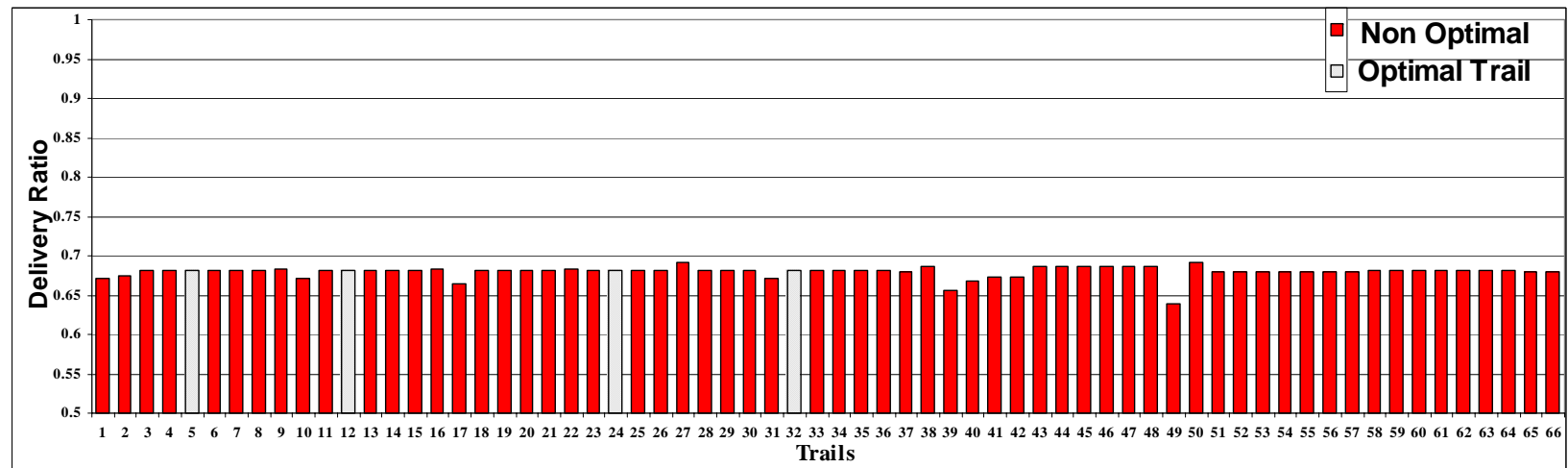


(d)

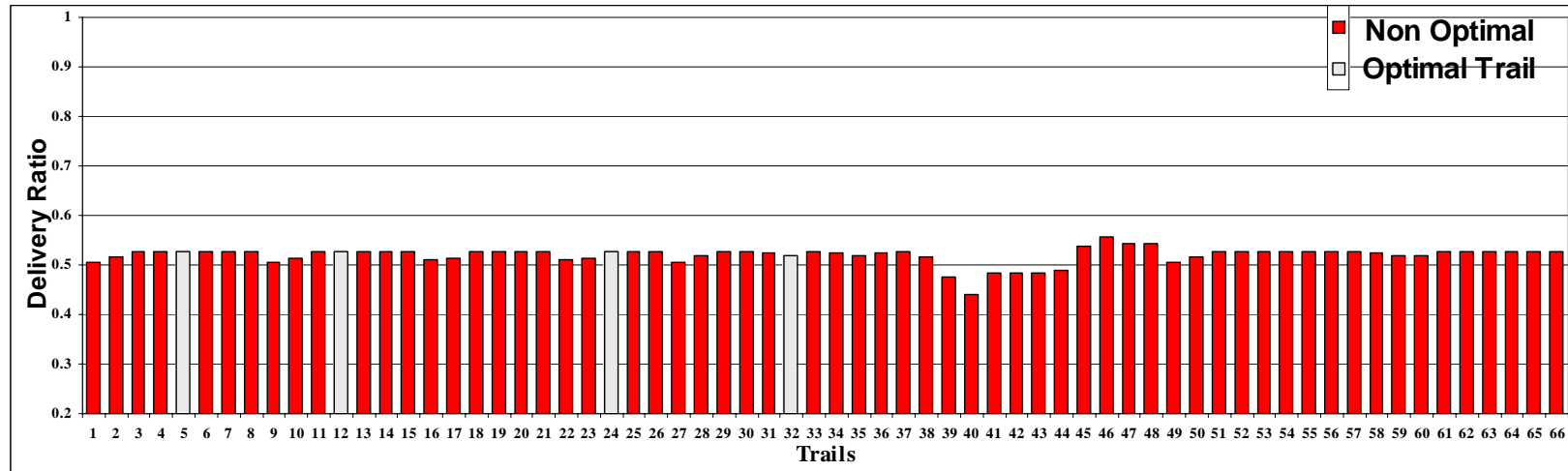
**Figure 4.3** Network Topology a) low density, b) medium density, c) high density , d) high density with several sources



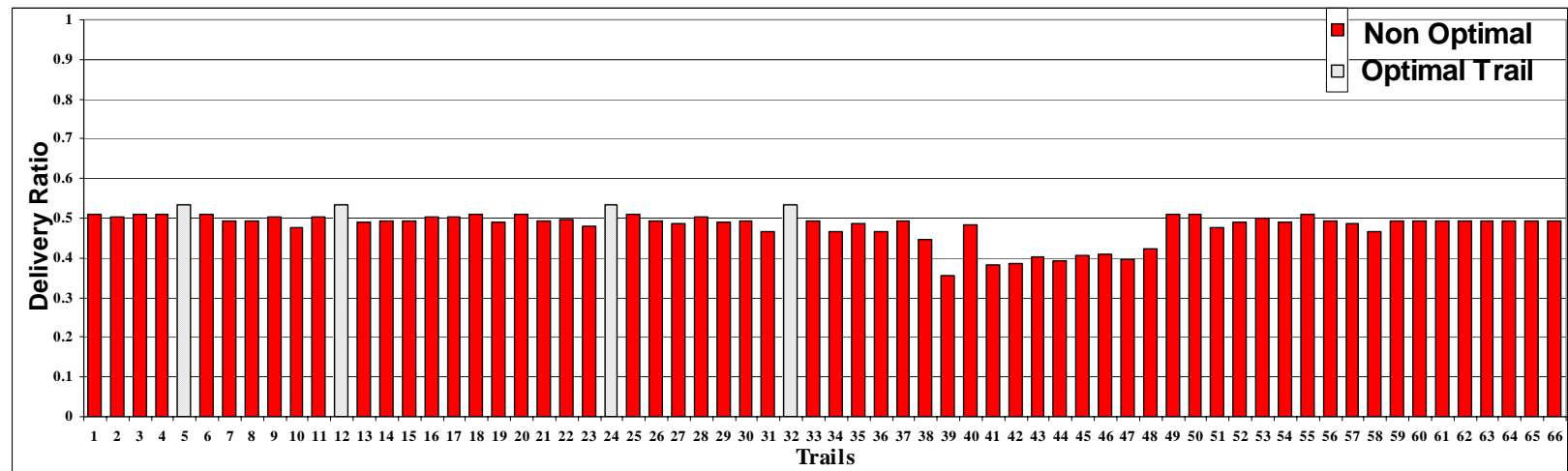
(a)



(b)

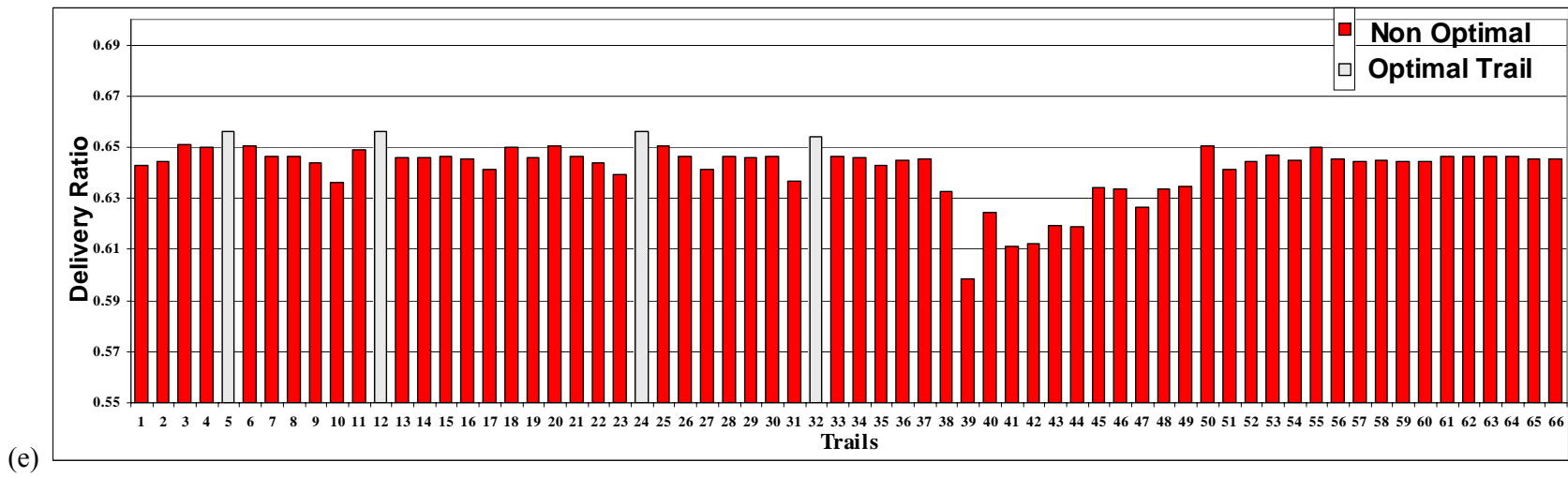


(c)

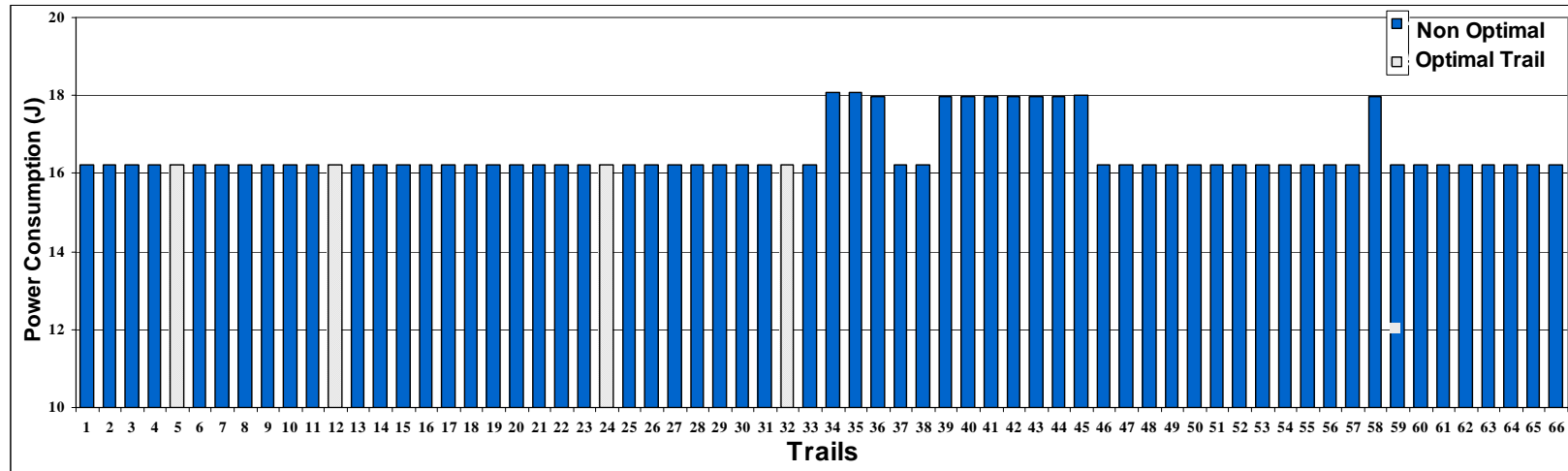


(d)

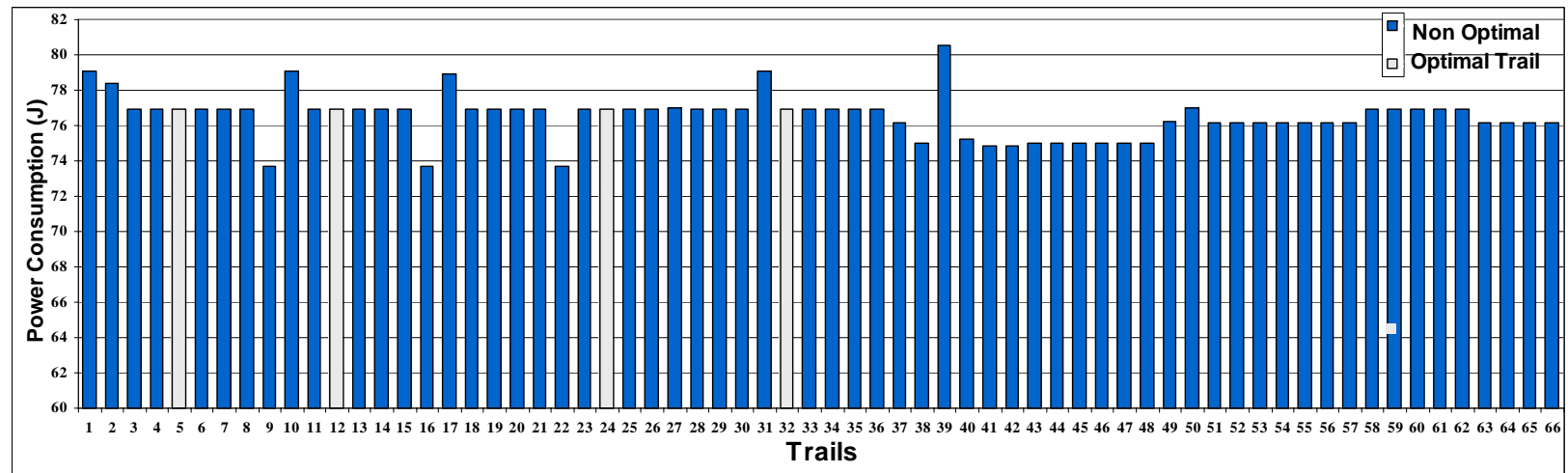




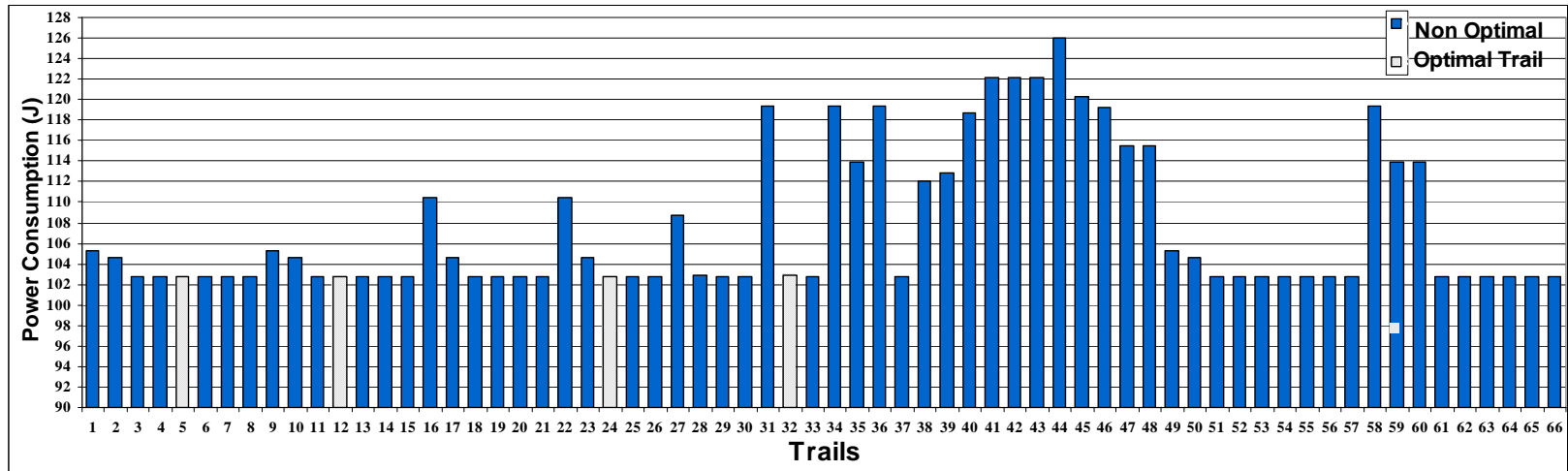
**Figure 4.4** Delivery Ratio at 10 packet/s; a) low density, b) medium density, c) high density d) high density with several sources and e) average



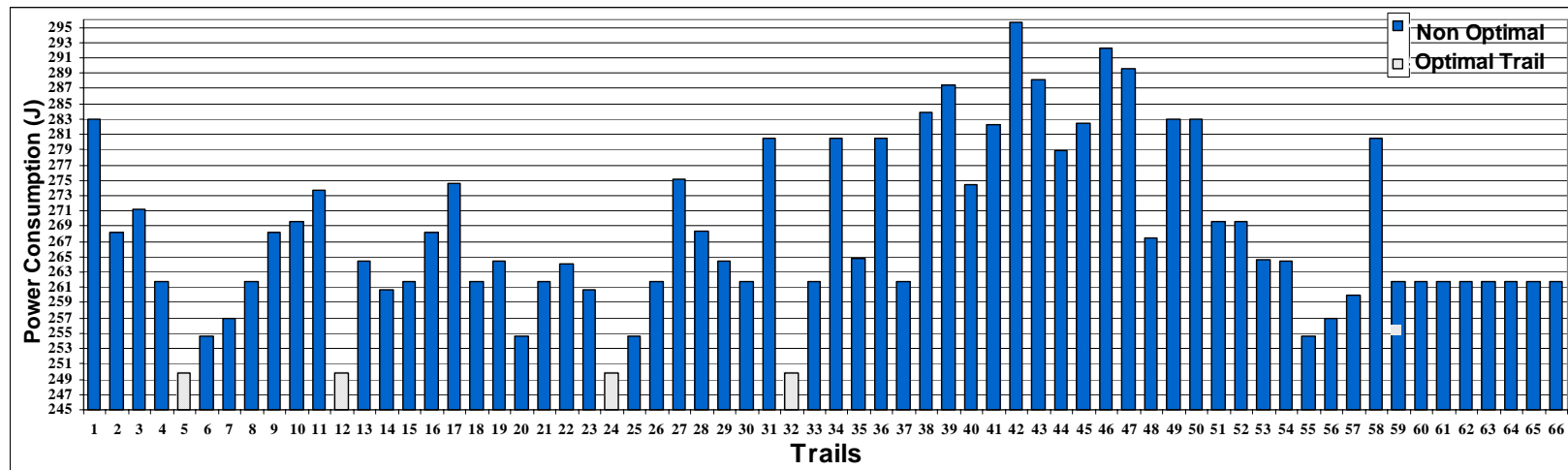
(a)



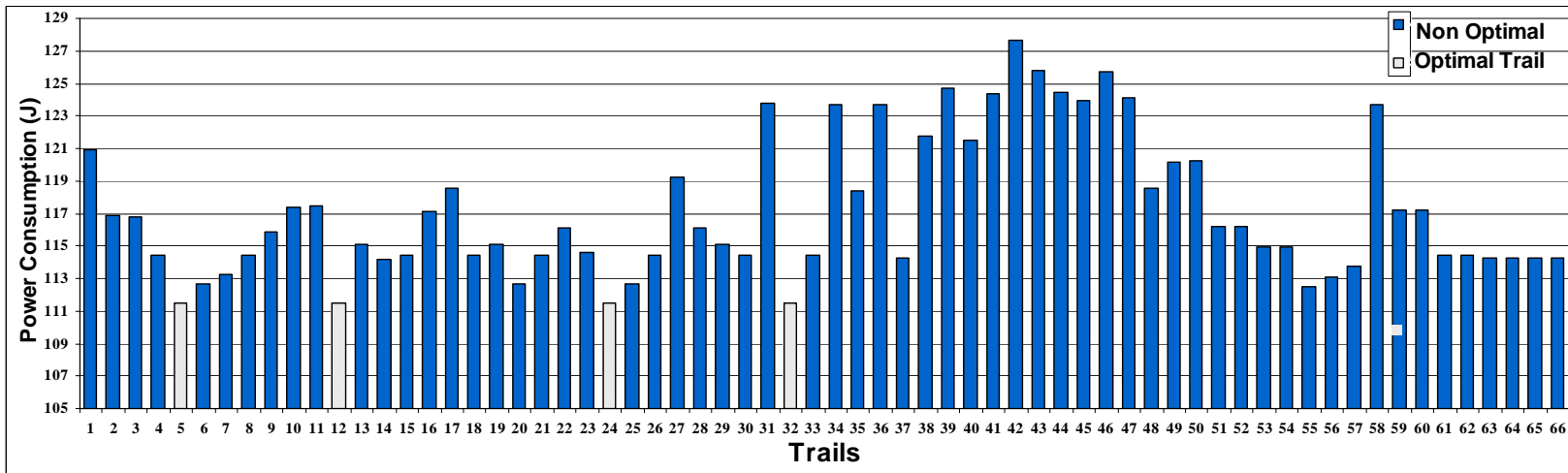
(b)



(c)



(d)



(e) **Figure 4.5** Power Consumption at 10 packet/s; a) low density, b) medium density, c) high density, d) high density with several sources and e) average

#### 4.3.1.2 Development of Routing Problem Handler

The algorithm of routing problem handler mechanism is illustrated in Figure 4.6. When the sensor node receives forwarding request from its parent, it will look for the sink in the neighbour table. If the sink is found, sensor node will forward the data packet to the sink. Otherwise, it will look for OF node in the neighbour table. If the sensor node finds an OF node, it will forward the data packet to that OF node. Otherwise, it will use the recovery mechanism. In the fast recovery mechanism, the sensor node will request the neighbourhood management to invoke neighbour discovery with maximum transmission power. If the fast recovery does not solve the problem, the sensor node will send the feedback packet to inform its parent about the hole problem. When the parent receives five feedback packets from the sensor node, it will look for new sensor node to forward data toward the sink.

```

Feedback Mechanism ( )
{
    feedback flag =1;
    if node receive packet to be forwarded then
        for all node in the neighbour table do
        {
            if look for the sink at the neighbour table = 0 then
                if look for OF node !=0 AND feedback flag!=0 then
                    Forward data packet to OF node
                Else
                    Forward data packet to the sink
            }
        }
    if no OF node in the neighbour table then
        Apply neighbour discovery with maximum transmission power
    if no OF node after increasing transmission power then
        Send feedback packet to node's parent;
    if node receive feedback packet from OF node then
        feedback flag = feedback flag -0.2;
    }

```

**Figure 4.6** Feedback mechanism algorithm

#### ♦ Effect of Routing Problem Handler

In order to show the effect of routing holes problem, the WSN model has been simulated using random distribution topology in NS-2. In this scenario, 50

nodes are distributed in random topology as shown in Figure 4.7. Node 10 is the source node and node 44 is the sink. In this case, nodes 20 and 26 have low battery power (0.9 J) and their energy diminishes shortly after packet forwarding. It is interesting to know that in NS-2, the sensor node will stop receiving and transmitting when the energy reaches zero. However, the MICAZ sensor node stops receiving and transmitting once the remaining energy reaches 2.4 V.

In Figure 4.7, the data packet can flow through two paths according to the chosen OF strategy; the first path is from the source 10 to node 8, 2, 5, 0, 29, 45, 20, 47 and 44; and the second path is from the source 10 to node 8, 2, 5, 0, 29, 36, 26, 14 and 44. If the first path experiences routing hole problem along the path, feedback packet is sent to the parent of the data packet to stop data forwarding as shown in Figure 4.8. In this figure, the node 29 forwards the data packet to the node 45 but unfortunately node 20 encounter energy problem at 222.524918282 simulation time as shown in Figure 4.8. According to the feedback mechanism, node 45 which is the parent of node 20 will discover that node 20 anticipates energy problem using the neighbour table. In this case, node 45 sends feedback packet to its parent, which is node 29 since it does not have other neighbour to forward the packet. Once node 29 receives the feedback packet, it will decrease the forwarding flag by 20%. This mean after five feedback packets are received at node 29 from node 45, node 29 will stop sending data packet to node 45. The feedback mechanism allows the node five chances to identify its case whether it has problem or not. After that, node 29 will search in its neighbour table for another neighbour that has active forwarding flag (1). As in Figure 4.8, node 29 selects node 36 as an OF node.

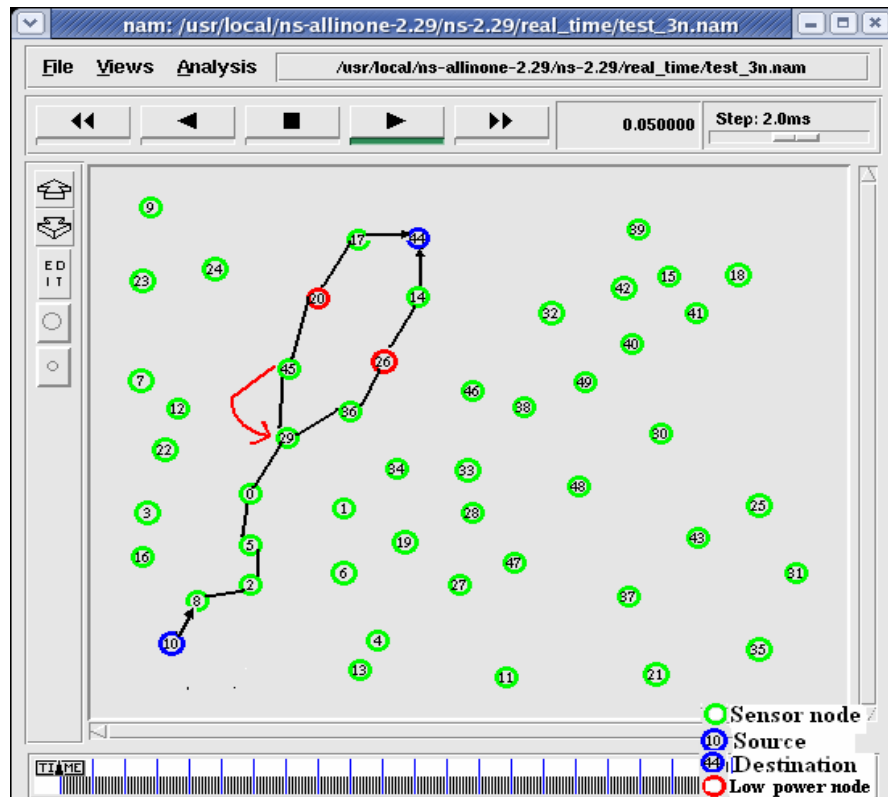


Figure 4.7 Random Distribution Topology

```

root@localhost:/usr/local/ns-allinone-2.29/ns-2.29/real_time
File Edit View Terminal Tabs Help
At time= 220.143078. I am 29 the data packet will send to 45
At time= 220.316158. I am 29 the data packet will send to 45
At time= 220.798520. I am 29 the data packet will send to 45
At time= 221.147368. I am 29 the data packet will send to 45
At time= 221.490406. I am 29 the data packet will send to 45
At time= 221.630640. I am 29 the data packet will send to 45
At time= 221.805910. I am 29 the data packet will send to 45
At time= 221.993376. I am 29 the data packet will send to 45
At time= 222.339678. I am 29 the data packet will send to 45
At time= 222.497270. I am 29 the data packet will send to 45
-----At time 222.524918282. I am 45, this is feedback pack
et to my parent 29 to stop packet forwarding -----
At time= 222.657600. I am 29 the data packet will send to 45
At time= 223.488248. I am 29 the data packet will send to 36
At time= 223.648080. I am 29 the data packet will send to 36
At time= 223.819958. I am 29 the data packet will send to 36
At time= 223.980288. I am 29 the data packet will send to 36
At time= 224.317040. I am 29 the data packet will send to 36
At time= 224.479368. I am 29 the data packet will send to 36
At time= 224.671456. I am 29 the data packet will send to 36
At time= 224.820166. I am 29 the data packet will send to 36
At time= 225.020190. I am 29 the data packet will send to 36
At time= 225.175528. I am 29 the data packet will send to 36
At time= 225.350236. I am 29 the data packet will send to 36

```

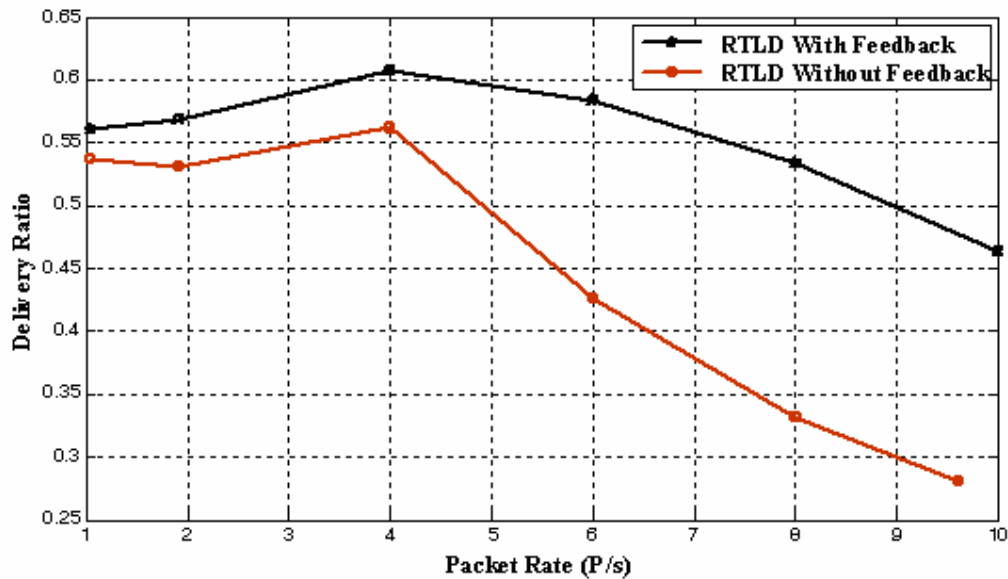
Figure 4.8 Feedback Mechanism result

In the simulation, the traffic load is varied from 1 to 10 packet/s while the end-to-end deadline and simulation time were fixed at 250 ms and 300s respectively.

The result in Figure 4.9(a) shows that routing with feedback increases the delivery ratio by 15% as the packet rate varies. This is mainly due to the routing management is having heavy flexibility to deal with the routing problem. Besides, the feedback allows route recovery hence achieving higher throughput. The throughput without feedback becomes worst as the traffic load increases. This is due to the fact that the source node does not know the path status after its one-hop neighbour. Thus, more packets are lost as more traffic is generated.

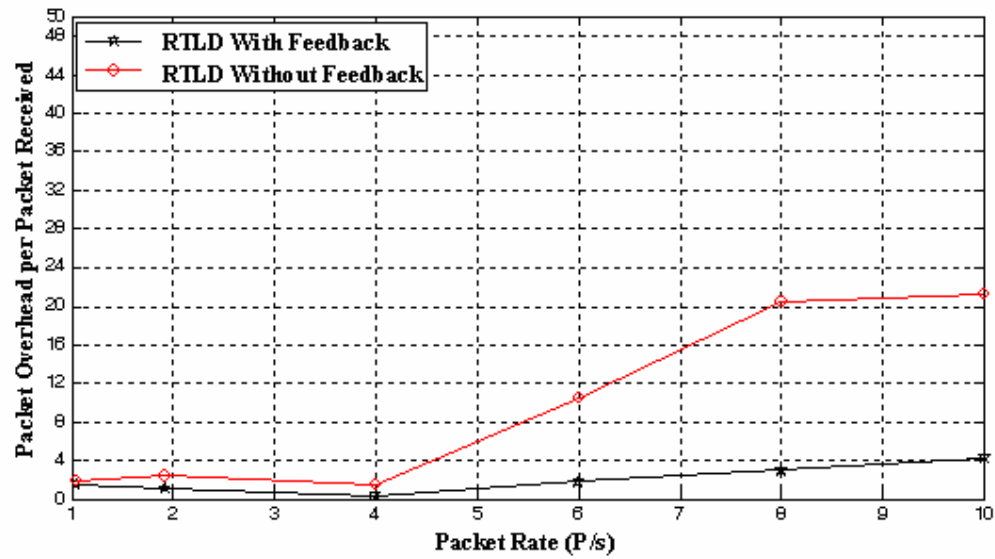
Figure 4.9(b) shows that routing with feedback packet spends less number of packets overhead compared to routing without feedback packet. This is primarily due to the parent node in the routing without feedback is sending more packets overhead to recover the bad forwarding path even if there is no neighbour reply. Routing with feedback permits the parent node to send packets overhead only for a short time to confirm that the forwarding path is unstable. If the forwarding path is not recovered, the sensor node will launch feedback packet to its parent. Otherwise, the parent node refrains from sending packet to sensor node.

Figure 4.9(c) shows that routing with feedback packet consumes less power compared to routing without feedback packet. This is mainly due to higher packet overhead in the routing without feedback.

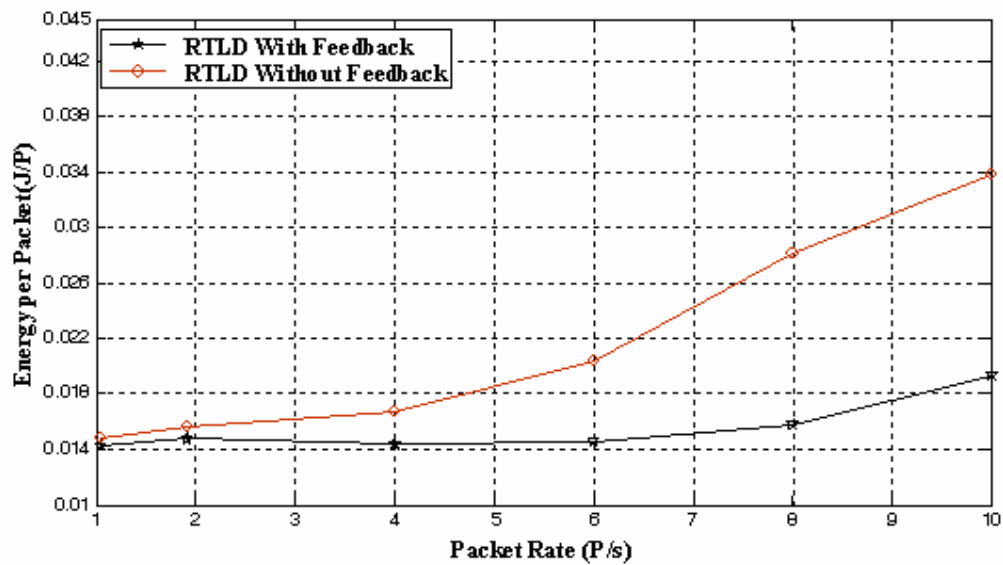


(a)





(b)



(d)

**Figure 4.9** Performance of RTLD using feedback packet at different traffic load  
a) Delivery ratio; b) Normalized packet overhead; and c) Energy consumption.

#### 4.3.1.3 Development of Forwarding Mechanisms

In this section, the developments of forwarding mechanisms are elaborated. Figure 4.10 shows the algorithm for the forwarding mechanisms. In this figure, there are two types of forwarding; unicast and geodirectional-cast. The unicast forwarding

needs to know the OF node from the neighbour table. If OF node is not available, the unicast forwarding will activate the neighbour discovery. The geodirectional-cast mechanism broadcasts the data packet at the source node only and unicasts the data packet at intermediate nodes between the source and the sink as in unicast forwarding mechanism.

```

Forwarding Services ()
{
  Switch (type of forwarding)
  {
    Case 0: // Unicast forwarding
      Look for OF node in the neighbour table;
      If OF node > -1 then
        Implement Unicast_forwarding();
      Else
        Implement neighbourhood_discovery();
      End if
    Break;

    Case 1: //Geo-directinalcast forwarding
      If the transmitter is the packet source then
        Set next hop address= broadcast address;
        Set Destination address = broadcast address;
        Implement broadcasting;
      Else
        Implement Unicast_forwarding;
      End if
    } }

```

**Figure 4.10** Algorithm of forwarding mechanisms

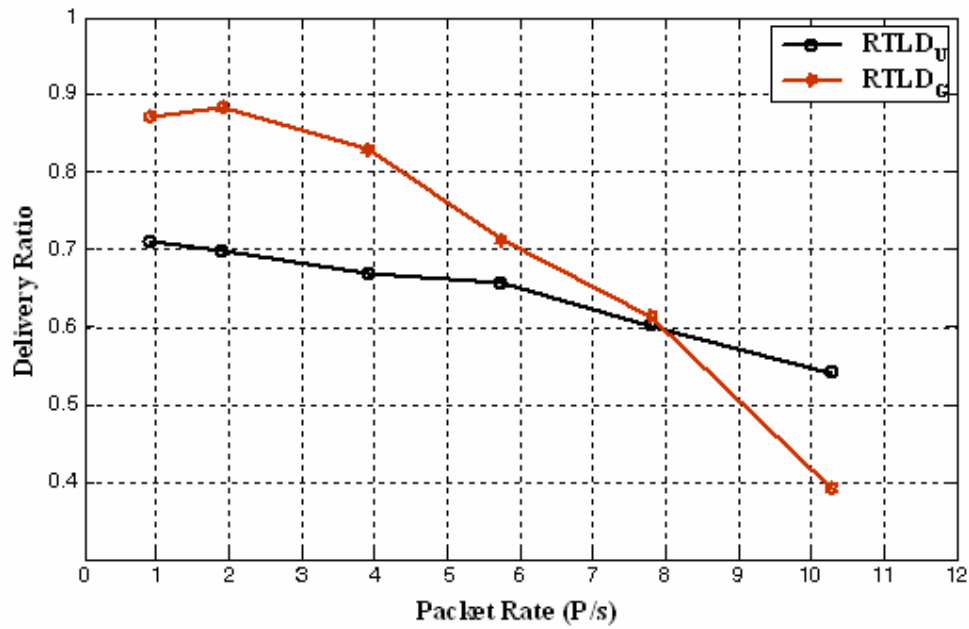
#### ♦ Influence of Forwarding Mechanism

RTLD routing that uses geodirectional-cast forwarding is defined as (RTLD<sub>G</sub>) while RTLD routing that uses unicast forwarding is termed as (RTLD<sub>U</sub>). Simulation study on the influence of the forwarding mechanism is carried out using parameters configured in Table 3.1. The packet rates were varied while the packet lifetime and simulation time were fixed at 250 ms and 100s respectively. The traffic load is varied from 1 to 10 packet/s. The simulation results in Figure 4.11(a) show that the RTLD<sub>G</sub> increases delivery ratio by 20% compared to RTLD<sub>U</sub>. This is due to feasible multiple paths forwarding in RTLD<sub>G</sub>. However, RTLD<sub>G</sub> drops sharply when

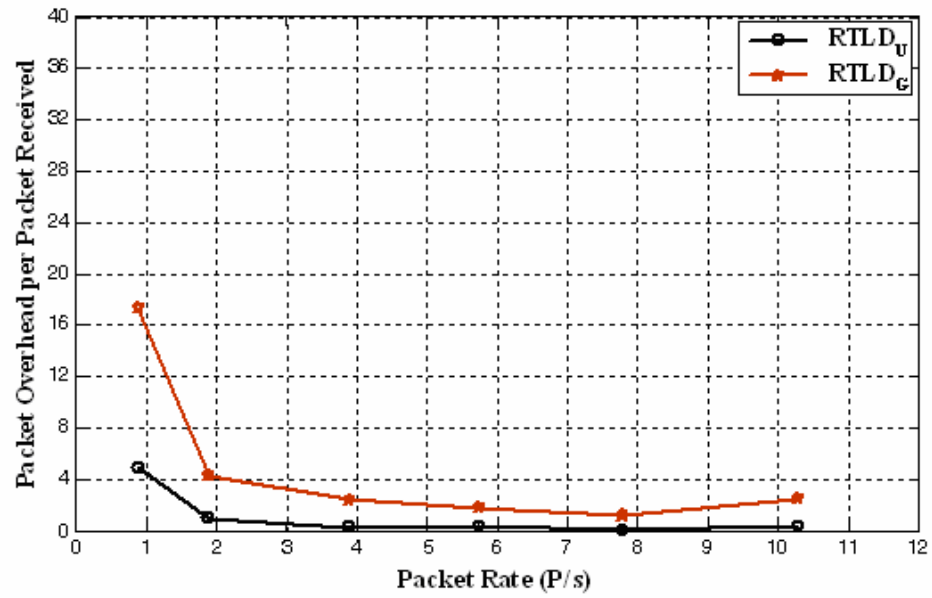
the traffic load is high mainly due to congestion in the network. Moreover, the IEEE 802.15.4 MAC is designed for low traffic rate and does not work well with high traffic load [7]. The flooding in the direction to the destination causes congestion near the source of the data packet, channel contention and interference.

Figure 4.11 (b) shows  $RTLD_G$  spends 4% higher packet overhead compared to  $RTLD_U$ . This is generally due to broadcasting of data packet in the first one-hop when the packet is travelling from the source to the destination.

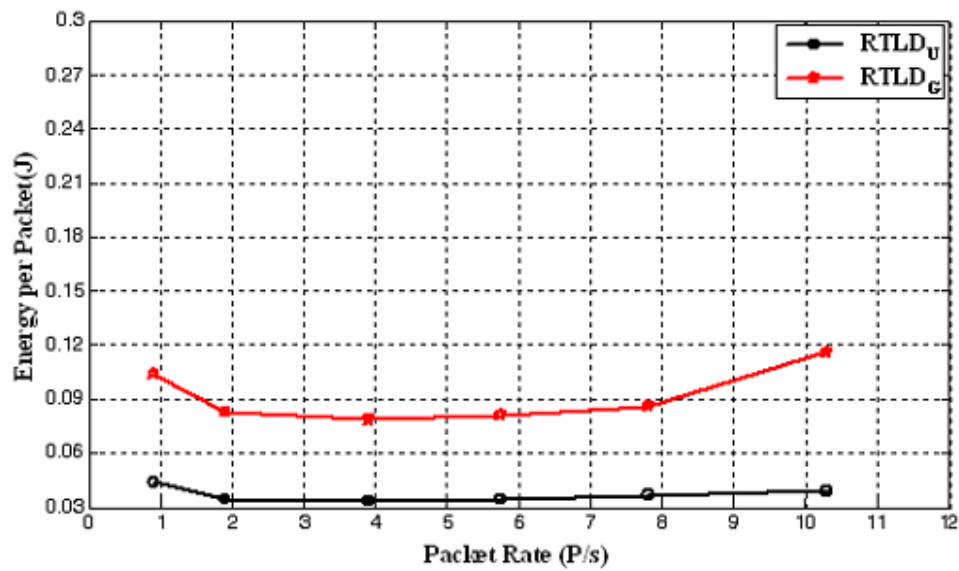
Figure 4.11 (c) shows  $RTLD_G$  consumes 9% more power compared to  $RTLD_U$  to achieve high delivery ratio. This is largely due to its forwarding strategy spending more packets overhead for the initial broadcasting of packets.



(a)



(b)



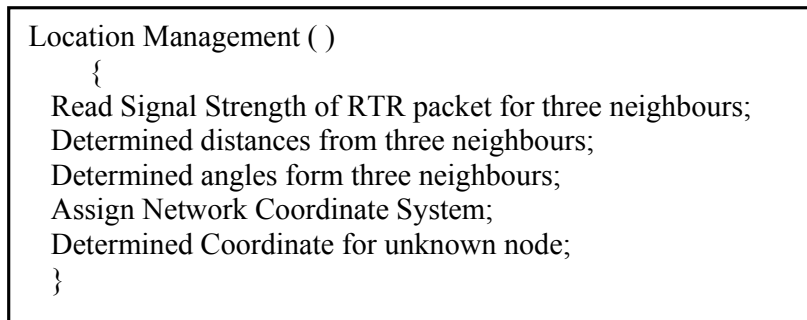
(c)

**Figure 4.11** Performance of  $RTLD_G$  and  $RTLD_U$  at different packet rate a) Delivery ratio; b) Normalized packet overhead; and c) Energy consumption.

### 4.3.2 Development of Location Management

The location management has been developed based on equations (3-21) and (3-22) mentioned in chapter 3. Due to the demand for information such as node location, PRR, packet velocity and remaining power in the initial routing process, the first 10 s is used for initialization of the neighbour table in each sensor node. In order to calculate node location, RTR packets are broadcast from every sensor node to its one-hop neighbour.

Figure 4.12 illustrates the location management algorithm. In this figure, once unknown node receives RTR packet, it obtains the location of the RTR transmitter and calculates its distance to the RTR transmitter. In addition, unknown node will use three pre-determined nodes to calculate the angles and to assign NCS. Based on NCS information and angles, the unknown node location will be obtained.



**Figure 4.12** Algorithm of location management

In Figure 4.13, a network grid of 25 sensor nodes is simulated to implement location management algorithm. Node 24 is the source and node 0 is the sink. Three pre-determined nodes 0, 10 and 13 are assumed known and the locations of remaining nodes are determined based on the location management mechanism. Figure 4.14 shows the results of location management mechanism for remaining node in the grid. Each line shows the angle from NCS, sensor node address and the coordination of sensor node.

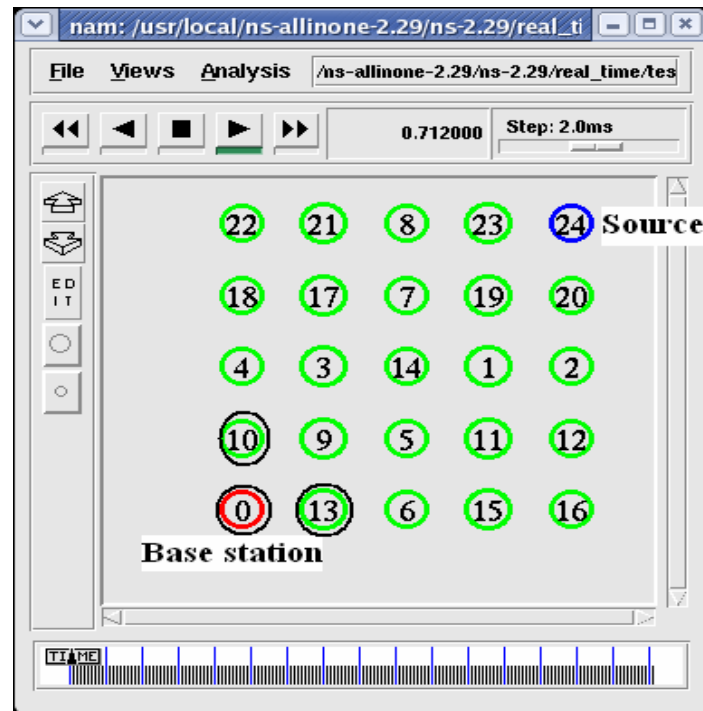


Figure 4.13 Network grid with three pre-determined nodes

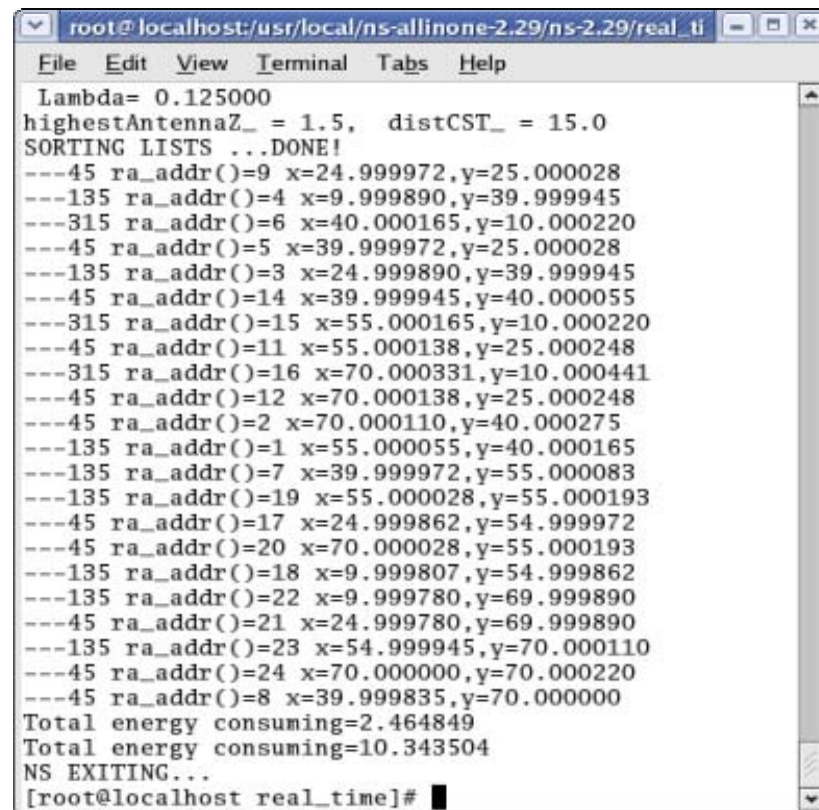
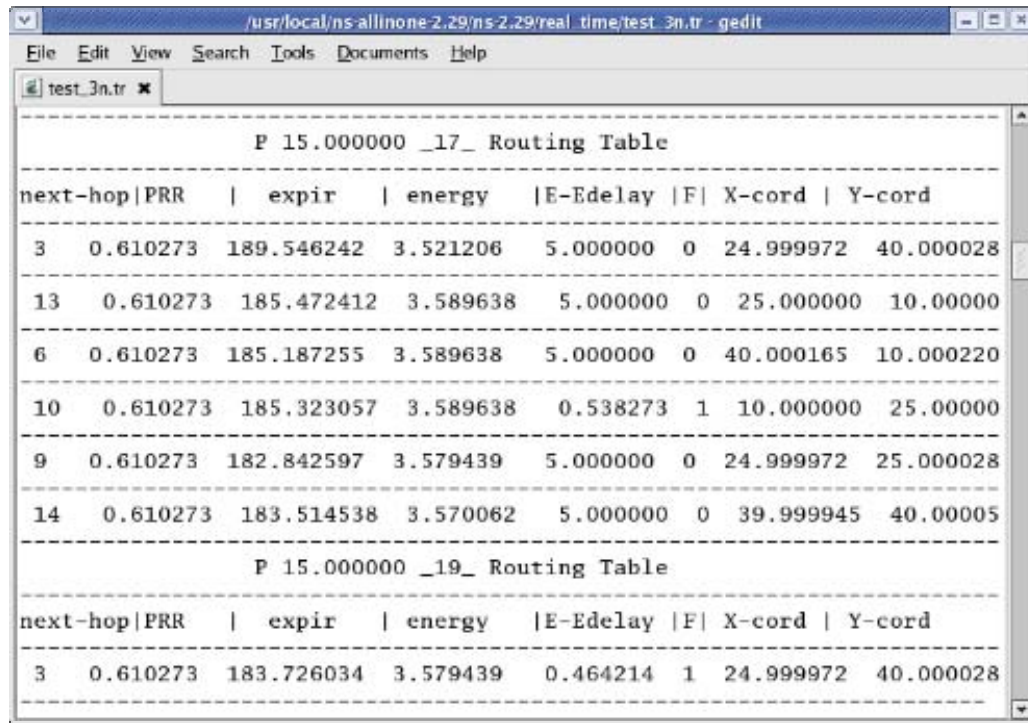


Figure 4.14 Locations determined for sensor node in WSN

### 4.3.3 Development of Neighbourhood Management

RTLD uses neighbour table to record information about one-hop neighbours. The neighbour table class can be realized as different class or as any other data structure (e.g. a hash table). For each entry in the neighbour table, the information such as node id, remaining power, one-hop end-to-end delay, PRR, forward flag, location information and expiry time are stored. Figure 4.15 shows the snapshot of the neighbour table. Doubly linked list is used as the storage structure because it allows simple mechanism to add or delete a record to or from the neighbour table.



