Energy Aware Real-Time Routing in Wireless Sensor Network

Developed By:
Sohail Jabbar (01-244072-035)

Supervised By:
Engr. Dr. Abid Ali Minhas

COMPUTER SCIENCE AND ENGINEERING
DEPARTMENT
BAHRIA UNIVERSITY
ISLAMABAD
Session 2007-2009

Dated: --------------------

Final Approval

This is to certify that we have read the thesis submitted by **Sohail Jabbar, Enrollment # 01-244072-035**. It is our judgment that this project is of standard to warrant its acceptance by the Bahria University, Islamabad, for the Degree of **MS in Telecommunication and Networking**.

Project Evaluation Committee

External Examiner: ___________________________

Internal Examiner: ___________________________

Supervisor:
Engr. Dr. Abid Ali Minhas ___________________________
Associate Professor
(CS&Engg. Deptt.)
Bahria University, Islamabad.

Dedication to Those Who Pray and Strive for the Highness of ISLAM (A way of peace and blessings) with Devotion and Limpidness

A Dissertation Submitted To

Computer Science and Engineering Department,

Bahria University, Islamabad

As a partial Fulfillment of Requirements for the Award of the Degree

Of

MS in Telecommunication and Networking

Declaration

I hereby declare that this Thesis "**Energy Aware Real-Time Routing in Wireless Sensor Network"** neither as a whole nor as a part has been copied out from any source. It is further declared that I have done this research with the accompanied report entirely on the basis of my personal efforts, under the dexterous guidance of my teachers especially my supervisor Engr. Dr. Abid Ali Minhas. If any of the system is proved to be copied out of any source or found to be reproduction of any project from any of the training institute or educational institutions, I shall stand by the consequences.

Sohail Jabbar
Enrollment # 01-244072-035

Acknowledgement

Praise be to God (Allah), the Cherisher and Sustainer of the Worlds (Rabbul Aalameen), Most Gracious (Al-Rehman), Most Merciful (Al-Raheem) and Master of the Day of Judgment, whose bounteous blessings enable us to pursue and perceive higher ideas of life. He is he, who sent his prophets for the guidance of Human beings and Ginns. Darood and Salaam upon his last prophet, Muhammad (Peace be upon him), his family and his companions, who has the ultimate and eternal way of complete success for this world and the hereafter in the form of Quran: the ultimate manifestation of ALLAH's grace to man, the ultimate wisdom, and the ultimate beauty of expression: in short, the word of God. After this i must mention that it was mainly due to my family's moral and financial support during my entire academic career that enabled me to complete my work dedicatedly. I would like to thanks to my respected teacher's and especially consider it a proud privileged to express my gratitude and deep sense of obligation to my reverend supervisor Engr. Dr. Abid Ali Minhas for his dexterous guidance and kind behavior during my thesis work. I would like to thank my brother, and sisters who encouraged me at those moments when I got exhausted. I also would like to say gratitude to my truly friends especially "the prayer (Nemaaz)" that helped me in every difficulty. I once again would like to admit that I owe all my achievement to my most loving parents who mean most to me, for their prayers are more precious then any treasure on earth. May they be live long with special and unlimited blessing of Allah (Subhana hu Wata'ala). Ameen Ya Rabbul AaLameen.

Sohail Jabbar
Enrollment # 01-244072-035

Project In Brief

Project Title:	Energy Aware Real-Time Routing in Wireless Sensor Network
Undertaken By:	Sohail Jabbar (01-244072-035)
Supervised By:	Engr. Dr. Abid Ali Minhas Associate Professor (CS&Engg. Deptt.) Bahria University, Islamabad.
Start Date:	February 2009
Completion Date:	August 2009
Tools & Technologies:	TinyOS 1.x TOSSIM MS Office Adobe Photo Shop 9 MS Paint Adobe Acrobat Professional 6.0
Operating System:	Windows XP
System Used:	Pentium 4 (1.7 GHz Genuine Intel) RAM 512 MB 80 GB Hard Disk

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Real-time communication demands perfection not only at hardware level but also at the software level design. Real-Time Operating System (RTOS) plays a vital role in empowering the timeliness sensitive communication. Kernel design must be sensitive to compensate the delay due to hardware interaction and layered performance. From layered view, a messy contribution is required from each layer design and its relevant protocols as well as cross layer protocols to lessen different delays. The history of plantation of ideas and protocols purely in wireless sensor network field is hardly more than a decade. Ubiquitous application of these miniatures sometimes requires tight end-to-end delay bound in surveillance, monitoring, tracking, caring and controlling. Even probabilistic end-to-end delay may lead to severe loss and failure of system. Putting constraints on performance of a system in the temporal domain, some times turns right into wrong and update into outdate. These are the scenarios where apposite value of time inveterate in the reality. But such timing precision not only requires tightly scheduled performance constraints but also requires optimal design and operation of all system components. Any malfunctioning at any relevant aspect may cases a serious disaster and even loss of human lives. Managing and interacting with such real-time system becomes much intricate when these sources are limited as in wireless sensor nodes. Real-Time routing is the core concept in QoS aware network layer issues which ensure reaching the destination within the limited required time and resources. Non-coherent data processing, effective and efficient routing protocols as well as optimally designed RTOS (Real Time Operating System) can add much contribution in the real-time context. Our proposed mechanism of approaching the sink from multi-sources in the fast and efficient manner is simply relies on in-network processed information which appreciatory decreases too much beacons and messaging among neighboring nodes resulting in increasing the network lifetime. More-over, through our proposed mechanism, every node is aware of its time to get its message to reach the destination. Thus calculation time and decision making time also decreases leading in favor to real-time communication.

INTRODUCTION

Chapter 1

Introduction

1.1. Wireless Sensor Network Technology

MEMS (Micro Electro Mechanical System) technology is one that will change the world. These MEMS devices feature the integration of mechanical sensors, actuators and operating electronics on a common silicon substrate with the use of micro-fabrication technology [1]. To measure the environmental physical quantities, numerous types of MEMS devices in our vicinity categorized in pressure sensors, vacuum sensors, optical sensors, flow sensors, intelligent sensors, accelerometers and inclinometers, chemical sensors, biosensors, LPD (Level, Position, Distance) sensors, temperature sensors and magnetic sensors are either in isolated working or networked through wired or wireless medium. Encroachment evolution in such micro and nano-technologies represents the next step in wireless communication miniaturization and their power and size. This makes it feasible to embed them into wearable vital-sign monitoring and other environmental physical quantities monitoring, sniper localization, wild life monitoring, localization and tracking tags and gears [2].

> *"WSN is cooperative network of low cost, self-organized, self-configured, and short lived, less computational power and less memory containing nodes. Each node consists of processing capability (one or more microcontroller, CPUs or DSP chips), may contain multiple types of memory (Flash memory, data or program memory), have an RF Transceiver (usually with single omni directional antenna), have a power source (energy storage or energy scavenging cells) and accommodate various sensors and actuators [1]."*

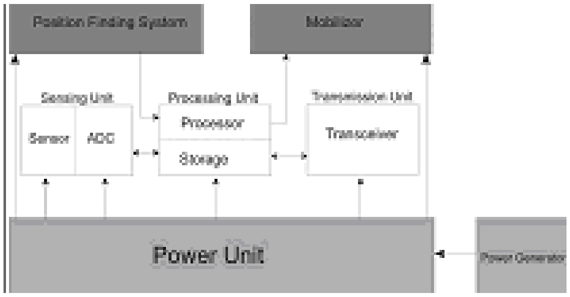


Figure 1: Typical Components of a Node

This network can be connected to the IP-based network through some gateway as shown in Figure 2. Sensor nodes are deployed either in random fashion or planted manually according to the requirements and available situations. In the outdoor environment; the LOS communication range is 75 to 100 m with ½ wave dipole antenna and in the indoor environment; the communication range is 20 to 30 m with ½ wave dipole antenna [3]. Multiple communication bands of 433 MHz, 869-915 MHz and 2.4 GHz and each with multiple channels for supple solution of different application requirements is available. Radios are half-duplex bidirectional. Signal is transmitted with a maximum data rate of 250 Kbps depending upon choice of radio and configuration. Communication data security is obtained by DSSS.

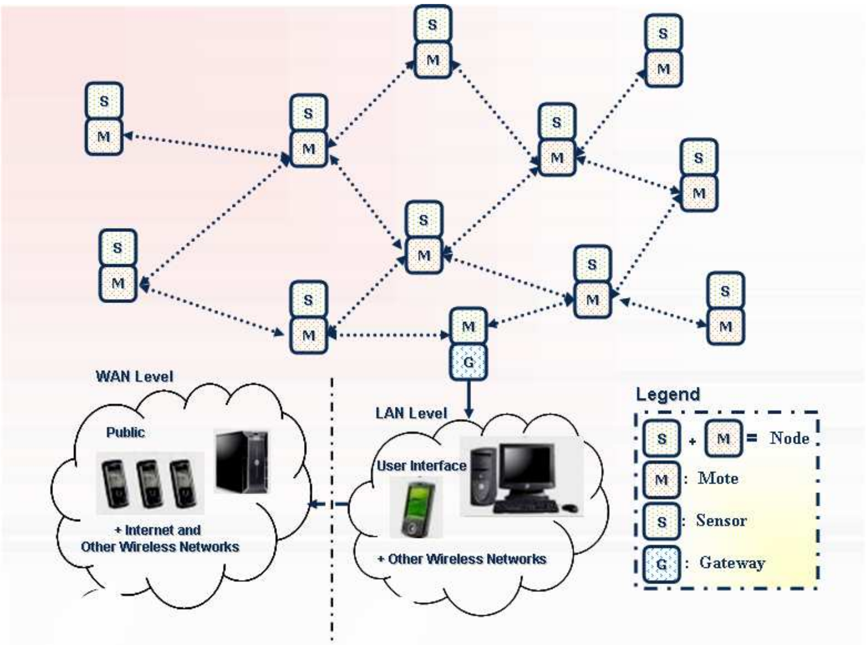


Figure 2: Wireless Sensor Networks Connected with external Network

Data dissemination from source to sink and vice versa usually requires transit nodes' positioning information either through GPS-free localization, relative localization or absolute localization techniques. Required environmental physical quantity is sensed by the source node(s) and is disseminated through the network up-to the data fusion center or base station. That information can then be used in Ethernet based networks as well as world wide by connecting the sink to the IP-based network [12] [14]. In all of these operations, special techniques should be employed to tactfully manage the constraint resources of WSN. Among these constraint resources, energy conservation is the core issue. The major energy sources of these miniature sensor nodes are either energy storage devices like batteries or energy scavenging devices like vibration or a combination of both. Solar radiation is the most abundant energy source and yields around 1mW/mm^2

($1J/day/mm^2$) in full sunlight or $1\mu W/mm^2$ under bright indoor illuminations [2]. Vibration has been proposed as an energy source that can be scavenged. So for as energy consumption is concerned, sensor acquisition can be achieved at 1 *nJ* per sample, and modern processors can perform computation as low as 1 *nJ* per instructions. Current transmission techniques (e.g. Bluetooth) consume about 100 *nJ* per bit for a distance of 10 to 100 m, making communication very expensive compared to acquisition and processing [2]. Stateless and light-weight protocols, dynamic power adjustment and different power saving modes are excellent candidate solutions in order to minimize computation and to save transmitting, receiving, sensing and processing power to add more life to the constraint resource environment of WSN.

1.2. Application of Wireless Sensor Network

The cradle of Wireless Sensor Network (WSN) research was first nourished by Defense Advanced Research Projects Agency (DARPA) through the military applications. It continues to explore this technology by funding a number of prominent research projects e.g. Smart Dust [5], Network Embedded System Technology (NEST) [6]. Other military application of WSN includes surveillance and battle field monitoring, urban warfare, vehicle and personnel movement near border area, protection of ammunition deports, military headquarters, atomic plants, oil and gas pipelines and communication towers monitoring and control. It has also its application in self-healing minefields system. Wireless sensor nodes deployment at border area for surveillance purpose and their communication with the control room is shown in

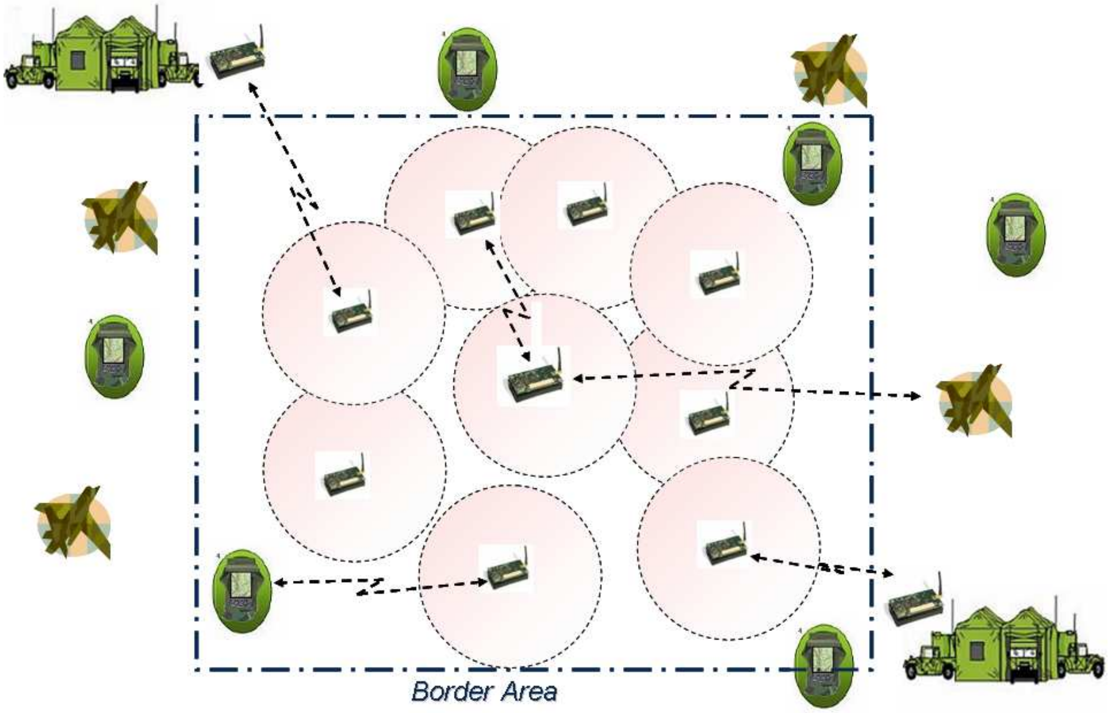


Figure 3: Wireless Sensor Nodes Deployment at Border Area

The civilian application domain of Wireless Sensor Network (WSN) has also been considered a lot, such as in general engineering, civil engineering, agriculture and environmental monitoring and health monitoring and surgery.

With the passage of time, relentless progress in hardware as well as software based technologies also make it feasible to indulge sensors in various components of the environment. Due to the employment of variety of tools and concepts, a rich and multidisciplinary area of research is offered by the field of wireless sensor network which in turn addresses a diverse set of applications. In the closer view, this emerging technology has its application in wildlife monitoring [7], cold chain monitoring [8], Glacier monitoring [9], rescue of avalanche victims [10], cattle herding, geographical monitoring, monitoring of structures, vital sign monitoring [11], ocean water and bed monitoring, monitoring of fresh water quality, tracking vehicles, sniper localization [12], volcano monitoring and tunnel monitoring and rescue. Underwater sensor network that are typically based on ultrasound, is also a key application of WSN [13] [14]. Figure 4 illustrates the emergence of wireless sensor networks of various applications.

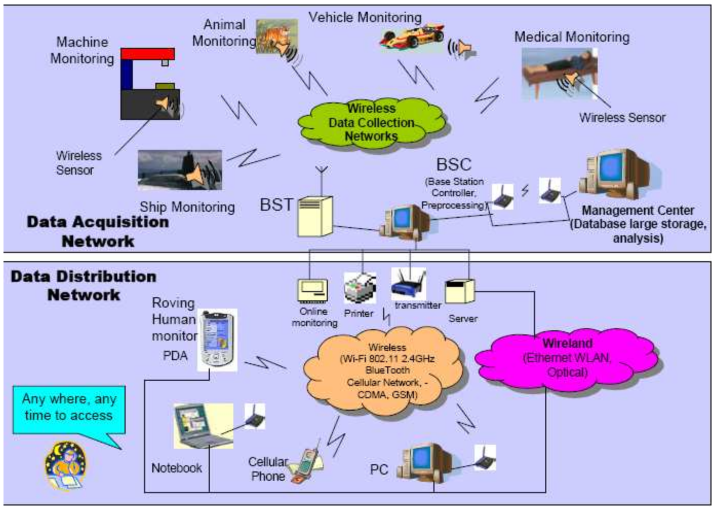
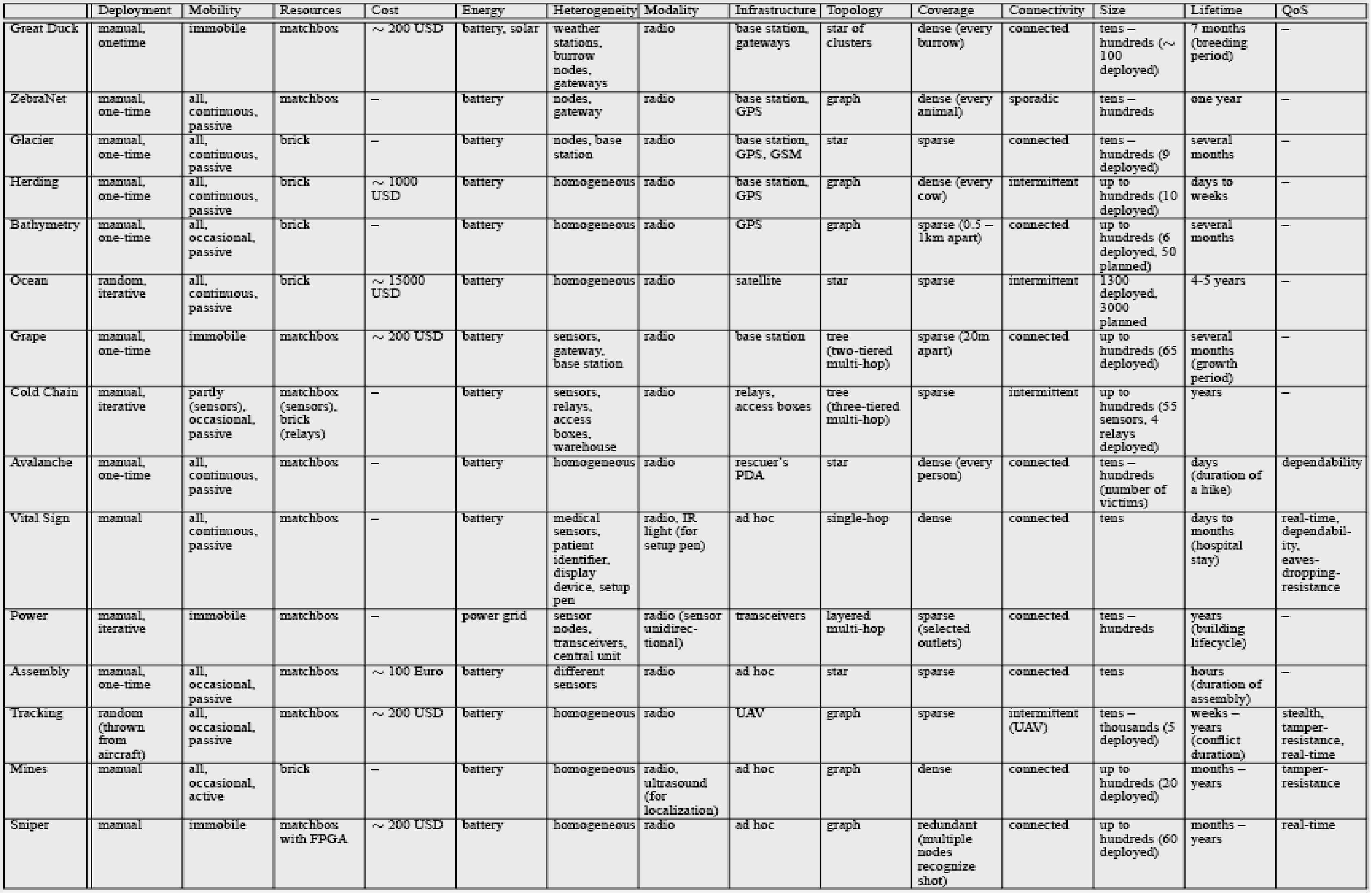

Machine Monitoring
Animal Monitoring
Vehicle Monitoring
Medical Monitoring
Wireless Data Collection Networks
Wireless Sensor
Wireless Sensor
Ship Monitoring
BST
BSC
(Base Station Controller, Preprocessing)
Management Center (Database large storage, analysis)
Data Acquisition Network
Data Distribution Network
Roving Human monitor
PDA
Online monitoring
Printer
transmitter
Server
Wireless (Wi-Fi 802.11 2.4GHz BlueTooth Cellular Network, - CDMA, GSM)
Wireland (Ethernet WLAN, Optical)
Any where, any time to access
Notebook
Cellular Phone
PC

Figure 4: Emergence of Wireless Sensor Network in Various Applications [5]

Most renowned and prominent projects and applications according to the design space are shown in **Table 1**.

Table 1: Classification of the sample applications according to the design space



	Deployment	Mobility	Resources	Cost	Energy	Heterogeneity	Modality	Infrastructure	Topology	Coverage	Connectivity	Size	Lifetime	QoS
Great Duck	manual, onetime	immobile	matchbox	~ 200 USD	battery, solar	weather stations, burrow nodes, gateways	radio	base station, gateways	star of clusters	dense (every burrow)	connected	tens – hundreds (~ 100 deployed)	7 months (breeding period)	–
ZebraNet	manual, one-time	all, continuous, passive	matchbox	–	battery	nodes, gateway	radio	base station, GPS	graph	dense (every animal)	sporadic	tens – hundreds	one year	–
Glacier	manual, one-time	all, continuous, passive	brick	–	battery	nodes, base station	radio	base station, GPS, GSM	star	sparse	connected	tens – hundreds (9 deployed)	several months	–
Herding	manual, one-time	all, continuous, passive	brick	~ 1000 USD	battery	homogeneous	radio	base station, GPS	graph	dense (every cow)	intermittent	up to hundreds (10 deployed)	days to weeks	–
Bathymetry	manual, one-time	all, occasional, passive	brick	–	battery	homogeneous	radio	GPS	graph	sparse (0.5 – 1km apart)	connected	up to hundreds (6 deployed, 50 planned)	several months	–
Ocean	random, iterative	all, continuous, passive	brick	~ 15000 USD	battery	homogeneous	radio	satellite	star	sparse	intermittent	1300 deployed, 3000 planned	4-5 years	–
Grape	manual, one-time	immobile	matchbox	~ 200 USD	battery	sensors, gateway, base station	radio	base station	tree (two-tiered multi-hop)	sparse (20m apart)	connected	up to hundreds (65 deployed)	several months (growth period)	–
Cold Chain	manual, iterative	partly (sensors), occasional, passive	matchbox (sensors), brick (relays)	–	battery	sensors, relays, access boxes, warehouse	radio	relays, access boxes	tree (three-tiered multi-hop)	sparse	intermittent	up to hundreds (55 sensors, 4 relays deployed)	years	–
Avalanche	manual, one-time	all, continuous, passive	matchbox	–	battery	homogeneous	radio	rescuer's PDA	star	dense (every person)	connected	tens – hundreds (number of victims)	days (duration of a hike)	dependability
Vital Sign	manual	all, continuous, passive	matchbox	–	battery	medical sensors, patient identifier, display device, setup pen	radio, IR light (for setup pen)	ad hoc	single-hop	dense	connected	tens	days to months (hospital stay)	real-time, dependability, eavesdropping-resistance
Power	manual, iterative	immobile	matchbox	–	power grid	sensor nodes, transceivers, central unit	radio (sensor unidirectional)	transceivers	layered multi-hop	sparse (selected outlets)	connected	tens – hundreds	years (building lifecycle)	–
Assembly	manual, one-time	all, occasional, passive	matchbox	~ 100 Euro	battery	different sensors	radio	ad hoc	star	sparse	connected	tens	hours (duration of assembly)	–
Tracking	random (thrown from aircraft)	all, occasional, passive	matchbox	~ 200 USD	battery	homogeneous	radio	UAV	graph	sparse	intermittent (UAV)	tens – thousands (5 deployed)	weeks – years (conflict duration)	stealth, tamper-resistance, real-time
Mines	manual	all, occasional, active	brick	–	battery	homogeneous	radio, ultrasound (for localization)	ad hoc	graph	dense	connected	up to hundreds (20 deployed)	months – years	tamper-resistance
Sniper	manual	immobile	matchbox with FPGA	~ 200 USD	battery	homogeneous	radio	ad hoc	graph	redundant (multiple nodes recognize shot)	connected	up to hundreds (60 deployed)	months – years	real-time

1.3. Issues in Wireless Sensor Network

Ad hoc network inherited some of the tribulations of wireless communication networks like lossy links, unreliable time in-varying and asymmetric channel, hidden node and exposed node, improperly defined coverage boundary etc. Ad hoc network puts a messy contribution in this knotted portfolio in the form of multi hop environment, location awareness, and node mobility, dynamically changing topology, vulnerability of channel and nodes and different aspects in QoS. Specifically in WSN, in addition to most of the aforementioned issues, network partitioning, localization, calibration, data fusion, aggregation and dissemination, coverage issues, self organizing and self administration, scalability, load balancing, node clustering, topology management, end-to-end delay constraint routing, security and privacy, heterogeneity, and other energy, memory, power and bandwidth constraints are the active challenges. In the closer view, node scheduling, hole problem, avoiding and coping with void node areas, node failure and QoS relating factors are under great concentration of researchers where QoS is a level of service in achieving the target with sufficient resources by fulfilling the requested QoS parameters. End-to-End performance, delay, bandwidth, energy consumption, transmission power, memory usage, probability of packet loss, jitter, bit error rate, miss ratio, packet over-head and packet success delivery are considered under the umbrella of QoS parameters. Real-time routing has the core importance in QoS-aware network layer issues which ensures catching the destination within a limited required time and resources.

