STATUS OF THESIS

Title of thesis: Efficient Dynamic Addressing Based Routing For Underwater Wireless Sensor Networks

I, MUHAMMAD AYAZ ARSHAD

hereby allow my thesis to be placed at the Information Resource Center (IRC) of Universiti Teknologi PETRONAS (UTP) with the following conditions:

1. The thesis becomes the property of UTP
2. The IRC of UTP may make copies of the thesis for academic purposes only.
3. This thesis is classified as

☐ Confidential

☑ Non-confidential

If this thesis is confidential, please state the reason:

_____
_____
_____

The contents of the thesis will remain confidential for _____ years.
Remarks on disclosure:

_____
_____
_____

Endorsed by

_____ _____

Signature of Author | Signature of Supervisor

Permanet Address: | Name of Supervisor
Chak Rasulpur, Adda Hari Minor, | Assoc. Prof. Dr. Azween bin Abdullah
Tehsil Mailsi, Vehari, Punjab (Pakistan)

Date:_____ | Date: _____

UNIVERSITI TEKNOLOGI PETRONAS

EFFICIENT DYNAMIC ADDRESSING BASED ROUTING FOR UNDERWATER WIRELESS SENSOR NETWORKS

By

MUHAMMAD AYAZ ARSHAD

The undersigned certify that they have read, and recommend to the Postgraduate Studies Programme for acceptance this thesis for the fullfilment of the requirements for the degree stated.

Signature: _____

Main Supervisor: Assoc. Prof. Dr. Azween Bin Abdullah

Signature: _____

Cosupervisor: Dr. Ibrahima Faye

Signature: _____

Head of Department: Dr. Mohd Fadzil Bin Hassan

Date: _____

EFFICIENT DYNAMIC ADDRESSING BASED ROUTING FOR UNDERWATER WIRELESS SENSOR NETWORKS

by

MUHAMMAD AYAZ ARSHAD

A Thesis

Submitted to the Postgraduate Studies Programme

As a requirement for the degree of

DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES

UNIVERSITI TEKNOLOGI PETRONAS

BANDAR SERI ISKANDAR

PERAK

SEPTEMBER 2011

DECLARATION OF THESIS

Title of thesis	Efficient Dynamic Addressing Based Routing For Underwater Wireless Sensor Networks

I, MUHAMMAD AYAZ ARSHAD

hereby declare that the thesis is based on my original work except for quotations and citations, which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTP or other institutions.

Witnessed by

Signature of Author	Signature of Supervisor
Permanent address:	Name of Supervisor
Chak Rasulpura, Adda Hari Minor, Mailsi, Vehari, Punjab (Pakistan)	Assoc. Prof. Dr. Azween Bin Abdullah
Date :	Date :

To my beloved prophet (PBUH)

ACKNOWLEDGEMENT

At the end of this journey of dissertation, I realize the importance of acknowledging. First of all, I would like to say Alhamdulillah, praise Allah The Most Gracious and The Most Merciful, for munificence in bestowing me with so many of His favours throughout my life particularly in enhancing my courage for the completion of this work successfully.

I would like to express my gratitude to my respected supervisor, *AP Azween B. Abdullah* for his consistent support, guidance and well rounded experience, which I will treasure in my career. He has always been tough but fair in his criticism of my work and I have great appreciation for his management in maintaining such a difficult balance. I also want to acknowledge Dr. Ibrahima Faye for his significant contribution for this achievement as a co-advisor. I cannot forget our long discussions we have had over my work during that he sat and listened, no matter how tight his schedule was.

Further I want to extend my gratitude to Professor Milica Stojanovic of Massachusetts Institute of Technology, Jun-Hong Cui of University of Connecticut, Dr. Alan G. Downe of Universiti Teknologi PETRONAS, Dr. M.A. Ansari of FUUAST, who provided me with extremely useful technical input and directions. Without their encouragement and helping hands, it was not easy to achieve this goal.

I definitely cannot miss this opportunity to thank Imran Baig and Dr. Yasir Batira for all those valuable discussions at certain times even at my doorstep when I was stuck up with my work. To all my distinguished friends and colleagues, Dr. Muhammad Imran Chaudhry, Dr. Iftikhar Ahmed, Dr. Asfand Yar Khan, Dr. Imran Razzaq, Dr. Low TJ, Dr. Mahamat Issa, Atif Jamil, Irshad Sumra, Tariq Ali, who supported me along the whole this journey, your goodwill shall not be forgotten.

There certainly exist no words that could possibly express the extent of gratitude

I owe my family being there as ever lasting source of support and encouragement. I am deeply grateful to my respected father Hafiz Muhammad Sharif and brother Sijad Sharif for the uncountable sacrifices they made for my studies throughout the years. I strongly believe that, if I were here today then its all due to an answer to my mother's prayers. My beloved mother and sister, whose hands always rise for praying for my success. My love and respect to them are endless and immense.

The author wants to extend his gratefulness to the many anonymous reviewers as their thoughtful and unselfish comments greatly improved the quality of my research articles from which this thesis has been partly extracted.

Last but not least, I appreciate the efforts and cooperation of management of Universiti Teknologi PETRONAS, Malaysia as without their support, this project would not have been possible.

ABSTRACT

This thesis presents a study about the problem of data gathering in the inhospitable underwater environment. Besides long propagation delays and high error probability, continuous node movement also makes it difficult to manage the routing information during the process of data forwarding. In order to overcome the problem of large propagation delays and unreliable link quality, many algorithms have been proposed and some of them provide good solutions for these issues, yet continuous node movements still need attention. Considering the node mobility as a challenging task, a distributed routing scheme called Hop-by-Hop Dynamic Addressing Based (H2-DAB) routing protocol is proposed where every node in the network will be assigned a routable address quickly and efficiently without any explicit configuration or any dimensional location information. According to our best knowledge, H2-DAB is first addressing based routing approach for underwater wireless sensor networks (UWSNs) and not only has it helped to choose the routing path faster but also efficiently enables a recovery procedure in case of smooth forwarding failure. The proposed scheme provides an option where nodes is able to communicate without any centralized infrastructure, and a mechanism furthermore is available where nodes can come and leave the network without having any serious effect on the rest of the network. Moreover, another serious issue in UWSNs is that acoustic links are subject to high transmission power with high channel impairments that result in higher error rates and temporary path losses, which accordingly restrict the efficiency of these networks. The limited resources have made it difficult to design a protocol which is capable of maximizing the reliability of these networks. For this purpose, a Two-Hop Acknowledgement (2H-ACK) reliability model where two copies of the same data packet are maintained in the network without extra burden on the available resources is proposed. Simulation results show that H2-DAB can easily manage during the quick routing changes where node movements are very frequent yet it requires little or no overhead to efficiently complete its tasks.

ABSTRAK

Tesis ini membentangkan kajian mengenai masalah pengumpulan data di persekitaran air ganas. Selain kelewatan penyebaran panjang dan kebarangkalian ralat tinggi, pergerakan nod berterusan juga menyukarkan untuk menguruskan maklumat laluan semasa proses penghantaran data. Untuk mengatasi masalah kelewatan penyebaran besar dan kualiti link yang tidak boleh dipercayai, algoritma ramai telah dicadangkan dan sebahagian daripada mereka menyediakan penyelesaian terbaik untuk isu-isu ini, namun pergerakan nod berterusan masih memerlukan perhatian. Memandangkan pergerakan nod sebagai tugas yang mencabar, skim laluan diedarkan dipanggil Hop-by-Hop Dynamic Addressing Based(H2-DAB) routing protokol adalah dicadangkan di mana setiap nod dalam rangkaian akan diberi alamat routable dengan pantas dan cekap tanpa sebarang konfigurasi yang jelas atau mana-mana maklumat lokasi dimensi. Menurut pengetahuan terbaik kami, H2-DAB mula-mula menangani pendekatan berasaskan laluan untuk rangkaian sensor air wayarles (UWSNs) dan bukan sahaja telah ia membantu untuk memilih jalan laluan yang lebih cepat tetapi juga cekap membolehkan satu prosedur pemulihan dalam kes kegagalan penghantaran lancar. Skim yang dicadangkan memperuntukkan satu pilihan di mana nod dapat berkomunikasi tanpa sebarang infrastruktur terpusat, dan mekanisme tambahan pula boleh didapati di mana nod boleh datang dan meninggalkan rangkaian tanpa mempunyai apa-apa kesan serius terhadap negara lain di rangkaian. Selain itu, satu lagi isu serius dalam UWSNs adalah yang menghubungkan akustik adalah tertakluk kepada kuasa penghantaran yang tinggi dengan kecacatan saluran yang tinggi yang mengakibatkan kadar kesilapan yang lebih tinggi dan kerugian laluan sementara, yang sewajarnya menyekat kecekapan rangkaian ini. Sumber-sumber yang terhad telah menyukarkan untuk reka bentuk protokol yang boleh memaksimumkan kebolehpercayaan rangkaian ini. Untuk tujuan ini, Pengakuan Dua-Hop (2H-ACK) kebolehpercayaan model di mana dua salinan data paket sama dikekalkan dalam rangkaian tanpa beban tambahan pada sumber

yang ada adalah dicadangkan. Model ini membolehkan untuk menetapkan saiz paket yang optimum mengikut keadaan persekitaran. Keputusan simulasi menunjukkan bahawa H2-DAB boleh menguruskan semasa perubahan laluan pantas di mana pergerakan nod sangat kerap tetapi ia memerlukan atas sedikit atau tiada langsung dengan cekap menyelesaikan tugasnya.

In compliance with the terms of the Copyright Act 1987 and the IP Policy of the university, the copyright of this thesis has been reassigned by the author to the legal entity of the university,

Institute of Technology PETRONAS Sdn Bhd.

Due acknowledgement shall always be made of the use of any material contained in, or derived from, this thesis.

© Muhammad Ayaz, 2011
Institute of Technology PETRONAS Sdn Bhd
All rights reserved.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

2H-ACK	Two Hop Acknowledgment
AoA	Angle-of-Arrival
ACK	Acknowledgment
AUV	Autonomous Underwater Vehicle
BER	Bit Error Rate
CTS	Clear To Send
DET	Detachable Elevator Transceiver
DNR	Dive and Rise
DTN	Delay/Disruption Tolerant Network
ETX	Expected Transmission Count
FEC	Forward Error Correction
GPS	Global Positioning System
H2-DAB	Hop-by-Hop Dynamic Addressing
ICN	Intermittently Connected Network
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
MAC	Medium Access Control
MANET	Mobile Ad Hoc Network
MBS	Mobile Beacon and Sink
PER	Packet Error Rate
PSU	Practical Salinity Unit
QoS	Quality of Service

RF	Radio Frequency
RSSI	Receiver-signal-strength-index
RTS	Request To Send
RTT	Round Trip Time
SACK	Selective Acknowledge
SNR	Signal-to-Noise Ratio
TCP	Transmission Control Protocol
TDoA	Time-Difference-of Arrival
ToA	Time-of-Arrival
UAN	Underwater Acoustic Networks
UDP	User Datagram Protocol
UW-A	UnderWater Acoustic
UW-ASN	UnderWater Acoustic Sensor Network
UW-Sink	UnderWater Sink
USN	Underwater Sensor Network
WSS	WaterSide Security
UWSN	Underwater Wireless Sensor Network

CHAPTER 1

INTRODUCTION

The ocean is vast for covering around 140 million square miles and more than 70% of the earth surface, and half of the world's population is found within the 100 km of the coastal areas. Not only has it been a major source of nourishment production, but also with time taking a vital role for transportation, presence of natural resources, defensive and adventurous purposes. Even with all its importance to humanity, surprisingly some people know very little about water bodies of the Earth. Only less than 10% of the whole ocean volume has been investigated, while a large area still remains unexplored. With the increasing role of ocean in human life, discovering these largely unexplored areas has gained more importance during the last decades. At one side, traditional approaches used for underwater monitoring missions have several drawbacks and at the same time, these inhospitable environments are not feasible for human presence as unpredictable underwater activities, high water pressure and vast areas are major reasons for unmanned exploration. Due to these reasons, Underwater Wireless Sensor Networks (UWSNs) are lately attracting many researchers, in particular for those working on terrestrial sensor networks.

Sensor networks used for underwater communications are different in many aspects from traditional wired or even terrestrial sensor networks [1, 2]. Firstly, energy consumptions are different because some important applications require large amount of data, but very infrequently. Secondly, these networks usually work on a common task instead of representing independent users. The ultimate goal is to maximize the throughput rather than fairness among the nodes. Thirdly, for these networks, there is an important relationship among the link distance, number of hops and reliability. For energy concerns, packets over multiple short hops are preferred instead of long links, as multi-hop data deliveries have been proven to be more energy efficient for underwater networks than the single hop [3]. At the same time, it is

observed that packet routing over more numbers of hops ultimately degrades the end-to-end reliability function especially for the harsh underwater environment. Finally, most of the time, such networks are deployed by a single organization with economical hardware, causing strict interoperability with the existing standards to be not required.

Due to these reasons, UWSNs provide a platform that supports to review the existing structure of traditional communication protocols. The current research in UWSNs aims to meet the above criterion by introducing new design concepts, developing or improving existing protocols and building new applications.

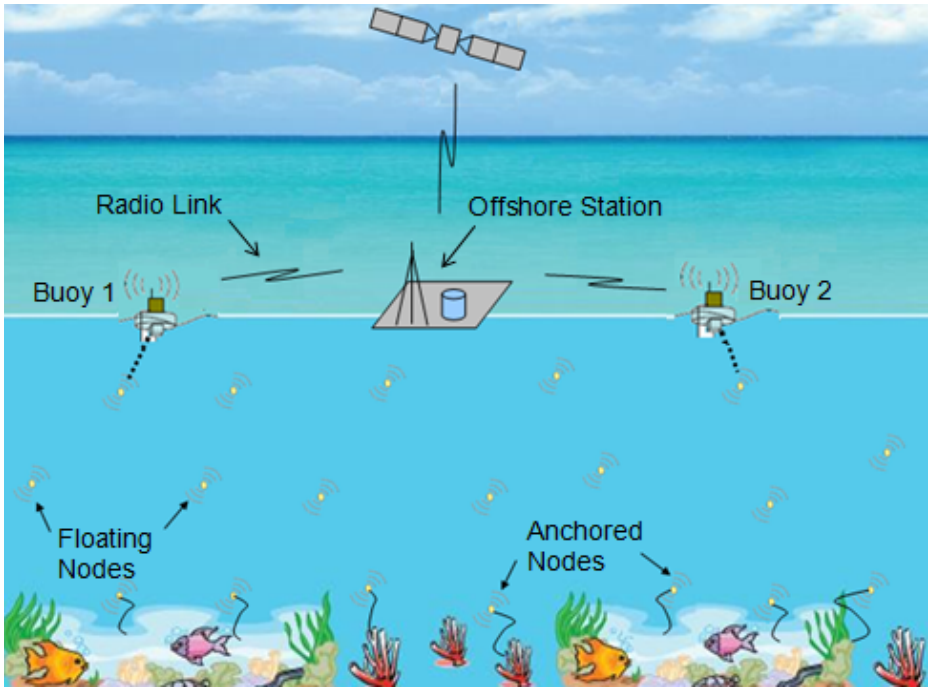

Figure 1.1 A general scenario of the mobile UWSN architecture

1.1 Motivation

A scalable UWSN provides a promising solution for efficiently discovering and observing the aqueous environments which operate under the following constraints:

- Underwater conditions are not suitable for human exploration. High water pressures, unpredictable underwater activities, and vast areas of water are major reasons for un-manned exploration.

- In traditional approaches, there is no any support for the interactive communications between the different communication ends. In most of the cases, the recorded data can only be retrieved at the end of mission, which can take several months, and any failure during the mission can lead all the collected data to the loss. Not only does underwater WSN support interactive communications but also data start to receive at processing center as soon as the network starts to work
- Localized exploration is more precise and useful than remote exploration now that underwater environmental conditions are typically localized at each place with a variable nature. Remote sensing technologies additionally may not be able to find appropriate knowledge about the events occurred in the unstable underwater environment.
- Traditional underwater exploration depends on either a single high cost underwater system or a small scale underwater network of small and low cost devices. But none of the existing technology is suitable to applications covering a large area. Enabling a scalable underwater sensor network technology is then to be something essential for exploring a vast underwater space.

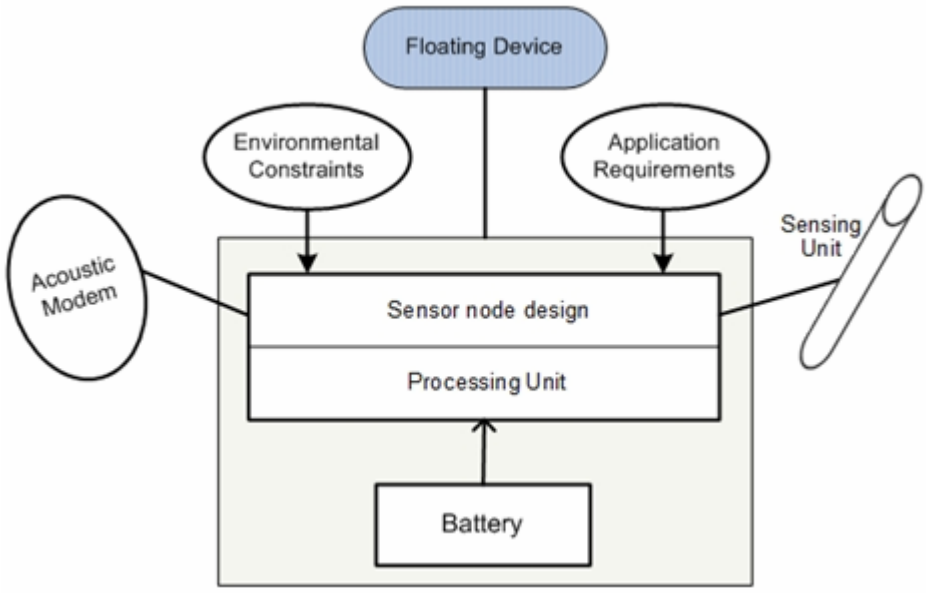

Figure 1.2 A possible system design for UWSN

1.2 Applications of Underwater Wireless Sensor Networks

Underwater Acoustic Networks (UANs) as a platform for oceanic research have gained much attention during the last decade and a strategy for the development of different potential applications is required. Monitoring the aquatic environment and dynamical changes of the ocean is becoming a complicated assignment. To preserve marine resources and to obtain a sustainable development, changes occurred in the marine environment have to be effectively monitored. The threat of climate changes and increased water-born activities may have great impacts on oceanic life and ecosystems and a rapid change in the marine environment may have great influence on terrestrial life and environment. All the discussed issues provide a base where underwater sensor networks can be used in a broad range of applications including as follows.

Undersea exploration: Geometry of the reservoir in deep water can be different from the familiar onshore even in shallow waters. During the last couple of years, the world's deepwater reserves have more than doubled and further this production is expected to substantially grow in the following few years. Not only can underwater sensor networks be helpful to detect new underwater oil/gas fields, exploration of other valuable minerals and to monitor these areas but also be essential to find out the routes for laying undersea cables.

Environmental monitoring: Monitoring marine pollution is dependent on advanced chemical analysis to detect the most dangerous contaminants. Underwater sensor networks can perform pollution detection, ocean wind monitoring, better weather predictions, identification of climate changes, understanding and predicting the effect of human activities on marine and examining the ecosystems and biological such as tracking of fishes or micro-organisms.

Ocean sampling networks: Networks of sensors and Autonomous Underwater Vehicles (AUV) can perform synoptic, cooperative adaptive sampling of the 3D coastal ocean environments.

Disaster prevention: Sensor networks that measure seismic activity from remote locations can provide tsunami warnings to coastal areas or study the effects of

submarine seaquake.

Distributed tactical surveillance: Autonomous Underwater Vehicles (AUVs) and fixed underwater sensors can collaboratively observe the areas for surveillance, targeting and intrusion detection.

Mine reconnaissance: Marine biology is characterized by many cumbersome methods for field research. Operation of multiple AUVs concurrently with acoustic and optical sensors can be used to execute quick environmental estimations and detect mine-like objects

Assisted navigation: Sensor can be used to locate hazards on the seabed, to find out dangerous rocks in shallow waters, safe locations, and to perform bathymetry profiling.

1.3 Problems and Challenges

When considering underwater wireless sensor networks, a due consideration must be given to the possible challenges that may be encountered in the subsurface environment. Continuous node movement and 3-d topology are major issues posed by the host conditions. Most of the times, sensor nodes are considered to be static but sensor nodes underwater in fact can move up to 1-3/msec due to different underwater activities. Further, radio communications are not suitable for deep water, accordingly requiring acoustic communications as a substitute. Due to this substitution, available propagation speed is five orders of magnitude less than the radio frequency as the characteristics of communication shifted from the speed of light to the speed of sound. Moreover, some of the underwater applications, including detection or rescue missions, tend to be ad hoc in nature; some requiring network deployment not only in short times, but also without any proper planning. In such circumstances, the routing protocols should be able to determine the node locations without any prior knowledge of the network. Moreover, the network also should be capable of reconfiguring itself with dynamic conditions in order to provide an efficient communication environment.

Other than these, some more challenges in these environments necessarily need to

be concerned. Some of the underwater applications, such as rescue and detection missions, can require deploying the network in short time, without any proper planning. In this scenario the routing protocols should be able to determine the node locations without any prior knowledge of the network. Additionaly, the network should be able to configure automatically and to be capable of reconfiguring itself dynamically to provide an efficient communications environment.

Cost is an important issue for Acoustic Networks [4]. A modem for acoustic communications currently costs around $2k. Even, underwater sensors can be more expensive than the modem itself. Supporting hardware such as underwater cable connectors also drive up costs as its price is around $100. Another reason that can increase the cost is the sophisticated constructions required in order to survive against harsh environment. The pressure increases by an additional atmosphere for every 10m of depth, so even a shallow-water (around 100m) instrument must be able to withstand 10 atmospheres, while deep-water instruments (around 4km) must be rated to at least 400 atmospheres. Significantly less expensive sensors, vehicles, and modems (500m-range acoustic and very short-range optical and radio) are being designed and built. These efforts may change the economics for dense underwater sensor networks in the near future.

Furthermore, a significant issue in selecting a system is to establish what the real range and data rate will be for a specific use. A system designed for deep-water may work poorly in shallow water or even when configured for too high data rate when reverberation is present in the environment [5]. For establishing the upper performance bound, manufacturer's specifications of maximum data rates are useful, but frequently unachievable, particularly in some challenging acoustic environments. The well funded users have resorted to purchase and test the multiple systems in specific environments to determine if the systems will meet their needs. An international effort to standardize the tests for acoustic communications systems is needed. However to undertake this effort is costly. In other word, it seems to be difficult for an impartial body to establish, while private organizations or government institutes, those who perform such comprehensive tests tend to not publish the results.

1.4 Research Questions

This research concerns with the variety of problems that affect the performance of underwater wireless sensor networks. Significant improvements are possible against different issues especially efficient data deliveries that can be made by revealing different questions posed by the underwater environments. Different authors have proposed various strategies to handle these issues and attempted to increase network efficiency by increasing the data delivery ratios and decreasing the propagation delay and network resource consumption at the same time. Based on the given problems and issues, the following research questions are separately formulated in this dissertation - aimed to achieve the goals of our research.

- Why do we need to replace currently available ocean monitoring systems with UWSN, what are the benefits actually?
- What are the basic differences between terrestrial WSN and Underwater WSN in terms of physical architecture and communication requirements?
- What problems do we have to face to port the existing tools available in conventional terrestrial WSN to UWSN.
- What are the transmission options available for UWSN and what are the advantages and disadvantages of each, and which system is best for our requirements?
- What are the deployment methods available for UWSN? During these deployments, how can an arrangement of sensor nodes be designed in a way that allows these nodes to collaborate with each other for efficient routing of the sensed data packets. How these methods can help to reduce the resource consumption?
- How sensor nodes can manage and organize their locations and positions in an unstable underwater environment?
- How can we minimize the effect of water currents with the help of routing decisions and how can these help to minimize the routing overheads?
- Different applications and network architectures pose new set of challenges, how can we fulfill the requirements of these applications and architectures?

- After considering all these questions, how can we balance performance of the whole network that adequately increase the network throughput, reduces end-to-end delays while at the same time considering the energy efficiency?

1.5 Research Objectives

The main objective of this thesis is to design and implement a dynamic addressing based routing protocol for underwater environment where scalability and resource efficiency becomes an essential requirement of the network. From literature review, it is proved that UWSNs are with some specific characteristics that are not found in the terrestrial sensor networks.

In our research work, the following points shall be investigated.

- Porting the common information and tools available in traditional WSN like basic routing ideas and trying to implement them for Underwater Wireless Sensor Networks
- Highlighting the future challenges that can be possible due to a new underwater volatile environment.
- Delay sensitive and delay tolerant applications here will be separate mechanisms in order to handle the connectivity issues. For delay tolerant applications, a mechanism to handle the loss of connectivity, instead of provoking immediate retransmissions will be developed.
- According to different conditions and applications, packet priorities will be dynamically calculated by adjusting their weights, so resource consumptions can be considered during the data forwarding according to the packets of different priorities.
- For reliability concerns, some acknowledgment mechanisms have to be defined, so the problem of packet losses and retransmissions can be properly coped with.
- By considering all these issues, algorithms; those give better routing result as well as provide strict or loose latency bounds for both delay tolerant and time critical applications will be developed.

- From these algorithms, a model with a complete solution in order to cope with these challenges according to the requirements of different underwater applications will be provided.

1.6 Contributions

It is a challenging task to find and maintain the routes for dynamic underwater environments with energy restriction and sudden topology changes due to some node failures. For these circumstances, many routing schemes have recently been proposed for UWSNs. Most of them require or assume special network setups and generally can be divided in two categories; (1) those requiring special network setups such as extra hardware as multiple types of sensor nodes are required in order to complete the task and (2) on the other hand, those being of geographical based routing schemes and requiring full dimensional location information of the network. In the interest of simplicity, most of the protocols of this type assume that every node in the network has already identified its own location and location of final destination. For comparison purpose, a short summary of some routing schemes is described in table 1.

Table 1.1 Summary of protocols which require special network setups

Algorithm	**Requirement/Assumption (s)**
DBR[6]	Every node should be equipped with a depth sensor.
Localization Scheme[7]	i) Special DET nodes are required and equipped with an elevator. ii) Some nodes require anchoring at different depths and locations in whole area.
Localization For USN[8]	i) All nodes must be clocked synchronized. ii) GPS communication and Time of Arrival (ToA) method required.
FBR[9]	Assuming that every node knows its own location.
VBF[10]	Assuming that full dimensional location information of whole network is available.

SBR-DLP[11]	Every node knows its own location information and pre-planned movement of destination.
REBAR[12]	Assuming that every node knows its own location and location of sink.
DFR [13]	All nodes know not only their own location but also the location of one hop neighbors and location of sink.
LASR[14]	i) Accurate timing required for synchronization and range estimation. ii) Network should consist of small number of nodes; adding new nodes will expand the protocol header overhead.
Multi-path Virtual Sink [15]	i) Two special types of nodes are required ii) Local sinks at different depths and locations are connected with each other via high speed links (RF link or Optical Fiber).
UW-HSN[16]	i) Every node should be equipped with both radio and acoustic modems. ii) Every node uses a mechanical module to emerge and submerge operation.
Resilient Routing[17]	i) Every sensor node is connected with a long wire which is anchored at the bottom. ii) Sensor should have an electronically controlled engine to adjust the length of the wire.