P 15.000000 _17_ Routing Table							
next-hop	PRR	expir	energy	E-Edelay	F	X-cord	Y-cord
3	0.610273	189.546242	3.521206	5.000000	0	24.999972	40.000028
13	0.610273	185.472412	3.589638	5.000000	0	25.000000	10.000000
6	0.610273	185.187255	3.589638	5.000000	0	40.000165	10.000220
10	0.610273	185.323057	3.589638	0.538273	1	10.000000	25.000000
9	0.610273	182.842597	3.579439	5.000000	0	24.999972	25.000028
14	0.610273	183.514538	3.570062	5.000000	0	39.999945	40.000005

P 15.000000 _19_ Routing Table							
next-hop	PRR	expir	energy	E-Edelay	F	X-cord	Y-cord
3	0.610273	183.726034	3.579439	0.464214	1	24.999972	40.000028

**Figure 4.15** RTLD neighbour table

Figure 4.16 shows neighbourhood management algorithm. Here, neighbour discovery is invoked to discover the one-hop neighbour of the sensor node. If the sensor node receives RTR reply, the neighbourhood management checks the neighbour table size. If the neighbour table is not full, the add function is invoked. Otherwise, the replacement function is attempted. The add function is also used to update old neighbour record. The neighbour record will be removed from the neighbour table after 180s if the neighbour table does not receive neighbour update.

```

Neighbourhood management ( )
{
  Implement Neighbour Discovery;
  for (all node in the neighbour table)
  {
    if the reply is new then
      Check neighbour table size
      If neighbour table is not full then
        Add_new_record();
      Else
        Replecement_record();
      End if
    Else
      Update_record();
    End if
  }
  Remove_record(180s);
}

```

**Figure 4.16** Neighbourhood management algorithm

#### 4.3.3.1 Neighbour Discovery Function

The neighbour discovery functions according to the role defined as RTR and RTR reply packets. Figure 4.17 shows the sending control packet algorithm. In order to send a control packet, NS-2 must first allocate the packet using `allocpkt()` function. Then the packet header fields will be completely filled depending on the packet specification. The `next_hop` field is filled with broadcasting address. Finally, RTR packet is put in queue to be scheduled for transmission purpose.

```

Send control packet ( )
{
  Packet is allocated;
  The header is filled;
  Next_hop is broadcasting;
  Packet_dest is broadcasting;
  Packet_type is RTLD;
  Packet is scheduled;
}

```

**Figure 4.17** Sending control packet algorithm



The RTR reply algorithm is shown in Figure 4.18. When the RTR packet is received, the distance between the packet receiver and the sink is calculated and compared with the distance between the packet transmitter and the sink. If the distance between the packet receiver and the sink is less than the distance between the packet transmitter and the sink, the receiving source code will check the quadrant of the packet receiver and compared with the quadrant of the sink. Otherwise, RTR packet will be ignored. This process is important to ensure convergence between the forwarding node and the sink. It is interesting to note that the quadrant of the forwarding node and the sink is determined according to the transmitter.

```

Receive control packet ( )
{
  if the received packet is RTR then
  {
    if distance(receiver, sink) < distance(transmitter, sink) then
    {
      if Quadrant of receiver = Quadrant of sink then
        RTR reply sends to transmitter;
      Else
        RTR packet is ignored;
    }
    else
      RTR packet is ignored;
  }
  else
    RTR reply is received;
    Forwarding metrics are calculated;
    Send neighbour information to neighbourhood management;
  }
}

```

**Figure 4.18** Receiving control packet algorithm

#### 4.3.4 Development of Power Management

The radio transceiver has four different states: down or sleep state (1  $\mu$ A) with voltage regulator off, idle state (426  $\mu$ A) with voltage regulator on, transmit state (17 mA) at 1 mW power transmission and receive state (19.7 mA) [80, 83]. According to the data sheet [80], the receive mode has the higher power consumption than all other states. The power management algorithm is elaborated in Figure 4.19. The sensor node changes the transceiver

state to idle if it discovers a neighbour in the same quadrant direction of the destination. When the sensor node broadcasts RTR, it changes the transceiver state to transmit mode. Otherwise, the transceiver will be in the receive mode waiting for replies from its neighbours or waiting data packet from its parent.

Since the time taken to switch from sleep state to idle state takes close to 1 ms [83], it is recommended that a sensor node should stay in the idle state if it has neighbours with forward flags equal to 1. Thus, the total delay from the source to the destination will be decreased. The power management also proposes that a sensor node should change its state from idle to sleep if it does not have at least one neighbour in the neighbour table that can forward data packet towards the destination.

```

Power Management ( )
{
Initial state = sleep;
if neighbour discovery is invoked or data packet is forwarded then
    Change power state to idle then to transmit mode
Else
    If sensor node waits RTR reply or data packet then
        Change power state to receive mode
    Else
        Change power state to idle mode
}

```

**Figure 4.19** Power management algorithm

Beside that, the power management distributes the forwarding load to the forwarding candidates in the neighbour table. It updates the neighbour table after 3 minutes and the previous OF node may not be selected because the link quality, velocity and remaining power have changed. It is important to note that if the remaining power of forwarding nodes decreases, the probability to be selected again for the next period also decreases as indicated in equation (4-6). Hence, the forwarding load will be distributed to all nodes in the direction of the destination. Section 4.3.4 will explain the significance of the load distribution in more detail.

#### 4.4 Simulation Analysis of Proposed Routing Protocol

The proposed RTLD routing protocol is studied through simulation process. Its performance is analyzed and compared with three other baseline protocols that consider link quality (LQ), velocity with energy efficiency (RTPC) and multiple communication speeds (MM-SPEED) in the routing decisions. In LQ, the forwarding policy selects next hop based on the highest PRR in the neighbour table. MM-SPEED selects the next hop based on the proper speed options that meets the end-to-end deadline. The feedback control and differentiated reliability in MM-SPEED routing protocol is not taken into account in this work because it requires modification to the MAC layer protocol. RTPC protocol forwards the packets to the most energy efficient forwarding node that meets the packet's velocity [33]. In this study, all the above baseline protocols and RTLD operate at a default transmission power level of 0 dBm (1 mW). The simulation evaluates the performance of both forwarding policies assuming the neighbour table of each node does not have forwarding choices. The OF node is determined on-line according to equation (4-6). The simulation is also designed to evaluate the performance of the forwarding policies running in conjunction with various management policies. In the following simulation study, RTLD utilizes on demand neighbour discovery scheme. When the periodic beacon scheme is employed, data packets will transmit after 10s to allow neighbour table forwarding metrics to be initialized. It is important to note that the data packet travels between 5 and 10 hops to reach the sink.

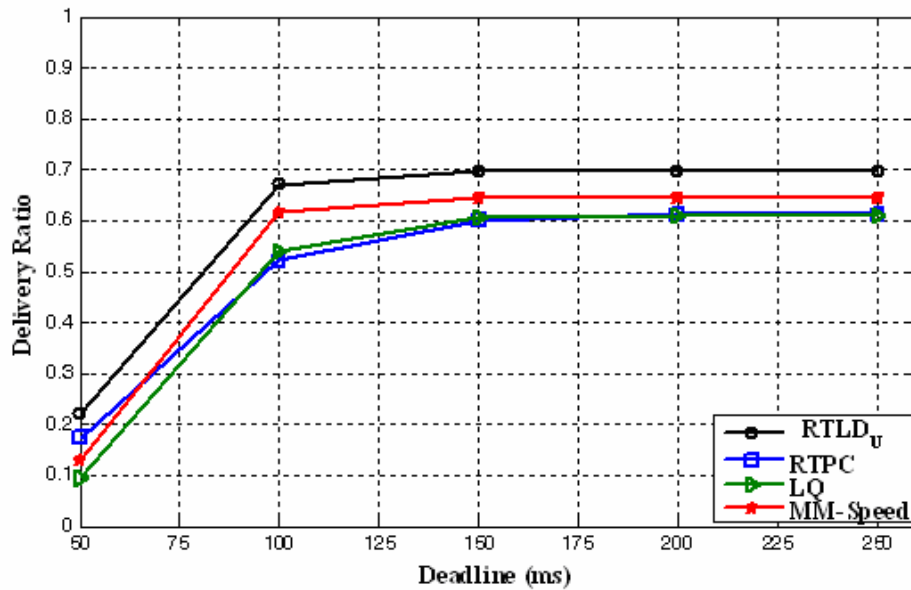
##### 4.4.1 Effect of End-to-End Packet Deadline

The real-time transfer requires that a packet reaches its destination within the deadline period. The deadline delimits the lifetime of a packet traversing the WSN. In the simulation, the end-to-end packet deadline was varied while the simulation time and traffic load were fixed at 100s and 10 packet/s respectively. The simulation results in Figure 4.22 show that  $RTLD_U$  experiences higher delivery ratio by 4% to 7% than the baseline routing protocols. The finding also shows that  $RTLD_U$  provides the highest delivery ratio for all packet deadlines. This is primarily due to its

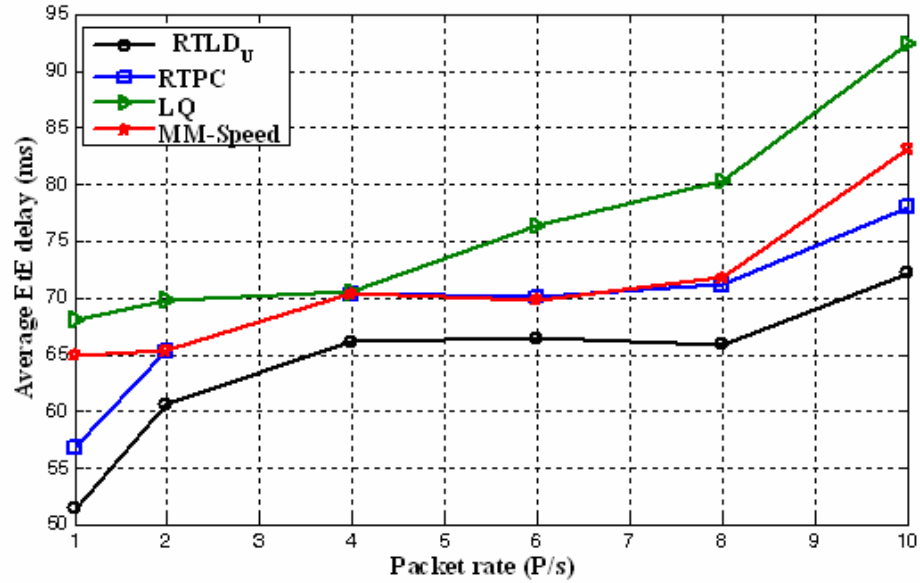
forwarding strategy which chooses the next hop that has the optimal combination of the best link quality, remaining power and packet velocity. Besides, Figure 4.22 shows the minimum packet deadline is about 150 ms. Beyond this, the packet delivery ratio remains unchanged at its maximum throughput.

Figure 4.23 shows the average end-to-end delay comparison between  $RTLD_U$  and baseline routing protocols for different packet rate.  $RTLD_U$  possesses short average delay compared to baseline routing protocols. This is primarily due to its forwarding strategy which considers link quality with packet velocity that minimize the average delay.

The results in Figures 4.22 and 4.23 justify that the end-to-end delay experience does not exceed the set limit 250 ms that also defined in [9, 33]. The proposed system recommends more than 250 ms if the distance between the source and the sink is far away (more than 17 hops). Appendix C presents more results showing the effect of packet deadline at different traffic loads.



**Figure 4.20** Comparison between RTLD and baseline routing protocols for different packet deadline



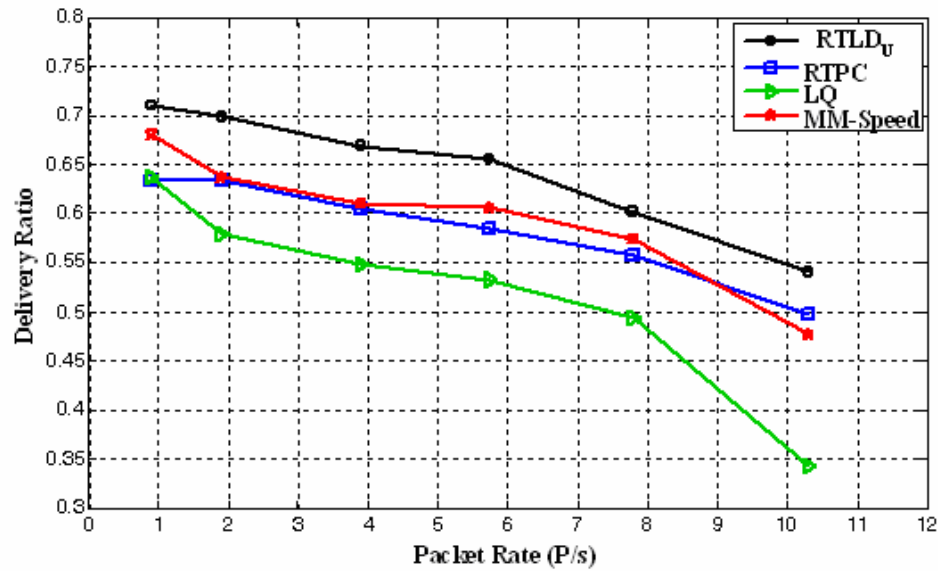
**Figure 4.21** Comparison average end-to-end delay between RTLD and baseline routing protocols for different packet rate

#### 4.4.2 Impact of Varying Network Load

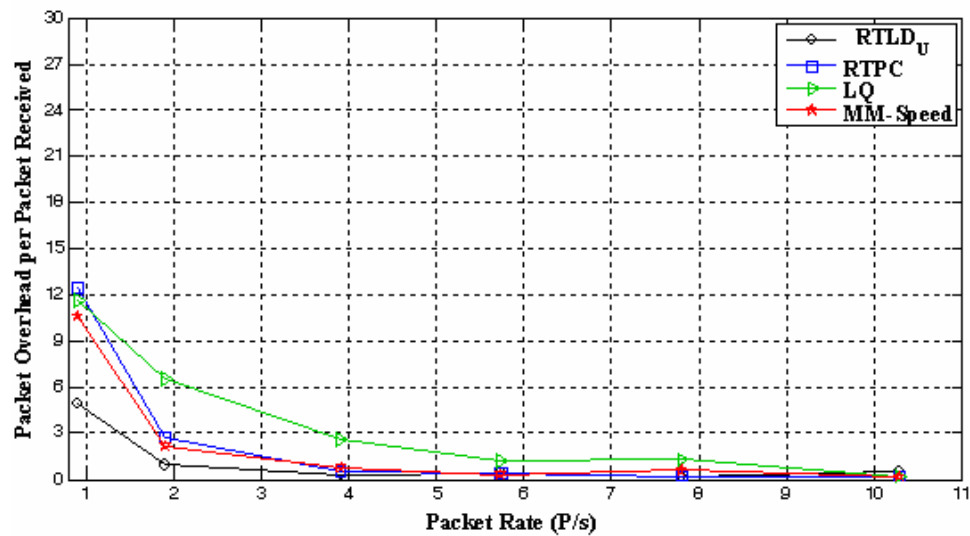
In this simulation, the packet rates were varied while the end-to-end deadline and simulation time were fixed at 250 ms and 100s respectively. The traffic load is varied from 1 to 10 packet/s to emulate low data rate in IEEE 802.15.4. The simulation results in Figure 4.24(a) show that RTLD<sub>U</sub> experiences higher delivery ratio than the baseline protocols by 4% to 7% as observed in section 4.4.1. This is because RTLD<sub>U</sub> takes into consideration link quality with packet velocity that guarantee high delivery ratio and energy efficient [10]. In addition, Figure 4.24(a) shows the delivery ratio decreases as the load in the network increases. This is mainly due to packet loss because of network congestion and packet collision.

Figure 4.24(b) shows that RTLD<sub>U</sub> spends less number of packets overhead compared to baseline routing protocols. This is largely due to its neighbour discovery which does not allow the one-hop neighbour to reply if it is not in the direction to the destination. Hence, the probability of collision is reduced and packet overhead is

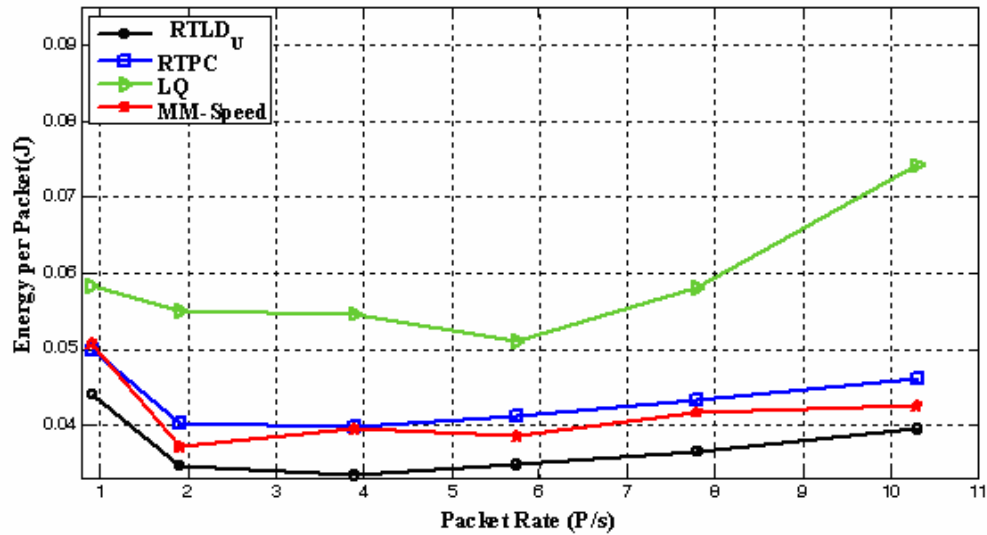
minimized. On the other hand, the baseline forwarding strategy does not consider probability of collision due to neighbour discovery which degrades the delivery ratio and energy efficiency. Figure 4.24(c) demonstrates that  $RTLD_U$  consumes less power compared to baseline routing protocols due to less packet overhead.



(a)



(b)



(c)

**Figure 4.22** Comparison between RTLD and baseline routing protocols at different packet rate a) Delivery ratio; b) Normalized packet overhead; and c) Normalized energy consumption.

#### 4.4.2.1 Effect of Poisson Traffic

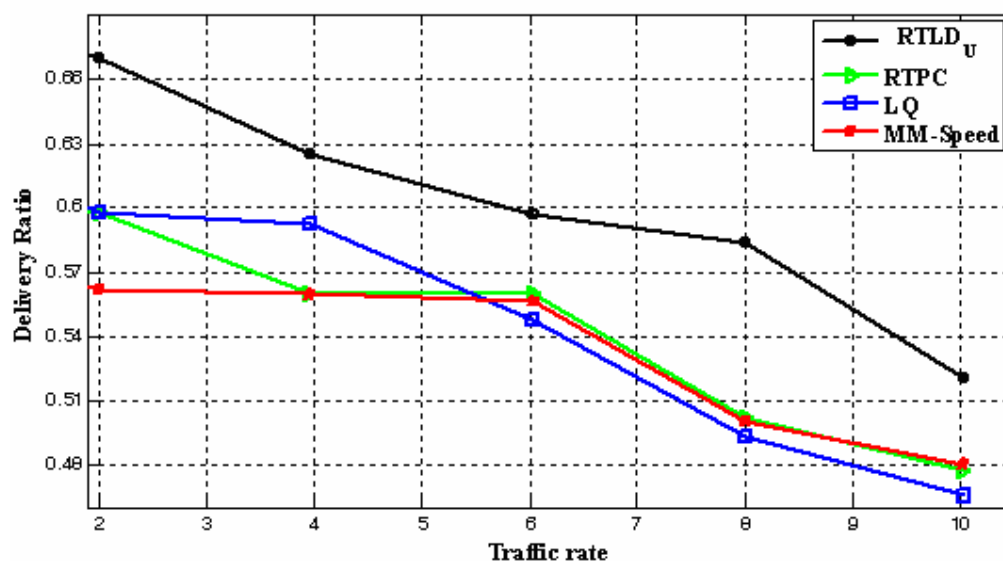
In this simulation, the Poisson traffic is used to evaluate the performance of RTLD for different traffic loads. It is interesting to note that Exponential On/Off generator in NS-2 has been configured as Poisson traffic by setting burst time to 0 and the variable rate to a very large value [86]. The burst time is the average “on” time for the generator, idle time is the average “off” time for the generator and the packet rate is the sending rate during “on” time, which equals 250 kb/s as stated in IEEE 802.15.4 and MICAZ specifications.

The traffic rate in this simulation is varied. During the burst time, the number of packet is set as a random number and the C++ coding guarantees that even if the burst time is zero, at least one packet is sent [86].

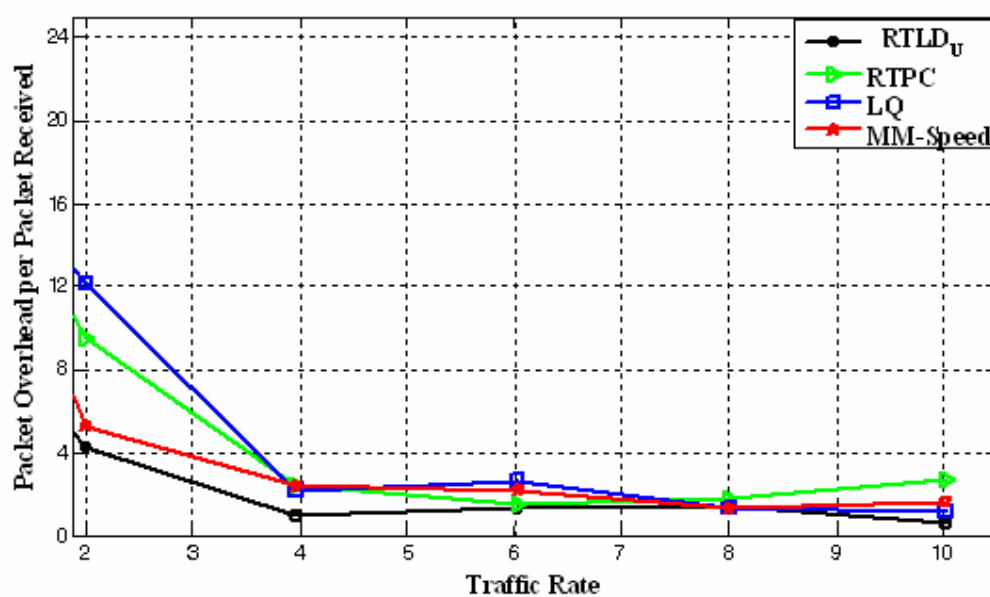
The simulation results in Figure 4.25(a) show that RTLD<sub>U</sub> provides the highest delivery ratio as the traffic rate varies. RTLD<sub>U</sub> experiences up to 6% more delivery ratio than the baseline routing protocols while the results in Figure 4.25(b)

show that  $RTLD_U$  spends 2% less packet overhead than the baseline routing protocols. This is due to the similar reasons stated in the section 4.4.2.

The results in Figure 4.25(c) shows that  $RTLD_U$  consumes 15% less power consumption compared to baseline routing protocols. This is primarily due to power management that switches off the transceiver (sleep state) for one period time if the sensor node does not receive any RTR packet. Therefore, the power consumption in  $RTLD_U$  is decreased.

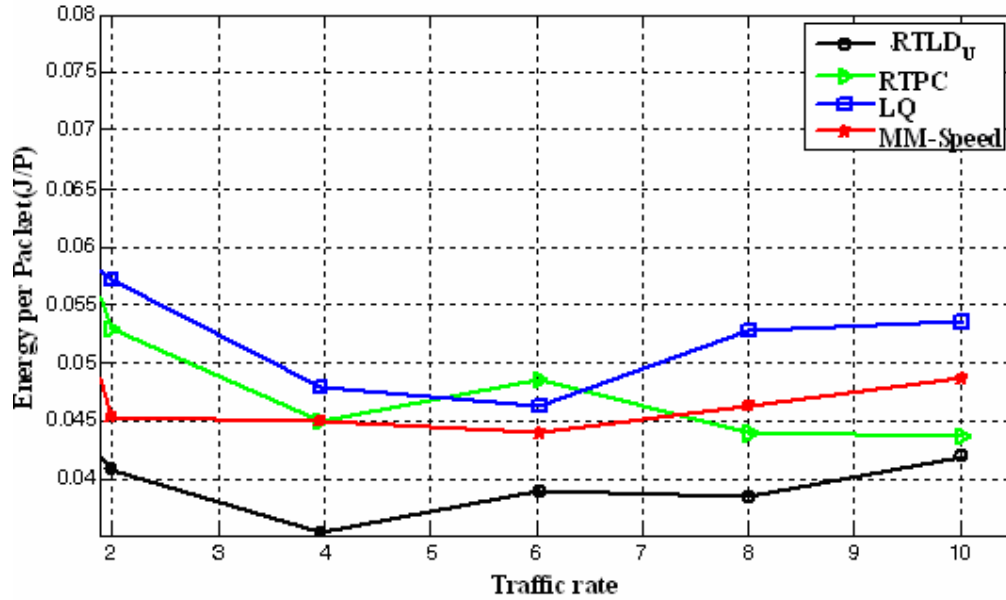


(a)





(b)

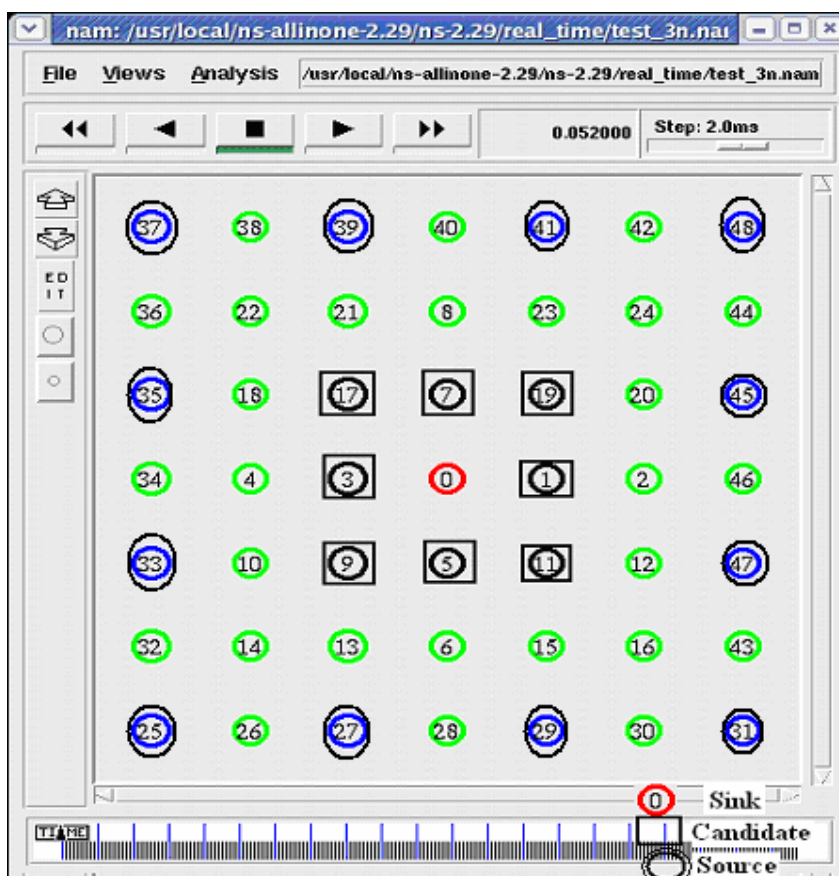


(c)

**Figure 4.23** Comparison between RTLD<sub>U</sub> and baseline routing using Poisson traffic at different traffic load a) Delivery ratio; b) Normalized packet overhead; and c) Normalized energy consumption.

#### 4.4.3 Load Distribution Using RTLD Routing

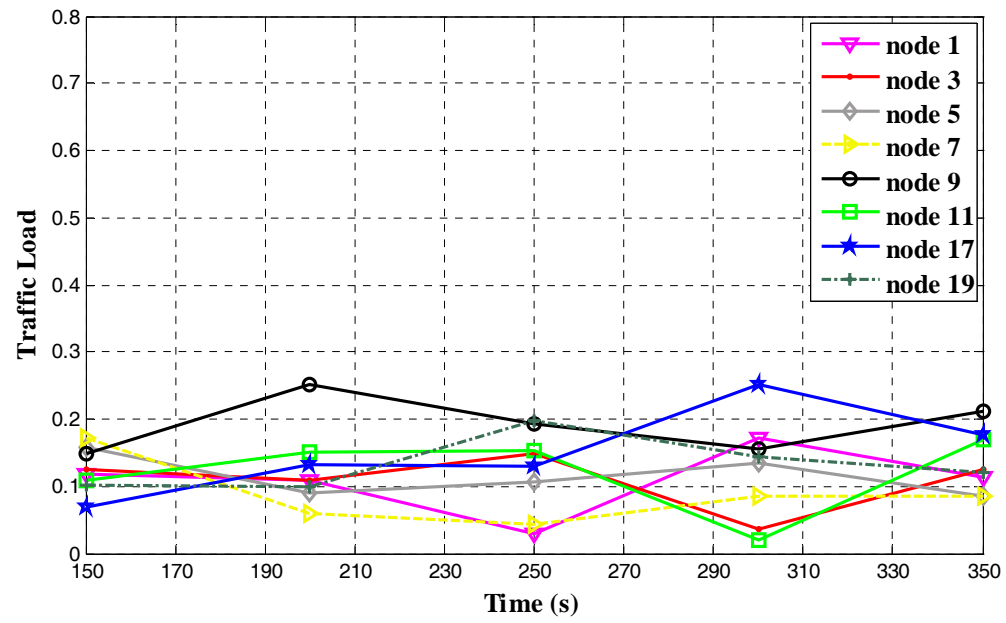
In order to show the effect of load distribution in RTLD routing protocol, an additional simulation is developed. In this simulation, 49 sensor nodes are deployed in grid topology with 12 data sources, one sink in the middle and 8 candidates as OF nodes as shown in Figure 4.26. The distance between a pair of sensor nodes is 15 m. The initial energy in the sensor node is 7.0 J. Many-to-one traffic pattern is used.



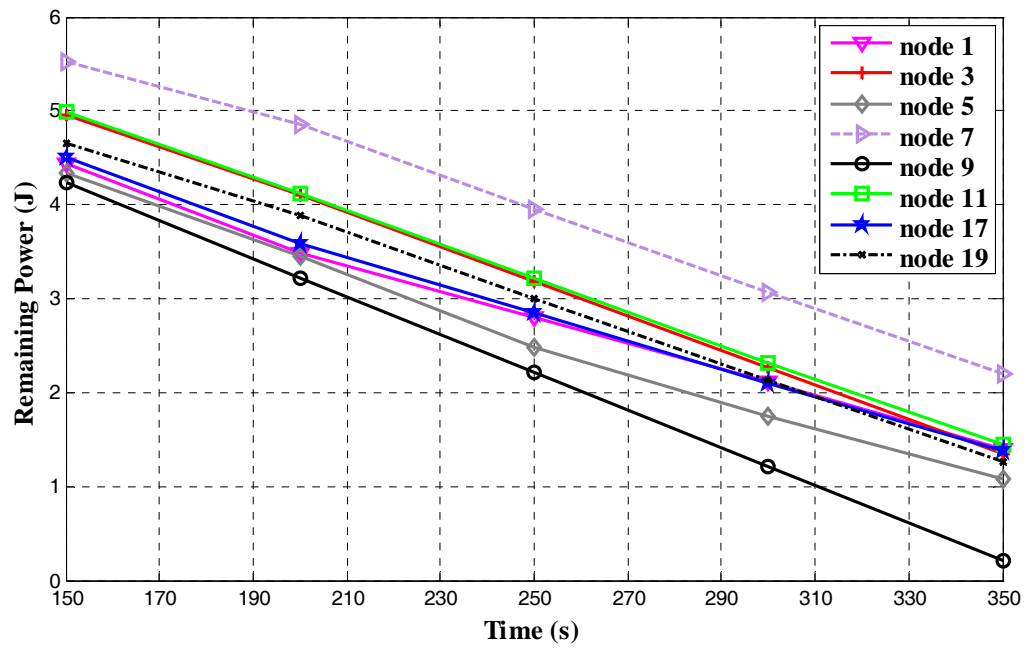
**Figure 4.24** Network simulation grid of load distribution

In this simulation, the load distribution is observed over a time limit of 350 s in WSN using RTLD. The end-to-end deadline and packet rate are fixed at 250 ms and 10 packet/s respectively. The simulation results in Figure 4.27(a) show that the traffic loads of 12 sources are distributed among OF nodes throughout the whole simulation time. However, nodes 9, 11, 17 and 19 relay more packets because they are nearer to the three sources for example node 9 is near to sources 33, 25 and 27.

Figure 4.27(b) shows that the OF nodes maintain similar trend of remaining power which correlates with the traffic load. The results show RTLD ensures load distribution as its forwarding strategy uses remaining power as one of the metric of selection next hop. This eventually prolongs the lifetime of the WSN as elaborated in the following section.



(a)



(b)

**Figure 4.25** Effect of load distribution; a) traffic load and b) remaining power.

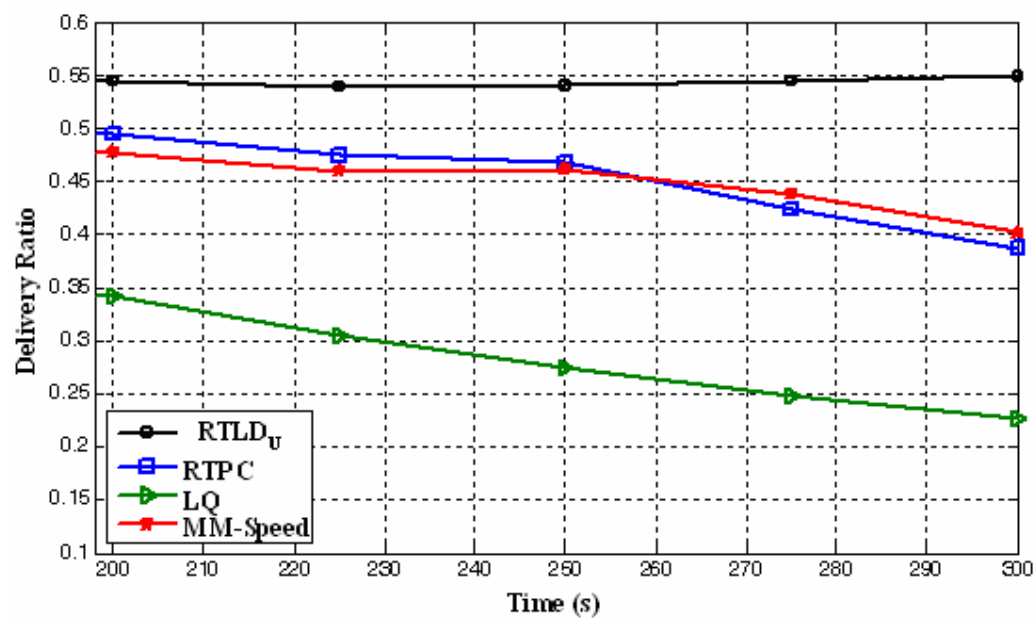
#### 4.4.4 Prolonging WSN Lifetime

Network lifetime measures the amount of time before the first node runs out of battery power [104]. Based on this definition of WSN lifetime, this section will analyze the influence of the remaining power on the performance of the WSN. End-to-end deadline and packet rate are fixed at 250 ms and 10 packet/s respectively. The simulation results in Figure 4.28(a) show that the delivery ratio of  $RTLD_U$  is higher by 5% up to 15% compared to the baseline routing protocols. Table 4.2 also shows WSN lifetime is prolonged by 16 % in  $RTLD$  compared to the baseline routing protocol. The baseline routing protocols suffer decreasing packet delivery ratio due to packet dropping over a long period of time. One major reason is that due to the baseline routing ignores spreading of the traffic load among neighbouring nodes, hence creating routing holes problem. The routing holes problem may also appear due to power termination in the forwarding candidate node. In contrast,  $RTLD_U$  distributes the load to forwarding candidates to overcome routing holes problem and hence, balancing the load among the neighbouring nodes and maintains the delivery ratio to a comparable level.

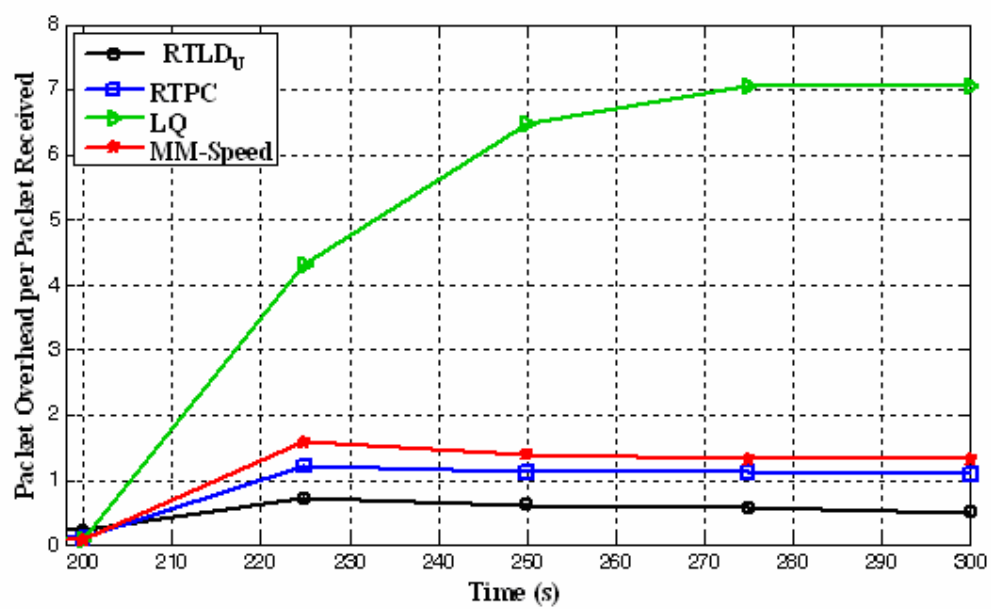
Figure 4.28 (b) shows that  $RTLD_U$  experiences the least packet overhead. Since baseline routing protocols are not capable of countering the routing hole problem, the nodes around the hole are required to send more packet overhead.  $RTLD_U$  consumes up to 5% less power compared to baseline protocols. The reduced power consumption is the consequence of sending and distributing the load throughout the neighbouring nodes.

**Table 4.2** Comparison WSN lifetime between  $RTLD$  and baseline routing protocols

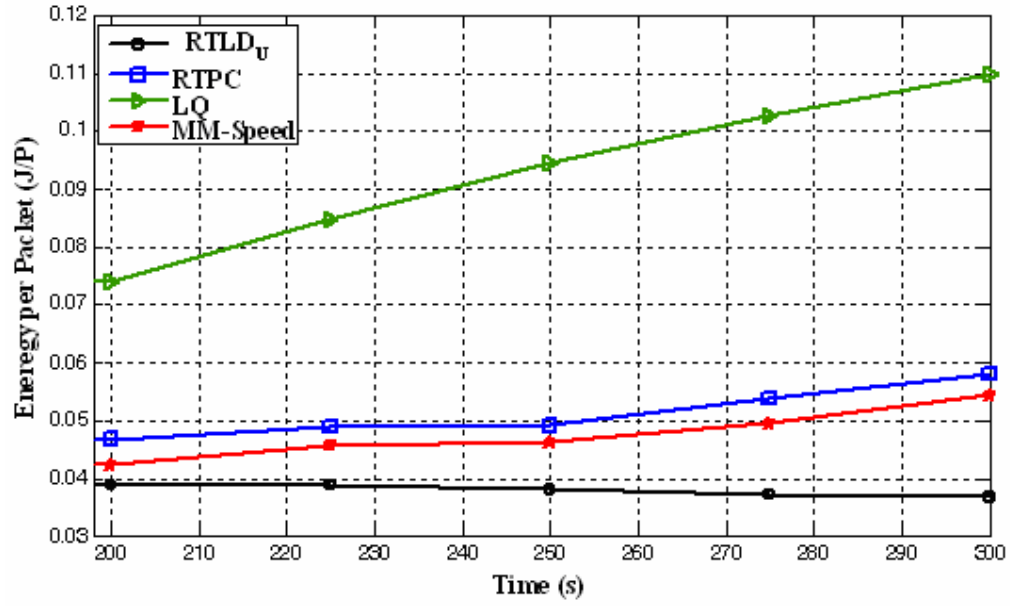
	<b>RTLD</b>	<b>RTPC</b>	<b>MM-Speed</b>	<b>LQ</b>
<b>Lifetime</b>	287.496	239.56	230.395	204.226
<b>Normalized Lifetime</b>	96%	80%	77%	68%



(a)



(b)



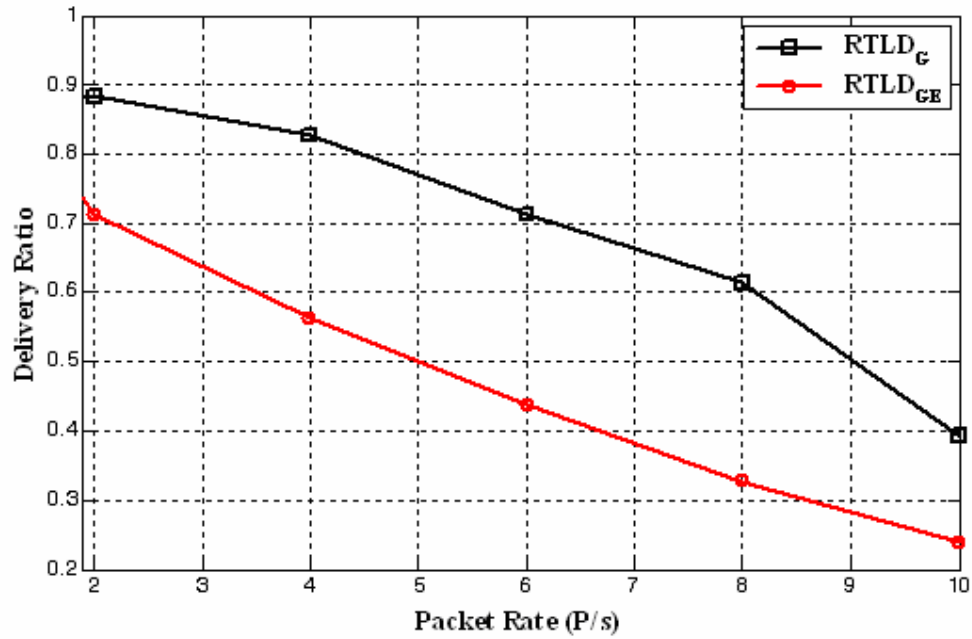
(c)

**Figure 4.26** Comparison prolonging lifetime between RTLD<sub>U</sub> and baseline routing at fixed packet rate a) Delivery ratio; b) Normalized packet overhead; and c) Normalized energy consumption.

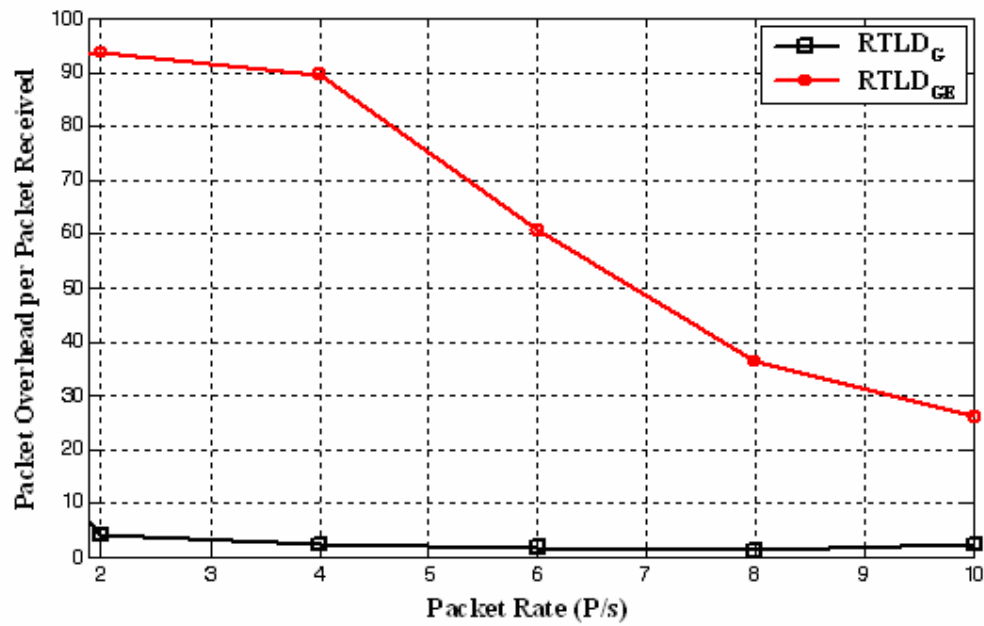
#### 4.4.5 Comparison between Geocast and Geodirection-cast Forwarding

In geocast forwarding (RTLD<sub>GE</sub>), the packets are broadcast to all nodes in a given geographic region in a natural way [49]. However, packets in RTLD<sub>G</sub> are broadcast similar to geocast in the initial hop only and subsequently, packets are forwarded (unicast) in the direction toward the destination. The simulation results in Figure 4.29(a) show that the delivery ratio decreases as the packet rate increases. In addition, RTLD<sub>G</sub> experiences higher delivery ratio by 15% to 28% compared to RTLD<sub>GE</sub>. This is largely due to smaller number of packets being broadcast in RTLD<sub>G</sub> than RTLD<sub>GE</sub>. Broadcasting of packets only occurs in the first hop. As mentioned earlier, the packet broadcasting affects the delivery ratio and power consumption due to collision and congestion in WSN [90].