Ubiquitous opportunities of these miniatures also unveil the dilemmas in energy, memory and computational prongs, specifically in the networked communication environment of wireless sensors. There also comes up some differences from the traditional wireless network like cellular network and especially from its close ancestor, ANet (Ad hoc Network) like unattendant operation for a long period of time in the energy constraint environment, densely and randomly deployed nodes, high node failure and other constraint resources. Most of the algorithms designed for wireless network and dhoc networks need to be improved due to the aforementioned grounds. Non-coherent data processing, effective and efficient routing protocols as well as optimally designed RTOS (Real Time Operating System) can add much contribution in the real-time context.

The possibility of building such emergent wireless sensor network which has its lot of applications requires some fundamental advances in enabling technologies. Hardware miniaturization is the first and foremost among these technologies. The concept of smart dust technology makes this target achievable. Smaller size of sensor, microcontroller and memory chip have driven down the power consumption but it has made their construction much complicated. Radio modems which are responsible for wireless communication have also become energy efficient. The counterpart of this basic hardware technology is software. Operating system and other communication protocols must be designed to cope with the energy, memory, processing and bandwidth issues [15].

1.4. WSN Hardware Profile

1.4.1. What is Mote?

Lexical meaning of Mote is “A very small particle; a speck” (Figure 5). The usage of this term with respect to wireless sensor network was popularized by work at University of California, Berkley in the late 1990’s. A typical mote consists of a processor and RF module. In hardware design context, the researchers’ emphasis is on its low power consumption and small size to a level to make it as small as a speck of dust (a mote) and spread it around every where. That’s why this technology is also termed as Smart Dust Technology. The Smart Dust concept was formally introduced by Kristofer S. J Pister of University of California in a publication [16]. Smart dust is a hypothetical wireless network of tiny MEMS, sensors, robots or devices that can detect (for example) light, temperature or vibration [17].

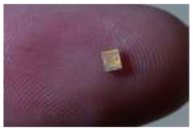


Figure 5: Speck (2003)

Various events can be recognized by adding sensor modules. These motes when clustered together, automatically create highly flexible, low-power networks with application ranging from climate control system to entertainment devices that further interact with information appliances such as smart phones, PDA’s, mobiles devices, wireless devices or IP-Networks as shown in Figure 6.

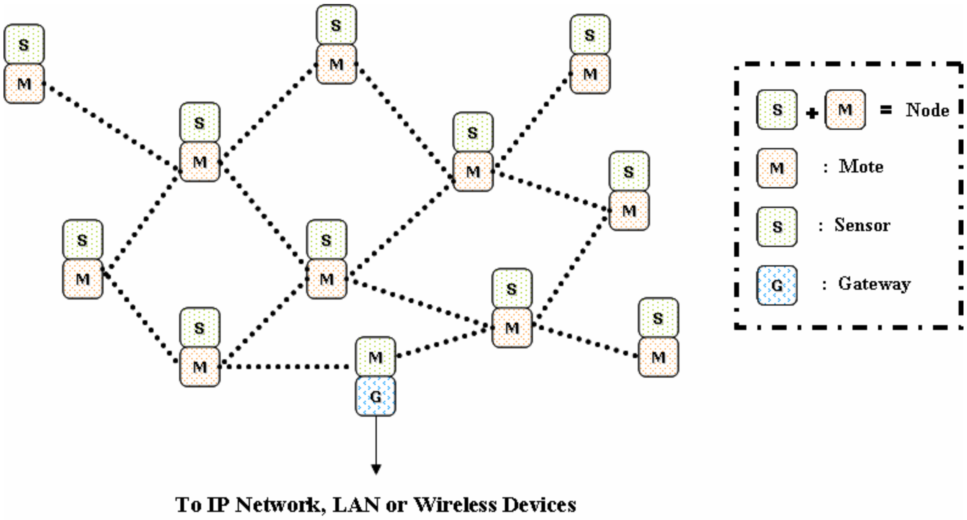


Figure 6: WSN Constructed by Motes

1.4.2. Some examples of sensor nodes

This time the sensor market is rich of sensor nodes manufacturers. There are quite a number of sensor nodes available for use in WSN R&D from number of manufacturers such as MICA Mote by UC Berkley, EYES nodes by Infineon, BTnodes by ETH Zürich, Scatterweb plateform by Freie Universität Berlin and many others. Depending on the intended application scenarios, they have to fulfill quite different requirements regarding battery, transceiver sensitivity, mechanical robustness of the node's placement, size, and so on. But the most popular and widely used are MICA motes which are not only used in academics but also in industrial fields. Nearly a hundred research groups currently use MICA nodes [18]. For our further discussion, MICA motes will be our point of concentration regarding its architecture and its evolutionary process of hardware and radio subsystem architecture.

1.4.3. What is MICA

MICA is a commercially available product that has been used widely by researchers and developers. It has all the typical features of a mote and therefore this wireless platform serves as a foundation for the emerging possibilities [18]. Nearly more than hundred research groups currently use MICA nodes. An additive edge of its design is that it is manufactured with off-the-shelf hardware.

A list of typical motes that were seen in the market is shown in **Table 2**.

Table 2: Representative Motes

	Mote								
	WeC	**RENE**	**DOT**	**MICA**	**MICA2**	**IMote**	**BTNode**	**Telos**	**MICAz**
Year	1999	2000	2001	2002	2003	2003	2003	2004	2004
Clock	4MHz	4MHz	4MHz	4MHz	7MHz	12MHz	12MHz	20MHz	7MHz
CPU	Atmel	Atmel	Atmel	Atmel	Atmel	ARM	Atmel	TI	Atmel
Flash (KB)	8	8	16	128	128	512	128	60	128
RAM (KB)	0.5	0.5	1	4	4	64	4	4	4
Baud rate	10K	10K	10K	40K	40K	460K	460K	250K	250K
Radio Type	RFM	RFM	RFM	RFM	RFM	Blue tooth	Blue tooth	ZigBee	ZigBee

1.5. Node Architecture

A typical node consists of a mote and a sensor board along with power supply and memory devices. A MICA mote is responsible for processing, storage, power supply and data communication with other nodes or base station while sensor board is responsible for sensing task. MICA mote is plugged into the sensor board as in Figure 7.

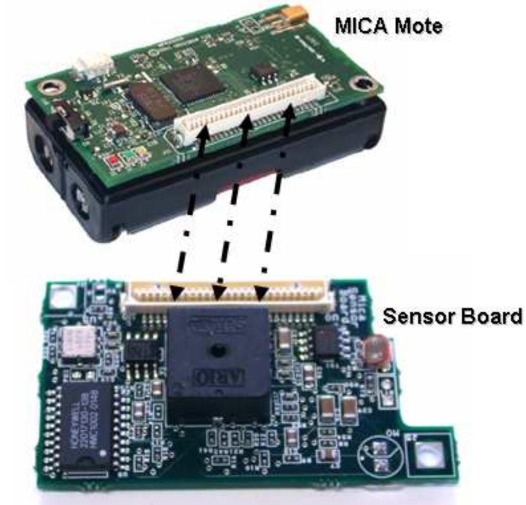


Figure 7: MICA Mote plugged into Sensor Board

A brief description of the functions of a node performed by different components is illustrated in Figure 8.

Processing
Application execution, Peripheral interaction, Resource management

I/O Sub-System
Interface with sensing boards, Program and communicate with other devices, Interface with programming boards

Secondary Storage
Stores sensor data logs, temporarily holds program images received over the network interface

Power Management
Regulate the system's supply voltage.

RF Communication
Transmit and receive data wirelessly, Coordinate with other nodes

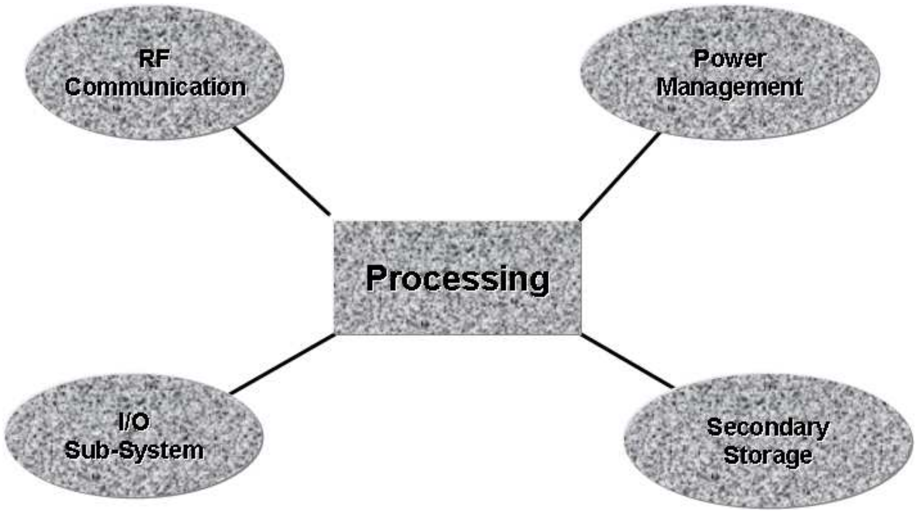


Figure 8: Logical Architecture of a Node

A basic sensor node comprises of five main components which performs the above mentioned key functions of a sensor node (Figure 9). Following is the detail of node's key components [19].

1.5.1. Controller

It can be called as the Central Processing Unit of the node. Its main job is to collect data from sensor(s) as well as from transceiver, process this data and decides when and where to send it. Moreover, it has to execute various programs ranging from communication protocols and time-critical signal processing to application programs. Various controlling architectures are available to perform such a variety of processing tasks. But there must be trade-off among flexibility, performance, energy, efficiency and costs. A controller used as the solution of above mentioned issues must have the following key characteristics.

I. Flexibility in connecting with other devices (like sensors)
II. Instruction set amenable to time-critical signal processing
III. Low power consumption
IV. Smaller in size and better to have built in memory
V. Freely programmable

The two competing solutions for the aforementioned issues are general-purpose microprocessor and micro-controllers. But later one is step ahead to have all the above mentioned five characteristics. One of the key issues in the architecture design of node is to save power consumption. Microcontrollers have the better solution for this issue by having the possibility to minimize the power consumption by going into to sleep mode.
Microcontrollers that are used in several wireless sensor node prototypes include the Atmel processor or Texas instrument's MSP 430. In older prototypes, the Intel StrongArm processor have also been used, but this is no longer considered as a practical option. But the most famous among these is Atmel. Motes using microcontrollers from

different vendors are listed in **Table 2**. Two of the Atmel microcontrollers used in MICA mote series are:

1- Atmel AVR Atmega 103L (MCU) with Atmel AVR AT90S2313 coprocessor and Maxim DS2401 memory.
2- Atmel AVR Atmega 128L (MCU) with no coprocessor as it is self reprogrammable and no Maxim DS2401memory.

First one has passed from the scene while the second one is an updated version of Atmel microcontroller using in MICAz mote. Its key characteristics are as follows.

- 133 Instructions - Most Single Clock Cycle Execution
- Up to 16 MIPS Throughput at 16MHz
- self reprogramable
- hardware multiplier
- 4K Bytes Internal SRAM
- 53 Programmable I/O Lines
- 3 hardware timers, 2 external UART, 1 SPI port
- 128k Bytes of In-System Programmable Flash
- 4K Bytes of In-System Programmable EEPROM
- JTAG debugging support (real-time, in-system debugging)

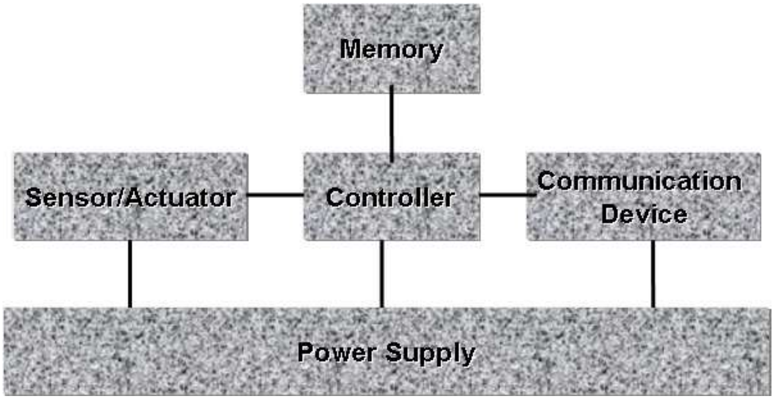


Figure 9: Main Sensor Node Hardware Components

1.5.2. Memory

MICA mote contains three types of memory. Electronically Erasable Programmable Read Only Memory (EEPROM), program flash memory and logger flash memory. Code can be stored in EEPROM or program flash memory. Logger flash memory acts as RAM to store data over time that is acquired by sensing or through transceiver or after processing for later usage. MICAz mote contains 4K bytes EEPROM, 512K bytes logger flash memory and 128K bytes program flash memory. All Motes feature a 4-Mbit serial flash (Atmel AT45DB041) for storing data, measurements, and other user-defined information. The AT45DB041B is an (Serial Peripheral Interface) SPI compatible serial interface Flash memory ideally suited for a wide variety of digital voice-, image-, program code- and data-storage applications. It consumes only 15 mA of current when writing data.

1.5.3. Communication Device

Transceiver (‘Trans’ for Transmitter and ‘ceiver’ for Receiver) is the device used for communication. It exchanges data between individual nodes. RFM TR 1001, the Chipcon CC 1000 and CC 2420 (802.15.4 ZigBee) and Infineon TDA525x family transceiver are commonly used in WSN which are customized to different regulatory restriction on carrier frequency in different regions of the World. MICA mote uses TR 1000; MICA 2 mote uses CC 1000 while MICAz uses CC 2420 transceiver. Selecting the transmission medium for wireless communication in the energy, memory and bandwidth constraint environment of wireless sensor network has its own concern. Radio Frequency (RF), Optical Communication (LASER) and infrared are the candidates as wireless transmission media. LASER is low energy consuming technology but line of sight (LOS) and favorable atmospheric conditions are required as it is less prone to the environmental factors. So for as the usability of infrared is concerned, it needs no LOS but the Communication range is limited. For WSN the most appropriate choice for communication is Radio Frequency (RF). MICA motes uses Industrial, Scientific, Medical (ISM) band (433 MHz to 2.4 GHz).

1.5.4. Sensor/Actuator

WSN would be beside the right point entirely without the actual sensor and actuator. A mini device called sensor produces the measurable information based on the given parameter from the area of interest and is digitized by ADC which further sent to microcontroller for further processing. A long list of sensor for measuring different environment physical quantities is available. Different measurements for wireless sensor network are shown in **Table 3** [20] roughly categorized the sensor into three categories.

Passive, omni-directional sensor: These sensors are passive in the sense that they measure the physical quantity at the point of the sensor node without actually manipulating the environment by active probing. More-over self-powered capability exists in some of these sensors as they obtain the energy they need from the environment since energy is only needed amplify their analog signal. Examples include thermometer, light sensors, smoke detectors, mechanical stress or tension in material, humidity, vibration, microphones, and so on. These types of sensors have no notion of direction of measurement.

Passive, narrow-beam sensors: These sensors have well-defined notion of direction of measurement even these sensors are passive as well. Typical example is a camera, which can “take measurements” in a given direction and rotated as needed.

Active sensors: These types of sensors actively probe the environment, for example, some types of seismic sensors, sonar or radar sensor.

Actuators are just about as diverse as sensors [19]. They may take actions in response to the sensed and processed environmental physical quantity.

1.5.5. Power Supply

It regulates the system's supply voltage. Battery produces energy between 3.2V and 2.0V. Maxim1678 DC-DC converter provides a constant 3.0V supply (Figure 10).Converter takes input voltage down to 0.8V and boosts it to 3.0V. A solid 3V supply is required for radio operation. Lower voltage can be used to conserve energy when the radio is not in use or in idle state.

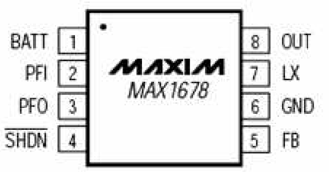


Figure 10: Maxim1678 DC-DC converter

The major energy sources of these miniature sensor nodes are either energy storage devices like batteries or energy scavenging devices like vibration or a combination of both. Solar radiation is the most abundant energy source and yields around 1mW/mm^2 (1J/day/mm^2) in full sunlight or 1μW/mm^2 under bright indoor illuminations [2]. Vibration has been proposed as an energy source that can be scavenged. So for as energy consumption is concerned, sensor acquisition can be achieved at 1 *nJ* per sample, and modern processors can perform computation as low as 1 *nJ* per instructions. Current transmission techniques (e.g. Bluetooth) consume about 100 *nJ* per bit for a distance of 10 to 100 m, making communication very expensive compared to acquisition and processing [2].

Table 3: Measurements for Wireless Sensor Network [4]

MEASUREMENTS FOR WIRELESS SENSOR NETWORK		
	Measurements	Transduction Principle
Physical Properties	Pressure	Piezoresistive, Capacitive
	Temperature	Thermistor, Thermo-Mechanical, Thermocouple
	Humidity	Resistive, Capacitive
	Flow	Pressure change, Thermistor
Motion Properties	Position	E-mag, GPS, Contact Sensor
	Velocity	Doppler, Hall Effect, Optoelectronic
	Angular Velocity	Optical encoder
	Acceleration	Piezoresistive, Piezoelectric, Optical fiber
Contact Properties	Strain	Piezoresistive
	Force	Piezoelectric, Piezoresistive
	Torque	Piezoresistive, Optoelectronic
	Slip	Dual torque
	Vibration	Piezoelectric, Piezoresistive, Optical Fiber, Sound, Ultrasound
Presence	Tactile/Contact	Contact switch, Capacitive
	Proximity	Hall Effect, Capacitive, Magnetic, Seismic, Acoustic, RF
	Distance/Range	E-mag (Sonar, Radar, Idar), Magnetic, Tunneling
	Motion	E-mag, IR, Acoustic, Seismic (Vibration)
Biochemical	Biochemical Agent	Biochemical transduction
Identification	Personal feature	Vision
	Personal ID	Fingerprints, Retinal scan, Voice, Heat plume, Vision motion analysis

A little description of transduction principles is listed below:

Piezoresistive: The Piezoresistive effect describes the changing electrical resistance of a material due to applied mechanical stress

Piezoelectric: The ability of certain crystals to generate a voltage in response to applied mechanical stress

Capacitive: A capacitor is a passive electrical component that can store energy in the electric field between a pair of conductors

Thermistor: The word is a portmanteau of thermal and resistor. A Thermistor is a type of resistor with resistance varying according to its temperature.

Thermocouple: A device that senses temperature changes by using a pair of joined wires made of dissimilar metals that produce a voltage that changes with temperature

Thermo mechanical: Of or pertaining to the variation of the mechanical properties of a material with temperature

Resistive: relating to electrical resistance

Doppler: Ultrasound device that a technician may use to sense the presence or absence of flow

Hall Effect: An effect used in the measurement of magnetic fields

Optoelectronic: that deals with the interaction of light with electronic devices, or the production of light from such devices

1.6. Profile of MICAz Mote

The MICAz is a 2.4GHz, IEEE 802.15.4 compliant, Mote module used for enabling low-power, wireless, and sensor networks. The MICAz Mote features several new capabilities that enhance the overall functionality of Crossbow's MICA family of wireless sensor networking products. MICAz runs TinyOS 1.1.7 and higher. It provides Wireless Communications with Every Node as Router Capability.

Figure 11: MPR2400 (MICAz)

The CC2420 is a true single-chip 2.4 GHz IEEE 802.15.4 compliant RF transceiver designed specifically for low-power and low-voltage wireless applications. CC2420 includes a digital direct sequence spread spectrum baseband modem providing an effective data rate of 250 kbps. MPR2400CA (Figure 11) is a single processor board can be configured to run sensor application/processing and the mesh networking radio stack simultaneously. A typical block diagram of MPR2400 is shown in Figure 12.

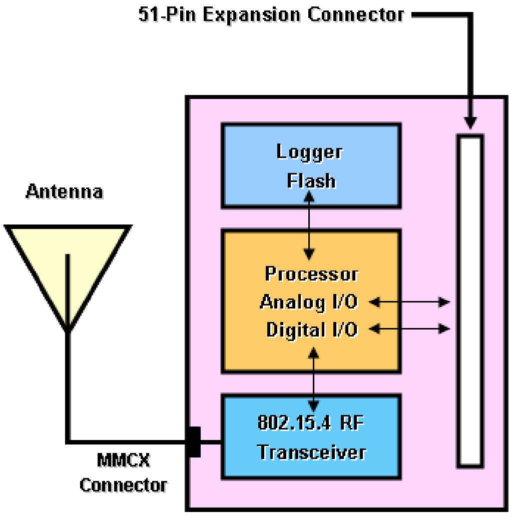


Figure 12: MPR2400 Block Diagram

All antennas pose a "Keep-Out" region. Batteries, sensors and circuits boards should not be in this rear-field region due to electromagnetic interference. The area of this region is calculated as:

1.7. Sensor, Data Acquisition Boards

MICA mote can be plugged into the sensor/data acquisition board for acquiring the environmental physical measurements (Figure 7). The MTS300 (Figure 13a) and MTS310 (Figure 13b) are flexible sensor boards with a variety of sensing modalities. These modalities can be exploited in developing sensor networks for a variety of applications including low-performance seismic sensing, vehicle detection, robotics, movement, acoustic ranging, and other applications. The design of MICAz is backward compatible regarding connectivity with sensor boards. Hence all of these boards can be connected to the MICAz via the standard 51-pin expansion connector [21].

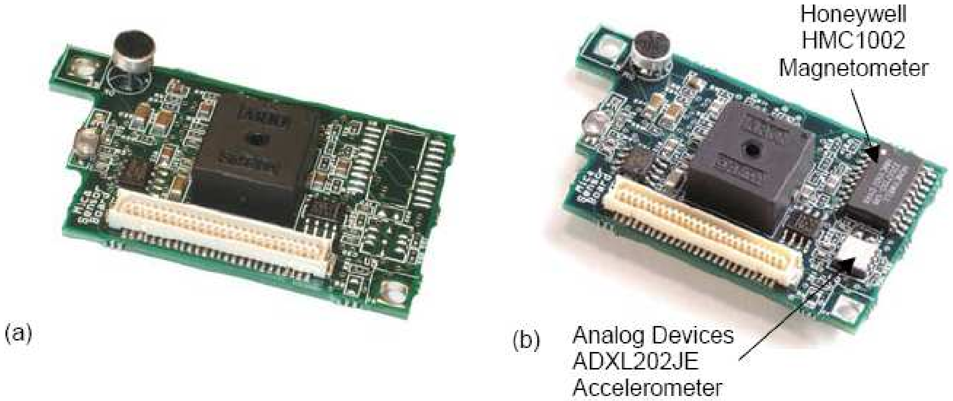


Figure 13: (a) MTS300 and (b) MTS310 with the accelerometer and magnetometer highlighted

For the identification of Motes deployment in the inaccessible environments, the GPS module offered on the MTS420 (Figure 14) [21]. It can also used for location tracking of cargo, vessels, vehicles, and wildlife and for other related application.