By considering these issues, this research concentrates on reliable data deliveries and proposes a novel routing protocol called **Hop-by-Hop Dynamic Addressing Based (H2-DAB)** for critical underwater monitoring missions. H2-DAB is scalable, robust and energy efficient and completes its task without making any assumptions as most of the other schemes do that of the same area.

Our proposed technique follows a multi-sink architecture in which surface buoys will be used to collect the data at the surface and some nodes will be anchored at the bottom. Remained nodes will be deployed at different levels from surface to bottom. Nodes near the surface sinks will have smaller addresses that will increase as the

nodes descend towards the bottom. These dynamic addresses will be assigned with the assistance of Hello packets; those that are generated by the surface sinks. Any node which collects the information will try to intensely deliver it towards the upper layer nodes. Packets already reaching any one of the sinks will be considered to be successfully delivered to the final destination as these buoys have the luxury of radio communications, where they can communicate with each other at higher bandwidths and lower propagation delays. For better resource consumption and to increase the reliability, this research suggests some special nodes called Courier nodes. These Courier nodes collect the data packets from lower layer nodes, especially from the nodes anchored at the bottom. After collecting, the Courier node will deliver these packets directly to the surface sinks. For performance evaluation, the proposed scheme in NS-2 is tested and also compared with DBR before drawing qualitative as well as quantities conclusions.

The main advantages of H2-DAB are as follows:

I. Proposed protocol completes its task not only without making any assumption but also without requiring any extra or specialized network equipment.
II. H2-DAB is highly adaptive to network dynamics causing the new nodes to be able to join or leave the network without any effects on the rest of the network.
III. Node movements with water currents can be easily coped with as address of any node will remain smaller from the addresses of lower nodes and larger from the upper nodes.
IV. The size of routing table is not affected by the network size as it will remain of the equal size and every node maintains a table of one entry even when the network consists of a large number of sensor nodes. In short, there is no need to maintain complex routing tables.
V. The proposed technique has no any problem of address exhaustion as addresses will remain of 2 digits per *HopID* and multiple nodes can use the same address without any problem during data deliveries.
VI. It will take the advantage of multi-sink architecture (For single sink, nodes near the sink entertain large amount of data packets, not only can it lead to the problem of congestions and data losses but also these nodes can die early due to frequent energy consumption)

After all these advantages, further reliability mechanism 2H-ACK is provided for H2-DAB in order to cope with the problem of node or packet losses and with a help of this reliability model, more precise results could be achieved then. The proposed reliability model could cope with the following issues

I. Guaranteeing data deliveries with two hop ACKs, especially the one proposed for highly dynamic environments such as underwater sensor networks.
II. Controlled or reduced congestion
III. Identical power consumption enables an increase of the sensor nodes life

1.7 Research Activities

To achieve mentioned goals, the research activation has been organized as follows:

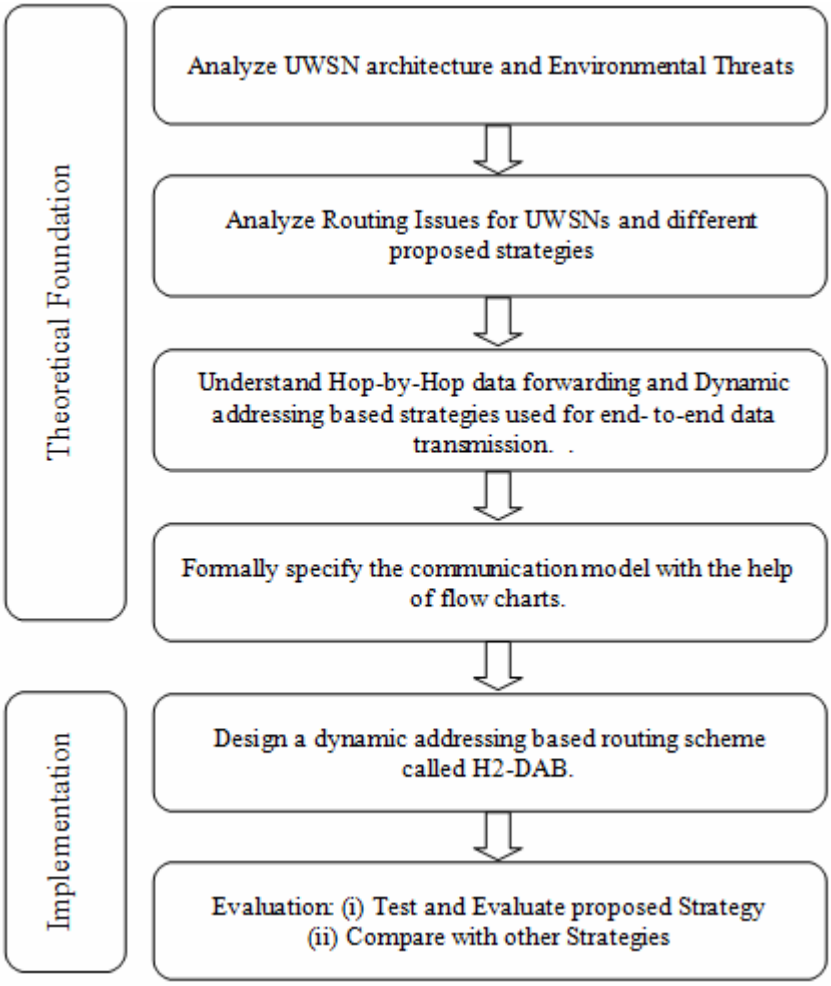

Figure 1.3 Research activities

1.8 Organization of the Thesis

This thesis is organized in 7 chapters and begins with a brief overview of this work. The remaining chapters are organized as follows.

In chapter 2, the overall background of underwater acoustic networks is discussed and several relevant fundamental key aspects and issues are investigated as well. Further, a detailed review, comparison and classification of existing protocols of same area, with their pros and cons are presented. Furthermore, open research issues are also discussed and possible solution options are outlined.

Chapter 3 deals with a novel dynamic addressing based routing technique and explains a complete procedure for address assigning and data packet forwarding. An analytical energy consumption model is also included first for static nodes (best case) and afterward complete mobile network (worst case) is considered. Later on, 2H-ACK reliability model is also included which is proposed particularly for dynamic environments such as underwater sensor networks.

Chapter 4 of this dissertation begins by presenting algorithms of H2-DAB either without courier nodes or with courier nodes and also includes the relevant flow charts.

In Chapter 5, experimental results obtained by the proposed scheme are presented. The propsed technique is evaluated by examining with different parameters such as the ones with and without courier nodes, and investigated different offer loads and different number of sink nodes, changing the interval life and node movements with varying speeds. Further, the performance of H2-DAB is varifired by comparing with two recently proposed routing techniques called VBF and DBR.

In Chapter 6, case studies of underwater wireless sensor networks are discussed and the areas where we can implement these networks are highlighted.

Finally, Chapter 7 concludes the work by summarizing the main contributions and findings of the study with some possible future directions.

CHAPTER 2

BACKGROUND AND RELATED WORK

Underwater Acoustic Networks (UANs) as a platform for oceanic research have gained much attention during the last decade and a strategy is required for the development of different potential applications. Monitoring the aquatic environment and dynamic changes of the ocean is not an uncomplicated assignment. To preserve marine resources and obtain a sustainable development, changes occurring in the marine environment have to be monitored effectively. The threat of climate changes and increased water-born activities may have great impacts on oceanic life and ecosystems. A rapid change in the marine environment may have great influence on terrestrial life and environment. Although, underwater acoustics has been studied from decades, still underwater networking and routing protocols are at infant stage as a research field.

This chapter reviews a wide range of UWSN literature. The purpose of this literature review is to attempt to identify previous work that could provide a good basis to establish the requirements for developing new routing techniques. The first section discusses an overview of the basics of acoustic communication. Section 2 presents deployment and network architecture related issues. Sections 3 and 4 give the idea of localization and reliability respectively. Section 5 provides some differences between traditional underwater acoustic networks (UAN) and UWSNs while section 6 presents the problems in existing terrestrial wireless sensor networks with a detailed comparison with characteristic of both types of networks. Then, sections 7 and 8 present several important routing schemes proposed to date for UWSN, and highlight the advantages and performance issues of each routing scheme with different classification and performance comparisons. Finally, section 9 covers the evaluation methods where different tools developed for this purpose are discussed.

2.1 Basics of Acoustic Communications

Acoustic signal is considered as the only feasible medium that works satisfactorily in underwater environments. Although couple of more options are availabe in the form of electromagnetic and optical waves, but underwater characteristics and sensor communication requirements have ruled them out. Considering electromagnetic wave, at high frequencies it has very limited communication range due to high attenuation and absorption effect, as measured less than 1 meter in fresh water [18]. Though propagation is acceptable with low frequencies, but at the cost of high transmission power and long antenna size. Recently, electromagnetic modems for underwater communication have been developed, however available technical details are vague [19]. It has been shown that, the absorption of electromagnetic signal in sea water is about $45\times\sqrt{f}$ dB/km, where f is frequency in Hertz [20]. While, the absorption of acoustic signal with the frequencies commonly used for underwater is lesser by three orders of magnitude.

Optical link, even though it is good for point to point communication especially in very clean water, but it is not good enough for distributed network structure due to its short range (less than 5m) [21]. Not only this, also a precise positioning is required for narrow beam optical transmitters. In short, it is not considered as a good choice for long distance underwater communications, particularly when the water is not so clean like shallow water.

On the other hand, acoustic signal is the only reliable and most suitable medium for low cast, ad hoc and densely deployed underwater sensor network. It provides the facility of omnidirectional transmission and distributed channel access with acceptable signal attenuation. Despite all the attractions (relative to electromagnetic and optical waves), underwater acoustic signal introduces a set of new communication challenge. The erroneous acoustic channel faces the problem of temporary path losses, high bit error rate, small bandwidth and large propagation delays. Path losses are not only due to transmission distance, but also depend on signal frequency. Severely limited bandwidth leads to low data rates, which again depend on both the communication range and frequency [22, 23]. Long range systems that operate over kilometers cannot exceed the bandwidth of more than few kHz. On the other hand, a

short range system operating over tens of meters can communicate with a bandwidth of more than a hundred kHz. Although, acoustic communications are classified in different categories in terms of range and bandwidth, but it can hardly exceed 40kb/sec at a range of 1 km.

Although the speed of sound is assumed to be constant in most of the situations, but actually it depends on water properties like temperature, salinity and pressure. Normally, the speed of sound is around 1500 m/s near the ocean surface which is 4 times faster than the speed of sound in air, but five orders of magnitude slower than the speed of light [24]. However, the speed of sound increases with the increase in any of these factors including temperature, depth and practical salinity unit (PSU). Approximately, temperature rise of 1°C, depth increase of every 1 km and increase of 1 PSU results to increase the speed of sound by 4 m/sec, 17 m/sec and 1.4 m/sec respectively. The routing schemes that consider these variations are expected to provide better results compared to those which assume uniform speed.

Table 2.1 Comparison of acoustic, EM and optical waves in seawater environments

	Acoustic	**Electromagnetic**	**Optical**
Nominal speed (m/s)	~ 1,500	~ 33,333,333	~ 33,333,333
Power Loss	> 0.1 dB/m/Hz	~28 dB/1km/100MHz	∞ turbidity
Bandwidth	~ kHz	~ MHz	~ 10-150 MHz
Frequency band	~ kHz	~ MHz	~1014-1015 Hz
Antenna size	~ 0.1 m	~ 0.5 m	~ 0.1 m
Antenna complexity	medium	high	medium
Effective range	~ km	~ 10 m	~ 10-100 m
Transmission range	~ 50m-5km	~ 1m-100m	~ 1m-100m
Major hurdles	bandwidth-limited, interference-limited	power-limited	environment-limited
Data rate	up to 100 kbps	up to 10 Mbps	up to 1Gbps

The table given above presents the summery of acoustic, electromagnetic and optical communications. After comparing all the characteristics presented here, it becomes clear that electromagnetic and optical communications are not suitable for underwater wireless sensor networks especially with densely deployed nodes. Not only this, but also it seems that currently available techniques have not made electromagnetic and optical communications practical for UWSNs. About acoustic communications, although these are applicable to UWSNs environments, related to networking design still a number of challenges left to be solved.

2.2 Deployment and Network Architecture

Underwater sensor networks (USNs) consist of a variable number of sensor nodes that are deployed to perform collaborative monitoring over a given volume. Similar to terrestrial sensor networks, for USNs it is essential to provide communication coverage in such a way that the whole monitoring area is covered by the sensor nodes, where every sensor node should be able to establish multihop paths in order to reach the surface sink. Many important deployment strategies for terrestrial sensor networks have been proposed such as [25-27], but deployment for USNs requires more attention due to its unique 3-d characteristics.

The work done in [1] is considered as the pioneering effort towards the deployment of sensor nodes for underwater environments. The authors have proposed two communication architectures i.e., *two-dimensional* and *three-dimensional*. In two-dimensional architecture, sensor nodes are anchored at the bottom where these can be organized in different clusters and are interconnected with one or multiple underwater gateways by means of acoustic links. The underwater gateways are responsible for relaying data from ocean bottom to surface sink. In three-dimensional architecture, sensor nodes float at different depth levels covering the entire volume region being monitored. . These nodes are attached with the surface buoys by means of wires and their lengths can be regulated in order to adjust the depth of the sensor nodes. They have used a purely geometric based approach to determine the required number of sensor nodes in order to cover the whole monitoring area. However, the minimum requirement of sensor nodes is shown in the order of hundreds or even thousands which is not feasible in terms of cost.

Further, a different approach for the same idea is proposed in [28] where sensor nodes are equipped with the same wire, but anchored at the bottom instead of anchoring to the surface buoys. These nodes are also equipped with a floating buoy that can be inflated by a pump, so it can move towards the surface and then back to its position. Although, this enhanced architecture helps to increase the reliability of network, but it makes the network more costly especially when we are interested in large monitoring areas.

In [29], a deployment strategy is proposed for water quality management in lakes in order to check the level of pollution due to the presence of toxins. The remotely sensed information is used to find the hot spots where relatively more sensors are deployed. In order to find the hot spots and those regions that do not require as many nodes, a mesh of triangle or rectangle is created. The sensing range of the nodes is defined by a probabilistic sensing model, and nodes are deployed in a weighted approach which depends on the density of the mesh. Although, the proposed technique can be a good solution for geographically irregular areas, however no information is available on how the sensor nodes can communicate with each other. Ultimately, it is assumed that sensors must be retrieved physically in order to get the sensed information.

Efficient deployment of multiple radio-enabled surface sinks can enhance the performance of network in many aspects. On the basis of this fact, some deployment techniques including [30, 31] are proposed, which tried to maximize the efficiency of the network by choosing proper locations for gateway placement. However, these methods are only for gateway deployment in 2-d ocean surface, but no information is available about the deployment of ordinary sensor nodes in 3-d areas.

2.3 Localization

For aquatic applications, it is important for every sensor node to know its current location information and synchronized timeliness with respect to other coordinating nodes. Due to GPS impracticality, UWSNs can rely on distributed GPS-free localization or time synchronization schemes known as cooperative localization. The

schemes related to this technique, especially for mobile networks, strongly depend on range and direction measurement processes. The commonly used approach for terrestrial networks of measuring Time-Difference-of Arrival (TDoA) between the RF and acoustic signals is not feasible due to failure of the RF signal under water [32]. Receiver-signal-strength-index (RSSI) schemes are highly vulnerable to acoustic interferences such as multi-path, doppler frequency spread and near-shore tide noise, so these cannot provide the accuracy for more than few meters. Next, the schemes like Angle-of-Arrival (AoA) require special devices for directional transmission and reception which can increase the cost of the network. Finally, the approaches like Time-of-Arrival (ToA) seem promising; they even provide accuracy at short ranges due to the acoustic mode of communication. Moreover, the acoustic signal is affected by the water currents, and variations in temperature and pressure, which requires sophisticated signal processing in order to overcome these error sources.

In some applications, sensed data become meaningless without time and location information. Localization is essential for data labeling while some time critical applications require timely information. In [8], the authors have combined both of these tasks in a localization framework called "*catch up or pass*", where these tasks mutually help each other. It benefits from the uncontrolled motion of underwater sensor nodes, where these nodes use the position and velocity information that help to decide whether to carry the data packet until they *catch up* with a sink or *pass* it to a faster or slower relay node.

The proposed framework uses a limited number of special nodes called Mobile Beacon and Sink (MBS). These MBS nodes have the ability to dive in and then return back vertically by modifying the density. The rest of the ordinary nodes stay under water at different locations, and can move with the water currents. MBSs periodically visit at different depths in order to localize underwater sensor nodes and collect data packets from them. These MBSs receive the coordinates from the GPS when they are floating on the surface, and upload the collected data to a ground station. At first stage, localization is done iteratively. Initially, MBSs get their location information via GPS. Then periodically broadcast their coordinates while diving to the deepest position of the network.

After receiving coordinate information from several beacons (at least four in this scenario) an ordinary node gets its location information. A localized node is considered as an *active* node, and can help in the localization process. It acts as a beacon and further distributes self coordinates. Every localization phase has a fixed duration which is announced in the localization message. The location and velocity of the MBS nodes, and the neighbors are learnt during the localization phase with the help of MBS message. This MBS message also includes time stamp field which helps to determine the distance via Time of Arrival (ToA). After completing localization round, next starts the routing phase. Sensor nodes that have data packets to be sent can select an MBS and forward these packets towards the sink.

During this localization and routing framework, the authors assumed that all the nodes are clock synchronized throughout the network. Such assumptions can be made for short term applications, but for the long term missions we require some additional mechanism in order to achieve synchronization. Moreover, they used ToA method when determining the distance between two nodes. Although, ToA is considered more promising than other techniques of the same type, but still it is not able to provide accuracy at long distances and is only feasible for short ranges.

Location information can be used to design network architecture and routing protocols. In [33], the authors proposed an idea of Dive and Rise (DNR) for positioning system. They used mobile DNR beacons to replace static anchor nodes. The major drawback of this DNR scheme is that it requires large number of expensive DNR beacons, which is further improved in [7]. In this scheme, they tried to decrease the requirement of mobile beacons by replacing them with four types of nodes, which are surface buoys, Detachable Elevator Transceivers (DETs), anchor nodes and ordinary sensor nodes. Surface buoys are assumed to be equipped with GPS facility. DETs are attached to the surface buoys, and remainly composed of an elevator and an acoustic transceiver. The elevator helps the DET to dive in vertically in the water and then rise up back towards the water surface. Acoustic transceiver is used to communicate with the anchored nodes especially for broadcasting coordinate messages. Further, many nodes are anchored at different positions and depth levels throughout the area of interest. These are special nodes as they have more energy, and

help to locate the ordinary nodes by communicating with DETs with the help of acoustic transceiver. The fourth type of nodes is ordinary sensor node, used for the sensing task and listening to coordinate messages broadcasted by the anchored nodes. When it receives more than 3 messages from different anchored nodes, then it will start to calculate its own position in the network.

After such specialized hardware deployments, this localization scheme has some assumptions. First of all, it assumes that all the sensor nodes are equipped with pressure sensor in order to provide depth position or z-coordinate information. Then, after requiring this entire infrastructure they assume that the network is static. Although, it can be enhanced for mobile network but still during their simulation study, mobility was not considered. Aside from the unfeasibility of these arrangements for long term applications, cost will become a major issue particularly for large area of interest.

2.4 Reliability

Reliability is a challenging factor for any sort of communication. For underwater environments, reliable delivery of sensed data to the surface sink is a challenging task as compared to forwarding the collected data to the control centre. In terrestrial sensor networks, multiple paths and packet redundancy are exploited in order to increase the reliability. For underwater sensor networks, many authors are also proposing schemes based on packet redundancy [15, 34], but for resource constraint underwater environments, techniques like this are not easily affordable. Usually, acknowledgements and retransmissions provide reliability by recovering lost data packets, however these efforts result in additional traffic and large end-to-end delays.

Transmission control protocol (TCP) is an end-to-end based technique and is considered as the most popular solution for reliable data communication. However, it has been shown that TCP and other congestion control mechanisms like this are highly problematic for wireless multihop networks [35, 36]. It requires 3-way handshake between the sender and sink before starting the actual data packet transmission due to its connection oriented nature. When we talk about UWSN, where

most of the time actual data might be a few bytes, the 3-way handshake process followed by TCP can be a burden for such a small volume of data. As we know, underwater wireless communications are based on multihop nature, where each of the inter-hop link is considered as error prone due to its pathetic acoustic channel, so the time required to establish a TCP connection might be very high especially when both of the end nodes are significant number of hops away from each other. Most of the TCP based techniques used for flow control, use a window based mechanism for this purpose. However, for acoustic channel the propagation time is larger than the transmission time which can provide a base for well known bandwidth*delay product problem [37]. Moreover, TCP assumes that only congestion is responsible for packet losses so it focuses mainly on those congestion control mechanisms that try to decrease the transmission rate. However, for UWSNs, the threatening conditions like error prone acoustic channel and node failure can also be the reason of packet losses; therefore it is not necessary to decrease the transmission rate in order to maintain throughput efficiency.

On the other hand, user datagram protocol (UDP) uses a simple transmission model without any hand-shaking procedure but it doesn't offer any flow or congestion control for reliability concerns. During congestion, it simply drops the data packets without providing any mechanism for recovering them. Besides, UDP also doesn't provide ACKs as it relies on some lower or upper layers when recovery is required for lost data packets. Obviously, approaches like UDP are not considered as a good choice for problematic underwater conditions.

Finally, rate based transport protocols also seem unsuitable for underwater acoustic communication [1]. Although, these protocols do not implement a window-based procedure, still their performance depends on the feedback control messages sent by the destination in order to adopt the dynamic transmission rate according to the packet losses, if they occur during the communication. These feedback messages help to regulate the transmission rate during different circumstances, which are not appropriate for underwater environment. Not only this, but in the absence of end-to-end paths, large propagation delays and delay variations also can create instability during these feedback control messages.

One of the main reasons that help to increase the congestion in the network is the convergent nature of the routing protocols, since all the sensor nodes forward their data packets towards a single sink. The degree of congestion increases as the data packets start to progress towards the destination; ultimately the nodes around the destination are seriously affected. For underwater sensor networks, many techniques like [6, 38] have been proposed in order to solve this problem as they suggest multiple sinks on surface. With limited resource availability like buffer space, if these congestions are not detected or some appropriate avoidance techniques are not implemented, a significant amount of data packets can be lost. These packet losses lead to retransmissions, which not only cause a significant amount of energy losses, but also it can lead to large end-to-end delays.

In order to address the challenges of UWSN for reliable data deliveries, a transport layer protocol called Segment Data Reliable Transport (SDRT) is proposed in [39]. SDRT uses Tornado codes in order to recover the errored data packets which help to reduce the retransmission. During the forwarding process, the data packets are transmitted block-by-block while for reliability concern each block is forwarded hop-by-hop. SDRT continues to send data packets inside a block, until it receives a successful acknowledgement, which causes energy wastage. In order to reduce this energy consumption, a window control mechanism is introduced where data packets are transmitted quickly within the window and remaining packets at slower rates. However, SDRT follows the hop-by-hop reliability while for unreliable underwater environments, where node failure or lost are common, this one hop reliability is not considered enough. Moreover, packet redundancy depends on error probability and this overhead will be high due to underwater error prone channel.

2.5 Difference between UANs and UWSNs

Mobile underwater wireless sensor networks are considered as a next step with respect to existing small scale Underwater Acoustic Networks (UANs). UANs are combination of nodes that collect the information using remote telemetry or assuming point to point communication. Current UWSNs has many differences with traditional UANs, some of them are as follows.

- **Scalability:** A UWSN is a scalable sensor network, which depends on coordinated networking among a large number of sensor nodes in order to complete its task of localized sensing and delivering this data. While, existing UAN is a small scale network which depends on data collecting strategies like remote telemetry or long-range signals remotely collect data. UANs are considered not only less précised due to the effect of environmental conditions but also very expensive when we use them for highly precision required applications. For UAN, due to sparse deployment, multi-access techniques are not required as point to point communication is assumed while in mobile sensor networks, nodes are densely deployed in order to achieve a better spatial coverage which requires a well designed multi-access and routing protocol.

- **Self-Organization:** Usually, in underwater acoustic networks nodes are fixed, while underwater sensor network considered as a self organizing network as here nodes can move and continue to redistribute due to underwater activities. Thus, these nodes should not only be able to adjust their buoyancy but also can move up and down according to measured data density. This is the reason; the protocols used for UAN, usually borrowed from terrestrial wireless sensor networks can not be used directly for mobile UWSN.

- **Localization:** In UANs, localization is not required because nodes are fixed in most of the cases, either anchored at sea bottom or attached to a surface buoy. While, for underwater sensor networks, some sort of localization is required as nodes can move continuously due to water currents. Now, determining the location information of mobile sensor nodes in aquatic environment is a challenging task. At one side, we have to face the limited communication capabilities of acoustic channel. On the same time, we have to consider immature localization accuracy.

2.6 Problems in Existing Terrestrial Routing Protocols

The existing routing protocols developed for terrestrial sensor networks are usually divided into two categories, Proactive and Reactive. However, both of these extremis has some problems like, Proactive or Table Driven protocols provoke a large signaling overhead in order to establish the routs, especially for the first time and every time when the topology is modified. So, due to the continuous nodes movements, topologies changes continuously. Then, if we talk about the Reactive scheme, it's no doubt that protocols belong to this category are more suitable for the dynamic environments, but they incur large delays and also require source initiated flooding of control packets in order to establish the paths. Plus experiments show that, they give better results when links are symmetrical throughout the network. But for underwater environments we know that, propagation delays are already high and mostly the links are asymmetrical, so the protocols of the both of these types are not suitable for the underwater networks.

Geographical Routing, where typically routes are not stored, is another promising option for the ground sensor networks. The protocols, use this approach establish the paths from source to destination by leveraging the localized information of the neighbors. Here each node decides about the next hope based on the information of its neighbor's location and the location of the destination. Its no doubt, in future this technique has much potential but only for ground based WSN where GPS easily available, because these protocols required accurate localized information, but for underwater networks, it's not easily possible. In fact, GPS uses the waves of 1.5 GHz band and the waves of this range can't propagate in the water environments.

For the wired networks, the routing problems are not complex, as topology is static, nodes are stationary as well as links are stable. Then, for the ad hoc networks nodes are mobile and links are not stable. An ad hoc or MANET can experience continues and random topological change due to the relative movement of the nodes. Forwarding data across such type of network is not an easy task. Further, a detailed comparison of different characteristics of the terrestrial and underwater sensor networks is provided in table 2.2.