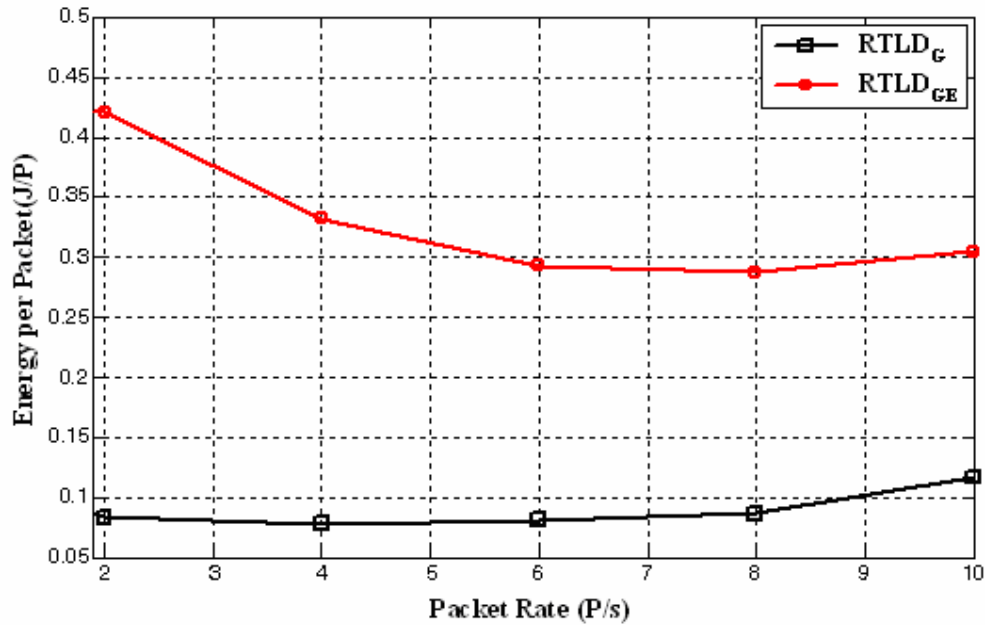
Figure 4.29(b) shows that  $RTLD_{GE}$  spends 12 times higher packet overhead than  $RTLD_G$  because the broadcasting in  $RTLD_G$  is limited to one-hop around the source while Figure 4.29(c) shows that  $RTLD_G$  consumes 6 times power compared to  $RTLD_{GD}$ . This is because the flooding of packets due to broadcast wastes control packet overhead and uses up battery power.



(a)



(b)



(c)

**Figure 4.27** Comparison between RTLD<sub>G</sub> and RTLD<sub>GE</sub> at different packet rate a) Delivery ratio; b) Normalized packet overhead; and c) Energy consumption.

## 4.5 Summary

This chapter presents the NS-2 simulation study of RTLD routing protocol in WSN. It also elaborates the optimization process of finding the best OF nodes. The findings show that there are four trials out of sixty-six trials that present good performance in terms of delivery ratio and power consumption. It also shows that RTLD experiences real-time forwarding within 250 ms.

RTLD routing protocol enhances the previous works by [5, 9, 31, 33] in order to achieve high delivery ratio, minimum control packet overhead and efficient power consumption. In general, the finding concludes that RTLD<sub>U</sub> provides high delivery ratio and spends less number of control packet overhead with comparable power consumption compared to the baseline routing protocols. RTLD<sub>G</sub> improves the throughput compared to RTLD<sub>U</sub> at low traffic loads as it allows forwarding of data



packets through multiple paths. However,  $RTLD_G$  consumes more power due to the original sources broadcasting data packets in the initial one-hop neighbour to allow more than one possible multi-path. The significant feature of RTLD is that it distributes the task of load forwarding to OF candidates in order to avoid packet dropping due to power termination and hence, prolongs the network lifetime. This chapter also shows that RTLD with feedback has a good performance and can solve the routing hole problem which provides more flexibility and fault tolerant features.

## **CHAPTER 5**

### **DEVELOPMENT OF RTLD TEST BED**

#### **5.1 Introduction**

The proposed RTLD routing protocol for WSN has been deliberately studied through simulation process in the previous chapter. The simulation results show that RTLD experiences higher delivery ratio, less power consumption and less packet overhead. This chapter discusses the development and implementation of RTLD test bed. RTLD test bed has been developed on a network using MICAZ and TELOSB radio sensor boards. MICAZ and TELOSB consist of low power transceiver based on CC2420 ChipCon chip [80] that employs IEEE 802.15.4 physical and MAC layers specifications. The multipurpose sensor board is attached to MICAZ to read Humidity/Temperature sensor reading, whereas TELOSB uses built-in Humidity/Temperature sensor.

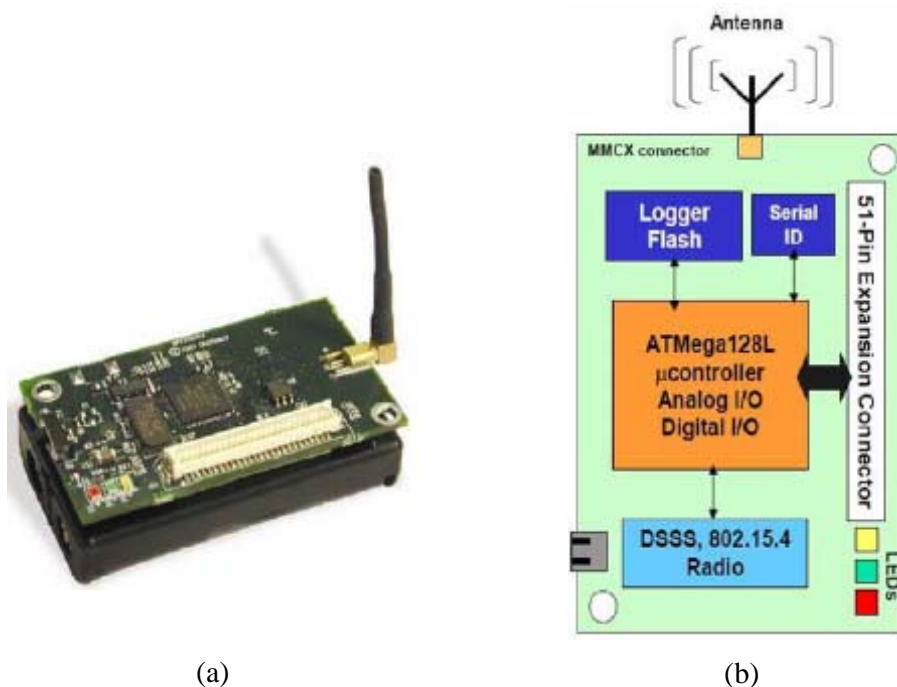
In order to avoid hardware error in sensor board, TinyOS simulator (TOSSIM) is used in this research to debug and test the proposed algorithms in a controlled and repeatable environment. The following sections elaborates the realization of RTLD in TOSSIM, the development of RTLD in MICAZ and TELOSB and the practical implementation of RTLD test bed. Finally, at the end of the chapter an example of real-time monitoring of temperature using RTLD in WSN is highlighted.

## **5.2 Development of WSN Test bed**

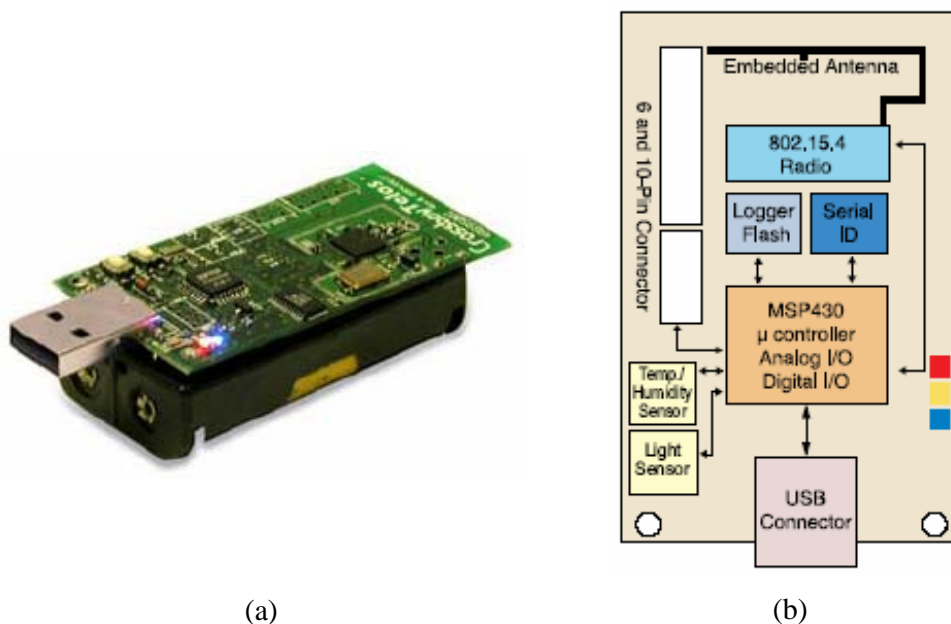
The test bed consists of two components: hardware component and software component. The hardware component consists of processor/radio sensor board, multifunction sensor board and programming board. The software component consist of TinyOS, TOSSIM, TinyViz, nesC programming language, Java programming language, avr-gcc and msp-gcc compilers, and serial forwarder GUI.

### **5.2.1 Hardware Components**

The test bed used MICAZ and TELOSB which are developed by Crossbow Technology Inc[24]. MICAZ and TELOSB are processor/radio sensor boards. They use the same 2.4GHz ISM band based on IEEE 802.15.4 radio standard. MICAZ and TELOSB provide the same functions. The differences between the two are memory capacity, the microcontroller type, and the connection to the external devices including sensors. TELOSB can be connected to a PC through USB port while MICAZ is linked to a PC through serial/USB programming board. TELOSB does not have On/Off switch but MICAZ has. The battery is plugged out in order to switch Off TELOSB. Figure 5.1 illustrates the MICAZ board and Figure 5.2 illustrates the TELOSB board. Appendix A describes the specifications of MICAZ and TELOSB.



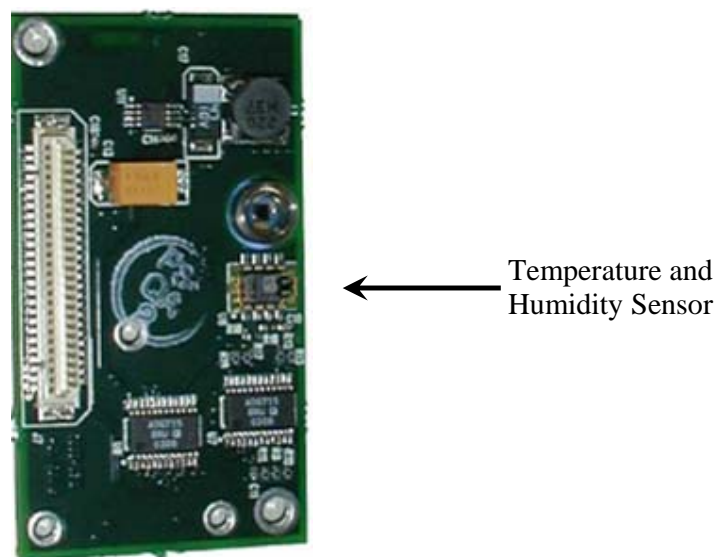
**Figure 5.1** MICAZ mote a) MICAZ board, b) MICAZ block diagram [85]



**Figure 5.2** TELOSB mote a) TELOSB board, b) TELOSB block diagram [85]

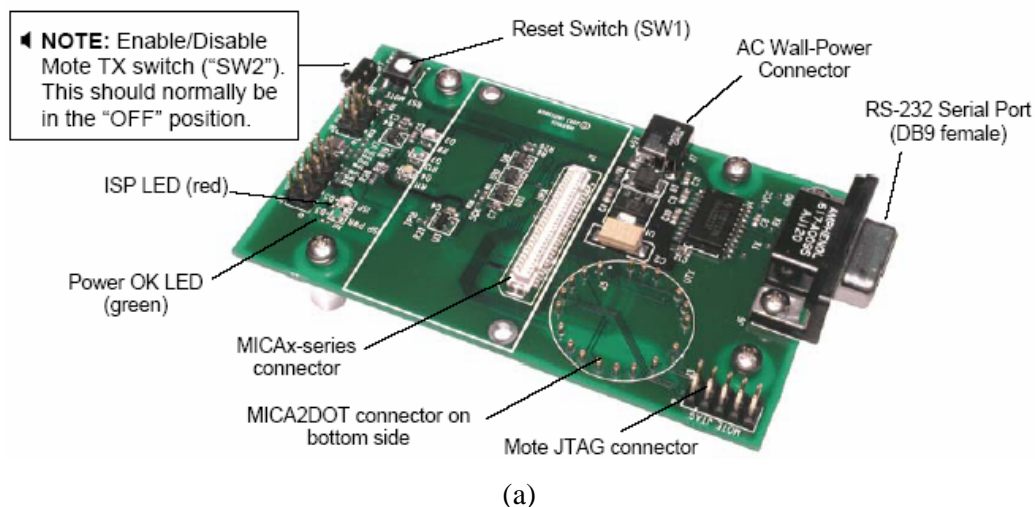
MTS 400CA sensor board is connected to MICAZ mote as a multipurpose sensor board. As shown in Figure 5.3, the sensor board consists of a cluster of five basic environmental sensors that include a single-chip humidity and temperature multi sensor module comprising a calibrated digital output. The chip has an internal

14-bit analog-to-digital converter and serial interface [91]. An analog-to-digital converter in the sensor does the conversion from humidity and temperature to digital units.



**Figure 5.3** Multipurpose sensor board [85]

A programming board MIB 510CA is used to program the motes with the desired applications. Figure 5.4 shows a block diagram of the MIB510 programming board. It has an RS-232 port, which is the programming communication link to a laptop or any other external device that holds the application programs. The MIB510 has an on-board in-system processor (ISP) to program the motes. The application code is downloaded into the ISP through the RS-232 serial port. Then, the ISP uploads the application code into the mote.





language, avr-gcc and msp-gcc compilers, Serial Forwarder GUI and surge application.

Cygwin is a Linux-like environment for Windows. It consists of two parts: the first part is a DLL (cygwin1.dll) which acts as a Linux API emulation layer and the second part is a collection of tools which provide Linux platform [92].

TinyOS is the most widely used operating system for WSNs. It was written in nesC [17], which is a high-level programming language that emulates the syntax and functionality of hardware description languages, and provides components for communication, thread coordination, and hardware abstraction. TinyOS does not contain a single network stack; rather application designers build their own stack by selecting among compatible components for, e.g., multi-hop routing [94].

TOSSIM is a discrete event simulator for TinyOS sensor networks. Instead of compiling a TinyOS application for a mote, users can compile it into the TOSSIM framework, which runs on a PC. This allows users to debug, test, and analyze algorithms in a controlled and repeatable environment. As TOSSIM runs on a PC, users can examine their TinyOS code using debuggers development tool [95].

TinyViz is a Java-based GUI that allows the user to visualize and control the simulation as it runs, inspecting debug messages, radio and UART packets, and so forth [96]. The simulation provides several mechanisms for interacting with the network; packet traffic can be monitored, packets can be statically or dynamically injected into the network.

NesC is a programming language for networked embedded systems that represent a new design space for application developers. It creates C executable code that provides all the low-level features necessary for accessing hardware resources [17]. NesC's contribution is to support the special needs of this domain by exposing a programming model that incorporates event-driven execution, a flexible concurrency model, and component-oriented application design. Restrictions on the programming model allow the nesC compiler to perform the whole program analyses, including data race detection which improves reliability [17].

Java programming is used to enable connection between the motes and the PC. Java Communication API package contains supports for serial and parallel ports on Windows PCs. The package needs to be installed before it can run Java program on PC and communicate with attached mote. Java application can be selected as an option TinyOS installation.

The Serial Forwarder GUI is a program written in Java, and it is used to read data packet from a computer's serial port and forward it over a server port connection, so that other programs can communicate with the sensor network via a sensor network gateway. Serial Forwarder does not display the data packet itself, but rather updates the packet counters in the lower-right hand corner of the window. Once running, the serial forwarder listens for network client connections on a given TCP port (9001 is the default for MIB510CA), and simply forwards TinyOS messages from the serial port to the network client connection, and vice versa.

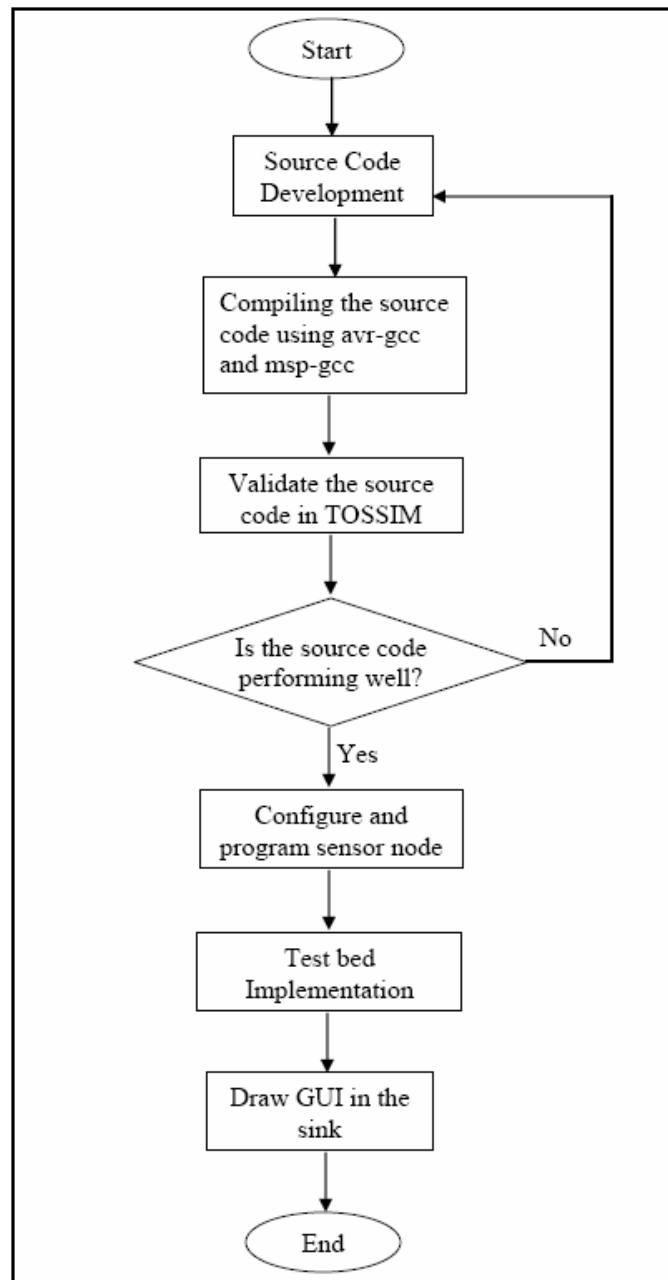
Avr-gcc is used to compile the application programming code for ATmega128L microcontroller inside MICAZ motes and msp-gcc is used to compile the application programming code for MSP430 microcontroller inside TELOSB motes. Both avr-gcc and msp-gcc compilers are installed with TinyOS. It is interesting to note that the programming code that is written for MICAZ does not necessary work with TELOSB. This is mainly due to difference in the mathematical function for each microcontroller.

### **5.2.3 Development RTLD Routing Protocol in Test bed**

The test bed is divided into two parts development of RTLD routing protocol and TinyOS application. TinyOS application performs periodic sensor reading and delivers the data packet to the sink using RTLD routing protocol. RTLD and TinyOS application have been programmed based on nesC programming language. The source codes of functional modules had been written based on the algorithms that were explained in the previous chapter.



Figure 5.5 shows the flow chart diagram of RTLD routing protocol in the test bed. In order to check the source codes of RTLD, avr-gcc and msp-gcc have been used to compile and to check the syntax error of the source codes. TOSSIM has been used to validate the functional modules of RTLD routing protocol and thus ensuring error-free hardware and energy saving.



**Figure 5.5** Flow chart diagram of development RTLD routing protocol in test bed

It is interesting to note that TOSSIM is mainly used to test and analyze the functional modules of RTLD routing protocol before the source code is implemented in the real test bed. After RTLD routing protocol has been validated in TOSSIM, the sensor nodes are programmed with more stable source code of RTLD routing protocol. Finally, the GUI is developed using Java programming language to show the communication between sensor nodes in WSN. Appendix D shows the source codes of RTLD routing protocol in the test bed.

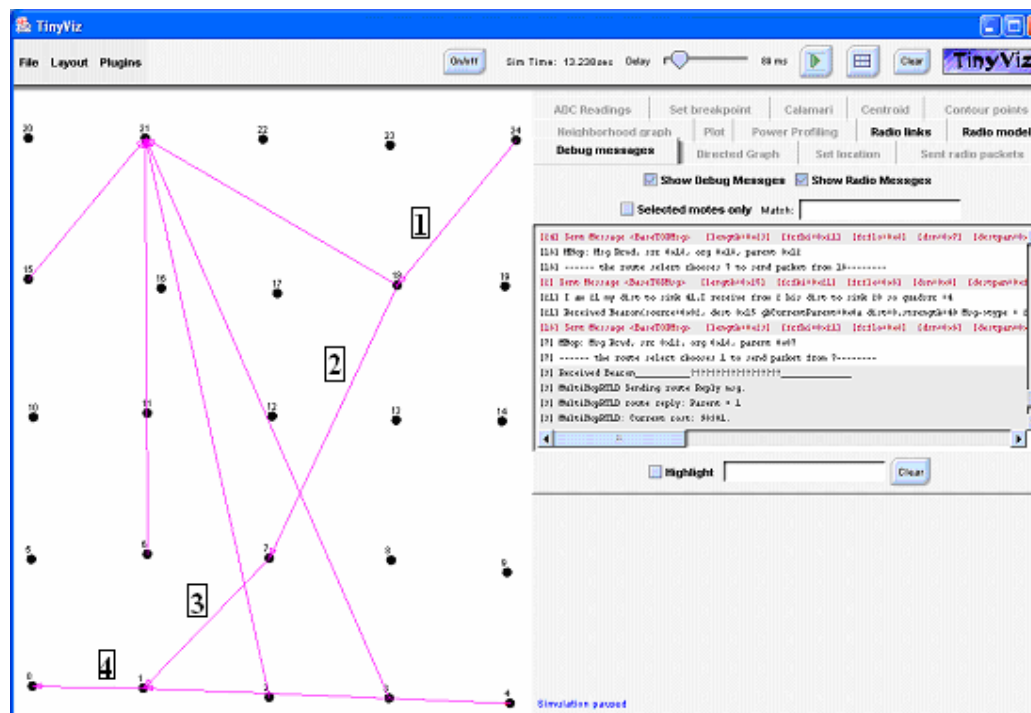
### **5.2.3.1 Execution of RTLD Routing Protocol in TOSSIM**

TOSSIM has the ability to check and change the programming code before it is uploaded into the sensor node. It cooperates with TinyViz GUI to visualize and control the simulation. The source codes of RTLD routing protocol in TOSSIM are exactly similar to the source codes of RTLD in the test bed. Beside, the proposed research used the main options in TOSSIM which are debugging message, radio link, power profile and radio model. The debugging message option is used to monitor all the routing message, neighbour table and determination of OF nodes. TOSSIM are configured to specify; the sensor node type (MICAZ or TELOSB), application that will be uploaded to node, routing algorithm, MAC and physical layers. The RTLD routing protocol validation in TOSSIM has been implemented using 25 sensor nodes. The sink is node 0 and the data packet source is node 24. Detailed explanation of the RTLD functional modules validated in TOSSIM as are in the following sections.

- **Routing Management Operation**

The routing management operation has been validated in TOSSIM. The OF node has been determined and the forwarding mechanism has been selected. Figure 5.6 shows the TinyViz GUI of 25 sensor nodes. Node 24 sends RTR broadcast message to its neighbour in order to implement neighbour discovery. RTR reply sends back to node 24. In this scenario, node 4 sends its data packet to the sink through node 3, 2 and 1. Figure 5.3 shows the forwarding trip from node 24 to the sink. Node 24 selects node 18, node 7 and finally node 1 before reaching the sink.

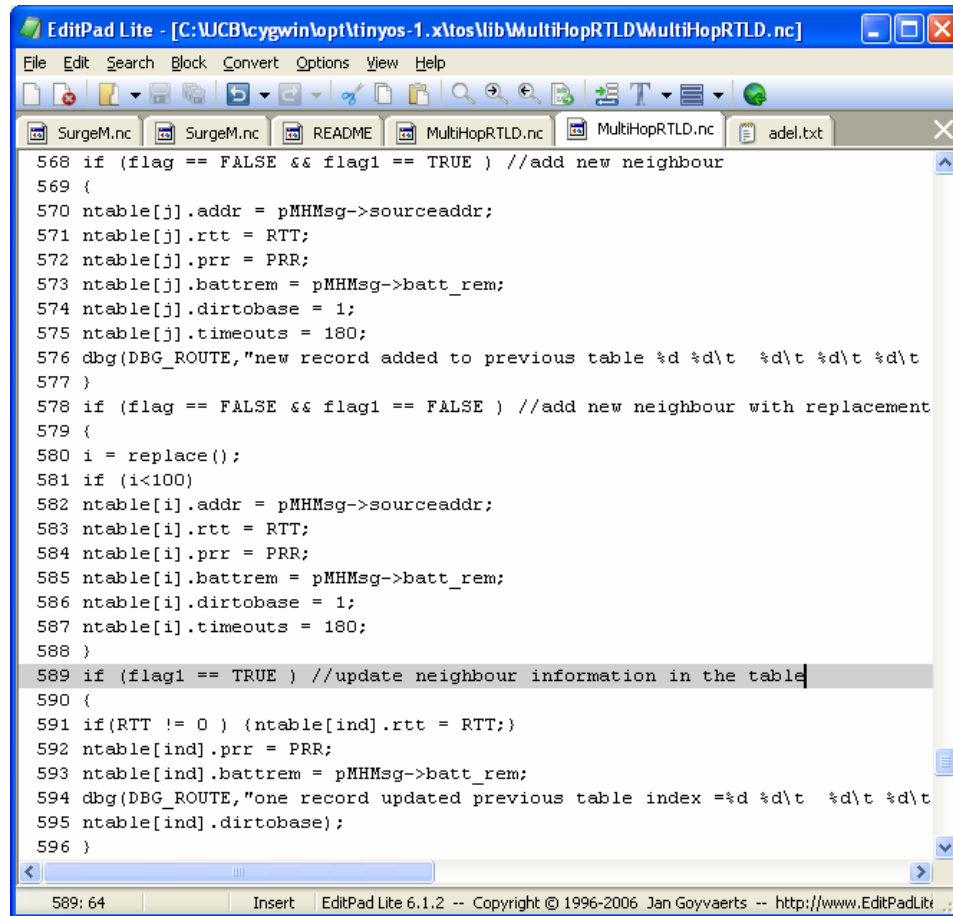
Then node 7 selects node 1 as the next hop to the sink. Finally, node 1 forwards data packet to the sink.



**Figure 5.6** Forwarding trip from source node 24 to the sink 0

- **Neighbourhood Management Operation**

Figure 5.7 shows the source code of the neighbourhood management operation which includes adding new neighbour, updating the existing record and replenishing it with the optimal neighbour. The adding new neighbour function performs inserting new neighbour information into the neighbour table while updating function performs by modifying the expiry record. The replacing function performs by swapping between the new record and the existing record in the neighbour table. If the neighbour table is not full, the new neighbour record is inserted to the neighbour table. In case the neighbour table is full, it will immediately replace the neighbour with least met criteria. The neighbour table is normally updated every 180 seconds.



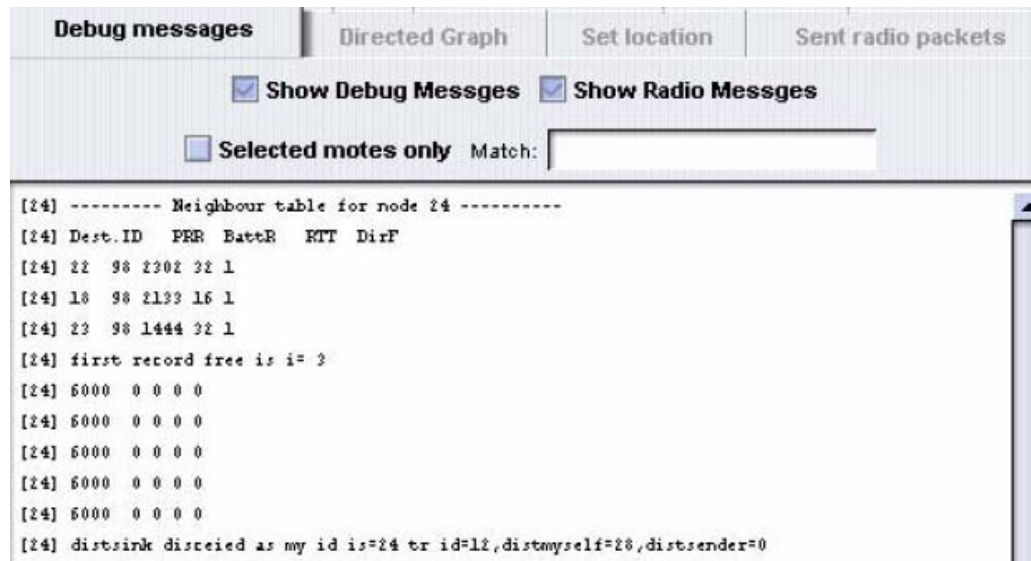
```

EditPad Lite - [C:\UCB\cygwin\opt\Tinyos-1.x\lib\MultiHopRTLD\MultiHopRTLD.nc]
File Edit Search Block Convert Options View Help
SurgeM.nc SurgeM.nc README MultiHopRTLD.nc MultiHopRTLD.nc adel.txt
568 if (flag == FALSE && flag1 == TRUE ) //add new neighbour
569 {
570 ntable[j].addr = pMHMsg->sourceaddr;
571 ntable[j].rtt = RTT;
572 ntable[j].prrr = PRR;
573 ntable[j].battrem = pMHMsg->batt_rem;
574 ntable[j].dirtobase = 1;
575 ntable[j].timeouts = 180;
576 dbg(DBG_ROUTE,"new record added to previous table %d %d\t %d\t %d\t %d\t
577 )
578 if (flag == FALSE && flag1 == FALSE ) //add new neighbour with replacement
579 {
580 i = replace();
581 if (i<100)
582 ntable[i].addr = pMHMsg->sourceaddr;
583 ntable[i].rtt = RTT;
584 ntable[i].prrr = PRR;
585 ntable[i].battrem = pMHMsg->batt_rem;
586 ntable[i].dirtobase = 1;
587 ntable[i].timeouts = 180;
588 }
589 if (flag1 == TRUE ) //update neighbour information in the table
590 {
591 if(RTT != 0 ) {ntable[ind].rtt = RTT;}
592 ntable[ind].prrr = PRR;
593 ntable[ind].battrem = pMHMsg->batt_rem;
594 dbg(DBG_ROUTE,"one record updated previous table index =%d %d\t %d\t %d\t
595 ntable[ind].dirtobase);
596 }
589: 64 Insert EditPad Lite 6.1.2 -- Copyright © 1996-2006 Jan Goyvaerts -- http://www.EditPadLite

```

**Figure 5.7** Neighbourhood management source code

Figure 5.8 shows the neighbour table of sensor node 24. It also shows the contents of the table such as next hop, PRR, remaining battery and the one hop delay. In this scenario, node 24 has three neighbours 18, 23 and 22 in order to forward the data packet to the sink. As can be seen in Figure 5.8, node 24 selects node 18 as OF due to the fact that node 18 has the optimal forwarding metrics in this scenario.



**Figure 5.8** Neighbour table in TOSSIM

- **Power Management operation**

The power management of RTLD is explained in chapters 3 and 4. Figure 5.9 shows the power usage profile of 25 sensor nodes that are deployed in grid topology. It also shows the power consumption of all units in sensor node such as central processing unit (CPU), radio unit and sensor unit. The total power in Figure 5.8 is measured in mJ. In this scenario, node 18 consumes 36 mJ more power than node 23 because it forwards more data packet to the sink.

The power management has been used to adjust the transceiver state in order to minimize the power usage. Figure 5.10 shows the transceiver power state for the sensor node such as ON (idle), TX (transmit), RX (receive) and OFF (sleep). Mark 1 shows that node 7 is in the transmission state (TX) at simulation time 37921903.

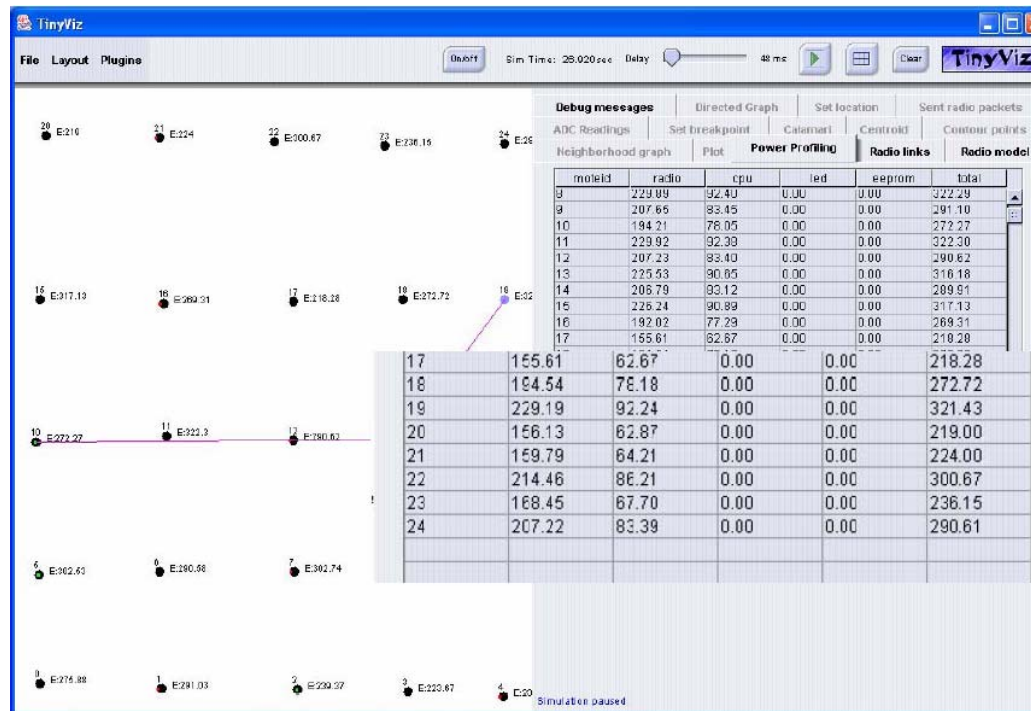


Figure 5.9 Power usage of sensor node in TOSSIM

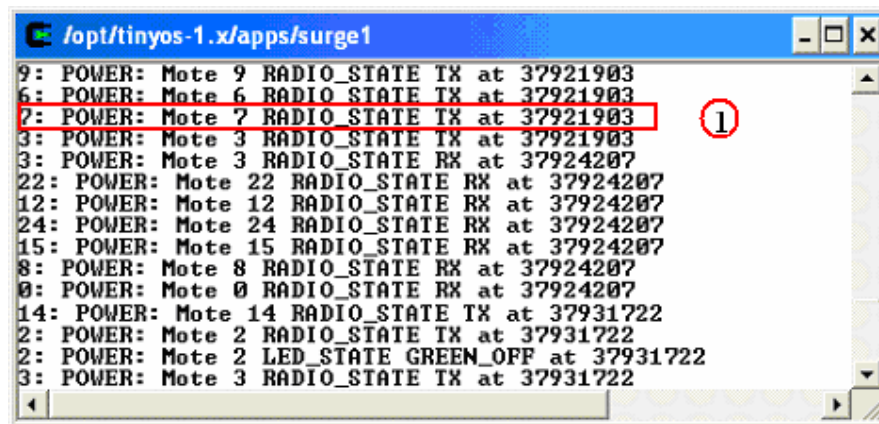


Figure 5.10 Transceiver states of sensor nodes in TOSSIM

### 5.2.3.2 Configuration and Programming Sensor Node

The transmission channel and power level are configured based on IEEE 802.15.4 physical layer. The MICAZ radio transmitter can be tuned within the IEEE 802.15.4 channels that are numbered from 11 (2.405 GHz) to 26 (2.480 GHz) each

separated by 5 MHz. The channel can be selected at run-time in TinyOS using *CC2420Control.TunePreset(uint8\_t chnl)* function. By default channel 11 (2480 MHz) is selected. Beside that, RF transmission power is programmable from 0 dBm (1 mW) to -25dBm. Lower transmission power can be advantageous by reducing interference and dropping radio power consumption from 17.5 mA at full power to 8.5 mA at lowest power. RF transmit power is controlled using *CC2420Control.SetRFPower(uint8\_t power)* function where power is an 8-bit code selected from Table A.4 in appendix A.

When the program is uploaded to the sensor node, the node ID and the uploading port are important to success the programming process. In order to compile MICAZ or TELOSB mote for real test bed application, the following instruction is typed in cygwin console at the directory of desired application:

```
$ make micaz      (for MICAZ mote)
$ make telosb     (for TELOSB mote)
```

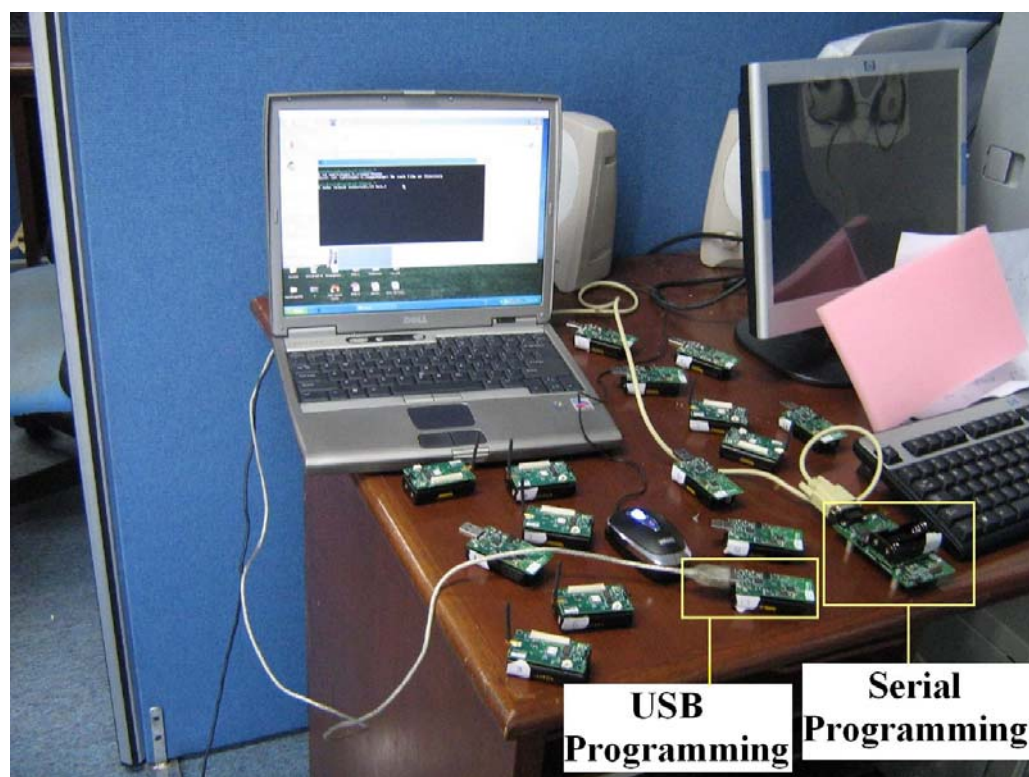
In order to upload the execution code inside the mote, the following instruction is typed in the cygwin console at the directory of desired application:

```
$ make micaz reinstall.0 mib510,/dev/ttyS0  (for MICAZ mote)
$ make telosb reinstall,0 bsl, COM3          (for TELOSB mote)
```

The uploading command for MICAZ means the execution code uploaded with node ID is 0, programming board is MIB510CA and the port number is COM1(/dev/ttyS0). However, the uploading command for TELOSB means the execution code uploaded with node ID is 0 through USB port as a virtual COM port (COM3). In order to identify the COM port that TELOSB is used, the motelist command is typed in the cygwin console. This command will find the COM port of TELOSB and brings the virtual COM number (COM4 in the previous command). The Boot Strap Loader (bsl) is responsible for loading the execution program to TELOSB.

### 5.3 Experimental Results of RTLD Routing

The RTLD routing protocol has been realized in real test bed using 25 sensor nodes (10 TELOSB and 15 MICAZ). Figure 5.13 shows the picture of sensor nodes configuration and code uploading into the sensor through serial programming board for MICAZ and USB port for TELOSB. Table 5.1 shows the code size of RTLD in MICAZ and TELOSB. The code size of RTLD routing protocol in the MICAZ is 25912 bytes of flash memory and 1896 bytes of RAM. However, the code size of RTLD routing protocol with enhanced security in the MICAZ is 26172 bytes of flash memory and 1896 bytes of RAM. The code size of RTLD routing protocol in the TELOSB is 32212 bytes of flash memory and 1207 bytes of RAM. However, the code size of RTLD with security in the TELOSB is 32414 bytes of flash memory and 1207 bytes of RAM. The code size is higher in TELOSB because its microcontroller and code compiler are different. Therefore, RTLD routing protocol is lightweight mechanisms, which can applied for different type of radio sensor board with different platform.



**Figure 5.11** Programming sensor node



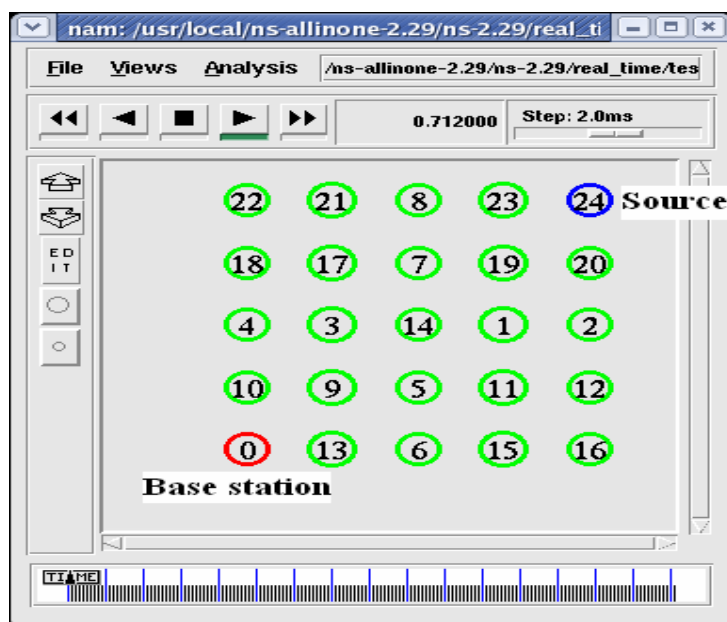
**Table 5.1** Coding size of RTLD and enhanced RTLD.

Comparison	MICAZ		TELOSB		Difference in Byte
	Flash memory	RAM	Flash memory	RAM	
<b>RTLD</b>	25912	1896	32212	1207	5611

### 5.3.1 Test Bed Network

The running of RTLD routing protocol in WSN test bed has been verified. The test bed performance in term of packet delivery ratio and average packet delay from the source to the destination are analysed. The results are compared with the simulation output.

Many-to-one traffic pattern is used in RTLD routing protocol in the case of unicast forwarding mechanism. One-to-many traffic pattern is used in the geodirection-cast forwarding mechanism. In this work, 25 nodes are distributed in a 40m x 40m region as shown in Figures 5.14 and 5.15. Node numbered as 24 is the source node and node 0 is the sink. To increase the hop count between the source and the sink, we select the sink at the left down corner of the grid and the source at the right up corner. The traffic is CBR and locations of all nodes are known.

**Figure 5.12** Network simulation grid



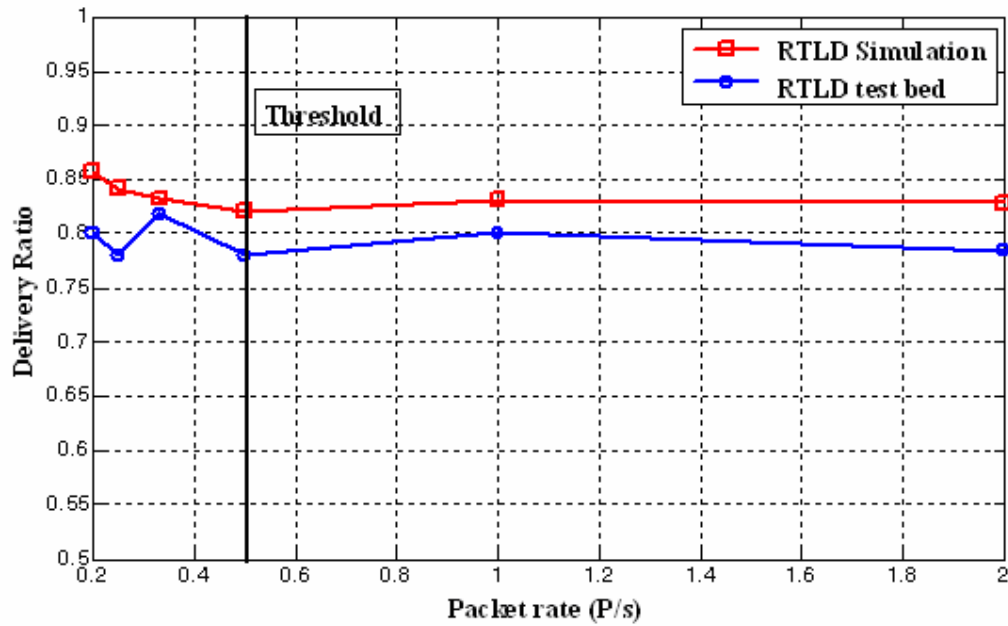
**Figure 5.13** Network test bed field

### 5.3.2 Results of RTLD Routing Protocol in Test Bed

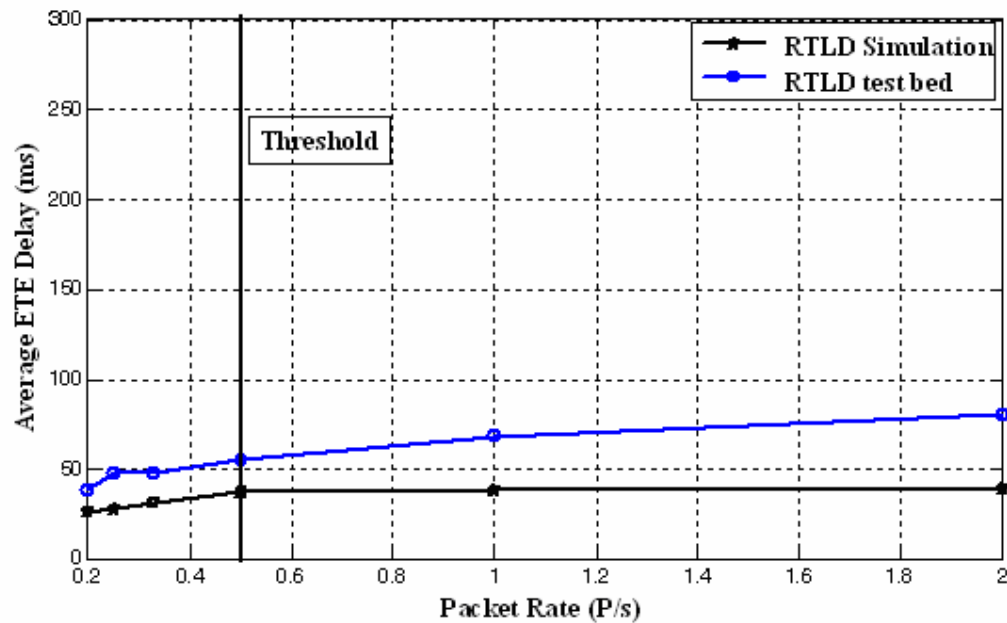
The network in the test bed has been configured similar to the network in the simulation study. In real test bed, end-to-end deadline and the experiment time were fixed at 250 ms and 100s respectively. The traffic load is varied from 0.2 to 2 packet/s to emulate low data rate for IEEE 802.15.4. The results in Figure 5.16(a) show that RTLD routing protocol in the simulation environment experiences slightly higher delivery ratio (about 5%) compared to the real test bed implementation. This may be due to the propagation model in the simulation differs from the real test bed environment. In practice, many parameters in the propagation model affect the signal strength including fading, reflection, diffraction and interference. In addition, it has been recommended by Crossbow Technology Inc. that the threshold packet rate for MICAZ and TELOSB should be set to 0.5 packet/s for multi-hop communication because higher packet rates can lead to congestion and or overflow of the communication queue [24]. This is applicable to Figures 5.18 and 5.19 as well.

Figure 5.16 (b) shows that the end-to-end delay in the real test bed is higher compared to the simulation study. The delay is largely due to the processing delay caused by the slow microprocessor in MICAZ and TELOSB compared to personal computer processing in the simulation. The microprocessor in MICAZ and TELOSB runs at 8MHz while the processor in simulation runs at 1700 MHz which is more the 200 times faster. In addition, the delay can be due to unreliable communication links

in wireless networks. The link failures cause retransmissions of the packet at the MAC layer which increase the average delay. Nevertheless, the end-to-end delay in the test bed is below the end-to-end deadline limit which is 250 ms.



(a)



(b)

**Figure 5.14** Performance of RTLD test bed and simulation at different packet rate: (a) Delivery ratio; and (b) Average ETE delay.