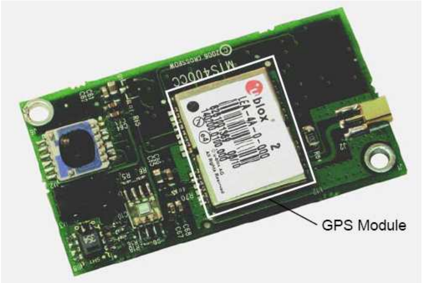


Figure 14: Photo of MTS420CC. The MTS400 does not have the GPS module (highlighted by the box)

1.8. Evolution in Architecture of MICA mote Series

Necessity is the mother of invention comes up this quotation true that “Variability is the only constant in this created universe. Due to the employment of variety of tools and concepts, a rich and multidisciplinary area of research is offered by the field of wireless sensor network which in turn addresses a diverse set of applications. To cope with the challenges offered by these diverse set of application, relentless progress in hardware as well as software based technologies also make it feasible to indulge sensors in various components of the environment [18].

1.8.1. MICA mote

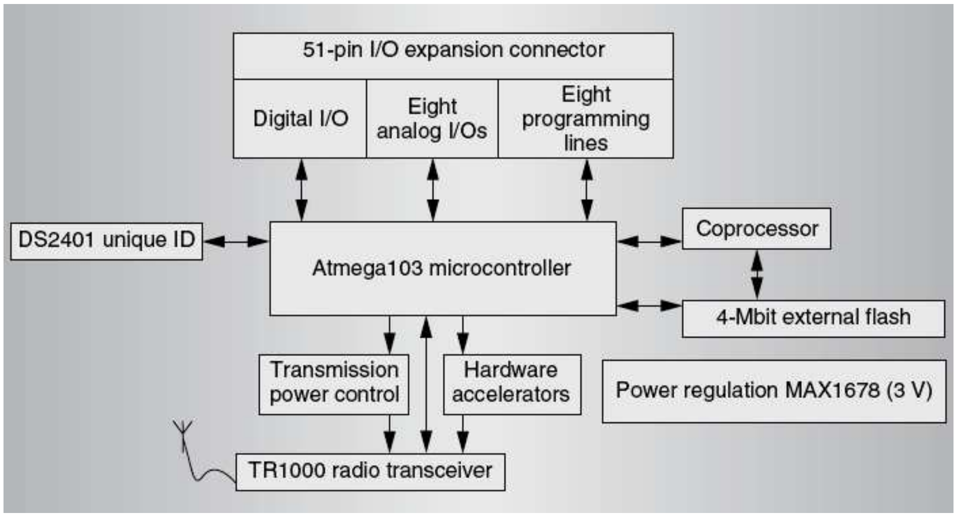


Figure 15: MICA Architecture

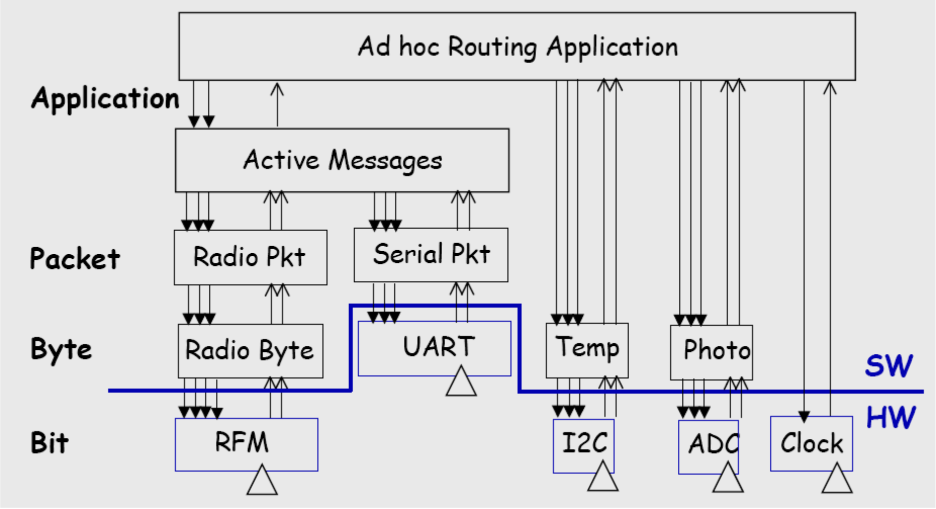


Figure 16: MICA radio Sub System Architecture

UC Berkley lab contributes a lot in this context and presented a series of MICA mote versions. Figure 15, Figure 16, Figure 17, Figure 18, Figure 19, Figure 20 shows this evolutionary process in the MICA architecture and MICA radio subsystem architecture [18].

1.8.2. MICA2 mote

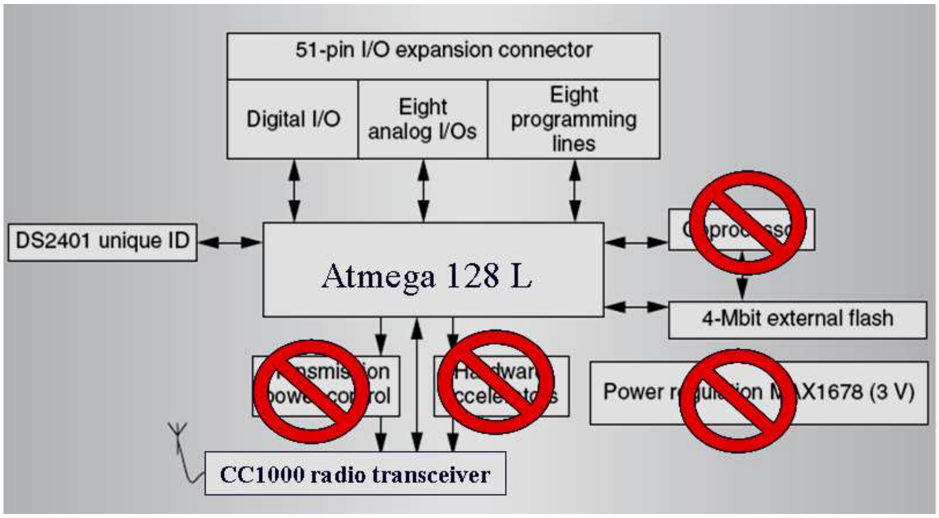


Figure 17: MICA2 Architecture

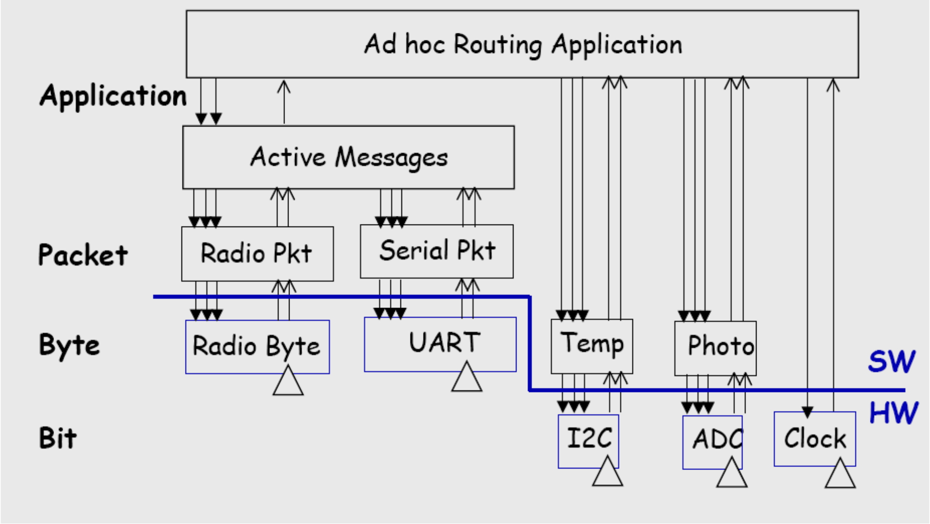


Figure 18: MICA2 radio Sub System Architecture

1.8.3. MICAz mote

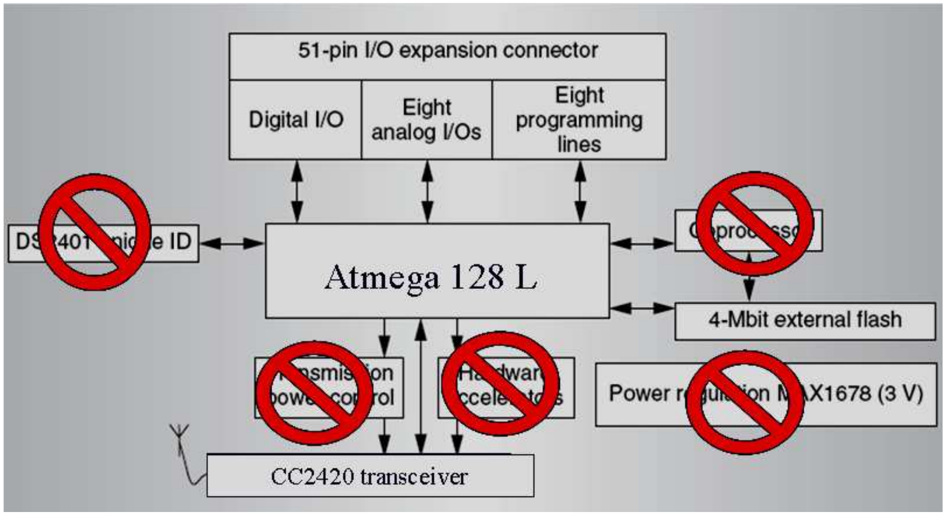


Figure 19: MICAz Architecture

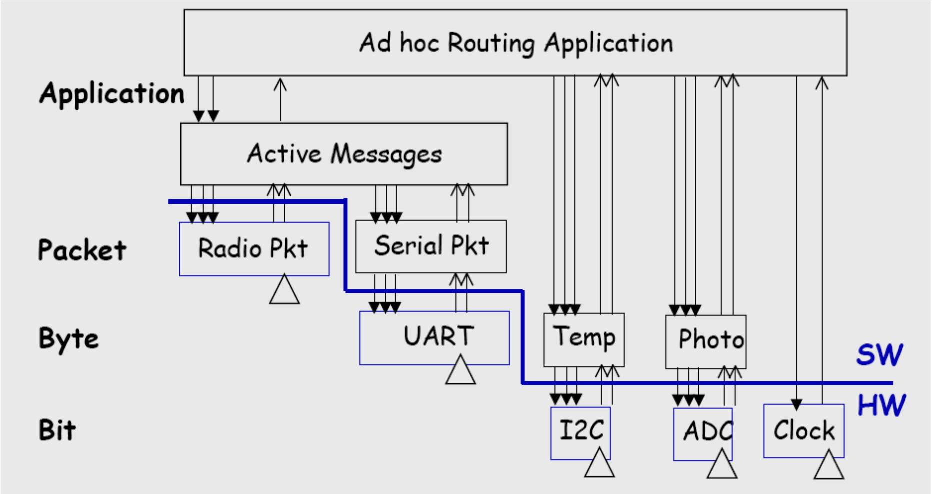


Figure 20: MICAz radio Sub System Architecture

1.9. WSN Software Profile

If hardware is body then software is its “soul”. Without soul, body is dead. To get alive this body, different forms of souls have become the part of this body. Embedded software (firmware) is one, Operating System (OS) is second and communication layer protocol is another one. Some are under the custody of manufacturers (firmware) and some are from system programmer’s end (OS, Drivers and Protocols). Operating System plays a vital role in empowering the timeliness sensitive communication. It Kernel design must be sensitive to compensate the delay due to hardware interaction and layered performance. From layered view, a messy contribution is required from each layer design and its relevant protocols as well as cross layer protocols to lessen different delays. Non-coherent data processing, effective and efficient routing protocols as well as optimally designed OS add much contribution in the this context. In this section, we will present a prolific discussion on OS for sensor networked embedded system. In the next and proceeding sections, WSN Protocols will be under concentration.

1.10. Real-Time Operating System

> *"A Real-Time Operating System (RTOS) is a program that schedules execution in a timely manner, manages system resources and provides a consistent foundation for developing application code"* [22]

RTOS is a key part to many embedded systems providing a software platform for writing application specific code in a solid, reusable, reliable and compact manner. As the embedded system is a special purpose system designed to perform one or a few dedicated function often with real-time computing constraint [23]. So the operating system used in such systems must also be for special application e.g. TinyOS is an OS for wireless network embedded systems especially used in sensor nodes. In addition, RTOS can be easily tailored to use only particular application related components. So for as, the functional similarities and differences of RTOS to General Purpose Operating System (GPOS) is concerned, it is illustrated from Table 4.

Table 4: Similarities and Differences of RTOS from GPOS

Similarities	Differences
• Abstracting the Hardware from the Software • Provision of underlying OS services to application • Some level of multitasking • Software and hardware resource management	• Better reliability in embedded application context • Faster performance • Ability to scale up and down to meet application needs • Reduced memory requirement • Better portability to different hardware platform • Scheduled policies tailored for real-time embedded system

The building blocks of the edifice of RTOS may contain Kernel, Network Protocol Stack, a File System module and other components required for a particular application (Figure 21).

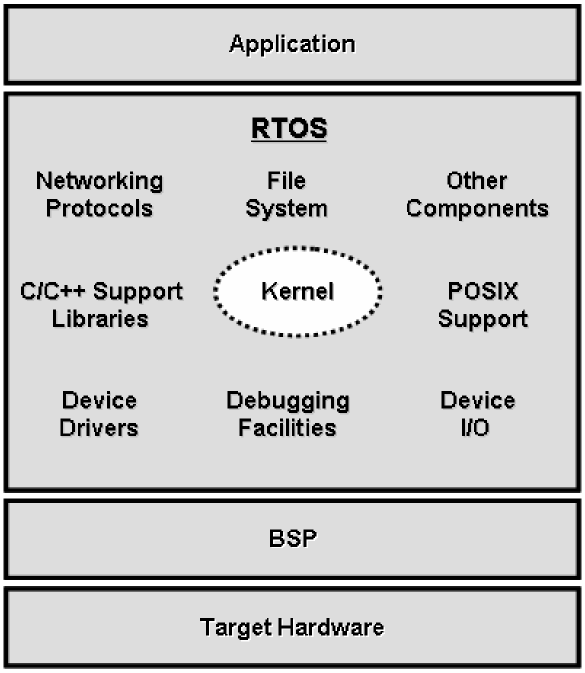


Figure 21: A high level view of RTOS components

Every RTOS has a kernel which the core supervisory software responsible for providing scheduling, resource management algorithms and scheduling. Commonly, RTOS kernel contains the following components (
Figure 22).

Scheduler: is at the heart of kernel and determines by following a set of algorithms that which task execute when. Some common examples of scheduling algorithms include round-robin and preemptive scheduling.

Object: is special kernel constructs that helps for real-time embedded system application development. Common kernel objects are tasks, message queues and semaphores.

Services: are the operations that the kernel performs on an object. These services comprise the set of Application Programming Interface (API) calls that can be used to perform operations on kernel objects or general operations like timing, resource management and interrupt handling.

Now if we study the RTOS by only considering the typical scenario of WSN, then RTOS must have some specific and appropriate programming model for the energy, memory and processing constraints environment of Wireless Sensor Network. More-over concurrency control is a crucial question for the programming paradigm of RTOS because the nodes have to handle data communicating from arbitrary sources. E.g. multiple sensors sense the physical environmental quantity or the radio transceiver receives the signal from the neighboring node at arbitrary points in time. So, a simple and sequential programming model for concurrency control is clearly insufficient.

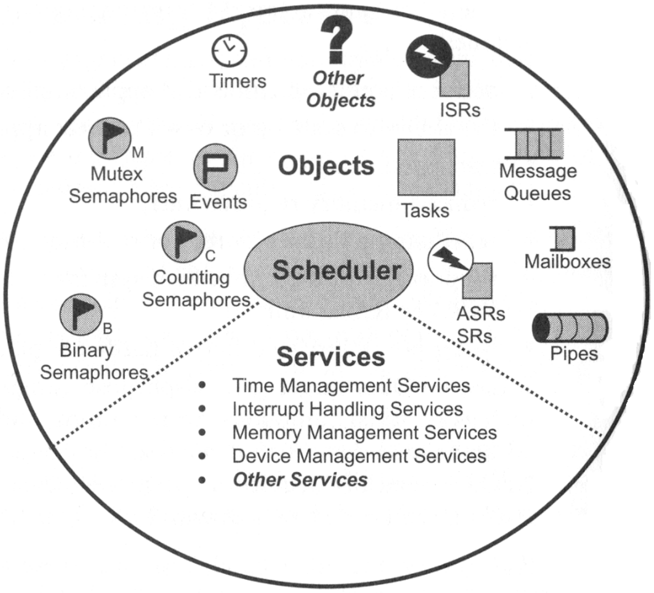


Figure 22: Common Components in an RTOS kernel that includes objects, the scheduler, and some services

Two concurrency models that seem to be suitable for such multitasking environment are process-based concurrency model and event-driven programming model.

- First candidate (process-based concurrency model) is used in GPOSs. It supports concurrent execution of multiple processes on a single CPU. But it is inappropriate and fits ill with the stringent memory constraint of sensor node. Because each process requires its own stack space in memory as well as equating individual protocol layers or functions with individual process would entail a high overhead in switching from one process to another.

- Second candidate for concurrency model to be adopted by RTOS in Wireless Sensor Network is event-based programming model. It embraces the reactive nature of WSN node and integrates it into the design of OS. On happening an event, a short sequence of instruction just stores the fact of event happening and some necessary related information. The actual processing of this stored information is not done in these event handler routine but separately decoupled from the actual occurrence of events. Thus this concurrency model has two threads of execution. One is task and the other is event handler. Event handler can interrupt the processing of task but vice versa is not allowable. While event handlers can not interrupt each other. As a consequence, one context is of time-critical event handlers where execution interruption is not permissible. Second is for the processing of task which is only triggered by the event handler.

The functional procedures of two inadequate programming models for WSN operating systems, purely sequential execution and process based execution, are illustrated in Figure 23.

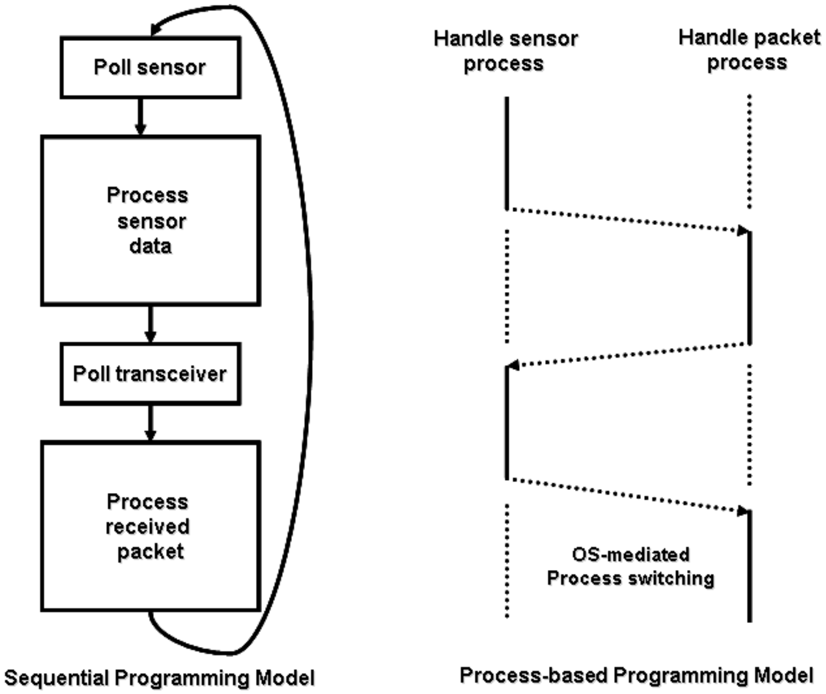


Figure 23: Two Inadequate Programming models for WSN operating system

Event-based programming model (Figure 24) seems to be the most preferable concurrency control model for the RTOS in the Wireless Sensor Network scenario due to its harmonizing nature with the reactive nature of WSN nodes.

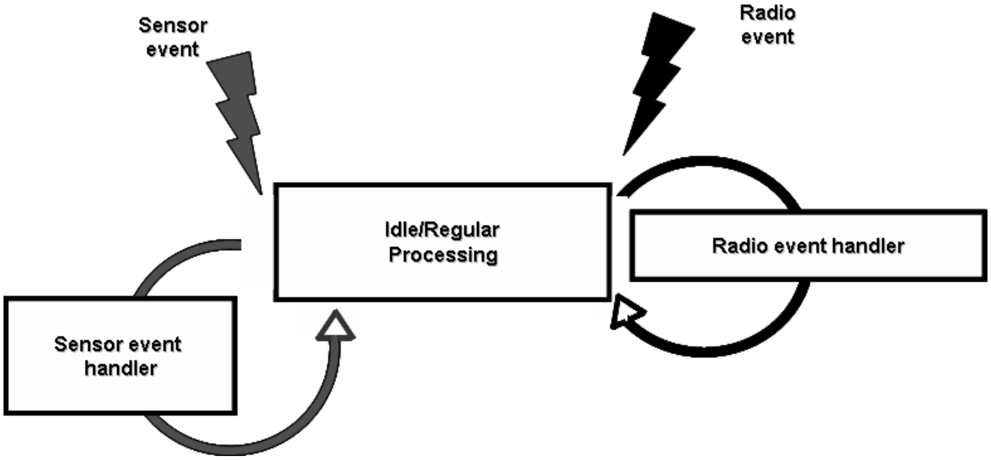


Figure 24: Event-based Programming Model

1.10.1. Example of OS for Embedded Systems: TinyOS

A typical example of such characteristics containing most widely used OS in embedded systems is TinyOS [29]. It is a small, open source, energy efficient and component based operating system designed by UC Berkley. Although it is designed specifically for networked sensors yet it is also used in other embedded systems. TinyOS supports modularity and event-based programming by concept of components. nesC — TinyOS's implementation language—was designed to cope with stringent energy, memory and processing constraints challenges of wireless sensor network. It is a C-based language with two distinguishing features: a programming model where components interact via interfaces, and an event-based concurrency model with run-to-completion tasks and interrupt handlers [24].

1.10.2. TinyOS Architecture

To cut short the size of TinyOS, its boundary is confined to include only the components that are direly needed to fulfill the requirements for a sensor network embedded system. It has no Process Management, no function pointer, no heap, no Virtual Memory, no Dynamic Allocation, no Blocking and no Exception Handling. TinyOS consists of Task Scheduler and a set of Components. A typical schematic diagram of TinyOS architecture is shown in Figure 25.

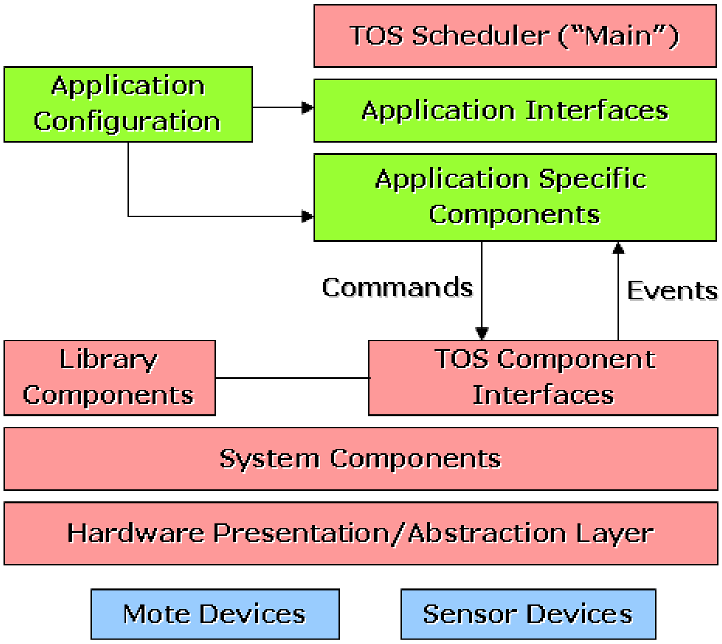


Figure 25: TinyOS Architecture

TinyOS component contain semantically related functionalities like handling the sensing interface or handling the radio interface. TinyOS contain two types of components: Configuration and Module. Former contains wires to assemble different components together and the later contains behavior for providing the application code. So, the components comprises of the required state information in a frame, the program code for normal task and handlers for commands and events.

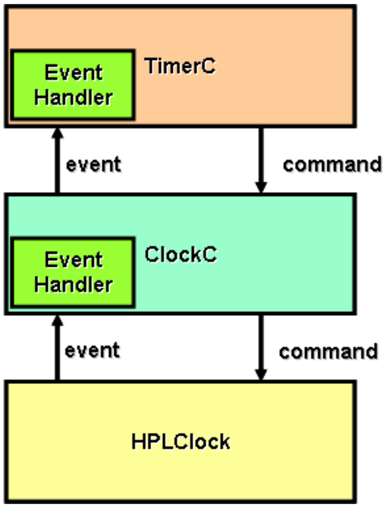


Figure 26: Hierarchical Arrangement of Commands and Events

Hierarchical arrangement of components is from low level components closer to hardware to high level components making up the actual application. Events originate in the hardware and pass upward from low level to high level components while the commands pass from high level to low level components (Figure 26). These commands and events are the only way of interaction between the components. A component can invoke certain commands at its lower component and receive certain event from it. Due to which structuring commands and events that belong together form an interface between two components. Components provide certain interfaces to their use and may also use other interfaces from underlying components.