Table 2.2 Comparison between terrestrial and underwater WSN

Terrestrial WSNs	**Underwater WSNs**
Most applications require dense deployment	Sparse deployment is preferred not only due to expensive equipment but also in order to cover large monitored areas.
Most of the network architectures assume that sensor nodes are stationary so different topologies can be applied.	Nodes continue to move 1-3m/s with water currents, so network cannot be viewed as a fixed topology [40].
A network with static nodes considered more stable especially in terms of communication links.	Routing messages from or to moving nodes is more challenging not only in terms of route optimization but also link stability becomes an important issue.
Generally considered more reliable due to a more matured understanding of the wireless link conditions evolved over years of R&D	Reliability is a major concern due to inhospitable conditions. Communication links face high bit error rate and temporary losses. Fault tolerant approaches are preferred.
Nodes are considered moving in 2D space even when they are deployed as ad hoc and as mobile sensor networks.	Nodes can move in a 3D volume without following any mobility pattern.
Usually the destination is fixed and seldom changes its location. In the event when destination is changes its location, still these movements are predefined.	Sinks or destinations are placed on water surface and can move with water current. Due to random water movement, predefined paths are difficult to or can't be followed.
Deployment affects the performance of the network. Generally, deployment is deterministic as nodes are placed manually so data is routed through pre-determined paths.	Non-uniform and random deployment is common. More self-configuring and self-organizing routing protocols are required to handle non-uniform deployment.
In most cases, nodes are assumed to be	Heterogeneous network is common.

homogenous throughout the network. Networks of this type provide better efficiency in most of the circumstances [41].	Inclusion of heterogeneous set of sensor nodes raises multiple technical issues related to data routing [42].
Radio waves are available; nodes can communicate with low propagation delays at speed of light (3×10^8 m/s) [43].	Acoustic waves replace radio waves (at speed of 1.5×10^3 m/s). Communication speed is decreased from speed of light to speed of sound, results in high propagation delays (five orders of magnitude) [2]. It can be problematic for real-time applications.
High data rate, normally in the order of MHz.	Low data rate, normally in the order of KHz. Hardly can exceed 40 kb/s at 1 km distance [44]. Moreover the attenuation of acoustic signal increases with frequency and range [45, 46].
Increased number of hops during the routing process.	Number of hops depends on depth of the monitoring area (normally 4-7 hops)
Low energy consumption [24].	High energy consumption due to longer distances (consequence of sparse nodes deployment) and complex signal processing. The power required to transmit may decay with powers greater than two of the distance [22].
Larger batteries can be used and can be replaced or recharged with ease.	Battery power is limited and usually cannot be easily replaced or recharged. The routing protocols should adopt a mechanism of power down during the communication and use minimum retransmission.
Nodes are less error prone and can continue to work for longer time.	Nodes are more error prone and can die (due to fouling or corrosion) or leave the

	working area. More reliable and self recovering routing algorithms are required.
Cooperative localization schemes like Time of Arrival (ToA) and Time-Difference-of Arrival (TDoA) are used for GPS free localization.	Techniques like TDoA are not feasible due to unavailability of accurate synchronization in under water [32].
Schemes like Receiver-signal strength-index (RSSI) can be used for cooperative localization.	RSSI is highly vulnerable to acoustic interferences such as multi-path, doppler frequency spread and near-shore tide noise, and cannot provide accuracy for more than few meters.
Automatic Repeat Request (ARQ) techniques are used for the error recovery and packet loss detections.	ARQ techniques are inefficient due to large propagation delays, as retransmissions incur excessive latency as well as signalling overheads [47].
Forward Error Correction (FEC) techniques are used to increase the robustness against errors.	FEC is not easily affordable due to redundant bits at extremely small bandwidth of acoustic communication.
GPS waves use 1.5 GHz band. For terrestrial sensor networks these frequencies are supported and GPS facility can be used for localization purpose.	Geographical routing is not supported as such high frequencies bands are impractical for UWSNs [48]. Ultimately, have to rely on distributed GPS-free localization or time synchronization schemes known as cooperative localization.

2.7 Related Work

The research in acoustic channel is not new, as three decades earlier, researchers have started to focus their interest in this area. Numerous review papers including [24, 49-51] are available, where the authors have examined the acoustic and underwater

communications. Many others like [1, 32, 52, 53] have addressed the challenges and issues posed by underwater environments, and proposed their solutions as well. Further, some authors have discussed energy efficiency and analysis [48, 54], deployment [55], potential applications [2, 56], network coding schemes [57], and multiple access techniques [58] but to the best of our knowledge, no review work is available where the routing protocols and networking issues of UWSN are classified and discussed thoroughly. Considering the importance of routing in UWSNs when a significant number of routing protocols are available, a comprehensive survey becomes necessary at this stage. The current effort in describing and categorizing the different approaches proposed recently is a step towards network layer and its related factors. The purpose of this study is to provide a detailed view of these routing schemes and to identify some research issues that can be further pursued.

Underwater Sensor Networks are attracting the attention of industry and academia as well [2, 22, 32]. At one side, it can enable a wide range of aquatic applications, and on the same time, adverse environmental conditions create a range of challenges for underwater communication and networking. The node mobility and sparse deployment can create problem for underwater sensor networks. Due to the continuous node movements with water currents, there may not be a persistent route from a source to a destination. That's why; an underwater sensor network can be viewed as a partially connected network, and the traditional routing protocols developed for terrestrial sensor networks, usually are not practical for such environment. Due to this intermittent connectivity, packets can be dropped when no routes are available to reach the destination.

2.7.1 Location-based Routing Protocols

Localization has been widely explored for terrestrial wireless sensor networks and many schemes have been proposed so far. In general, protocols of these types are classified into two categories: range based and range free schemes. Protocols belong to range based schemes like [59, 60] establish the paths from source to destination by leveraging the localized information of the neighbors. Here each node decides about the next hope based on the information of its neighbour's location and the location of

the destination. On the other hand, protocols of range free scheme like [61, 62]makes no assumption about the availability of range information. Although, range based schemes are proved more promising [63] but they need additional hardware in order to measure the distance which ultimately increase the cost of the network, while range free schemes do not require such additional equipment but can only provide coarse position estimation. Now when we talk about UWSN's; managing the location information of mobile sensor nodes is more crucial. Localization is at beginning for these environments as only a limited number of schemes like [64-66] have been proposed and most of these schemes are designed for small scale networks (usually tens of sensor nodes). In this section, we discussed the routing protocols that require partial or complete localized information in order to complete the task of routing the data packets from source to surface sinks.

2.7.1.1 Vector Based Forwarding (VBF)

Continuous node movements require frequent maintenance and recovery of routing paths, which can even more expensive in 3-d volume. In order to handle this issue, a position based routing approach called VBF has been proposed in [10] . For this, state information of the sensor nodes is not required since only a small number of nodes are involved during the packet forwarding. Data packets are forwarded along redundant and interleaved paths from the source to sink, which helps to handle the problem of packet losses and node failures. It is assumed that every node already knows its location, and each packet carries the location of all the nodes involved including the source, forwarding nodes and final destination. Here, the idea of a vector like a virtual routing pipe is proposed and all the packets are forwarded through this pipe from the source to the destination. Only the nodes closer to this pipe or "vector" from source to destination, can forward the messages. By using this idea, not only the network traffic can be reduced significantly, but also it is easy to manage the dynamic topology.

VBF has some serious problems. Firstly, the use of a virtual routing pipe from source to destination, as the creation of such pipe can affect the routing efficiency of the network with different node densities. In some areas, if nodes are much sparsely

deployed or become sparser due to some movements, then it is possible that very few or even no node will lie within that virtual pipe which is responsible for the data forwarding; even it is possible that some paths may exist outside the pipe. Ultimately, this will result in small data deliveries in sparse areas. Secondly, VBF is very sensitive about the routing pipe radius threshold, and this threshold can affect the routing performance significantly; such feature may not be desirable in the real protocol developments. Moreover, some nodes along the routing pipe are used again and again in order to forward the data packets from concrete sources to the destination which can exhaust their battery power. Other than these issues, VBF has much communication overhead due to its 3-way handshake nature; while during this, it doesn't consider the link quality.

In order to increase the robustness and overcome these problems, an enhanced version of VBF called Hop-by-Hop Vector-Based Forwarding (HH-VBF) has been proposed by [67]. They use the same concept of virtual routing pipe as used by VBF, but instead of using a single pipe from source to destination, HH-VBF defines per hop virtual pipe for each forwarder. In such way, every intermediate node makes decision about the pipe direction based on its current location. By doing so, even when a small number of nodes is available in the neighborhood, HH-VBF can still find a data delivery path, as long as a single node is available in the forwarding path within the communication range. Although, simulation results show that HH-VBF significantly produces better results for packet delivery ratio especially in sparse areas compared to VBF, but still it has inherent problem of routing pipe radius threshold, which can effect its performance. Additionally, due to its hop-by-hop nature, HH-VBF produces much more signaling overhead compared to VBF.

2.7.1.2 Focused Beam Routing (FBR)

Without any prior location information of nodes, a large number of broadcast queries can burden the network which may result in reducing the overall expected throughput. In order to reduce such unnecessary flooding, [9] presented Focused Bream Routing (FBR) protocol for acoustic networks. Their routing technique assumes that, every node in the network has its own location information, and every source node knows

about the location of the final destination. Other than this information, the location of intermediate nodes is not required. Routes are established dynamically during the traversing of data packet for its destination, and the decision about the next hop is made at each step on the path after the appropriate nodes have proposed themselves.

Figure 2.1 explains the data forwarding method used in FBR. Node A has a data packet that needs to be sent the destination node D. To do so, node A multicast a request to send (RTS) packet to its neighboring nodes. This RTS packet contains the location of source (A) and the final destination (D). Initially, this multicast action will be performed at the lowest power level which can be increased if no node is found as the next hop in this communication range. For this, they define a finite number of power levels, *P1* through P_N, that can be increased only if necessary. Now all the nodes that receive this multicast RTS will calculate their current location relative to the line AD. After calculating, those nodes that lie within a cone of angle $\pm\ \theta/2$ emanating from the transmitter towards the final destination are considered as the next hop candidates. After calculating this angle, if a node determines that it is within the transmitting cone, it will reply to the RTS.

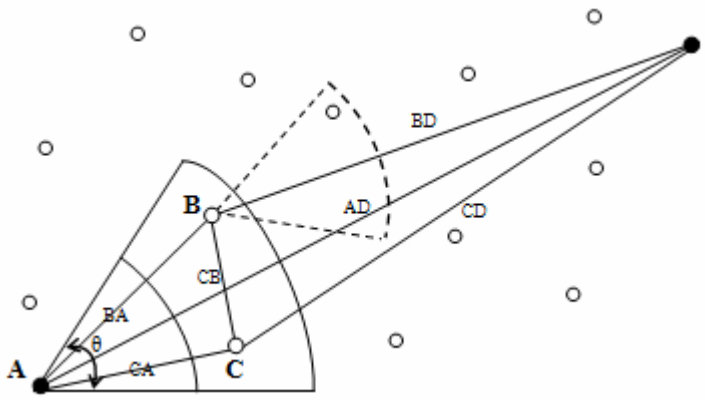

Figure 2.1 Illustration of the FBR routing protocol: nodes within the transmitter's cone θ are candidate relays [9].

However, the approach followed by FBR might have some performance problems. First of all, if nodes become sparse due to water movements, then it is possible that no node will lie within that forwarding cone of angle. Also, it might be possible that

some nodes, which are available as candidates for the next hop, exist outside this forwarding area. In such cases, when it is unable to find the next relay node within this transmitting cone, it needs to rebroadcast the RTS every time which ultimately increase the communication overhead, consequently affecting data deliveries in those sparse areas. Secondly, it assumes that the sink is fixed and its location is already known, which also reduces the flexibility of network.

2.7.1.3 A Reliable and Energy Balanced Routing Algorithm (REBAR)

It is common analysis that, water movements make the underwater environment more dynamic, but [12] considered node mobility as a positive factor which can be helpful to balance energy depletion in the network. The provided reason is that, when nodes move then nodes start to alternate around the sink which brings an effect of balance in energy consumption in the whole network. They tried to solve the problem of network partitioning by altering the node positions as nodes near the sink are prone to die much earlier due to their frequent involvement in the routing process. Their idea looks similar to [9] and [10] where they assumes that every node knows location of itself and location of sink, but they designed an adaptive scheme by defining data propagation range in order to balance the energy consumption throughout the network. Since, network wide broadcast results in high energy consumption, so here nodes broadcast in a specific domain between source and sink by using geographic information. Particularly, different sensor nodes have different communication radius depending on the distance between the nodes and sink. Nodes nearer to the sink are set to smaller values in order to reduce the chance of being involved in the routing process, which helps to balance the energy consumption among all the sensor nodes. According to their network model, all the sensor nodes are randomly deployed in an underwater hemisphere as shown in figure 2.2. The sink is stationary and fixed at the center of the surface. All the sensor nodes are assigned a unique ID and have a fixed range. It is assumed that every node knows its location and the location of sink through multi-hop routing. They also assume a data logging application where sensed data i are sent towards the sink at a certain rate.

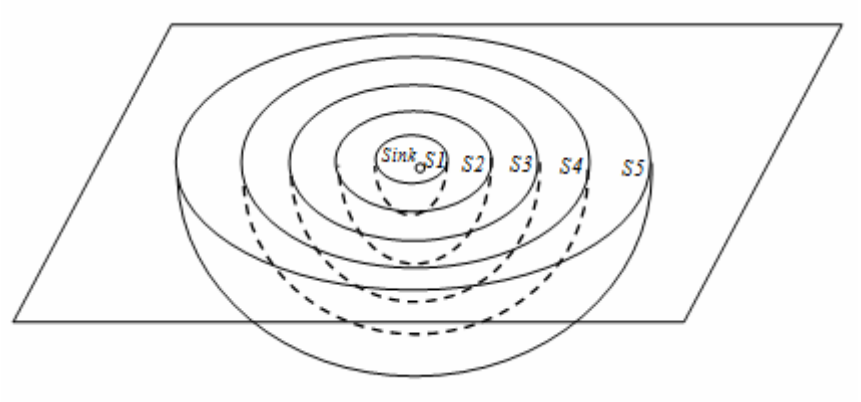

Figure 2.2 The sphere energy depletion model in REBAR [12]

However, the idea of altering node positions as in REBAR has a serious problem. At one side they advocate node movements as a positive sign; as simulation results show that with static nodes, delivery ratios are smaller and they start to increase with the increase of node movements. Due to some assumptions, they have made like at the start the nodes have the location information of their current position and final destination. For the simulation, they considered the node movements from 0 to 4 m/sec, and according to this phenomenon the delivery ratios should continue to increase when movements are more than 4 m/sec. In practical, these node movements are not always helpful, but it can create problem as well. Besides making the network sparser, large movements also could affect network performance since nodes would be required to update their location more frequently. Furthermore, it is also assumed that these movements are completely dynamic in terms of directions, both vertically and horizontally. In such a movement, a bottom node will move to the surface and then it will move back to the bottom. Again, in a real scenario, that might not be possible as only horizontal movements are common in the range of 2-3 m/sec while vertically, only small fluctuations are shown [32]. Moreover, the available simulation results have been focused only on delivery ratios and energy consumption with different node speeds, but have not provided any information about the end to end delays which can vary according to different node movements.

2.7.1.4 Directional Flooding-Based Routing (DFR)

For UWSNs, the path establishment requires much overhead in the form of control messages. Moreover, the dynamic conditions and high packet loss degrade reliability, which results in more retransmissions. Existing routing protocols proposed to improve the reliability, did not consider the link quality. That's why there is no guarantee about the data delivery especially when a link is error prone. In order to increase the reliability, [13] proposed Directional Flooding-Based Routing (DFR) protocol. DFR, basically, is a packet flooding technique which helps to increase the reliability. It is assumed that, every node knows about its location, location of one hop neighbors and final destination. Limited number of sensor nodes takes part in this process for a specific packet in order to prevent the flooding over the whole network, and forwarding nodes are decided according to the link quality. In addition, DFR addresses the void problem by allowing at least one node to participate in data forwarding process.

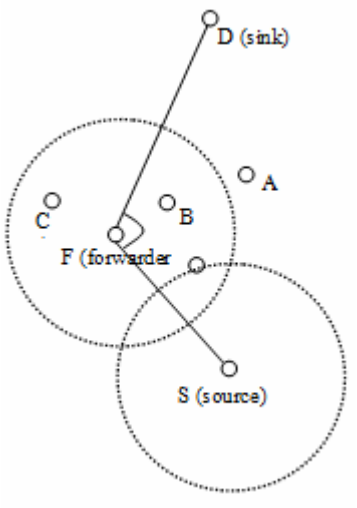

Figure 2.3 An example of a packet transmission in DFR [13]

As shown in figure 2.3, the flooding zone is decided by the angle between FS and FD; where F is the packet receiving node, while S and D present the source and destination node, respectively. After receiving a data packet, F determines

dynamically the packet forwarding by comparing SFD with a criterion angle for flooding, called BASE_ANGLE which is included in the received packet. In order to handle the high and dynamic packet error rate, BASE_ANGLE is adjusted in a hop-by-hop fashion according to the link quality, which helps to find a flooding zone dynamically. That is, the better the link quality is, the smaller the flooding zone is.

The performance of DFR depends on the number of nodes chosen as the next hop after flooding the data packet. Although, the problem of void region is addressed by making sure that at least one node must participate in this process. While, in areas where the link quality is not good then multiple nodes can forward the same data packet; so more and more nodes will join the flooding of the same data packet which ultimately increase the consumption of critical network resources. Secondly, they have controlled the void problem by selecting at least one node to forward the data packet towards the sink. However, when the sending node cannot find a next hop closer to the sink, the void problem would still be encountered as no mechanism is available for sending the data packet in the reverse direction.

2.7.1.5 Sector-based Routing with Destination Location Prediction (SBR-DLP)

Recently, several location based routing techniques have been proposed and it is said that they could achieve energy efficiency by decreasing the network overhead. Most of them assume that the destination is fixed and its location is already known to all the nodes throughout the network. This assumption may not be suitable for fully mobile networks. [11] proposed a routing algorithm called SBR-DLP which helps to route a data packet in a fully mobile underwater acoustic network ,where not only intermediate nodes but destination can be mobile as well.

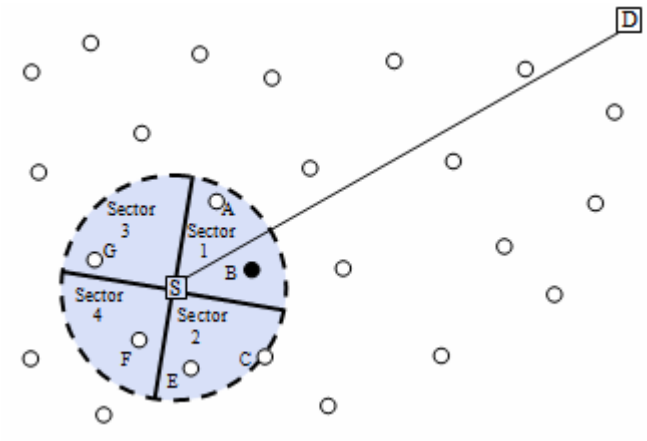

Figure 2.4 Forwarder selection at the sender in SBR-DLP [11]

The SBR-DLP is a location based routing algorithm where sensor nodes do not need to carry neighbor information or network topology. However, it is assumed that every node knows its own location information and pre-planned movement of destination nodes. Data packets are forwarded to the destination in a hop-by-hop fashion instead of finding end-to-end path in order to avoid flooding. As shown in figure 2.4, a node S has a data packet that needs to be sent to destination D. In order to do that, it will try to find its next hop by broadcasting a *Chk_Ngb* packet which includes its current position and packet ID. The neighbor node that receives *Chk_Ngb* will check whether it is nearer to the destination node D than the distance between nodes S and D. The nodes that meet this condition will reply to node S by sending a *Chk_Ngb_Reply* packet. This method is further depicted in table 1. SBR-DLP is a location based routing protocol such as [10] but it is different in many aspects from both of them. Firstly, instead of allowing all the candidate nodes to decide about the packet forwarding, in SBR-DLP the sender node decides which node will be the next hop using the information received from the candidate nodes. This solves the problem of having multiple nodes acting as relay nodes.

Table 2.3 How node S picks its next relay node

Sector	Candidates	Distance to D	After Filtering
1	A, B	500, 480	A, B
2	C	550	
3	-	-	
4	-	-	
		Next relay node	B

SBR-DLP handles the issue of destination mobility by assuming that pre-planned movements are completely known to all the sensor nodes before deploying them. However, this assumption has two issues. First, it reduces the flexibility of network; after launching the network it is not possible to change the position or location of destination nodes. Secondly, it is important to note that water currents can deviate the destination node from its scheduled movements.

2.7.1.6 Location-Aware Source Routing (LASR)

Dynamic Source Routing [68] is a well known routing protocol originally proposed for the MANET, but suffers from high latency in the underwater acoustic environment. In these conditions, the topology rate of change is very high compared to acoustic latency. Hence, topology continues to change more quickly such that DSR can adapt. In order to solve this problem without loosing the experience of DSR, [69] proposed LASR, a modification of DSR. LASR uses two techniques in order to handle the high latency of acoustic channel, first link quality metric and second is location awareness. DSR depends only the shortest path metric which leads to poor performance in highly mobile networks. LASR replaces this shortest path metric with expected transmission count (ETX) where link quality metric provides more-informed decisions, thus giving better routes through the network. Location awareness can be achieved from the incoming transmissions as an aid for estimating local network topology. Topology prediction uses a tracking system to predict the current location

of other vehicles in the network based on one way and range only measurements, while all the explicit information of the network including the routes and topology information is passed on in the protocol header.

After all these modifications, still LASR depends on source routing technique inherited from DSR. Therefore, as the hop count between source and destination increases, the packet header continues to increase as well. The increasing header size leads to overhead for acoustic communication with a narrow bandwidth. Secondly, it uses Expected Transmission Count (ETX) as a link quality metric, for which it assumes that links are symmetrical and are with same link quality in both directions, which is not easily possible for underwater acoustic communication

2.7.2 Hierarchical-based Routing Protocols

Hierarchical routing is the procedure of arranging sensor nodes in a hierarchical manner where nodes with different processing capabilities are used at different network locations. These arrangements can be physical as well as logical. Dividing the network in different groups called cluster is the common type of hierarchal routing where a head node is assigned to each group which is responsible for managing the whole cluster. The network is divided according to the location information or traffic burden and efficiency of the network is based on, how effectively the network is divided. Generally, hierarchical schemes are considered to be superior to flat schemes especially in terms of resource consumption. Routing protocols belongs to this scheme are further divided in two categories called dynamic and static. For dynamic hierarchal routing, there exists a temporary framework only for a single or few transmissions. For example, when we talk about cluster network then normally every node acts as a cluster head as well as can member of different clusters. For UWSNs, this type of routing is more common as in [70-72] due to unavailability of any fixed network infrastructure. On the other hand, in static hierarchical routing, there exists almost a fixed framework for multiple data transmissions while, for UWSNs, it’s not commonly used and only a few techniques like [15, 73] are belong to this. Although, hierarchical routing protocols have proved to have considerable savings in total energy consumption but still there are several key attributes that can affect their

performance. The designers must consider these issues particular for the dynamic underwater environments including clustering cost, synchronization, data aggregation, hierarchy repair mechanism etc. In short, every scheme has predominance depending on the network environment and requirements of different applications.

2.7.2.1 Distributed Underwater Clustering Scheme (DUCS)

Energy efficiency is a major concern for UWSNs because sensor nodes have batteries of limited power which are hard to replace, or recharge in such environments. It is a fundamental problem to design a scalable and energy efficient routing protocol for these networks. [70] presented a distributed energy aware and random node mobility supported routing protocol called Distributed Underwater Clustering Scheme (DUCS) for long term but non-time critical applications.

DUCS is an adaptive self-organizing protocol where the whole network is divided into clusters using a distributed algorithm. Sensor nodes are organized into local clusters where one node is selected as a cluster head for each. All the remaining nodes (non-cluster heads) transmit the data packets to the respective cluster heads. This transmission must be single hop. After receiving the data packets from all the cluster members, cluster head performs signal processing function like aggregation on the received data, and transmits them towards the sink using multihop routing through other cluster heads. Cluster heads are responsible for coordinating their cluster members (intra-cluster coordination), and communication among clusters (inter-cluster communication). The selection of cluster head is completed through a randomized rotation among different nodes within a cluster in order to avoid fast draining of the battery from the specific sensor node. DUCS completes its operation in two rounds. The first round is called *set-up*, where network is divided into clusters, and in the second round which is called *network operation*, transfer of data packets is completed. During the second round, several frames are transmitted to each cluster head where every frame is composed of a series of data messages that the ordinary sensor nodes send to the cluster head with a schedule. Simulation results have shown that DUCS not only achieves high packet delivery ratio, but also considerably reduces the network overhead and continues to increase throughput consequently.

Although, DUCS is simple and energy efficient, but it has a couple of performance issues. Firstly, node movements due to water currents can affect the structure of clusters which consequently decreases the cluster life. Frequent division of sectors can be a burden on the network as the *set-up* phase is repeated many times. Secondly, during the *network operation* phase, a cluster head can transmit its collected data towards another cluster head only. Again, water currents can move two cluster head nodes away, where they cannot communicate directly even a few non-cluster head nodes are available between them.

2.7.2.2 Location-Based Clustering Algorithm for Data Gathering (LCAD)

Data transmission phase is the main source of energy consumption for a sensor node. Dissipation of energy during the data transmission is proportional to the distance between the sender and receiver. As discussed earlier, another problem with the multi-hop approach is that sensor nodes around the sink process a large number of data packets which rapidly drain their energy. In order to solve both of these problems, [71] suggested a cluster based architecture for 3-dimensional underwater sensor networks. Here, sensor nodes are deployed in the whole area of interest at fixed relative depths from each other. These sensor nodes at each tier are organized in clusters with multiple cluster heads. They suggest an algorithm for the cluster head selection at each cluster according to the node position in the network. Horizontal acoustic links are used for intra cluster communication. For energy concern, the length of this horizontal acoustic link is restricted to a maximum of 500 m as it has been shown that the performance of acoustic link can be optimal at this communication distance [74].

In the proposed architecture, the entire network is divided into 3-dimensional grids where each grid is set approximately to 30m×40m×500m. The entire communication process is completed in three phases: (i) Setting up phase, where the cluster head is selected. (ii) Data gathering phase, where data are sent to the cluster head by the nodes in the same cluster. (iii) Transmission phase, where data gathered by the cluster heads are delivered to the base station with the help of Autonomous Underwater Vehicles (AUVs). About cluster head (ch-node), some of the sensor

nodes in every cluster have additional resources like memory and energy, and such nodes can qualify as ch-head. Having multiple ch-nodes increase not only the reliability, but also load balancing in the network. These ch-nodes are located approximately at the centre of the grid which helps to communicate with maximum number of ordinary sensor nodes. These grids are organized just like the cells in a cellular network.

AUVs are used as data mules for collecting data packets from these cluster heads instead of from every sensor node in the network. As it has been proven that acoustic link is not suggested for distances of more than 500m, so the required number of tiers depends on the average depths of oceans. For the best results, they advocate a dense deployment of sensor nodes at the lower tiers and sparser distribution at the higher tiers.

Nonetheless, the proposed protocol seems to have some serious performance issues. The performance of LCAD depends on grid structure specially the position of ch-node inside it. For terrestrial sensor networks, such type of structure is easily possible. However, for underwater environments where node movements are frequent, the assumption of such grid structure is not so simple as nodes can come and leave different grids frequently. For performance analysis, they evaluated the performance of LCAD in terms of network lifetime but have not provided any information about the node movements.