Figure 5.17 shows the main fields of packet received in the test bed of RTLD routing protocol. In this figure, mark 1 shows that the packet travels 4 hops and takes 112 ms end-to-end delay between the sink 0 and the source 24. Mark 2 shows that the packet traverses 3 hops and takes 48 ms end-to-end delay from source 24 to the sink 0. In general, RTLD experiences real-time communication for WSN and can forward the packet within very short time (less than 250 ms).

```

EditPad Lite - [E:\PHD writing\program code\test_bed_result\test_25_sensor_no...
File Edit Search Block Convert Options View Help
endtenddelay.txt RSSIresult_2.txt adel_p_2s(good).txt adel_5.txt
At Evening Time
-----
Message <MultihopMsg>
[sourceaddr=0x0]
[originaddr=0x18]
[seqno=0x122]
[originseqno=0xa7]
[hopcount=0x4]
[batt_rem=0xb7f]
[deadline=0x8a]
[data=0x0 0x0 0x37 0x1a 0x11 0x0 0x9e 0x0 0x0 0x0 temperature=27.36780C
Remaining_Battery=2.94300 V Time is = 1185207214975 s
-----
At Noon Time
-----
Message <MultihopMsg>
[sourceaddr=0x0]
[originaddr=0x18]
[seqno=0x154]
[originseqno=0x28]
[hopcount=0x3]
[batt_rem=0xb65]
[deadline=0xca]
[data=0x0 0x0 0xcc 0x1b 0x6 0x0 0xa 0x0 0x0 0x0 temperature=31.33680C
Remaining_Battery=2.91700 V Time is = 1183219606343 s
7274: 2 Insert EditPad Lite 6.1.2 -- Copyright © 1996-2006 Jan Goyvaerts -- http://www.E

```

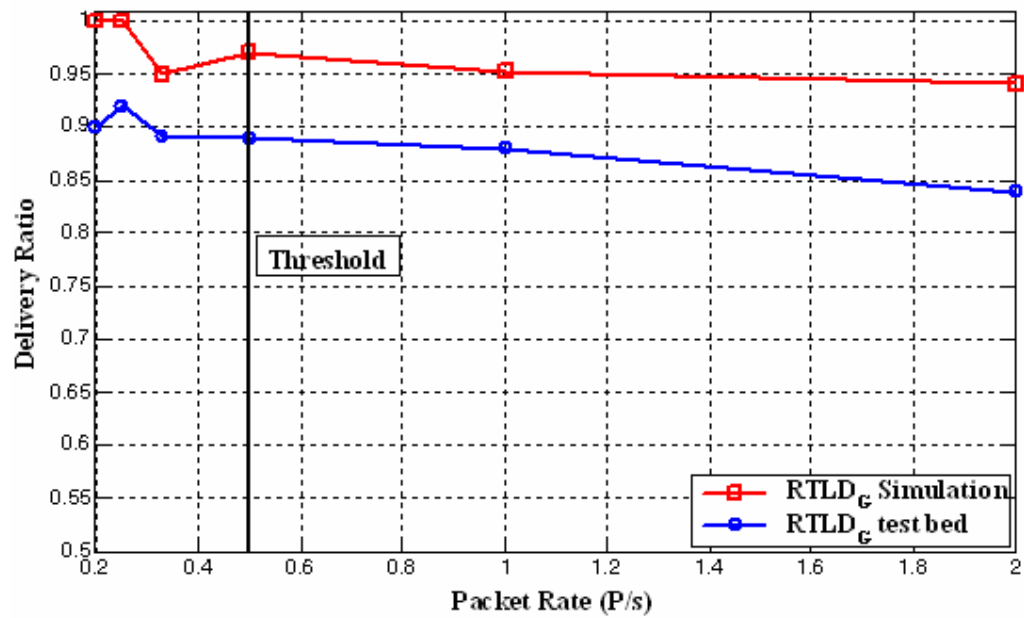
**Figure 5.15** Packet receiving in test bed.

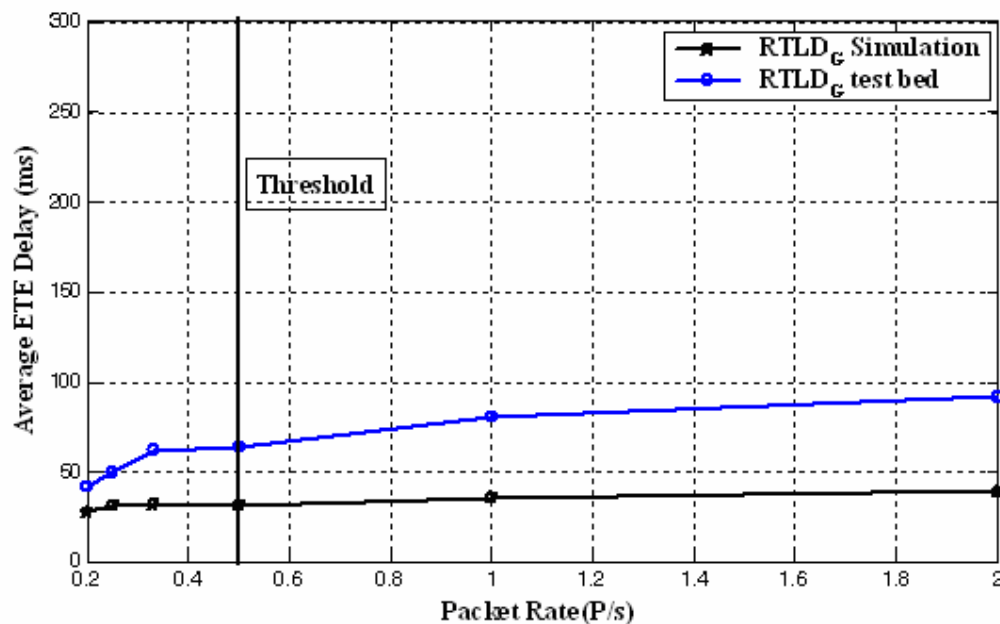
### 5.3.4 Geodirection-cast Forwarding in Test Bed

The results in Figure 5.19(a) show the geodirection-cast forwarding mechanism in both simulation and test bed enhance the throughput 10 % higher than

unicast forwarding mechanism due to the multi-path forwarding. This is extremely important in the real-time communication. However, this enhancement is achieved at the expense of more power consumption. In addition, the results show the simulation environment experiences higher delivery ratio by 7.5% compared to test bed as the packet rate is varied. This is primarily due to the propagation model is affected by unpredictable parameters such as fading, reflection, diffraction and interference.

Figure 5.19(b) shows that the average delay in the test is higher than the simulation. This is mainly due to the similar reasons in section 5.3.2.





**Figure 5.16** Performance of geodirection-cast in the test bed and simulation at different packet rate a) Delivery ratio; and b) Average ETE delay

#### 5.4 Monitoring Ambient of Temperature in WSN

This section presents the TinyOS temperature application that interacts with RTLD routing protocol. RTLD routing protocol has been used in WSN to monitor temperature of an environment. The network is developed using 25 sensor nodes: 10 sensor nodes with temperature sensor and 15 sensor nodes are intermediate nodes. Figure 5.20 shows the GUI of sensor nodes that are distributed in the test bed area. In this figure, the circle around a node denotes node with temperature sensor. The lines between sensor nodes show the communication path and number of hops between each sensor node and the sink. The temperature reading is forwarded in real-time to the sink. The update of temperature data at the sink depends greatly on configuration of the transmission interval at the source node. It can be seen from Figure 5.20 that the temperature monitoring in WSN works well using RTLD routing protocol. The GUI displays the temperature data on line and stores these data for further analysis. It is interesting to note that the temperature monitoring application is an example to prove the successful of RTLD routing protocol. However, another application such

as humidity or lightening monitoring also can be used if the sensor devices are attached to the radio sensor board.

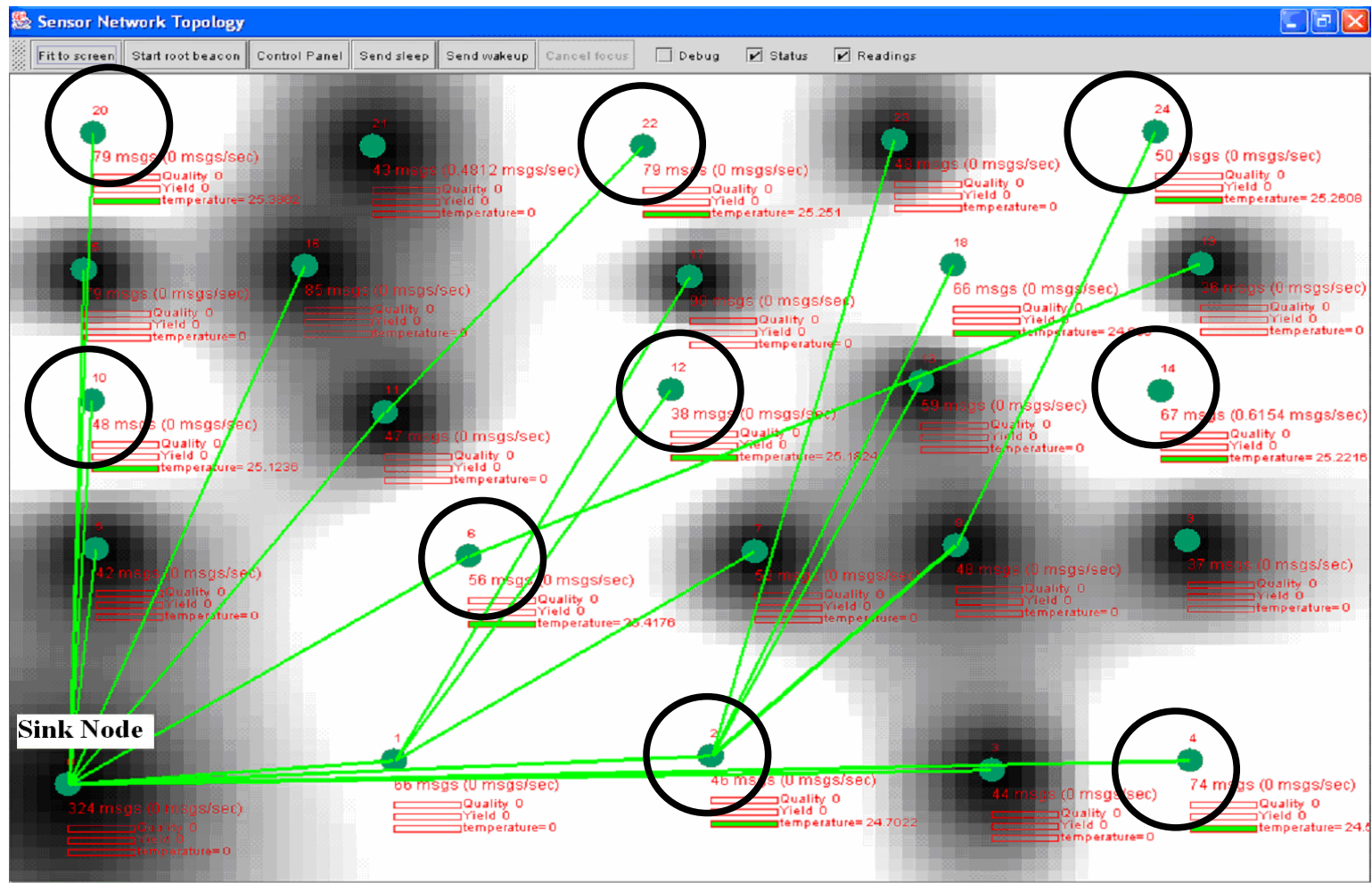


Figure 5.17 GUI of sensor nodes with on line displaying temperature



## 5.5 Summary

This chapter presents the development of the real test bed of RTLD routing protocol. The test bed consists of 25 sensor nodes (10 TELOSB and 15 MICAz), MIB 510CA programming board and multipurpose sensor board. TOSSIM was used to compile RTLD routing protocol on the PC. In order to save time and avoid hardware problem, TOSSIM was used to debug, test, and analyze algorithms before the codes is uploaded into the mote. The test bed results of RTLD routing protocol experience slightly degraded the performance compared to the simulation study. Nevertheless, The WSN test bed with RTLD routing protocol perform well within the end-to-end deadline. Simulation based RTLD experiences 5% higher delivery ratio compared to the test bed based. Geodirection-cast in the simulation study experiences 7.5% higher delivery ratio compared to the test bed. Moreover, the average delay in simulation study is lower than in the test bed based, this due to the fact that the processor of the PC used in the simulation is faster than the microcontroller processor in MICAz and TELOSB.

## **CHAPTER 6**

### **CONCLUSION**

#### **6.1 Introduction**

In this research, a novel RTLD routing protocol for WSN has been developed and verified experimentally in real test bed. The proposed routing protocol consists of five functional modules that include location management, power management, neighbourhood management, and routing management. These modules collaborate with each other to provide real-time routing protocol that distributes load among sensor nodes.

The RTLD routing protocol selects the optimal nodes to forward packets to the next hop neighbour. An optimization process based on exhaustive search has been performed to determine the OF equation. The OF equation takes into account of the physical layer conditions in the form of signal strength, packet timestamp and battery power level. These physical layer parameters are then transformed into forwarding metrics in the form of PRR, packet velocity and remaining power level. Specifically, the chosen optimal nodes rely on the link quality of the hop, the delay per hop and the remaining battery level of the forwarding nodes. Since forwarding nodes with the best link quality are chosen, the network improves the data throughput in terms packet delivery ratio. By choosing the forwarding nodes with the minimum delay limit, the

network ensures real-time packet transfer in the WSN. Additionally, choosing nodes with the highest remaining power level ensures sporadic selection of forwarding neighbour nodes. The continuous selection of such nodes spread out the traffic load to neighbours in the direction of the sink, hence, prolonging the WSN lifetime.

The RTLD routing protocol possesses built-in security measure. The built-in security features is created by random selection of forwarding nodes based on OF that relies on varying parameters of signal strength and SNR. These physical parameters can not be easily altered by other sensor nodes. The built-in security gauge in RTLD routing protocol can be relatively secure against wormhole, sinkhole, and Sybil attacks. Besides, RTLD routing protocol proposes a new type of forwarding mechanism in WSNs called geodirectional-cast forwarding based on quadrant. Geodirectional-cast forwarding combines geocast with directional forwarding to forward data packet through multiple paths to the destination.

The proposed RTLD routing protocol has been studied through simulation and experimental implementation in real test bed. In both simulation and test bed networks, IEEE 802.15.4 MAC and physical layers are used. The RTLD routing protocol has been compared with the existing baseline routing such as RTPC, LQ and MM-SPEED. In the simulation work, 121 nodes are distributed in a 100m x 100m region. Four sensor nodes represent the source nodes and one node (node 0) is the sink. The simulation shows that RTLD routing protocol experiences packet delay of 150 ms to forward a packet through 10 hops. In the unicast mode, it endures higher delivery ratio up to 7 %, and spends less packet overhead compared to the baseline routing. Beside that, it utilizes less power consumption by 15% compared to the baseline routing protocols. The routing problem handler with feedback packet in RTLD routing protocol overcome routing hole problems and consequently increases the throughput by 30% higher than RTLD without feedback packet. In addition, the simulation results show that RTLD routing spreads out and balances the forwarding load among sensor nodes along the path and consequently prolongs the lifetime of the WSN by as much as 16% compared to the baseline protocol. The outstanding results owes to the forwarding strategy choosing the next hop that has

the optimal combination of the best link quality, remaining power and packet velocity. Beside that, the neighbour discovery in RTLD routing protocol does not allow one-hop neighbour to reply if it is not in the direction to the destination and hence reduces the probability of collision and minimize packet overhead.

When the proposed RTLD routing protocol employs geodirectional-cast forwarding mechanism the throughput of the WSN is raised up to 20% compared to using unicast forwarding mechanism (at moderate traffic load) as it allows forwarding of data packets through multiple paths. However, the gain in the throughput is achieved at the cost of increased power consumption and packet overhead by 9% and 4% respectively. This is due to the original sources broadcasting data packets to one hop neighbours to allow multi-path forwarding.

The RTLD routing protocol has been verified experimentally in real test bed network setup. The experimental test bed consists of 25 sensor nodes (10 TELOSB and 15 MICAZ) with MIB 510CA programming board and multipurpose sensor board. TOSSIM was used to debug, test, and analyze algorithms before the code compared with was uploaded into the sensor node. RTLD in simulation based experiences 5% higher delivery ratio than RTLD test bed based. RTLD<sub>G</sub> simulation based experiences between 7.5% higher delivery ratio than RTLD<sub>G</sub> test bed based. This is mainly due to the link quality in test bed is affected by unpredictable parameters in wireless communication such as fading, reflection and diffraction. However, the processing delay in the simulation study is less than the test bed because the processor of the PC that simulates RTLD is better and faster than the microprocessor in MICAZ and TELOSB. Perhaps it is very important to note that unreliable links in wireless networks may cause retransmissions of the packet at the MAC layer which increase the average delay. Nevertheless, the end-to-end delay in the test bed is below the end-to-end deadline limit which is 250 ms.

In general, RTLD routing protocol has been successfully studied through simulation and experimentally successfully tested in a real test bed implementation. It

offers better performance than the existing baseline routing protocols. The RTLD routing protocol can be used in WSN applications that required real-time forwarding such as disasters fighting, forest fire detection and volcanic eruption detection. The good results open up for future work to further improve the RTLD as elaborated in the following section.

## 6.2 Future Works

RTLD routing protocol in WSN provides a channel for further enhancement towards future applications such as coexistence with wireless network, monitoring applications for indoor and outdoor application and etc. However, further work can be carried out to enhance the performance of the proposed routing protocol. The suggestions for future works are as follows:

- To develop tiny secure systems for real-time communication in WSN which can be used to defend against other type of attacks including manipulating routing information, Sybil, sinkhole and wormhole attacks.
- To develop an artificial intelligent algorithms using fuzzy logic to exclude the repeating sensory data before packets are sent to the sink in the next hop neighbors.
- To investigate the performance of TCP/IP data traffic in the RTLD routing protocol for real-time multimedia applications.
- To investigate using Ultra-wideband (UWB) transceiver with WSNs. UWB differs substantially from conventional narrowband radio frequency (RF) used in the project. UWB is able to transmit higher bit rates than the more traditional technologies which means UWB is more suitable for multimedia transmission applications. The RTLD routing protocol with UWB transceivers can be used to forward multimedia towards the sink.

- To develop Wireless Multimedia Sensor Networks (WMSNs) which are a new and emerging type of sensor networks that contain sensor nodes equipped with cameras, microphones, and other sensors producing multimedia content. These networks have the potential to enable a large class of applications such as multimedia surveillance networks, target tracking, environmental monitoring, and traffic management systems. WMSN can be established using the RTLD routing protocol and UWB transceivers.

## REFERENCE

1. M. Schar and P. Chou. Multimedia over IP and Wireless Networks, 1st Edition, Elsevier Inc., 2007.
2. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, A Survey on Sensor Networks, IEEE Communication Magazine, 2002.
3. W. Woon and T.C. Wan. Extensive Performance Evaluation of IEEE 802.15.4 Ad Hoc Wireless Sensor Networks: Simulation and Hardware Approach, SMC '06. IEEE International Conference ,Volume: 2, Oct. 2006. Page(s): 1443-1448.
4. J. Zheng and MJ Lee, A Comprehensive. Performance Study of IEEE 802.15.4, IEEE Press. Book, 2004.
5. M. Chen, V.C.M. Leung, S. Mao, Y. Yuan. Directional Geographical Routing for Real-Time Video Communications in Wireless Sensor Networks, Elsevier Computer Communications Journal, Volume 30, Issue 17, November 2007. Pages: 3368-3383.
6. K. Romer, F. Mattern. The design space of wireless sensor networks, IEEE Wireless Communications Journal, Volume 11, Issue 6, Dec. 2004 Page(s):54 – 61.
7. E. Felemban, C. G. Lee, E. Ekici, R. Boder and S. Vural, Probabilistic qos guarantee in reliability and timeliness domains in wireless sensor networks, 24th Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE Proceedings, 2005. Pages: 2646 – 2657.
8. J.N. Al-Karak and A.E. Kamal , Routing techniques in wireless sensor networks: a survey, wireless communications IEEE journal, Volume: 11, Dec. 2004. Pages. 6- 28,

9. T. He, J. Stankovic, C. Lu, and T. Abdelzaher, SPEED: A stateless protocol for real-time communication in sensor networks, 23rd International Conference on Distributed Computing Systems, IEEE Proceedings, 2003. Pages: 46 - 55.
10. J. Zhao and R. Govindan, Understanding Packet Delivery Performance in Dense Wireless Sensor Networks, Proceedings of the 1st international conference on Embedded networked sensor systems, USA, (2003)
11. A. Cerpa, J. L. Wong, L. Kuang, M. Potkonjak and D.Estrin, Statistical Model of Lossy Links in Wireless Sensor Networks, in Proc. ACM/IEEE IPSN, Los Angeles, USA, April 2005.
12. D. Son., B. Krishnamachari, J. Heidemann. Experimental study of the effects of transmission power control and blacklisting in wireless sensor networks, In Proceedings of the First IEEE Conference on Sensor and Ad hoc Communication and Networks, Santa Clara, California, USA, IEEE. October, 2004, page(s): 289-298
13. J. P. Walters, Z. Liang, W. Shi, V. Chaudhary, Wireless Sensor Network Security: A Survey, in Journal of Parallel and Distributed Computing, to appear, Wayne State University, USA, 2005.
14. C. Karlof and D. Wagner. Secure routing in wireless sensor networks: attacks and countermeasures, Elsevier's Ad Hoc Network Journal, Special Issue on Sensor Network Applications and Protocols, September 2003. Pages 293–315
15. X. Du, Y. Xiao, H. Chen, and Q. Wu, Secure Cell Relay Routing Protocol for Sensor Networks, Wireless Communications and Mobile Computing (WCMC) Journal, John Wiley & Sons, Special issue on Wireless Network Security, Vol. 6, No. 3, May 2, 2006, Pages 375 – 391
16. A.D. Wood, J. A. Stankovic JA, Denial of service in sensor networks. Computer 2002; 35(10): 54–62.
17. D. Gay, P Levis, R. von Behren, M. Welsh. E. Brewer, and D. Culler, The nesc language: A holistic approach to networked embedded systems, in Proc. ACM SIGPUN 2003, conference on Programming Language design and implementation, June 2003, pp. 1-1 1
18. M. Conti, J. Crowcroft, A. Passarella. Multi-hop Ad hoc Networks from Theory to Reality, Hindawi Publishing Corporation, EURASIP Journal on



Wireless Communications and Networking, Volume 2007, Article ID 60740. 2007.

19. Koubâa. A, Alves. M, and Tovar. E. IEEE 802.15.4 for Wireless Sensor Networks: A Technical Overview. Technical Report. 14 July 2005. <http://www.dei.isep.ipp.pt/~akoubaa/publications/TR-050702.pdf>. (Accessed on 01/03/2007).
20. IEEE 802.15.4 Standard (2003) Part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), IEEE Standard for Information Technology, IEEE-SA Standards Board, 2003.
21. A. Koubâa, M. Alves, E. Tovar, IEEE 802.15.4: a wireless communication technology for large-scale ubiquitous computing applications, in Conference on Mobile and Ubiquitous Systems (CSMU 2006), Guimarães, June 29-30, 2006.
22. I. Howitt, and J.A Gutierrez, IEEE 802.15.4 low rate-wireless personal area network coexistence issues. IEEE Wireless Communications and Networking Conference (WCNC'03), vol. 3, pp. 1481- 1486, 16-20 Mar. 2003.
23. S.Y. Shin, S. Choi, H.S. Park, and W.H. Kwon, Packet error-rate analysis of IEEE 802.15.4 under IEEE 802.11b interference, In Proc. of Conference on Wired/Wireless Internet Communications (WWIC'2005), Springer LNCS, vol. 3510, Xanthi (Greece), 11-13 May, 2005.
24. Crossbow Tech, Avoiding RF interference between WiFi and Zigbee, Crossbow Technical Report, 2005
25. C. Ching and S. P. Kumar. Sensor Networks: Evolution, Opportunities, and Challenges, Invited paper, Proceedings of IEEE, Vol.91, No.8, Aug. 2003.
26. D. W. Carman, P. S. Krus, and B. J. Matt, Constraints and approaches for distributed sensor network security, Technical Report 00-010, NAI Labs, Network Associates, Inc., Glenwood, MD, 2000.
27. Crossbow technology, MPR/ MIB User's Manual, Rev. B, April 2005, Document 7430-0021-06.
28. J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. Pister. System architecture directions for networked sensors. In Architectural Support for Programming Languages and Operating Systems, pages 93–104, 2000.

29. N. Bulusu, J. Heidemann, and D. Estrin. GPS-less Low Cost Out Door Localization for Very Small Devices, Tech. rep. 00729, Comp. Sci. Dept., USC, Apr. 2000.
30. J.A Stankovic, T. Abdelzaher, C. Lu, L. Sha and J. Hou, Real time communication and coordination in embedded sensor networks, Proceedings of the IEEE journal vol. 91, num 7, July 2003. Pages: 1002-1022.
31. C. Lu, B.M Blum, T.F Abdelzaher, J.A Stankovic, and T. He, RAP: A real-time communication architecture for large-scale wireless sensor networks, Real-Time and Embedded Technology and Applications Symposium, IEEE conference, 2002. Pages:55 - 66.
32. H. Li, P. Shenoy and K. Ramamritham, Scheduling Messages with Deadlines in Multi-hop Real-time Sensor Networks, IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2005), San Francisco, California, March , 2005.
33. O. Chipara, Z. He, G. Xing, Q. Chen, X. Wang, C. Lu, J. Stankovic, and T. Abdelzaher, Real-time Power-Aware Routing in Sensor networks, Quality of Service, IWQoS 2006, 14th IEEE International Workshop on June 2006 Page(s):83 - 92.
34. M. Zuniga and B. Krishnamachari, Analyzing the Transitional Region in Low Power Wireless Links, Sensor and Ad Hoc Communications and Networks, IEEE SECON 2004, 1st Annual IEEE Communications Society Conference, 2004. Pages: 517 - 526.
35. M. Caccamo, L.Y. Zhang, L. Sha and G. Buttazzo, An Implicit Prioritized Access Protocol for Wireless Sensor Networks, Proceedings of the 23rd IEEE IEEE Real-Time Systems Symposium(RTSS'02), December 2002.
36. H. Peng, Z. Xi, Li Ying, C. Xun, and G Chuanshan. An Adaptive Real-Time Routing Scheme for Wireless Sensor Networks, Advanced Information Networking and Applications Workshops, AINAW '07, 21st International Conference on May 2007, Volume 2, Page(s):918 – 922.
37. D. De Couto, D. Aguayo, J. Bicket, and R. Morris, High-throughput Path Metric for Multi-hop Wireless Routing, 9th Annual International Conference on Mobile Computing and Networking (MOBICOM), Sep 14-19, 2003

38. T. Roosta, Probabilistic Geographic Routing in Ad Hoc and Sensor Networks, in Proc. of International Workshop on Wireless Ad-hoc Networks (IWWAN), London, UK, May 2005
39. V.C Gungor, C. Sastry, Z. Song and R. Integlia Resource-Aware and Link Quality Based Routing Metric for Wireless Sensor and Actor Networks, ICC '07. IEEE International Conference on Publication Date: 24-28 June 2007. Page(s): 3364-3369
40. P. Jiang, Q. Huang, J. Wang, X. Dai, R. Lin. Research on Wireless Sensor Networks Routing Protocol for Wetland Water Environment Monitoring, First International Conference on Innovative Computing, Information and Control (ICICIC'06), China, 2006. Page(s): 251-254.
41. A. Mahapatra, K. Anand, D. P. Agrawal. QoS and energy aware routing for real-time traffic in wireless sensor networks, Computer Communications, Elsevier Journal, Volume 29, Issue 4, 20 February 2006. Pages: 437-445.
42. K. Akkaya and M. Younis. An energy-aware QoS routing protocol for wireless sensor networks, Proceedings of the 23rd International Conference on Distributed Computing Systems, IEEE Computer Society, USA, May 2003. Page(s): 710- 715
43. J.H. Chang, and L. Tassiulas. Maximum lifetime routing in wireless sensor networks, IEEE Journal, IEEE/ACM Transactions on Volume 12, Issue 4, Aug. 2004. Page(s):609 – 619
44. J. Park and S. Sahni. An online heuristic for maximum lifetime routing in wireless sensor networks, Computers IEEE Journal, IEEE Transactions on Volume 55, Issue 8, Aug. 2006. Page(s):1048 - 1056
45. S. Sahni, Data Structures, Algorithms, and Applications in Java, second ed. Silicon Press, 2005.
46. K. Seada, M. Zuniga, A. Helmy, and B. Krishnamachari, Energy-Efficient Forwarding Strategies for Geographic Routing in Lossy Wireless Sensor Networks, in IEEE SECON 2004, Santa Clara, CA, Oct. 2004.
47. S. Dulman, T. Nieberg, J. Wu, P. Havinga. Trade-Off between Traffic Overhead and Reliability in Multipath Routing for Wireless Sensor Networks, WCNC Workshop, New Orleans, Louisiana, USA, March 2003.

48. B. Karp, H. Kung, GPSR: greedy perimeter stateless routing for wireless networks, ACM MOBICOM 2000, Boston, Massachusetts. 6-11 August, 2000.
49. Y. Ko, N. Vaidya, Flooding-based geocasting protocols for mobile ad hoc networks, ACM/Baltzer Mobile Networks and Applications (MONET) Journal (2002).
50. Y. Ko, Anycasting-based protocol for geocast service in mobile ad hoc networks, Computer Networks, Elsevier North-Holland Journal, Volume 41 , Issue 6 2003. Pages: 743 – 760.
51. V. Park, M. Corson, A highly adaptive distributed routing algorithm for mobile wireless networks, IEEE INFOCOM (1997).
52. K. Seada and A. Helmy. Efficient and Robust Geocasting Protocols for Sensor Networks. Elsevier Computer Communications Journal, Special Issue on Dependable Wireless Sensor Networks, 2005.
53. C. Intanagonwiwat, R. Govindan, and D. Estrin, Directed diffusion: a scalable and robust communication paradigm for sensor networks, Proceedings of ACM MobiCom '00, Boston, MA, 2000. Pages: 56-67.
54. H. Shokrzadeh, A. T. Haghighat, F. Tashtarian and A. Nayeibi. Directional rumor routing in wireless sensor networks, ICI 2007, 3rd IEEE/IFIP International Conference in Central Asia, Sept. 2007. Pages: 1-5
55. J. Kulik, W. R. Heinzelman, and H. Balakrishnan. Negotiation-Based Protocols for Disseminating Information in Wireless Sensor Networks, Wireless Networks, vol. 8, 2002. Pages: 169–85.
56. K. Sohrabi and J. Pottie. Protocols for Self-Organization of a Wireless Sensor Network, IEEE Pers. Commun., vol. 7, no. 5, 2000. Pages: 16–27.
57. J.R. Douceur, The Sybil Attack. Proceedings of IPTPS'02, March 2002.
58. Castro and Liskov, “Practical byzantine fault tolerance,” in OSDI: Symposium on Operating Systems Design and Implementation. USENIX Association, Co-sponsored by IEEE TCOS and ACM SIGOPS, 1999.
59. A. Banerjea, “A taxonomy of dispersity routing schemes for fault tolerant real-time channels,” in Proceedings of ECMAST, vol. 26, May 1996, pp. 129–148.

60. K. Ishida, Y. Kakuda, and T. Kikuno, "A routing protocol for finding two node-disjoint paths in computer networks," in *International Conference on Network Protocols*, November 1992, pp. 340–347.
61. Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad hoc routing," in *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking*, 2001.
62. B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *ACM Wireless Networks Journal*, vol. 8, no. 5, September 2002.
63. Y.-C. Hu, A. Perrig, D.B. Johnson, Packet leashes: a defense against wormhole attacks in wireless networks, in *IEEE Infocom*, 2003.
64. C. Karlof, N. Sastry, D. Wagner, TinySec: a link layer security architecture for wireless sensor networks. *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, Baltimore, Maryland, November 2004.
65. HangRok Lee, YongJe Choi, HoWon Kim. Implementation of TinyHash based on Hash Algorithm for Sensor Network. *Transaction on Engineering, Computing and Technology*, Krakow, Poland, Volume 10, December , 2005.
66. D. J. Wheeler and R. M. Needham. TEA: a Tiny Encryption Algorithm. In *Fast Software Encryption, Second International Workshop Proceedings*, SpringerVerlag, pages 97-110, 1995.
67. Y. Kanamori, E. Jovanov, and S.-M. Yoo. Performance comparison between tea and rijndale encryption algorithm for wireless sensor networks. In *ISCA 15th International Conference on Computer Applications in Industry and Engineering (CAINE)*, San Diego, pages 209-212, Nov. 2000.
68. Shuang Liu, Olga V. Gavrylyako, Phillip G. Bradford: Implementing the TEA algorithm on sensors. *ACM Southeast Regional Conference 2004*: 64-69.
69. Hao Yang, Fan Ye, Yuan Yuan, Songwu Lu, William Arbaugh. Toward Resilient Security in Wireless Sensor Networks, *ACM MOBIHOC 2005*, Urbana-Champaign, IL, USA. May 2005.

70. Deng, R. Han and S. Mishra, Enhancing Base Station Security in Wireless Sensor Networks, Technical Report CU-CS-951-03, Department of Computer Science, University of Colorado, April 2003.
71. D. Kahn, The Codebreakers: The Story of Secret Writing, New York: Macmillan Publishing Co., 1967.
72. B. Schneier, Applied Cryptography. J. Willey & Sons, second edition, 1996.
73. A.K. Pathan, H.W Lee, C.S Hong. Security in Wireless Sensor Networks: Issues and Challenges, Proceedings of 8th IEEE ICACT 2006, Volume II, February 20-22, Phoenix Park, Korea, 2006. Pages: 1043-1048
74. W. Su, T.L Lim. Cross-Layer Design and Optimization for Wireless Sensor Networks, IEEE conference, 7th ACIS International Conference on 19-20 June 2006 Page(s):278 – 284.
75. L.A. Latiff, A. Ali, C.C Ooi and N.Fisal, Development of an Indoor GPS-free Self Positioning System for Mobile Ad Hoc Network (MANET), MICC-ICON 2005, Malaysia, 16-18 November 2005
76. A. Ali, L.A Latiff and N. Fisal, GPS-free indoor location tracking in mobile ad hoc network (MANET) using RSSI, RF and Microwave Conference, IEEE Proceedings, (2004) 251 – 255
77. L. Dhananjay, A. Manjeshwar, F. Herrmann, E.U. Biyikoglu, and A. Keshavarzian, Measurement and characterization of link quality metrics in energy constrained wireless sensor networks, Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE Volume 1, 1-5 Dec. 2003.
78. B. Sklar , Digital Communications, Fundamentals and Applications (2nd Edition), Prentice Hall PTR, January 11 2001
79. IEEE P802.15 Working Group: Draft Coexistence Assurance for IEEE 802.15.4b. IEEE 15-05-0632-00-004b. 2005
80. Chipcon. CC2420 low power radio transceiver, <http://www.chipcon.com>.
81. L.A. Latiff, A. Ali, C.C. Ooi and N.Fisal, Implementation of a Quadrant-Based Directional Routing Protocol (Q-DIR) In Wireless Mobile Ad Hoc Network , NCS 2006, Chiang Mai, Thailand, 28-30 March 2006
82. C. Blondia, N. V. den Wijngaert, G. Willems, O. Casals, L. Cerdà, M. Bagnulo, I. Soto: Mobile Networking. Quality of Future Internet Services, Springer Berlin / Heidelberg, Volume 2856, 2003. Pages: 180-206.

83. B. Bougard, F. Catthoor, D. Daly, A. Chandrakasan, and W. Dehaene, Energy efficiency of the IEEE 802.15.4 standard in dense wireless microsensor networks: Modeling and improvement perspectives. IEEE DATE 2005, pages 196 – 201, March 2005.
84. P. Lewis, A. Goodman and J.M. Miller. A pseudo-random number generator for the System/360. IBM Systems Journal , Volume 8, Number 2, Page 136 (1969).
85. Crossbow Technology, Inc. <http://www.xbow.com>, Accessed on 30/12/07
86. The VINT Project. 2006. The ns Manual, formerly ns Notes and Documentation. August 14, 2006. <http://www.isi.edu/nsnam/ns>. accessed on 11/03/2007
87. T. S. Rappaport. Wireless communications, principles and practice. Prentice Hall, 1996.
88. Wikipedia, optimization problem, [http://en.wikipedia.org/wiki/Optimization\\_problem](http://en.wikipedia.org/wiki/Optimization_problem), Wikimedia Foundation, Inc Accessed on 17/10/2007
89. A.M. Mood, F.A. Graybill and D.C. Boes. Introduction to the theory of statistics. 3<sup>rd</sup> Ed. McGraw-Hill, 1974. pages: 25 – 27
90. S. Ni , Y. Tseng , Y. Chen , J. Sheu, The broadcast storm problem in a mobile ad hoc network, Wireless Networks, Volume 8, 2002. Pages: 153 - 167
91. Crossbow technology, MTS/MDA Sensor and Data Acquisition Board User's Manual, Rev. B, April 2005, Document 7430-0020-03
92. Cygwin Information and Installation, <http://www.cygwin.com/>, Accessed on 04/07/2007.
93. P. Levis, N. Lee, M. Welsh, and D. Culler, TOSSIM: Accurate and scalable simulation of entire TinyOS applications, in Proceedings of SenSys'03, First ACM Conference on Embedded Networked Sensor Systems, Los Angeles, California, USA, 2003.
94. K. Chang and D. Gay, Language Support for Interoperable Messaging in Sensor Networks, SCOPES 2005, 1-9 9th International Workshop on Software and Compilers for Embedded Systems Dallas, Texas, 2005
95. P. Levis, N. Lee, TOSSIM: a simulator for TinyOS Networks, User's manual, 2003. <http://www.cs.berkeley.edu/~pal/research/tossim.html>. accessed on 02/03/2007.

96. UC Berkeley, TinyOS Tutorial, <http://www.tinyos.net/tinyos-1.x/doc/tutorial>. Last updated 23 September 2003
97. USC Information Sciences Institute, Marina del Rey, CA. Network Simulator – NS2. <http://www.isi.edu/nsnam/ns>. accessed on 11/03/2007
98. Y-C. Hu, A. Perrig, D.B. Johnson, Wormhole detection in wireless ad hoc networks. Department of Computer Science, Rice University, Technical Report TR01-384.
99. S. Capkun and J.-P. Hubaux, Secure positioning in wireless networks, IEEE Journal on Selected Areas in Communications, Volume: 24, Issue 2, 2006. Pages:221–232.
100. L. Lazos and R. Poovendran. Serloc: Robust localization for wireless sensor networks. ACM Trans. Sen. Netw., Vol 1, 2005. Pages:73–100.
101. Crossbow technology, Getting Starting Guide, Rev. A, April 2005, Document 7430-0022-06.
102. USC Information Sciences Institute, Marina del Rey, CA. Network Simulator – NS2. (<http://www.isi.edu/nsnam/ns>).
103. J. Chung, M. Claypool, NS by Example, Worcester Polytechnic Institute (WPI), computer science, <http://nile.wpi.edu/NS/> , 2006
104. R. Madan and S. Lall. Distributed algorithms for maximum lifetime routing in wireless sensor networks, Wireless Communications Journal, IEEE Transactions, Volume 5, Issue 8, Aug. 2006. Page(s):2185 - 2193



## APPENDIX A

### MICAZ and TELOSB specifications

**Table A.1:** MICAZ Datasheet

Processor/Radio Board	MPR2400CA	Remarks
<b>Processor Performance</b>		
Program Flash Memory	128K bytes	
Measurement (Serial) Flash	512K bytes	> 100,000 Measurements
Configuration EEPROM	4K bytes	
Serial Communications	UART	0-3V transmission levels
Analog to Digital Converter	10 bit ADC	8 channel, 0-3V input
Other Interfaces	Digital I/O, I2C, SPI	
Current Draw	8 mA	Active mode
	< 15 $\mu$ A	Sleep mode
<b>RF Transceiver</b>		
Frequency band <sup>1</sup>	2400 MHz to 2483.5 MHz	ISM band, programmable in 1 MHz steps
Transmit (TX) data rate	250 kbps	
RF power	-24 dBm to 0 dBm	
Receive Sensitivity	-90 dBm (min), -94 dBm (typ)	
Adjacent channel rejection	47 dB	+ 5 MHz channel spacing
	38 dB	- 5 MHz channel spacing
Outdoor Range	75 m to 100 m	1/2 wave dipole antenna, LOS
Indoor Range	20 m to 30 m	1/2 wave dipole antenna
Current Draw	19.7 mA	Receive mode
	11 mA	TX, -10 dBm
	14 mA	TX, -5 dBm
	17.4 mA	TX, 0 dBm
	20 $\mu$ A	Idle mode, voltage regular on
	1 $\mu$ A	Sleep mode, voltage regulator off
<b>Electromechanical</b>		
Battery	2X AA batteries	Attached pack
External Power	2.7 V - 3.3 V	Molex connector provided
User Interface	3 LEDs	Red, green and yellow
Size (in)	2.25 x 1.25 x 0.25	Excluding battery pack
(mm)	58 x 32 x 7	Excluding battery pack
Weight (oz)	0.7	Excluding batteries
(grams)	18	Excluding batteries
Expansion Connector	51-pin	All major I/O signals

**Table A.2:TELOSB Datasheet**

Specifications	TPR2420CA	Remarks
<b>Module</b>		
Processor Performance	16-bit RISC	
Program Flash Memory	48K bytes	
Measurement Serial Flash	1024K bytes	
RAM	10K bytes	
Configuration EEPROM	16K bytes	
Serial Communications	UART	0-3V transmission levels
Analog to Digital Converter	12 bit ADC	8 channels, 0-3V input
Digital to Analog Converter	12 bit DAC	2 ports
Other Interfaces	Digital I/O,I2C,SPI	
Current Draw	1.8 mA	Active mode
	5.1 $\mu$ A	Sleep mode
<b>RF Transceiver</b>		
Frequency band <sup>1</sup>	2400 MHz to 2483.5 MHz	ISM band
Transmit (TX) data rate	250 kbps	
RF power	-24 dBm to 0 dBm	
Receive Sensitivity	-90 dBm (min), -94 dBm (typ)	
Adjacent channel rejection	47 dB	+ 5 MHz channel spacing
	38 dB	- 5 MHz channel spacing
Outdoor Range	75 m to 100 m	Inverted-F antenna
Indoor Range	20 m to 30 m	Inverted-F antenna
Current Draw	23 mA	Receive mode
	21 $\mu$ A	Idle mode
	1 $\mu$ A	Sleep mode
<b>Sensors</b>		
Visible Light Sensor Range	320 nm to 730 nm	Hamamatsu S1087
Visible to IR Sensor Range	320 nm to 1100nm	Hamamatsu S1087-01
Humidity Sensor Range	0-100% RH	Sensirion SHT11
Resolution	0.03% RH	
Accuracy	$\pm$ 3.5% RH	Absolute RH
Temperature Sensor Range	-40°C to 123.8°C	Sensirion SHT11
Resolution	0.01°C	
Accuracy	$\pm$ 0.5°C	@25°C
<b>Electromechanical</b>		
Battery	2X AA batteries	Attached pack
User Interface	USB	v1.1 or higher
Size (in)	2.55 x 1.24 x 0.24	Excluding battery pack
(mm)	65 x 31 x 6	Excluding battery pack
Weight (oz)	0.8	Excluding batteries
(grams)	23	Excluding batteries

**Table A.3:** Comparison between MICAZ and TELOSB.

	<b>MICAZ</b>	<b>TELOSB</b>
<b>Microcontroller</b>	7.37 MHz Atmel ATMega128L	8 MHz TI MSP430
<b>Memory</b>	128 KB program flash memory, 4 KB RAM and 512KB External Serial Flash Memory	48 KB program flash memory, 10 KB RAM and 1 MB External Serial Flash Memory
<b>Radio Transceiver</b>	Data rate: 250Kbps Encoding: DSSS Modulation: O-QPSK Antenna: 1/2 wave dipole Freq: 2400-2483Mhz	Data rate: 250Kbps Encoding: DSSS Modulation: O-QPSK Antenna: Integrated Onboard Antenna Freq: 2400-2483Mhz
<b>Power options</b>	2xAA batteries and 51-pin or 2-pin molex	2xAA batteries and USB
<b>Upload program</b>	Using external programming board such as MIB510CA/ MIB520CA serial/USB interface board.	Using USB port without external board
<b>Read data sensory</b>	Using external sensor board such as MTS (300/310) CA or MTS (400/420)CA	Using integrated light, temperature and humidity sensor

**Table A.4:** Power Regulation in MICAZ and TELOSB

<b>Power Register (code)</b>	<b>MICAz TX RF Power (dBm)</b>
31	0
27	-1
23	-3
19	-5
15	-7
11	-10
7	-15
3	-25

**Table A.5:** Summary of SensirionSHT11 specification

<b>Sensor Type</b>	Sensirion SHT11	
<b>Channels</b>	Humidity	Temperature
<b>Range</b>	0 to 100%	-40°C to 80°C
<b>Accuracy</b>	± 3.5% RH (typical)	± 2°C
<b>Operating Range</b>	<b>3.6 to 2.4 volts</b>	
<b>Interface</b>	Digital interface	

## **APPENDIX B**

### **Integrating SRTLD in NS-2**

SRTLD is implemented using C++ and then the simulations describing scenarios are implemented using Tcl scripts. To allocate SRTLD, firstly a new directory called real-time is created inside NS-2 base directory. The following files are created as follows:

#### **srtld.h**

This is the header file where will be defined all necessary timers, all necessary routing function and routing agent which performs protocol's functionality.

#### **srtld.cc**

In this file are actually implemented all timers, routing function, routing agent and Tcl hooks.

#### **srtld\_pkt.h**

Here are declared all packets of srtld protocol needs to exchange among nodes in the WSN.

#### **srtld\_rtable.h**

Header file where srtld neighbour table is declared.

#### **protoname\_rtable.cc**

Neighbour table implementation with adding, deleting and updating.

The previous files create the physical structure files for SRTLD. To implement a routing protocol in NS-2, an agent is create by inheriting from Agent class. Agents represent endpoints where network-layer packets are constructed or consumed, and are used in the implementation of protocols at various layers [91]. This is the main class that we have to code in order to implement our routing protocol. In addition, this class offers a linkage with Tcl interface, so we will be able to control our routing protocol through simulation scripts written in Tcl. SRTLD routing agent will maintain an internal state and a neighbour table which is not

always needed. Internal state can be represented as a new class or as a collection of attributes inside the routing agent. We treated neighbour table as a new class “`srtd_rtable`”. In addition, SRTLD routing protocol must define at least one new packet type which will represent the format of its control packets. As we said these packet types are defined in `.../NS-2/real-time/srtd_pkt.h`. When the protocol needs to send packets periodically or after some time from the occurrence of an event, it is very useful to count on a Timer class. Timers are also useful in lots of other cases. Imagine SRTLD needs to store some sort of internal information that must be erased at a certain time. The best solution is to create a custom timer capable of doing such job. A timer should also be used to specify time life of an entry in the neighbour table. In general, we used a timer whenever we have to schedule a task at a given time. We must know another important class before going into details. The Trace class is the base for writing log files with information about what happened during the simulation. And the last hint for now: when you want to print a debug message in your code, it is helpful to use the `debug ()` function. This allows you to turn debugging on or off from your simulation scripts and is easier to read for other programmers.

## **B.1 Developing SRTLD Packet Header**

The file called `srtd_pkt.h` is created and all data structures, constants and macros related to SRTLD new packet type is putted. Figure B.1 shows the `srtd_pkt.h`. In this figure, Lines 7-25 declare `hdr_srtd_pkt` structure that represents the new packet type we are defining. In lines 9-12 we can see three raw attributes in SRTLD packet. They are of following types:

### **`nsaddr_t`**

Every time the user want to declare a network address in NS-2, he can use this type.

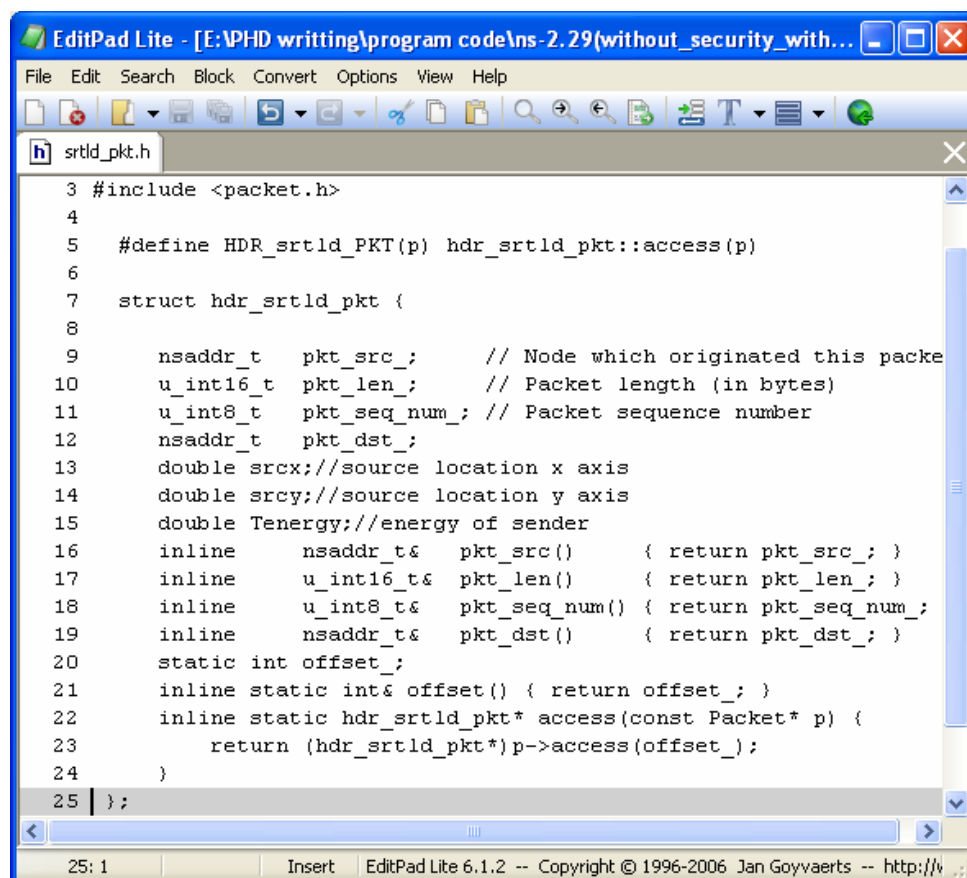
### **`u_int16_t`**

16 bits unsigned integer.