Apart from TinyOS, there is also a few other execution environments or operating system available especially designed for WSN nodes (Table 5). A short description of some of these OSes is as follows.

Stefan Dulman and Paul Havinga [26] have proposed an operating system for a European project named as Energy Efficient Networks (EYES). They included the characteristics of small size, power awareness, distribution and reconfiguration in their project.

Another operating system "Contiki" [26] for WSNs is presented by Adam Dunkels et al. [27] implemented in the C language. It is teamwork of European Research Consortium for Informatics and Mathematics (ERCIM) [28]. Contiki is open source operating system for embedded systems, it is portable and supports multi-tasking. It has been tested for a platform Embedded Sensor Board (ESB). Contiki has been ported to a number of microcontroller architectures, including the Texas Instruments MSP430 and the Atmel AVR.

In [30] Lin Gu has proposed a new technique for the development of an operating system with low cost and high performance perimeters using t-kernel. *The t-kernel mitigates the constraints on the sensor nodes so that we can extend the functionality of software services on these platforms and include better programming support, such as object* *oriented programming, which was not available on severely resource constrained platforms before* [30].

A Sensor Network Operating System (SOS) by Chih-Chieh Han et al. [31]. In TinyOS [29] there is a problem of memory protection. Authors have also mentioned that some interrupt concurrency bugs are caught by the nesC compiler. Some improvements in this regard have done in their work. Major ideas behind SOS are the same as that of TinyOS. TinyOS is widely used if compared with SOS. Further details of SOS e.g. presentation, description etc. can be found from its web address [32]. In TinyOS version 2.x, many improvements have been made.

Table 5: Execution Environments or Operating Systems for WSN Nodes [33]

Sensor Node Name	Microcontroller	Tranceiver	Program + Data Memory	External Memory	Programming	Remarks
BTnode	Atmel ATmega 128L (8 MHz @ 8 MIPS)	Chipcon CC1000 (433-915 MHz) and Bluetooth (2.4 GHz)	64 + 180K RAM	128K FLASH ROM, 4K EPROM	C and nesC	BTnut and TinyOS Support
Dot	ATMEGA163		1K RAM	8-16K Flash	weC	
IMote	ARM core 12 MHz	Bluetooth with the range of 30m	64K SRAM	512K Flash		TinyOS Support
MICA	Atmel ATMEGA103 4 MHz 8-bit CPU	RFM TR1000 radio 50 kbit/s	128+4K RAM	512K Flash	nesC	TinyOS Support
MICA2	ATMEGA 128L	Chipcon 868/916 MHz	4K RAM	128K Flash	nesC	TinyOS, SOS and MantisOS Support
MICAz	ATMEGA 128	802.15.4/ZigBee compliant RF transceiver	4K RAM	128K Flash	nesC	TinyOS, SOS, MantisOS and Nano-RK Support
RENE	ATMEL8535	916 MHz radio with bandwidth of 10 kbit/s	512 bytes RAM	8K Flash		TinyOS Support
Iris Mote	Atmel ATmega 1281	Atmel rf230	8K RAM	128K FLASH ROM, 4K EPROM	C and NesC	TinyOS Support

1.11 Publications arising from this thesis

1.11.1 Conference Papers

- **Sohail Jabbar**, Abid Ali Minhas, Raja Adeel Akhtar, "***SPERT: A STATELESS PROTOCOL FOR ENERGY-SENSITIVE REAL-TIME ROUTING FOR WIRELESS SENSOR NETWORK***", presented at the Third International Conference on Information & Communication Technologies *(ICICT'09)*, August 15th to 16th, 2009, Karachi, Pakistan.

- **Sohail Jabbar**, Abid Ali Minhas, Raja Adeel Akhtar, Muhammad Zubair Aziz, "***REAR: REAL-TIME ENERGY-AWARE ROUTING FOR WIRELESS SENSOR NETWORK***", presented at the The 8th International Conference on Pervasive Intelligence and Computing (*PICom09*), December 12-14, 2009, Chengdu, China.

- **Sohail Jabbar**, Abid Ali Minhas, "***ENERGY-AWARE REAL-TIME COMMUNICATION IN WIRELESS AD-HOC MICRO SENSORS NETWORK***", presented at the 7th event of Frontiers of Information Technology *(FIT'09)* Ph.D. Symposium, December 16-18, Abottabad, Pakistan

WIRELESS SENSOR NETWORK ROUTING PROTOCOLS

Chapter 2

Wireless Sensor Network Routing Protocols

2.1. What is Routing?

> *Routing is a process of selecting paths in the network along which to send network traffic destined to certain destination.*

Node sends packet to its neighboring node(s) which is closer to the destination. This packet is further passed on to the node(s) closer to the destination. This passing on process, which is called forwarding, continues and ultimately the packet reaches the destination. There may exist one-to-many or many-to-one or many-to-many relationship between source and destination. Due to which routing may follow static path or dynamic path as well as can be Unipath or Multipath depending on the designed algorithm and required scenario. In their delivery semantics, routing schemes differ to form following four types (Figure 27).

- **Unicast**: transmit the packet to a single specified node.
- **Broadcast**: transmit the packet to all nodes in the radio range.
- **Multicast**: transmit the packet to a specified group of interested nodes.
- **Anycast**: transmit the packet to any one nearest to the source out of a group of nodes.

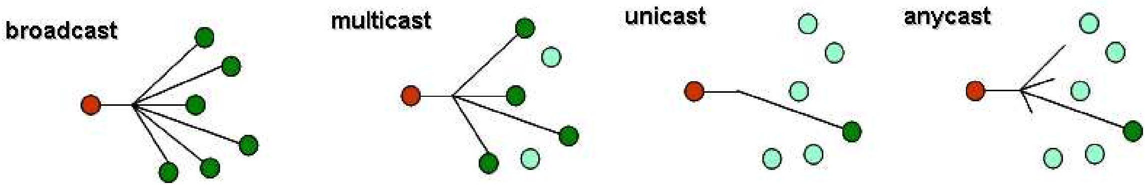


Figure 27: Routing Schemes

2.2. Routing in WSN

The various famous existing protocols for ad-hoc network are not suitable for power, memory and bandwidth constraint environment of Wireless Sensor Network. Ad-hoc network protocols like AODV [34], DSR [35], TORA [36], LAR [37] and other has not specifically designed to cope with the stringent and constraint challenges of sensor network. More-over the publication years of most of these protocols is before the advent of WSN technology. As the history of plantation of ideas and protocols purely in wireless sensor network field is hardly more than a decade. In its early stages, most of the WSN routing protocols inherit the ideas from existing Ad-hoc routing protocols like GPSR [38]. Later on new ideas came on the scene specially targeting the unique characteristics of WSN and addressing the new challenges faced by the WSN technology due to its ubiquitous application.

2.3. Routing Challenges and Design Issues in WSN

Researchers have to cope with multi-facet challenges in the un-reliable and fluctuating environment of wireless sensor network. So, these pinching prongs must be overcome to achieve efficient and reliable communication. Jamal et al. listed down some of the main routing challenges and design issues in [39] which are bulleted with short description as follows.

Node Deployment:
Can be deterministic or randomized

Data Reporting Model:
Different scenario my exists like Time-driven, Event-driven, Query-driven or Hybrid

Node/Link Heterogeneity:
Sensor nodes can have different roles and capability

Fault Tolerance:
Multiple levels of redundancy in case of node failure or environmental interference

Scalability:
Must care for the scale of node's deployment density

Network Dynamics:
Mobility of source or destination (source or sink)

Transmission media:
Wireless medium is itself an issue

Connectivity:
High density and possible random distribution of nodes

Coverage:
Actual radio range of nodes

Data Aggregation:
For duplicate suppression, minima, maxima or average

Quality of Service:
Bounded latency for data delivery

Energy Consumption without losing accuracy:

2.4. Categorization of WSN Routing Protocols

In Wireless Sensor Network, a special approach to define multicast group is by specifying the geographic region so that all node in this specified region should receive the packet. This technique is called as geographic routing. It requires nodes to know about their position through GPS or any other localization techniques. Two broad views on routing types based on destination identification are:

Node-Centric Network Routing: Certain node(s) is specifically addressed by the source node and the packet(s) should be delivered to this node(s).

Data-Centric Network Routing: Certain node or a set of target node is not specifically addressed rather implicitly described by providing certain attributes. The node(s) which fulfill the mentioned criteria of query is liable to act as required.

To draw a clear distinguishable line between these two routing types is a doubtable decision. Aforementioned geographical routing is an example of data-centric network routing.

There exists a saturated container of routing algorithms but it still seems to “ask for more” to satisfy the requirements of a lot of hungry applications. The simplest one is to flood the network by sending an incoming packet to all the neighbors. As long as there does not exist deep network partitioning, the packet is sure to reach the destination. The refined flooding technique is to sending only the newly received packet which is called as controlled flooding. A close counterpart of flooding technique is Gossip routing which states to forward packet to an arbitrary node. The packet randomly traverses the network in the hope of ultimately finding the destination. This technique is also termed as random walk routing. There is a subtle difference between broadcasting and flooding. The former intentionally distribute the packet to all nodes which in later case, every node forward every newly received message only due to the lack of any better option available. Hence flooding is one implementation option for broadcasting [41].

Elizabeth M. Royer et al. in [40] classifies the routing protocols into

Proactive Routing Protocol: It ensures maintaining the updated routing information and instantly provides the same on need. It is also called as Table-Driven Routing Protocols.

Reactive Routing Protocols: It does not maintain updated routing table all the time but the routes are computed on demand. It is also called as On-Demand Routing Protocols.

Table 6: Hierarchical vs. Flat topologies routing

Hierarchical routing	**Flat routing**
• Reservation-based scheduling	• Contention-based scheduling
• Collisions avoided	• Collision overhead present
• Reduced duty cycle due to periodic sleeping	• Variable duty cycle by controlling sleep time of nodes
• Data aggregation by clusterhead	• Node on multihop path aggregates incoming data from neighbors
• Simple but non-optimal routing	• Routing can be made optimal but with an added complexity.
• Requires global and local synchronization	• Links formed on the fly without synchronization
• Overhead of cluster formation throughout the network	• Routes formed only in regions that have data for transmission
• Lower latency as multiple hops network formed by clusterheads always available	• Latency in waking up intermediate nodes and setting up the multipath
• Energy dissipation is uniform	• Energy dissipation depends on traffic patterns
• Energy dissipation cannot be controlled	• Energy dissipation adapts to traffic pattern
• Fair channel allocation	• Fairness not guaranteed

Proactive routing is preferable in static network topology while in the scenario of frequently changing network topology; reactive routing is a suitable option. But this classification is of the time when the Wireless Sensor Network Technology was at its infant stage and was not self-sufficient in the field of protocols. But now there exists a saturated container of routing algorithms but it still seems to "ask for more" to satisfy the requirements of a lot of hungry applications.

Later on, most popular survey report on routing techniques in wireless sensor network is [39]. Jamal N. Karaki et al. in [39] states that the routing techniques are classified into three categories based on the underlying network structure: flat, hierarchical, and location-based routing. Furthermore, these protocols can also be classified into multipath-based, query-based, negotiation-based, QoS-based, and coherent-based depending on the protocol operation.

Jamal et al. assumes that in flat routing, due to the large number of nodes, it is not feasible to assign a global identifier to each node. This consideration has led to data centric routing. But each node may or may not be assigned a global identifier based on the easiness, protocol design and application requirements. So, on such basis, we further categorized flat routing into attribute-based routing and location based routing. Attribute-based routing has lead to the data-centric routing while location-based routing has led to the node-centric routing.

Multipath Routing Protocols: Multiple paths are setup between the source and destination pair in order to enhance the network performance and to assist the resiliency and fault tolerance capability of the network but at the cost of increased energy consumption and traffic generation. Examples include [42][43][44][45].

Query-Based Routing: Destination or sink node propagates its interest in the form of a query packet through the network. The node (now source) which fulfils the parameters of the query act as required and sends the reply of the interest to the sink node. Examples include [15] [46] [47].

Negotiation-Based Routing:

QoS-Based Routing: Protocol must give much weight to the tradeoff among QoS parameters when delivering data to the base station. QoS metrics for WSN are energy consumption, data quality, delay, bandwidth, etc. Examples include [48] 49].

Table 7: WSN Routing Protocols: Classification & Comparison

	Classification	Mobility	Position Awareness	Power Usage	Negotiation Based	Data Aggregation	Localization	QoS	Scalability	Multipath	Query Based
SPERT	QoS	Very Limited	No	Limited			Yes	Yes	Good	No	Yes
SPEED	QoS	No	No	N/A	No	No	No	Yes	Limited	No	Yes
SPIN	Flat			Limited	Yes	Yes	No	No	Limited	Yes	Yes
Directed Diffusion	Flat	Possible	No	Limited	Yes	Yes	Yes	No	Limited	Yes	Yes
Rumor Routing	Flat	Very Limited	No	N/A	No	Yes	No	No	Good	No	Yes
GBR	Flat	Limited	No	N/A	No	Yes	No	No	Limited	No	Yes
MICFA	Flat	No	No	N/A	No	No	No	No	Good	No	No
CADR	Flat	No	No	Limited	No	Yes	No	No	Limited	No	No
COUGAR	Flat	No	No	Limited	No	Yes	No	No	Limited	No	Yes
ACQUIRE	Flat	Limited	No	N/A	No	Yes	No	No	Limited	No	Yes
EAR	Flat	Limited	No	N/A	No	No		No	Limited	No	Yes
LEACH	Hierarchical	Fixed BS	No	Maximum	No	Yes	Yes	No	Good	No	No
TEEN & APTEEM	Hierarchical	Fixed BS	No	Maximum	No	Yes	Yes	No	Good	No	No
PEGASIS	Hierarchical	Fixed BS	No	Maximum	No	No	Yes	No	Good	No	No
MECN & SMECN	Hierarchical	No	No	Maximum	No	No	No	No	Low	No	No
VGA	Hierarchical	No	No	N/A	Yes	Yes	Yes	No	Good	Yes	No
GAF	Location	Limited	No	Limited	No	No	No	No	Good	No	No
GEAR	Location	Limited	No	Limited	No	No	No	No	Limited	No	No
SPAN	Location	Limited	No	N/A	No	No	No	No	Limited	No	No
MFR, GEDIR	Location	No	No	N/A	No	No	No	No	Limited	No	No
SAR	QoS	No	No	N/A	Yes	Yes	No	Yes	Limited	No	No

LITERATURE SURVEY

Chapter 3

Literature Survey

31. Background and Motivation

The various famous existing protocols for ad-hoc network are not suitable for power, memory and bandwidth constraint environment of Wireless Sensor Network. More-over, to achieve the real-time packet delivery, various aspects of communication must be real-time sensitive like route discovery process if no route is available for a new destination, per-hop delay guarantee and congestion management. Ad-hoc network protocols like AODV [34], DSR [35], TORA [36], LAR [37] and others have not specifically designed to provide real-time guarantee for sensor network [50]. The history of plantation of ideas and protocols purely in wireless sensor network field is hardly more than a decade. A typical stateless energy-aware real-time routing algorithm (TSER) should encompasses the features like stateless architecture, localized behavior, efficient route discovery, energy awareness, loop free routing, traffic management and congestion management. SPEED [50] by Tian He provides the foundation for the edifice of real-time routing protocol of WSN in may 2003. Since then, many real-time routing protocols presented in [47] [51] [52] [53] [54] are accepted in the publications. In between times of almost six years, no prolific and comprehensive report in this context is displayed on the scene. Some endeavors for the same are enlightened in [55] [56] [57] [58].

GRSP [38] is obviously designed by keeping in view the requirements of purely ad hoc network in contrast to the distinguished qualities of wireless sensor network where dynamic topology, constraint resources, nodes density and higher node failure probability are core issues. In GPSR, Brad et al.'s presents an idea of reaching the destination in a greedy fashion where perimeter mode forwarding assists when void offers obstruction to access the destination. Forwarding node's immediate position is the only required value in addition to the destination position in greedy forwarding that is achieved by exchanging the beacons between neighboring nodes with a jitter by 50% of the interval between bacons to avoid synchronization of neighbor's beacons. Too much beacons, specifically in a highly dynamic environment, results in more bandwidth consumption which may leads to congestion. This situation is much drastic in a real-time environment where end-to-end delay matters much which is suffered by the time consumed in the settlement of congestion. An alternate approach adopted by Brad in [38] is piggybacking the local sending node's position on all data packets it forwards at a cost of additional twelve bytes per packets. More data sending require more power consumption which is not compensable in a WSN life, energy constraint environment. More-over, if the network is not too much dynamic then this process of piggybacking the local sending node's position is unnecessary and waste of precious resources. In a greedy forwarding fashion, a node attempts to destined the packet for a neighbor which is closest to the packet's destination but on the contrary, when no neighbor is closer i.e. void occurs, he node marks the packet into perimeter mode and the packet remain in the perimeter mode until a greedy forwarding scenario is met. A Relative Neighborhood Graph (RNG) [59] and Gabriel Graph (GG) [60] are two planer graphs: A graph in which no two edges cross, which are the target points to achieve to escape from the perimeter mode. An important property that must be considered is "Removing edges from the graph to reduce it to RNG or GG must not disconnect the graph; this would amount to partition the network".

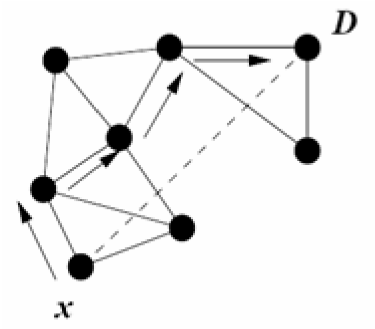


Figure 28: Perimeter Forwarding Example. D is the destination; x is the node where the packet enters perimeter mode; forwarding hops are solid arrows; the line xD is dashed [38].

For performance comparison of GPSR, they choose Dynamic Source Routing, DSR [35] as the represented of the related prior work. Simulations results are evaluated in mobile network by ns-2 [61] using the wireless extension developed at Carnegie Mellon and found GPSR, a better approach in packet delivery success ratio, routing protocol over-head, optimality of path length taken by data packets and robust packet delivery in densely deployed wireless network where by caching only the state proportional to the number of its neighbors. More over, GPSR permits backtracking and can achieve 100% delivery rate as long as the network is not partitioned [50].

SPEED [50], a stateless adoptive geographically greedy based routing protocol, is a member among the prior real-time geographical location based protocols. It ensures end-to-end soft real-time communication by per-hop delay guarantee and greedy forwarding with min. hop count from source to target. Tian He achieves the required goal of real-time communication in a memory, energy and processing constraint environment of small nodes called sensors by promising some additional design objectives. They ensure minimizing the memory requirement by stateless architecture, congestion management by backpressure re-routing scheme with min. control over-head, traffic load balancing by non-deterministic forwarding techniques, and localized behavior by processing almost all the distributed operations locally and loop-free routing by greedy geographic forwarding. The key factors that kill more of the objectives of real-time routing are node-failure, congestion, and flooding and re-route discovery process. SPEED uses "Neighbor beacon exchange" to detect node failure by refreshing rate of neighbor node entry, to obtain location information by "*position*" field in beacon packet and node receive delay by calculating round trip time minus node processing delay.

More beacons consume more bandwidth resulting in congestion as well as energy consumption resulting in early node failure. So in addition to the beacon packet, piggybacking on data packet at some specific interval can share the load of information exchange among the nodes in the network. Such as *SPEED estimate the load of nodes by calculating the "single hop delay" metric by piggybacking the information on the packet data.* In a sensor network, for a protocol claiming to be a real-time, congestion management is a million dollar question to be answered because it dreadfully increases the end-to-end delay and in severe cases, creating a dead-lock, keeping the sink in an expectant situation for the message. SPEED removes this bone of contention by back-pressure rerouting in assistance with Stateless Non-deterministic Geographical Forwarding (SNGF) and Neighborhood Feedback Loop (NFL). The neighborhood set of node *i* are determined as $\mathrm{NS}i(\mathrm{k}) = \{\mathrm{node/k} \leq \mathrm{distance(node,\ node}\ i) \leq \mathrm{R}\}$ and the forwarding candidate set of node *i* are determined as

$$\mathrm{FS}i(\mathrm{Destination}) = \{\mathrm{node} \in \mathrm{NS}i\ (\mathrm{k})\ /\ \mathrm{L\text{-}L\_next} \geq \mathrm{k}\}.$$

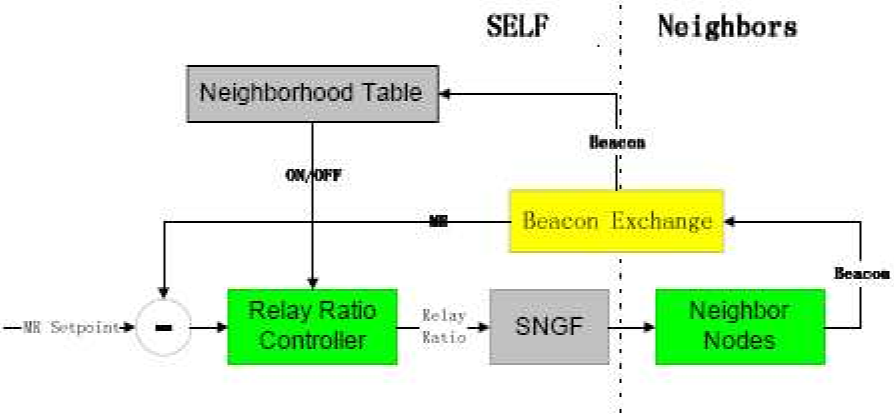


Figure 29: Neighborhood Feedback Loop (NFL)

Packet is forwarded only to that node that is the member of forwarding candidate set and ensures the *SendToDela*y less than a certain single hop delay D (SendToDelay is the delay value received from beacon message coming from neighbor and used to make routing decisions by SNGF) and the *SendToDelay* for a path is calculated by NFL. NFL is established through the neighbor beacon exchange. On the other hand, void problem in routing is also somewhat pointed out by Tian He. SPEED handles rerouting around a void in the same way as it handles congestion [62]. So when a node has an empty forwarding candidate set of node *i*, it then sends a backpressure beacon with its ID, the destination and *Avg SendToDelay* = $\propto$. The receiving node sets that node's *SendToDelay* to $\propto$ and stops sending packets to that node.

SPEED was simulated on GloMoSim [63], which is a scalable discrete-event simulator developed by UCLA and its performance is evaluated by comparing it with other five protocols namely AODV [34], DSR [35], GF [64], SPEED-S, SPEED-T. Taker et al. repeated the experiments 16 time with different random nodes topologies and found SPEED a better approach in the end-to-end delay under different congestion levels, miss ratio, control overhead, communication energy consumption and packet delivery ratio as compared to its competitors in a its simulation environment.

Real-time speed of SPEED [50] reaches a stuck node and faces a void that is handled by dropping the packets and a backpressure beacon is sent by the stuck node to notify the source node to stop sending packets to it. Lei Zhao et al. in [65] pointed out that the number of packets dropped by the SPEED void avoidance mechanism is very high due to the following packets may be sent to the void point node again by other routing paths and proposed a Fault-Tolerant Real-Time Routing Protocol, FT-SPEED. All the components and ideas in FT-SPEED are the same to handle the real-time packet delivery in greedy forwarding except the void avoidance mechanism.