2.7.2.3 Distributed Minimum-Cost Clustering Protocol (MCCP)

In LEACH [75], a protocol proposed for terrestrial sensor networks, clusters are formed with optimal number of cluster heads by using the prior knowledge of uniform node distribution. However, due to continuous movement of ocean current, usually node deployment becomes non-uniform which makes LEACH unsuitable for these environments. HEED [76] solves this problem where clusters are formed without assuming a uniform distribution of the sensor nodes. Although, a cluster head rotation scheme is also implemented, still the traffic loads in different areas remain unbalanced. Moreover, both of these are based on cluster-head-centric scheme, in

which the cluster head is selected first followed by each of the non-cluster-head node joining its nearest cluster. In order to handle these problems, a distributed minimum-cost clustering protocol (MCCP) is proposed in [77] with an objective to improve energy efficiency and prolong the network life.

The proposed scheme uses a cluster based approach where clusters are formed by computing the following three parameters: total energy required by the cluster members for sending data to the cluster head, the residual energy of the cluster head and its entire members, and relative location of the cluster head and underwater-sink (uw-sink). To solve the problem, a centralized algorithm MCCA (minimum-cost clustering algorithm) is proposed where clusters are selected using a centralized approach. The MCCA is further extended to a distributed approach called MCCP. In this approach, initially all the sensor nodes are candidates for cluster head as well as the cluster member. Every candidate constructs its neighbor set and uncovered neighbor set in order to form a cluster. The average cost of that particular cluster is calculated and broadcasted to all the candidates within its 2-hop range with its cluster-head ID. After receiving this cost, every candidate node will compare with its own calculated cost. If it has minimum average cost, then it becomes a cluster-head and advertises an *INVITE* message to other cluster nodes to become its cluster member, otherwise it sends a JOIN message to the specific cluster head. Finally, all the nominated clusters define a TDMA schedule and forwarded it to the respective cluster members.

MCCP has many advantages as it avoids the formation of hot spots around the uw-sink by generating more cluster heads which helps to balance the traffic load. The number of cluster members depends on the cluster-head and uw-sink locations, which mean clusters closer to uw-sink will have less cluster members. Further, it has the ability to balance the traffic load by re-clustering the sensor nodes periodically. However, it does not support multi-hop routing and for this it depends on some other scheme. Secondly, the period for re-clustering the network is defined in the range of days or months. For underwater environments, nodes are in continuous movements from 2-3m/sec (3-5 km/h) [24]. Due to that nodes can leave and enter different clusters during such long periods which ultimately affect the cluster efficiency.

2.7.2.4 Pressure Routing for Underwater Sensor Networks (HydroCast)

For UWSNs, geographic routing is preferable due to its stateless nature. However, geographic routing requires distributed localization of mobile sensor nodes which not only can be expensive in terms of energy, but also can take long time to converge. In order to provide an alternate of geographic routing, [78] presented HydroCast, a hydraulic pressure based routing protocol. HydroCast uses any cast routing by exploiting the measured pressure levels in order to forward the data packets towards surface buoys. The proposed hydraulic pressure based protocol is stateless and completes its task without requiring expensive distributed localization.

The basic idea of HydroCast is similar to DBR where routing decisions are made after comparing the local pressure or depth information, such that data packets are greedily forwarded towards a node with the lowest pressure level among the neighbor nodes. DBR faces a serious problem of local maximum when a data forwarding node cannot find the next hop with lower depth among its neighbor nodes. In such void regions, it does not provide any solution to handle such situation. While in HydroCast scheme, each local maximum node maintains a recovery route towards a neighboring node with higher depth than itself. After one or several forwardings through local maxima, a data packet can be routed out of the void region and can be switched back to the greedy mode.

The problem of void regions that existed in DBR has been successfully solved by the HydroCast. The authors have considered the quality of wireless channel for simultaneous packet reception among the neighbor nodes. These simultaneous receptions enable the opportunistic forwarding by a subset of the neighbors that have received the data packet correctly, which ultimately increase the delivery ratios. At the same time, due to this opportunistic routing, multiple copies of the same data packet can be delivered to a sink, which will be a burden on the network resources. Although, the simulation results have shown that HydroCast is able to provide high delivery ratios with small end-to-end delays, still no information is available about the energy usage; consumed by the pressure sensor in order to find its depth.

2.7.2.5 Temporary Cluster Based Routing (TCBR)

Many of the multihop routing protocols have been proposed for underwater sensor networks, but most of them face the problem of multihop routing where nodes around the sink drain more energy and are suspected to die early. In order to solve this problem and make equal energy consumption throughout the network, [72] proposed a Temporary Cluster Based Routing (TCBR) algorithm.

In TCBR architecture, multiple sinks are deployed on the water surface and data packets received at any sink are considered as delivered successfully because they can communicate at higher bandwidth and small propagation delay with the help of radio communication. Two types of nodes are used: ordinary nodes and some special nodes called Courier nodes. Ordinary sensor nodes are used to sense the event happening, collect information and try to forward these data packets to a nearer courier node. A small number of courier nodes (2 to 4% of total sensor nodes) are used, and these can sense as well as receive the data packets from the other ordinary sensor nodes and deliver them to a surface sink. These Courier nodes are equipped with a mechanical module which helps to push the node inside the water at different predefined depths and then pull back the node to the ocean surface. Any node equipped with a piston can do this by creating the positive and negative buoyancy. These Courier nodes will reach different depth levels and stop for a specified amount of time. After reaching the specified position, these will broadcast hello packets so that ordinary nodes around them will be aware of their presence. These hello packets can be forwarded within only 4 hops, and if an ordinary node receives them from more than one Courier node then it will forward the data packet to a nearer one within a specified amount of time, which is defined in the Hello packet.

TCBR completes its task of equal energy consumption throughout the network with requiring a small number of Courier nodes, instead of equipping the mechanical module with every sensor node. However, data can be collected when a courier node reaches the communication range of every sensor node. Due to this, all the sensor nodes will hold their generated data packets in a limited buffer until a Courier node visits them. Despite this feature, the TCBR is not suitable for time critical applications.

2.7.2.6 Multi-Sink Opportunistic Routing Protocol

[73] proposed Multi-Sink Opportunistic routing protocol for underwater mesh networks. They defined a tiered architecture for deploying the underwater sensor nodes, where an acoustic mesh network is located between underwater network and central monitoring system which acts like a backbone network for sensor nodes. A quasi-stationary 2D UWSN architecture is considered for shallow-water coastal area. This architecture is composed of five types of elements including ordinary sensor node, mesh node, UW-sink, surface buoy and monitoring center. Among these, three of them including sensor node, mesh node and UW-sink are anchored to the sea bed and surface buoy are placed at the ocean surface. Further, both the UW-sink and surface buoy are connected through a wire. An onshore central monitoring system is used which is connected to the internet. Compared with ordinary sensor node, a mesh node is more sophisticated as it has more memory, longer transmission range and better processing power. In order to help the network survive for longer period, an underwater man controlled vehicle is used for recharging these mesh nodes.

After observing the occurred phenomena, each sensor node transmits its sensed data to the nearer mesh node. Mesh nodes first aggregate the received data and then send it to the UW-sinks via multi-hop acoustic channel. Finally, the aggregated packets are delivered to the surface buoy and from here these packets are sent to the onshore monitoring system. The proposed scheme is the best protocol where data packets are forwarded along redundant and interleaved paths. The source node transmits the data packets simultaneously, but not sequentially, over multiple UW-sinks located at different locations. Different from the opportunistic routing, this protocol exploits the packet duplications to increase packet delivery ratio.

However, the proposed routing protocol has some serious performance issues. First of all, it is assumed that each mesh node has information not only about its adjacencies but also about all the UW-sinks, like node IDs and their geographic positions. Secondly, the authors considered a quasi-stationary network but not completely mobile. , which is the reason for assuming that the mesh nodes and their neighbors are relatively static, which can be different in practical. Moreover, packets are forwarded along redundant and interleaved paths, so multiple copies of the same

packet can be generated and these duplications will continue to increase as the number of hops along the path starts to increase.

2.7.2.7 Multipath Virtual Sink Architecture

The network topology is important for determining network reliability, capacity and energy consumption. A sufficient robustness and redundancy must be available in the network in order to ensure that it will continue to work even when a significant portion of the network is not working properly. Based on these facts, [15] proposed a Multipath Virtual Sink architecture in order to make a robust network. In the proposed architecture, the whole network is divided into clusters of sensor nodes where each cluster has either one or multiple local aggregation points. These aggregation points will form a small mesh network that connects to local sinks as shown in figure 2.5. Here it is assumed that local sinks are connected via high speed links, possibly RF communications to a network where resources are more than sufficient in order to fulfill the communication needs of different applications. The ultimate goal of this architecture is to ensure that data packets are received at any one, or more of these local sinks which collectively form a virtual sink.

As the acoustic channel is intermittent in terms of connectivity, and available bandwidths are very small, it can be better for the sensor nodes to cache the sensed data and transmit when the channel conditions are favorable instead of making multiple transmission attempts. For delayed sensitive data, instead of caching, the system will try to forward data packets through multiple paths which increase the probability of successful data delivery. The local aggregation points form a wireless mesh network where multiple paths are available to reach the multiple local sinks. Each sink broadcasts a hopcount message in order to identify itself. All the sensor nodes that receive this message will update their hopcount value, and rebroadcast this message after making an increment of one. When a sensor node has data packet for sending, it can forward this packet towards any connected local sink by using the previous hop recursively. They check the performance of the architecture by making multiple transmissions with single path then forwarding multiple copies at different routes to ensure that the transmissions reach different sinks.

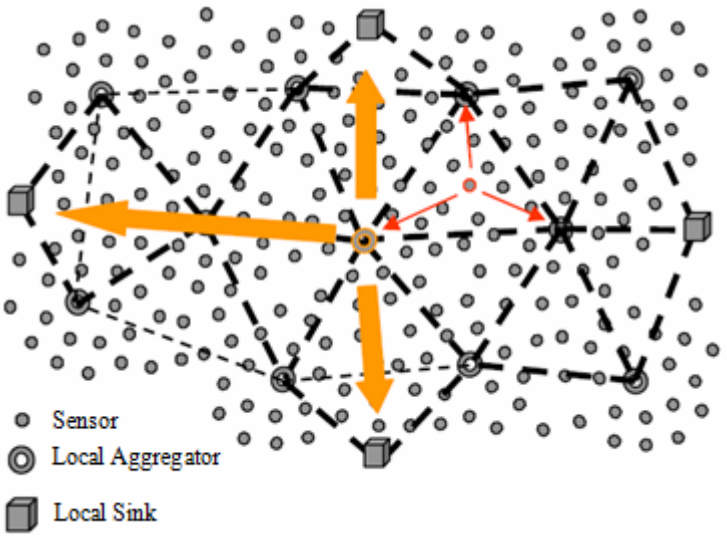

Figure 2.5 Proposed underwater network topology for Multipath Virtual Sink architecture [15]

In the proposed scheme, reliability is improved as duplicate packets are delivered towards multiple sinks through multiple paths. However, the problem of redundant transmission exists which can consume critical underwater resources.

2.7.3 Flat Routing Protocols

The routing protocols that follow flat model assume an unstructured sensor network topology and due to this reason these schemes are considered more suitable for dynamic underwater sensor networks. In this scheme, all the sensor nodes are on the same level and typically assigned equal roles *or* functionality in whole network where sensor nodes collaborate with each other to perform the assigned task. Due to large number of sensor nodes, it is not feasible to assign a unique global identifier to every node and this phenomena lead to data centric routing where base station sends queries to specific nodes or certain regions and then wait for their replies. However, the drawbacks of such flooding mechanism include implosion, which results duplicate packets to same node and reception of similar data packets to the same neighbor. These all issues results the blindness in resource consumption especially large amounts of energy wastage without considering the energy constraints. Designers

suggest different techniques like data negotiation and elimination of redundant data packets in order to decrease the burden of energy consumption. Instead of all these issues, flat routing is a more suitable choice for underwater scenarios and most of the proposed routing protocols like [6, 34, 79] are belong to this routing scheme.

2.7.3.1 Depth Based Routing (DBR)

For location based routing schemes, most of the protocols require and manage full-dimensional location information of the sensor nodes in the network which itself is a challenge left to be solved for UWSNs. Instead of requiring complete localized information, DBR [6] needs only the depth information of sensor node. In order to obtain the depth of current node, the authors suggested to equipping every sensor node with an inexpensive depth sensor. In their architecture, multiple data sinks placed on the water surface are used to collect the data packets from the sensor nodes. DBR takes decision on the depth information, and forwards the data packets from the higher to lower depth sensor nodes. When a node has a data packet to be sent, it will sense its current depth position relative to the surface and place this value in the packet header field “Depth” and then broadcast it. The receiving node will calculate its current depth position and can only forward this packet if its depth is smaller than the value embedded in the packet, otherwise it will simply discard the packet. This process will be repeated until the data packet reaches at any of the data sink. Packets received at any of the data sink are considered as successful delivery at the final destination as these data sinks can communicate efficiently with much higher bandwidth through radio channel.

However, it has some serious problems. Firstly, DBR has only greedy mode which alone is not able to achieve high delivery ratios in sparse areas. In such areas, it is possible that no node can be eligible as a forwarding node due to greater depth as compared to sending node; and current node will continue to make more and more attempts. Though some nodes can be available here at the higher depths i.e. those that can forward these packets towards the data sink successfully, but the mechanism which can handle such situations is not available. So in sparse areas, the performance of the protocol can decrease. Secondly, forwarding the data packets in a broadcast

fashion can decrease the performance of the network. The authors have even defined a mechanism when two or more nodes are candidates for further forwarding of the same data packet; then which node will be eligible for the task. Still, as a result of these broadcasts more and more nodes will receive the data packets and calculate their depth every time, which is an inefficient use of limited available energy. In short, both much sparser and high density areas are problems for DBR as increasing densities not only increase the energy consumption but also create complexities which can lead towards inefficient use of memory and packet losses.

2.7.3.2 Efficient Data Delivery with Packet Cloning

In mobile sensor networks, possibly multiple paths can exist from a sensor node to the destination and these paths may or may not be disjointed. It has been shown that, routing over these multiple paths not only helps to increase the data delivery ratios, but also achieves timeliness of delivery. As these paths start to converge at the destination, the possibility of contention starts to increase as well. The contention that arises among nodes in close proximity can be viewed positively. In order to get benefit from the proximity of nodes, [34] proposed a Packet Cloning technique which helps to enhance the data delivery ratios. The proposed scheme utilizes this idea to selectively clone data packets during the forwarding process to the destination. Different from the controlled broadcast or conventional multipath routing, where duplicate packets are indistinguishable because the involved nodes have no idea how many duplicates have been introduced, the current technique has the ability to control the number of packet clones according to the link quality and channel conditions in order to minimize the contention and energy expenditures.

During the packet cloning process, a relay node will not resend an incoming packet if it has already received one copy. This will help to prevent excessive network traffic. However, the authors want to exploit the advantage of having two distinct copies of the same data packet along two disjointed paths. For this, distinct copies of original packet are created while the number of distinct copies is a parameter that can be adjusted according to the conditions. A source node will first determine how many distinct copies it wants, and then it will start to send each copy sequentially with some

interval between them. The total number of copies produced and the identification number of the particular copy are mentioned in the packet header. When a clone packet is received by an intermediate relaying node then it can derive some information from the incoming packet. This extracted information is useful for detecting the duplicates and packet losses. Duplicate packet received is simply discarded, and new packet clones are relayed, while missed or lost packet clones are generated and transmitted. When a source node performs the packet cloning then it sends out each clone after selecting a proper value of interval which depends on the physical channel parameters. By doing so, it will help to reduce the chances of clones contending and interfering with each other.

Although, multipath routing schemes are able to increase network robustness not only by increasing the delivery ratios, but also by decreasing end to end delays, however, the acoustic channel is power-hungry compared to RF based. Thus, in order to increase the delivery ratio, more paths are suggested and these multiple paths continue to produce duplicates if the channel quality is not good. In short, RF based communications can support these schemes but for highly power consuming acoustic environment, techniques like packet cloning are not easily affordable.

2.7.3.3 Information Carrying Based Routing Protocol (ICRP)

Most of the routing protocols, even for terrestrial or underwater sensor networks, use separate packets for *control information* and *data transmission*. [79] proposed a novel reactive protocol called Information-Carrying based Routing Protocol (ICRP) in order to address the routing problem for underwater communications. ICRP is used for energy efficient, real-time and scalable routing where *control packets* used for information sharing are carried by the *data packets*. Most importantly, it doesn't require state or location information of the nodes, and in addition only a small fraction of the nodes are involved in the routing process.

In ICRP, the route establishment process is initiated by the source node. When a node has a data packet in order to send, first it will check the existing route for this destination. If no route exists then it will broadcast the data packet which carries the

route discovery message. All the nodes receiving this packet will also broadcast it, and maintain the reverse path through which this packet passes. Finally, when the destination node receives this data packet then it gets the complete reverse path from the source to its destination. Now, the destination node can use this path to send the acknowledgment. The path will remain valid for the data packet transmission, till the source node receives the acknowledgment packets. Each path has a time priority which denotes the time that this route is not used for transmission and it is called route lifetime. The larger the lifetime of a route, the longer the route can be valid or even remain unused. When the lifetime exceeds the threshold value TIMEOUT, the route becomes invalid. After which, all the nodes using this route needs a route rediscovery when the route is needed again.

Although, ICRP has been evaluated by both simulation and experimental means using a test bed, but the test bed only consisted of three sensor nodes which do not reflect the traffic of most real life UWSN scenarios. Basic routing mechanism does have some performance problems. Firstly, when a node doesn't have route information for a specified destination then it will broadcast the data packet. More broadcasts will result in wastage of node energy, which decrease the life of the whole network. Secondly, every route has an expiry time which can be very sensitive for delivery ratios. On one hand, if it is very long then nodes can move and this route can create complexity, while if too short then it will help to increase more and more broadcasts. Moreover, routing decisions are totally based on the cached route information. For UWSN where nodes move continuously at 2-3 m/sec with the water currents, in such situations any intermediate node of the route can be unavailable.

2.7.3.4 Mobile Delay-tolerant Approach (DDD)

Acoustic channel imposes higher energy consumption than radio signal. Due to higher power usage of acoustic modems, energy saving for underwater sensor networks becomes even more critical than in traditional sensor networks. In order to increase the energy efficiency in resource constraint underwater environment, [80] proposed a Delay-tolerant Data Dolphin (DDD) scheme for delay tolerant applications. DDD exploits the mobility of collector nodes called dolphins to harvest information sensed

by the stationary sensor nodes. The proposed scheme avoids energy expensive multi-hop communication, and each sensor node is only required to transmit its collected data directly to the nearest dolphin when it reaches its communication range.

In their architecture, stationary sensor nodes are deployed on the seabed in the whole area of interest. These nodes collect the information from the environment and the sensed data are stored locally after processing. These sensors periodically wake up for sensing and event generation. The acoustic modem is based on two components. The first component is used for acoustic communication with the near dolphin, and the other is a low power transceiver used to determine the presence of dolphin nodes (by a special signal transmitted from the dolphin) and to trigger the first component. Besides the sensor nodes, a number of dolphin nodes are used to collect the data packets when they move within the one-hop range of scattered sensor nodes. The dolphins can move either with random or controlled mobility according to the network condition. A dolphin node broadcasts beacons to advertise their presence. Beacons are transmitted at such acoustic frequencies, those which are compatible with the low-power sensor modem. Advertising period t is adjusted according to deployment and communication range r of sensor nodes, and to the speed of dolphin v. Finally, the dolphins deliver gathered data packets as soon as they reach a base station on the surface.

The quantity of dolphin nodes is the most important parameter for evaluation of DDD performance. If the number of dolphin nodes is not enough, they will not be able to gather all the data packets from the sensor nodes. Since dolphins move randomly, therefore it is possible that they cannot visit some sensors directly; which results in loss of existing data packets from the limited memory of the sensor node when there is no memory space left. Increasing the number of dolphin nodes, say 7 dolphins for 25 sensor nodes as in the simulation, then cost will becomes a major issue.

2.7.3.5 Adaptive Routing

An underwater sensor network can be easily partitioned due to continuous node mobility and sparse deployment. This results in unavailability of persistent route from the source to destination. Therefore, an underwater sensor network can be viewed as Intermittently Connected Network (ICN) or Delay/Disruption Tolerant Network (DTN). Traditional routing techniques are not usually suitable for ICN or DTN, since data packets will be dropped when routes are not available. Further, an USN is frequently required to provide distinguished packet deliveries according to different application requirements. Therefore, it is desirable to design a smart routing technique that could manage different application requirements adaptively.

For this purpose, [81] proposed a novel routing technique called Adaptive routing for underwater Delay/Disruption Tolerant Sensor Networks where it assumed that all nodes know about their 3-d position. Here routing decisions are made according to the characteristics of data packets and the network conditions. The purpose of this protocol is not only to satisfy different application requirements, but also to achieve a good trade-off among delivery ratios, end-to-end delays and energy consumption for all data packets. The packet priorities are calculated from the packet emergency level, packet age, density of the neighbors around a node and battery level of the node. The novelty of their work is that here different number of message copies are created according to the characteristics of data packets and network. In order to make the protocol flexible according to the conditions, all the elements in the information are variable except the emergency level. They divide the whole routing spectrum into four states, and routing is conducted according to calculated results. Simulation results show that, such a strategy can satisfy different application requirements like delivery ratio, average end-to-end delay and energy consumption. However, the proposed scheme calculates these priorities separately for each data packet after receiving them. Such calculations require high frequent communication with the neighbor nodes; which not only can be burden on node energy but also it can help to increase end-to-end delays.

2.8 Classification and Performance Comparison

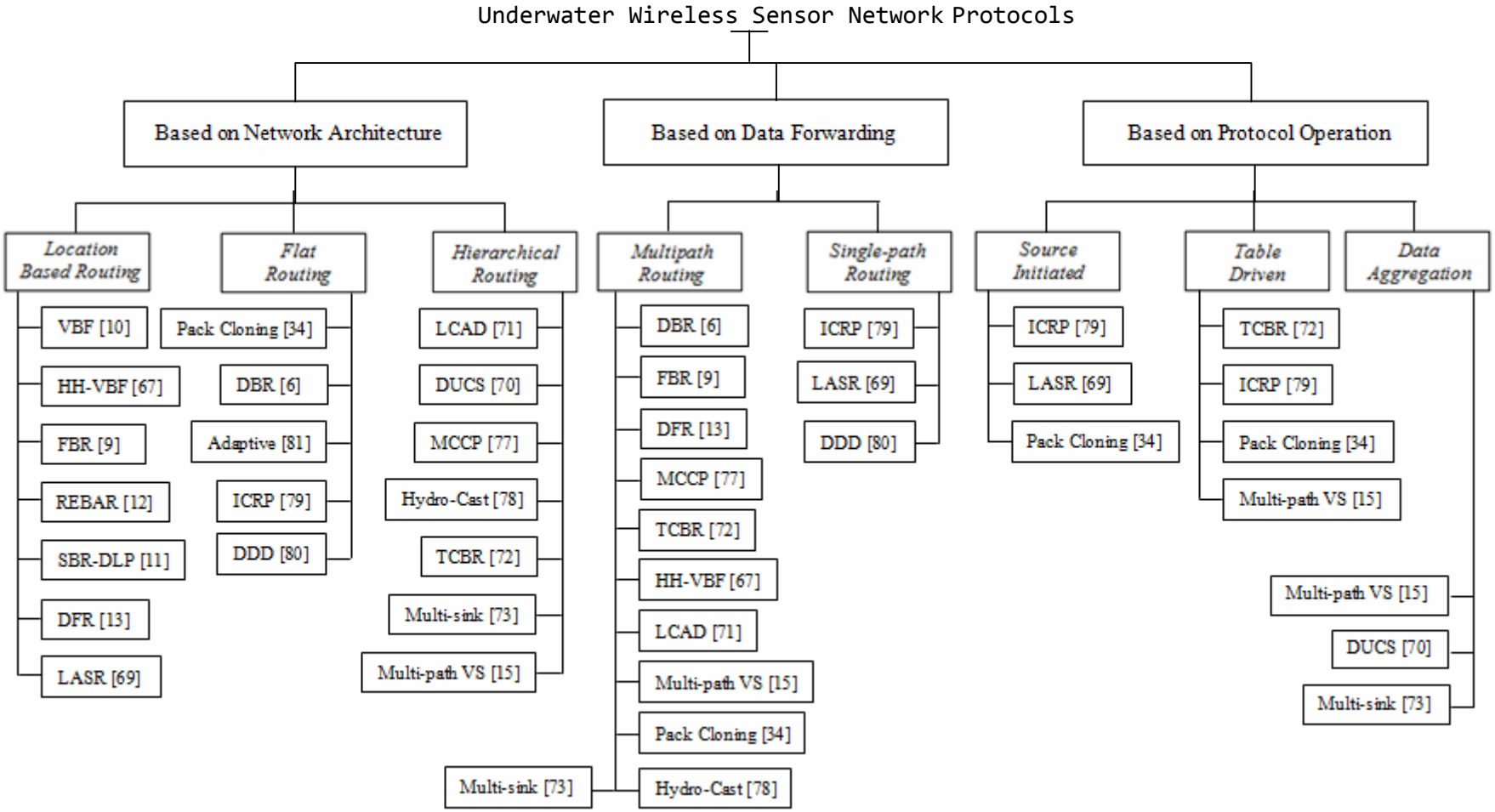

Figure2.6 General classification of UWSN routing protocols discussed during this chapter.