### **`u_int8_t`**

8 bits unsigned integer.

All these types and more are defined in the header file config.h. It is also worth mentioning raw attributes names are expected to finish with an underscore to distinguish them from other variables. Lines 16-19 are member functions for defined attributes. Line 3 includes file common/packet.h that defines packet class. Packets are used to exchange information between objects in the simulation, and our aim is to add `hdr_srtld_pkt` structure to them. Doing so the control packets will be able to be sent and received by nodes in the simulation. by using an array of unsigned characters where packets' fields are saved. To access a concrete packet header is necessary to provide the offset where it is located. In addition, that is exactly what the code does through lines 20-24. A static offset (common to all `hdr_protoname_pkt` structs), a member function to access it and a function which returns a `hdr_srtld_pkt` given a packet are defined. Moreover, in line 5 a macro is create to use this last function.



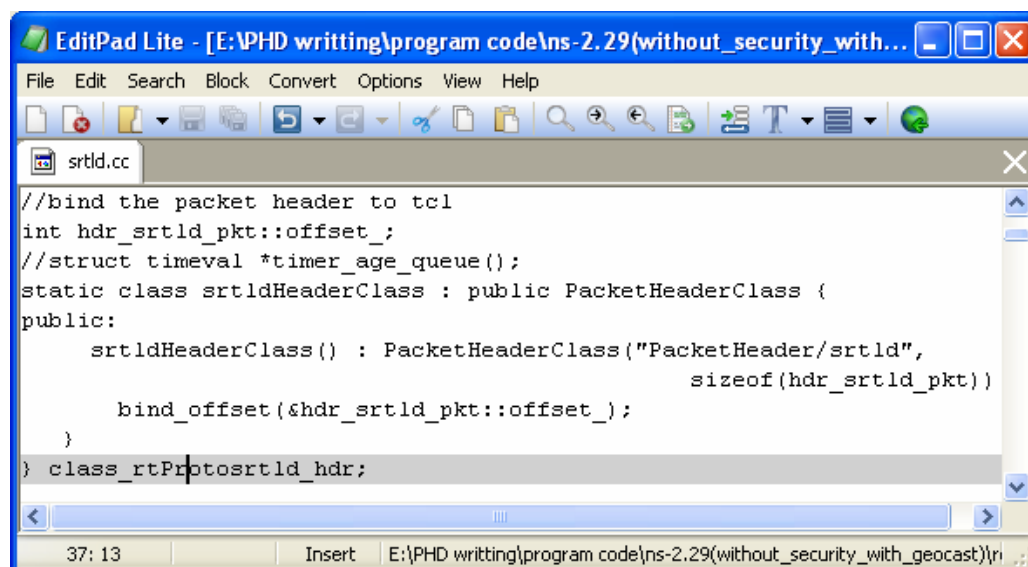
```

EditPad Lite - [E:\PHD writting\program code\ns-2.29(without_security_with...)
File Edit Search Block Convert Options View Help
srtld_pkt.h
3 #include <packet.h>
4
5 #define HDR_srtld_PKT(p) hdr_srtld_pkt::access(p)
6
7 struct hdr_srtld_pkt {
8
9     nsaddr_t    pkt_src;      // Node which originated this packe
10    u_int16_t    pkt_len;      // Packet length (in bytes)
11    u_int8_t     pkt_seq_num;  // Packet sequence number
12    nsaddr_t     pkt_dst;
13    double srcx; //source location x axis
14    double srcy; //source location y axis
15    double Tenergy; //energy of sender
16    inline nsaddr_t& pkt_src() { return pkt_src; }
17    inline u_int16_t& pkt_len() { return pkt_len; }
18    inline u_int8_t& pkt_seq_num() { return pkt_seq_num; }
19    inline nsaddr_t& pkt_dst() { return pkt_dst; }
20    static int offset_;
21    inline static int& offset() { return offset_; }
22    inline static hdr_srtld_pkt* access(const Packet* p) {
23        return (hdr_srtld_pkt*)p->access(offset_);
24    }
25 };
25: 1 Insert EditPad Lite 6.1.2 -- Copyright © 1996-2006 Jan Goyvaerts -- http://w

```

**Figure B.1** SRTLD packet header in NS-2

The ./real-time/srtld.cc source code should be implemented in order to bind SRTLD packet header to Tcl interface. Figure B.2 shows the binding code in srtld.cc file.



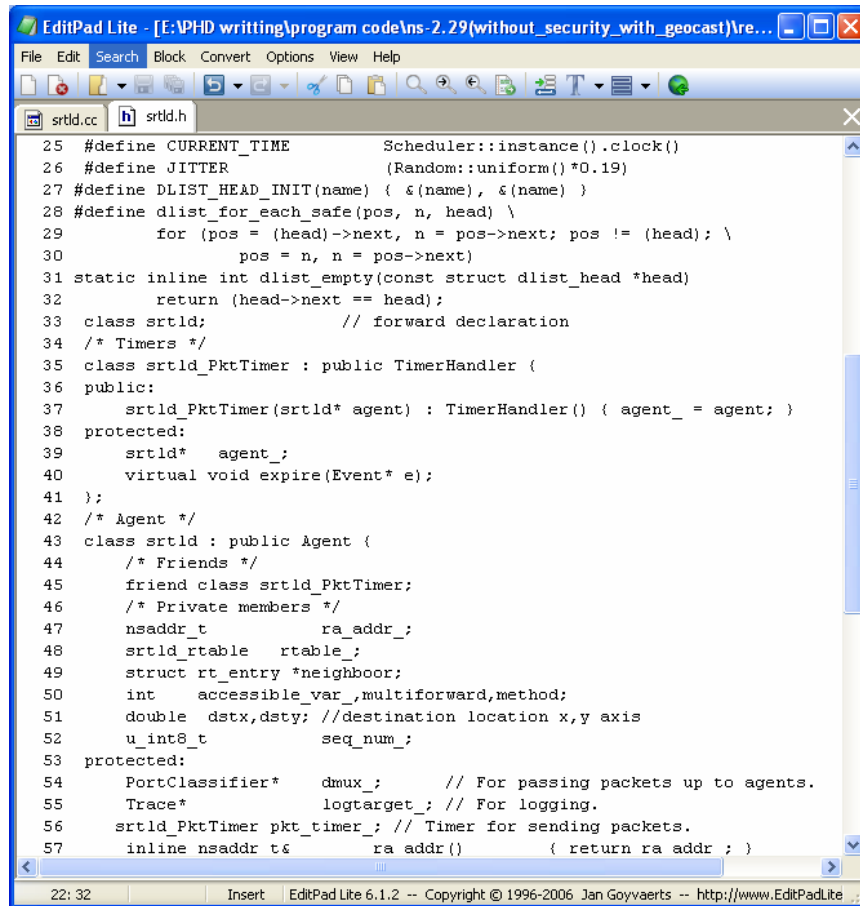
**Figure B.2** Binding SRTLD packet header with Tcl interface.

## B.2 Developing SRTLD Routing Agent

The agent itself is programmed to be able applying RSTLD. Inside real-time/srtld.h, we define a new class called SRTLD containing the attributes and functions needed to assist the protocol in doing its job. To illustrate the use of timers, we assume that SRTLD is a proactive routing protocol that requires sending out some control packets periodically. Figures B.3(a, b) show the source code of srtld.h. In this figure, line 25 defines a useful macro for getting current time in the simulator clock. That is done by accessing the single instance of scheduler class. This object manages all events produced during simulation and simulator's internal clock. Another macro is in line 26. It is just an easy way to obtain a random number inside [0-0.19] interval. This is commonly used to randomize the sending of control packets to avoid synchronization between a node and its neighbours that would eventually produce collisions and therefore delays of sending these packets. Lines 34-41 declare SRTLD custom timer for sending periodical control packets. The srtld\_PktTimer class inherits from Timer Handler and has a reference to the routing agent that



creates it. This is used as a callback for telling routing agent to send a new control packet and to schedule the next one. To do these callbacks routing agent needs to treat `srtld_PktTimer` as a friend class (line 45). The `srtld` class is defined within lines 41 and end of file. It encapsulates its own address, internal state, neighbour table, an accessible variable from Tcl and a counter for assigning sequence numbers to output packets (lines 46-52). `Dstx` and `dsty` are thought to be read and written from Tcl scripts or shell commands. This is useful in many situations because it allows users to change simulation behaviour through their scripts without re-compiling the simulator. A Port Classifier object is declared in line 54. A node consists of an address classifier and a port classifier. The first is used to guide incoming packets to a suitable link or to pass them to the port classifier, which will carry them to appropriate upper layer agent. That is why the routing agent needs a port classifier. When it receives data packets destined to it, it will use `dmux_` in order to give them to corresponding agent. Another important attribute is the Trace object (see line 55). It is used to produce logs to be store in the trace file. In Figure B.3 (a), we use it to write the contents of the neighbour table whenever the user requests it from the Tcl interface. In that case, those logging functions are implemented in other location. Line 56 declares SRTLTD custom timer. Line 59-67 in Figure B.3 (b) defined the functions that will be used to forward data packets to their correct destination; to broadcast control packet; to receive control packet and to schedule SRTLTD custom timer expiration. Lines 68-73 contain public functions of class `srtld`. Constructor receives as an argument an identifier used as the routing agent's address. `Srtld` inherits from Agent base class two main functions which need to be implemented: `recv()` and `command()`. `recv()` is called whenever the agent receives a packet. This may occur when the node itself (actually an upper layer agent such as UDP or TCP) is generating a packet or when it is receiving one from another node. The `command()` function is invoked from Tcl to ask the C++ object to do some task from Tcl code.

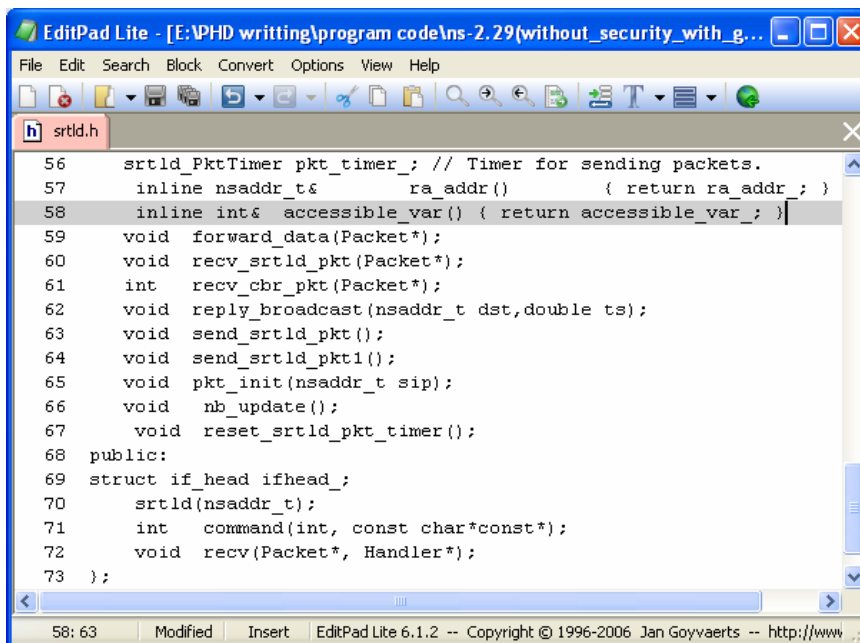


```

25 #define CURRENT_TIME      Scheduler::instance().clock()
26 #define JITTER             (Random::uniform()*0.19)
27 #define DLIST_HEAD_INIT(name) { &(amp;name), &(name) }
28 #define dlist_for_each_safe(pos, n, head) \
29     for (pos = (head)->next, n = pos->next; pos != (head); \
30         pos = n, n = pos->next)
31 static inline int dlist_empty(const struct dlist_head *head)
32     return (head->next == head);
33 class srtld;           // forward declaration
34 /* Timers */
35 class srtld_PktTimer : public TimerHandler {
36 public:
37     srtld_PktTimer(srtld* agent) : TimerHandler() { agent_ = agent; }
38 protected:
39     srtld*    agent_;
40     virtual void expire(Event* e);
41 };
42 /* Agent */
43 class srtld : public Agent {
44     /* Friends */
45     friend class srtld_PktTimer;
46     /* Private members */
47     nsaddr_t    ra_addr_;
48     srtld_rtable rtable_;
49     struct rt_entry *neighbor;
50     int    accessible_var_, multiforward, method;
51     double dstx, dsty; //destination location x,y axis
52     u_int8_t    seq_num_;
53 protected:
54     PortClassifier*  dmux; // For passing packets up to agents.
55     Trace*          logtarget; // For logging.
56     srtld_PktTimer pkt_timer; // Timer for sending packets.
57     inline nsaddr_t& ra_addr() { return ra_addr; }

```

(a)



```

56 srtld_PktTimer pkt_timer; // Timer for sending packets.
57 inline nsaddr_t& ra_addr() { return ra_addr; }
58 inline int& accessible_var() { return accessible_var; }
59 void forward_data(Packet*);
60 void recv_srtld_pkt(Packet*);
61 int  recv_cbr_pkt(Packet*);
62 void reply_broadcast(nsaddr_t dst, double ts);
63 void send_srtld_pkt();
64 void send_srtld_pkt1();
65 void pkt_init(nsaddr_t sip);
66 void nb_update();
67 void reset_srtld_pkt_timer();
68 public:
69 struct if_head ifhead;
70 srtld(nsaddr_t);
71 int  command(int, const char*const*);
72 void recv(Packet*, Handler*);
73 };

```

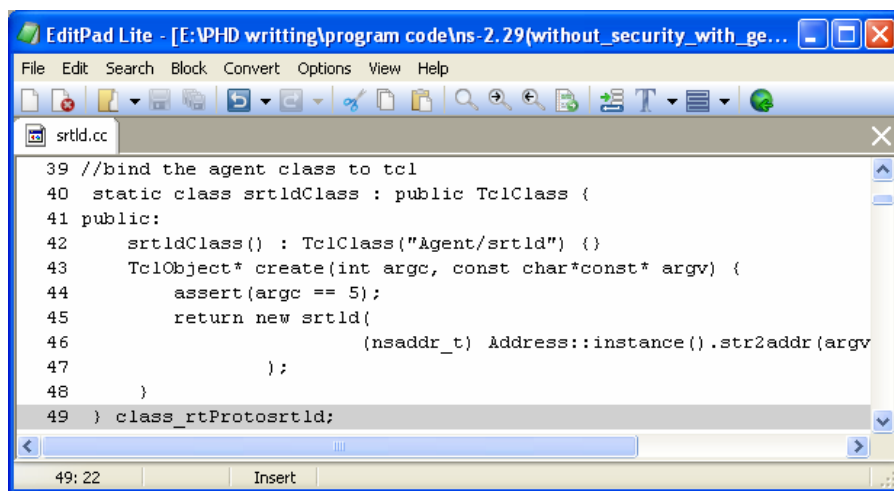
(b)

Figure B.3 Source code of srtld.h.

The following section will explain the implementation of SRTLD routing protocol. It is related to the real-time/srtld.cc file.

### B.3 Binding SRTLD Packet to Tcl

The previous section shows how to bind SRTLD packet to Tcl. This section will do the same for agent class. The aim is to let SRTLD to be instantiated from Tcl. To do so we must inherit from the class TclClass as depicted in figure B.4.



```

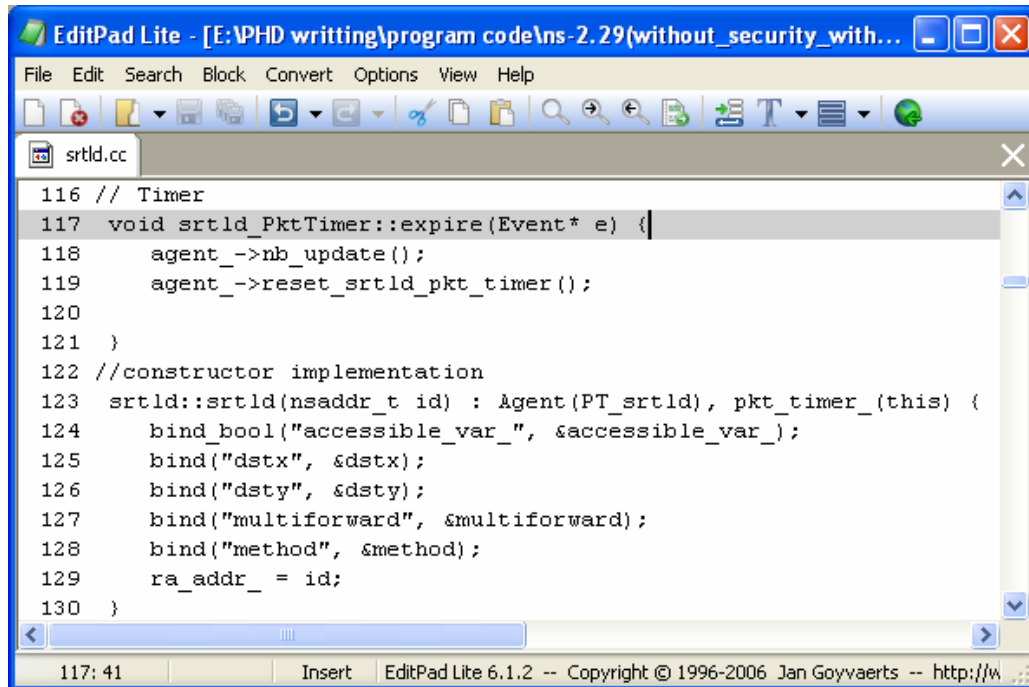
39 //bind the agent class to tcl
40 static class srtldClass : public TclClass {
41 public:
42     srtldClass() : TclClass("Agent/srtld") {}
43     TclObject* create(int argc, const char*const* argv) {
44         assert(argc == 5);
45         return new srtld(
46             (nsaddr_t) Address::instance().str2addr(argv
47             );
48     }
49 } class_rtProtoSrtld;
  
```

**Figure B.4** Tcl hooks

The class constructor is in line 42 and it merely calls the base class with the string “Agent/srtld” as an argument. This represents class hierarchy for this agent in a textual manner. In lines 43-47, we implement a function called create () that returns a new srtld instance as a TclObject. We use the Address class to get nsaddr\_t type from a string.

### B.4 Developing Timers

The expire method had to code in real-time/srtld.cc. Implementing this is easy because it only is sending a new control packet and rescheduling the timer. According to design decisions these two tasks must be executed by the routing agent, callbacks are invoked as shown in figure B.5.



**Figure B.5** SRTLD timer

## B.5 Agent

### B.5.1 SRTLD Agent Class Constructor

This section had explained constructor implementation. As Figure B.6 shows that the constructor started by calling the base class passing PT\_srtd as an argument. SRTLD routing agent uses this constant to identify control packets sent and received. Just after that dstx, dsty, multiforward, and method are binded as a integer attribute which now may be read and written from Tcl. Line 129 saves the given identifier as the routing agent's address. Accessing from Tcl scripts is simple as shown in srtd.tcl as "Agent/srtd set dstx 10"

### B.5.2 Command()

The section explains the command function in SRTLD. It consists of the implementation of the command method that SRTLD agent inherits from the agent

class. An argv[0] contains the name of the method (always “cmd”) being invoked, argv[1] is the requested operation, and argv[2..argc-1] are the rest of the arguments which were passed. Within this function print\_rtable operation is coded to make accessible from Tcl. The print\_rtable operation dumps the contents of the neighbour table to the trace file. SRTLD code case has three arguments and must finish its execution returning either TCL\_OK (if everything was fine) or TCL\_ERROR (if any error happened). Lines 55-58 describe a mandatory command that we always have to implement: start. The expected behaviour of this command is to configure the agent to begin its execution. In SRTLD case, it starts its packet sending timer. All the required actions that the routing agent must perform in order to begin its operation should be implemented here. Lines 60-78 implement SRTLD print\_rtable command. Firstly, logtarget\_ is checked if it is initialized (line 61). Then the table is dumped into the trace file as is showed in lines 62-64. To understand this piece of code it would be useful that you look into the trace/trace.h header file. There is where the trace class is defined. It has a reference to pt\_ of the BaseTrace class. This last class implements buffer() and dump() functions which are used to get the variable where output is buffered and to flush that buffer to the output file respectively. Finally, line 65 calls the print() function of SRTLD neighbour table for writing into trace file its own content. The TCL code below shows how to execute the print\_rtable operation at a certain time from a simulation script. It assumes that ns\_ contains an instance of Simulator and node\_ is a Node created by ns\_. 255 as argument is passed because this is the number of the port where a routing agent is attached. In srltd.tcl

```
$ns_ at 1B.0 "[$node_ agent 255] print_rtable"
```

Another mandatory command to implement is port-dmux. Its implementation is provided in lines 82-87. NS-2 stores a reference to every compiled object (C++ object) in a hash table to provide a fast access to each of them given its name. We make use of that facility in line 83 to obtain a PortClassifier object given its name. Similarly, there is another mandatory operation called tracetarget (note that we allow it to be called log-target as well) which simply obtains a trace object given its name.

```

55     if (strcasecmp(argv[1], "start") == 0) {
56         pkt_timer_.resched(0.0);
57         //schedule_next_event();
58         return TCL_OK;
59     }
60     else if (strcasecmp(argv[1], "print_rtable") == 0) {
61         if (logtarget_ != 0) {
62             sprintf(logtarget_>pt_>buffer(), "\nP %f %d Routing Table",
63                     CURRENT_TIME, ra_addr());
64             logtarget_>pt_>dump();
65             rtable_.print(logtarget_);
66             x=0.0;
67             for (j=0; j<121; j++)
68             {
69                 node1=Node::get_node_by_address(j);
70                 x=x+node1->energy_model()->energy();
71             }
72             printf("Total energy consuming=%f\n", 435.6-x);
73         }
74         else {fprintf(stdout, "%f %d If you want to print this routing table "
75                     "you must create a trace file in your tcl script",
76                     CURRENT_TIME, ra_addr());
77         }
78         return TCL_OK;
79     } }
80     else if (argc == 3) {
81         // Obtains corresponding dmux to carry packets to upper layers
82         if (strcmp(argv[1], "port-dmux") == 0) {
83             dmux_ = (PortClassifier*) TclObject::lookup(argv[2]);
84             if (dmux_ == 0) {
85                 fprintf(stderr, "%s: %s lookup of %s failed\n", __FILE__, argv[1],
86                         argv[2]);
87                 return TCL_ERROR;
88             }
89             return TCL_OK; }

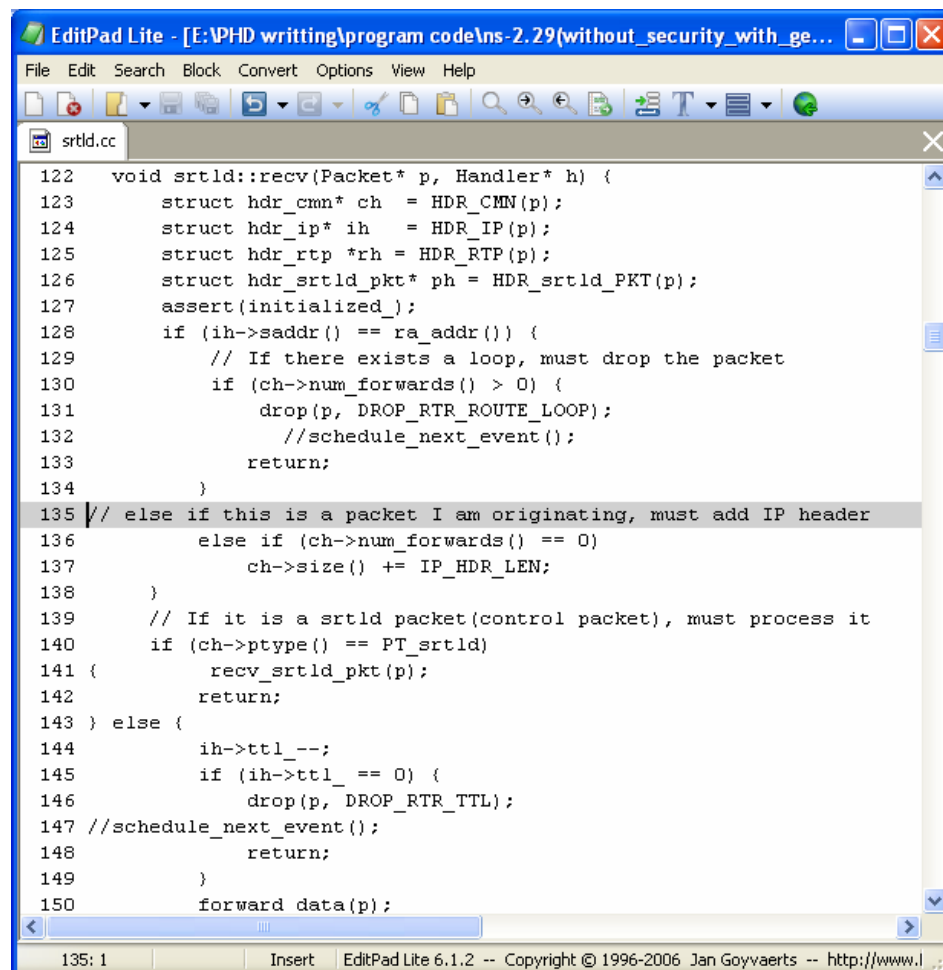
```

**Figure B.6** SRTLD command function

### B.5.3 Receiving Packet

Next function is `recv()` and it is invoked whenever the routing agent receives a packet. Every Packet has a common header called `hdr_cmn` defined in `common/packet.h`. To access this header there is a macro like the one we defined before for SRTLD packet type, and we use it at line 123 in figure B.7. Line 124 does the same but in order to get IP header, `hdr_ip`, described in `ip.h`. First thing, the receiving packet is checked to drop the loop packet that sent by same node. In addition, if the packet has been generated within the node (by upper layers of the node) we should add to packet's length the overhead that the routing protocol is adding (in bytes). We assume SRTLD works over IP as shown in lines 136-137.

When the received packet is of type `PT_srtd` then we will call `recv_srtd_pkt()` to process it (lines 140-141). If it is a data packet then we should forward it (if it is destined to other node) or to deliver it to upper layers (if it was a broadcast packet or was destined to receiving node), unless TTL reached zero. Lines 143-150 do what we have just described making use of the `forward_data()` function. The `drop()` function is used for dropping packets. Its arguments are a pointer to the packet itself and a constant giving the reason for discarding it. There exist several of these constants.



```

122 void srtd::recv(Packet* p, Handler* h) {
123     struct hdr_cmh* ch = HDR_CMH(p);
124     struct hdr_ip* ih = HDR_IP(p);
125     struct hdr_rtp *rh = HDR_RTP(p);
126     struct hdr_srtd_pkt* ph = HDR_srtd_PKT(p);
127     assert(initialized_);
128     if (ih->saddr() == ra_addr()) {
129         // If there exists a loop, must drop the packet
130         if (ch->num_forwards() > 0) {
131             drop(p, DROP_RTR_ROUTE_LOOP);
132             //schedule_next_event();
133             return;
134         }
135     // else if this is a packet I am originating, must add IP header
136     else if (ch->num_forwards() == 0)
137         ch->size() += IP_HDR_LEN;
138     }
139     // If it is a srtd packet (control packet), must process it
140     if (ch->ptype() == PT_srtd)
141     {
142         recv_srtd_pkt(p);
143         return;
144     } else {
145         ih->ttl--;
146         if (ih->ttl == 0) {
147             drop(p, DROP_RTR_TTL);
148             //schedule_next_event();
149             return;
150         }
151         forward_data(p);

```

**Figure B.7** SRTLD receive function

#### **B.5.4 Receiving SRTLD Packet**

If the routing agent has received a srtld packet, `recv_srtld_pkt()` is invoked. Lines 160-163 in Figure B.8 shows that IP header and SRTLD packet header is specified. After that, we make sure source and destination ports are `RT_PORT` at lines 166-167. This constant is defined in `common/packet.h` and it equals 25B. This port is reserved to attach the routing agent. After that, the SRTLD packet must be processed according to the routing protocol's specification as follows: firstly, when the SRTLD packet is received, the distance between the packet receiver and the destination is calculated and compared with the distance between the packet transmitter and the destination. If the former is less, the source code will check the quadrant of the packet transmitter and compare with the quadrant of the destination. Otherwise, it will return.



```

EditPad Lite - [E:\PHD writing\program code\ns-2.29(without_security_with_ge...
File Edit Search Block Convert Options View Help
srtld.cc
159 void srtld::recv_srtld_pkt(Packet* p) {
160     struct hdr_cmh* ch = HDR_CMH(p);
161     struct hdr_ip* ih = HDR_IP(p);
162     struct hdr_srtld_pkt* ph = HDR_srtld_PKT(p);
163     struct hdr_rtp *rh = HDR_RTP(p);
164     // All routing messages are sent from and to port RT_PORT,
165     // so we check it. default value of RT_PORT is 255 in file comm
166     assert(ih->sport() == RT_PORT);
167     assert(ih->dport() == RT_PORT);
168     double mynode_x, mynode_y, Dsd, Dmd, x=0, y=0, z=0;
169     Node *node=Node::get_node_by_address(ra_addr());
170     ((MobileNode *)node)->getLoc(&x, &y, &z);
171     mynode_x=x; mynode_y=y;
172     int quadsrc, quaddest[2], j, flag=0;
173     for(j=0; j<2; j++) {quaddest[j]=-1; j=0; /* ... processing of srtld pa
174         Dsd=sqrt((dstx-ph->srcx)*(dstx-ph->srcx)+(dsty-ph->srcy)*(dsty-p
175         Dmd=sqrt((dstx-mynode_x)*(dstx-mynode_x)+(dsty-mynode_y)*(dsty-m
176     if(Dsd<Dmd)
177     { //to determine where is the destination
178         if ((dstx >= ph->srcx) && (dsty >= ph->srcy))
179             {quaddest[0] = 1; j++;}
180         if ((dstx >= ph->srcx) && (dsty <= ph->srcy))
181             {quaddest[j] = 4; j++;}
182         if ((dstx <= ph->srcx) && (dsty >= ph->srcy))
183             {quaddest[j] = 2; j++;}
184         if ((dstx <= ph->srcx) && (dsty <= ph->srcy))
185             quaddest[j] = 3;
186         if ((ph->srcx >= mynode_x) && (ph->srcy >= mynode_y))
187             quadsrc = 1;
188         else if ((ph->srcx >= mynode_x) && (ph->srcy <= mynode_y))
189             quadsrc = 4;
190         else if ((ph->srcx <= mynode_x) && (ph->srcy >= mynode_y))
191             quadsrc = 2;
192         else if ((ph->srcx <= mynode_x) && (ph->srcy <= mynode_y))
193             quadsrc = 3;
173: 18      Insert      Find the next search match after the current position of the text cursor

```

**Figure B.8** Receive SRTLD control packet

### B.5.5 Sending SRTLD Packet

To send a packet we need first to allocate it. We use the `allocpkt()` function for that. This function is defined for all agents. Then we get common, IP and SRTLD packet headers as lines 321-323 in Figure B.9. The headers of common, IP and SRTLD packet should be filled with specific values. SRTLD packet header is filled in lines 327-330. In SRTLD, source address of the agent, length in bytes of the message, a sequence number, location of the source, and the remaining energy are

needed. These fields are completely dependent on SRTLD packet specification. The common header in NS-2 has several fields. We focus only on those in which we are interested (lines 331-336). We need to set the packet type to a SRTLD packet (line 331). We also assign the packet direction in line 332. As we are sending a packet, it is going down, what is represented by “hdr\_cmn::DOWN” constant. The size of the packet is given in line 333. It is in bytes and this is the value used for NS-2 computations. What we mean is that it does not matter real size of your `hdr_protoname_pkt` structure. To calculate things such as propagation delay NS-2 will use the value you put in here. Continuing with common header, in line 334, error in transmission is selected. Line 335 assigns the next hop to which the packet must be sent to. This is a very important field, and in SRTLD protocol it is established as `IP_BROADCAST` because we want all of the neighbouring nodes to receive this control packet. That constant is defined in `common/ip.h` and you can check there for other macros. The last field we fill is the address type. It can be `NS_AF_NONE`, `NS_AF_ILINK` or `NS_AF_INET`. We choose `NS_AF_INET` because we are implementing an Internet protocol. Now we proceed with the configuration of the IP header. It is very simple as we can see in lines 337-341. There is a new constant called `IP_DEF_TTL`, which is defined in `common/ip.h` and represents the default TTL value for IP packets. The IP header has other fields used for IPv6 simulations, but we do not need them for SRTLD. The packet sending needed to be scheduled. In fact, sending a packet is equivalent to schedule it at a certain time. Line 342 shows how to send a packet introducing some jitter. The `Packet` class inherits from the `connector` class that has a reference to a `TclObject` called `target_`. This is the handler which will process the event, and is passed as an argument to the `schedule ()` function.

```

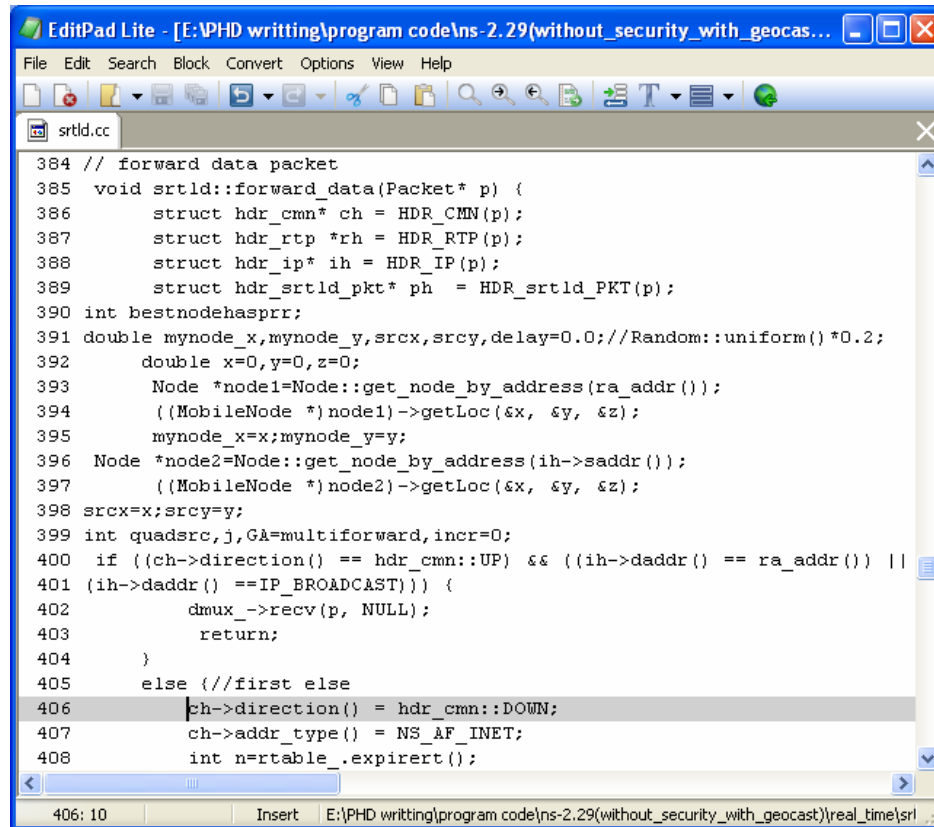
318 // send when the timer expire
319 void srtld::send_srtld_pkt() {
320     Packet* p                = allocpkt();
321     struct hdr_cmn* ch        = HDR_CMN(p);
322     struct hdr_ip* ih         = HDR_IP(p);
323     struct hdr_srtld_pkt* ph  = HDR_srtld_PKT(p);
324     double x=0,y=0,z=0;
325     Node *node=Node::get_node_by_address(ra_addr());
326     ((MobileNode *)node)->getLoc(&x, &y, &z);
327     ph->srcx=x;ph->srcy=y;ph->Tenergy=node->energy_model()->energy();
328     ph->pkt_src()              = ra_addr();
329     ph->pkt_len()              = 13;
330     ph->pkt_seq_num()          = seq_num++;
331     ch->ptype()                = PT_srtld;
332     ch->direction()            = hdr_cmn::DOWN;
333     ch->size()                  = IP_HDR_LEN + ph->pkt_len();
334     ch->error()                 = 0;
335     ch->next_hop()              = IP_BROADCAST;
336     ch->addr_type()             = NS_AF_NONE;
337     ih->saddr()                 = ra_addr();
338     ih->daddr()                 = IP_BROADCAST;
339     ih->sport()                 = RT_PORT;
340     ih->dport()                 = RT_PORT;
341     ih->ttl()                   = IP_DEF_TTL;
342     Scheduler::instance().schedule(target_, p, JITTER1);

```

**Figure B.9** Send SRTLD control packet

## B.5.6 Forwarding Data Packet

The `forward_data()` is responsible to forward data packet and decides whether a packet has to be delivered to the upper-layer agents or to be forwarded to other node. Lines 400-404 in Figure B.10 check the received data packet. When the received data packet is an incoming packet and destination address is the node itself or broadcast, then the node's `dmux_` (it is a `PortClassifier` object) is used to accept the incoming packet. Otherwise, the packet must forward. This is accomplished by properly setting the common header with as we do in lines 405-408. If the packet is a broadcast one, then next hop will be filled accordingly. If not, the next hop will be filled from the neighbour table as the algorithm of SRTLD explained in chapter 4.



```

384 // forward data packet
385 void srtld::forward_data(Packet* p) {
386     struct hdr_cmn* ch = HDR_CMN(p);
387     struct hdr_rtp* rh = HDR_RTP(p);
388     struct hdr_ip* ih = HDR_IP(p);
389     struct hdr_srtld_pkt* ph = HDR_srtld_PKT(p);
390     int bestnodehaspr;
391     double mynode_x, mynode_y, srcx, srcy, delay=0.0; //Random::uniform()*0.2;
392     double x=0, y=0, z=0;
393     Node *node1=Node::get_node_by_address(ra_addr());
394     ((MobileNode *)node1)->getLoc(&x, &y, &z);
395     mynode_x=x; mynode_y=y;
396     Node *node2=Node::get_node_by_address(ih->saddr());
397     ((MobileNode *)node2)->getLoc(&x, &y, &z);
398     srcx=x; srcy=y;
399     int quadsrc, j, GA=multiforward, incr=0;
400     if ((ch->direction() == hdr_cmn::UP) && ((ih->daddr() == ra_addr()) ||
401     (ih->daddr() == IP_BROADCAST))) {
402         dmux_->recv(p, NULL);
403         return;
404     }
405     else //first else
406     {
407         ch->direction() = hdr_cmn::DOWN;
408         ch->addr_type() = NS_AF_INET;
409         int n=rttable_.expirert();

```

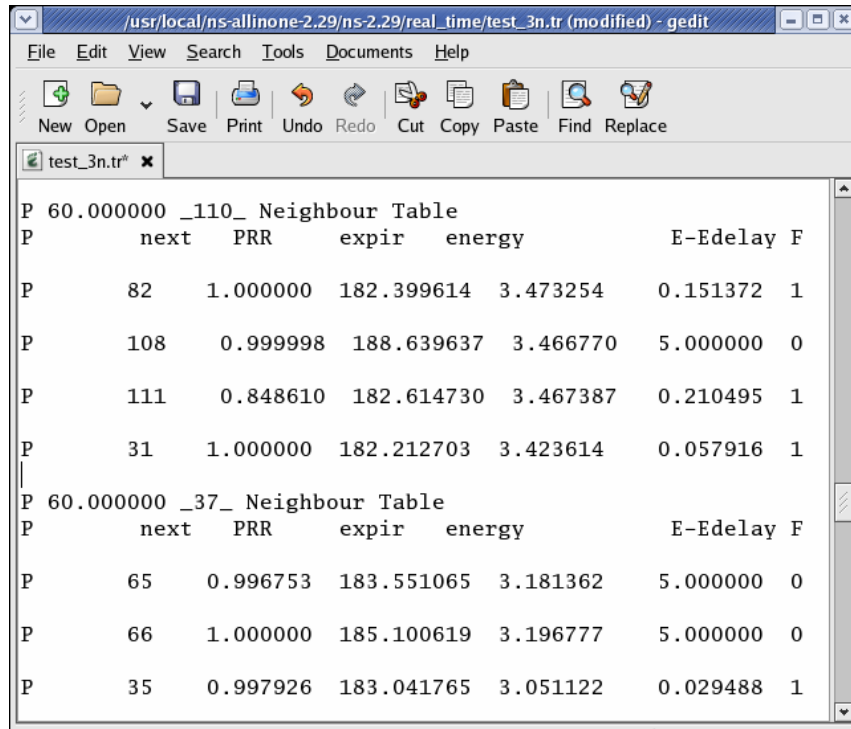
**Figure B.10** Forward data packet

## B.6 Developing Neighbour Table

SRTL D does not need a routing table, but it has neighbour table, which it uses for records information about one-hop neighbour. The neighbour table class can be implemented as a different class or as any other data structure (e.g. a hash table). a class encapsulating the functionality is explained and a neighbour table is created. For each entry in neighbour table, the information such as node id, remaining power, one-hop end-to-end delay, PRR, forward flag, location information and expiry time are needed to store neighbour information. Figure B.11 shows the snapshot of the SRTL D neighbour table. Doubly linked list is used as the storage structure because it has simple way to delete and add a record to the neighbour table.

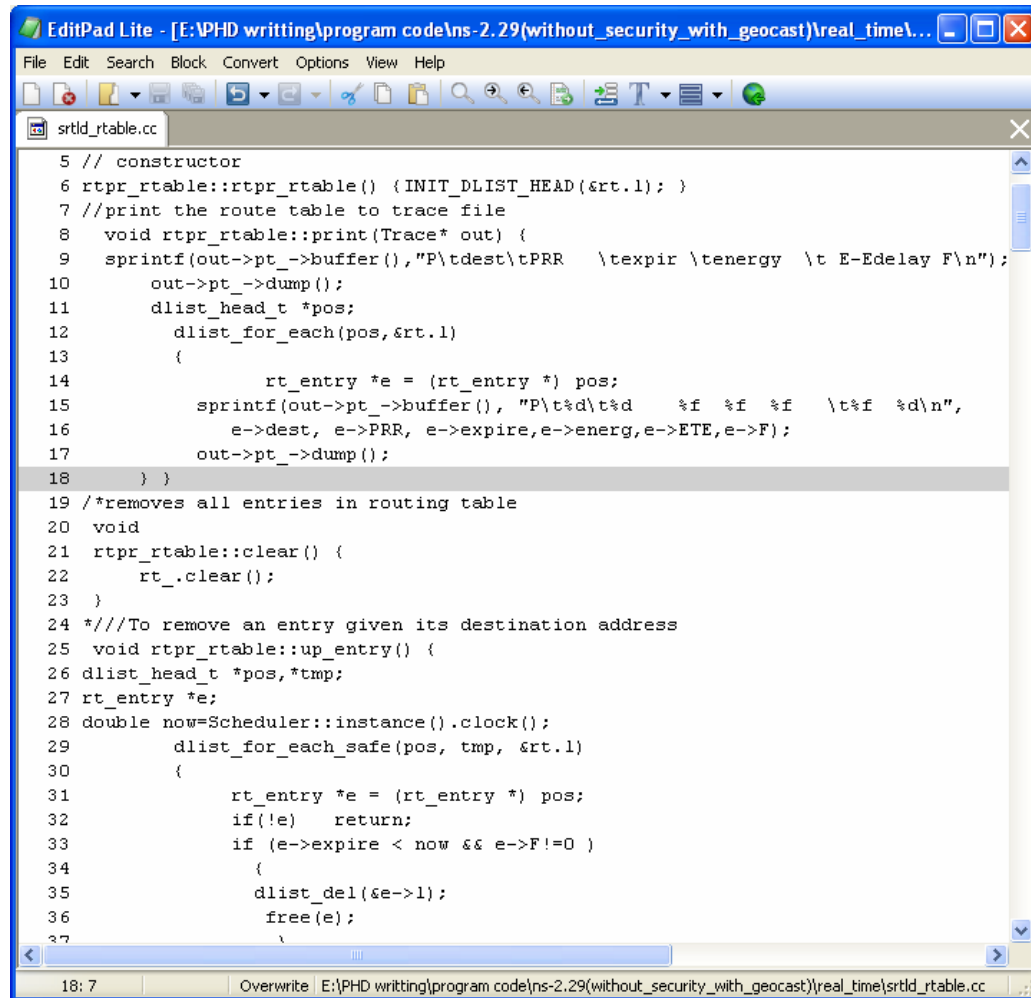
The constructor of the neighbour table class defines and unitizes the Double linked list that is required for implementing the neighbour table as shown in line 6 of Figure B.12. The print () function will dump the contents of the node's neighbour

table to the trace file as shown in lines 7-18 of figure B.12. To do that we had used the trace class that mentioned above. The function at lines 25- 37 removes an entry that its destination address is given. The add function is used to insert new neighbour record or update old neighbour record. It considers the limitation of the neighbour table as described in chapter 4. The lookup function is used to search about the neighbour record using address of neighbour.



P 60.000000 _110_ Neighbour Table						
	next	PRR	expir	energy	E-Delay	F
P	82	1.000000	182.399614	3.473254	0.151372	1
P	108	0.999998	188.639637	3.466770	5.000000	0
P	111	0.848610	182.614730	3.467387	0.210495	1
P	31	1.000000	182.212703	3.423614	0.057916	1
P 60.000000 _37_ Neighbour Table						
	next	PRR	expir	energy	E-Delay	F
P	65	0.996753	183.551065	3.181362	5.000000	0
P	66	1.000000	185.100619	3.196777	5.000000	0
P	35	0.997926	183.041765	3.051122	0.029488	1

**Figure B.11** SRTLD neighbour table



```

EditPad Lite - [E:\PHD writing\program code\ns-2.29(without_security_with_geocast)\real_time\...
File Edit Search Block Convert Options View Help

srtld_rtable.cc

5 // constructor
6 rtpr_rtable::rtpr_rtable() {INIT_DLIST_HEAD(&rt.l); }
7 //print the route table to trace file
8 void rtpr_rtable::print(Trace* out) {
9     sprintf(out->pt->buffer(), "P\tdest\tPRR    \texpir \tenergy \t E-Edelay F\n");
10    out->pt->dump();
11    dlist_head_t *pos;
12    dlist_for_each(pos, &rt.l)
13    {
14        rt_entry *e = (rt_entry *) pos;
15        sprintf(out->pt->buffer(), "P\t%d\t%d    %f %f %f \t%f %d\n",
16            e->dest, e->PRR, e->expire, e->energy, e->ETE, e->F);
17        out->pt->dump();
18    } }

19 /*removes all entries in routing table
20 void
21 rtpr_rtable::clear() {
22     rt_.clear();
23 }
24 *///To remove an entry given its destination address
25 void rtpr_rtable::up_entry() {
26 dlist_head_t *pos, *tmp;
27 rt_entry *e;
28 double now=Scheduler::instance().clock();
29     dlist_for_each_safe(pos, tmp, &rt.l)
30     {
31         rt_entry *e = (rt_entry *) pos;
32         if(!e) return;
33         if (e->expire < now && e->F!=0 )
34         {
35             dlist_del(&e->l);
36             free(e);
37         }

```

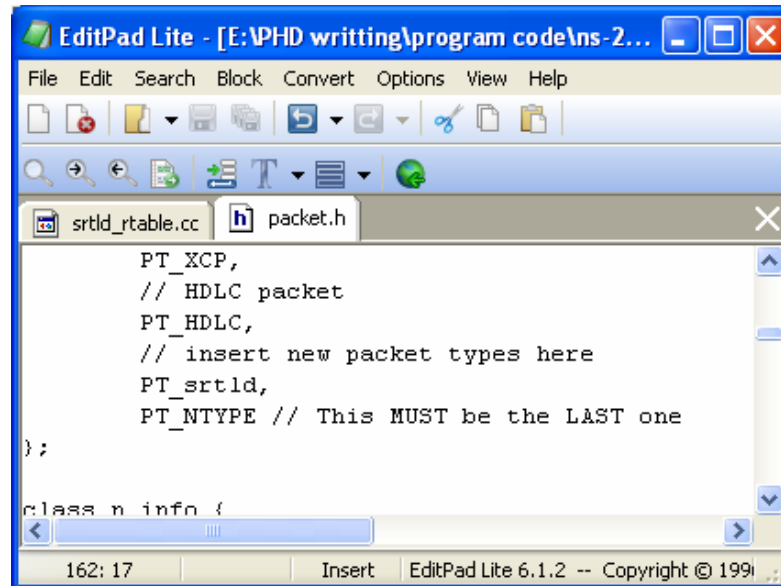
**Figure B.12** Functions of neighbour table class

## B.7 Integrated SRTLD Code into NS-2

A routing agent is implemented for SRTLD protocol inside NS2. However, there are some changes need to do in order to integrate SRTLD inside NS-2 simulator such as packet type declaration, tracing support, tcl library, priority queue, radio propagation model and makefile. The following subsections describe them in details.