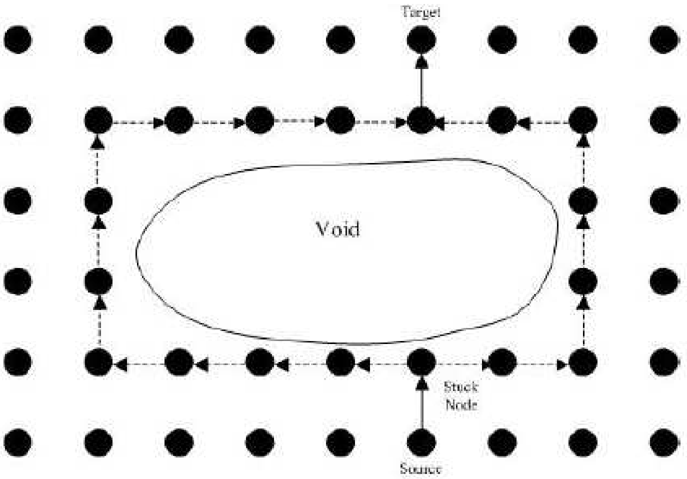


Figure 30: Example of Void Bypass [65]

Two ideas are introduced to cope with the above mentioned problem. First one is the "void announce scheme" to avoid the packets being sent to the void area and the second one is to face the issue if the packet reaches to the void area in any way namely "void bypass scheme" having the idea of routing the packets around both sides of the void, not just drop it. The node whose forwarding neighbors are stuck nodes are termed as void area nodes while the nodes n the boundary of the void are referred to as void edge nodes in the paper. If the node identifies itself a stuck node as the FS is null. It broadcast a backpressure beacon with setting the field "*AvgSendToDelay*" equal to $\propto$ which stops the neighbors to deliver packets to it and let the neighboring nodes to choose other forwarding nodes as its next hop. If there is also a red signal then it will declare itself a void area node and broadcast backpressure beacon to its neighboring nodes stopping them to send packets to these void area nodes. Same strategy works through all the way down until green signal shows a clear way for greedy packet forwarding to go ahead to destination. But if the packet reaches void area node that is the void edge nodes, then the void bypass scheme is followed to get rid of this dead end. In case the void area node is not the void edge node, the packet is sent to the forwarding void node whose *ActualDelay* is shortest until it reaches a void edge node. To guarantee the real-time requirements, a novel two sides bypassing void methods is introduced in [65]. BOUNDHOLE [66], a void finding algorithm, technique is used to send the packet to the following void edge node in the counterclockwise order and when the packet reaches in a nearer node, it can be sent with greedy forwarding mechanism again.The authors analyzed their proposed protocol using J-SIM network simulator in a environment that is mainly drawn from [50] [51] for fair comparison. FT-SPEED is compared with SPEED [50] and MMSPEED [51] for packet success delivery, average delay and control packets as performance perimeters.

Noor et al.'s "**A Real-time Energy-aware Routing Strategy**"[52] is based on LNA [67] (Logical Network Abridgement): *A procedure capable of describing the intrinsic state of health of the over all network*, to find the best tradeoff between energy consumption and message latency with strengthening the resiliency and performance of the network. They use $\cos tI$ function for time awareness and $\cos tII$ function for energy awareness. Source receives these two cost function values of each path that may lead to destination from each of its neighbor. Packet is forwarded only to that which has low value of I$_p$ (path impact factor) as well as fulfilling the condition of $(T_L < T_p)$ i.e. propagation time of node *i* is less then the time left. T$_{P,I}$ is calculated as $T_{p,i} = \dfrac{\cos tI_{ij}}{c} + n\tau$ and $\cos tI_{ij}$ is calculate s $\cos tI_{ij} = \sum_{m=1}^{k-1} D_{ij}\{m\}$ and if $\forall_i \in \{1,...,n\},(T_L < T_p)$, packet is dropped. If more than one node satisfies the $(T_L < T_p)$ condition then the node having minimum $\cos tII$ function will be given the highest priority. On the other hand, taking into consideration the energy-awareness: $\cos tII$ function, I$_P$: a global cost function is calculated by the formula $I_P = \sum_{i=1}^{K} I^2{}_{N,i}$ where $I_{N,i}$ is calculated as

$$I_{N,i} = \frac{I_{V,i}}{I_{E,i}} \text{ and } I_{V,i} = \frac{N^{\{i\}}_{before}}{N^{\{i\}}_{after}} \times \frac{L^{\{i\}}_{before} + 1}{L^{\{i\}}_{after} + 1} \quad , \quad I_{E,i} = \frac{B_i}{B_i^o}$$

In the notational form, first of all current battery level B_i of i$_{\text{th}}$ node is divided by starting battery level B_i^o resulting in energy impact factor $I_{E,i}$, in this way the energy level of each node is calculated.

In the parallel way, how much the removal of the nodes (due to failure or power depletion) in relation to its vulnerability, impacts the depth of resiliency of the network is calculated. Division of vulnerability index $I_{V,i}$ of the i*th* node by the energy impact factor $I_{E,i}$ of the same i*th* node ends up with node impact factor I_N (Local cost function). While performing the same calculation to all nodes in the path, we get the value of path impact factor, I_P (global cost function) and hence the route having low I_P value produces the best energy efficient routing.

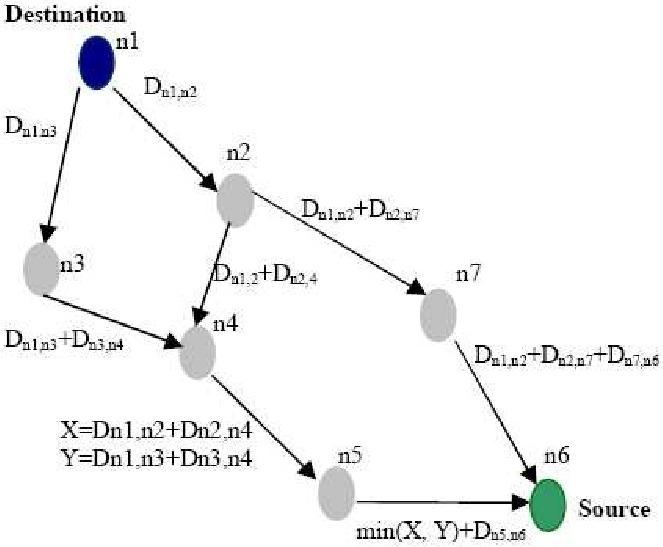


Figure 31: Calculating CostI Function [52]

The authors work on providing "reliable" and "real-time" packet delivery services differentiating packets depending on their urgency and importance. The proposed protocol follows geographical routing. Although energy efficiency in the proposed algorithm is tried to achieve yet it requires special concentration in lessening the overhead. Also transmitting of total path information pertinent to energy and path not only bandwidth hungry but also consumes much energy. Moreover, network scalability and void-node management aspects are untouched.

The authors evaluated their proposed idea in a real practical scenario with 50 nodes and concluded after extensive simulations that for real-time packet delivery in an energy constraint environment like WSN, two factors T_p and $I_p$value have much importance. Moreover, packet delivery percentage is comparatively high even in aggressive deadline.

O. Chipara et al. cater for miss ratio and energy efficiency as the two performance perimeters in Real-time Power-Aware Routing Protocol for Wireless Sensor Network [68]. The basic principles of RPAR [68] are workload and packet deadline based transmission power adaptation and routing decisions, lossy links [69] and memory, energy, bandwidth constraints based strategy and localization behavior with the assumptions that node is stationary and knows its position via GPS or other location service [70], radios can adjusts their transmission power and MAC protocol does not uses RTS/CTS. To achieve the objectives, RPAR comprises of four components: Dynamic Velocity assignment policy: to map a packet's deadline to a required velocity, Forwarding Policy: to forward the packet to the most energy-efficient forwarding choice that meets the packet's required velocity, Delay Estimator: to evaluate the one-hop delay of each forwarding choice in the neighbor table, and a Neighborhood Manager: to find a new forwarding choice through power adaptation and neighbor discovery process. Key processes performed by each of the above mentioned four components are as follows.

In Dynamic Velocity Assignment Policy, the required velocity $V_{req}(S,D)$, of each the packet is calculated by the equation:

$$V_{req}(S,D) = \frac{d(S,D)}{slack_{rec}(S) - (t_{head}(S) - t_{rec}(S))}$$

On the parallel, in the forwarding policy component, the provided velocity of each of each forwarding choice is calculated as:

$$V_{prov}(S,D(N,p)) = \frac{d(S,D) - d(N,D)}{delay(S(N,p))}$$

A forwarding choice (N,p) is an eligible forwarding choice if the $V_{prov}(S,D(N,p))$ of a path is higher then the $V_{req}(S,D)$ of the packet.

The energy cost of all eligible forwarding choices is estimated as:

$$E(S,D(N,p)) = E(p).R(S,(N,p)).\frac{d(S,D)}{d(S,D) - d(N,D)}$$

The third component, Delay Estimator, of RPAR estimate the delay of a packet sent by S to neighbor N using transmission power p by the formula:

$$delay(S,(N,p)) = (delay_{cont}(S) + delay_{tran}).R(S,(N,p))$$

Single contention estimation per node and transmission count estimation per forwarding choice are the two functions of Delay Estimator. This link estimation scheme is more supportive to real-time communication in dynamic environment as compared to existing link estimators that are designed to estimate the average link quality [50][71].

A significant issue in real-time communication is re-routing path discovery that should be efficient otherwise all the active performance of protocol may lead to death. Because in WSN, due to the probabilistic nature of wireless link [71], a node has to choose the best link among a potentially large number of neighbors which may have the links that bears poor link quality and long delays. Tarek et al. introduces Neighborhood Manager having power adaptation and neighbor discovery mechanism that dynamically discovers eligible forwarding choices and are triggered on-demand and manages the table.

In power adaptation scheme, RPAR adjust the transmission power to a neighbor already in the neighbor table on the basis of required velocity of the packet using a multiplicative increase by a factor of $\alpha(\alpha > 1)$ and linear decrease by $\beta(\beta > 0)$ scheme. If increasing the transmission power to a suitable extend does not achieve the required velocity then RPAR invokes neighbor discovery to find new neighbors.

To discover the proper neighbor, node S broadcasts an RTR packet at some power maximum enough to be heard by far away nodes that may provide high delivery velocities. But only the nodes that satisfy the following inequality should reply in order to avoid from severe network contention and hence chance of packet collision.

$$V_{req}(S,D) \le v_{\max}(S,D,N) = \frac{d(S,D) - d(N,D)}{delay_{cont}(S) + delay_{tran}}$$

In the above equation the maximum distance between any eligible neighbor N and destination D can be derived as follows:

$$d_{\max}(N,D) = d(S,D) - V_{req}(S,D).(delay_{cont}(S) + delay_{tran})$$

S piggybacks dmax(N,D) in the RTR, and a neighbor N that hears the RTR replies only if $d(N.D) \le d_{\max}(N,D)$.

RPAR uses the FREQUENCY algorithm [72] to manage the neighbor table. The FREQUENCY algorithm associates a frequency counter with each forwarding choice and only the frequently used forwarding choices remain in the table.
The Authors implemented RPAR in a Matlab-based network simulator called Prowler [73] to create a realistic simulation environment that is configured based on the characteristics of MICA2 motes [3]. They focus on the “many-to-one” traffic pattern with the deployment strategy as 130 nodes in a 150 m x 150 m region divided into 11.5 m x 15 m grid. They evaluated their protocol, RPAR, with the performance of “Forwarding Policy”, “Neighborhood Management” and under different workloads. Moreover, the simulations showed that RPAR significantly reduces the deadline miss ratio and energy consumption compared with existing real-time and energy-efficient routing protocols.

Linfeng Yuan [54] tried to achieve real-time energy efficient protocol using Effective Transmission Model (ET) and Constrained Equivalent Delay Model (CED). ET model serves the job of “forwarding candidate selection”. This selection process needs two steps. The First step is to determine the Forwarding Candidate Set (FCS) on the basis of Candidate Selection Function (CSF) which is an adoptive selection adjustment process.

$$d(Forwarding\_Candidate, Destination) > d(Source, Destination) \& d(Source, Farwarding\_Candidate) > (Source, Sender)$$

If the above mentioned criterion is not fulfilled by any neighboring node then a slightly looser following criterion is set for forwarding candidate selection.

$$d(Forwarding\_Candidate, Destination) > d(Source, Destination)$$

Thus the nodes fulfilling the above mentioned criterion are placed in the FCS. The second step of the selection is to determine the optimal candidate within the FCS. Decisive Energy Factor (DEF) figure out the total energy consumption from sender to forwarding node and then to destination node. So for a specific data size, the energy consumption is decided by the DEF. The sender node compares each node in the FCS by computing each DEF and hence the node with the least DEF and the best favorable value in the CSF is designated as the optimal candidate. CED model uses the mathematical concept of projection. The ratio of the CED of a link to the end-to-end delay requirement is equal to the ratio of the link’s projection length to the whole length on the line from source to sink. Hence the candidate node for which the ratio of the actual link delay to the link’s CED is smaller is preferred in selection. So the resultant of the above two model provide the selection of path with the lower energy consumption and lower end-to-end delay.
This protocol has priority in better energy awareness management over many other position based routing protocol [50] [74] [75]. In SPEED [50] and GRS [74], the sender selects its neighbor which is the closest one to its destination. MFR [75] necessitate that the packet be forwarded to the neighbor whose progress is the maximum. COMPASS [76] selects the neighbor that has its direction to the sender is the closest to the direction from the sender to the destination.
On further pondering, it is explored that the authors only consider the energy consumption and did not care for the remaining energy of the node. If the node with least DEF is already run out of energy then in addition to other faults there is also abrupt increase of packet loss. Implying this check prior to the DEF will improve the performance of protocol.

In “Energy Efficient Real-Time Routing in Wireless Sensor Network” [77], the idea of real-time data gathering and energy efficiency is borrowed from Linear Algorithm Network (LNA) [67]. They improved LNA called Energy-Efficient Linear Algorithm Network (ELNA) to make energy use more efficient than LNA. To maximize the number of messages that can reach the BS where each message has its own due-date, they designed the ERTR algorithm, which on simulation resulting in reducing the deadline miss number compare with LNA.

[78] is a good effort. IGF [78], a unicast, stateless, location based routing protocol for highly dynamic sensor network. To prevent the adverse effect that dynamic factors such as highly mobility have on the state-based eager-binding routing protocols, He et. al. exploit the concept of lazy binding to cope with high dynamics in highly dynamic sensor network. Exploiting the exquisites of location address and lazy-binding semantics, IGF becomes a pure state-free protocol independent of knowledge of network topology or the presence/absence of other nodes resulting in support to fault tolerance and make protocol robust to real-time topology shift or node state transition. A state-free routing solutions presented in SPEED [50] and IGF [78] not only an obstacle in bandwidth consuming packets required in the state-based solution for routing and neighbor table upkeep but also conserve the precious and constraint resource of energy. To achieve the real-time communication factor over the highly dynamic sensor network, stateless nature of protocol salt away from communication delay resulted by excessive delay in neighbor and routing table updating an d eliminating the binding delay by lazy-binding process.

Emad et al. [51] present a novel packet delivery mechanism called Multi-Path and Multi-SPEED Routing Protocol (MMSPEED) for probabilistic QoS guarantee in wireless sensor networks. The QoS provisioning is performed in timeliness domain and reliability domain. Multiple QoS levels are acquired in the timeliness domain by promising multiple packet delivery speed options. While probabilistic multi-path forwarding mechanism assist in achieving the reliability requirements in the reliability domain. The authors claim to guarantee end-to-end requirements in a localized way. In-network and non-coherent data processing routing augment scalability and adaptability factors to large scale dynamic sensor networks. But Emad et al. also did not consider the energy and memory constraint factors of WSN so as to be a energy and memory greedy algorithm.

In [79], Adil et al. applied geo-directional-cast forwarding based on quadrant for real-time routing with load distribution (RTLD). RTLD comprises of geographical location management, power management, neighborhood management and routing management components. Packet reception rate, remaining power of sensor nodes and maximum velocity of the packet moving through one-hop are functional arguments of optimal forwarding choice. A specific sensor node in the network calculates its location based on three predetermined neighbor nodes and its distance to those neighbors in the geographical location management stage. Power management determines the remaining power and the power level of transmission in the node. Neighborhood table of the forwarding candidate nodes is upkeep by the neighborhood management process and on the basis of this, the selection of optimal forwarding node is carried out in the routing management process.

3.2. Limitation of Existing Relevant Literature

Memory management, energy awareness and real-time constraint features are not perfectly satisfied by any protocol which may earn the idiosyncratic alias of three-in-one. SPEED [50] and RAP [53] consider stringent memory and real-time constraints. Linfeng et al. [54] and Noor et al.[52] heed the energy awareness and real-time routing. Greedy Perimeter Stateless Routing [38] pays more attention to memory awareness and in some aspect to temporal constraint. Multi-Path Multi-SPEED [51] and RPAR [68] step ahead to cover aforementioned three features but improvement is needed to meet safety critical real-time constraint. Achieving the target in the limited required time in accordance with real-time boundaries with minimum energy and saving the battery life is a crucial dilemma in the multi-factor-constraint environment of WSN [15]. Hence the problem identified is:

- Considering the temporal constraint with less computational and processing, minimum E2E delay and packet overhead

- Considering the network lifetime in non-replenished battery environment with min. beacon message exchange, effective load balancing, less packet overhead and lower energy consumption

3.3. Objectives

Real-time communication with optimal energy utilization in WSN is really a fascinating area with great potential of research. Comprehensive literature survey reveals that a protocol satisfying energy, memory and real-time constraint issues is still a need of real-time applications in wireless sensor networks. Since the wireless sensor nodes are in-general battery-operated and are placed inattentive in many applications. Therefore, energy utilization by each node is one of major constraint factor in the design of routing protocol. Hence, Incorporation of real-time capability with energy awareness characteristics is under consideration of our proposed algorithm

PROPOSED SOLUTION

Chapter 4

Proposed Solution

4.1. Tradeoff Management

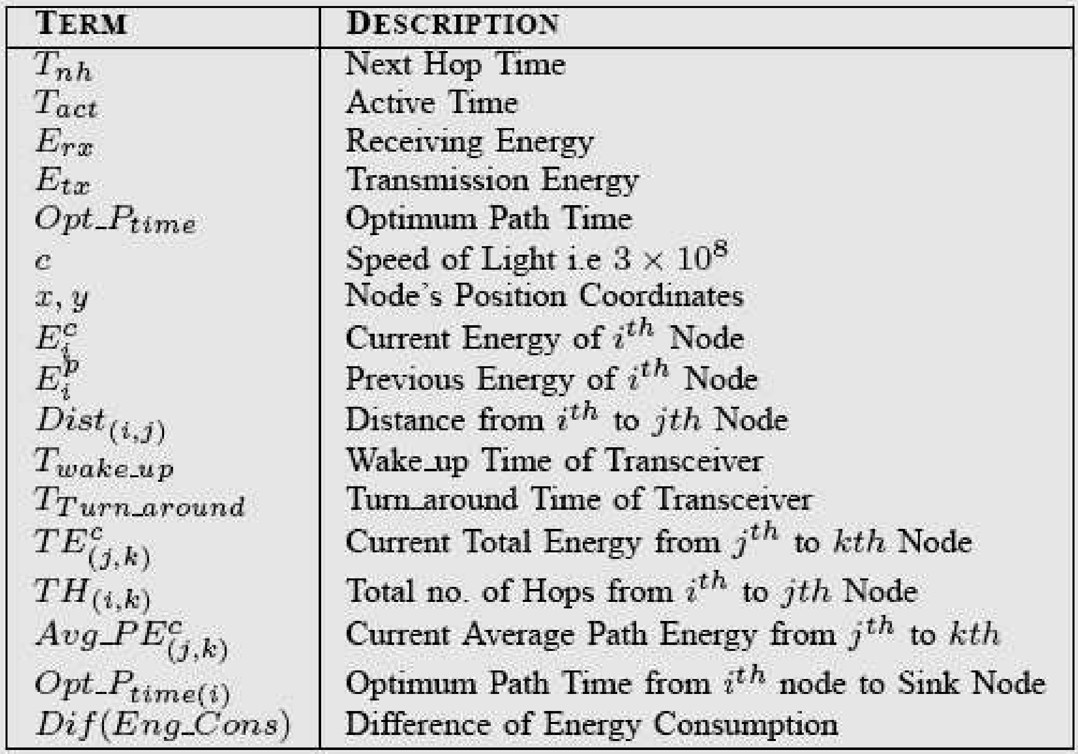
Setting up tradeoff among competing factors especially where each one has close attachment to each other is a million dollar question. This scenario turns into a crucial dilemma when the competing factors are in stringent constraint. Energy, memory, processing and bandwidth constraint factors of wireless sensor network present like wise challenge to the researchers. All these constraint factors must be in the front panel when thinking of contribution in the software architecture of wireless sensor network. Considering the delay sensitiveness in its various aspects improve the effectiveness and efficiency of the work performance especially in the application where temporal constraint is of indispensable factor. Real-Time routing in WSN is a non-separable part of this delay sensitiveness. Ubiquitous opportunities of these miniature devices: sensor node, also unveil the dilemmas in energy, memory and computational prongs, specifically in the networked communication environment of wireless sensors. There also comes up some differences from the traditional wireless network like cellular network and especially from its close ancestor, ANet (Ad hoc Network) like unattended operation for a long period of time in the energy constraint environment, densely and randomly deployed nodes, high node failure and other constraint resources. Most of the algorithms designed for wireless networks and Adhoc networks need to be improved due to the aforementioned grounds.

We have proposed customized proactive routing, in-network processing, Localized behavior of nodes and customized flooding techniques to optimistically target the temporal constraint and network life time parameters. Simulation results have intuited that REAR presents a better solution in energy consumption with the effect of improving the network life as well as increasing the performance efficiency in end-to-end delay.

4.3. Proposed Solution

A. Overview

Location identification of a node is the first step for many operations especially of routing in WSN. In the stringent energy and memory constraint environmental of WSN, performing the right action at the right time is the core issue to be solved in many real-time applications. To cope with these issues, our proposed algorithm consists of three modules: Location Manager, Time Manager, and Power Manager. In addition to these, forwarding node selection is the distinguished attribute of a routing protocol especially that claiming to be the candidate of real-time routing. So, Forwarding Manager module is added for the optimum forwarding node selection. On acquiring the node position through Node Id's for static node and through GPS for mobile nodes, Location Manager performs location-aware calculations and forwards the necessary results to Time Manager (TM) and Power Manager (PM) (discussed in part B of this section). Both TM and PM then perform time-aware calculations and power-aware calculations respectively and hand over the optimum paths time and optimum paths average energies data to the forwarding manager (discussed in part C and D of this section). On the basis of different selection criteria, FM ultimately forward the packet to the optimum forwarding node (discussed in part E of this section). Figure 2 illustrates the above discussion. All the aforementioned four modules are discussed in subsequent paragraphs.