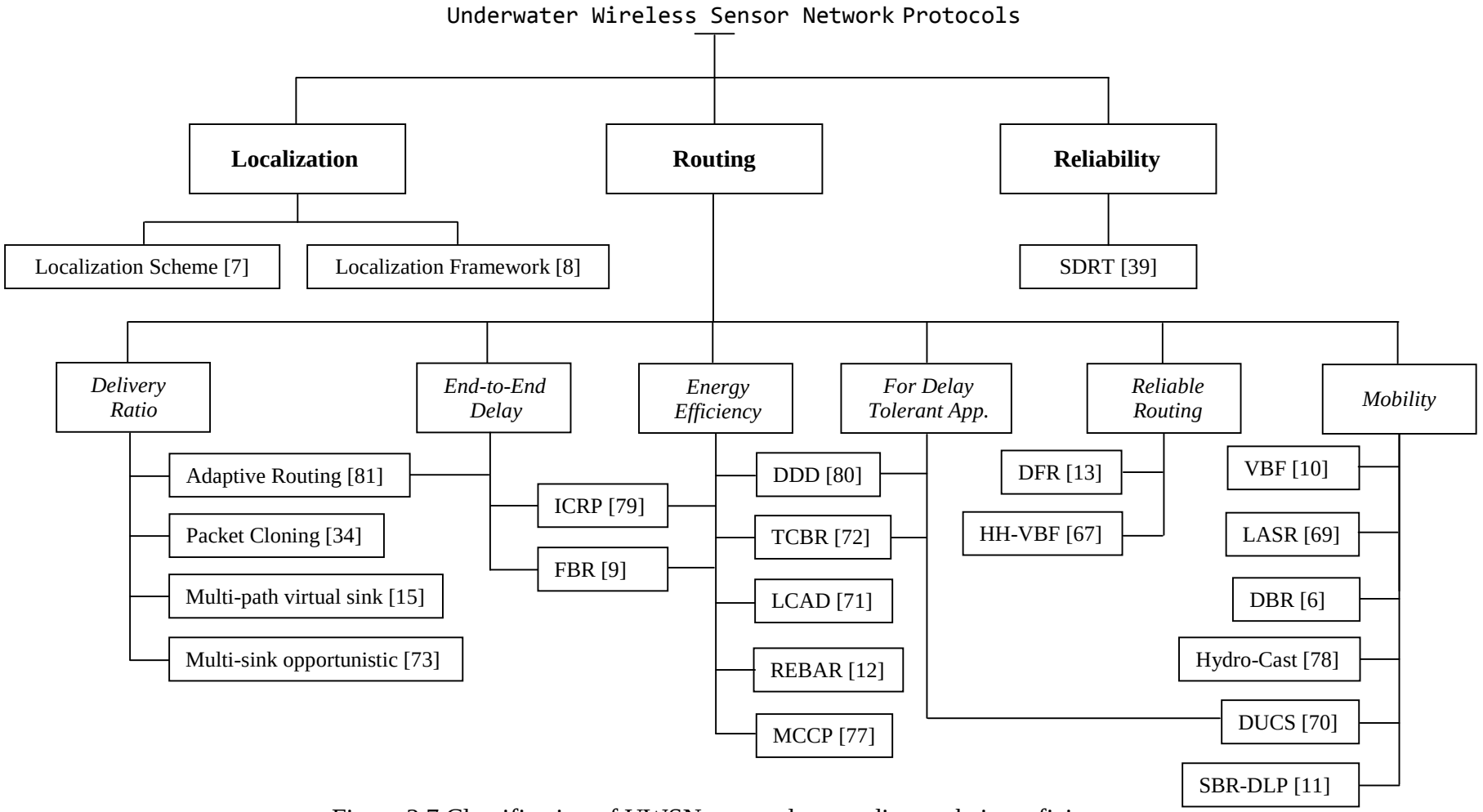

Figure 2.7 Classification of UWSN protocols according to their proficiency

Table 2.4 Performance comparison of UWSN protocols

Protocol/ Architecture	Single/ Multiple Copies	Hop-by-hop / End to end	Clustered / Single entity	Single/ Multi Sink	Hello or Control Packets	Requirements & Assumptions	Knowledge Required/ Maintained	Remarks	Year of Pub.
VBF [10]	Multiple	End-to-end	Single-entity	Single-sink	No	Geo. Location is available	Whole network	Considered as first geographic routing approach for UWSN.	2006
HH-VBF [67]	Multiple	Hop-by-hop	Single-entity	Single-sink	No	Geo. Location is available	Whole network	Enhanced version of [40], robustness improved by introducing hop-by-hop approach instead of end-to-end.	2007
FBR [9]	Single-copy	Hop-by-hop	Single-entity	Multi-sink	Yes	Geo. Location is available	Own & Sink location	A cross layer location based approach, coupling the routing, MAC & physical layers.	2008
DFR [13]	Multiple	Hop-by-hop	Single-entity	Single-sink	No	Geo. Location is available	Own, 1-hop neighbors & sink info.	A controlled packet flooding technique, which depends on the link quality, while it assumed that, all nodes can measure it.	2008
REBAR [12]	Single-copy	Hop-by-hop	Single-entity	Single-sink	No	Geo. Location is available	Own & Sink location info.	Similar with [9], but use adaptive scheme by defining propagation range. Water movements are viewed positively.	2008
DUCS [70]	Single-copy	Hop-by-hop	Clustered	Single-sink	Yes	n/a	Own cluster info. (1-hop)	A self-organizing algorithm for delay tolerant applications. Assumes that sensor nodes always have data to send.	2007
Packet Cloning [34]	Multiple	Hop-by-hop	Single-entity	Multi-sink	No	n/a	amount and sequence of clones	Unlike controlled broadcast, discernible clones of a data packet are forwarded according to network	2007
SBR-DLP [11]	Single-copy	Hop-by-hop	Single-entity	Single-sink	Yes	Geo. Location is available	Own location & sink movement	Similar with [42], but doesn't assumes that destination is fixed plus it consider entire communication circle instead of single transmitting cone.	2009
Multipath Virtual Sink [15]	Multiple	Hop-by-hop	Clustered	Multi-sink	Yes	Network with special setup	Own cluster information	Advantage of multipath routing without creating any contention near the sink.	2006

ICRP [79]	Multiple	End-to-End	Single-entity	Single-sink	No	n/a	Source to sink information	Control packets of route establishment are carried out by the data packets.	2007
DDD [80]	Single-copy	Single hop	n/a	n/a	Yes	Network with special setup	About dolphin node presence.	A sleep and wake-up scheme, which requires only one-hop transmission.	2007
DBR [6]	Multiple	Hop-by-hop	Single-entity	Multi-sink	No	Nodes with Special H/W	No network information maintained	Considered 1st depth based routing. After receiving data packet, nodes with lower depth will accept and remaining discards.	2008
HydroCast [78]	Multiple	Hop-by-hop	Clustered	Multi-sink	No	Nodes with Special H/W	2-hop neighbour's	Anycast pressure based routing, a subset of forwarder nodes are selected to maximize greedy progress.	2010
TCBR [72]	Single-copy	Hop-by-hop	Clustered	Multi-sink	Yes	Network with special setup	3-hop neighbors	Temporary clusters are formed to balance energy consumption in whole network.	2010
MCCP [77]	Single-copy	Hop-by-hop	Clustered	Multi-sink	Yes	n/a	2-hop neighbors	2-hop cluster formation algorithm, but does not support multi-hop communication.	2007
Multi-Sink Opportunistic [73]	Multiple	Hop-by-hop	n/a	Multi-sink	No	Network with special setup	Location of nearer mesh node	A best effort protocol. Data packets are forwarded along redundant and interleaved paths, towards multi-sinks.	2008
LCAD [71]	Single-copy	Hop-by-hop	Clustered	Single-sink	Yes	Nodes with Special H/W	Own cluster Information	Clusters are formed, in order to avoid multi-hop communication.	2008
LASR [69]	Single-copy	End-to-End	Single-entity	Single-sink	Yes	Network with special setup	Source to sink information	A DSR modification. Location and link quality awareness is included. Preferred only for small networks.	2006
Adaptive Routing [81]	Multiple	Hop-by-hop	Single-entity	Single-sink	Yes	n/a	Own & 1 hop neighbors info	Both, the packet and network characteristics are considered before deciding about the packet forwarding.	2008

Table 2.5 Performance comparison of UWSN protocols

Protocol / Architecture	Delivery Ratio	Delay Efficiency	Energy Efficiency	Bandwidth Efficiency	Reliability	Cost ($) Efficiency	Performance
VBF [10]	Low	Low	Fair	Fair	Low	n/a	Low
HH-VBF [67]	Fair	Fair	Low	Fair	High	n/a	Fair
FBR [9]	Fair	High	High	Fair	Fair	n/a	High
DFR [13]	Fair	Fair	Low	Fair	High	n/a	Fair
REBAR [12]	Fair	Low	High	Fair	Fair	n/a	Fair
ICRP [79]	Low	Low	Fair	Fair	Low	High	Low
DUCS [70]	Fair	Low	Fair	Fair	Low	High	Low
Pack Cloning [34]	High	Fair	Low	Low	Fair	High	Fair
Multipath VS[15]	Fair	Fair	Low	Fair	High	Low	Fair
DDD [80]	Low	Low	High	Fair	Fair	Low	Low
DBR [6]	High	High	Low	Fair	High	Fair	High
HydroCast [78]	High	High	Fair	Fair	Fair	Fair	High
TCBR [72]	Fair	Low	Fair	Fair	Fair	Low	Low
MCCP [77]	Low	Low	High	Fair	Fair	High	Fair
LCAD [71]	Fair	Low	Fair	Fair	Low	Low	Low
LASR [11]	Fair	Low	Fair	Fair	Fair	High	Fair
Adaptive [81]	High	Fair	Flexible	Flexible	Flexible	n/a	Fair

2.9 Evaluation Methods

Analytical modeling, real deployment and numerical simulation are the most commonly used techniques for the analysis of the performance of terrestrial and underwater acoustic sensor networks. Each of the available techniques has its own advantages and limitations depending on the considered network characteristics. First of all, consider about analytical modeling whose methods are very complex especially for underwater scenarios and certain simplifications are usually assumed to predict the performance of the proposed scheme. Such assumptions and simplifications may lead to imprecise results with limited confidence. Further, it may not be feasible to evaluate the performance of these schemes through real experiments due to unavailability of appropriate hardware in terms of technical and design limitations [82, 83]. Even we are able to arrange some suitable hardware; it may still be not practical to experiment in an appropriate environment like deep water. Usually, such experiments require hundreds of sensor nodes consequently positioning the cost to be another important issue. In brief, evaluating any scheme proposed for UWSNs through real deployment is not only complex and costly but also time-consuming. Among all the techniques discussed in this chapter, only [79] was evaluated through physical deployment. However this experiment was based on only three sensor nodes which may not practically reflect the traffic of most of the real life UWSN scenarios.

To solve this problem, some underwater acoustic research laboratories like [84] prefer to analyze protocol performance through physical test bed as presented in [85, 86]. The test bed in [86] provides a set of acoustic communication hardware and software including an emulator with realistic network settings. Such tools are very helpful to provide an environment that is similar to the real deployment. The test bed not only is considered as an effective option when real deployment is not feasible, but also can provide more trusted results than software based tools. However, it may be limited in scope and may fail to offer realistic environment.

Most of the routing protocols proposed for UWSNs consider different design philosophies and application requirements. None of them can work efficiently for all the performance parameters like network size, communication overhead, node mobility, etc. The large variations in the performance metrics make it difficult task to

present a comprehensive evaluation for a large number of routing strategies [87].

Compared to real implementation, simulation is the most popular and effective approach to design and test any routing scheme, not only in terms of cost and time, but also for higher level of detail existence. Discussing about underwater environments, evaluation through simulation becomes an ideal choice in that highly sophisticated hardware is required and difficulty factor at the same time is presented in setting up real deployment scenario. However, the appropriate selection of a simulation framework according to problem and network characteristics is a critical task [49]. Excluding a few, the performance of all the schemes discussed during this study, has been evaluated with the help of different simulation tools. Most commonly used in terrestrial sensor network simulations, open source ns-2 [88] is being used in underwater sensor networks as well like [89, 90]. Generally, it does not support acoustic communication characteristics, so it can be used with two options. Firstly, many authors used ns-2 by inserting larger propagation delays and other channel problems in order to produce more realistically accurate results i.e. nearly or equal to real conditions. Multi InteRfAce Cross Layer Extension (MIRACLE) [91] is another important example of NS2 extension which supports multiple wireless interfaces and cross layering features. Further a module for ns-miracle is developed [92] and used in [93, 94] for detailed simulation of underwater channel according to the propagation speed in underwater environment including the impact of depth, salinity and temperature. Furthermore, some have developed ns-2 extensions for the aquatic environment. AquaSim [95] used in [6, 96] is an important example, which not only supports acoustic links but also 3-D topology. Some commercial network simulators like Opnet [97] was used in [98, 99] and Qualnet [100] in [78]. Tools, especially developed for underwater environments like AUVNetSim [101] was used in [9]. Some custom simulators written in different languages have also been implemented, including VC++ [102] and C++ based simulator DEVCES [103] used in [104].

Although, the discussed simulators play an important role for developing and testing new protocols for UWSNs, there are always some kinds of risk involved, as simulation results may not be accurate due to unrealistic underwater characteristics like continuous mobility in a 3-d volume. To analyze a protocol more effectively, it is

important to know different available tools and understand the associated benefits and limitations. Due to different performance requirements according to specific applications, a general tool for underwater acoustic networks is still lacking at present.

2.10 Research Directions

Based on the work discussed in previous sections, it is clear that many issues are left to be solved. In this section, we are listing some of the open research issues, those must be considered during the future work for underwater environments.

- As the available data rates are extremely low, the routing overheads for the protocols of such networks should be kept as minimum as possible.
- Research required on variable packets lengths to increase the channel utilization.
- The routing must be self-configuring in case of any failure because equipment is deployed far from the experts.
- According to the underwater environments, the algorithms should provide strict or loose latency bounds for time critical applications.
- Taking the routing decision on the latest available information.
- Idea of *Per-contact routing* is better, instead of *source routing* or per-*hope routing*, although it can require more processing for large networks, but it provide more reliability for dynamic conditions of underwater environment.
- For delay-tolerant applications, trying to develop mechanisms to handle loss of connectivity without immediate retransmissions.
    - Integration of transport layer with the data link layer can be helpful for this.
- Many of the ground based algorithms use the node movement models and their directions, so water current movement models similarly can give the idea of node movements for the better routing.
- For energy efficiency, local route optimization algorithms are needed in order to manage the consistent variations of the network.
- For energy concern, it is better to develop a routing protocol that sends message over multiple short steps, instead of sending over long links.

- Distributed protocols can give better results as they divide the processing load into different nodes, and it can help to increase the life of the network.
- In the case of multi-copy algorithms, when one copy has successfully reached at the destination, the way of the intermediate nodes can be informed to discard the remaining copies of the same packet for the best utilization of the resources.

CHAPTER 3

METHODOLOGY

The idea of implementing the sensor networks into underwater communications has received more and more interest during the last couple of decades. Although underwater sensor networks (UWSN's) shares many characteristics with terrestrial sensor networks, such as the requirements of energy issues and large number of nodes, UWSNs are still different in some ways from the conventional terrestrial sensor networks. As a result of very different environment features and the unique nature of the aquatic applications, the protocols developed for ground sensor networks are not directly applicable to underwater sensor networks. All these issues as a result require an extensive research to develop specialized routing protocols that can provide the best results according to these new situations. Water current movements for example are considered as a major issue, yet a good routing technique can be helpful to minimize this effect. It is then found that the available data rates are extremely low, leading the routing overheads for the protocols of such networks to be kept as minimal as possible for the increase of the delivery ratios with small end-to-end delays. Routing skills afterward can be used in order to cope with many energy concerns; hence we can increase the network life. In short, the complexity of underwater environment and the expectations of the user scenarios in turn require a more scalable, robust and networked solution. To fulfill these demands, a new research at every layer of the protocol stack is required. These all factors provide a platform where a resource aware routing strategy plays a vital role to fulfill the different application requirements with dynamic environmental conditions.

Routing is a fundamental issue for any network and routing protocols are considered to be in charge of discovering and maintaining the routes. In discussing about underwater senor networks, most of the research is performed on the issues related to physical layer while issues related to network layer such as routing

techniques relatively are a new area. For this point, it is becoming essential task to provide an efficient routing algorithm.

Based on the issues identified by means of literature review presented in Chapter 2, this chapter describes a methodology incorporated into this research work and is differently divided into six sections. Section 1 presents the basic idea of H2-DAB with its complete explanation, and section 2 presents a discussion of different theoretical comparison of H2-DAB with DBR. Section 3 provides the improved H2-DAB in which the courier nodes are used in order to increase the efficiency of resource consumption is introduced. While, section 4 presents the analytical model of energy consumption for H2-DAB. Further, section 5 presents a 2H-ACK model in order to increase the reliability of H2-DAB. Lastly, section 6 identifies future issues with potential research areas for underwater routing and communication.

3.1 Hop-by-Hop Dynamic Addressing Based Routing Protocol

In this dissertation, a novel routing protocol called **Hop-by-Hop Dynamic Addressing Based (H2-DAB)** for critical underwater monitoring missions is proposed. H2-DAB is scalable and energy efficient that uses multi-sink architecture. Several surface buoys will be used to collect the data at the surface and some nodes will be anchored at the bottom. Remaining nodes will be deployed at different levels from surface to bottom. Nodes near the surface sinks will have smaller addresses that in turn will increase as the nodes descent towards the bottom. These dynamic addresses will be assigned with the help of Hello packets - those generated by the surface sinks. Any node which collects the information will try to intensely deliver it towards the upper layer nodes. Packets that reach any one of the sinks will be considered to be successfully delivered to the final destination in that these buoys have the luxury of radio communications enabling them to be able to communicate with each other at higher bandwidths and lower propagation delays. For better resource consumption and an increase of reliability, some special nodes called Courier nodes will be used. These Courier nodes will collect the data packets from lower layer nodes, particularly from the nodes anchored at the bottom. Having collected the data packets, the Courier nodes will deliver these packets directly to the surface sinks.

The main advantages of H2-DAB are as follows:

I. Not requiring any location information.
II. Able to easily cope with the node movements with water currents.
III. No need to maintain complex routing tables.
IV. Taking advantage of multisink architecture (For single sink, nodes near the sink entertain large amount of data packets, not only can it lead to the problem of congestions and data losses but also these nodes can stop working early due to frequent energy consumption)

The primary goals of Hop-by-Hop Dynamic Addressing Based (H2-DAB) routing protocol include as follows:

i) Maximizing the delivery ratio
ii) Minimizing the message latency
iii) Optimizing energy consumption.
iv) Completing all these, without requiring any extra or specialized equipment.

3.1.1 Problem Setting and Network Architecture

During this research, the application of underwater oil/gas field monitoring is considered. For this purpose, underwater sensor nodes are deployed in the whole monitoring area to collect the information from the surroundings and report to the surface buoys. As already mentioned, our protocol is based on the multisink architecture, which is very helpful to increase not only the delivery ratios but also the network life by decreasing the energy consumption of the nodes around the sink. Surface sinks are equipped with radio and acoustic modems, where RF modems will be used to communicate with each other and with the final data processing centre. Acoustic modems are used to communicate with the sensor nodes deployed at different depth levels with the buoyancy control [81, 90, 105]. In horizontal directions, they can freely move with the water currents but in vertical one, a node may have small variations, which can be negligible [67, 81].

By doing so, nodes will form layers from the surface to the bottom. The numbers of layers depend on the depth of the monitoring area and the communication range of

the sensor nodes. The average depth of oceans is around 2.5km to 3km [71, 106], and acoustic communication range of sensor nodes is not preferred more than 1 km. However, by considering every layer at 500 meter, then maximum of 5 to 7 layers are required to deliver the data packets from bottom to surface at the average ocean depths. It is important to note that the performance of our protocol not depend on the number of layers. The proposed algorithm can easily support more layers, but if we increase the number of layers, the cost of the network will increase as more nodes are required for the same depth. Conversely, decreasing these layers since acoustic communications support up to the range of 5 km is not preferred in that long distance communications drain more energy [9, 107]. In order to save energy and extend the network life time, the acoustic communication range of sensor nodes up to 800 meter is defined. Here, it is found that acoustic communications are considered as a short range up to the distance of 1 km and able to provide a bandwidth of 20-30 KHz [24, 53]. Although in special cases, we can increase this [108, 109], during normal cases there is no need to increase more than the range suggested.

In many applications we are more interested to collect data from the nodes anchored at the bottom such as oil/gas field monitoring; events in such applications more frequently occurred at the bottom. To collect this frequent data from the lower layers fast with the involvement of fewer nodes, Courier nodes in turn are preferred to be used. These special nodes can sense and receive the data packets from other ordinary nodes and will deliver it to the surface sinks directly. These nodes further can move vertically with the help of a mechanical module to push the node beneath the water in reaching the bottom where it will stop for a specified amount of time and then reappear to the surface. Equipped piston can do this by creating the positive and negative buoyancy. Here one thing should be clear that, these Courier nodes can reach at any place at the bottom and surface. Due to the water currents that make it difficult to reach any specific place, it is further assumed that any Courier node to be able to reach at any place at the bottom and reappear at any area on the surface. .

H2-DAB completes its task in two phases. In the **first phase**, route creations are done by assigning dynamic *HopIDs* to every ordinary node in the network. In the **second phase**, data packets are delivered towards the surface sinks by using these *HopIDs*.

3.1.2 Addressing Schemes

Every surface sink will have two types of addresses,

- **Sink ID: a** unique ID for every sink, Hello packets will use this to recognize the sink.

- **DestID:** a static ID “0”, same for all the sinks. All the data packets will use this ID as Destination ID.

Sensor nodes will use two types of addresses (other than the anchored nodes)

- **Node ID:** a unique ID only for floating nodes, it will help to recognize the nodes during Inquiries and Data packets forwarding (*Section 3.1.6*).

- **HopID:**
  - Floating nodes will use dynamic HopIDs. It can reach to a maximum of “99” and each node has the highest value as the default HopID before receiving the Hello Packets. Once receiving the Hello packets, it will update its new HopID according to its locations (*section 3.1.4*).
  - Nodes anchored at bottom will use static HopID “100”.

3.1.3 Hello Packet Format

Hello packet consists of three fields; namely *Sink ID*, *HopID* and *Maximum Hop Count* as shown in the Figure 3.1.

Sink ID will be used when every sink broadcasts Hello packets during first phase. This ID will help the floating nodes to differentiate when they receives Hello packets from the multiple sinks.

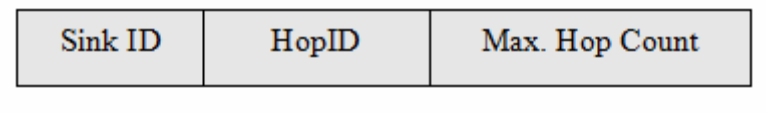

Figure 3.1 H2-DAB hello packet format

HopID, this two-digit ID shows the number of hops that every node is away from any of the one or two sinks. Left hop number has more priority and will consider as a primary route, as compared to the right hop number, which can be used as a backup route.

Maximum Hop Count field has value of 10 at start when sink broadcasts Hello packet. After receiving the packet, every node will apply the decrement of one, meaning that it will make the zero value and will not forward further to any other nodes when the 10^{th} node is received.

Nodes anchored at the bottom will not entertain these packets but will discard them. So, the life of the Hello packets can be a maximum 10 hops, or until visiting any anchored node depending on which one is firstly occurred.

3.1.4 Calculating and Assigning the HopIDs

As discussed in *section 3.1.2*, default HopIDs of every floating node will remain "99", until having not received any Hello packet. Once receiving the Hello packet from a sink less than 9 hops, a node will start to update, for example updating its ID as "19" after directly receiving the Hello packet from the sink. This new ID shows that this node is only one hop away from a sink, and can be 9 hops away from some other sinks. After updating, it will forward the Hello packet with its new ID. Similarly, the receiving nodes will use the increment of one in their hop numbers and forward those towards their neighbours. This will continue until the Hello Packets reach the anchored nodes or its hop count value becomes zero. During this time, if some nodes receive the Hello packet from any other sink, they will then start to update its right hop number (used as a back up) as the left hop value will. In some situations, if a node receives any Hello packet from the 3^{rd} sink, or a sink, from which it has already received, it will update only if the arrived packet has small hop numbers. Otherwise, it will be discarded. This whole process is described in the Figure 3.2.

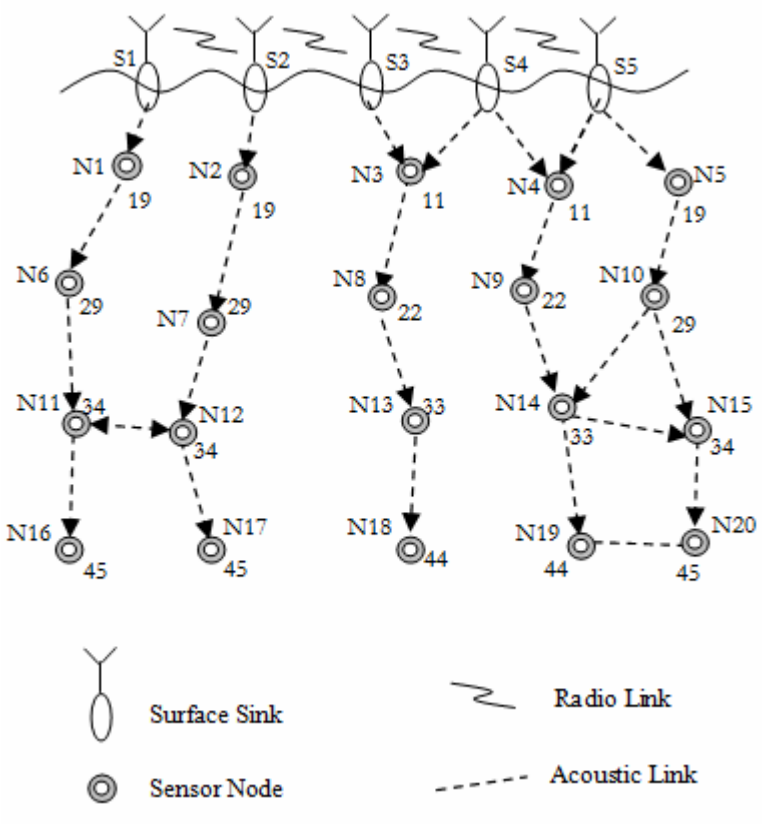

Figure 3.2 Assigning *HopIDs* with the help of hello packets

3.1.5 Data Packet Format

The Data packet format required for the H2-DAB is simple for requiring only four fields for control purposes, namely *Sender HopID*, *Next Node ID*, *Packet Sequence Number* and *Destination ID* as shown in the Figure 3.3

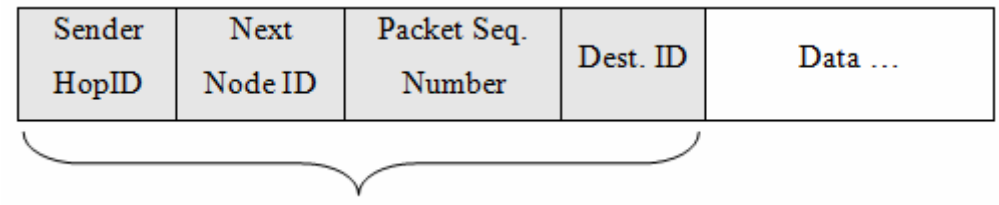

Figure 3.3 H2-DAB data packet format

Sender HopID, the HopID of current node forwarding the Data Packet, if it is an anchored node then it will use its static HopID "100". *Next Node ID*, the unique ID of a node is eligible for next hop among the neighbors, usually a node from the upper layers. *Packet Sequence Number* is a unique number assigned by the source node to the packets. *Destination ID* is a fix value "0", which is the DestID of all the sinks on the surface positioning packets likely to be delivered to any of the reachable sink.

3.1.6 Data Packet Forwarding

Figure 3.4 describes how to make the decisions about Data Packets. Here a source node N7 has a data packet, with its own HopID 66. With the help of a simple *Inquiry Request* it will ask its neighbors about their HopID's, This Inquiry contains only the *Node ID* of the requesting node. *Inquiry Reply* will be used to reply and contains only two fields, those are *Node ID* and *HopID* of replying nodes. Nodes N4, N5, N6, N8 and N9 are in the communication range and will reply with their *Node ID*s and *HopID's*. After comparing the neighbors HopID's, node N4 and N5 are declared as the candidates for the Next Hop because both have small HopID's compared to source node. N4, the backup link of which is smaller than the N5, becomes the chosen for the Next Hop. So, the source node will forward the data packet with N4 *Node ID* as a Next Hop

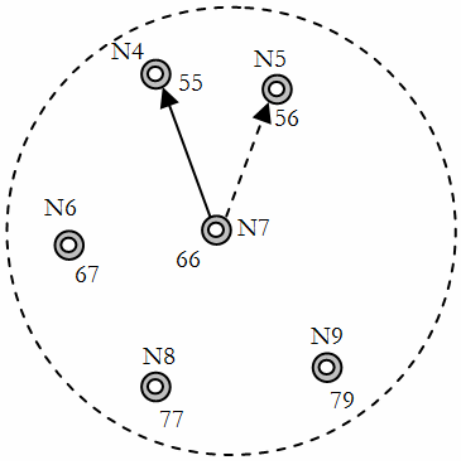

Figure 3.4 Selecting the Next Hop for the data delivery

In some cases, if two nodes reply with the same minimum HopID's, the node which will reply first will be eligible as energy concerns and packets over multiple short hops are preferred instead of sending over long links.