### B.7.1 Packet Type declaration

A constant `PT_srtld` had used to indicate SRTLD packet type. It had defined inside file `common/packet.h`. `PT_srtld` is added to the `packet_t` enumeration, where all packet types are listed. As shown in Figure B.13 (a). Just below in same file there is definition of `p_info` class. Inside constructor, we will provide a textual name for SRTLD packet type as `srtld` as shown in Figure B.13 (b).



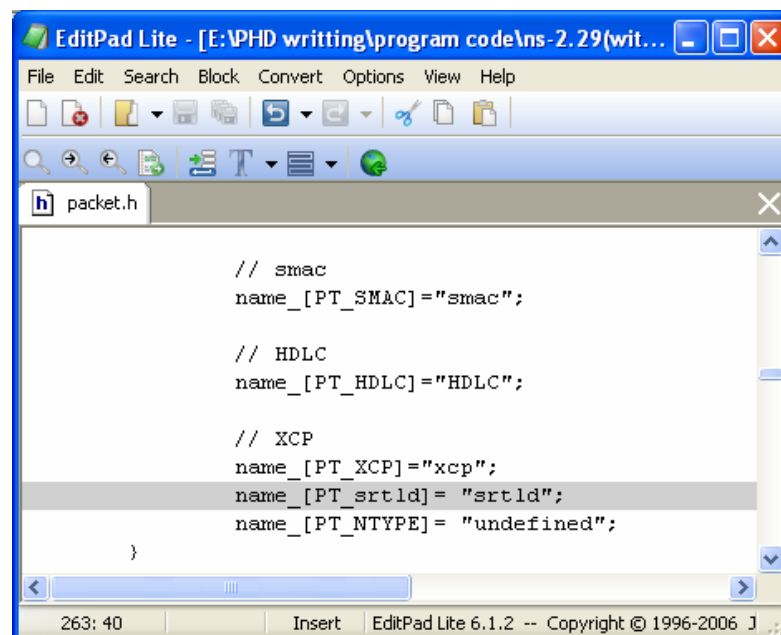
```

EditPad Lite - [E:\PHD writing\program code\ins-2...
File Edit Search Block Convert Options View Help
srtld_rtable.cc packet.h
PT_XCP,
// HDLC packet
PT_HDLC,
// insert new packet types here
PT_srtld,
PT_NTTYPE // This MUST be the LAST one
};

class p_info {
162: 17 Insert EditPad Lite 6.1.2 -- Copyright © 199

```

(a)



```

EditPad Lite - [E:\PHD writing\program code\ins-2.29(wit...
File Edit Search Block Convert Options View Help
packet.h
// smac
name_[PT_SMAC]="smac";

// HDLC
name_[PT_HDLC]="HDLC";

// XCP
name_[PT_XCP]="xcp";
name_[PT_srtld] = "srtld";
name_[PT_NTTYPE] = "undefined";
}
263: 40 Insert EditPad Lite 6.1.2 -- Copyright © 1996-2006 J

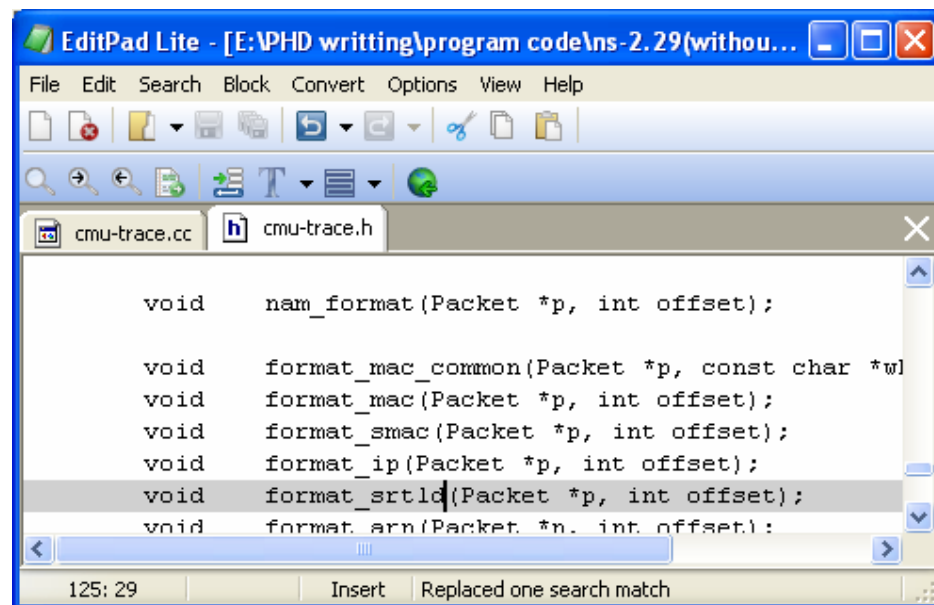
```

(b)

**Figure B.13** Declaration of SRTLD packet: (a) Packet type; (b) Named packet type

### B.7.2 Tracing Support

The purpose of the simulation is to get a trace file describing what happened during execution. A trace object is used to write wanted information of a packet every time it is received, sent or dropped. To log information regarding SRTLD packet type we implement the `format_srtld()` function inside the CMUTrace class. CMUTrace provides trace support for wireless simulations. To insert the SRTLD trace code, `trace/cmu-trace.h` file is edited. The `format_srtld()` function is added as shown in Figure B.14.



**Figure B.14** Declaration of SRTLD trace file format

The next piece of code in Figure B.15 (extracted from `trace/cmu-trace.cc`) shows different types of traces. We can deduce from above code that there are three different trace formats: tagged traces, new format traces, and classical traces. The syntaxes followed by each, although different is very easy and intuitive as you can tell. Both in tagged and new trace formats there exists tag used to identify each field of information being printed. We have decided to use “S” as source address (origin), “D” as destination of corresponding packet. In order to call this recently created function we must change the `format()` in `trace/cmu-trace.cc`.

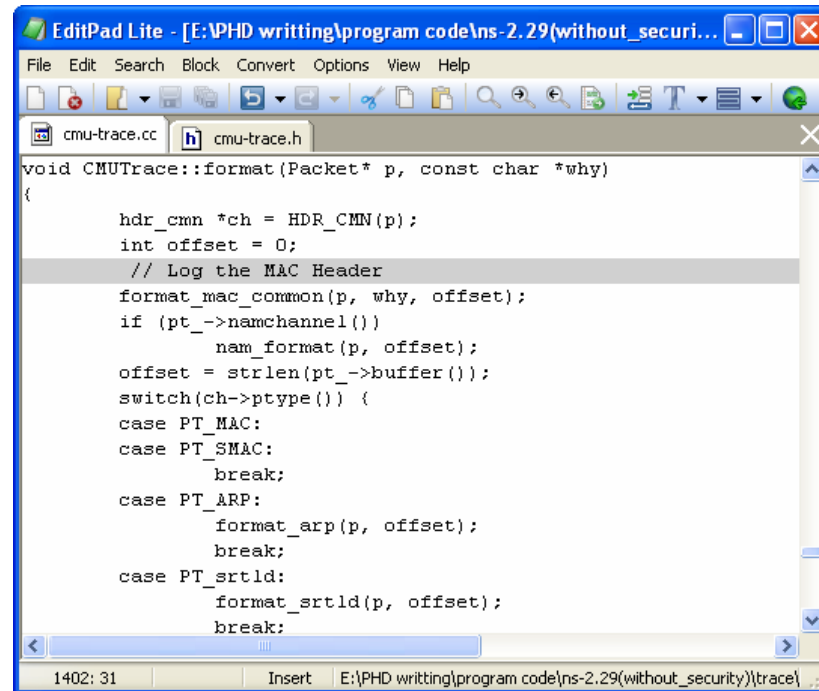


```

EditPad Lite - [E:\PHD writting\program code\ns-2.29(without_securit...
File Edit Search Block Convert Options View Help
cmu-trace.cc cmu-trace.h
//***** real-time routing
void CMUTrace::format_srtld(Packet *p, int offset)
{
    struct hdr_srtld_pkt* ph = HDR_srtld_PKT(p);
    if (pt_>tagged()) {
        sprintf(pt_>buffer() + offset,
            "-srtld:o %d -srtld:s %d -srtld:l %d ",
            ph->pkt_src(),
            ph->pkt_seq_num(),
            ph->pkt_len(), ph->pkt_dst());
    }
    else if (newtrace_) {
        sprintf(pt_>buffer() + offset,
            "-P srtld -Psr %d -Pse %d -Ple %d -Pds %d",
            ph->pkt_src(),
            ph->pkt_seq_num(),
            ph->pkt_len(), ph->pkt_dst());
    }
    else {
        sprintf(pt_>buffer() + offset,
            "[srtld S:%d %d %d D:%d] ",
            ph->pkt_src(),
            ph->pkt_seq_num(),
            ph->pkt_len(),
            ph->pkt_dst());
    }
}
//*****
326: 6 Insert E:\PHD writting\program code\ns-2.29(without_security)\trace\ci

```

(a)



```

EditPad Lite - [E:\PHD writting\program code\ns-2.29(without_securi...
File Edit Search Block Convert Options View Help
cmu-trace.cc cmu-trace.h
void CMUTrace::format(Packet* p, const char *why)
{
    hdr_cmh *ch = HDR_CMH(p);
    int offset = 0;
    // Log the MAC Header
    format_mac_common(p, why, offset);
    if (pt_>namchannel())
        nam format(p, offset);
    offset = strlen(pt_>buffer());
    switch(ch->ptype()) {
    case PT_MAC:
    case PT_SMAC:
        break;
    case PT_ARP:
        format_arp(p, offset);
        break;
    case PT_srtld:
        format_srtld(p, offset);
        break;
    }
}
1402: 31 Insert E:\PHD writting\program code\ns-2.29(without_security)\trace\

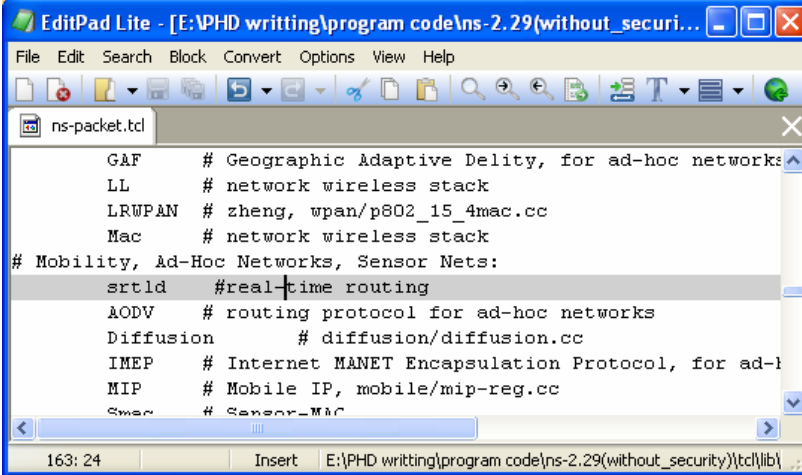
```

(b)

**Figure B.15** Definition of SRTLD trace file format: (a) SRTLD definition; (b) SRTLD calling

### B.7.3 Modified Tcl library

Now we need to do some changes in Tcl files. Actually, we are going to add SRTLD packet type, give default values for binding attributes and providing the requirement infrastructure to create wireless nodes running SRTLD routing protocol. In tcl/lib/ns-packet.tcl, the code in Figure B.16(a) should be located and SRTLD word added to the list. Default values for binding attributes have to be given inside tcl/lib/ns-default.tcl. The code in Figure B.16(b) is added to the end of tcl/lib/ns-default.tcl.

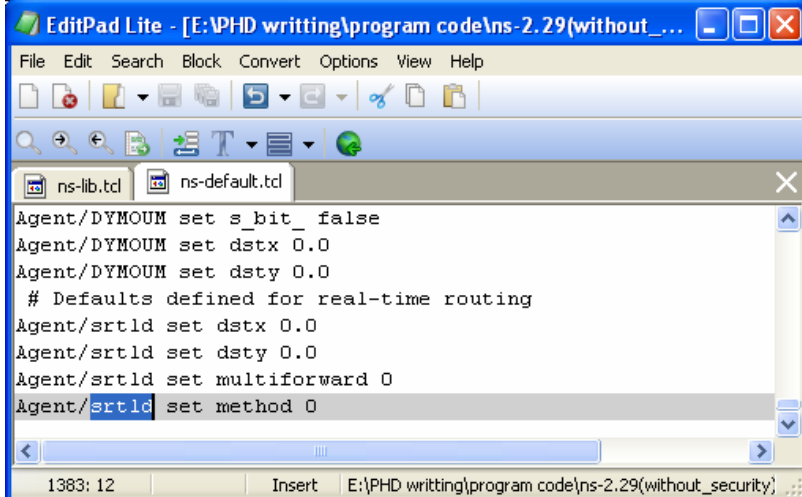


```

GAF      # Geographic Adaptive Delity, for ad-hoc networks
LL       # network wireless stack
LRWPAN   # zheng, wpan/p802_15_4mac.cc
Mac      # network wireless stack
# Mobility, Ad-Hoc Networks, Sensor Nets:
srtld    #real-time routing
AODV     # routing protocol for ad-hoc networks
Diffusion # diffusion/diffusion.cc
IMEP     # Internet MANET Encapsulation Protocol, for ad-h
MIP      # Mobile IP, mobile/mip-reg.cc
Smar     # Sensor-MAC

```

(a)



```

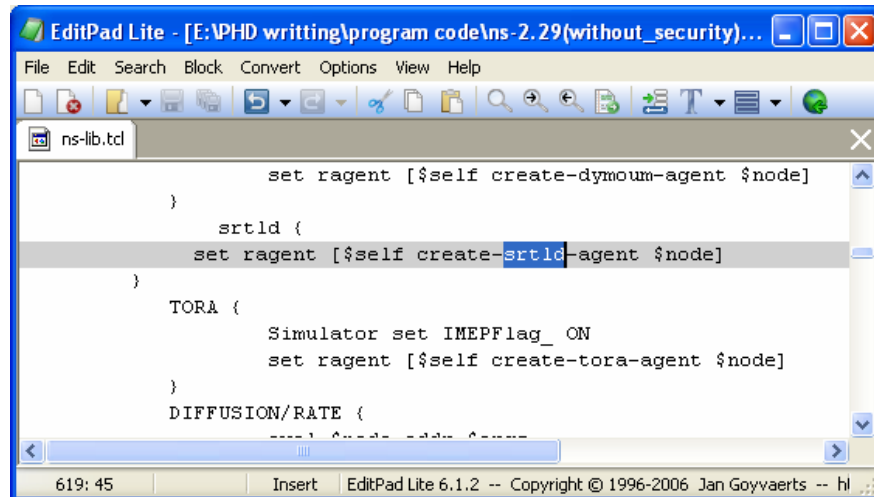
Agent/DYMOUM set s_bit_false
Agent/DYMOUM set dstx 0.0
Agent/DYMOUM set dsty 0.0
# Defaults defined for real-time routing
Agent/srtld set dstx 0.0
Agent/srtld set dsty 0.0
Agent/srtld set multiforward 0
Agent/srtld set method 0

```

(b)

**Figure B.16** Definition of SRTLD in tcl list: (a) Tcl list; (b) Default values of binding attributes

Finally tcl/lib/ns-lib.tcl is modified. A procedure for creating a node is added and will be centred on creating a wireless node with SRTLD as routing protocol. The procedure node calls to the create-wireless-node procedure. This last one, among other tasks is intended to set the routing agent for a node. The procedure is hacked to create an instance of SRTLD protocol as shown in Figure B.17 (a). Then create-srtld-agent will be coded below as shown in the Figure B.17 (b). Line 2256 in Figure B.17 (b) creates a new SRTLD agent with the node's address. This agent is scheduled to start at the beginning of the simulation (line 2257), and is assigned as the node's routing agent in line 2258.

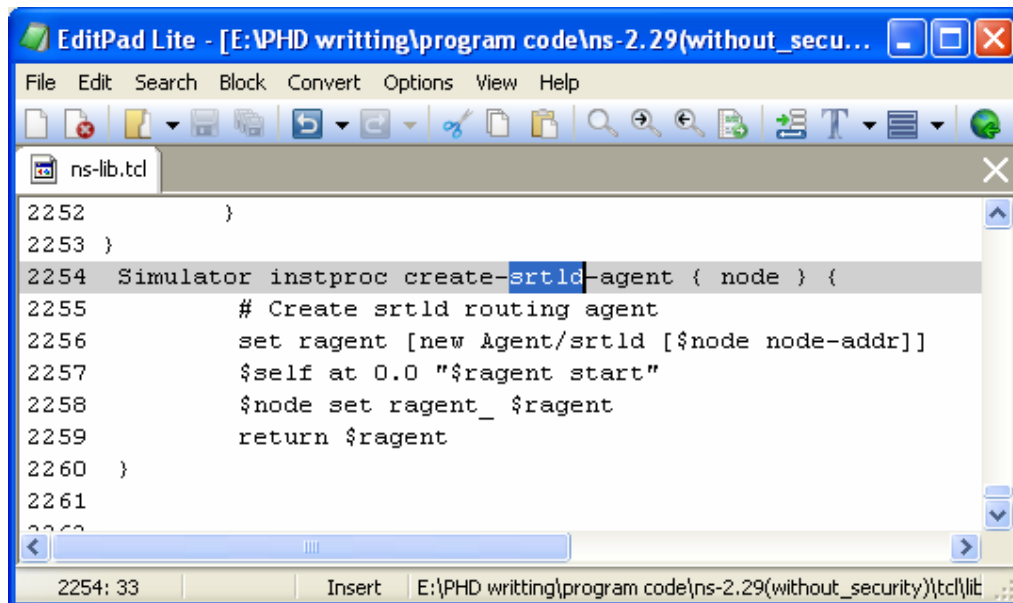


```

set ragent [$self create-dymown-agent $node]
}
srtld {
set ragent [$self create-srtld-agent $node]
}
TORA {
Simulator set IMEPFlag_ON
set ragent [$self create-tora-agent $node]
}
DIFFUSION/RATE {

```

(a)



```

2252 }
2253 }
2254 Simulator instproc create-srtld-agent { node } {
2255     # Create srtld routing agent
2256     set ragent [new Agent/srtld [$node node-addr]]
2257     $self at 0.0 "$ragent start"
2258     $node set ragent_ $ragent
2259     return $ragent
2260 }
2261

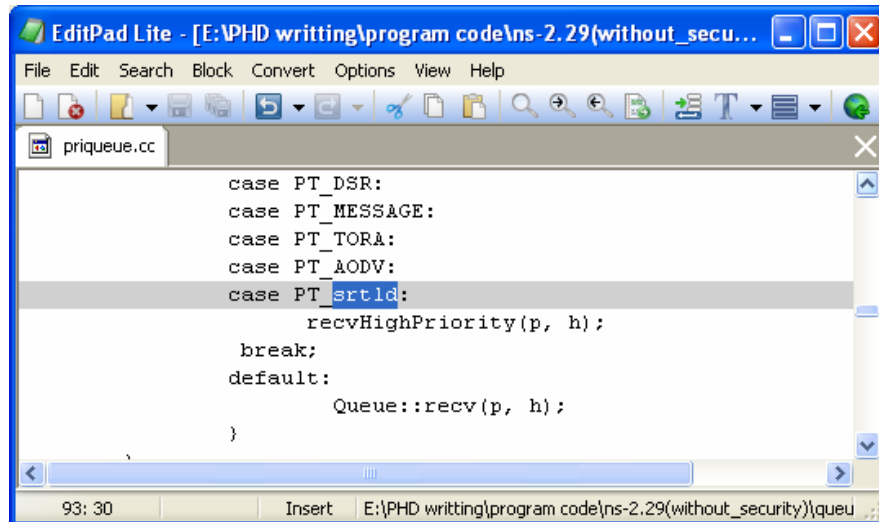
```

(b)

**Figure B.17** Define routing agent for a node: (a) Set routing agent; (b) Create srtld agent

#### B.7.4 Using Priority Queue

It is very likely if priority queues is used in the simulations. This queue type treats routing packets as high priority packets, inserting them at the beginning of the queue. However, we need to tell the PriQueue class that SRTLD packets are routing packets and therefore treated as high priority. The `recv()` function is modified in `queue/priqueue.cc` file as shown in Figure B.18.

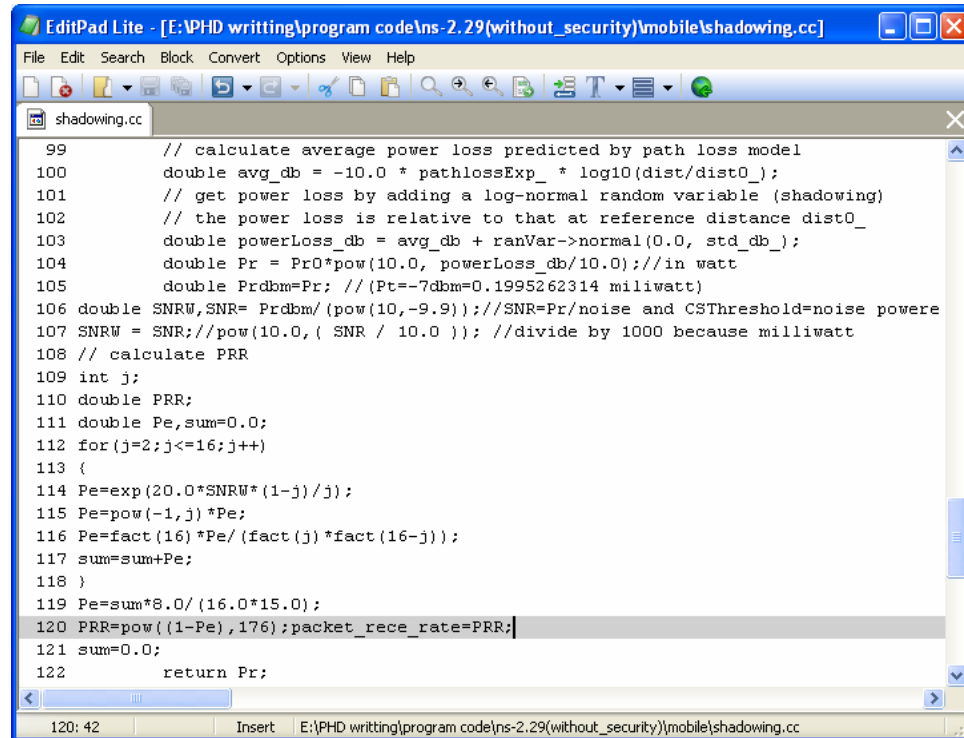


**Figure B.18** Assign the SRTLD packet as high priority

### B.7.5 Modified Shadowing Radio Propagation Model

The radio propagation models had been implemented in NS-2. These models are used to predict the received signal power of each packet. At the physical layer of each wireless node, there is a receiving threshold. When a packet is received, if its signal power is below the receiving threshold, it is marked as error and dropped by the MAC layer. Up to now, there are three propagation models in NS-2, which are the free space model, two-ray ground reflection model and the shadowing model. Their implementation can be found in `mobile/propagation.{cc,h}`, `mobile/tworayground.{cc,h}` and `mobile/shadowing.{cc,h}`. However, the free space model and the two-ray model predict the received power as a deterministic function of distance. They both represent the communication range as an ideal circle. In reality, the received power at certain distance is a random variable due to multipath propagation effects, which is also known as fading effects. In fact, the above two models predicts the mean received power at distance  $d$ . A more general and widely-used model is called the shadowing model [94]. Hence, SRTLD used shadowing model. Figure B.19 (a) shows the implementation of shadowing model with PRR. The calculation of PRR is based on equation 12 in chapter 4. Figure B.19 (b) shows calculation of threshold value. The receiving threshold reflects the specification of

IEEE 802.15.4. If the power received for a frame is below the threshold value, the MAC sub-layer will discard it.

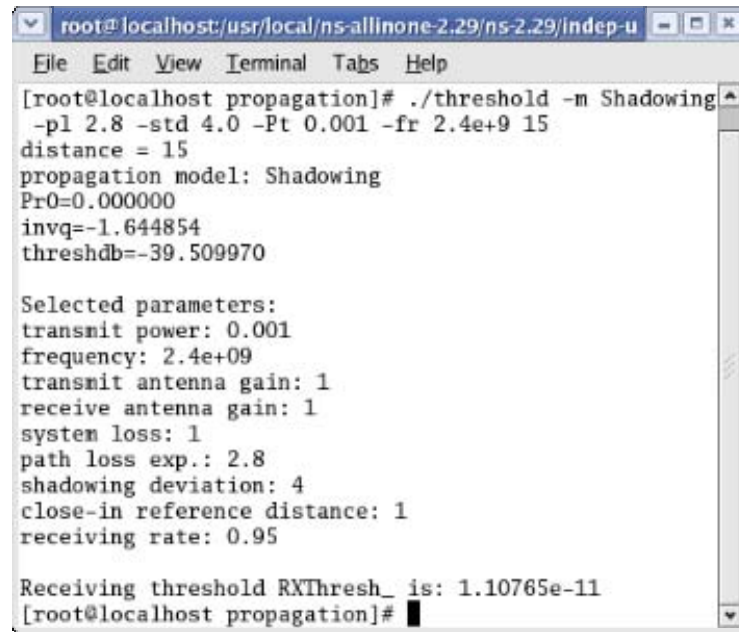


```

99      // calculate average power loss predicted by path loss model
100     double avg_db = -10.0 * pathlossExp_ * log10(dist/dist0_);
101     // get power loss by adding a log-normal random variable (shadowing)
102     // the power loss is relative to that at reference distance dist0_
103     double powerLoss_db = avg_db + ranVar->normal(0.0, std_db_);
104     double Pr = Pr0*pow(10.0, powerLoss_db/10.0); //in watt
105     double Prdbm=Pr; // (Pt=-7dbm=0.1995262314 milliwatt)
106     double SNRW,SNR= Prdbm/(pow(10,-9.9)); //SNR=Pr/noise and CThreshold=noise power
107     SNRW = SNR;//pow(10.0,( SNR / 10.0 )); //divide by 1000 because milliwatt
108     // calculate PRR
109     int j;
110     double PRR;
111     double Pe,sum=0.0;
112     for(j=2;j<=16;j++)
113     {
114         Pe=exp(20.0*SNRW*(1-j)/j);
115         Pe=pow(-1,j)*Pe;
116         Pe=fact(16)*Pe/(fact(j)*fact(16-j));
117         sum=sum+Pe;
118     }
119     Pe=sum*8.0/(16.0*15.0);
120     PRR=pow((1-Pe),176);packet_rece_rate=PRR;
121     sum=0.0;
122     return Pr;

```

(a)



```

root@localhost:/usr/local/ns-allinone-2.29/ns-2.29/indep-u
File Edit View Terminal Tabs Help
[root@localhost propagation]# ./threshold -m Shadowing
-pl 2.8 -std 4.0 -Pt 0.001 -fr 2.4e+9 15
distance = 15
propagation model: Shadowing
Pr0=0.000000
invq=-1.644854
threshdb=-39.509970

Selected parameters:
transmit power: 0.001
frequency: 2.4e+09
transmit antenna gain: 1
receive antenna gain: 1
system loss: 1
path loss exp.: 2.8
shadowing deviation: 4
close-in reference distance: 1
receiving rate: 0.95

Receiving threshold RXThresh_ is: 1.10765e-11
[root@localhost propagation]#

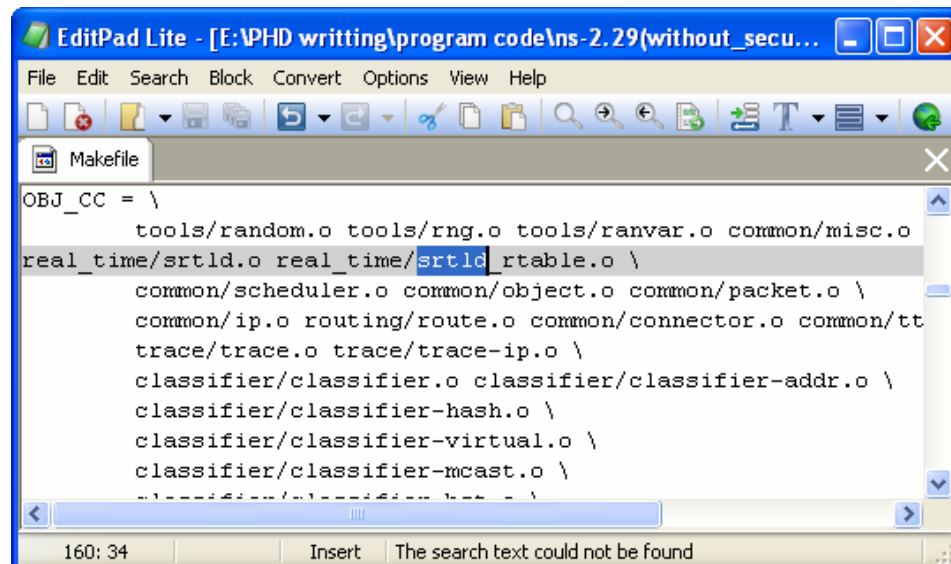
```

(b)

**Figure B.19** Shadowing propagation model: (a) PRR calculation; (b) Threshold calculation

### B.7.6 Modified Makefile

Makefile is edited to add SRTLD object files inside OBJ\_CC variable as depicted in Figure B.20. After that, we can execute “[ns-2.29]\$ make clean & make” to compile and integrate SRTLD routing protocol in NS-2.



```

EditPad Lite - [E:\PHD writing\program code\ns-2.29(without_secu...
File Edit Search Block Convert Options View Help
Makefile
OBJ_CC = \
    tools/random.o tools/rng.o tools/ranvar.o common/misc.o
real_time/srtld.o real_time/srtld_rtable.o \
    common/scheduler.o common/object.o common/packet.o \
    common/ip.o routing/route.o common/connector.o common/tt
trace/trace.o trace/trace-ip.o \
    classifier/classifier.o classifier/classifier-addr.o \
    classifier/classifier-hash.o \
    classifier/classifier-virtual.o \
    classifier/classifier-mcast.o \
    classifier/classifier-bst.o \

```

**Figure B.20** Makefile with SRTLD

## APPENDIX C

### More SRTL D Results

#### C.1 Flow Chart Diagrams

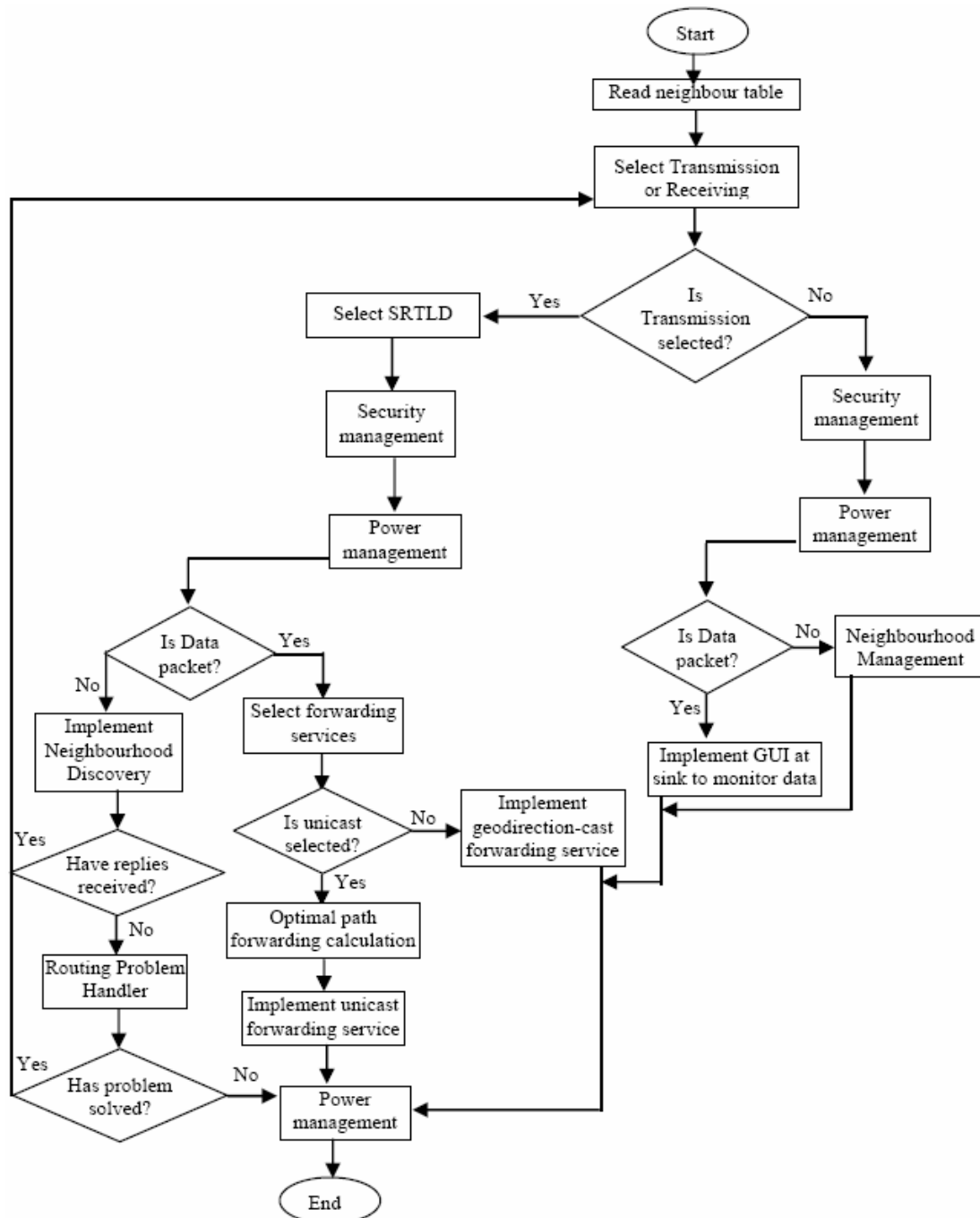
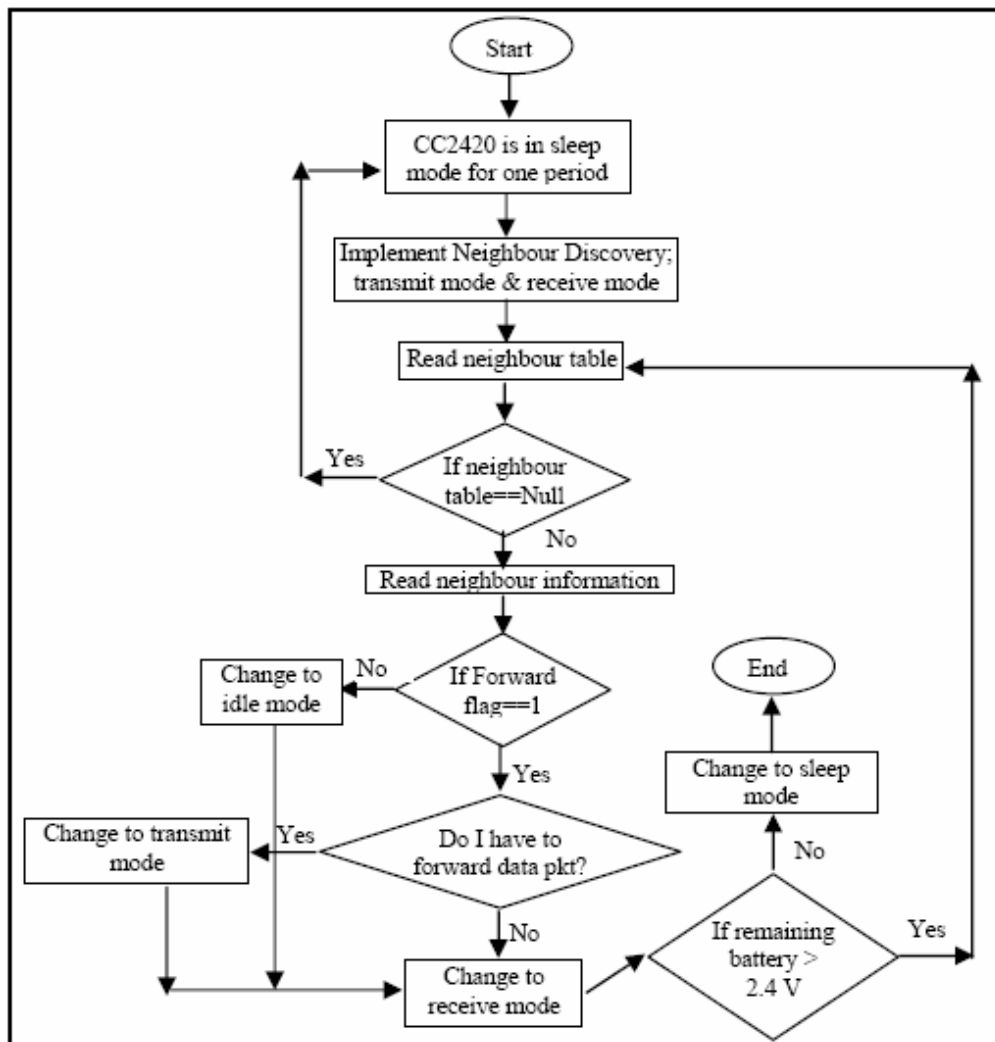


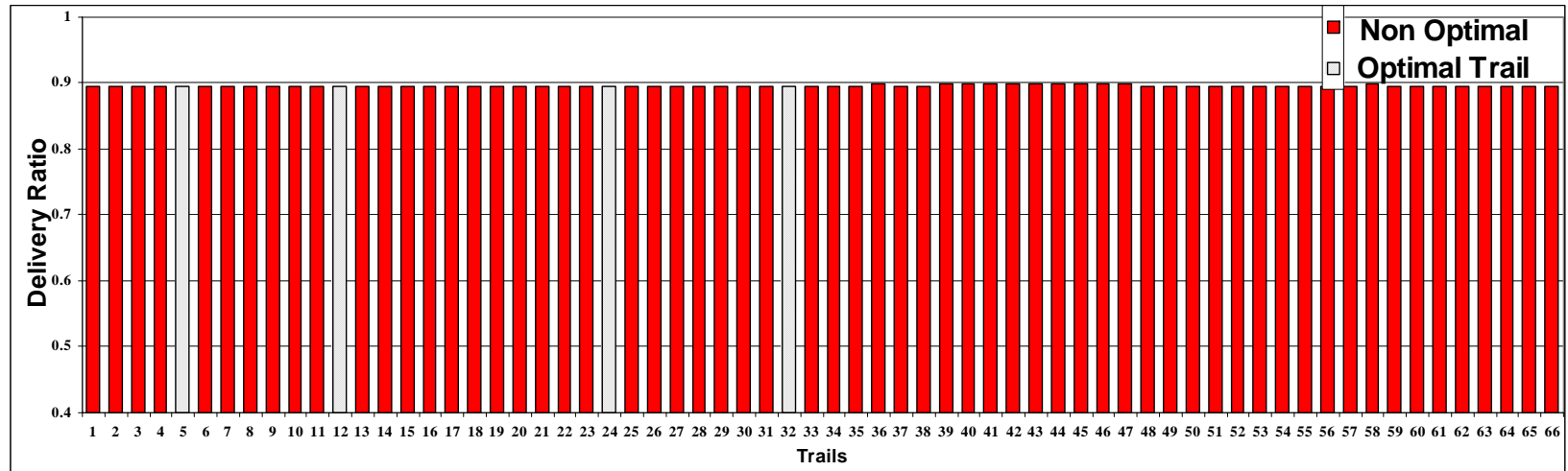
Figure C.1 Flow chart diagram of SRTL D



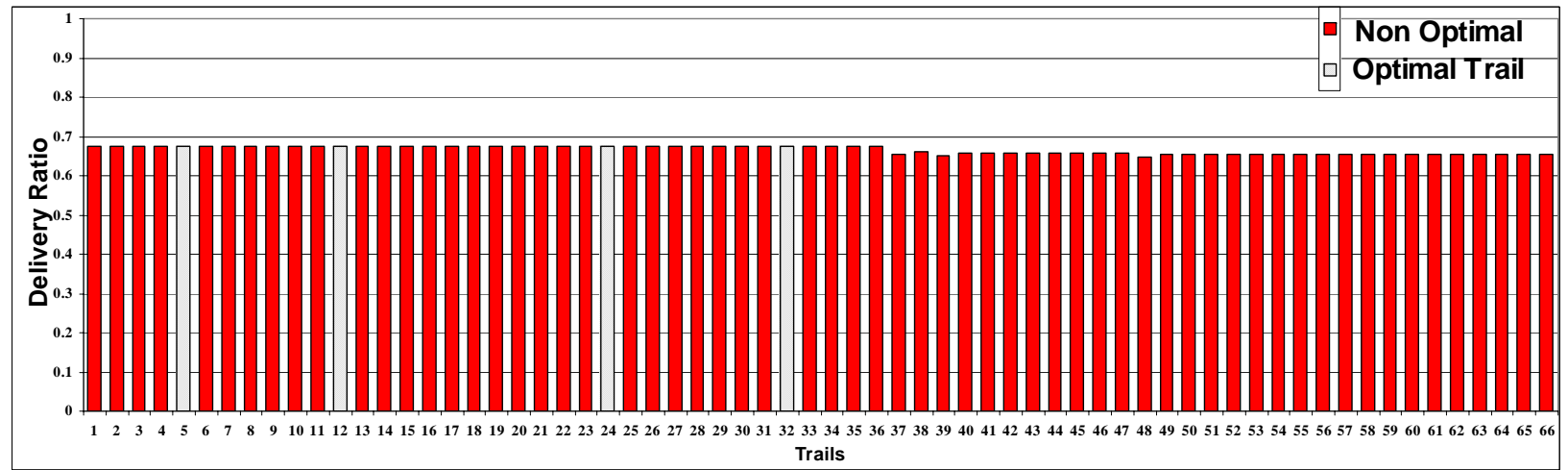


**Figure C.2** Flow chart for power management in SRTL D

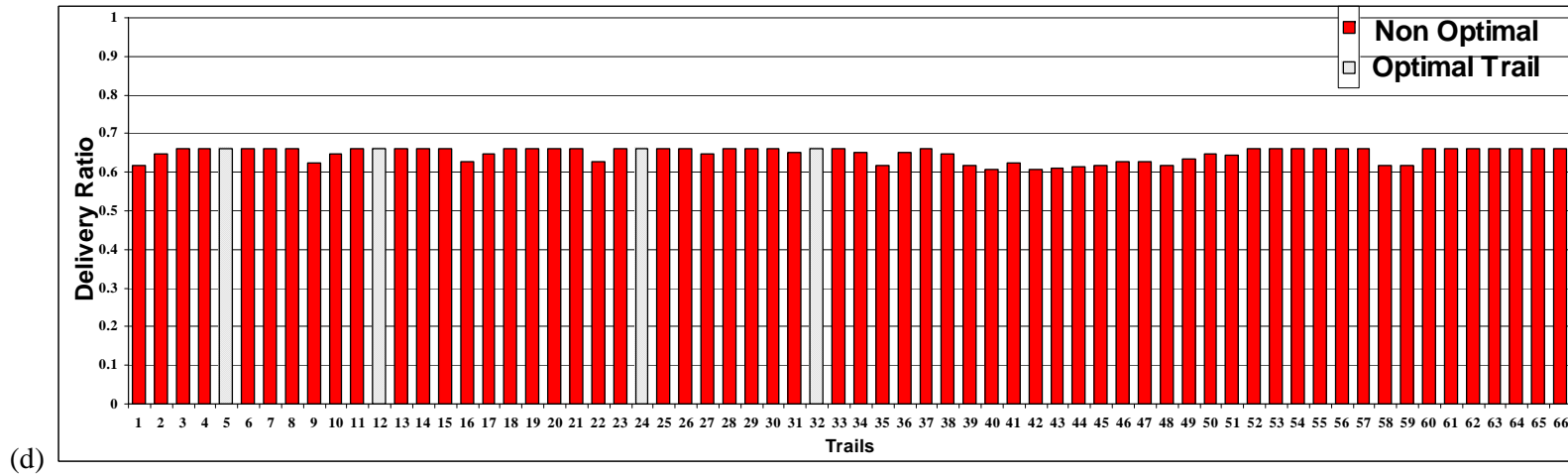
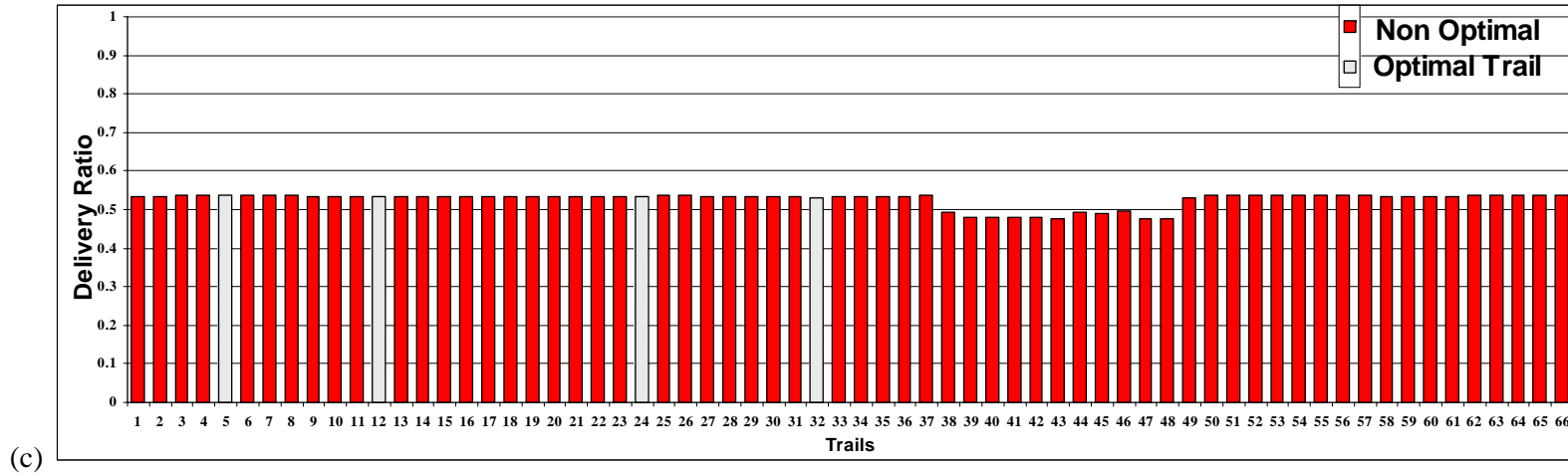
## C.2 Optimal Forwarding Investigation

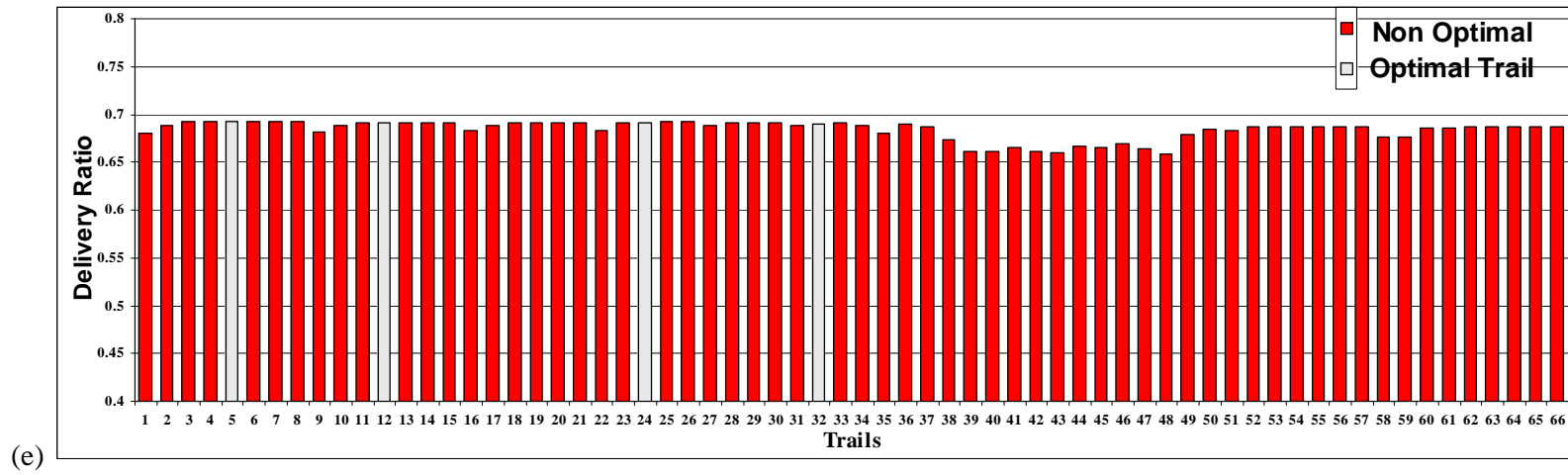


(a)