TERM	DESCRIPTION
T_{nh}	Next Hop Time
T_{act}	Active Time
E_{rx}	Receiving Energy
E_{tx}	Transmission Energy
Opt_P_{time}	Optimum Path Time
c	Speed of Light i.e 3×10^8
x, y	Node's Position Coordinates
E_i^c	Current Energy of i^{th} Node
E_i^p	Previous Energy of i^{th} Node
$Dist_{(i,j)}$	Distance from i^{th} to jth Node
T_{wake_up}	Wake_up Time of Transceiver
T_{Turn_around}	Turn_around Time of Transceiver
$TE^c_{(j,k)}$	Current Total Energy from j^{th} to kth Node
$TH_{(i,k)}$	Total no. of Hops from i^{th} to jth Node
$Avg\_PE^c_{(j,k)}$	Current Average Path Energy from j^{th} to kth
$Opt\_P_{time(i)}$	Optimum Path Time from i^{th} node to Sink Node
$Dif(Eng\_Cons)$	Difference of Energy Consumption

TABLE I

PARAMETERS DESCRIPTION

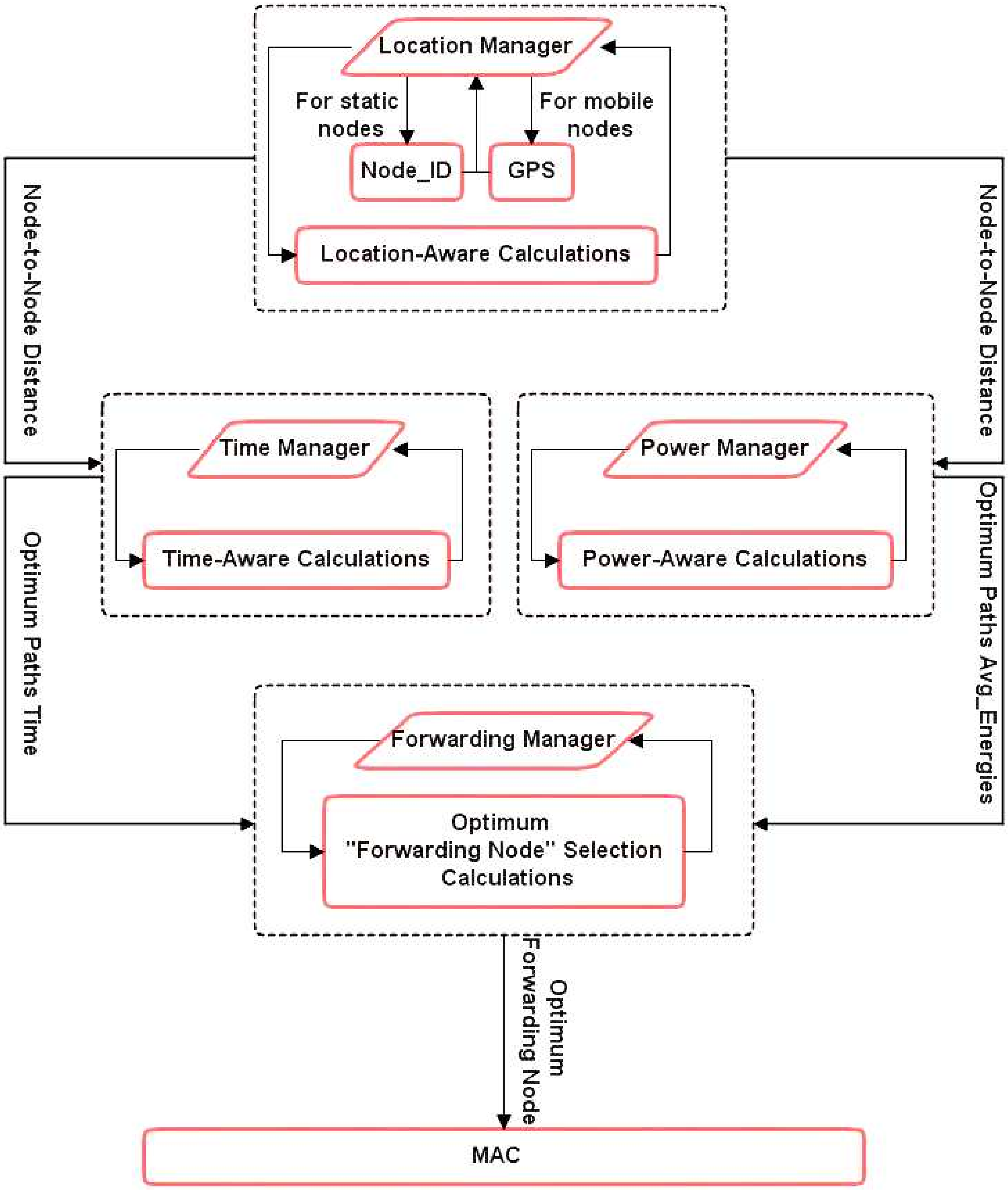


Figure 32: Operation Chart of REAR's Modules

B. Location Manager

Dense deployment fashion is the need of wireless sensor nodes to save from network partitioning and to keep the base station away from expectant situation for required data. Hence nodes are in dense deployment fashion and Node ID's are assigned in a sequence corresponds to the nodes' deployment sequence. So each node knows its x,y position coordinates. Location Manager thus has fault-tolerance feature in node localization system. If this technique of Node Id assignment is used in GPS-enabled nodes then on the failure of Global Positioning System (GPS), LM still works to communicate the node position coordinates. This geographic routing strategy has its own advantages according to its proponents such as, more scalability, no flooding and no state propagation for node discovery and only requires the information from its direct neighbors, and lower overhead [23]. In contrast to this, its opponents have their own arguments. Any how, dual node positioning capabilities of LM not only assist in fault tolerance but also cooperate the Time Manager (TM) and Forwarding Manager (FM) to successfully carrying on their tasks. Through the following formula, LM calculate Next hop distance (Dist(i,j)) whose results are used as parameter in the time calculation by TM.

$$Dist_{(i,j)} = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}$$

C. Power Manager

Power Manager (PM) manipulates its duty by monitoring the power consumption of node during reception, calculation, transmission and other related processes. PM calculates power consumption using following formula.

$$E_{consumed} = E_{rx} + E_{proces\sin g} + E_{tx}$$

Processing power consumption which typically includes decoding, encoding, demodulation, modulation and other required calculations. Current Node's residual power is calculated as:

$$E_i^c = E_i^p - E_{consumed}$$

Most of the active and effective functionality of power manager is its path energy awareness which assists in better decision of optimum path selection when query is disseminated from the BS to the intended source. Average path energy value at each node becomes updated during this query dissemination process. Relevant calculations for the aforementioned discussion are as follows.

$$Avg\_PE^c_{(j,n-1)} = \frac{(E_j^c + E^c_{(j-1)} + .. + E^c_{(j-(n-1))})}{n-1}$$

$$TE^c_{(j,k)} = Avg\_PE^c_{(j,k)} * TH_{(j,k)}$$

$$Avg\_PE^c_{(j,k)} = (TE^c_{(j,k)} + E_i^c) / TH_{(j,k)}$$

Where
j = i - 1
i = current node
k = destination area node

This calculation keep the $TE^{c}_{(i,k)}$ update for being the member for selection of 'Optimum Forwarding Node Selection' by the Forwarding Manager. PM also gives helping hand to Forwarding Manager for making fault tolerant decisions in case of node energy depletion or the death of optimum forwarding node (discussed in part E of this section).

D. Time Manager

Cooperation of REAR's functioning modules makes the life easy. In assistance with LM, Time Manager (TM) performs its time management calculations that results in better realtime target achievement. Following promise-aware calculations contribute in achieving the optimum path time. It is the good habit of nodes to go into hibernation for their energy saving which ultimately feed the network for more life. They deactivate their subsystems except sensors which may obtain the relevant and required environmental physical measurement. The deactivated subsystems of nodes then are awaken to perform to utilize and make the sensor data meaningful. Thus there are also other factors that have been added for precise End-to-End network time calculation such as active time that is as follows.

$$T_{act} = T_{set\_up} + T_{transmit}$$

And

$$T_{set-up} = T_{wake\_up} + T_{turn\_around}$$

Next Hop Time (T_{nh}) and Optimum Path Time (Opt_P_{time}) for the destination area node is calculated by using the following formula.

$$T_{nh} = \frac{Dist_{(i,j)}}{c} + T_{turn\_around} + T_{transmit}$$

$$Opt\_P_{time(k)} = T_{nh}$$

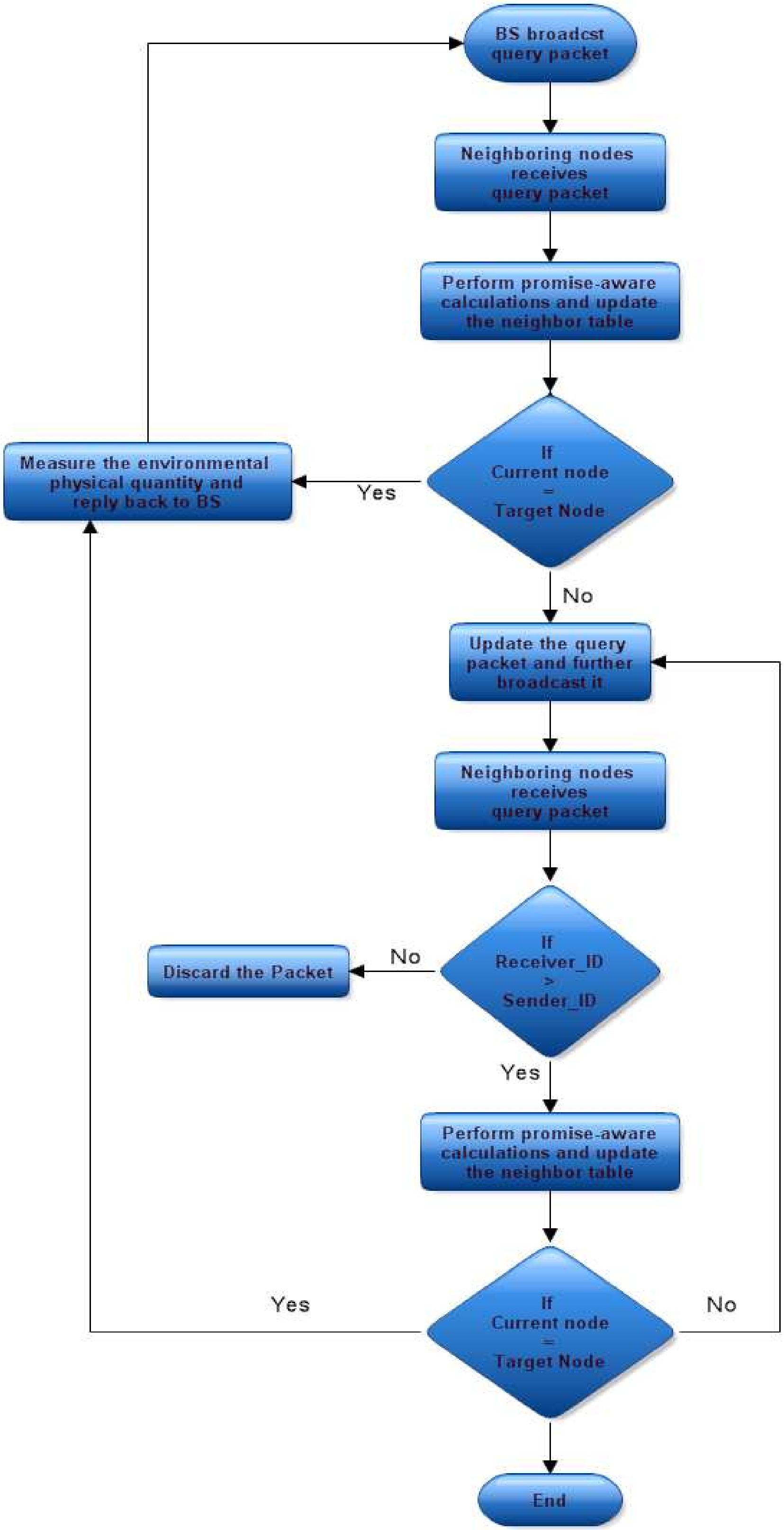

BS broadcst query packet
Neighboring nodes receives query packet
Perform promise-aware calculations and update the neighbor table
If Current node = Target Node
Yes
Measure the environmental physical quantity and reply back to BS
No
Update the query packet and further broadcast it
Neighboring nodes receives query packet
If Receiver_ID > Sender_ID
No
Discard the Packet
Yes
Perform promise-aware calculations and update the neighbor table
If Current node = Target Node
Yes
No
End

Figure 33: Query Dissemination Process Flow Chart

As there is no transit node between the destination area node and the sink node. T_{nh} and Opt_P_{time} for a specific node other then the destination area node is calculated from the following equations.

$$T_{nh} = \frac{Dist_{(i,j)}}{c} + T_{wake\_up} + T_{turn\_around} + T_{transmit}$$

Opt_P_{time} for the immediate first node outside the destination area is calculated as $Opt\_P_{time(k)} + Opt\_P_{time(k)}$.

For All other nodes Opt_P_{time} calculation is as follows.

$$Opt\_P_{time(i)} = Opt\_P_{time(j)} + \ldots + Opt\_P_{time(k)}$$

This calculation process keep the Opt_P_{time} update for taking part in 'Optimum Forwarding Node Selection'.

E. Forwarding Manager

A node farther from the source and nearer to the destination, more favorable regarding re-route discovery, and optimum in selection criteria among its neighboring nodes is most appropriate candidate from forwarding nodes selection. Forwarding Manager (FM) module promise with REAR to designate such characteristics containing node as the forwarding node for optimum path discovery. Time, Hop Count and Energy are the three selection parameters with priorities in the respective order. Although FM considers a path with $min(Opt\_P_{time})$, $min\,(Hop\_Count)$ and $Max\,(Avg\_Path\_Energy)$ as a idea forwarding node. If this case does not exist then a lower tight bound criteria of the optimum path is selected with $min(Opt\_P_{time})$ and $min((Hop\_Count)$. In case, this situation has no chance to exist then one by one performance parameters are designated as the selection criteria. Among the P_{time} values received from Time Manager, FM selects the path with the $min(Opt\_P_{time})$ as the forwarding path. If the path time to destination of all the neighboring nodes is same, then the Hop Count values received from TM are designated as selection parameter. The path with *min*(Hop Count) to destination becomes the appropriate choice for forwarding the packet to destination. (Avg_Path_Energy) values received from Power Manager are then considered, if the no. of Hop Count to destination of all the selected paths are same. $\mathrm{Max}\,(Path\_Energy)$ value then act as the decision parameter for the selection of optimum forwarding node selection. In the nutshell, $\mathrm{min}(Opt\_P_{time})$, $min((Hop\_Count))$ and $\mathrm{Max}((Avg\_Path\_Energy))$ are the selection parameters in a conditional order for the selection of optimum forwarding node selection.

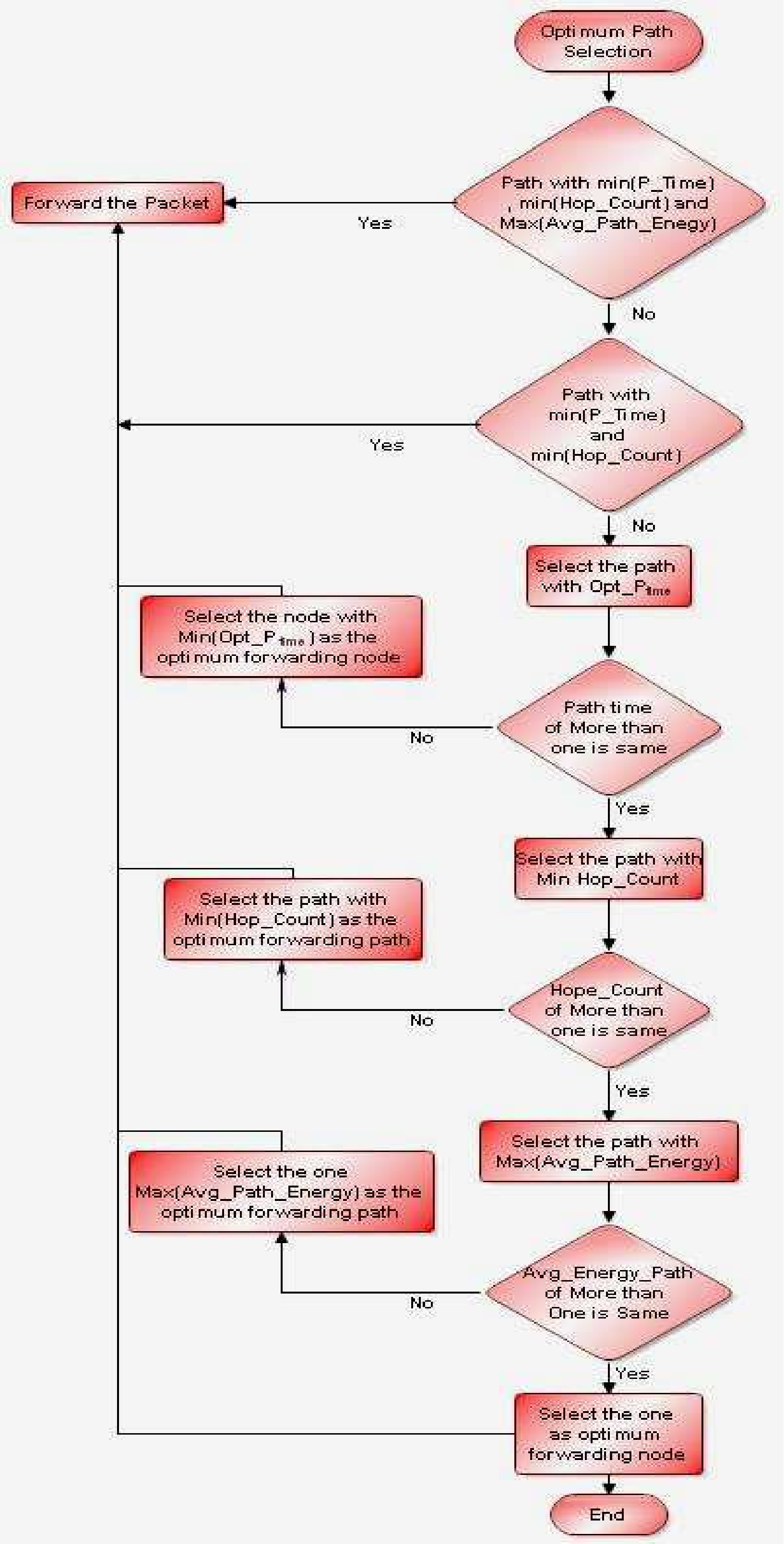

Optimum Path Selection
Path with min(P_Time) , min(Hop_Count) and Max(Avg_Path_Enegy)
Yes
Forward the Packet
No
Path with min(P_Time) and min(Hop_Count)
Yes
No
Select the path with Opt_P_time
Select the node with Min(Opt_P_time) as the optimum forwarding node
Path time of More than one is same
No
Yes
Select the path with Min Hop_Count
Select the path with Min(Hop_Count) as the optimum forwarding path
Hope_Count of More than one is same
No
Yes
Select the path with Max(Avg_Path_Energy)
Select the one Max(Avg_Path_Energy) as the optimum forwarding path
Avg_Energy_Path of More than One is Same
No
Yes
Select the one as optimum forwarding node
End

Figure 34: Forwarding Path Selection

Simulation Detail

Chapter 5

Simulation Detail

5.1. Simulation

A simulator is "a device, instrument, or piece of equipment designed to reproduce the essential features of something" [32]
There are many simulators available for wireless networks for example *Avrora* [82], *ATEMU* [86], *EmStar* [87], *SWANS* [88], *QualNet* [89], *OPNet* [90] and *SENSE* [91] and *TOSSIM* [84] are the available simulators in market for Wireless Sensor Network. Among these TOSSIM is widely used and is preferred due to its excellent features. Some of the key features and limitations of TOSSIM are listed as follows.

- It is a discrete event simulator for TinyOS based wireless sensor network [81]
- Operating environment is provided by TinyOS which is an operating system especially for embedded systems
- Programming language is NesC (C for Network embedded system) which is a C like language
- TOSSIM simulates the 40Kbit RFM mica networking stack, including the MAC, encoding, timing and synchronous acknowledgement [81]
- Provides fidelity based and reliable simulation environment
- Simulation time is in coincidence with CPU clock of MICA platform
- It provides different models as plug-in like

  Lossy Builder: a java-based tool to add radio model to the simulation
- Energy\Power consumption can be calculated by adding energy consumption parameter for any component
- No preemption involvement during simulation in contrast to real motes. So less authenticity in results
- Simulation results are not too much authoritative. It dictates high data loss between two algorithms but data loss scenario is not perfect like real mote

5.2. Simulation Environment of TOSSIM

In fact, TOSSIM is a Linux environment based simulator. To exploit its functionalities in Microsoft Windows based environment, an emulator named Cygwin [92] is available to provide the same Linux environment in Microsoft Windows. It consists of following two parts.

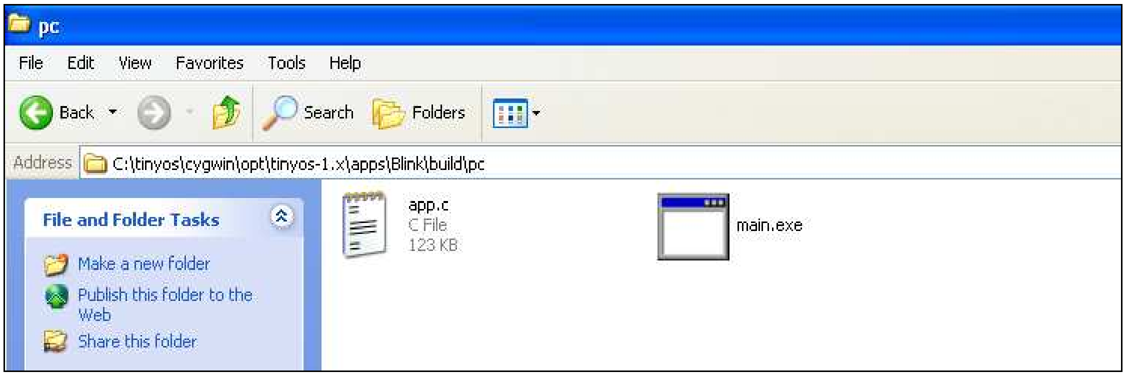
- A DLL (cygwin1.dll) which acts as a Linux API emulation layer providing substantial Linux API functionality
- A collection of tools which provide Linux look and feel

In our research work we used the later scenario. Latest Cygwin DLL release version is 1.5.25-15, which can be downloaded from [92]. Most of the limitation and drawbacks in the previous versions of TOSSIM are overcome in its current version of 2.x which is available at [84].

5.2.1. How to Compile an Application

Following is the procedure for application compilation in TOSSIM.

- Open cygwin shell,
- Go to the application directory
- Write "make" command in the shell to compile the application for simulation in TOSSIM. TinyOS 1.x uses "*make pc*" but TinyOS 2.x uses "*make sim*".
- On successful compilation, new directory named *build* with a main.exe file is created inside the application direction as shown in Figure 5-1.
- For compilation for other then simulation use*"make* platform". Valid platforms for TOSSIM 2.x are *mica, mica2, mica2dot, micaz* and *pc*.



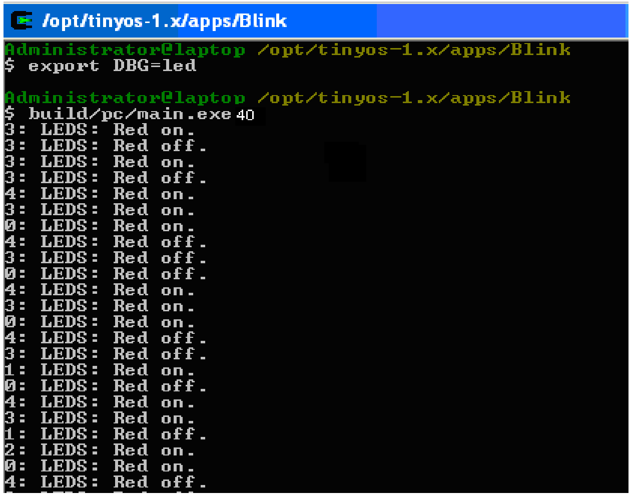
Address C:\tinyos\cygwin\opt\tinyos-1.x\apps\Blink\build\pc

Figure 5-1 Main .exe File Location

5.2.2. How to Run an Application

Following is the procedure for running the successful compiled application in TOSSIM.

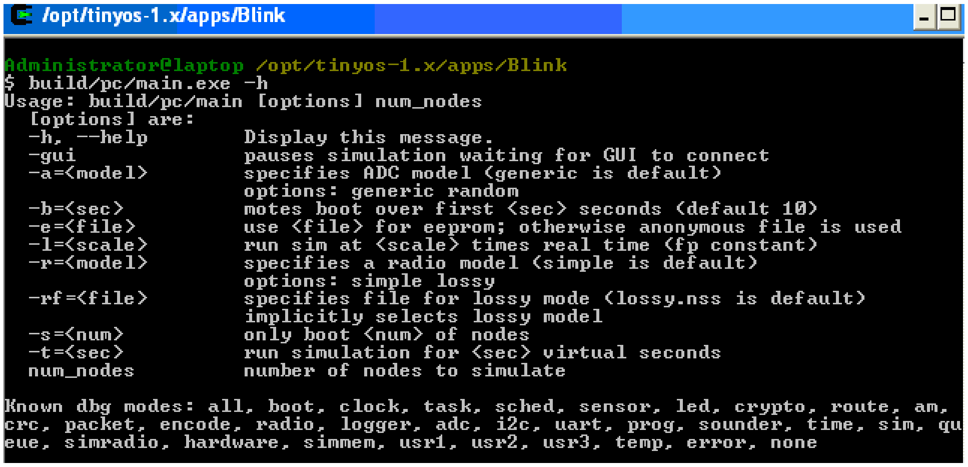
- Open the Cygwin shell
- Go to the required application's directory
- Write down the command "build/pc/main.exe 40" to run this application for a network of 40. Snapshot the running simulation is shown in Figure 5-2
- Snapshot of help command in Cygwin shell is shown in Figure 5-3.



```
Administrator@laptop /opt/tinyos-1.x/apps/Blink
$ export DBG=led

Administrator@laptop /opt/tinyos-1.x/apps/Blink
$ build/pc/main.exe 40
3: LEDS: Red on.
3: LEDS: Red off.
3: LEDS: Red on.
3: LEDS: Red off.
4: LEDS: Red on.
3: LEDS: Red on.
0: LEDS: Red on.
4: LEDS: Red off.
3: LEDS: Red off.
0: LEDS: Red off.
4: LEDS: Red on.
3: LEDS: Red on.
0: LEDS: Red on.
4: LEDS: Red off.
3: LEDS: Red off.
1: LEDS: Red on.
0: LEDS: Red off.
4: LEDS: Red on.
3: LEDS: Red on.
1: LEDS: Red off.
2: LEDS: Red on.
0: LEDS: Red on.
4: LEDS: Red off.
```

Figure 5-2 Snapshot of simulation running



```
Administrator@laptop /opt/tinyos-1.x/apps/Blink
$ build/pc/main.exe -h
Usage: build/pc/main [options] num_nodes
  [options] are:
  -h, --help          Display this message.
  -gui                pauses simulation waiting for GUI to connect
  -a=<model>          specifies ADC model (generic is default)
                      options: generic random
  -b=<sec>            motes boot over first <sec> seconds (default 10)
  -e=<file>           use <file> for eeprom; otherwise anonymous file is used
  -l=<scale>          run sim at <scale> times real time (fp constant)
  -r=<model>          specifies a radio model (simple is default)
                      options: simple lossy
  -rf=<file>          specifies file for lossy mode (lossy.nss is default)
                      implicitly selects lossy model
  -s=<num>            only boot <num> of nodes
  -t=<sec>            run simulation for <sec> virtual seconds
  num_nodes           number of nodes to simulate

Known dbg modes: all, boot, clock, task, sched, sensor, led, crypto, route, am,
crc, packet, encode, radio, logger, adc, i2c, uart, prog, sounder, time, sim, qu
eue, simradio, hardware, simmem, usr1, usr2, usr3, temp, error, none
```

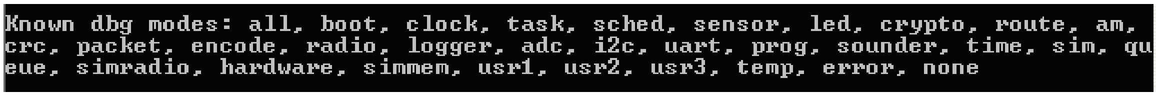
Figure 5-3 Snapshot of Help command

5.2.3. TOSSIM Debug modes

TOSSIM provides debugging environment at run time. For each debug statement, DBG flags are set in code. All the debug modes are enabled by default. To enable some specific debug modes, use the following command.

export DBG = “*debug mode name*”

TOSSIM’s known DBG modes are shown in Figure 5-3 and Figure 5-4.



```
Known dbg modes: all, boot, clock, task, sched, sensor, led, crypto, route, am,
crc, packet, encode, radio, logger, adc, i2c, uart, prog, sounder, time, sim, qu
eue, simradio, hardware, simmem, usr1, usr2, usr3, temp, error, none
```

Figure 5-4 Known DBG modes

5.2.4. TOSSIM GUI Tools

TinyViz provides the Graphical User Interface (GUI) for the simulation in TOSSIM. Following commands are used for graphical representation of TOSSIM simulation in TinyViz.