However it tends to be impossible that this source node can get the response from the neighbor nodes with the smaller HopID's every time. In such cases, when a source node is not able to get such a response, particularly in sparse areas, it will wait for a defined amount of time and reattempt to get the address. This waiting time in turn can vary according to the past history. After the second endeavour, if the results are similar, there will be no any node to reply from the upper layers. At this point it will prefer to forward the packet towards a node on the same layer with the HopID value close or equal to its own HopID. If there is still no node to be eligible; packets then can be forwarded towards the lower layer nodes. In some worst cases, if source node can not communicate with any other node, it for the last alternative can remove the restriction of suggested range of 1 km [16].

The above entire scenario clearly shows that delivery ratios for H2-DAB are not determined by network density or sparseness. Further we can improve the performance according to the nodes densities. If the network is dense, we can define that for simplicity, after receiving the *Inquiry Request*, only the nodes with smaller HopID's can reply. If the network is spars, all the nodes receiving the *Inquiry Requests* will reply.

3.1.7 Route Updating and Maintenance

Due to buoyancy control, nodes slightly fluctuate in vertical directions, but still can easily move in horizontal directions along with the water currents. A node, due to such movements, accordingly can change its neighbours with both upper and lower layers. However, the addresses can still be used in that address of any node will remain smaller from the addresses of lower nodes, and larger from the upper nodes. Hence, the address of any node shows how many layers or hops have to move upward to reach the surface sinks. In other words, it is not necessary to reach any specified sink, as the data packets can be delivered to any destination.

As previously mentioned in problem statement, the oil/gas field monitoring application is considered where the samples of data are required from time to time. During one interval, the data can be sensed and delivered in a short time and afterward nodes can go into a sleeping mode or can shut down the transceiver until the next sampling interval for energy saving. These sensing and sleeping intervals can be scheduled according to the situations and requirements of the network.

For the next interval, nodes will be assigned in new HopIDs according to the new locations till the next sampling interval. These intervals based HopIDs can give better results only when these intervals are not very long. If sampling intervals are long, the performance of the network with the node movements can decrease with the same HopIDs. However, it is not a serious problem as H2-DAB can easily handle such situations. For long term missions; these HopIDs have a *time property*, which denotes the validity time for these ID's. The larger the HopID lifetime is, the longer time these can continue to work. When the lifetime exceeds the threshold value EXPIRE, the HopIDs will be reset to "99" throughout the network, and again new ID's will be assigned with the help of new Hello packets. If any node has packets to be sent before the HopID expiry, it will wait and can forward these packets after getting the new HopID. Getting the new HopID's in such a way helps to handle the problem of even small vertical movements, as the next time nodes will get new ID's according to their new positions. At the same time, it makes the buoyancy control more flexible, as violations of strict layering structure can be acceptable.

3.2 H2-DAB and DBR

H2-DAB is a distributed routing protocol based on the local information of the sensor nodes. The basic network architecture and forwarding decisions about the data packets towards the sinks of our protocol are similar with DBR [6]. By comparing the aspects of both of the protocols, many differences however still exist. The actual routing idea is totally different, and on the base of this idea, here are many differences between both of these protocols.

- DBR requires that all nodes should be equipped with the Depth Sensor, which will increase the **cost**. While, in H2-DAB the idea of Dynamic Addressing is proposed as an alternate of depth sensor.

- Depth sensor will drain **energy**, every time it senses, leading it to be the burden on node's battery.

- DBR has only greedy mode in which by itself it is not able to achieve high delivery ratios in sparse networks. So, as the network becomes spars more and more, the **delivery ratios** in turn also will continue to decrease more and more. On the other hand, in H2-DAB node's sparseness there will be no serious effect on delivery ratios.

- In DBR, every packet waits in a queue every time, if network is much denser, more and more nodes in the communication range will increase the waiting queues and respectively increase more and more **delays**.

- During data forwarding, DBR suggests that every node should broadcast the data packets, which will increase **congestions and collisions** in that available bandwidths are already small. While, H2-DAB, does not require such broadcasts.

- In DBR, we can not eliminate the problem of **duplications**, as every node broadcasts the data packets. As a result such duplications can increase the **congestions as well as complexity** of the network. But in H2-DAB, there is no broadcast also meaning no duplication problems.

- In dens areas, as nodes continue to increase, the **memory requirements** for buffering will gradually increase in the case of DBR, that is not required in our H2-DAB protocol.

From the comparison above, it is clear that H2-DAB not only works for any single metric but also provides a better performance about all metrics including delivery ratios, battery usage, delays, cost, complexity, packet duplications as well as memory requirements. Now if we are using H2-DAB even for the short-term or long-term applications based on the short sampling intervals; there is then nothing specially to

do. If the applications are long-term and continuous, in such case we then have to reset the dynamic addresses after specified times according to the network situations and requirements. It is the only cost we have to pay in order to get all the advantages mentioned above.

3.3 Improving the H2-DAB

H2-DAB has many advantages such as requiring no specialized hardware, dimensional location information and easiness in handling node movements without maintaining complex routing tables. However, as it is based on multi-sink architecture, the problem of multi-hop routing still exists in which nodes near the sinks drain more energy for being frequently used causing them more susceptibly to die. To improve the life time of the nodes around the sinks, some special nodes, called Courier nodes are preferred for collecting data packets from the ordinary nodes and delivering directly to the surface buoys. Courier nodes are equipped with some piston modules helpfully to push the node under water at different defined depths and then pull back to the ocean surface. Equipped piston can do this by creating the positive and negative buoyancy. These Courier nodes will reach different depth levels and stop for specified amount of time. After reaching any specified position, they will broadcast Hello packets leading the ordinary nodes around to be able to identify their presence. Then ordinary nodes will forward the sensed data within a specified amount of time defined in the Hello packet.

3.3.1 Addressing Scheme

For H2-DAB with Courier nodes, *HopID (routing address)* is used for routing decision and *NodeID (node identifier)* separately. Every node will get its *HopID* dynamically and is changeable with the node movements according to new locations in the network. The *NodeID* is a unique address for every node throughout both its life time and the network.

Every surface sink will have two types of address,

- **SinkID:** a unique *SinkID* for every sink. Hello packets will use this ID to recognize the sink.

- **HopID:** a static *HopID* “00” is equal for all sinks. All the Data packets will use this ID as destination ID.

Sensor nodes of both types, Ordinary or Courier will also use two types of address.

- **NodeID:** a unique *NodeID* for all the nodes. It will help to recognize the nodes during Inquiries and Data packets forwarding (*Section 3.3.5*).

- **HopID:**
  - Ordinary nodes will manage and update two types of *HopIDs*, Sink HopID (S-HopID) and Courier HopID (C-HopID). *S-HopID* can reach a maximum of “99”, and each node has the highest value as the default *S-HopID* does before receiving the Hello packets. After receiving the Hello packet from any sink, it will update its new *S-HopID* according to its current location (*section 3.3.3*). Similarly, *C-HopID* will manage addresses built with the help of Hello packets received from the Courier nodes.
  - Courier nodes will use a static *HopID* “19”, and will not entertain Hello packets. If any Courier node receives Hello packet from any other node of the same type or even from the surface sink, it will simply discard it.

3.3.2 Hello Packet Format

H2-DAB will use two types of hello packets from surface sinks (S-hp) and from Courier nodes (C-hp).

Surface Sink Hello Packet (S-hp): *S-hp* consists of four fields; namely *Hello packet Type*, *SinkID*, *S-HopID* and *Maximum Hop Count* as shown in the Figure 3.5. *Hello packet Type* will be used to differentiate the type of Hello packet, as it is from the surface sink or from the Courier node. *SinkID* will be used when every sink

broadcasts Hello packets during the first phase then the *SinkID* will help the accepting nodes to differentiate the received Hello packets from the multiple sinks. *S-HopID* further consists of two digits and shows how many hops every node is away from any of the one or two sinks. Left hop number has more priority and will be considered as a primary route, as compared to the right hop number, which can be used as a backup route. Finally, the *Maximum Hop Count* field has a value of 9 at start when sink broadcasts Hello packet. After receiving, every node will apply the decrement of one, so when ninth node receives this Hello packet, it will make this value zero, and will not forward further to any other node.

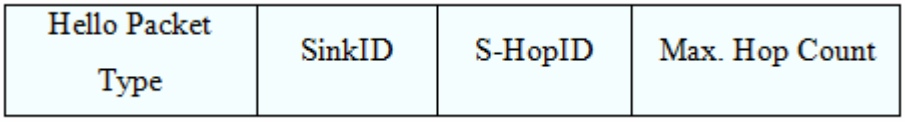

Figure 3.5 Surface sink hello packet (S-hp) format

Courier node Hello Packet (C-hp): Hello packets broadcasted by the Courier nodes have five fields as shown in figure 3.5. The first field is the same as in the *S-hp*, which shows the Hello packet is from the Courier node. Then *SinkID* is replaced by the *CourierID* of the Courier node, which has broadcasted it. Next, one extra field called "*Expiry Time*" will be used. This new field shows how long this Courier node will be available. Every ordinary node will update their *C-HopID* through these Hello packets and can forward and deliver data packets to the Courier node before the expiry time. Next, *C-HopID* consists of the same 2-digit, which shows how many hops the receiving nodes are away from the Courier node. Then the purpose of *Max Hop Count* field is also the same as in the surface sink Hello packet, but it has a value of 3 at start instead of 9, so this Hello packet can visit a maximum of 3 ordinary nodes.

Hello Packet Type	CourierID	Expiry Time	C-HopID	Max. Hop Count

Figure 3.6 Courier node hello packet (C-hp) format

3.3.3 Calculating and Assigning the HopIDs

Every ordinary sensor node uses a default value “99” as its *HopID* and “0000” as *SinkID* in routing table until it has not received any hello packet. When a node receives the Hello packet from any surface sink, Courier or ordinary node with a minimum power threshold (PT_{min}) will start to update its respective *HopID*. A node for example receiving the Hello packet directly from the sink will update its *S-HopID* as “19”. This new *S-HopID* shows that this node is only one hop away from a sink, and can be 9 hops away from any other sink. After updating, it will forward the *S-hp* with its new *S-HopID*. Similarly, the receiving nodes will use the increment of one in their *S-HopIDs*, and will forward them towards their neighbors, and this will continue until the hop count value of *S-hp* becomes zero. During this time, if some nodes receive the *S-hp* from any other sink, they will start to update its right hop number (used as a back up) just like the left hop value. In some situations, if a node receives any *S-hp* from the 3rd sink, or a sink, from which it has already received, it will update its *S-HopID* - only if the arrived packet has small hop numbers. Otherwise, it will discard it, if so, then Hello packet will not broadcast further. This whole process is depicted in Figure 7.

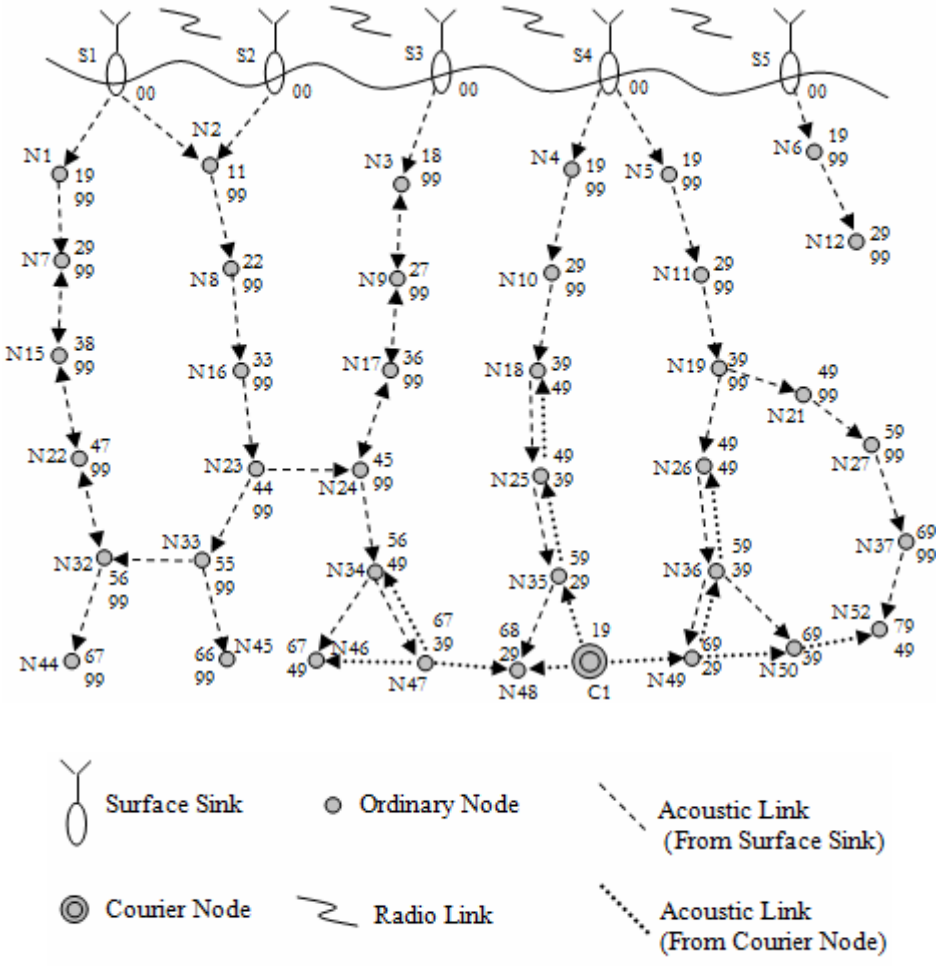

Figure 3.7 Assigning HopIDs with the help of hello packets when courier nodes are available

Similarly, a node can receive Hello packet from Courier node (*C-hp*). The procedure of updating the *C-HopIDs* from the Hello packets generated by the Courier nodes is simple compared to *S-HopIDs*. It will update only in the left hop and the right side or *backup hop* value will remain the same that is 9. Therefore, it is not necessary to check the *CourierID* before updating the *C-HopID*. If *C-HopID* of new received *C-hp* is smaller than its own, then it will update own *C-HopID*; if not, it then discard *C-HopID* simply. Here due to two reasons backup hop is not being used for Courier nodes include (1) the possibility of availability of the Courier nodes in a specific region is very small in that one or two Courier nodes can visit here at the same time and (2) Courier nodes visit for short intervals and consequently can leave

for any other location. For these short intervals and small Courier nodes, only the primary hop can provide the required results. Every ordinary node will maintain a simple routing table that consists of only one entry, which can be shown in table 2.

Table 3.1 Routing information maintained by ordinary nodes

	NodeID	**P**	**Q**	**SinkID P**	**SinkID Q**	**X**	**Y**	**CourierID**	**Courier Expiry**	**Next Hop**
t_0	107D	9	9	0000	0000	9	9	0000	N/A	Expire
t_1	107D	1	9	5D87	0000	9	9	0000	N/A	5D87
t_2	107D	1	6	5D87	8E23	2	9	4BB1	Adaptive	5D87

Table 3.1 provides an idea on how an ordinary node (107D in this example) maintains the information about the surface sinks and Courier nodes at different times. The first row provides the status of routing table at time t_0 where the first column presents its own NodeID. Next four columns maintain the surface sink *HopID* and the SinkIDs from which it has received the Hello packets. Similarly, next four columns maintain the *C-HopID,* the permanent ID of the Courier node and its expiry mentioned in the Hello packets. The last column presents the NodeID of the possible next hop, where current node can forward the data packet. The cases where no node is available as a next hop shows “expire” in this column, then the current node will follow the process of finding the next hop. When the current node is able to successfully find the next hop, it can store the NodeID of that node in this last column. Remaining two rows, t_1 and t_2, provide the updated status of first row at different time intervals which can be possible after receiving the Hello packets.

3.3.4 Data Packet Format

The Data packet format required for the H2-DAB is simple, as it requires only four fields for control purpose, namely *Source NodeID*, *Next NodeID, Packet Sequence Number* and *Destination ID*, as shown in Figure 3.8.

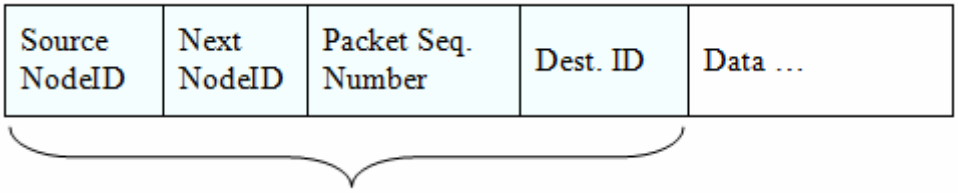

Figure 3.8 H2-DAB data packet format

Source NodeID, a node, which senses the information, will use its *NodeID* before forwarding the Data packet. *Next NodeID*, the *NodeID* of a node, considered as eligible for Next Hop among the neighbor nodes, is usually near the surface Sink or Courier node. *Packet Sequence Number* is a unique number assigned by the source node to the packets at the time of generation. *Destination ID* is a fix value "00", which is the *HopID* of all the sinks on the surface. At this point, the packets can be delivered to any of the reachable sink.

3.3.5 Data Packet Forwarding

Figure 3.9 describes how to make the decisions about Data packets. A source node N23 has a Data packet, with its own *HopIDs* 66 & 99 (*C-HopIDs* for all the nodes are 99 because no Courier node is available in this area). As a result, it will ask its neighbors about their *HopIDs* with the help of a simple "*Inquiry Request*". This Inquiry contains only the *NodeID* of the requesting node. "*Inquiry Reply*" will be used to reply and contains only three fields, those are *NodeID*, *S-HopID* and *C-HopID* of replying nodes. Nodes N15, N16, N22, N24 and N25 are in the communication range and will reply with their *NodeID*s and *HopIDs*. After comparing the neighbors' *HopIDs*, node N15 and N16, for having smaller *S-HopIDs* compared to *S-HopID* of the source node, are declared as the candidates for the Next Hop. For having smaller backup link than the n16, N15 wins this competition. So, the source node will forward the Data packet with N15 *NodeID* as a Next Hop. In some cases, if two nodes reply with the same minimum *HopIDs*, the node, which will firstly reply, will be eligible for energy concerns, and packets over multiple short hops are preferred instead of long links [44].

A more general scenario is shown in figure 3.10, describing how to forward the Data packets when Courier node is also available. The *HopIDs* of all the nodes are shown in the table 3. Two nodes, N62 and N65, have Data packets in order to send to the destination. N62 will ask its neighbors for their *HopIDs*. N51 and N74 in its range in turn will reply with their *HopIDs*. *C-HopID* of its neighbor nodes shows that the unavailability of Courier node makes the packet to be forwarded to the N51 and afterward N51 will repeat the same procedure, select N45 as the Next Hop and continue until the Data packet reaches the surface Sink.

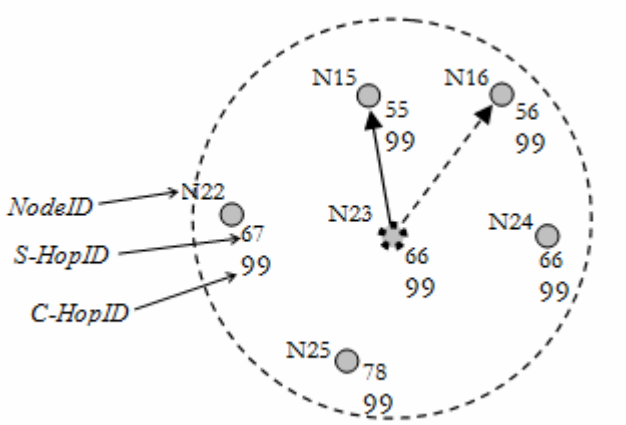

Figure 3.9 Selecting the next hop for the data delivery

In the second case, N65 also has Data packet causing it to ask its neighbor nodes for their *HopIDs*. Here N53, N74 and N75 which are in its range will reply their *HopIDs*. From the replied *HopIDs* it is clear that Courier node is also available and N75 that has smaller *C-HopID* will be considered as the Next Hop. N75 furthermore will repeat the same process and continue until Data packet reaches the Courier node "N90" which has "19" as the smallest *C-HopID*.

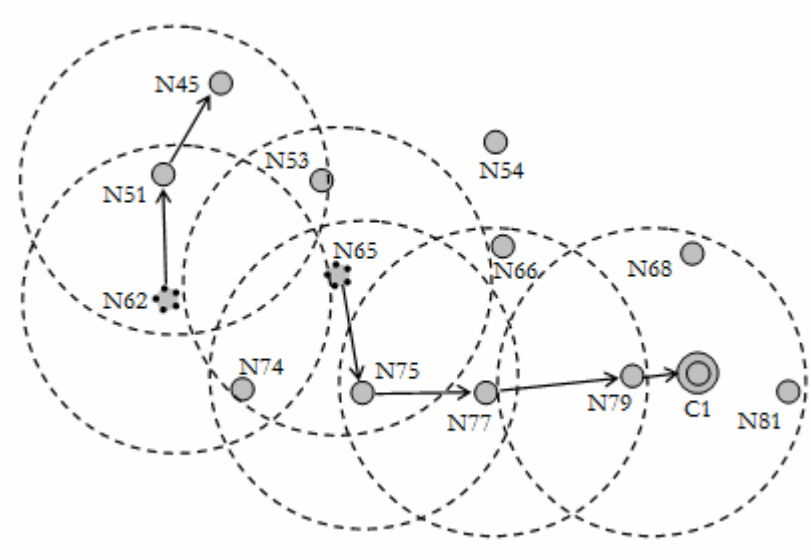

Figure 3.10 Selecting the next hop when courier node is available

But it is not possible that the source node can get the response from the neighbor nodes with the smaller *HopIDs* any time. In such cases, when a source node cannot get such a response, especially in sparse areas, it will wait for a defined amount of time and will reattempt to get the *HopIDs*. Here, the waiting time can fluctuate according to past history. After the third attempt, if the results remain the same, no node accordingly replies from the upper layers. Then it will prefer to forward the Data packet towards a node on the same layer with the *HopID* value nearly or equal to its own *HopID*. If there is still no node eligible as the Next Hop, the packets can be forwarded towards the lower layer nodes. In some worst cases, if source node can not communicate with any other node, it, as the last alternative, can remove the restriction of suggested range of 800 meter [90].

From the earlier scenario it clearly shows that delivery ratios for H2-DAB are not completely determined by the network density or sparseness. We can improve the performance further according to the nodes densities. If the network is dense, for simplicity we can define that only the nodes with smaller *HopIDs* can reply after receiving the *Inquiry Request*. If the network is sparse, all the nodes receiving the *Inquiry Requests* will reply then.

3.3.6 Calculating the Waiting Time

When a source or forwarding node cannot find next hop from upper layers after the first attempt, the source will make other two attempts to send current data packets towards the upper layers before sending them towards the nodes belonged to the same or lower layers. Waiting time of both intervals can be a value between [0, 100], where 0 mean no waiting time and 100 can be the maximum waiting in the worst situations. Before going for the second attempt, it will wait t_1 time, depending on the number of nodes replied in the first inquiry request. t_1 here can be calculated as

$$t_1 = \frac{C}{n_1 + 1} \tag{1}$$

where C is a constant that has the maximum value of the waiting time and n_1 is the number of neighbor nodes replied in the first inquiry request.

After the second trying, if it still cannot find any node from the upper layers, it will wait t_2 time depending on two parameters. First, the number of nodes replied after the second inquiry request and second, the difference between the number of nodes in the first and second inquiry request, then getting the average of these parameters.

$$t_2 = \frac{\left[\frac{C}{|n_2 - n_1| + 1} + \frac{C}{n_2 + 1}\right]}{2} \tag{2}$$

where C is a constant and n_2 is the number of neighbor nodes replied in the 2nd request. From equation (1) and (2) it is clear that the waiting time is determined by the availability of neighbor nodes and frequency of change in them. Both of these parameters are inversely proportional to the waiting time in that waiting time begins to decrease when any of one or both parameters start to increase.

3.3.7 Route Updating and Maintenance with Courier Nodes

Due to buoyancy control, nodes slightly fluctuate in vertical directions, but they can still easily move in horizontal directions with the water currents. For this, due to such movements, a node can change its neighbors both with upper and lower layers. But still, the addresses can be used, as address of any node will remain smaller from the addresses of lower nodes, and larger from the upper nodes. Hence, the address of any node shows how many layers or hops that have to move to reach any surface Sinks or Courier node. Simply, it is not necessary to reach any specified sink since the Data packets can be delivered to any of destination.

As it is mentioned earlier in problem statement that the oil/gas field monitoring application is considered, where most of the time the samples of the data are required from time to time. During one interval, the data can be detected and delivered in a short time and then to save the energy, nodes can go to sleeping mode or shut the transceiver down until the next sampling interval. These sensing and sleeping intervals can be scheduled according to the situations and requirements of the network.

For the next interval, nodes will be assigned using new *HopIDs* according to the new locations till the next sampling interval. These intervals based *HopIDs* can give better results only when these intervals are not very long. If sampling intervals are long, the performance of the network with the node movements can decrease with the same *HopIDs*. However, being not serious problem, this situation could be easily coped with. For long term missions; these *HopIDs* have a *time property*, denoting the validity time for them. The larger the *HopID* lifetime is, the longer time for which these can continue to work. When the lifetime exceeds the threshold value EXPIRE, the *HopIDs* will be reset to “99” throughout the network, and again new *HopIDs* will be assigned with the help of new Hello packets. If any node has packets to be sent before the *HopID* expires, it will wait and can forward these packets after getting the new *HopID*. A fixed life for every interval will not be suitable at different operational conditions. Subsequently, the life time of every interval will be dynamic as it depends on the environment. Determining a good life time, the duration of interval is important for good performance of the network. At one side, getting the new *HopIDs*

in such a way helps to handle the problem of even small vertical movements, as the next time nodes will get new *HopIDs* according to their new positions. At the same time, it makes the buoyancy control more flexible, as violations of strict layering structure can be acceptable.

3.4 Analytical Model for Energy Consumption for H2-DAB

For this purpose, N number of sensor nodes are considered and uniformly deployed in a monitored area A where these nodes have formed layers from surface to bottom. Each sensor node has an initial energy level ε unit and the energy consumption is considered for data packet as well as control packets such as *Inquiry Request* and *Inquiry Reply*. In the scenario, only the nodes with smaller *HopID* will send the *Inquiry Reply*. Both control packets are of same size and consume very little energy as compared to data packet. For our model, E_d is the complete energy required for forwarding a data packet from one layer to the other which includes e_d, the energy consumed for sending data packet and e_c, energy consumed for sending the control packet. While, $E_{i \rightarrow s}$ is the total energy consumed when i data packets are forwarded towards the sink, from i layers (each layer generate one data packet).