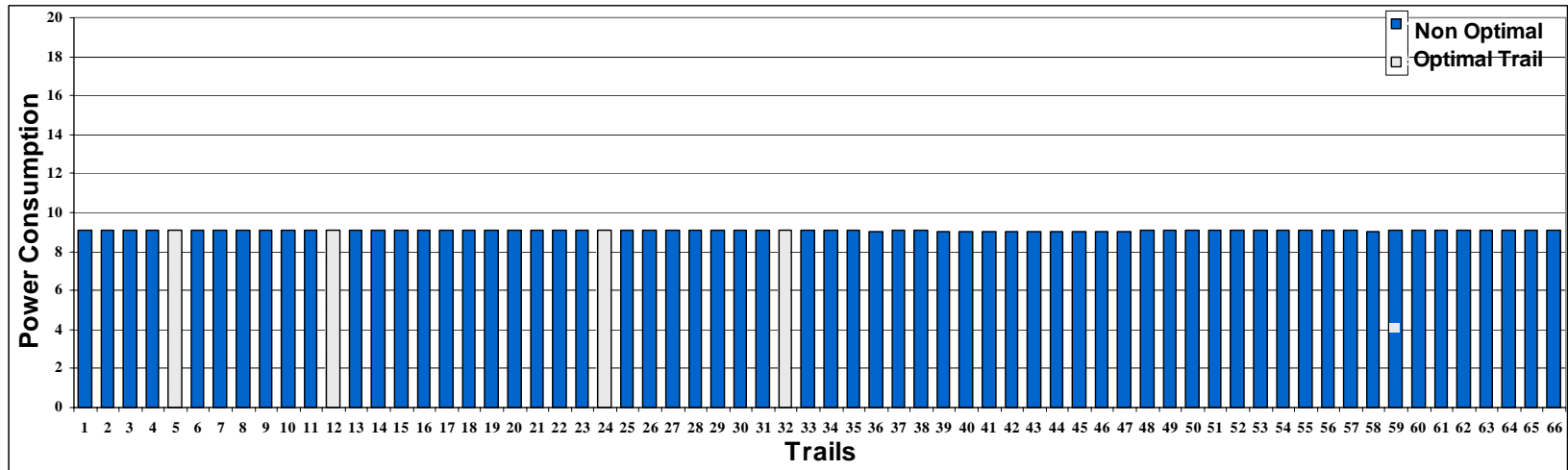


(b)

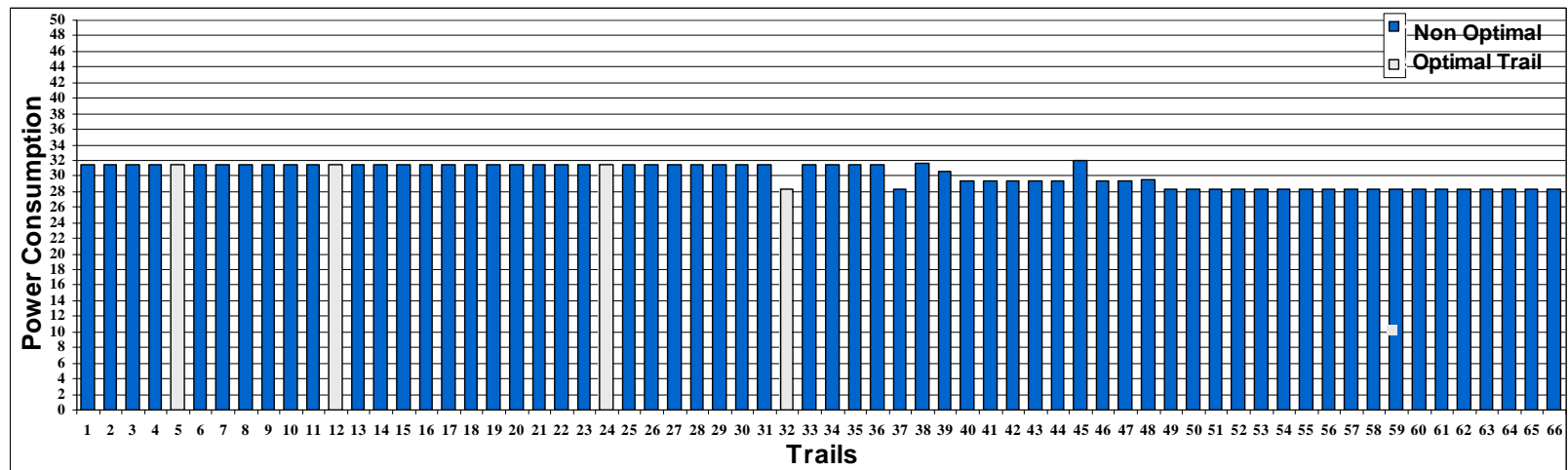




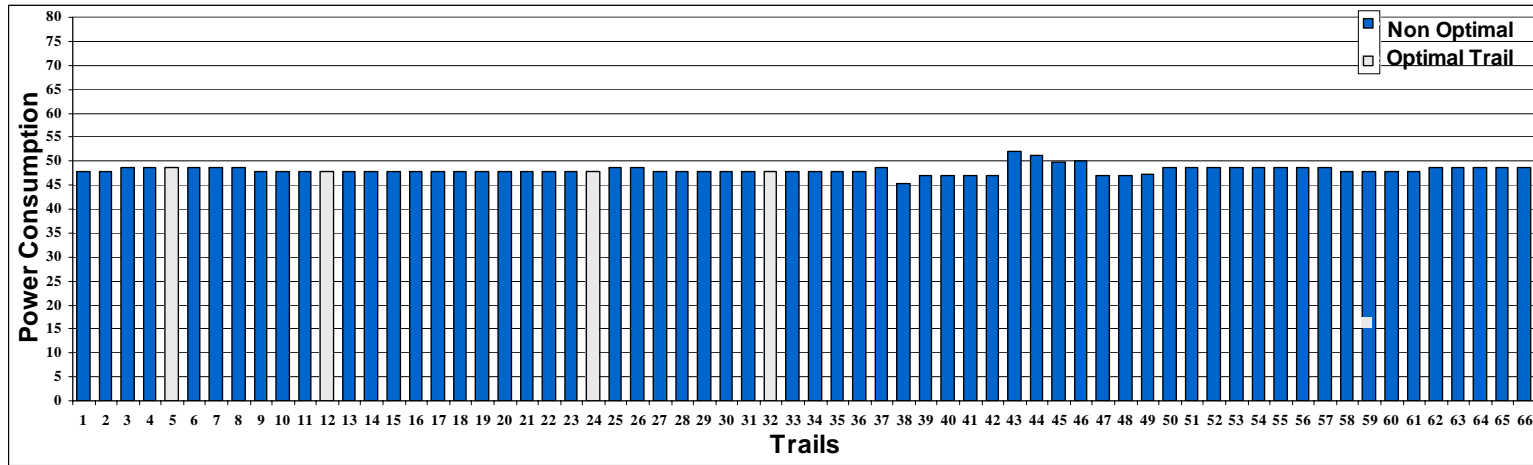
**Figure C.3** Delivery Ratio at 4 packet/s a) low density, b) medium density, c) high density, d) high density with several sources and e) average



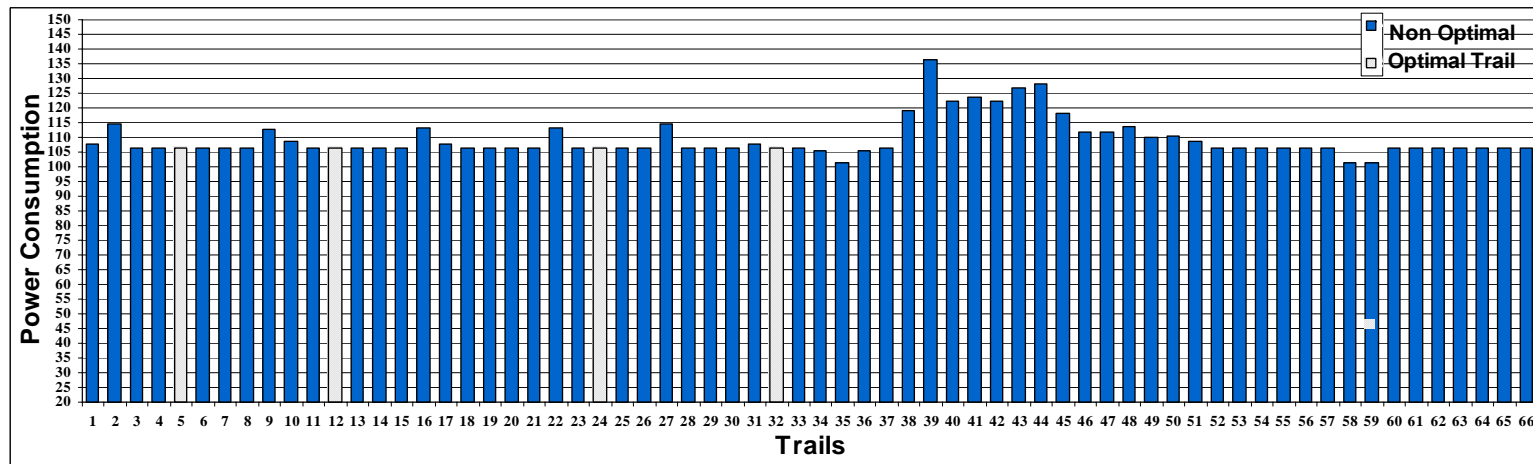
(a)



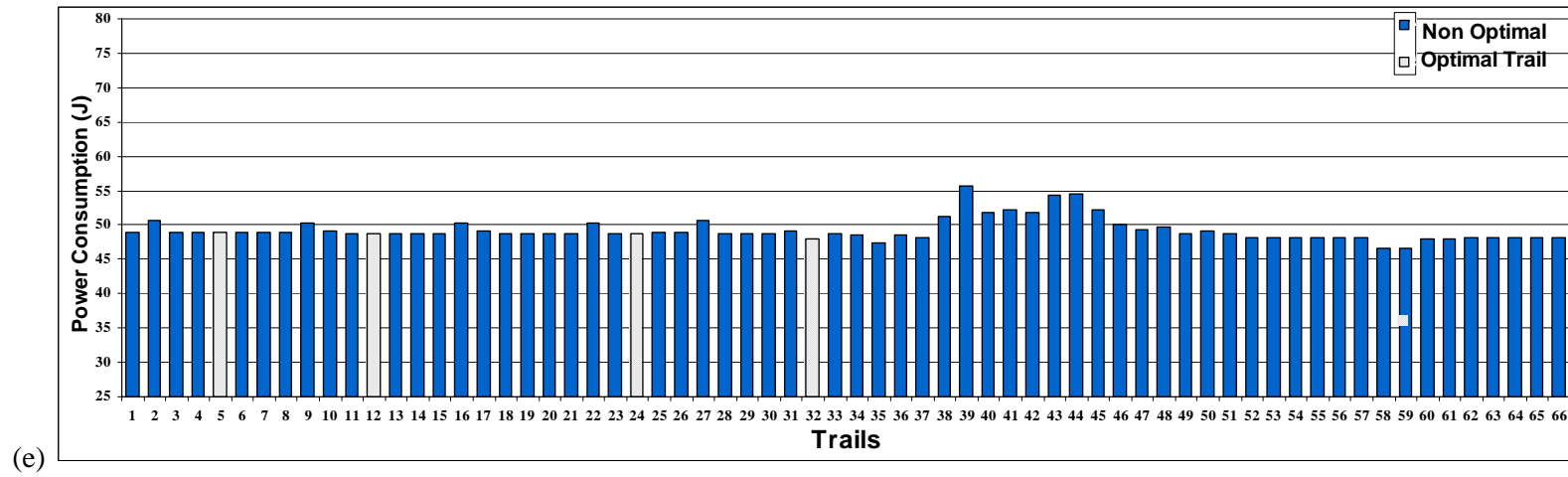
(b)



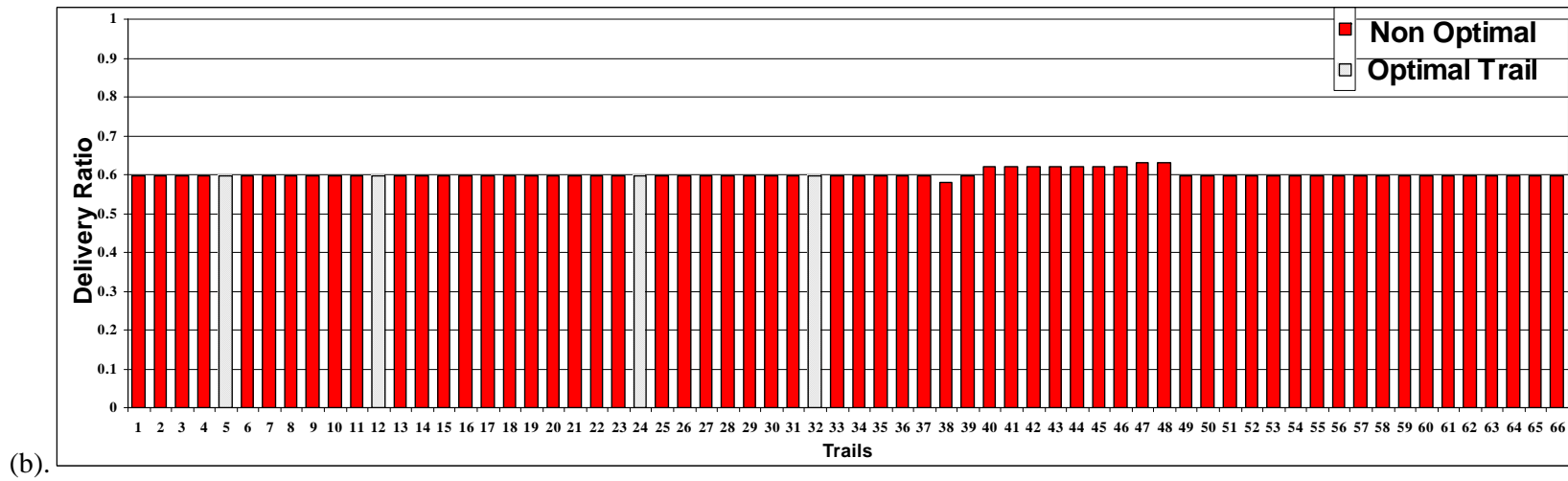
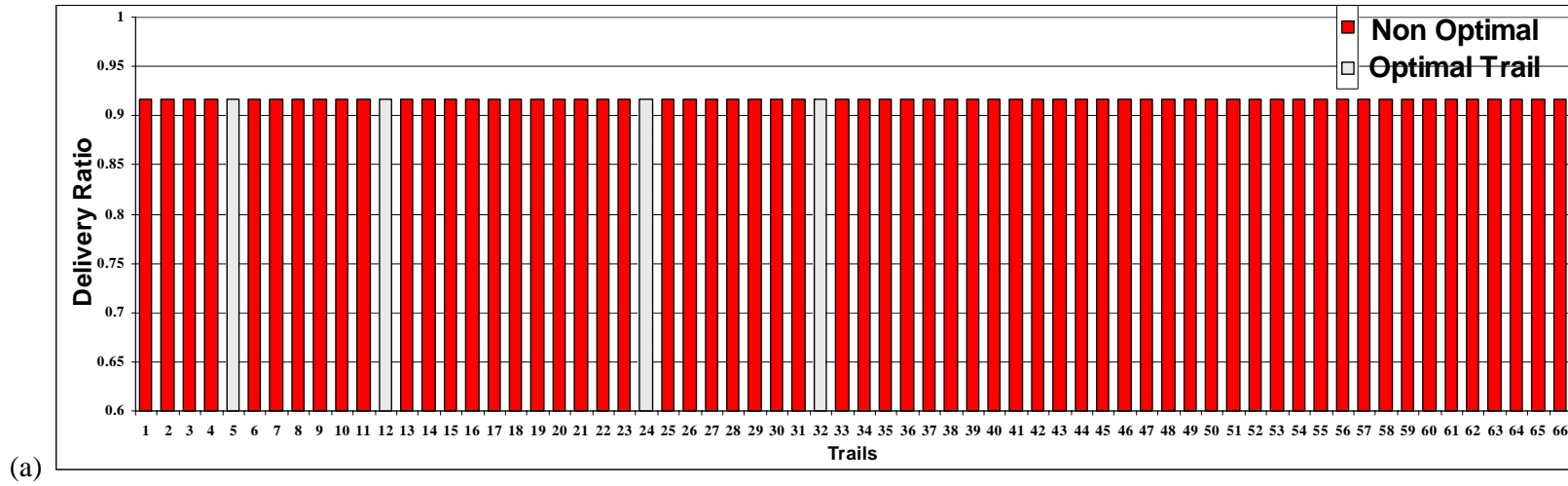
(c)



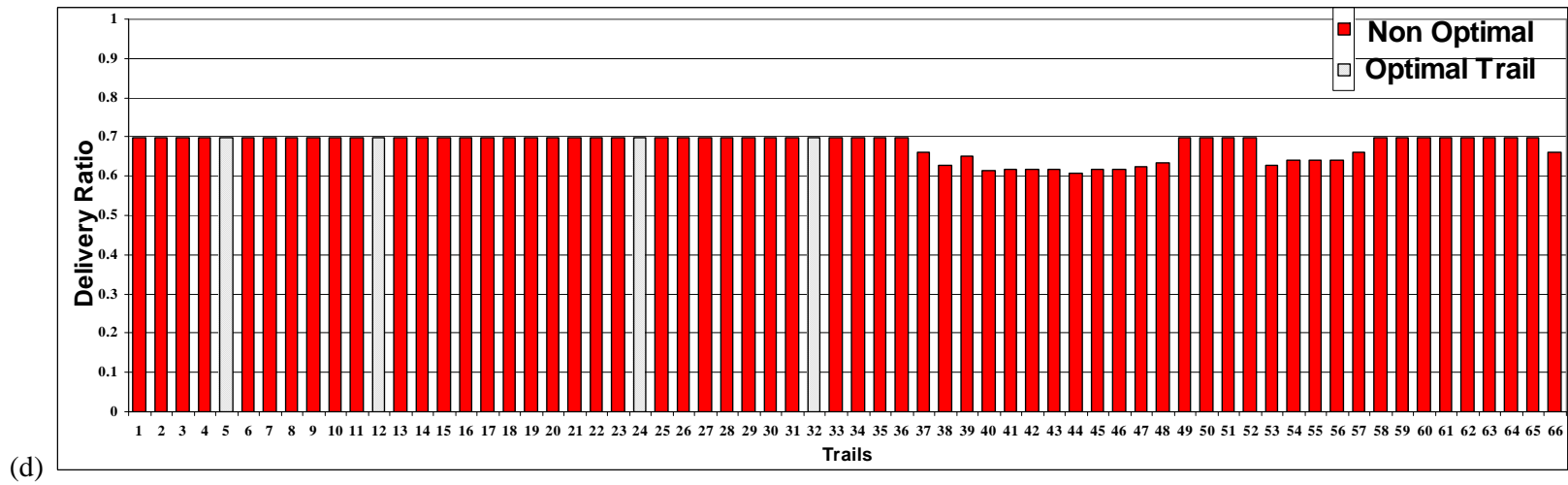
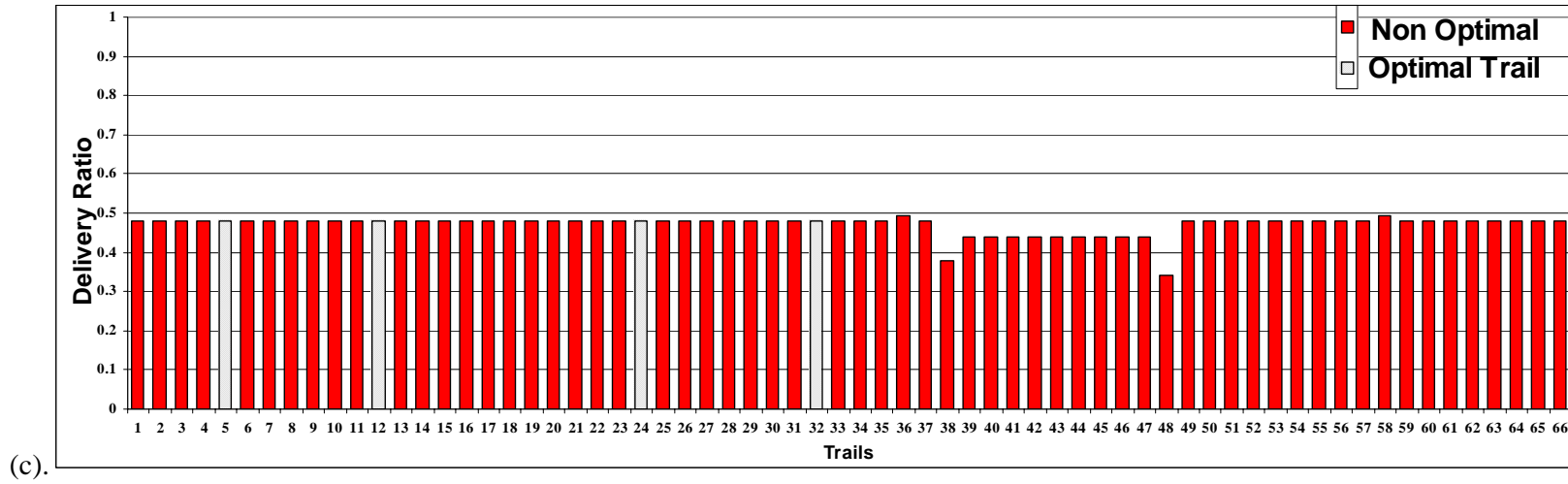
(d)

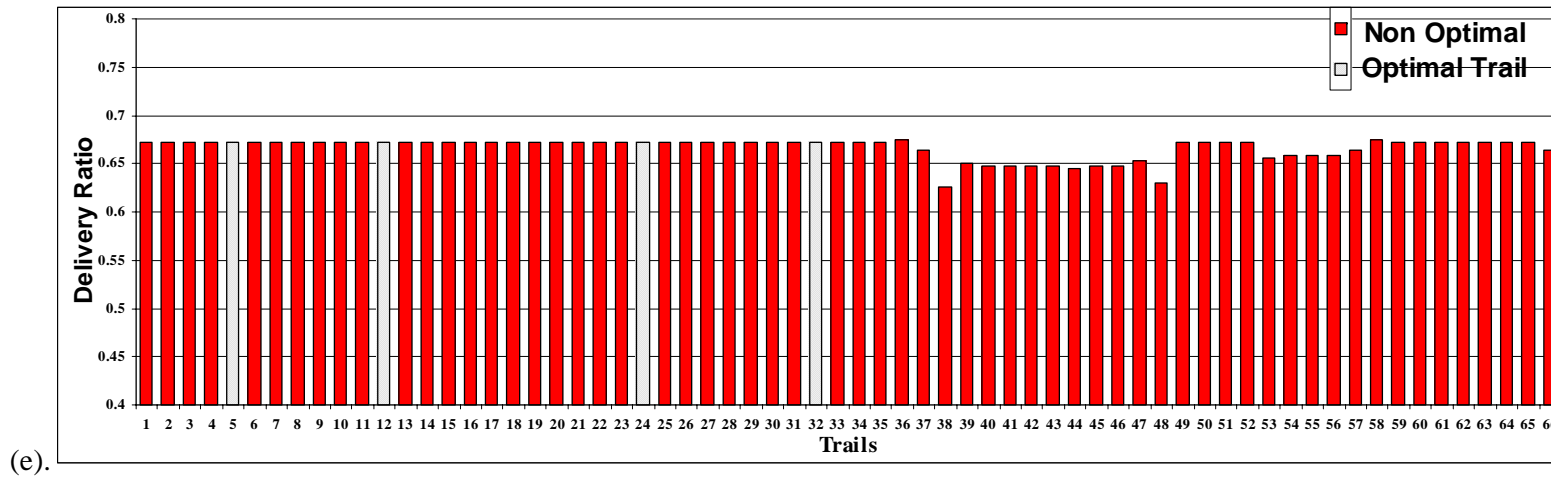


**Figure C.4** Power Consumption at 4 packet/s a) low density, b) medium density, c) high density , d) high density with several sources and e) average

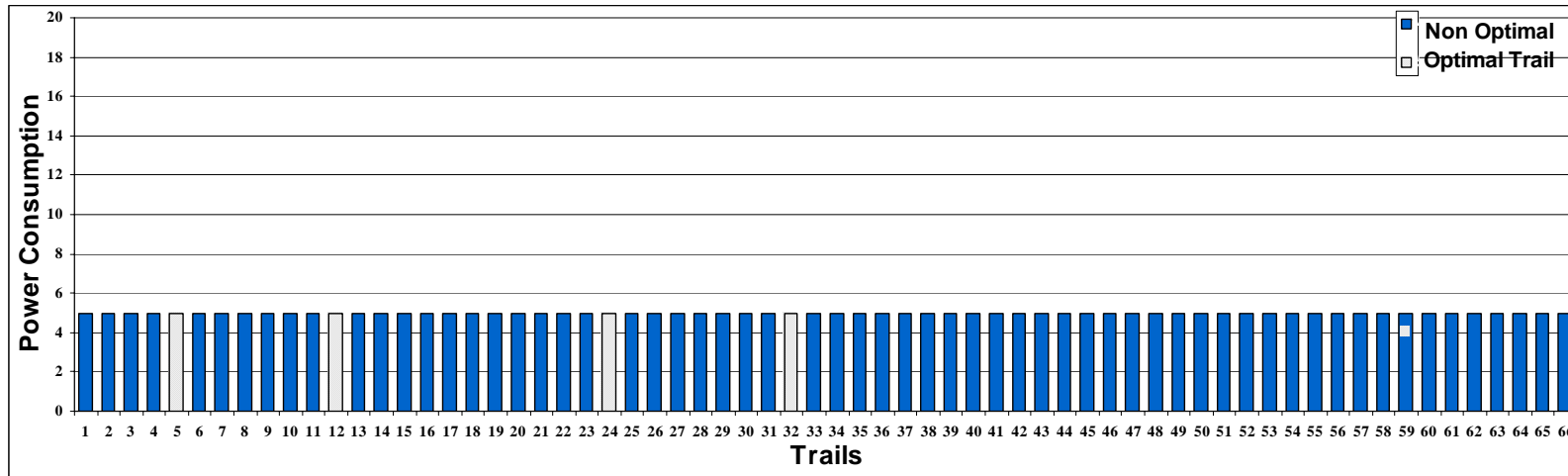




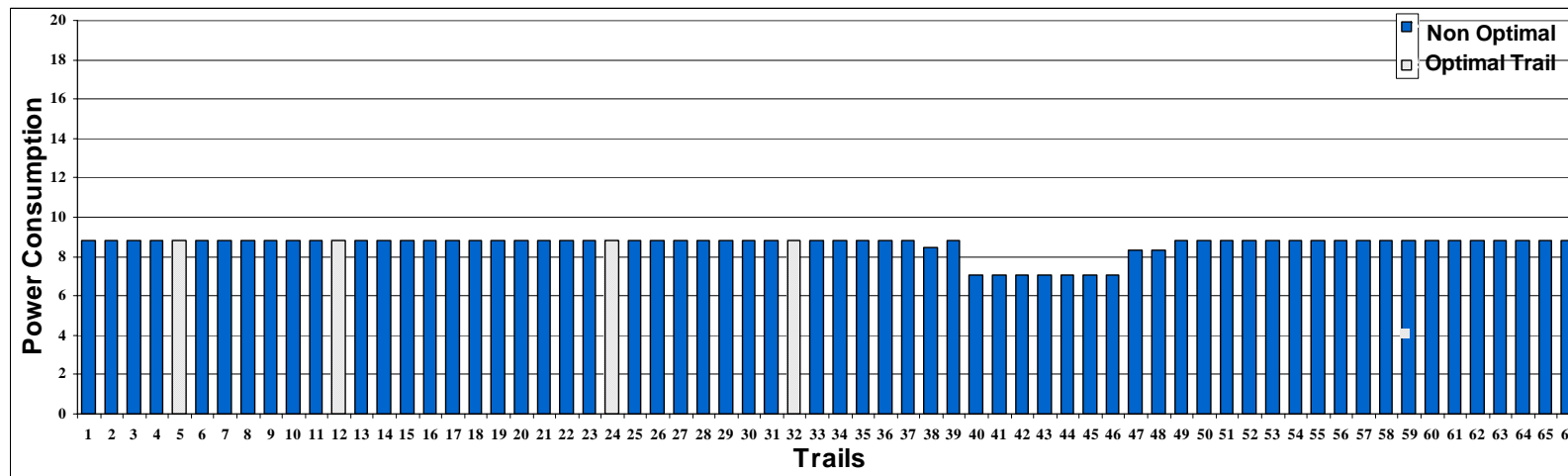




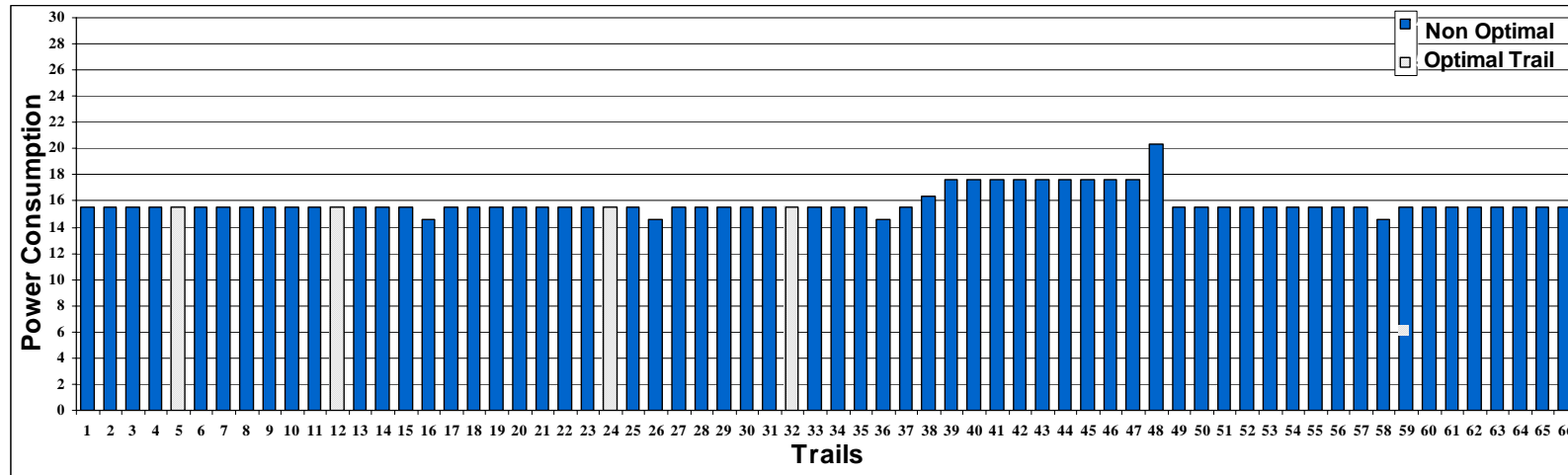
**Figure C.5** Delivery Ratio at 1 packet/s a) low density, b) medium density, c) high density, d) high density with several sources and e) average



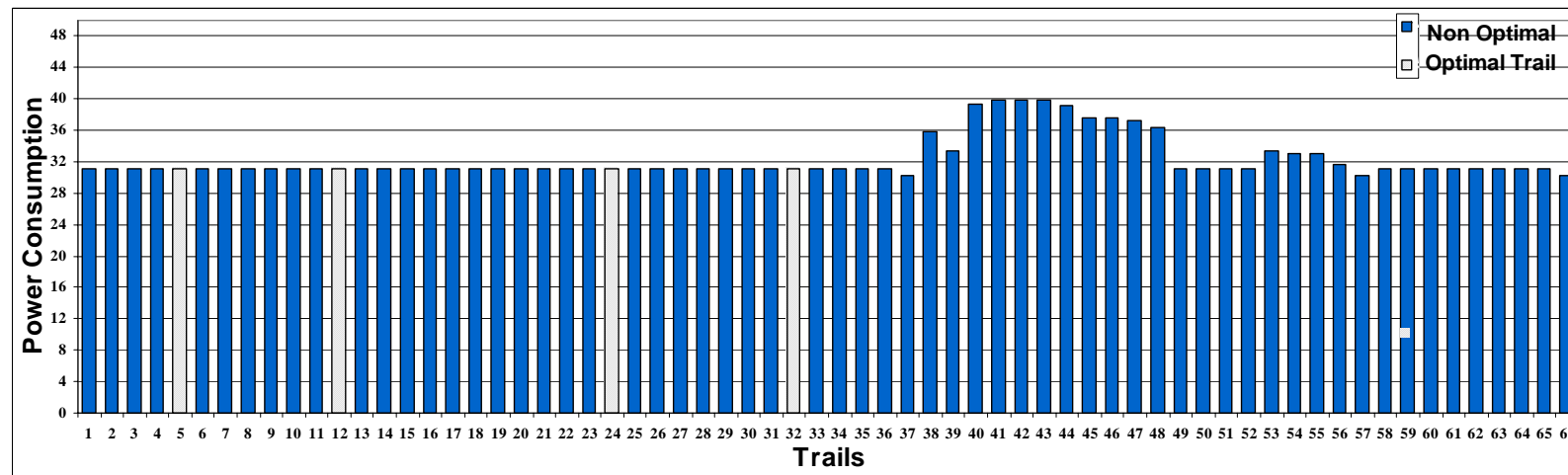
(a).



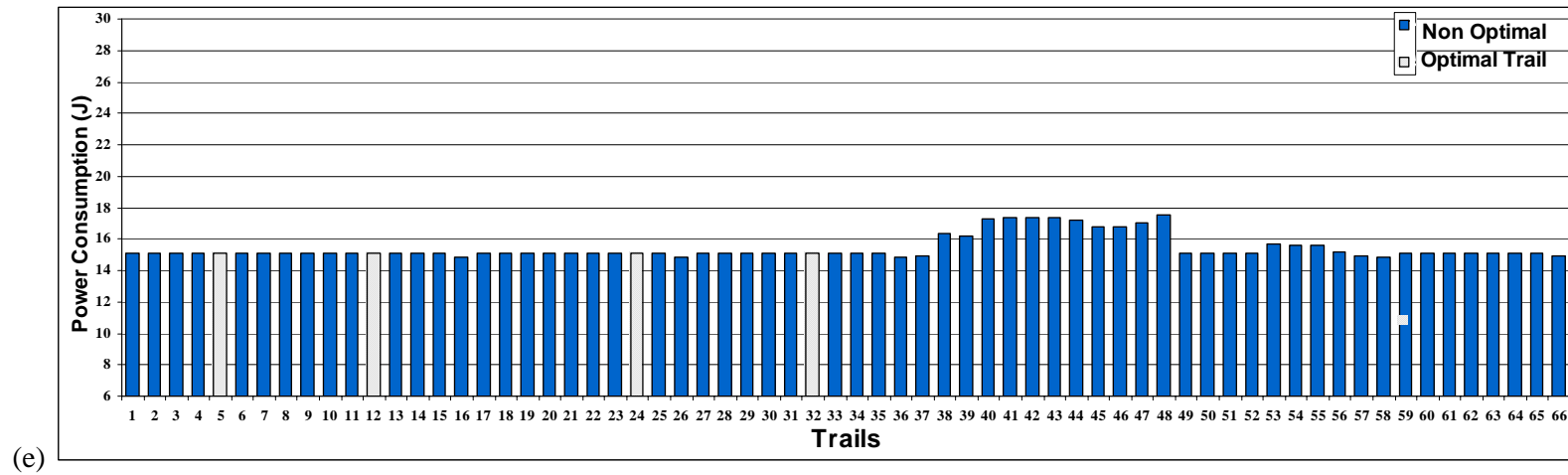
(b)



(c)

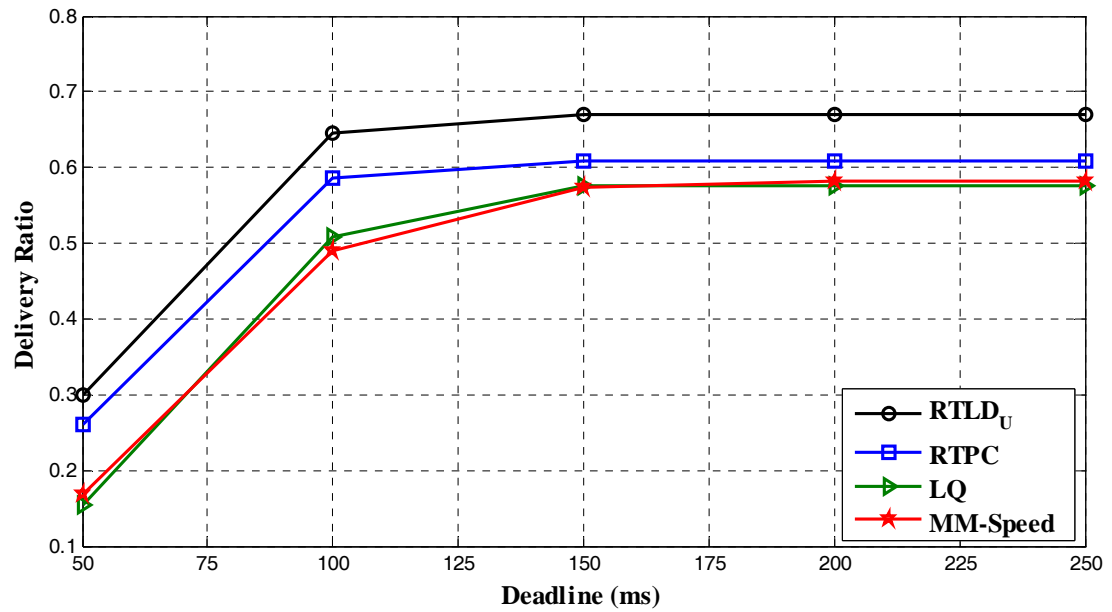


(d)

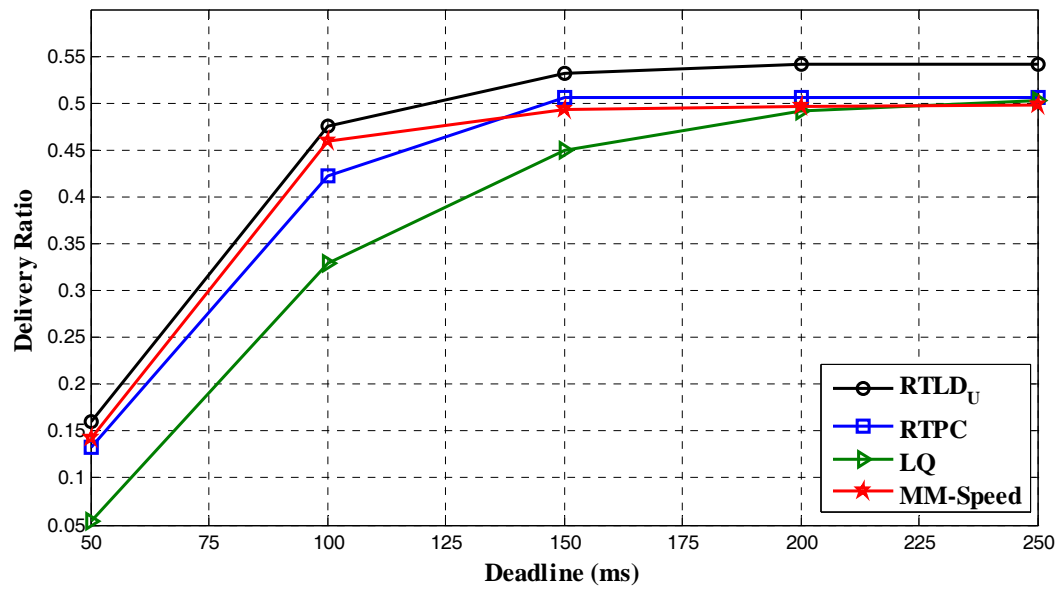


**Figure C.6** Power Consumption at 4 packet/s a) low density, b) medium density, c) high density , d) high density with several sources and e) average

### C.3 Packet Deadline Determination



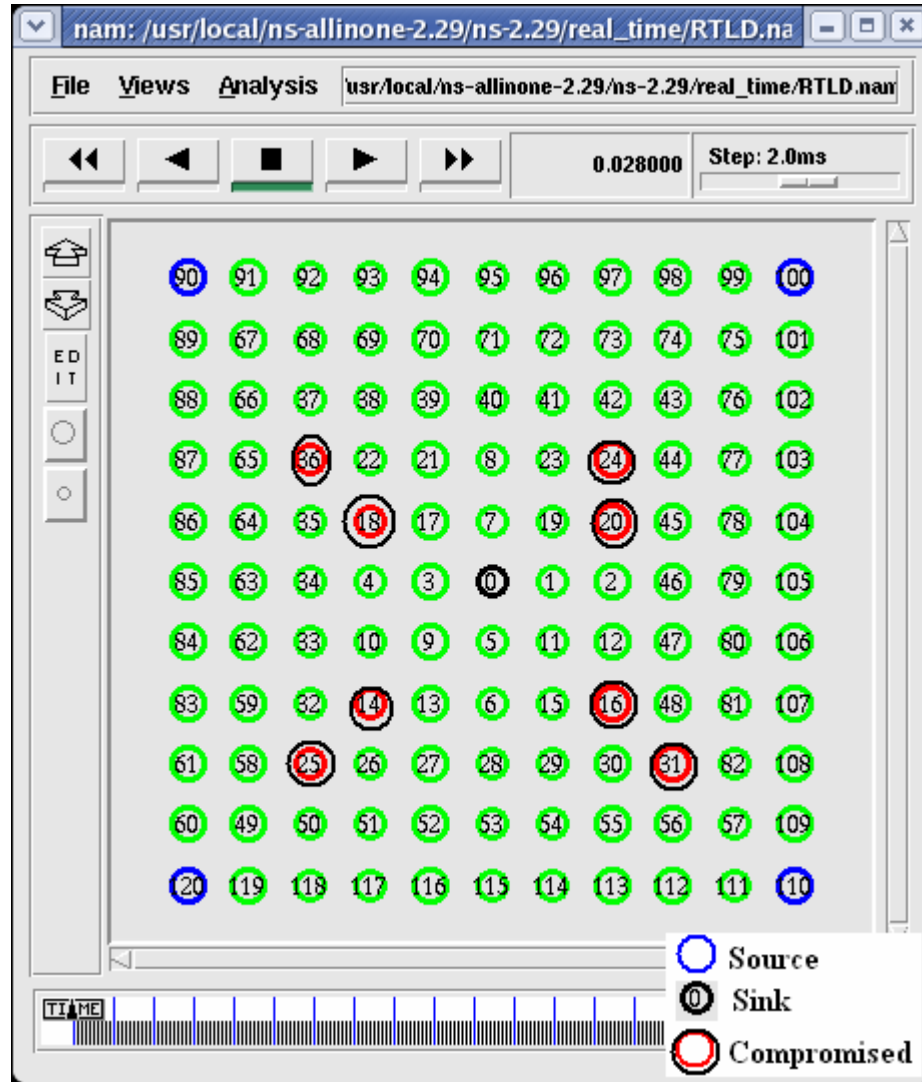
(a)



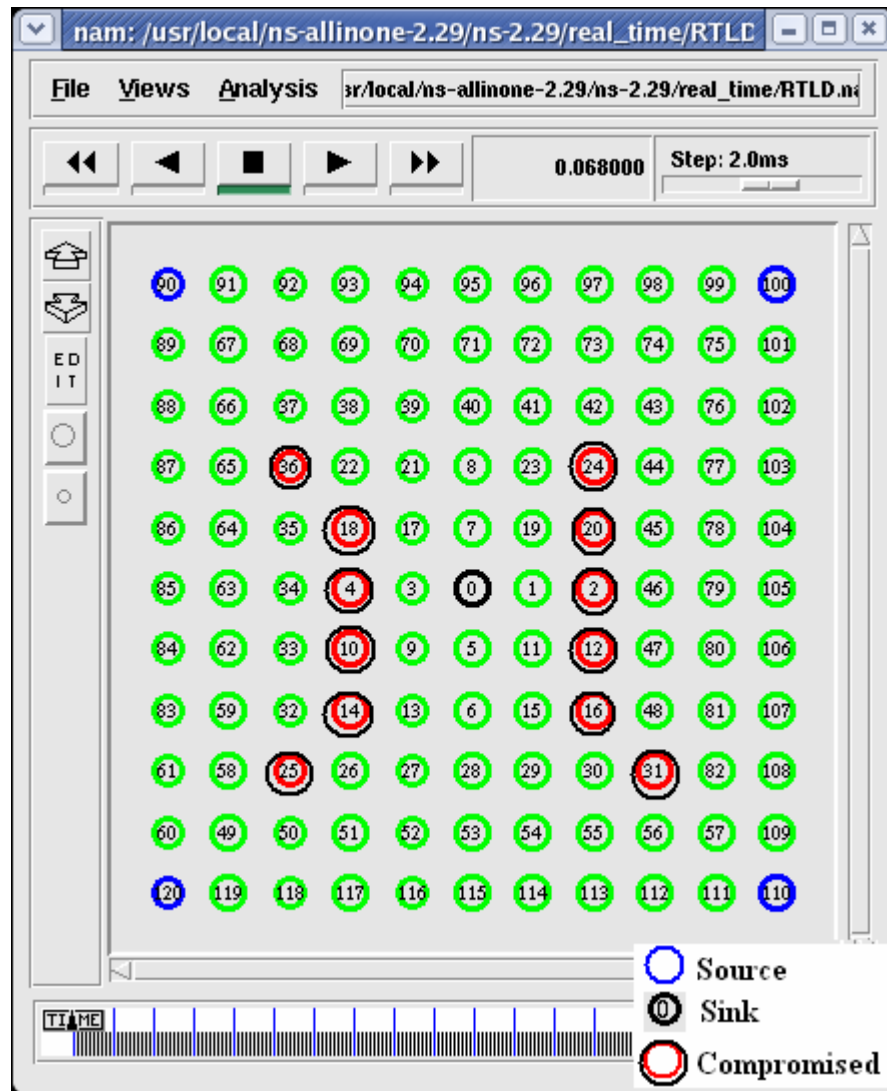
(b)

**Figure C.7** Comparison packet deadline between SRTLD and base line routing: a) 4 packet/s and b) 10 packets/s

### C.3 Influence compromised node in Enhanced Security of SRTL D

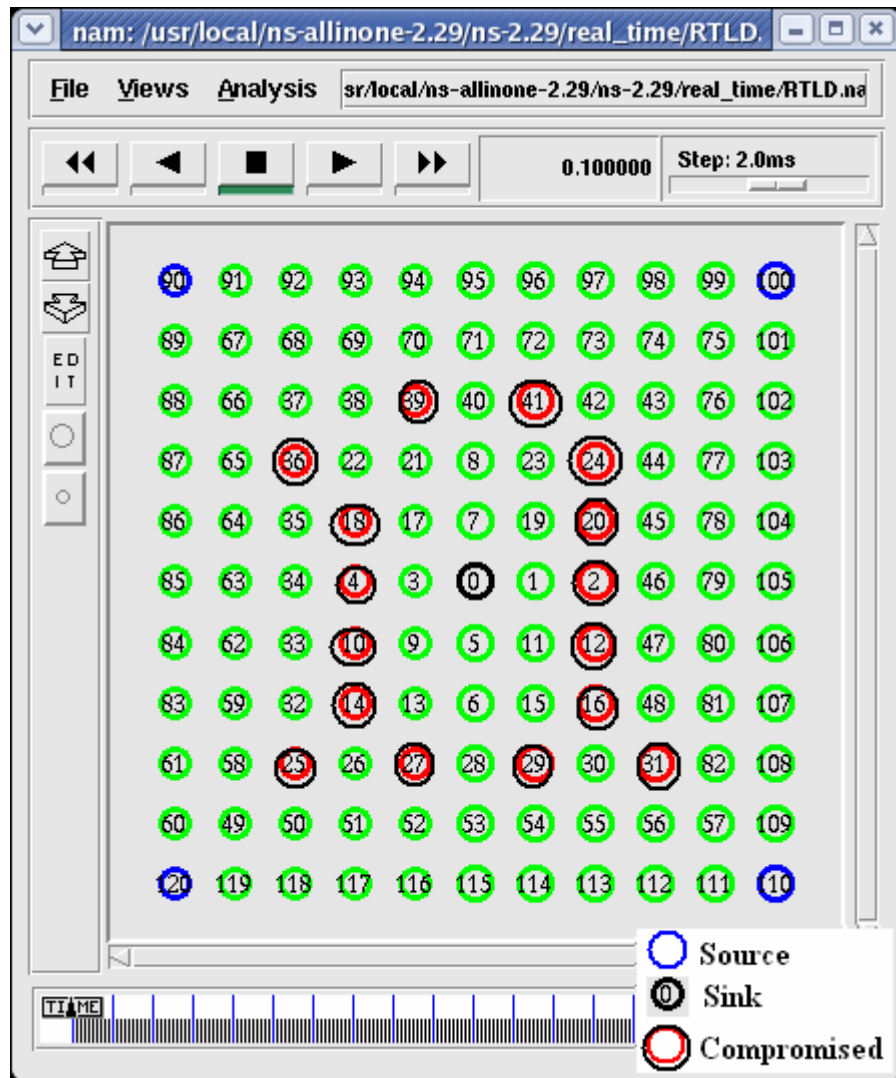


(a)



(b)





(c)



(d)

**Figure C.8** Increasing compromised nodes a) 8 nodes, b) 12 nodes, c) 16 nodes and d) 20 nodes

```

/usr/local/ns-allinone-2.29/ns-2.29(../ime_pad_security)/real_time/srtld.cc - ged
File Edit View Search Tools Documents Help
srtld.cc
//----- Encryption -----
u_int32_t srtld::hash(nsaddr_t x,u_int16_t seq) {
u_int32_t ha,z,Sprev_key;Sprev_key = 16807;
for(int i=1;i<=seq;i++)
{
z=16807*i; Sprev_key=(z*Sprev_key) % 2147483647;
}
ha=int(pow((x + Sprev_key),3)+121);
return ha;
}
//----- Decryption-----
u_int32_t srtld::revhash(nsaddr_t ha,u_int16_t seq) {
u_int32_t x,z,Dprev_key;
double integ_part;
Dprev_key = 16807;
for(int i=1;i<=seq;i++)
{
z=16807*i; Dprev_key=(z*Dprev_key) % 2147483647;
}
ha=pow(ha-121,0.333333333333333)-Dprev_key;
return x;
}

```

Figure C.9 Code of header encryption and decryption.