- Write down the following command in one command line window (shell)

```
Build/pc/main.exe –gui 40
```

- Open a new shell (command line window) and then write down the following command.

```
java net.tinyos.sim.TinyViz
```

This will run a java based GUI tool called TinyViz. TinyViz also provides different plug-ins for different functionalities. A snapshot of TinyViz with random node topology and opened list of available plugins is shown in Figure 5-5.

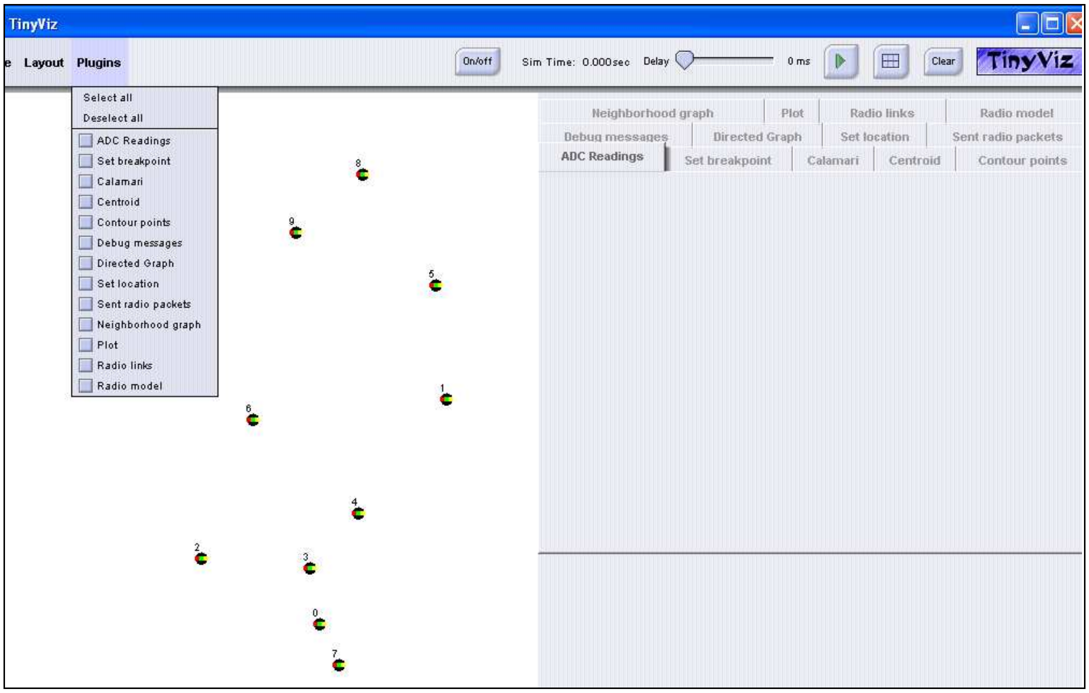


Figure 5-5 A snapshot of TinyViz with random node topology and available plugins

5.3. NesC Language

NesC, an improved version of C language for network embedded systems is introduced with TinyOS. TinyOS has built-in compiler for NesC language. Key features of this language are listed as under.

- Components based language
- The component may provide or use interface that is why the interfaces are called bidirectional.
- It defines a concurrency model, based on tasks and hardware, and detects data races at compile time [82].
- Any type of application has two files for nesC programming. One for wiring the interfaces and other is for actual implementation.
- NesC file extension is *.nc*. The file containing the wiring of the components is called configuration file and the other one with code is called Module file.

Detailed discussion on TinyOS and NesC language structure has already been carried out in section 1.10. A complete TinyOS application structure is shown in Figure 5-6.

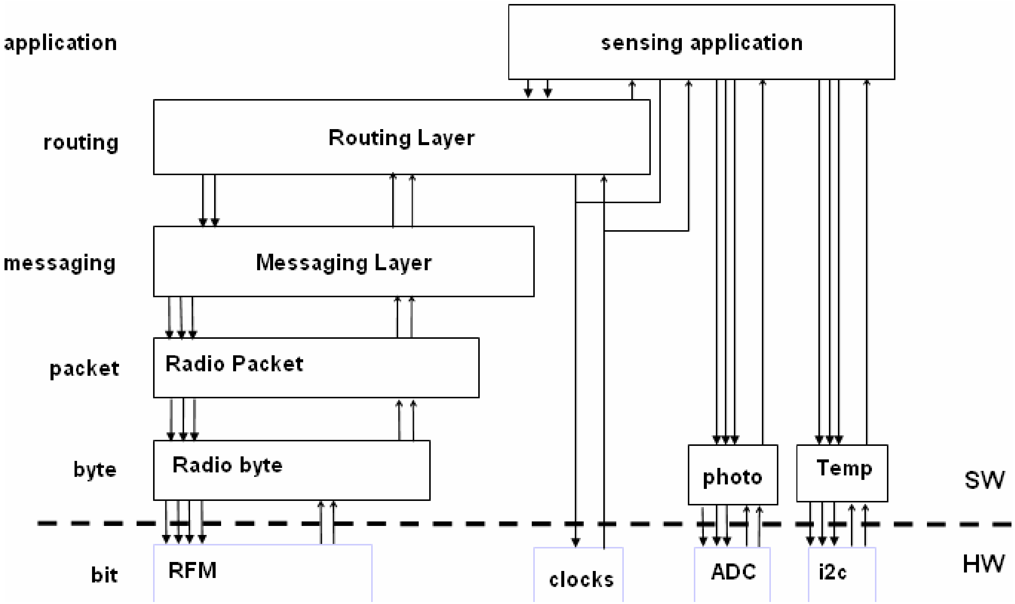


Figure 5-6 A complete TinyOS application [83]

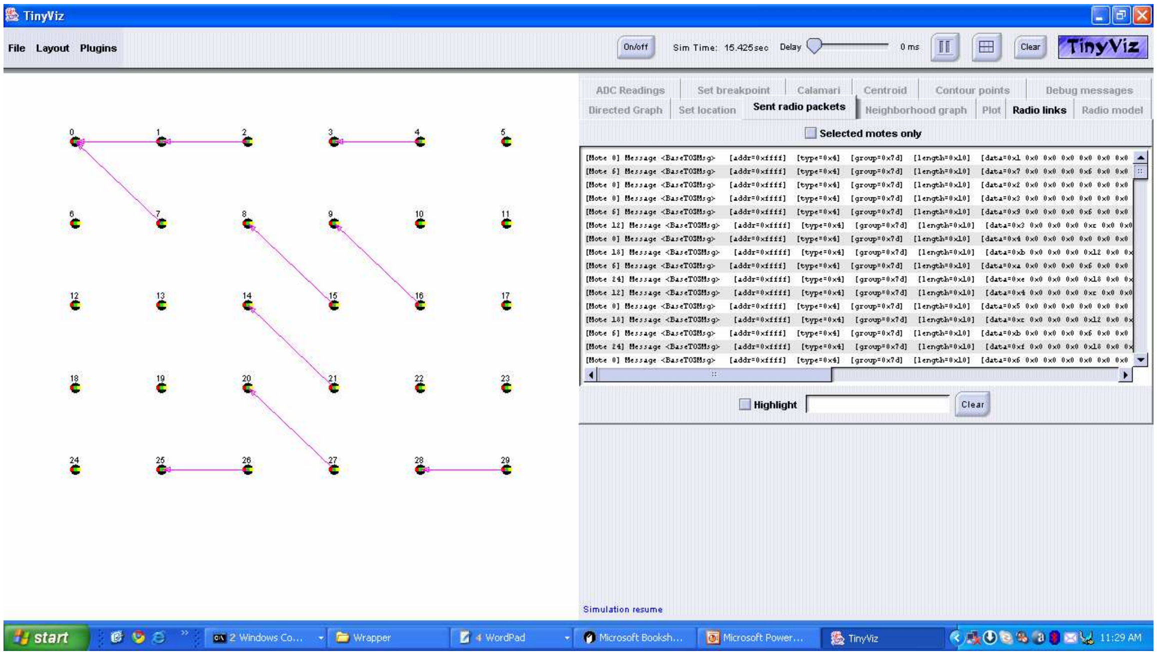
Results and Discussion

Chapter 6

Results and Discussion

6.1. Results and Discussion

For performance evaluation of our proposed algorithm, REAR, TinyOS SIMulator (TOSSIM) by UC Berkeley [41] [82] is used. Figure 44, Figure 45, Figure 46, Figure 47, Figure 48, Figure 49, Figure 50 and Figure 51 illustrate some of the initial results drawn from simulations of our proposed real-time energy aware routing protocol. Figure 3 shows the snapshots of simulation running. We compared REAR protocol to TRER protocol. All the environmental parameters of both the algorithms are same. We have considered energy consumption per node, no. of packets achieving the deadline, total delay on varying source to sink transit nodes and network life time as the performance parameters for evaluation the efficiency of REAR against TRER.



```
[Mote 0] Message <BaseTOSMsg>   [addr=0xffff]  [type=0x4]  [group=0x7d]  [length=0x10]  [data=0x1 0x0 0x0 0x0 0x0 0x0 0x0
[Mote 6] Message <BaseTOSMsg>   [addr=0xffff]  [type=0x4]  [group=0x7d]  [length=0x10]  [data=0x7 0x0 0x0 0x0 0x6 0x0 0x0
[Mote 0] Message <BaseTOSMsg>   [addr=0xffff]  [type=0x4]  [group=0x7d]  [length=0x10]  [data=0x2 0x0 0x0 0x0 0x0 0x0 0x0
[Mote 0] Message <BaseTOSMsg>   [addr=0xffff]  [type=0x4]  [group=0x7d]  [length=0x10]  [data=0x3 0x0 0x0 0x0 0x0 0x0 0x0
[Mote 6] Message <BaseTOSMsg>   [addr=0xffff]  [type=0x4]  [group=0x7d]  [length=0x10]  [data=0x9 0x0 0x0 0x0 0x6 0x0 0x0
[Mote 12] Message <BaseTOSMsg>   [addr=0xffff]  [type=0x4]  [group=0x7d]  [length=0x10]  [data=0x3 0x0 0x0 0x0 0xc 0x0 0x0
[Mote 0] Message <BaseTOSMsg>   [addr=0xffff]  [type=0x4]  [group=0x7d]  [length=0x10]  [data=0x4 0x0 0x0 0x0 0x0 0x0 0x0
[Mote 18] Message <BaseTOSMsg>   [addr=0xffff]  [type=0x4]  [group=0x7d]  [length=0x10]  [data=0xb 0x0 0x0 0x0 0x12 0x0 0x
[Mote 6] Message <BaseTOSMsg>   [addr=0xffff]  [type=0x4]  [group=0x7d]  [length=0x10]  [data=0xa 0x0 0x0 0x0 0x6 0x0 0x0
[Mote 24] Message <BaseTOSMsg>   [addr=0xffff]  [type=0x4]  [group=0x7d]  [length=0x10]  [data=0xe 0x0 0x0 0x0 0x18 0x0 0x
[Mote 12] Message <BaseTOSMsg>   [addr=0xffff]  [type=0x4]  [group=0x7d]  [length=0x10]  [data=0x4 0x0 0x0 0x0 0xc 0x0 0x0
[Mote 0] Message <BaseTOSMsg>   [addr=0xffff]  [type=0x4]  [group=0x7d]  [length=0x10]  [data=0x5 0x0 0x0 0x0 0x0 0x0 0x0
[Mote 18] Message <BaseTOSMsg>   [addr=0xffff]  [type=0x4]  [group=0x7d]  [length=0x10]  [data=0xc 0x0 0x0 0x0 0x12 0x0 0x
[Mote 6] Message <BaseTOSMsg>   [addr=0xffff]  [type=0x4]  [group=0x7d]  [length=0x10]  [data=0xb 0x0 0x0 0x0 0x6 0x0 0x0
[Mote 24] Message <BaseTOSMsg>   [addr=0xffff]  [type=0x4]  [group=0x7d]  [length=0x10]  [data=0xf 0x0 0x0 0x0 0x18 0x0 0x
[Mote 0] Message <BaseTOSMsg>   [addr=0xffff]  [type=0x4]  [group=0x7d]  [length=0x10]  [data=0x6 0x0 0x0 0x0 0x0 0x0 0x0
```

Figure 35: Snapshot of REAR’s Simulation in TinyViz

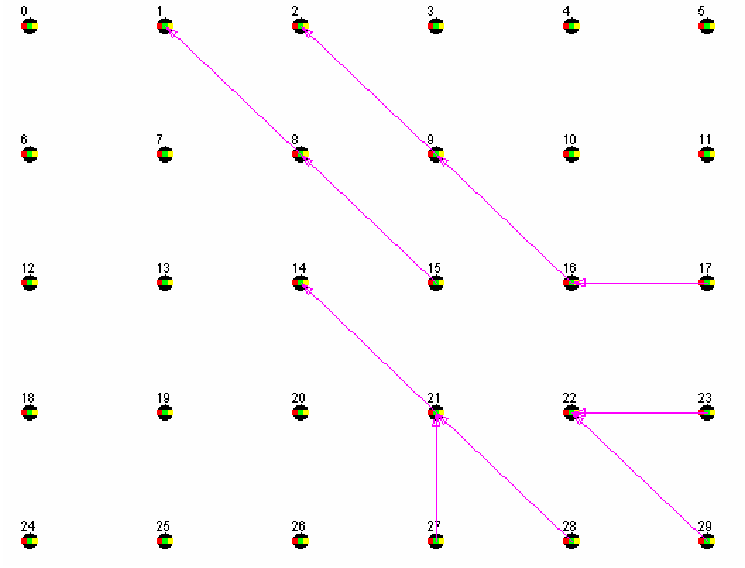


Figure 36: Snapshot of REAR's Simulation

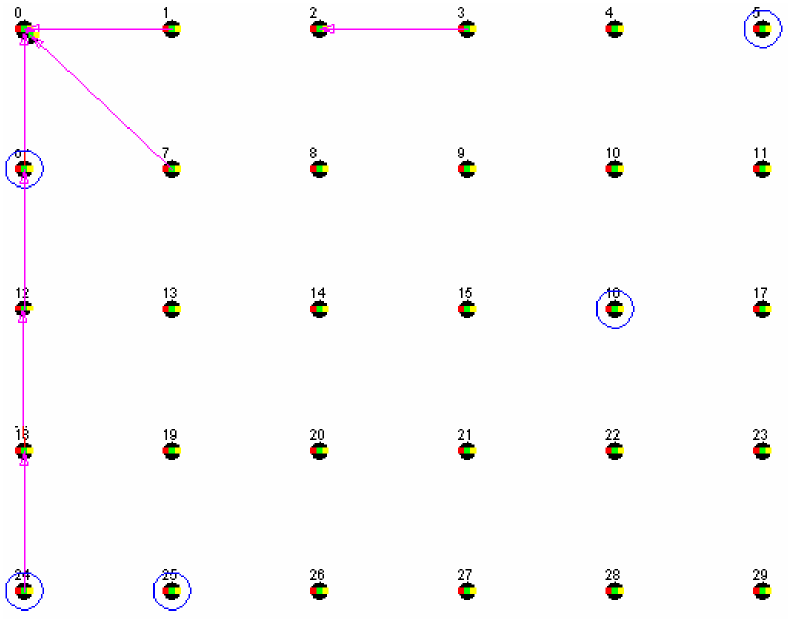
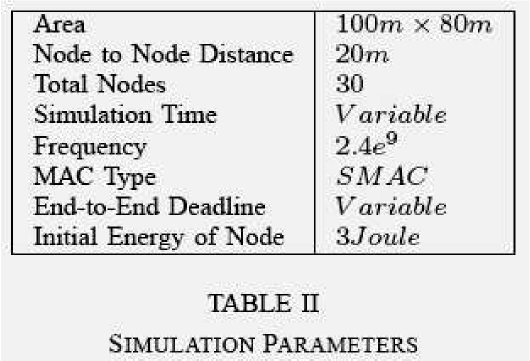


Figure 37: Snapshot of REAR's Simulation



Area	$100m \times 80m$
Node to Node Distance	$20m$
Total Nodes	30
Simulation Time	$Variable$
Frequency	$2.4e^9$
MAC Type	$SMAC$
End-to-End Deadline	$Variable$
Initial Energy of Node	$3Joule$

TABLE II

SIMULATION PARAMETERS

A. Network Life Time

Time of death of first node defines the life time of a network. Figure 45 shows the energy consumption per node during the operation of REAR and TRER. Figure 44 and Figure 46 illustrate that REAR consumes lesser energy because it forwards the packet to the node with minimum path time to destination and with minimum hop count to destination which results in prolonging the network life time as shown in Figure 45. Moreover, It is intuited from the results that following the path by taking care of end-to-end delay along with minimum hop count to destination favors the energy saving strategies resulting in more entertainment of packets by the network. Figure 47 also shows the better performance of REAR as compared to TRER regarding the packets entertained per node.

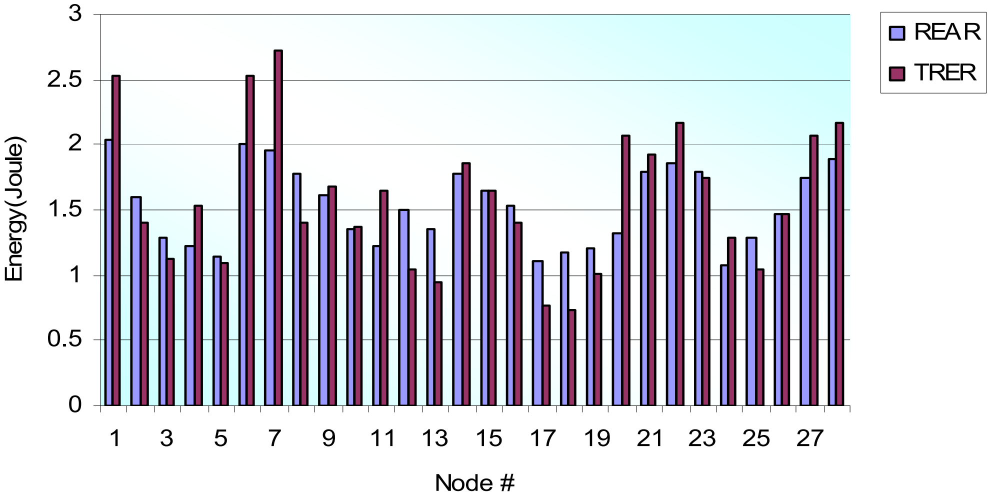


Figure 38: Energy Consumption per Node

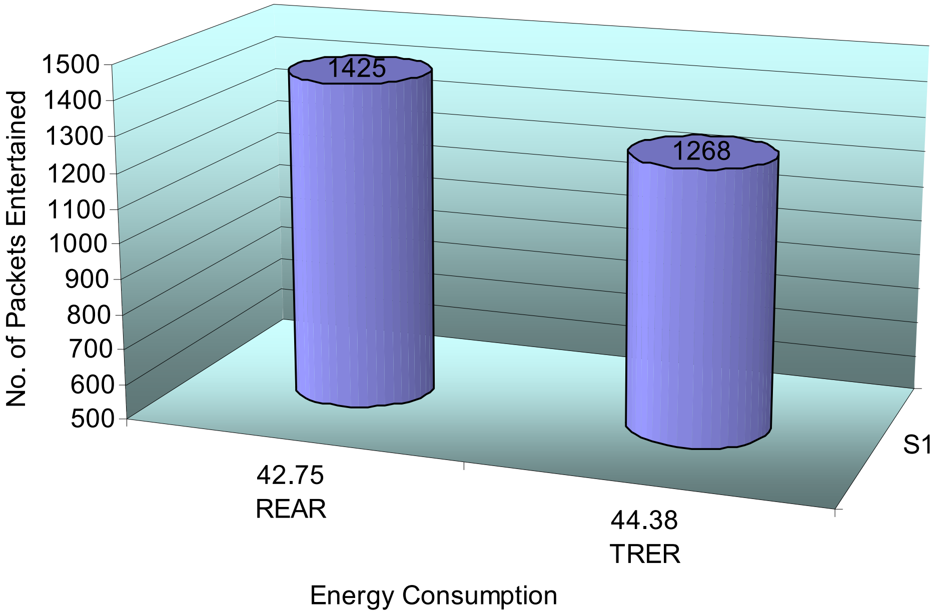


Figure 39: Network Utilization Vs Energy Consumption

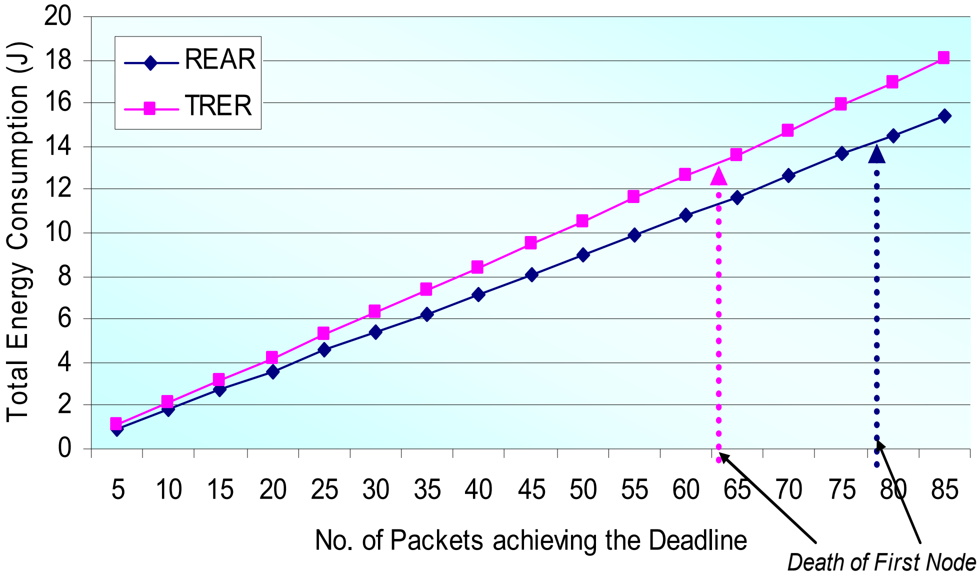


Figure 40: Network Life Time

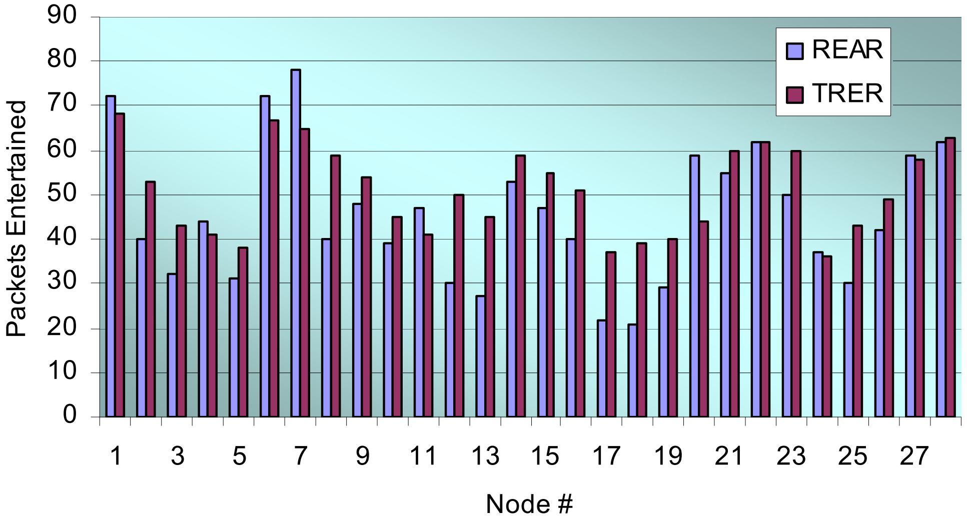


Figure 41: Packets Entertained per Node

B. End-to-End Delay

Figure 48 shows the relationship between the number of hops from source to sink and the end-to-end delay that a packet takes on passing through different number of hops. Performance of REAR is ahead of TRER in achieving the sink due to following the path with minimum Path time to destination and minimum hop count to destination as well as in-network processing strategy. Figure 49 shows the better performance of REAR as compared to TRER regarding total no. of packets entertained Vs total time consumed accordingly.