3.5 Energy Consumption in the Network

The whole depth is divide into m layers and each layer contains n number of nodes while total D data packets are generated in the network, where each node generates (D/N) data packets. We use k to present (D/N) data packets. The energy consumption at i-th layer is denoted as E_i and life time of this layer can be represented as T_i, while $T_{i/n}$ is the life time of each sensor node that belongs to this layer. All the layers can receive data packets from the below layers and forward these as well as their own generated data packets towards upper layers. It is assumed that that *HopIDs* are already assigned as it required only once for long intervals. For simplicity, it is considered that the energy consumption during the sending process. While, energy required for the reception process for underwater wireless sensor networks is usually 100 times smaller than transmitting [52].

The energy consumption is tested with two scenarios: (I) with static nodes and (II) with mobile nodes.

3.6.1 Energy Consumption with Static Nodes (Best Case)

For static scenario, every node will not only send only one *Inquiry Request* but also get single *Inquiry Reply*. Following that, *NodeID* of replying node is saved in the routing table and will be used as a next hop for all the remaining data packets.

For the first time, energy consumption for a single data packet from any lower layer to next upper layer is

$$E_d = 2e_c + e_d \quad (3)$$

Where, "$e_c + e_d$" is the consumption from current layer which has data packet. At first it sends an *Inquiry Request* and then forwards the data packet after receiving the *Inquiry Reply*. The remaining "e_c" meanwhile is the consumption from upper layer when a node replies with the *Inquiry Reply*. First we consider the case, when data packet is generated at a node in the first layer and it forwards directly to the sink "S". After that, data packet is similarly generated at the second layer and forwarded towards the sink through the first layer and so on. The effect of energy consumption at each layer can be represented by the following equations.

$$E_{1\rightarrow S} = (e_c + e_d)$$

$$E_{2\rightarrow S} = (2e_c + 2e_d) + (e_c + e_d)$$

$$E_{3\rightarrow S} = (2e_c + 3e_d) + (2e_c + 2e_d) + (e_c + e_d)$$

$$\vdots$$

$$E_{m-1\rightarrow S} = (2e_c + (m-1)e_d) + (2e_c + (m-2)e_d) + \ldots + (2e_c + 2e_d) + (e_c + e_d)$$

$$E_{m\rightarrow S} = (2e_c + m.e_d) + (2ec + (m-1)e_d) + \ldots + (2e_c + 2e_d) + (e_c + e_d) \quad (4)$$

Equation (4) shows how the upper layers are affected when one node at every layer generates a data packet and total *m* data packets are forwarded towards the sink. It is clear that layer 1 processes all *m* data packets and due to that it faces maximum energy consumption ($2e_c + m.e_d$) than any of the other layer. Layer *m* has the least energy consumption ($e_c + e_d$) as it processes only one data packet. Now, when *k* data packets are generated on the same node of each layer, we can represent equation (4) as follows.

$$E_{m.k \rightarrow S} = (2e_c + k.m.e_d) + (2e_c + k.(m-1)e_d) + \ldots + (2e_c + k.2e_d) + (e_c + k.e_d) \quad (5)$$

With the above equation, energy consumption at layer *i* can be calculated as it is to process its own generated data packets as well as energy to forward the data packets of all the lower layers.

$E_i = (m-i)k.e_d + k.e_d + 2e_c$ where $i < m$

We use ρ to denote $k.e_d$. Now we can write,

$E_i = (m-i)\rho + \rho + 2e_c = (m-i+1)\rho + 2e_c$

Life time of layer *i* can be calculated as

$$T_i = \frac{n.\varepsilon}{(m-i+1)\rho + 2ec} \quad (6)$$

From equation (4), we can get the life time of a node at layer *i* as below,

$$T_{i/n} = \frac{\varepsilon}{(m-i+1)\rho + 2ec}$$

3.6.2 Energy Consumption with Mobile Nodes (Worst Case)

When we talk about mobile nodes, equation (3) in worst case becomes

$$E_d = e_d + (n+3)\, e_c \qquad (7)$$

Where, "e_d+3e_c" is the energy consumption from current layer when it for the worst case has to make three *Inquiry Requests*. Now it is possible that no node replies in first two attempts and then after the 3rd request, all nodes have replied from the upper layer. In such case, "$n.e_c$" will be the consumption in the form of *Inquiry Replies*. Here it should be clear that we are considering the worst case in terms of energy, but not in terms of node failure.

Again, we firstly consider the case when the data packet is generated in the first layer and afterward remaining lower layers generate packets and forward them towards the sinks through the upper layers.

$$E_{1 \rightarrow S} = (e_d + 3e_c)$$

$$E_{2 \rightarrow S} = (2e_d + ne_c + 2.3e_c) + (e_d + 3e_c)$$

$$E_{3 \rightarrow S} = (3e_d + 2.ne_c + 3.3e_c) + (2e_d + ne_c + 2.3\, e_c) + (e_d + 3e_c)$$

$$\vdots$$

$$E_{m-1 \rightarrow S} = ((m-1)e_d + (m-2).ne_c + (m-1).3\, e_c) + \ldots + (2e_d + ne_c + 2.3\, e_c) + (e_d + 3e_c)$$

$$E_{m \rightarrow S} = (me_d + (m-1).ne_c + m.3e_c) + \ldots + (2e_d + ne_c + 2.3e_c) + (e_d + 3e_c) \qquad (8)$$

Equation (8) provides same effect for mobile nodes as equation (4) for static nodes. Similarly, when *k* data packets are generated at one node of each layer, then we can represent equation (8) as follows

$$E_{m.k \rightarrow S} = ((2.k.m.e_d - 1) + k(m-1).ne_c + k.m.3e_c) + \ldots$$
$$+ ((4k.e_d - 1) + k.ne_c + 2k.3e_c) + ((2k.e_d 1) + k.3e_c) \qquad (9)$$

For worst situations, we consider the case when sensor nodes try to forward the 2nd packet after receiving the 1st acknowledgment, but for 2nd data packet cannot receive it. It makes 2 tries for every data packet but other than the 1st one, as for 1st data packet, first it will find the next hop and then can forward it. Now, equation (9) can help to calculate the energy consumption at layer *i*, where it not only forwards its own generated data packets but also the data packets of lower layers are forwarded through it.

$E_i = (2k(m-i+1)e_d - 1) + k(m-i)\, ne_c + k(m-i+1)3e_c$ where $i < m$

$E_i = (2k(m-i)e_d - 1) + k(m-i)\, ne_c + k(m-i)3e_c + (2k.e_d - 1) + k.3e_c$

$E_i = (m-i)[(2k.e_d - 1) + k.ne_c + k.3e_c)] + (2k.e_d - 1) + k.3e_c$

We use ρ to denote $(2k.e_d - 1) + k.ne_c + k.3e_c$. Now we can write,

$E_i = (m-i)\,\rho + (2k.e_d - 1) + k.3e_c$

Life time of layer *i* can be calculated as

$$T_i = \frac{n.\varepsilon}{(m-i)\rho + (2k.e_d - 1) + k.3e_c} \qquad (10)$$

Similarly, from equation (10) we can get the life time of a node at layer *i* as follows,

$$T_{i/n} = \frac{\varepsilon}{(m-i)\rho + (2k.e_d - 1) + k.3e_c}$$

Equation (6) and (10) show that, as the value of *i* increases mean we go at deeper, the value of ρ decreases. On the other hand, upper layers face more energy consumption problem as the number of layers starts to increase in the network. However, compared to single sink architecture, the possibility of bottleneck

occurrence around the sink here is decreased due to multi sink architecture. For single sink architecture, only a few nodes around the sink process all the data packets generated in the network, while here this burden is divided on the whole upper layer instead of few nodes. Furthermore, to reduce this effect, Courier nodes that collect data packets directly from the lower layers is introduced so that upper layers process less data packets which ultimately increase the life of the network.

3.7 Hop-by-Hop Reliability

Reliable data transfer is an important task for every type of network, but for UWSN it requires special attention. Generally, two types of approaches are used for this: end-to-end and hop-by-hop. End-to-end approaches are suitable only for the stable environments such as wired networks where most of the time nodes remain static. Nevertheless, for the underwater environments, as we face additional problems of large propagation delays and continuous node movements, conventional end-to-end reliability solutions seem to be inapplicable and could lead to waste of the critical resources. On the other hand, in the absence of any Internet, such as unique addressing mechanism for the UWSN and frequent topology change, it makes hop-by-hop reliable data transfer approach more attractive.

Through experimental results, it is verified that providing the hop-by-hop reliability mechanism is more energy efficient compared to providing end-to-end reliability [110]. In such networks where transmission rates are adjusted at every intermediate node, hop-by-hop control methods react faster which not only results in energy saving but also helps to lessen the congestions with smaller end-to-end delays.

Proposed scheme H2-DAB has many advantages such as not requiring any specialized hardware, no dimensional location information required and easiness in handling node movements without maintaining complex routing tables. For getting more precise results, it furthermore still requires some reliability mechanisms to handle the problem of node or packet loss. For this, we have to focus on the following issues

- Guaranteed delivery with ACKs
- Congestion control
- Identical power consumption to increase the life of sensor nodes

3.7.1 Problem Setting

As previously discussed, it is infeasible to achieve end-to-end reliability due to the frequent network partitioning of UWSNs. For this, it is necessary to focus on the hop-by-hop reliability to make it more responsible for these environments.

In normal Hop-by-Hop ACK (HbH-ACK) scheme; only two nodes are involved, as receiving node will reply the ACK when it successfully receives an error free packet. When the sending node receives the ACK, it can discard the current packet and continue to process the next available data packet. For stable environments such as wired networks, this HbH-ACK has no objection, but for the unstable environments like underwater as nodes can die or get lost due to many reasons, this traditional ACK method is not suitable. Here, the receiving node is the only node in the network, which has the current data packet now that the sending node will discard it after receiving the ACK. For UWSNs, due to the continuous node movements and sparseness, it is possible that the receiving node can not find the next hop for long intervals to reach the destination. During this, it can die due to limited power or any failure can occur due to fouling and corrosion problems. In such cases all the packets held by the current node will be permanently lost because none of the other node maintains the backup of these lost data packets.

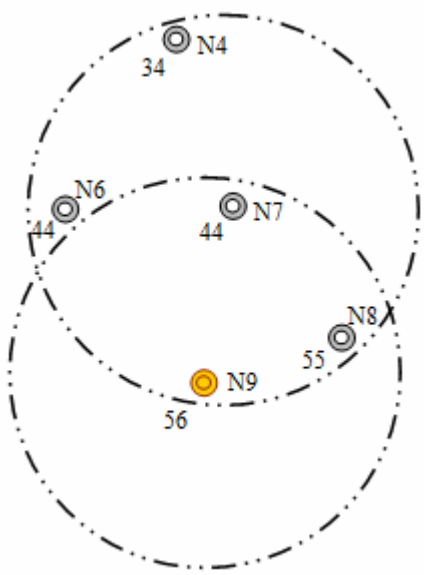

Figure 3.11 Selecting the next hop for data packet forwarding

Figure 3.11 presents the data forwarding method used in our scenario. Here, a source node N9 has a data packet in order to send towards surface sink with its own *HopID* 56. It will ask its neighbors for their *HopID*s, and Nodes N8 and N7 will reply as both of these are in the range of N9. After comparing their *HopID*s, N7 will be declared as a next Hop as its *HopID* is smaller than N8. Now, N7 will repeat the same procedure and N4 will be selected as the next hop for having smaller *HopID* than the N6. This process will continue until the current data packet reaches the destination.

3.7.2 Guaranteed Delivery

In order to increase the network reliability and provide a guaranteed delivery; every single packet in our model is maintained by two sensor nodes in the network. Figure 3.13 further depicts the scenario presented in figure 2.12 according to our ACK model.

Figure 3.13 is further divided, as task is completed in 6 steps. A source node N9 has a data packet in which it wants to forward towards the destination. In step (a) it asks its neighbors about their *HopID*s with the help of any *Inquiry Request* (figure 5

a). After receiving this *Inquiry Request*, the neighbor nodes will compare their *HopID*s with the requesting node's *HopID*, and only those nodes having smaller *HopID* will reply. In step (b) N7 replies with its *HopID* with the help of an *Inquiry Reply*, and then in step (c) data packet are transferred to node N7. After receiving the data packet, N7 will not immediately send an ACK to the sending node N9. It will try to find the next hop node in order to reach the destination, so it will repeat the same process as N9 does in order to get its neighbors *HopID*s as shown in (d). In (e) node N4 replies as its *HopID* is smaller than the N7. When node N7 gets *Inquiry Reply* from N4, it firstly will send ACK towards N9 and forward data packet towards N4. After receiving this ACK, N9 will clear the data packet from its buffer.

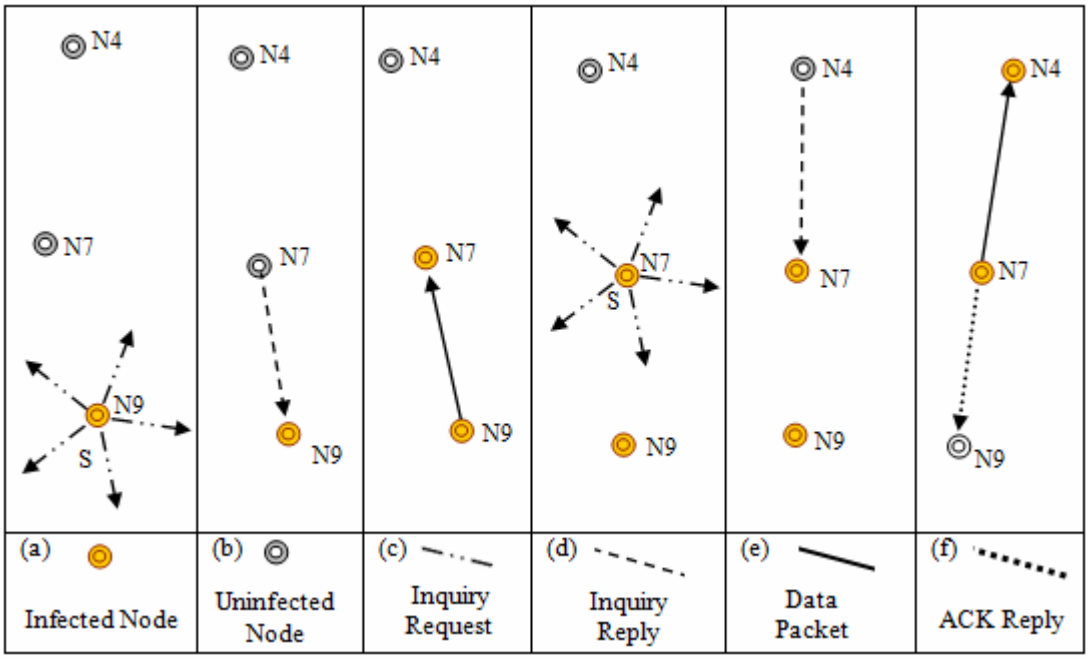

Figure 3.12 Data packet forwarding by using 2H-ACK reliability

From the whole procedure illustrated in figure 3.12, it is clear that in minimum two nodes will have the same copy of a data packet in the network. Due to any unwanted event, if a node is destructed, another copy is still available in the network and will be forwarded after a specified waiting time.

3.7.3 Calculating the Waiting Time

After sending the data packet, the source node *s* will wait without making any further request from neighbors for the same data packet if it does not receive ACK from the data receiving node. Node *n7* will not immediately reply the ACK but it firstly will try to find the next hop. When the next hop is available, it will send ACK to the source *s*. This waiting time at node *s* can be calculated from

$$W = 4t + t_p + \alpha \quad (9)$$

The total waiting time *W* depends on three parameters as shown in (9). *t* is the propagation time for every single hop, t_p is the processing time, and α is the time that *n7* can take in order to find the next hop. Among these parameters, *t* and t_p, can be a constant as propagation speed (1500 m/sec) and processing power of the node in that both are fixed values. The value of α meanwhile can vary according the environmental conditions and depends on network density *d* and speed of node movements *v*. In certain conditions, *n7* sends the inquiry request and then has to wait due to the unavailability of next hop with less *HopID* in the communication range. The effect of *d* can be represented as follows,

$$\alpha \propto 1/d \quad (10)$$

$$\alpha \propto v \quad (11)$$

The value of α will increase with the decrease of *d* and vice versa. With average densities, if requesting node can find neighbor nodes in its communication range, the value of α will be zero. In sparse areas, on the other hand, if requesting node can not find any neighbor node, it has to wait until some other node can come or go in the range of some other nodes due to water movements. In these situations, the value of α depends on *v*. A node generally can move 2-3 m/sec, but these movements are irregular in terms of directions. Due to these movements, some nodes can come and at the same time can leave the coverage area. If we assume $v=0$, the value of α depends only on *d* then.

3.7.4 Congestion Control

Congestions are considered as an important factor when we are talking about the performance of reliable transport protocols [110]. We can achieve a significant reduction for the buffer size requirements when using hop-by-hop approaches under high propagation delays. As already mentioned, H2-DAB is a hop-by-hop routing protocol and uses the idea of *per-contact routing* instead of *source routing* or *per-hop routing*. Hence, it is clear that data generating source node or even any other node on the route can not decide about the next hop from the information available in its own routing table. For this, all the neighbor nodes reply with an *Inquiry Reply* before receiving any data packet and they can be helpful in coping with the problem of congestions and battery usage. For H2-DAB, two fields are used for the *Inquiry Reply*; those are *Node ID* and *HopID*. Now, for the reliability concerns, the *Inquiry Reply* is enhanced from two fields to four fields as shown in Figure 3.13.

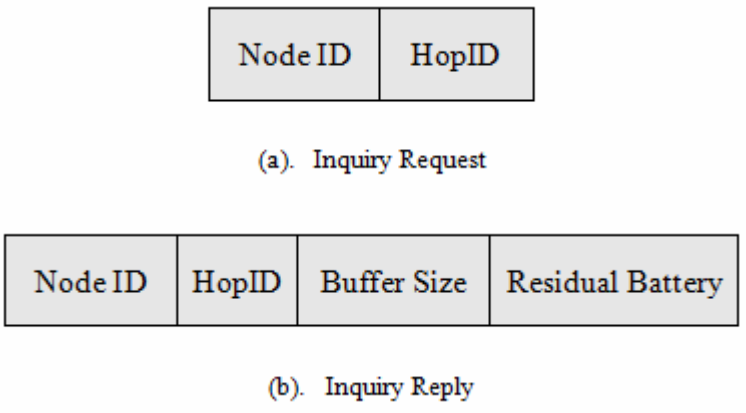

Figure 3.13 Fomat of inquiry request and inquiry reply

The current node having the data packet will send the *Inquiry Request* which has two fields as shown in figure 3.13(a). *Node ID* is a unique ID and will be used when neighbor nodes reply to this request. After receiving this *Inquiry Request*, all the neighbor nodes will compare their *HopID* with the *HopID* of requesting node and only those nodes having smaller *HopID* will reply with an *Inquiry Reply* as shown in figure 3.13(b). When current node receives the inquiry replies from all the neighbors,

it then can reliably decide about the next hop with the help of received information. From Inquiry Reply, first field, *Node ID* will help to send the data packet if this node is selected as a next hop. Next *HopID* field will present the dynamic *HopID* of that node which has replied this *Inquiry Reply*. Then, next both fields provide the information about the resources available at that node. If two nodes replied with the same *HopID* or nearly equal, then on the base of available buffer size and battery level, the current node can choose a suitable node as a next hop. If we can avoid congestion in a node, we can avoid the whole network being congested, which will lead to a reliable data transfer without dropping the data packets. At the same time, the information about the available battery level will help to balance the power consumption for all the nodes, which ultimately will lead to increase the life of the whole network.

3.8 Current Issues and Potential Research Areas

Due to its significant scalability and flexibility compared to traditional ocean monitoring approaches, underwater wireless sensor networks for some reasons are becoming more and more important in underwater data communications for the years to come. From all discussions in the preceding sections, it seems that UWSNs have received much attention with sufficient solutions proposed. However, harsh underwater environment, hardware limitations and complicated application scenarios still pose many challenges to many UWSN researchers. Based on the literature surveyed above, the following potential directions may deserve the attention of the current and future researchers interested in UWSN.

We can not neglect the issue of continuous node mobility, especially in the large scale UWSNs. Many techniques, such as those in [111, 112] have been proposed for mobility prediction and handling in terrestrial mobile and sensor networks. However, these techniques are not suitable for handling underwater mobility due to UWSN's 3-d nature. Mobility prediction is an important issue even more critical for clustered base routing needed to control the network topology. On other notes, the mobility models such as Gauss-Markov Mobility Model and Boundless Simulation Area Mobility Model are used for evaluating different ground based algorithms, which are

not suitable for underwater environment due to its unique characteristics. Although in [113] the authors have accepted this challenge and proposed a mobility prediction model for underwater environments and in [114], a mobility model is also proposed, and the mobility prediction problem is still worth some further investigation. More efforts are needed from the UWSN research community leading us to be ale to present more realistic conditions for better evaluation methods.

Cross layer design helps to increase the performance of WSN by optimizing the interaction between different layers. In fact, many cross layer techniques have been proposed for the terrestrial sensor networks. For UWSN, it however requires further optimization, particularly in the physical layer in which the communication channel is often of poor quality. While research carried out so for on underwater communication protocols has followed the traditional layered approach but performance can be improved by adopting the cross layer design. The objective of this approach is to overcome the shortcomings of traditional layered architecture that lacks the information sharing option across different layers by forcing the network to operate in a suboptimal mode. Presented in [115] is the only published work where the authors have explored the cross layer design to make the efficient use of the bandwidth-limited acoustic channel. Motivated by their works, this emergent design trend is worth to be followed up to ascertain better underwater wireless communication performance.

Testing is an important issue and most of the evaluations for UWSNs are conducted through simulation due to expensive hardware cost. Until now, very few specialized tools similar to those in [95] are available to more accurately present the underwater phenomena. The well funded research groups must consider the importance of these tools so as to present the underwater scenarios in a more realistic way. Even when these tools are available, whenever possible it is still necessary to conduct real experiments because of their realistic empirical importance. It is thus deemed necessary to develop inexpensive transceiver or modems for underwater communications in the future [116].

Moreover, following issues are also suggested for near future research. With poor quality channel and long propagation delays, data link protocols and its related issues

like optimization of data packet size and data packet segmentation can play a vital role. Unfortunately, very little published work like [117, 118] related to packet size optimization and [119] on packet fragmentation is available. More researches in turn are required in order to maximize the channel utilization. Different models like *Urick propagation model* [120] and *Rayleigh fading model* [121] can be helpful to understand and analyze the acoustic channel characteristics like refection, diffraction and scattering of sound. Optimized architecture, possibly of hybrid type, is preferred for efficient data collection and retrieval of large data volumes from these environments is a research issue still left to be resolved. Integration of routing protocols with other system functions including navigation, localization, data collection, compression etc, can help to improve their efficiency. For example, the protocols proposed in [15, 34, 81] use multiple copies of a data packet to increase the reliability and delivery ratios. These proposed protocols are targeted for resource scary underwater environment, where some of the mechanism suggested can be very useful to the destination node after receiving a data packet to inform those intermediate nodes that have the remaining copies.

CHAPTER 4

H2-DAB ALGORITHMS AND FLOW CHARTS

Based on the methodology proposed and described in Chapter 3, this chapter presents the algorithms and flow charts of the basic H2-DAB protocol and H2-DAB with courier nodes as well as for 2H-ACK reliability model. Current chapter is divided into three sections according to the proposed algorithms. Section 1 provides the algorithms and their explanation with the relevant flow charts for the basic H2-DAB protocol. Further, section 2 presents the algorithm of improved H2-DAB with introducing the courier nodes. Finally, section 3 presents the algorithm of 2H-ACK model and its related discussion.

4.1 Algorithms for H2-DAB Without Courier Nodes

The primary goals of H2-DAB routing protocol is to maximizing the delivery ratios, minimizing the end to end delays with optimum energy consumption and completing all these, without requiring specialized network equipment. In this section, we present algorithms that require during the routing process for H2-DAB. Basic H2-DAB complete its task in two phases, during first dynamic HopIDs are assigned with the help of hello packets while during the second phase data packets are forwarded using these HopIDs. For this purpose two routing algorithms are proposed that are provided in this section.

4.1.1 Assigning HopIDs using Hello Packets

To assign HopIDs, hello packets are broadcasted from the surface sinks (S-hp with HopID N_{00}). After receiving the hello packet, current node will extract the HopID from the hello packet ($N_{r\ s}$) and compare it with its own HopID ($N_{p\ q}$). First of all it

will verify about the value of '*r*' which can be '*0*' or not. If '*r=0'* meaning that sensor node has directly received hello packet from the surface sink and, at the same time, this surface sink is not involved in maintaining the HopID of current node. In second case when '$r \neq 0$' mean, hello packet is not directly received from the surface sink. In this case, if *"r is less than p and p is also less than or equal to s"* then in both of these cases, the current node will update its own HopID by replacing the value of '*p'* in '*q'* and value of '*p'* is replaced with the value of '*r+1'*. From both of these conditions, it is clear that hello packets from the surface sink are only being processed when these are from any new sink or when existing HopID is expired, otherwise hello packet from the same sink will be discarded.

If the received hello packet is not satisfying both of the mentioned conditions, it is clear that '$r \neq 0$', meaning that hello packet is received from any sensor node but not directly from surface sink. Further there can be many possibilities, each of which has its respective action but here we are more interspersed in two among them. Firstly, when "*r and s both are less than p"* or "*r is greater than or equal to p and at the same time s is less than q*" then the first option results in where the value of "*p is replaced by r+1 and q is replaced by s+1*". While, the second case results in where the value of '*p'* remains the same; only '*q'* is replaced by *r+1*. Other than these discussed cases, with any other possibility, the values of '*r'* and '*p'* will remain unchanged and current node will maintain its old HopID that it has before receiving the hello packet. However, after any of these cases when current HopID remains unchanged, the current node will replace the value of Max. Hop Count (Max. HC) by 1.

After following all this procedure in order to update own HopID, the current node will forward the hello packet towards the other neighbor nodes. Before going to forward, it firstly will make a decrement of 1 in Max. Hop Count value. After this decrement, if the Hop Count value is greater than zero, the first update then posseses HopID in *hp* and then broadcasts the *hp* otherwise it will be discarded.