## C.4 TOSSIM Results

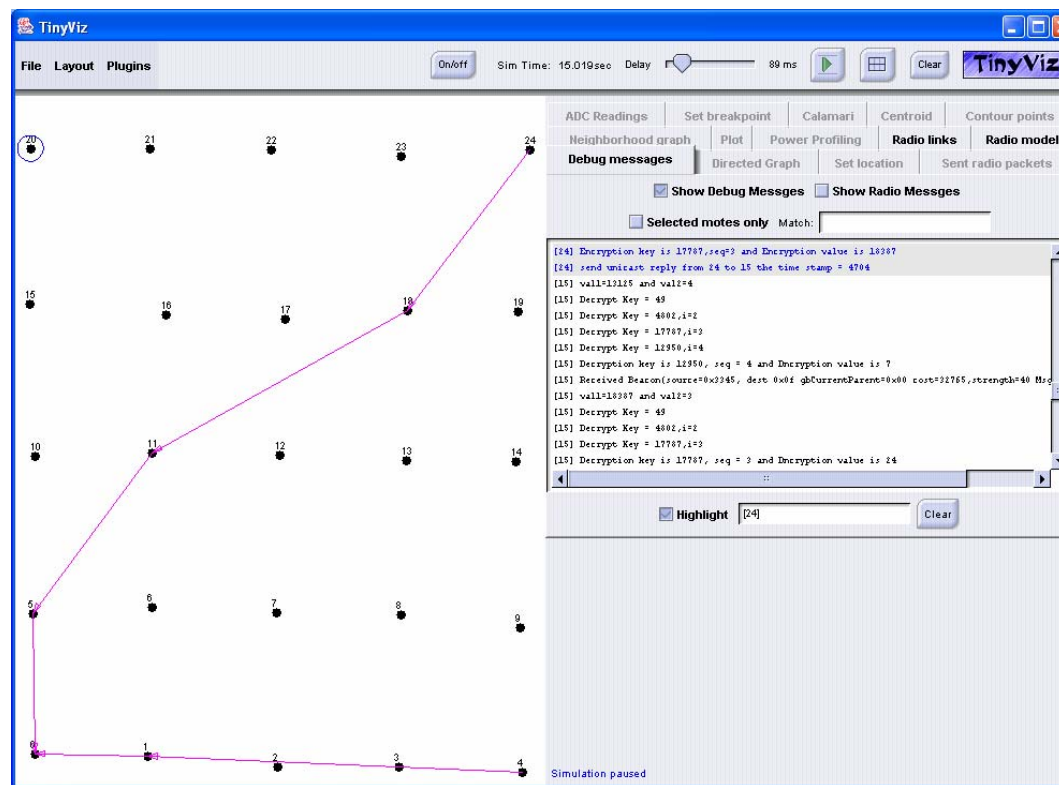


Figure C.10 Secure packet trip from node 24 to the sink

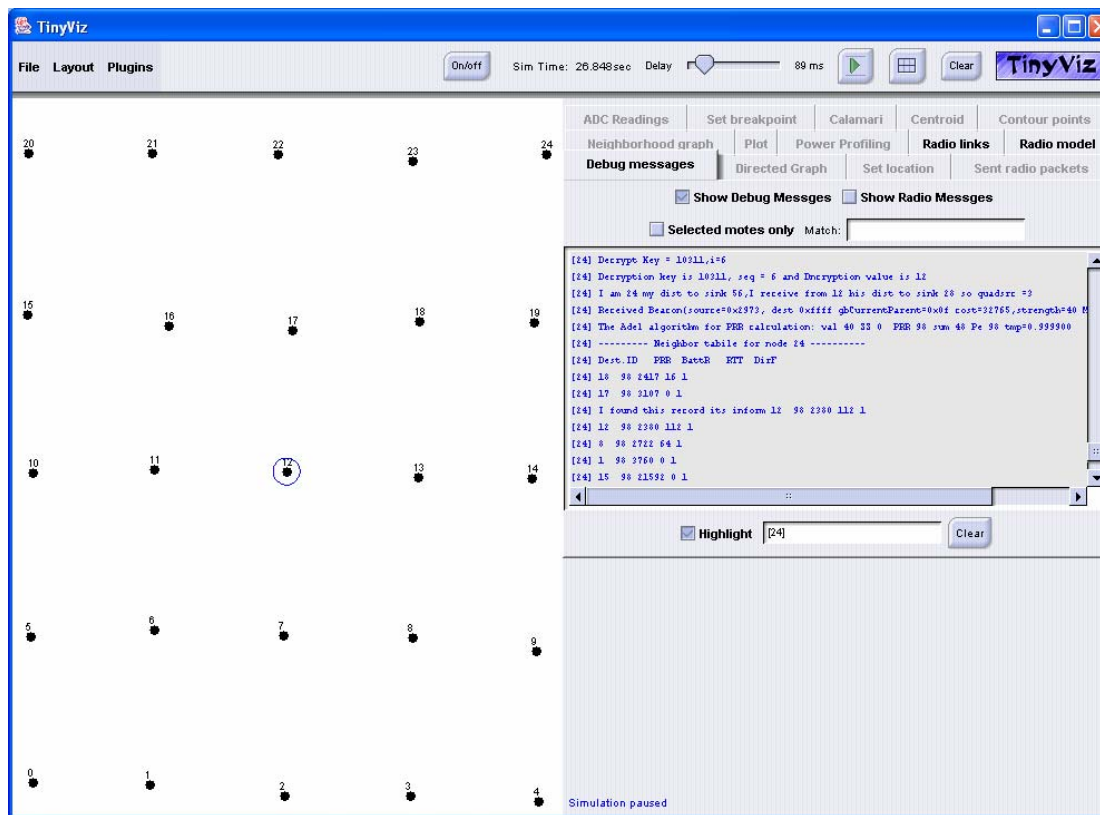


Figure C.11 RTR broadcasting at node 12

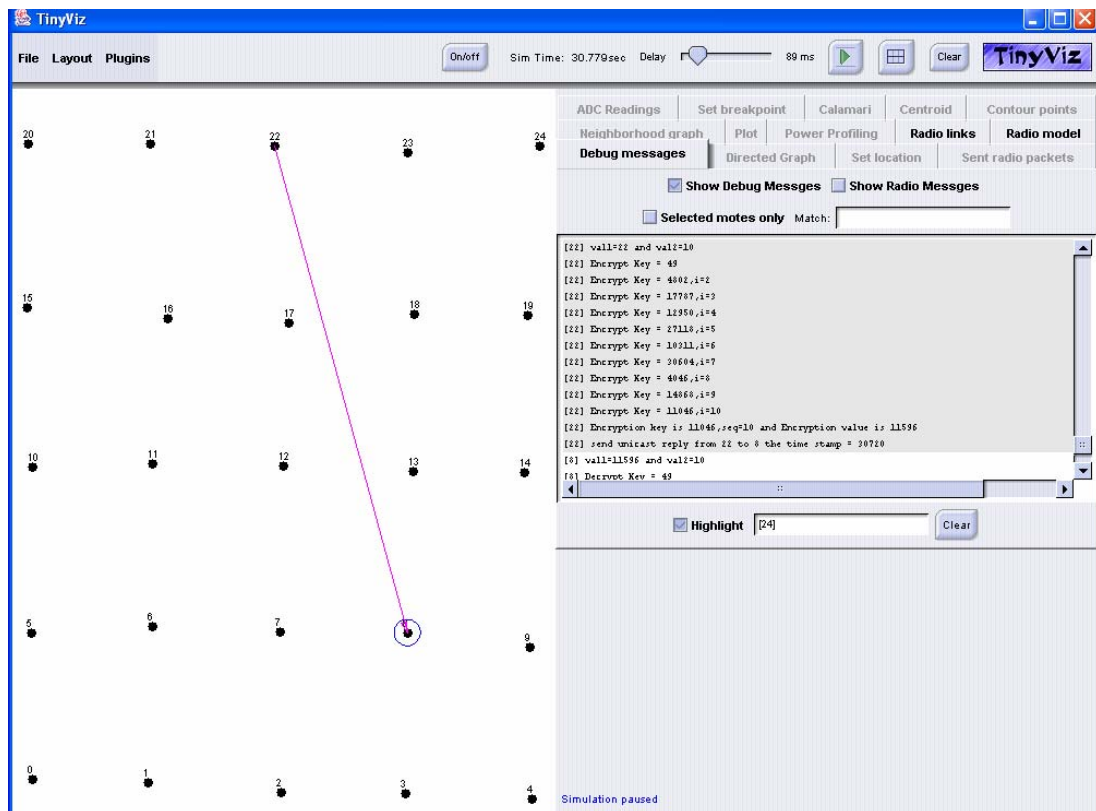


Figure C.12 RTR encryption at node 12

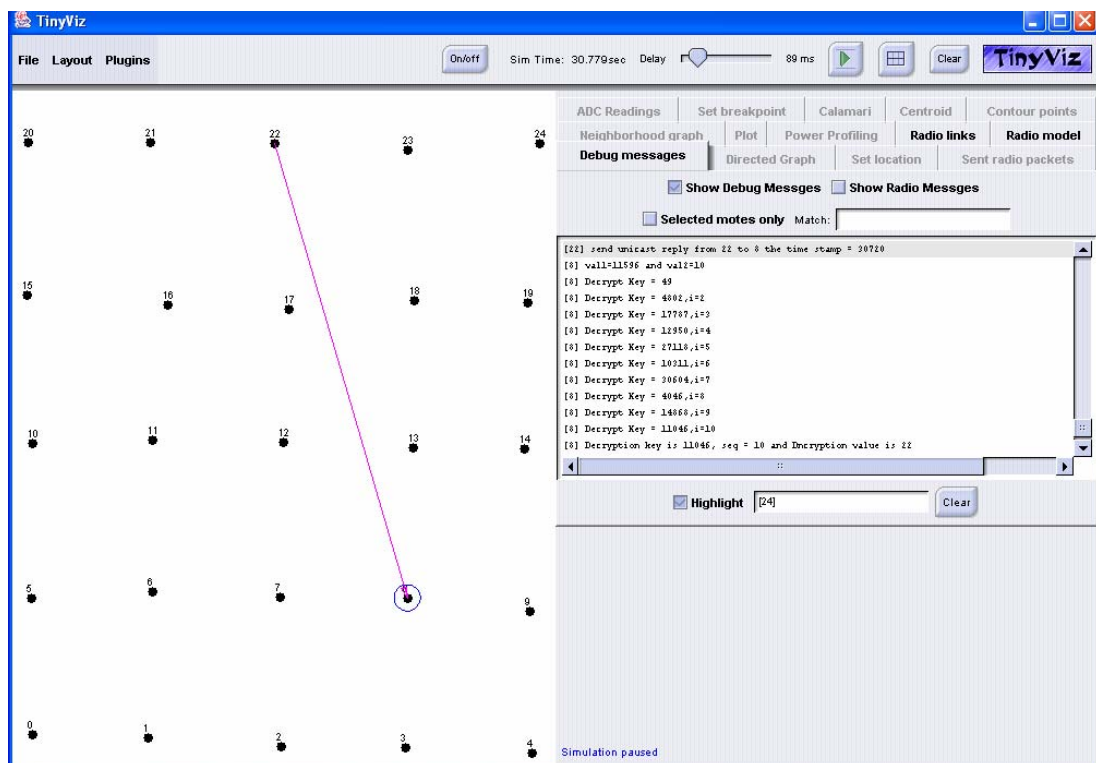


Figure C.13 Decryption RTR reply from node 22

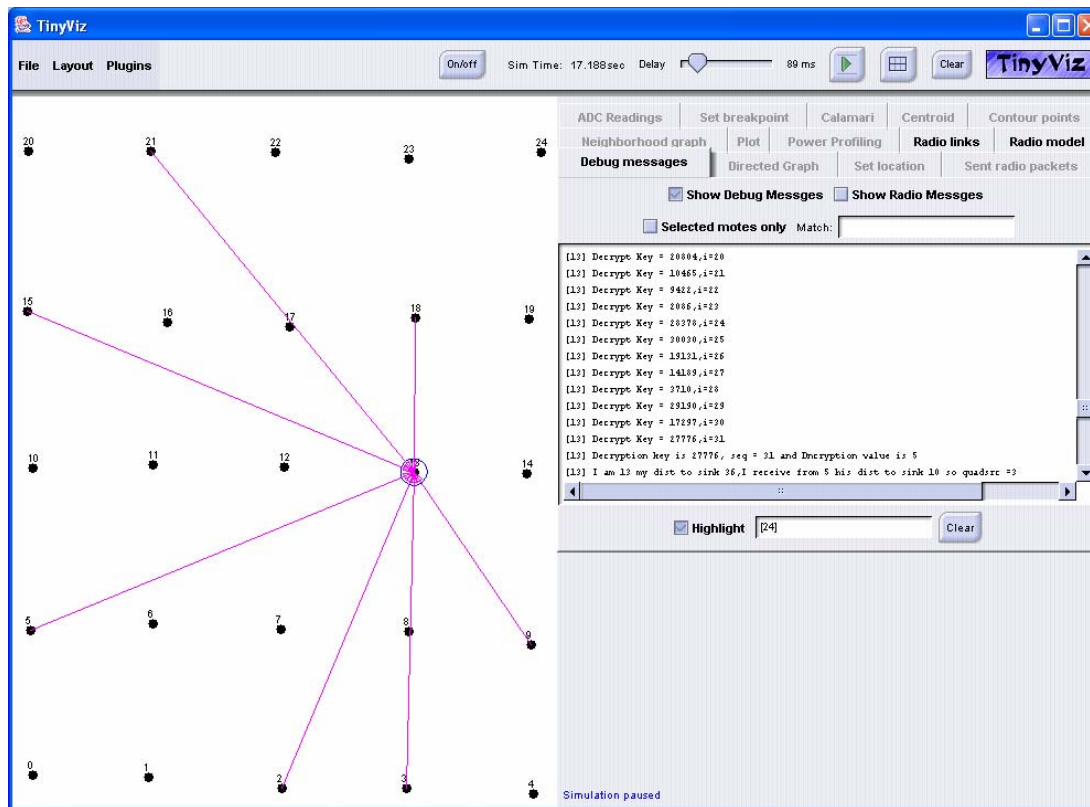


Figure C.14 Receiving RTR reply at node 12

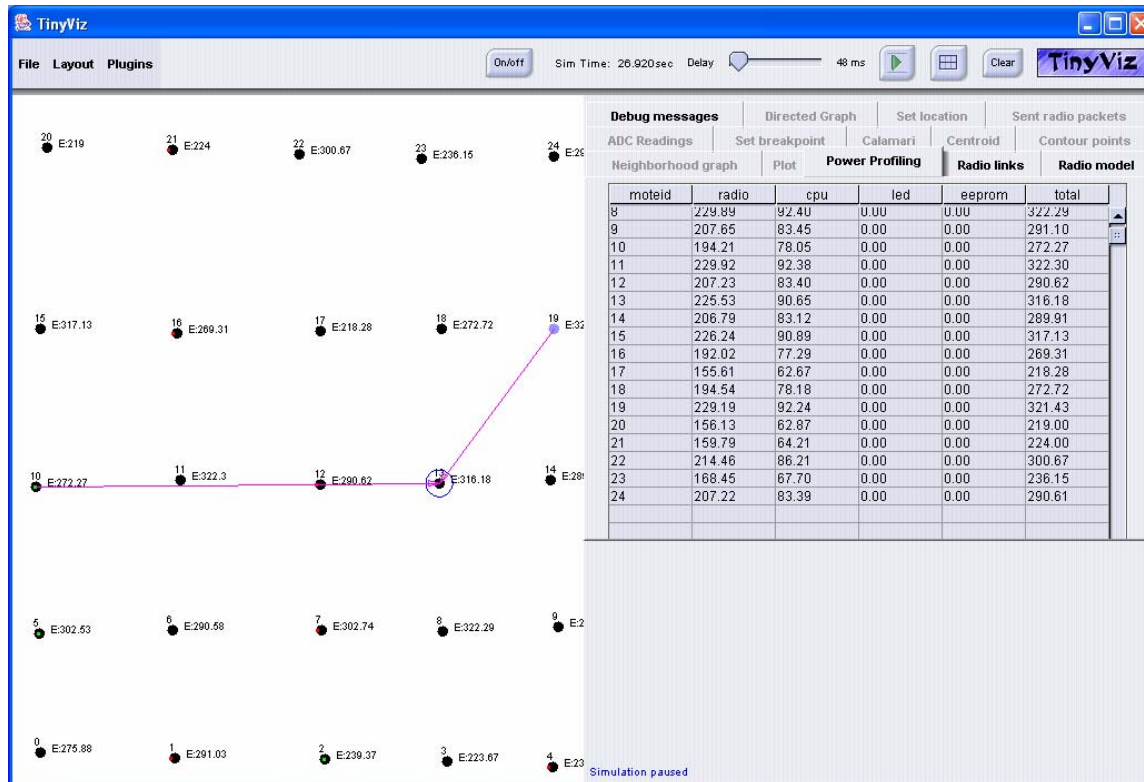
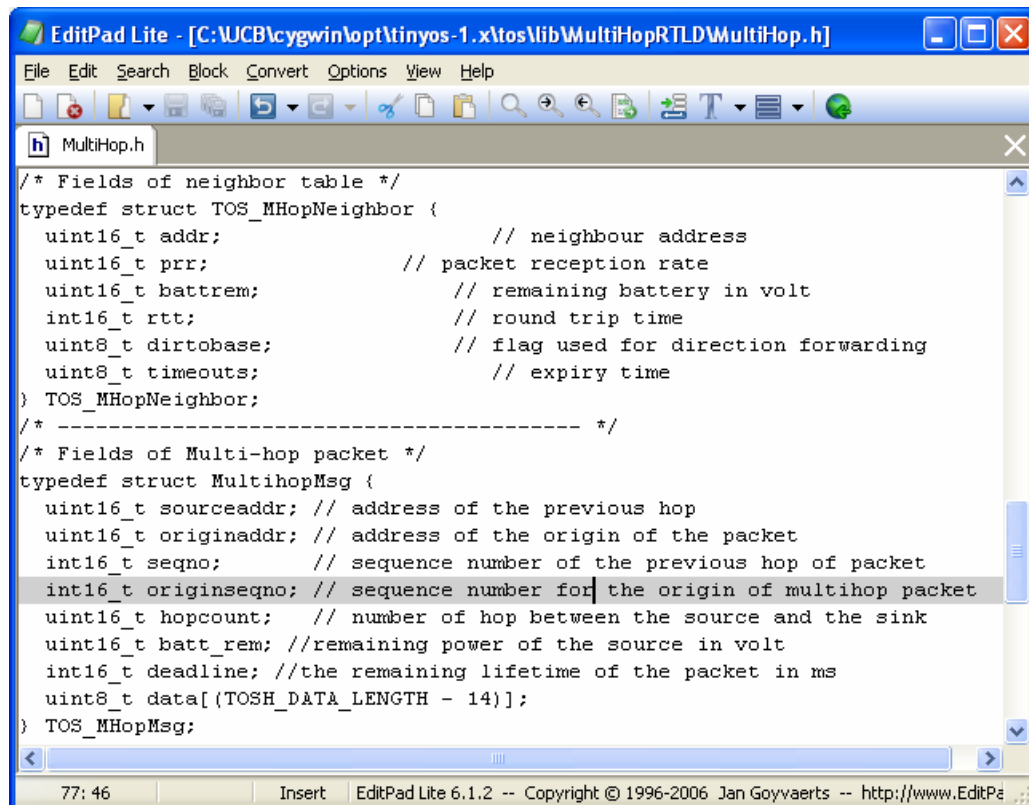


Figure C.15 Power monitoring in TOSSIM

## APPENDIX D

### Source Code of SRTLD in Real Test Bed

#### D.1 Source Code of Neighbour Table



```

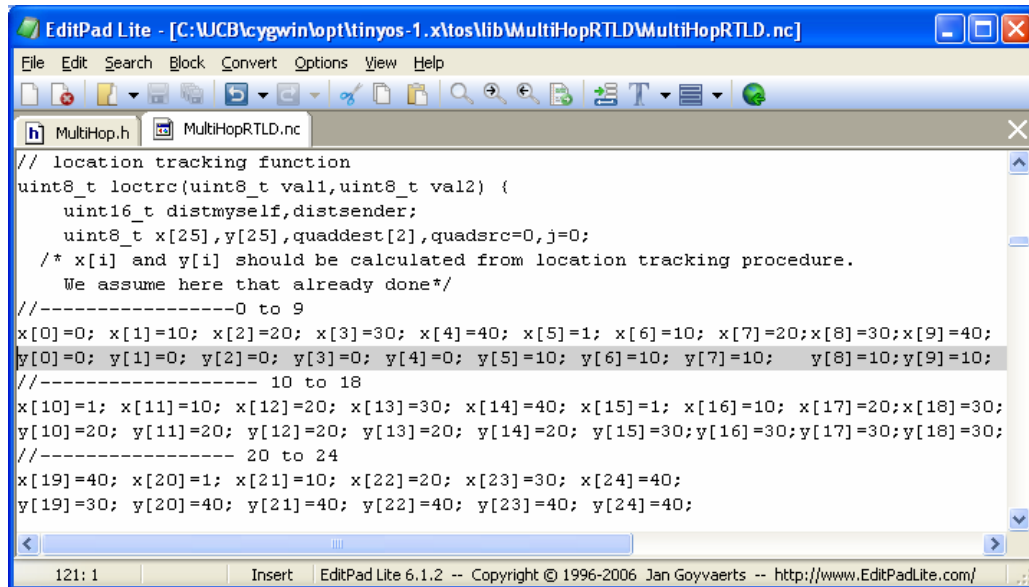
EditPad Lite - [C:\WCBA\cygwin\opt\Tinyos-1.x\lib\MultiHopRTLD\MultiHop.h]
File Edit Search Block Convert Options View Help
MultiHop.h
/* Fields of neighbor table */
typedef struct TOS_MHopNeighbor {
    uint16_t addr;                // neighbour address
    uint16_t prr;                 // packet reception rate
    uint16_t battrem;             // remaining battery in volt
    int16_t rtt;                  // round trip time
    uint8_t dirtobase;            // flag used for direction forwarding
    uint8_t timeouts;             // expiry time
} TOS_MHopNeighbor;
/* ----- */
/* Fields of Multi-hop packet */
typedef struct MultihopMsg {
    uint16_t sourceaddr; // address of the previous hop
    uint16_t originaddr; // address of the origin of the packet
    int16_t seqno;        // sequence number of the previous hop of packet
    int16_t originseqno; // sequence number for the origin of multihop packet
    uint16_t hopcount;    // number of hop between the source and the sink
    uint16_t batt_rem;    // remaining power of the source in volt
    int16_t deadline;     // the remaining lifetime of the packet in ms
    uint8_t data[(TOSH_DATA_LENGTH - 14)];
} TOS_MHopMsg;
77: 46      Insert  EditPad Lite 6.1.2 -- Copyright © 1996-2006 Jan Goyvaerts -- http://www.EditPa

```

**Figure D.1** SRTLD Multi-hop packet and neighbour table



## D.2 Positioning of Sensor Nodes



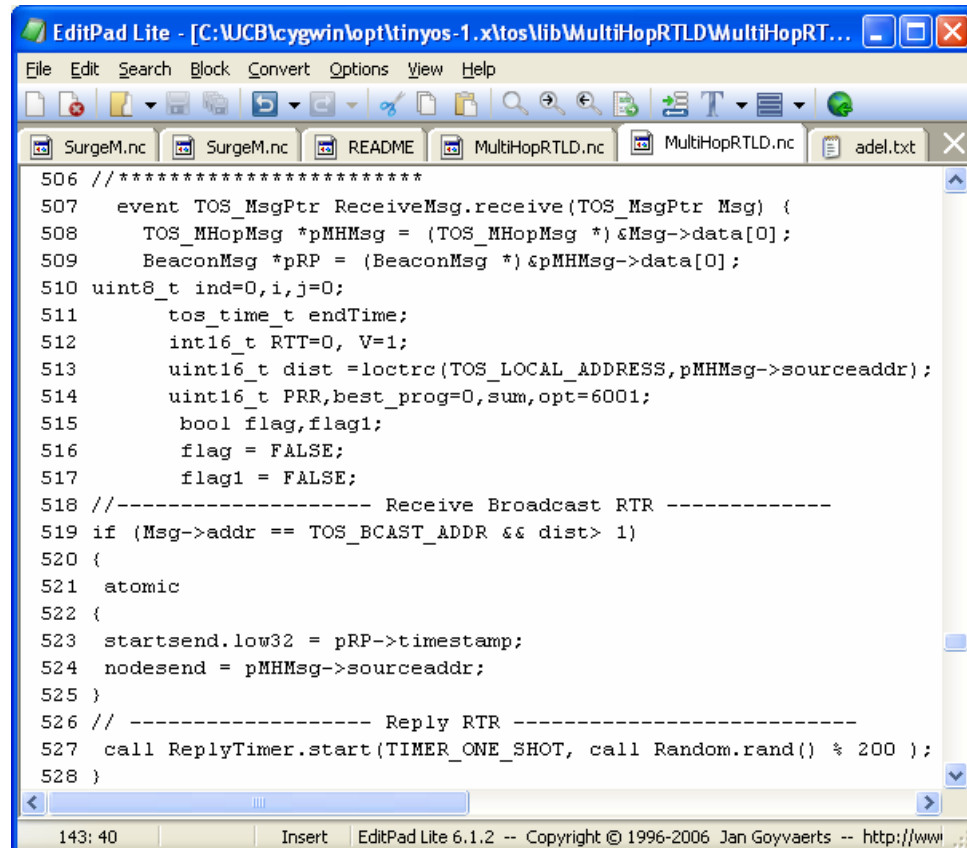
```

EditPad Lite - [C:\UCB\cygwin\opt\tinyos-1.x\Atos\lib\MultiHopRTLD\MultiHopRTLD.nc]
File Edit Search Block Convert Options View Help
MultiHop.h MultiHopRTLD.nc
// location tracking function
uint8_t loctrc(uint8_t val1,uint8_t val2) {
    uint16_t distmyself,distsender;
    uint8_t x[25],y[25],quaddest[2],quadsrc=0,j=0;
    /* x[i] and y[i] should be calculated from location tracking procedure.
       We assume here that already done*/
    //-----0 to 9
x[0]=0; x[1]=10; x[2]=20; x[3]=30; x[4]=40; x[5]=1; x[6]=10; x[7]=20;x[8]=30;x[9]=40;
y[0]=0; y[1]=0; y[2]=0; y[3]=0; y[4]=0; y[5]=10; y[6]=10; y[7]=10; y[8]=10;y[9]=10;
    //----- 10 to 18
x[10]=1; x[11]=10; x[12]=20; x[13]=30; x[14]=40; x[15]=1; x[16]=10; x[17]=20;x[18]=30;
y[10]=20; y[11]=20; y[12]=20; y[13]=20; y[14]=20; y[15]=30;y[16]=30;y[17]=30;y[18]=30;
    //----- 20 to 24
x[19]=40; x[20]=1; x[21]=10; x[22]=20; x[23]=30; x[24]=40;
y[19]=30; y[20]=40; y[21]=40; y[22]=40; y[23]=40; y[24]=40;
121: 1 Insert EditPad Lite 6.1.2 -- Copyright © 1996-2006 Jan Goyvaerts -- http://www.EditPadLite.com/

```

Figure D.2 Positioning of sensor nodes

## D.3 Source Code of Routing Management



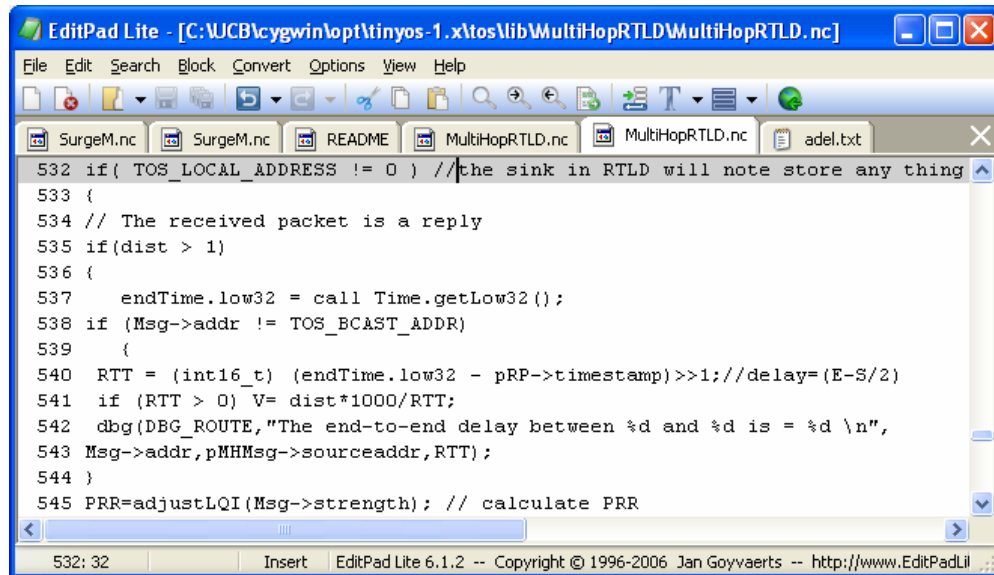
```

EditPad Lite - [C:\UCB\cygwin\opt\tinyos-1.x\Atos\lib\MultiHopRTLD\MultiHopRTLD.nc]
File Edit Search Block Convert Options View Help
SurgeM.nc SurgeM.nc README MultiHopRTLD.nc MultiHopRTLD.nc adel.txt
506 //*****
507 event TOS_MsgPtr ReceiveMsg.receive(TOS_MsgPtr Msg) {
508     TOS_MHopMsg *pMHMsg = (TOS_MHopMsg *) &Msg->data[0];
509     BeaconMsg *pRP = (BeaconMsg *) &pMHMsg->data[0];
510     uint8_t ind=0,i,j=0;
511     tos_time_t endTime;
512     int16_t RTT=0, V=1;
513     uint16_t dist =loctrc(TOS_LOCAL_ADDRESS,pMHMsg->sourceaddr);
514     uint16_t PRR,best_prog=0,sum,opt=6001;
515     bool flag,flag1;
516     flag = FALSE;
517     flag1 = FALSE;
518 //----- Receive Broadcast RTR -----
519 if (Msg->addr == TOS_BCAST_ADDR && dist> 1)
520 {
521     atomic
522     {
523         startsend.low32 = pRP->timestamp;
524         nodesend = pMHMsg->sourceaddr;
525     }
526 // ----- Reply RTR -----
527 call ReplyTimer.start(TIMER_ONE_SHOT, call Random.rand() % 200 );
528 }
143: 40 Insert EditPad Lite 6.1.2 -- Copyright © 1996-2006 Jan Goyvaerts -- http://www

```

Figure D.3 Receiving RTR packet



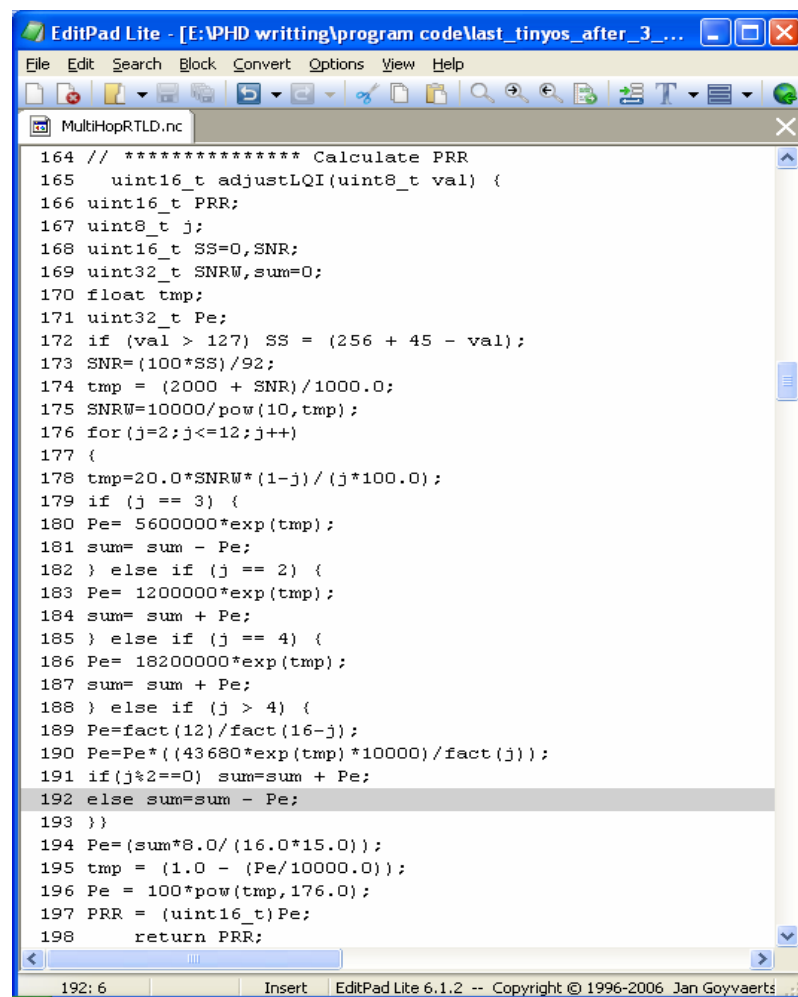


```

EditPad Lite - [C:\WCBC\cygwin\opt\tinyos-1.x\tos\lib\MultiHopRTLD\MultiHopRTLD.nc]
File Edit Search Block Convert Options View Help
SurgeM.nc SurgeM.nc README MultiHopRTLD.nc MultiHopRTLD.nc adel.txt
532 if( TOS_LOCAL_ADDRESS != 0 ) //the sink in RTLD will not store anything
533 {
534 // The received packet is a reply
535 if(dist > 1)
536 {
537     endTime.low32 = call Time.getLow32();
538     if (Msg->addr != TOS_BCAST_ADDR)
539     {
540         RTT = (int16_t) (endTime.low32 - pRP->timestamp)>>1; //delay=(E-S/2)
541         if (RTT > 0) V= dist*1000/RTT;
542         dbg(DBG_ROUTE,"The end-to-end delay between %d and %d is = %d \n",
543             Msg->addr,pMMsg->sourceaddr,RTT);
544     }
545     PRR=adjustLQI(Msg->strength); // calculate PRR

```

Figure D.4 Receiving RTR reply packet

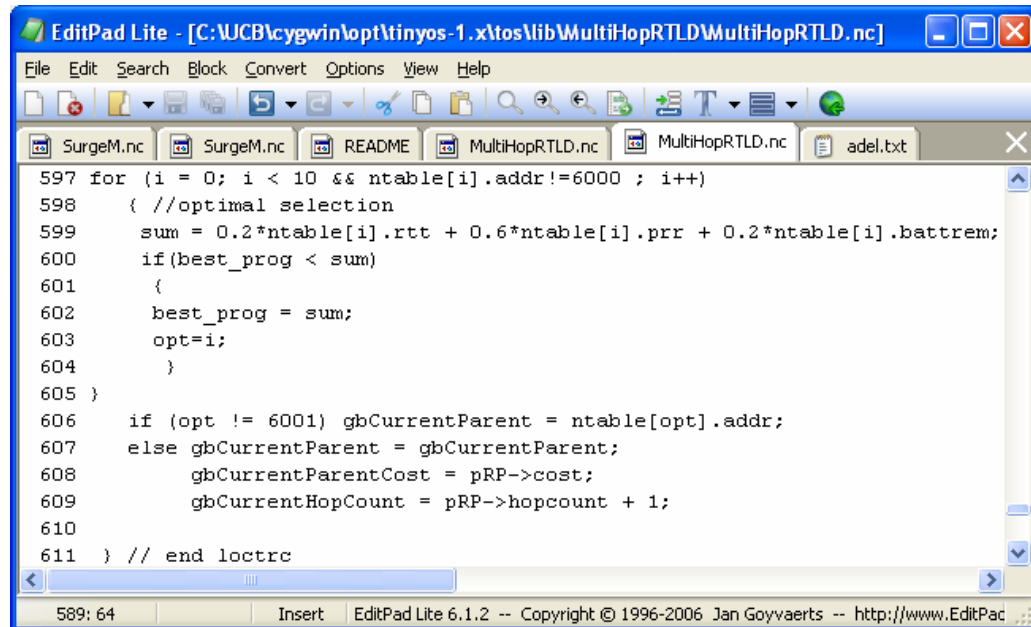


```

EditPad Lite - [E:\VPHD\writing\program code\last_tinyos_after_3...
MultiHopRTLD.nc
164 // ***** Calculate PRR
165 uint16_t adjustLQI(uint8_t val) {
166     uint16_t PRR;
167     uint8_t j;
168     uint16_t SS=0,SNR;
169     uint32_t SNRW,sum=0;
170     float tmp;
171     uint32_t Pe;
172     if (val > 127) SS = (256 + 45 - val);
173     SNR=(100*SS)/92;
174     tmp = (2000 + SNR)/1000.0;
175     SNRW=10000/pow(10,tmp);
176     for(j=2;j<=12;j++)
177     {
178         tmp=20.0*SNRW*(1-j)/(j*100.0);
179         if (j == 3) {
180             Pe= 5600000*exp(tmp);
181             sum= sum - Pe;
182         } else if (j == 2) {
183             Pe= 1200000*exp(tmp);
184             sum= sum + Pe;
185         } else if (j == 4) {
186             Pe= 18200000*exp(tmp);
187             sum= sum + Pe;
188         } else if (j > 4) {
189             Pe=fact(12)/fact(16-j);
190             Pe=Pe*((43680*exp(tmp)*10000)/fact(j));
191             if(j%2==0) sum=sum + Pe;
192         } else sum=sum - Pe;
193     }
194     Pe=(sum*8.0/(16.0*15.0));
195     tmp = (1.0 - (Pe/10000.0));
196     Pe = 100*pow(tmp,176.0);
197     PRR = (uint16_t)Pe;
198     return PRR;

```

Figure D.5 PRR calculation function



The screenshot shows the EditPad Lite application window with the title bar "[C:\UCB\cygwin\opt\tinyos-1.x\ntos\lib\MultiHopRTLD\MultiHopRTLD.nc]". The menu bar includes File, Edit, Search, Block, Convert, Options, View, and Help. The toolbar contains various icons for file operations and editing. The active tab is "MultiHopRTLD.nc". The code displayed is as follows:

```

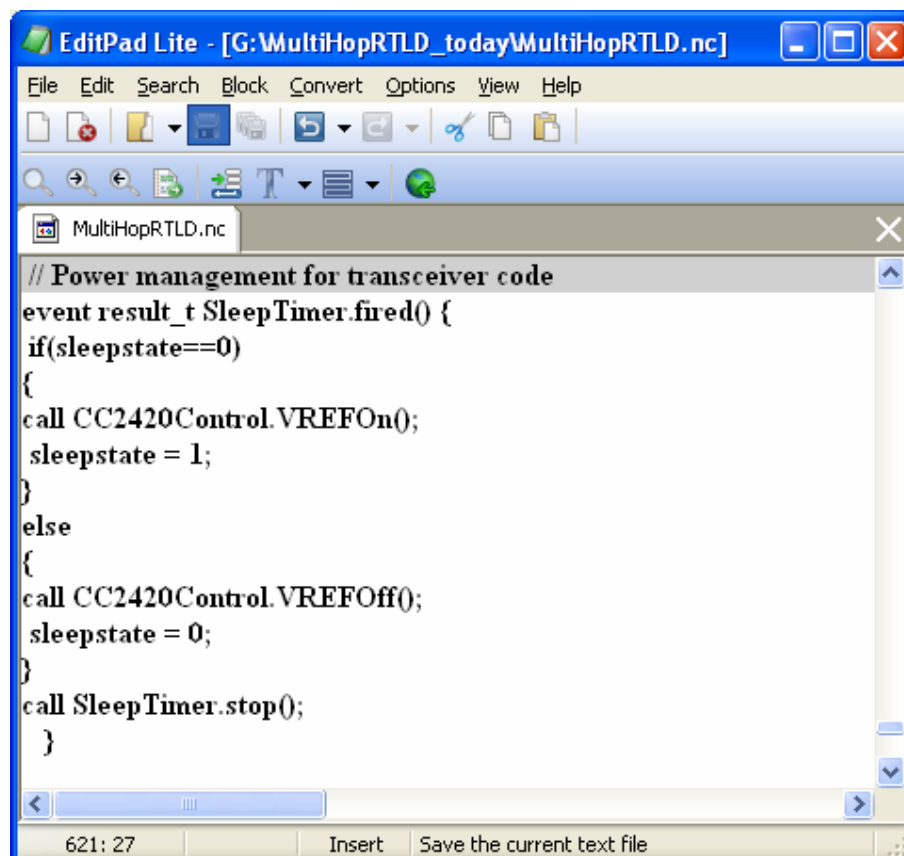
597 for (i = 0; i < 10 && ntable[i].addr!=6000 ; i++)
598 { //optimal selection
599     sum = 0.2*ntable[i].rtt + 0.6*ntable[i].prp + 0.2*ntable[i].battrem;
600     if(best_prog < sum)
601     {
602         best_prog = sum;
603         opt=i;
604     }
605 }
606 if (opt != 6001) gbCurrentParent = ntable[opt].addr;
607 else gbCurrentParent = gbCurrentParent;
608     gbCurrentParentCost = pRP->cost;
609     gbCurrentHopCount = pRP->hopcount + 1;
610
611 } // end loctrc

```

The status bar at the bottom shows "589: 64", "Insert", and "EditPad Lite 6.1.2 -- Copyright © 1996-2006 Jan Goyvaerts -- http://www.EditPac...".

**Figure D.6** Optimal forwarding candidate calculation

#### D.4 Source Code of Power Management for Transceiver



The screenshot shows the EditPad Lite application window with the title bar "[G:\MultiHopRTLD\_today\MultiHopRTLD.nc]". The menu bar includes File, Edit, Search, Block, Convert, Options, View, and Help. The toolbar contains various icons for file operations and editing. The active tab is "MultiHopRTLD.nc". The code displayed is as follows:

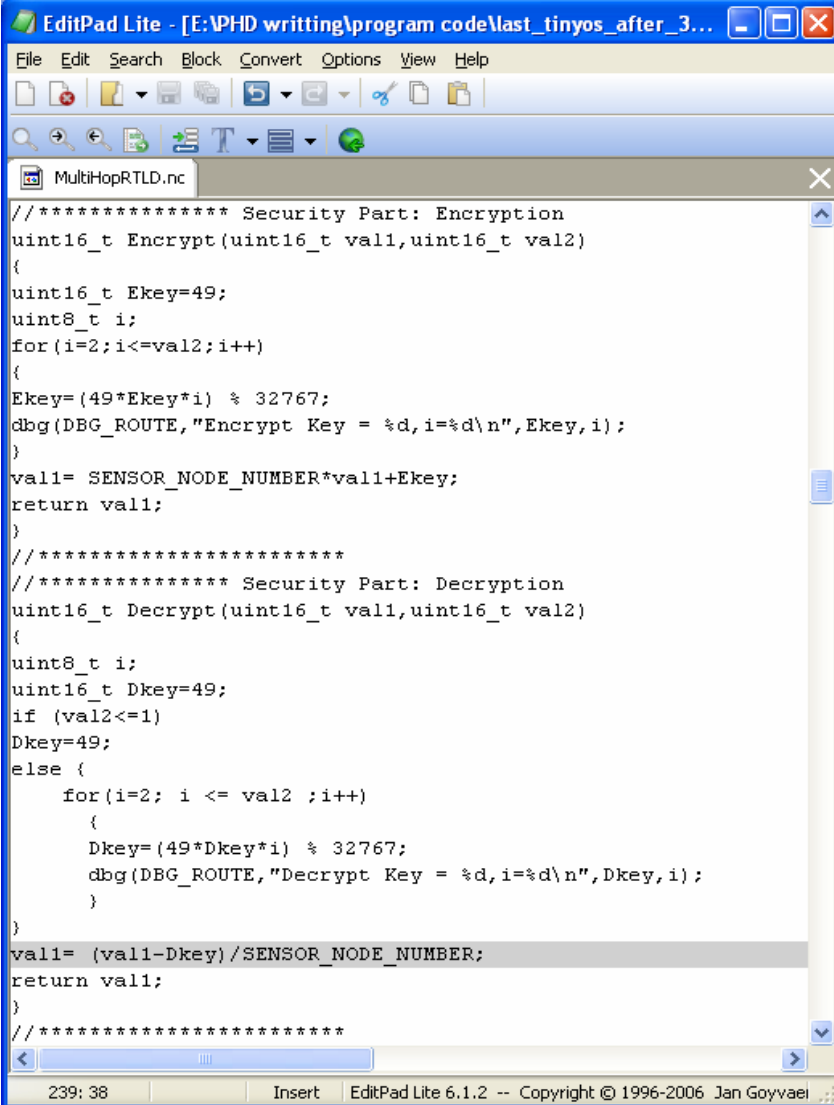
```

// Power management for transceiver code
event result_t SleepTimer.fired() {
if(sleepstate==0)
{
call CC2420Control.VREFOn();
sleepstate = 1;
}
else
{
call CC2420Control.VREFOff();
sleepstate = 0;
}
call SleepTimer.stop();
}

```

The status bar at the bottom shows "621: 27", "Insert", and "Save the current text file".

## D.5 Source Code of Security Management



```

EditPad Lite - [E:\PHD writing\program code\last_tinyos_after_3...
File Edit Search Block Convert Options View Help
MultiHopRTLD.nc
//***** Security Part: Encryption
uint16_t Encrypt(uint16_t val1,uint16_t val2)
{
uint16_t Ekey=49;
uint8_t i;
for(i=2;i<=val2;i++)
{
Ekey=(49*Ekey*i) % 32767;
dbg(DBG_ROUTE,"Encrypt Key = %d,i=%d\n",Ekey,i);
}
val1= SENSOR_NODE_NUMBER*val1+Ekey;
return val1;
}
//*****
//***** Security Part: Decryption
uint16_t Decrypt(uint16_t val1,uint16_t val2)
{
uint8_t i;
uint16_t Dkey=49;
if (val2<=1)
Dkey=49;
else {
for(i=2; i <= val2 ;i++)
{
Dkey=(49*Dkey*i) % 32767;
dbg(DBG_ROUTE,"Decrypt Key = %d,i=%d\n",Dkey,i);
}
}
val1= (val1-Dkey)/SENSOR_NODE_NUMBER;
return val1;
}
//*****
239: 38 Insert EditPad Lite 6.1.2 -- Copyright © 1996-2006 Jan Goyvaert

```

**Figure D.7** Security encryption and decryption function

The screenshot shows a window titled "EditPad Lite - [E:\PHD writting\program code\last\_tinyos\_after\_3\_paper\SRTL...". The menu bar includes File, Edit, Search, Block, Convert, Options, View, and Help. The toolbar contains various icons for file operations and editing. The active document is "MultiHopRTLD.nc". The code is as follows:

```

551     flag1 = FALSE;
552     uint16_t dist
553     uint16_t source=Decrypt(pMHMsg->sourceaddr,pMHMsg->originseqno);
554     //***** Authentication *****
555     if((SENSOR_NODE_NUMBER < source) || (source < p))
556         return 0;
557     dist=loctrc(TOS_LOCAL_ADDRESS,source);
558
559 //----- Receive Broadcast RTR -----
560 if (Msg->addr == TOS_BCAST_ADDR && dist> 1)
561

```

The status bar at the bottom shows "555: 51" and "Insert" mode. The file path in the title bar is partially visible as "E:\PHD writting\program code\last\_tinyos\_after\_3\_paper\SRTL\micaz\Mul".

**Figure D.8** Authentication after decryption