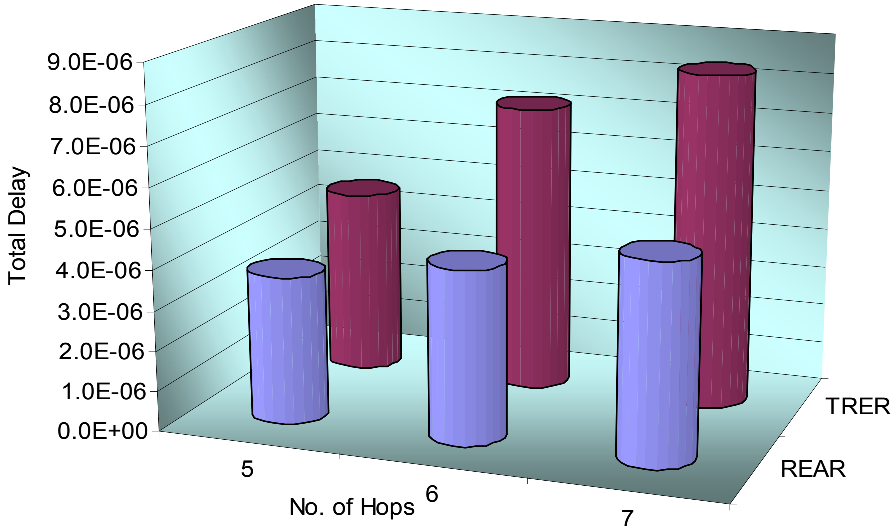


Figure 42: End-to-End Delay

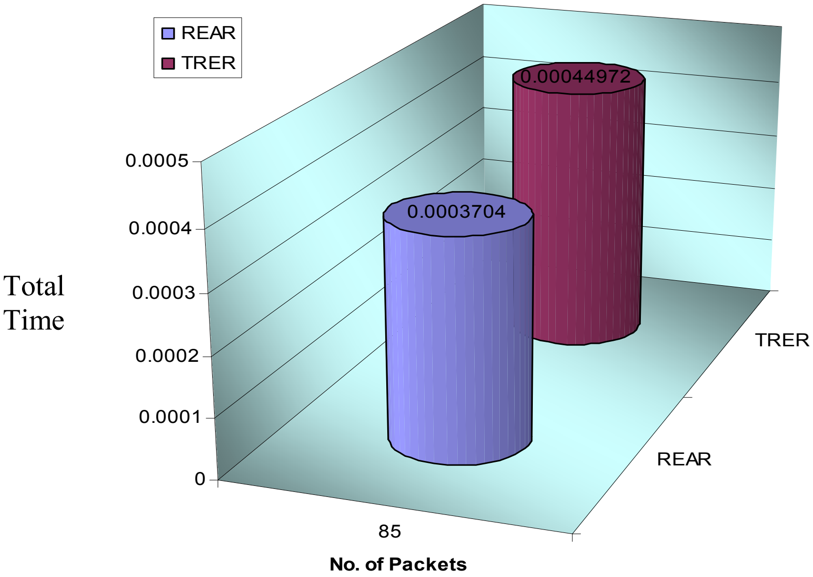


Figure 43: No. Packets Vs Total Time

C. Performance Efficiency

The ultimate target of our proposed algorithm is to achieve the target in real-time fashion and prolonging the network life time. Figure 50 and Figure 51 show the honor of REAR with these targets by its in-network processing with customized proactive routing strategies and localized behavior. We use the following formula to find the performance efficiency of REAR (Figure 50). Positive Trend line represents the better results in end-to-end delay from 20% to 40% as compared to TRER.

$$\xi = (1 - \frac{REAR's\ Result}{TRER's\ Result}) \times 100$$

More-over, by using the same above mentioned formula, calculation of performance efficiency of REAR in conserving the scarce resource of energy is up to 14% better results for the packets achieving the deadline (Figure 9). Figure 9 also illustrates the energy conservation efficiency of REAR in another aspect. The difference between the consumption of both the competing algorithms is in the favor of REAR.

$$Dif(Eng\_Cons_{(TRER,REAR)}) = Eng_{(TRER)} - Eng_{(REAR)}$$

On comparing the overall performance of both the algorithms, It is evident from Figure 6 that about more than 11% packets are entertained and more than 4% energy is conserved by using the REAR algorithm as compared to TRER.

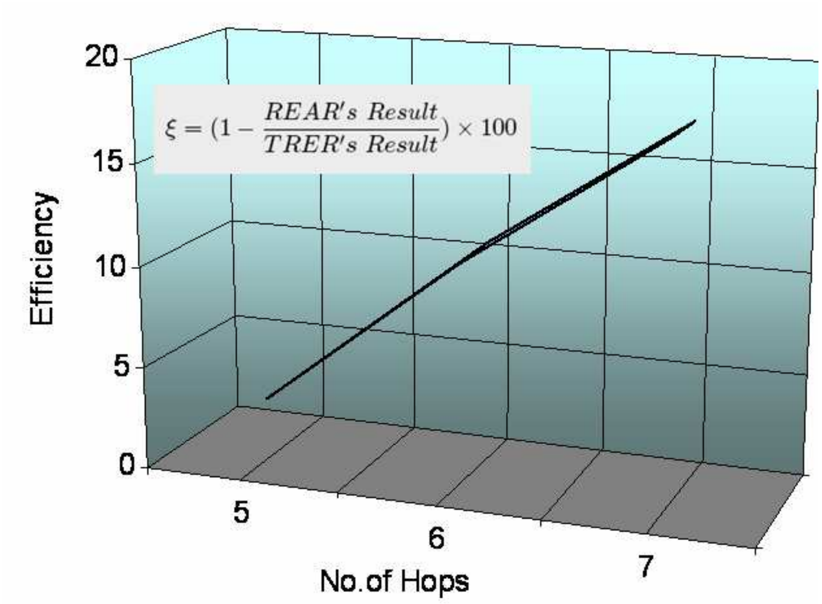


Figure 44: REAR's Performance Efficiency Chart (End-to-End Delay)

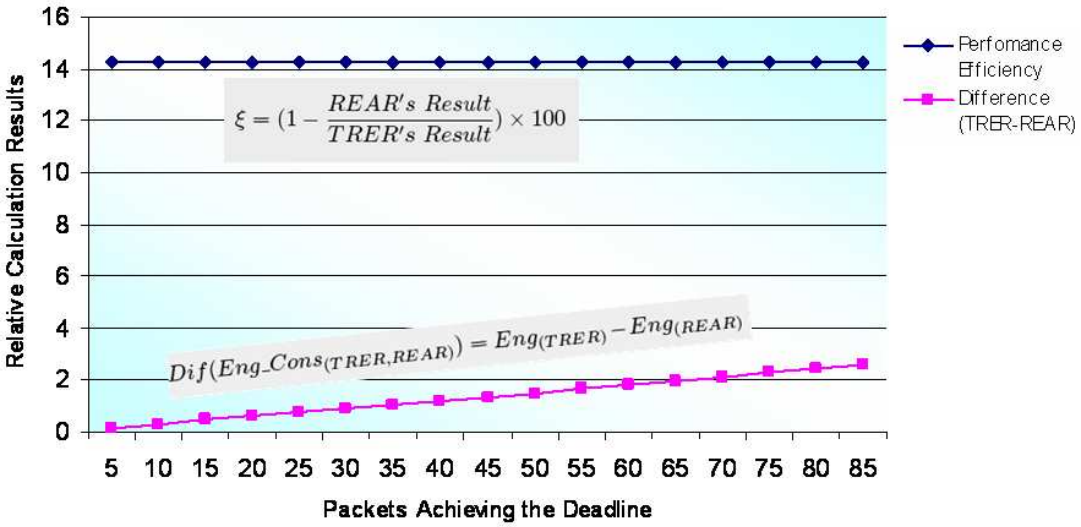


Figure 45: REAR's Performance Chart (Energy Consumption)

Conclusion and Future Work

Chapter 7

Conclusion

7.1. Conclusion

Ubiquitous application of this emerging and ambient technology with stringent constraint resources demand special techniques for setting up tradeoff management among the competing constraint factors along with achieving the defined targets. Our proposed algorithm; REAR, has targeted these objectives. Empirical results show that reactive routing offers an unexpected delay in contrast to proactive routing but proactive routing is hungry for energy consumption. Our customized proactive routing offers a better solution compared to above both. Lessening the beacon messages exchange and in-network processing not only assist in decreasing the energy consumption resulting in prolonging the network life time but also decreases end-to-end delay in processing, communication and decision making time. Simulation results show the better performance of REAR in energy consumption with the effect of improving the network life. Also the end-to-end delay performance efficiency as compared to TRER is better.

7.2. Future Work

Our future work includes the implementation of our idea in different real environments by using real motes. More-over, embedding of energy efficient load sharing mechanism will add better fault tolerance capability in our proposed solution. Following is one of the expected extension of our proposed solution.

- "***AREAR: ADAPTIVE REAL-TIME ENERGY-AWARE ROUTING FOR WIRELESS SENSOR NETWORK***",

Bibliography

1. Jacob Fraden, Handbook of Modern Sensors: Physics, Designs, and Applications. Birkhäuser – 2004. Ed. 3, ISBN - 0387007504, 9780387007502
2. M. Mohammad Ilyas, Ed., Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems. CRC Press, 2004.
3. Online: www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAZ_Datasheet.pdf
4. F. L. Lewis. Wireless Sensor Networks. To appear in Smart Environments: Technologies, Protocols, and Applications ed. D.J. Cook and S.K. Das, John Wiley, New York, 2004
5. Online: http://robotics.eecs.berkeley.edu/~pister/SmartDust/
6. Kay R¨omer and Friedemann Mattern, "The Design Space of Wireless Sensor Networks", IEEE Wireless Communications, Dec. 2004. Online: http://nest.cs.berkeley.edu/nest-index.html
7. Mohammad Ilyas, I. M. Ed., Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems. CRC Press, 2004.
8. Y. W. M. M. L. S. P. P. Juang, H. Oki and D. Rubenstein, "Energy efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet," ASPLOS X. San Jose, USA: IEEE, October 2002.
9. R. Riem-Vis, "Cold chain management using an ultra low power wireless sensor network." Boston, USA: WAMES 2004, June 2004.
10. R. O. J. K. H. K. Martinez and J. Stefanov, "Glacsweb: A sensor web for glaciers." Berlin Germany: EWSN 2004, January 2004.
11. S. F. Michahelles, P. Matter and B. Sciele, "Applying wearable sensors to avalanche rescue," no. 27(6), 2003.
12. K. K. H. Baldus and G. Muesch, "Reliable setup of body sensor networks." Berlin, Germany: EWSN2004, January 2004.
13. L. G. Simon and M. Maroti, "Sensor network-based counter sniper system." Baltimore, USA: SenSys, November 2004.
14. D. T. R. Beckwith and P. Bowen, "Pervasive computing and proactive agriculture." Viena, Austria: PERVASIVE2004, April 2004.
15. Sohail Jabbar, Abid Ali Minhas, Raja Adeel Akhtar, "SPERT: a stateless protocol for energy-sensitive real-time routing for wireless sensor network" 3rd International Conference on Information and Communication Technologies (ICICT'2009), August, 2009 at IBA, Karachi, Pakistan,
16. Smart Dust communication with a cubic-millimeter computer, Brett warneke, Mattlast, Brian Liebowitz and K.S.J. Pister, IEEE Computer, 34(1): 44-51, 2001
17. Online: http://en.wikipedia.org/wiki/Smart_dust
18. MICA: A wireless platform for deeply embedded networks by Hill, J. L. Micro, IEEE, vol. 22, issue 6, nov. dec. 2002, p 2-24
19. W. Holger Karl, Protocols and Architecture for Wireless Sensor Network. British Libr. cataloguing in publication data, July 2007.

20. V. Raghunthan, C. Schurger, S. Park, and M.B. Srivastava. Energy-Aware Wireless Microsensor Networks. IEEE Signal Processing Magazine, 19: 40-50, 2002

21. MTS/MDA Sensor Board Users Manual, Revision A, June 2007, PN: 7430-0020-05, Available: www.xbow.com

22. Qing Li, Caroline Yao, "Real-Time Concepts for Embedded Systems" CMP Press,2003, ISBN: 1-57820-124-1, page 53

23. Onine: http://en.wikipedia.org/wiki/Embedded_system

24. DAVID GAY, PHILIP LEVIS, DAVID CULLER, Software Design Patterns for TinyOS, ACM Transactions on Embedded Computing Systems, Vol. 6, No. 4, Article 22, Publication date: September 2007

25. Stefan Dulman and Paul Havinga. Operating system fundamentals for the eyes distributed sensor networks. University of Twente, Department of Computer sciemce Enschede, the Netherlands, [http://www.eyes.eu.org/publications/], October 2002.

26. Online: http://www.sics.se/contiki/

27. Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In 29th Annual IEEE International Conference on Local Computer Networks, 16

28. Online: http://www.ercim.org/.

29. Online: http://www.tinyos.net.

30. Lin Gu." Virtualizing operating system for wireless sensor networks". University of Virginia Charlottesville, VA, USA . Pages: 176 Year of Publication: 2006.

31. Chih-Chieh Han, Ram Kumar, Roy Shea, Eddie Kohler, and Mani Srivastava. A dynamic operating system for sensor nodes. In MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services, pages 163–176, New York, NY, USA, 2005. ACM Press

32. Online: http://nesl.ee.ucla.edu/projects/sos-1.x/publications/

33. Online: http://Sensornode-Wikipedia.thefreeencyclopedia.htm

34. C. E. Perkins and E. Royer, "Ad-hoc on demand distance vector router," 2nd IEEE Workshop on Mobile Computing System and Appliations(WMCS 99). New Orleans, Louisiana, USA: IEEE, February 1999.

35. D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless network," T. Imielinkski and H. Korth, Eds., Mobile Computing. Kluwer Academy, 1996, pp. 153–181.

36. V. Park and M. Corson, "A highly adaptive distribution routing algorithm for mwn," 1997, pp. 1405 – 1417.

37. Y.-B. Ko and N. H. Vaidya, "Location - aided routing," International Conference on Mobile Computing and Networking(MobiCom 1998). Dallas, Tx, USA: ACM/IEEE, October 1998.

38. H. K. Brad Karp, "Greedy perimeter stateless routing for wireless network." Boston, MA: Sixth Annual ACM/IEEE international Conference of MobiCom 2000, July 2000, pp. 243–254.

39. Jamal N. Karaki, Ahmed E. Kamal, "Routing Techiniques in Wireless Sensor Network: A Survey"IEEE Wireless Communicaion Journal, 204, Vol. 11, Pages 6-28

40. Elizabeth M. Royer, Chai-Keong Toh, "A review of current routing protocols for Adhoc Mobile Wireless Network", IEEE Personal communication, April 1999

41. Holger Karl, Andreas Willing, "Protocol and Architecture for Wireless Sensor Network", 2007 John Wiley and Sons Ltd. Page # 290.

42. 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, pp. 56-67.

43. J.-H. Chang and L. Tassiulas, \Maximum Lifetime Routing in Wireless Sensor Networks", Proc. Advanced Telecommu-nications and Information Distribution Research Program (ATIRP2000), College Park, MD, Mar. 2000.

44. C. Rahul, J. Rabaey, \Energy Aware Routing for Low Energy Ad Hoc Sensor Networks", IEEE Wireless Communications and Networking Conference (WCNC), vol.1, March 17-21, 2002, Orlando, FL, pp. 350-355

45. S. Dulman, T. Nieberg, J. Wu, P. Havinga, \Trade-Off between Traffc Overhead and Reliability in Multipath Routing for Wireless Sensor Networks", WCNC Workshop, New Orleans, Louisiana, USA, March 2003.

46. D. Braginsky and D. Estrin,\Rumor Routing Algorithm For Sensor Networks", International Conference on Distributed Computing Systems (ICDCS'01), November 2001.

47. Sohail Jabbar, Abid Ali Minhas, Muhammad Zubair Aziz, “QERT: Query Based Energy Aware Real-Time Routing for Wireless Sensor Network” ICET’09 at NU-FAST, Islamabad, Pakistan,

48. K. Sohrabi, J. Pottie, "Protocols for self-organization of a wireless sensor network", IEEE Personal Communications, Volume 7, Issue 5, pp 16-27, 2000.

49. T. He et al., SPEED: A stateless protocol for real-time communication in sensor networks", in the Proceedings of International Conference on Distributed Computing Systems, Providence, RI, May 2003.

50. C. L. T. A. T.He, J. Stankovic, Ed., SPEED:A stateless protocol for real-time communication in Sensor Network, no. 46 - 55. IEEE, 2003

51. E. E. Emad Felemban, Chang-Gun Lee, “Mmspeed:multipath multispeed protocol for qos guarantee of reliability and timeliness in wsn,” IEEE Transactions on Mobile Computing, vol. 5, no. 6, pp. 738–754, June 2006.

52. G. A. Noor M. Khan, Zubair Khalid, Ed., A Real-Time Energy-aware Routing Strategy for Wireless Sensor Network, Proceeding of Asia-Pacific Conference on Communication. IEEE, 2007.

53. T. F. A. J. A. S. T. H. Chenyang Lu, Brian M. Blum, “Rap: A real-time communicaiton architecture for large-scale wireless sensor networks.” San Jose, CA: Real-Time Technology and Application Symposium, September 2002.

54. X. D. Linfeng Yuan, Wenqing Cheng, “An energy-efficient real-time routing protocol for sensor networks,” vol. 30, ELSEVIER. ScienceDirect, June 2007.

55. Paul Bone. Real-Time Communication and Coordination in Wireless Embedded Sensor Network. April, 2004.

56. Yanjun Li, Chung Shue Chen, Ye-Qiong Song, Zhi Wang. Real-Time Qos Support In Wireless Sensor Networks: A Survey.

57. Mohamed Younas, Kemal Akkaya, Mohamed Eltoweissy, Ashraf Wadda. On Handling QoS Traffic in Wireless Sensor Networks.

58. R. K. Sachin Sharma, Dharmendra Kumar, “Qos-based routing protocol in wsn,” vol. 1, no. 1-3, Advances in Wireless and Mobile Communications. Research India Publications, 2008, pp. 51–57.

59. GABRIEL, K., AND SOKAL, R. A new statistical approach to geographic variation analysis. Systematic Zoology 18 (1969), 259–278.

60. TOUSSAINT, G. The relative neighborhood graph of a finite planar set,. Pattern Recognition 12, 4 (1980), 261–268.

61. THE CMU MONARCH GROUP. Wireless and Mobility Extensions to ns-2. Available at: http://apachepersonal.miun.se/~qinwan/resources/ns2%20resources/ns-cmu[1].pdf , August 2009.

62. 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, 91(7), 2003.

63. Xiang Zeng, Rajive Bagrodia, and Mario Gerla , GloMoSim: a Library for Parallel Simulation of Large scale Wireless Networks. In Proceedings of the 12th Workshop on Parallel and Distributed Simulations --PADS '98, May 26-29, 1998 in Banff, Alberta.

64. Stojmenovic and X. Lin. GEDIR: Loop-Free Location Based Routing in Wireless Networks, IASTED Int. Conf. on Parallel and Distributed Computing and Systems, November 3-6, 1999

65. L. Zhao, B. Kan), Y. Xu, X. Li. FT-SPEED: A Fault-Tolerant, Real-Time Routing Protocol for Wireless Sensor Networks. 1-4244-1312-5/07, 2007 IEEE

66. Q.Fang, J.Gao and L.J. Guibas. Locating and bypassing routing holes in sensor networks. In Proceedings of IEEE Conference on Computer Communications (INFOCOM'04) ,Mar.2004.

67. T. N. Arvanitis, C. C. Constantinou, A. S. Stepanenko, Y. Sun, B. Liu, and K. Baughan, "Network visualisation and analysis tool based on logical network abridgment," in Proc. Military Communication Conf. (MilCom'05), vol. 1, October 2005, pp. 106–112.

68. O. Chipara, Z. He, G. Xing, Q. Chen, X. Wang, C. Lu, J.A. Stankovic and T.F. Abdelzaher, Real-time Power-Aware Routing in Sensor Networks, IEEE International Workshop on Quality of Service (IWQoS'06), June 2006

69. J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in SenSys '03, 2003, pp. 1–13.

70. J. Hightower and G. Borriello, "Location systems for ubiquitous computing," IEEE Computer, 2001

71. A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in SenSys '03, 2003.

72. E. D. Demaine, A. Lopez-Ortiz, and J. I. Munro, "Frequency estimation of internet packet streams with limited space," in Proceedings of the 10th Annual European Symposium on Algorithms, 2002, pp. 348–360.

73. G. Simon, P. Volgyesi, M. Maroti, and A. Ledeczi, "Simulation-based optimization of communication protocols for large-scale wireless sensor networks," in IEEE Aerospace Conference '03, 2003.

74. G.G. Finn, "Routing and addressing problem in large metropolitan-scale internetworks". ISI res. Rep ISU/RF-87-180, March 1987.

75. H. Takagi, L. Kleinrock, "Optimal transmission ranges for randomly distributed packet radio terminals", IEEE transactions on communication 32(3) (1984) 246-257.

76. E. Kranakis, H. Singh, J. Urrutia, "Compass routing on geometric networks, proceedings of the 11th Canadian conference on computational geometry, Vancouver, Canada, 1999

77. Xin Lie, Yunsheng Liu, Tian Bai. Energy-efficient Real-Time Routing in Wireless Sensor Network. IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application. 2008.

78. Tian He et. al., Robust and timely communication over highly dynamic sensor networks, Real-Time Syst (2007) 37: 261–289

79. Adel Ali, Liza A. Latiff, Norsheila Fisal. Geodirectional-cast forwarding based on quadrant for Real-time routing in Wireless Sensor Network. Proceedings of 2007 IEEE International Conference on Telecommunications, Malaysia.

80. C. E. Perkins and P. Bhagwat, “Highly dynamic destination-sequenced distance-vector routing (dsdv).” London, UK: SIGCOMM Symposium on Communication Architecture, September 1994, pp. 212–225

81. TOSSIM: A Simulator for TinyOS Networks”. Philip Levis and Nelson Lee. UC Berkeley September 17, 2003.

82. Ben L. Titzer, Daniel K. Lee, and Jens Palsberg. Avrora: scalable sensor network simulation with precise timing. In Fourth International Symposium on Information Processing in Sensor Networks (IPSN), pages 477 – 482, UCLA, Los Angeles, CA, USA, April 2005

83. Online: http://people.ee.duke.edu/~romit/courses/s07/material/lecture-sensor-ddiffusion-leach.ppt

84. Philip Levis, Nelson Lee, Matt Welsh, and David Culler. TOSSIM: Accurate and scalable simulation of entire tinyos applications. In Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003). Available at: http://csl.stanford.edu/~pal/pubs.html

85. Microsoft ENCRATA World English Dictonary

86. Manish Karir, Jonathan Polley, Dionysys Blazakis, Jonathan McGee, Dan Rusk, and John S. Baras. Atemu: a fine-grained sensor network simulator. In First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, pages 145 – 152, October 2004.

87. Jeremy Elson, Lewis Girod, and Deborah Estrin. Emstar: Development with high system visibility. In Wireless Communications, pages 70 – 77, December 2004.

88. Rimon Barr. Swans: Scalable wireless ad hoc network simulation. In Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad hoc Wireless, and Peer to Peer networks. Ch. 19, CRC Press, pages 297–311, (http://jist.ece.cornell.edu/docs.html) September 2005.

89. Online: http://www.scalable networks.com/

90. Online: http://www.opnet.com/.

91. Online: http://www.cs.rpi.edu/ cheng3/sense/

92. Online: http://www.cygwin.com/