4.1.2 Algorithm for Assigning the HopIDs

Hello packets (hp) Broadcasts
From all Sinks with *HopID "$N_{0\,0}$" &*
Max Hop Count = 9

//Hello packet received

1. *Get Received New-HopID "$N_{r\,s}$" from hp*
2. *Get Own-HopID "$N_{p\,q}$"*

3. ***If*** *r = 0 && Sk_{ID} (p) != Sk_{ID} (r) // Existing sink ID != Receiving sink ID*
   Or r != 0 && r < p <= s ***Then***
4. *q ⟵ p*
5. *p ⟵ r+1*

6. ***If*** *r & s < p* ***Then***
7. *p ⟵ r+1*
8. *q ⟵ s+1*

9. ***If*** *r >= p && s < q* ***Then***
10. *q ⟵ r+1*
11. ***Else***
12. *Max Hop Count = 1 // In order to stop further broadcast*
13. ***End If***
14. ***End If***
15. ***End If***

16. *Max. Hop Count - 1*
17. ***If*** *Max Hop Count > 0* ***Then***
18. *Update hp ⟵ Own HopID*
19. *Broadcast hp further*
20. ***Else***
21. *No further broadcast for this hp*
22. ***End If***

4.1.3 Flow Diagram for Assigning the HopIDs

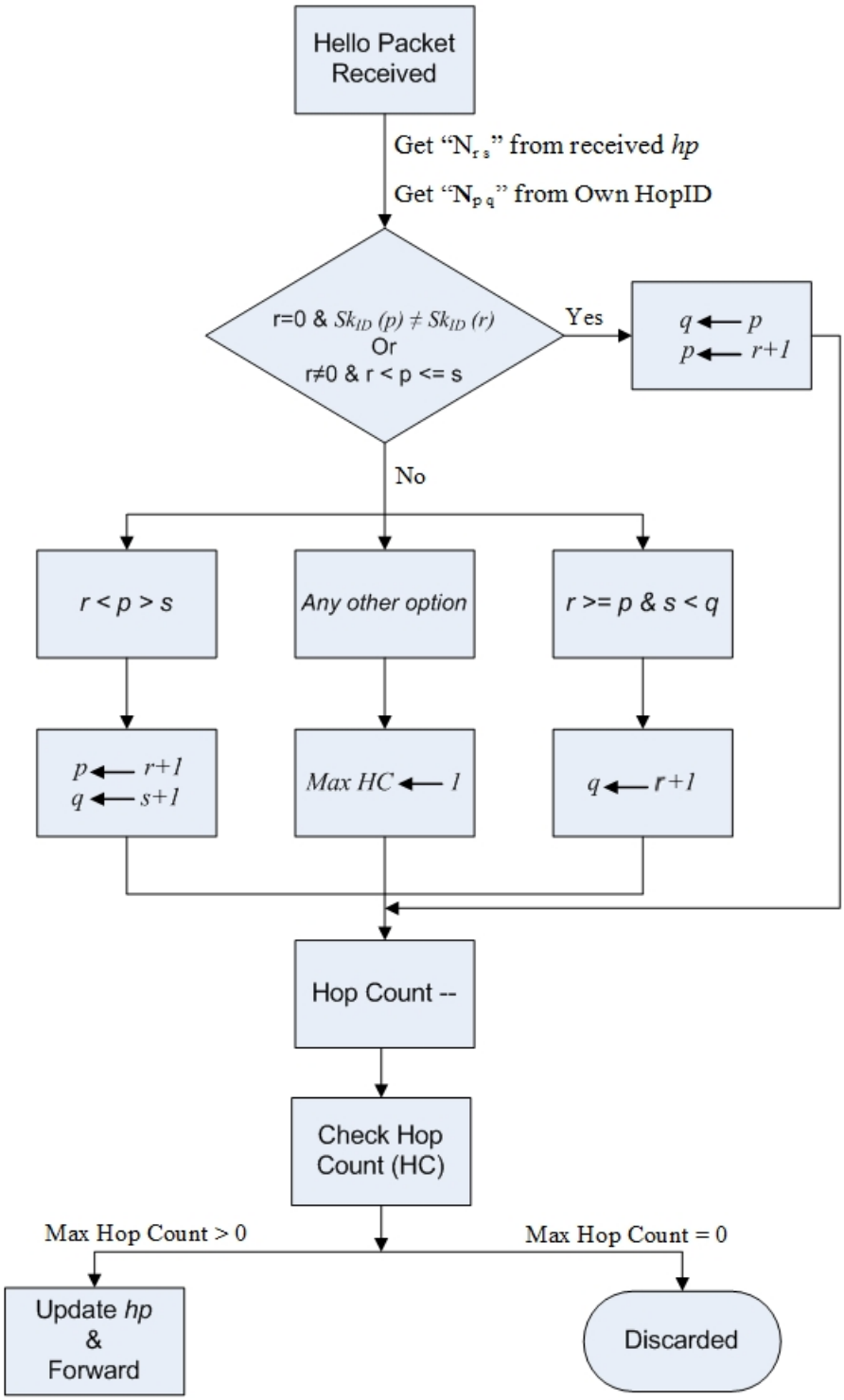

Figure 4.1 Assigning HopIDs without courier nodes.

4.1.4 Forwarding Data Packets using HopIDs

Any sensor node which has data packets ready in order to send, either by self-generating or by receiving from any sensor other node, it firstly will check its own routing table about the availability of the next hop node. If the next hop is available, it will send data packet and wait for the ACK. If the ACK is received, it will continue to send data packets until all the data packets are sent and the relevant ACKs are received successfully. The forwarding of the data packets will be stopped either next hop is expired or ACKs are not received.

Now in both cases, when the next hop field is expired or ACK is not received, the current node will start a request count and send an inquiry request to its neighbor nodes for their HopIDs. After sending this request, it will make an increment of 1 in the Request Count. Having received the Inquiry replies from the neighbor nodes, it then is possible that the existing source node can also send the inquiry reply from where current node has received the data packet. In order to control the loop occurrence, before going to sort out the replied HopIDs, it will decide about the Inquiry reply of its source node whether entertains it or not. For this purpose, it will make a check, if Request Count is less than three, it will discard the inquiry replies received from the source node. If this is not the case and has already requested three times, it will entertain all the replied HopIDs. The purpose of entertaining the Inquiry reply of its own source node is that, as it has made three attempts already and no other node is found out as a next hop so after three tries, the source node can be used as a next hop. Due to continuous water movements, it is possible that during this a more suitable node is reached in the communication range of the source node. After deciding about the source node's Inquiry reply, all the replied HopIDs are sorted out and current node will get smallest HopIDs (SML. HopID). Before going to save this SML-HopID node in its routing table as a next hop, it will check the request count afterward. If the request count is less than four, then it will compare this SML-HopID with its own HopID, while if this is not the case and has already made four requests, this node of SML-HopID can be used as next hop only for buffered data packets but can not be saved in the routing table for future use. On the other hand, if the request count is less than four, it will compare this SML-HopID with its existing own HopID. After this comparison, if SML-HopID is less than own HopID and the node with

SML-HopID will be assigned in the routing table as a next hop - not only for the buffered data packets but also for the future use. While, if this SML-HopID is greater than own HopID then it will wait a defined amount of time and send new Inquiry request for HopIDs.

4.1.5 Algorithm for Forwarding Data Packets

Data packet (*dp*) ready to send (Either own generated or received)

```
1.  If      Next Hop !=  Expire                  // From last column of routing table
2.          Forward dp to Next Hop
3.          If     Ack. Received = Yes
4.                 If      next  dp ready = yes
5.                         go to step 1
6.                 Else
7.                       go to idle or waiting state
8.                 End If
9.          Else
10.                go to step 13
11.         End If

12. Else
13.     Request Count for this dp= 0
14.     Request neighbors for HopIDs
15.     Req. Count+1

16.       If    Own-HopID >=  dp Source HopID  &   Request Count < 3  Then *
                    Discard Inuiry Reply of   Source node
17.             Replied HopIDs put in array
                Sort out and get SML-HopID            (Smallest)

18.       Else
19.             Replied HopIDs put in array
                Sort out and get the SML-HopID
20.       End If

21.        If   Request Count < 4  Then
22.             If    SML-HopID < Own-HopID  Then
23.                   Next Hop ← NodeID of SML-HopID
24.                   go to line 1
25.             Else
26.                 Wait defined amount of time              // Section 4.6
27.                 go to step 14
28.             End If
29.       Else
30.             Forward dp to NodeID of SML-HopID, Until packets in buffer are available
31.       End If
32. End If
```

4.1.6 Flow Diagram for Forwarding the Data Packet

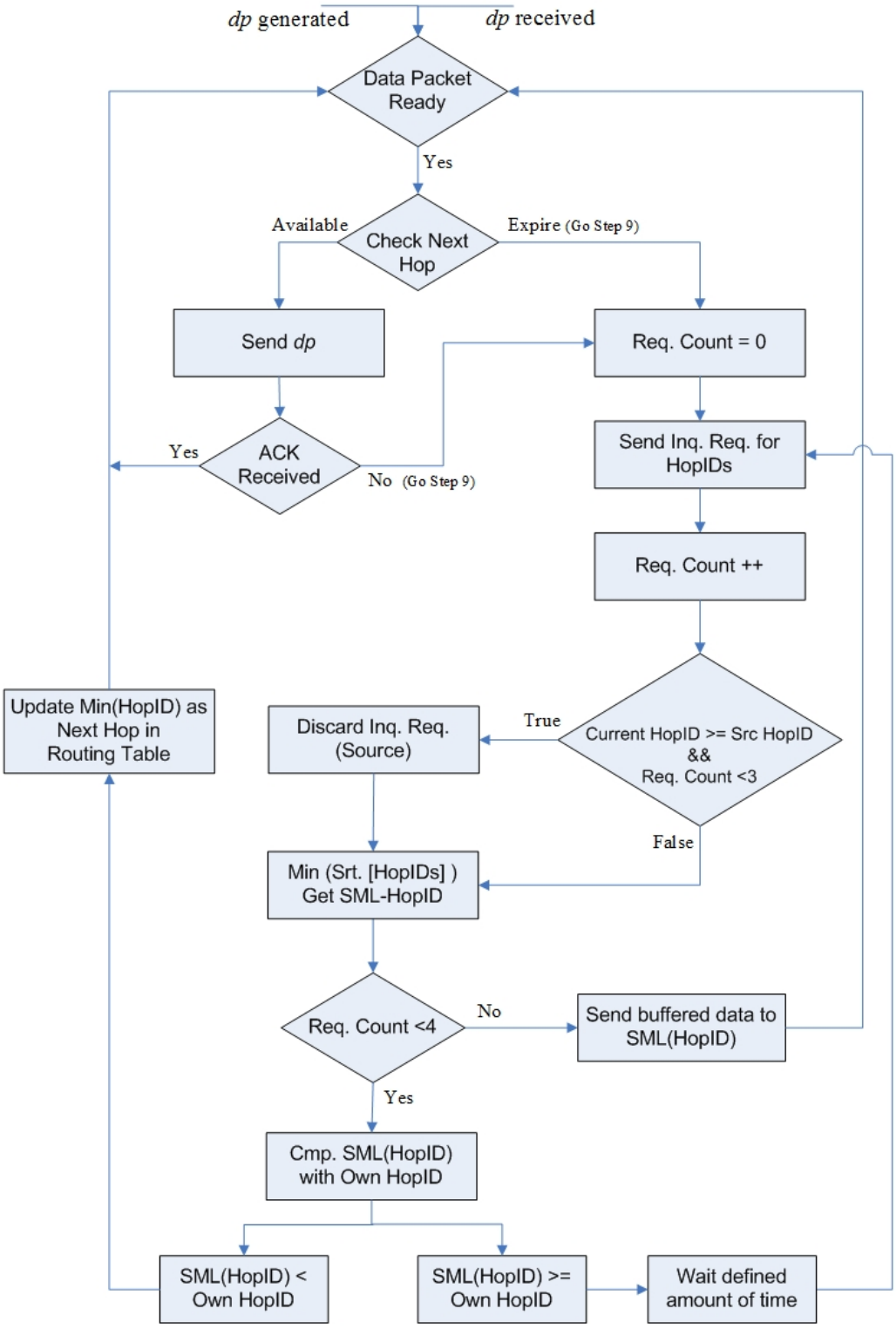

Figure 4.2 Forwarding data packets without courier nodes

4.2 Algorithms for H2-DAB with Courier Nodes

As discussed in chapter 3, the main advantage of H2-DAB is that it completes its task without making any assumption about the location information or requiring special network setup. Further, H2-DAB is improved by introducing courier nodes in order to increase the life of the sensor nodes. Although, our proposed algorithm can work even without introducing a single courier node, or in those areas where courier nodes are not available, however the involvement of courier nodes helps to increase data delivery ratios and reduce the number of communication hops in order to reach the surface sinks which ultimately increase the life of whole network.

Last section provides the algorithms for basic H2-DAB and further this section, present algorithms that require during the routing process for H2-DAB when courier nodes are available in the network. Improved H2-DAB also complete its task in two phases where first dynamic HopIDs are assigned with the help of hello packets and then data packets are forwarded using these HopIDs. For this purpose both routing algorithms are updated accordingly and are discussed in this section.

4.2.1 Assigning HopIDs when Courier Nodes are Available

To assign HopIDs, hello packets are broadcasted from the surface sinks (S-hp with HopID N_{00}) as well as courier nodes (C-hp with HopID N_{19}). When an ordinary node receives any hello packet, it firstly will check whether it is from the surface sink or from courier node. In first case, it is considered that the received hello packet was from one of the surface sinks (*S-hp*). After receiving the hello packet, current node will extract the HopID from the hello packet ($N_{r\,s}$) and compare it with its own HopID ($N_{p\,q}$). First of all it will verify about the value of '*r*' which can be '*0*' or not. If '*r=0*' meaning that sensor node has directly received hello packet from the surface sink and, at the same time, this surface sink is not involved in maintaining the HopID of current node. In second case when '$r \neq 0$' mean, hello packet is not directly received from the surface sink. In this case, if *"r is less than p and p is also less than or equal to s"* then in both of these cases, the current node will update its own HopID by replacing the value of '*p*' in '*q*' and value of '*p*' is replaced with the value of '*r+1*'. From both of

these conditions, it is clear that hello packets from the surface sink are only being processed when these are from any new sink or when existing HopID is expired, otherwise hello packet from the same sink will be discarded.

If the received hello packet is not satisfying both of the mentioned conditions, it is clear that '$r \neq 0$', meaning that hello packet is received from any sensor node but not directly from surface sink. Further there can be many possibilities, each of which has its respective action but here we are more interspersed in two among them. Firstly, when "*r and s both are less than p*" or "*r is greater than or equal to p and at the same time s is less than q*" then the first option results in where the value of "*p is replaced by r+1 and q is replaced by s+1*". While, the second case results in where the value of '*p*' remains the same; only '*q*' is replaced by r+*1*. Other than these discussed cases, with any other possibility, the values of '*r*' and '*p*' will remain unchanged and current node will maintain its old HopID that it has before receiving the hello packet. However, after any of these cases when current HopID remains unchanged, the current node will replace the value of Max. Hop Count (Max. HC) by 1.

After following all this procedure in order to update own HopID, the current node will forward the hello packet towards the other neighbor nodes. Before going to forward, it firstly will make a decrement of 1 in Max. Hop Count value. After this decrement, if the Hop Count value is greater than zero, the first update then posseses S-HopID in *S-hp* and then broadcasts the *S-hp* otherwise it will be discarded.

Now it is considered that if the received hello packet is from the courier node (*C-hp*) then there will be a more simple case as compared to the handling of surface sink hello packet (*S-hp*). In this case, the receiving node will follow the same procedure and extract the HopID from the hello packet ($N_{m\ n}$) and compare it with its own HopID ($N_{x\ y}$). If the value of '*m*' is less than '*x*', '*x*' will be replaced by *m+1*, while in any of the other case, the value of '*x*' remains the same but the Max Hop Count (Max. HC) value is replaced by 1. After processing courier hello packet, the current node will forward this hello packet as it does for the sink hello packet. For this, similarly it firstly will make a decrement of 1 in Max HC and after this decrement if the hop count value is greater than zero, then first update owns *C-HopID* in *C-hp* and broadcasts the courier hello packet.

4.2.2 Algorithm for Assigning the HopIDs

Hello packets (hp) Broadcasts
From all Sinks ***(S-hp)*** *with* *HopID "$N_{0\,0}$" &*
Max Hop Count = 9
From all Courier ***(C-hp)*** *with* *HopID "N_{19}" &*
Max Hop Count = 3

//Hello packet received

1. ***If*** *Packet Type =* ***S-hp***
   Get Received S-HopID "$N_{r\,s}$" from S-hp
   Get Own S-HopID "$N_{p\,q}$"
2. ***If*** *r = 0 && Sk_{ID} (p) != Sk_{ID} (r) // Existing sink ID != Receiving sink ID*
   Or r != 0 && r < p <= s ***Then***
3. *q ⟵ p*
4. *p ⟵ r+1*
5. ***If*** *r & s < p* ***Then***
6. *p ⟵ r+1*
7. *q ⟵ s+1*
8. ***If*** *r >= p && s < q* ***Then***
9. *q ⟵ r+1*
10. ***Else***
11. *Max Hop Count = 1 // In order to stop further broadcast*
12. ***End If***
13. ***End If***
14. ***End If***
15. *Max. Hop Count - 1*
16. ***If*** *Max Hop Count > 0* ***Then***
17. *Update S-hp ⟵ Own S-HopID*
18. *Broadcast S-hp further*
19. ***Else***
20. *No further broadcast for this S-hp*
21. ***End If***
22. ***End If***
23. ***If*** *Packet Type =* ***C-hp***
    Get Received HopID "$N_{m\,n}$" from C-hp
    Get Own HopID "$N_{x\,y}$"
24. ***If*** *m < x* ***Then***
25. *x ⟵ m+1*
26. ***Else***
27. *Max Hop Count =1 // In order to stop further broadcast*
28. ***End If***
29. *Max Hop Count - 1*
30. ***If*** *Max Hop Count >0* ***Then***
    Update C-hp ⟵ Own C-HopID
    Broadcast C-hp further
31. ***Else***
32. *No further broadcast for this C-hp*
33. ***End If***

4.2.3 Architecture of Assigning HopIDs

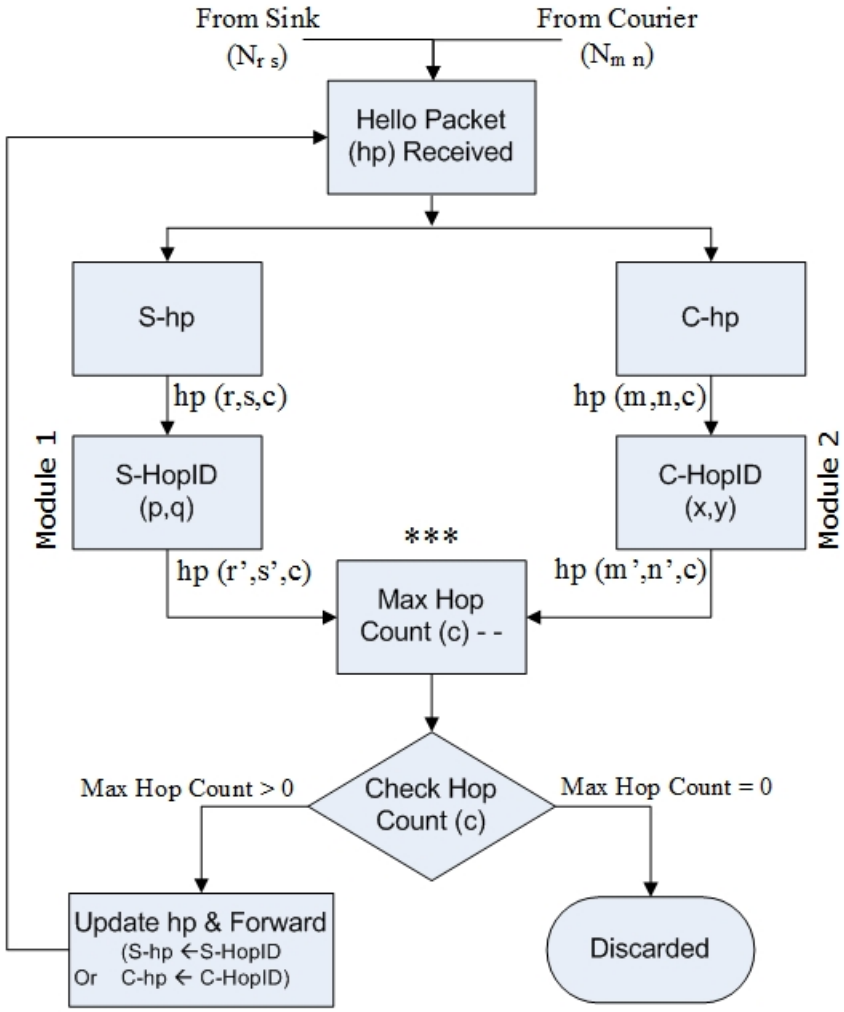

Figure 4.3 Assigning HopIDs when courier nodes are available

4.2.3.1 Architecture of Module-1

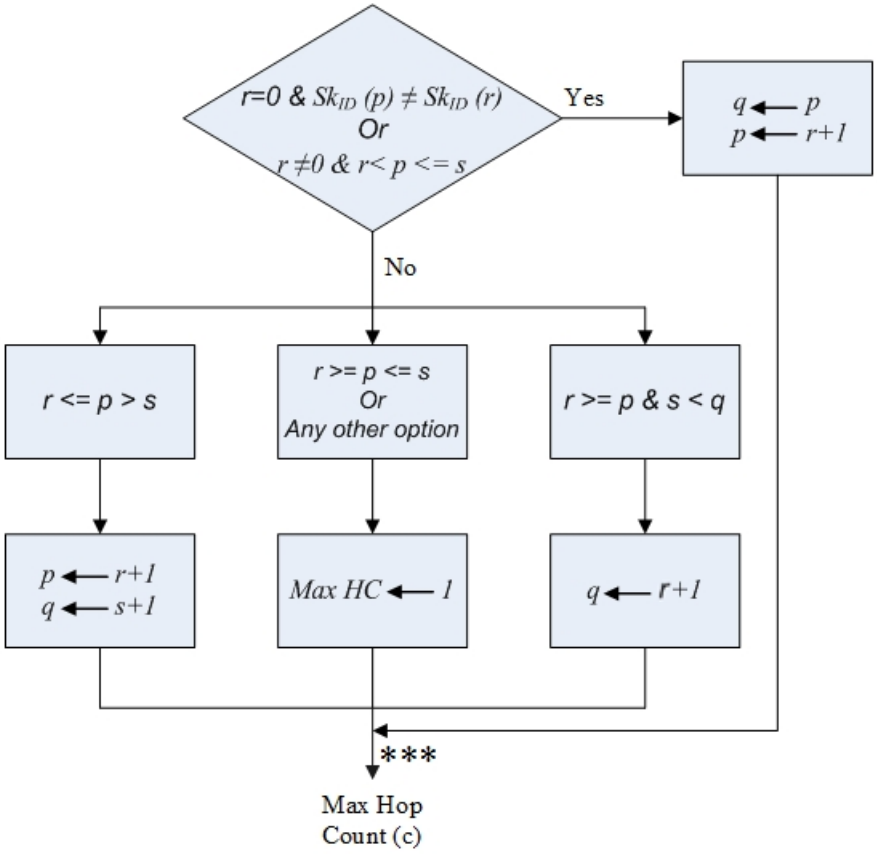

Figure 4.4 Module-1 of assigning HopIDs when courier nodes are available

4.2.3.2 Architecture of Module-2

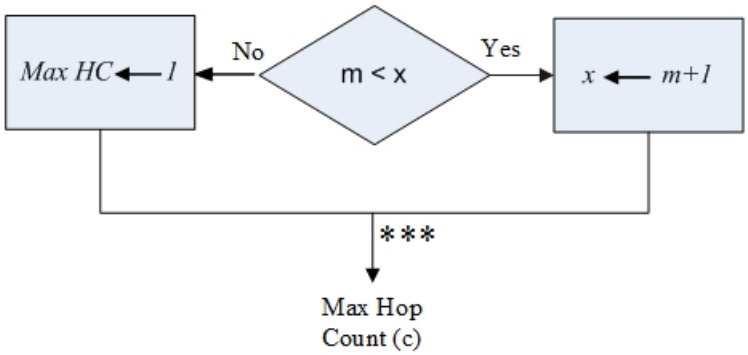

Figure 4.5 Module-2 of assigning HopIDs when courier nodes are available

4.2.4 Data Forwarding when Courier Nodes are Available

Any sensor node which has data packets ready in order to send, either by self-generating or by receiving from any sensor other node, it firstly will check its own routing table about the availability of the next hop node. If the next hop is available, it will send data packet and wait for the ACK. If the ACK is received, it will continue to send data packets until all the data packets are sent and the relevant ACKs are received successfully. The forwarding of the data packets will be stopped either next hop is expired or ACKs are not received.

Now in both cases, when the next hop field is expired or ACK is not received, the current node will start a request count and send an inquiry request to its neighbor nodes for their HopIDs. After sending this request, it will make an increment of 1 in the Request Count. Having received the Inquiry replies from the neighbor nodes, it then is possible that the existing source node can also send the inquiry reply from where current node has received the data packet. In order to control the loop occurrence, before going to sort out the replied HopIDs, it will decide about the Inquiry reply of its source node whether entertains it or not. For this purpose, it will make a check, if Request Count is less than three, it will discard the inquiry replies received from the source node. If this is not the case and has already requested three times, it will entertain all the replied HopIDs. The purpose of entertaining the Inquiry reply of its own source node is that, as it has made three attempts already and no other node is found out as a next hop so after three tries, the source node can be used as a next hop. Due to continuous water movements, it is possible that during this a more suitable node is reached in the communication range of the source node. After deciding about the source node's Inquiry reply, all the replied HopIDs are sorted out and current node will get minimum HopIDs of both types of hello packets (*Min. S-HopID* and *Min. C-HopID*). Further, smallest HopID (*SML-HopID*) is found out among both of these minimum HopIDs. Before going to save this SML-HopID node in its routing table as a next hop, it will check the request count afterward. If the request count is less than four, then it will compare this SML-HopID with its own HopID, while if this is not the case and has already made four requests, this node of SML-HopID can be used as next hop only for buffered data packets but can not be

saved in the routing table for future use. On the other hand, if the request count is less than four, it will compare this SML-HopID with its existing own HopID. After this comparison, if SML-HopID is less than own HopID and the node with SML-HopID will be assigned in the routing table as a next hop - not only for the buffered data packets but also for the future use. While, if this SML-HopID is greater than own HopID then it will wait a defined amount of time and send new Inquiry request for HopIDs.

4.2.5 Algorithm for Forwarding Data Packet

Data packet (*dp*) ready to send (Either own generated or received)

```
1.  If    Next Hop !=  Expire                    // From last column of routing table
2.        Forward dp to Next Hop
3.        If    Ack. Received = Yes
4.              If    next  dp ready = yes
5.                    go to step 1
6.              Else
7.                    go to idle or waiting state
8.              End If
9.        Else
10.             go to step 13
11.       End If

12. Else
13.     Request Count for this dp= 0
14.     Request neighbors for HopIDs
15.     Req. Count+1

16.     If    Current HopID >=  dp Source HopID  &   Request Count < 3   Then *
                Discard Inuiry Reply of   Source node
17.           Replied HopIDs put in array
              Sort out and get the Minimum of both S-HopID and C-HopID
                    (Min. S-HopID & Min. C-HopID)

18.     Else
19.           Replied HopIDs put in array
              Sort out and get the Minimum of both S-HopID and C-HopID
                    (Min. S-HopID & Min. C-HopID)
20.     End If

21.           Compare (Min. S-HopID) & (Min. C-HopID) and get Smaller
                    (SML-HopID)

22.     If    Request Count < 4  Then
23.           If    SML. HopID < Own HopID  Then
24.                 Next Hop ◄— NodeID of SML. HopID
25.                 go to line 1
26.           Else
27.                 Wait defined amount of time
28.                 go to step 14
29.           End If
30.     Else
31.           Forward dp to NodeID of SML. HopID, Until packets in buffer are available
32.     End If
33. End If
```

---

* Is current *dp* received from upper layer? If so then don't consider the inquiry reply from that source node.

4.2.6 Flow Chart of Forwarding Data Packet

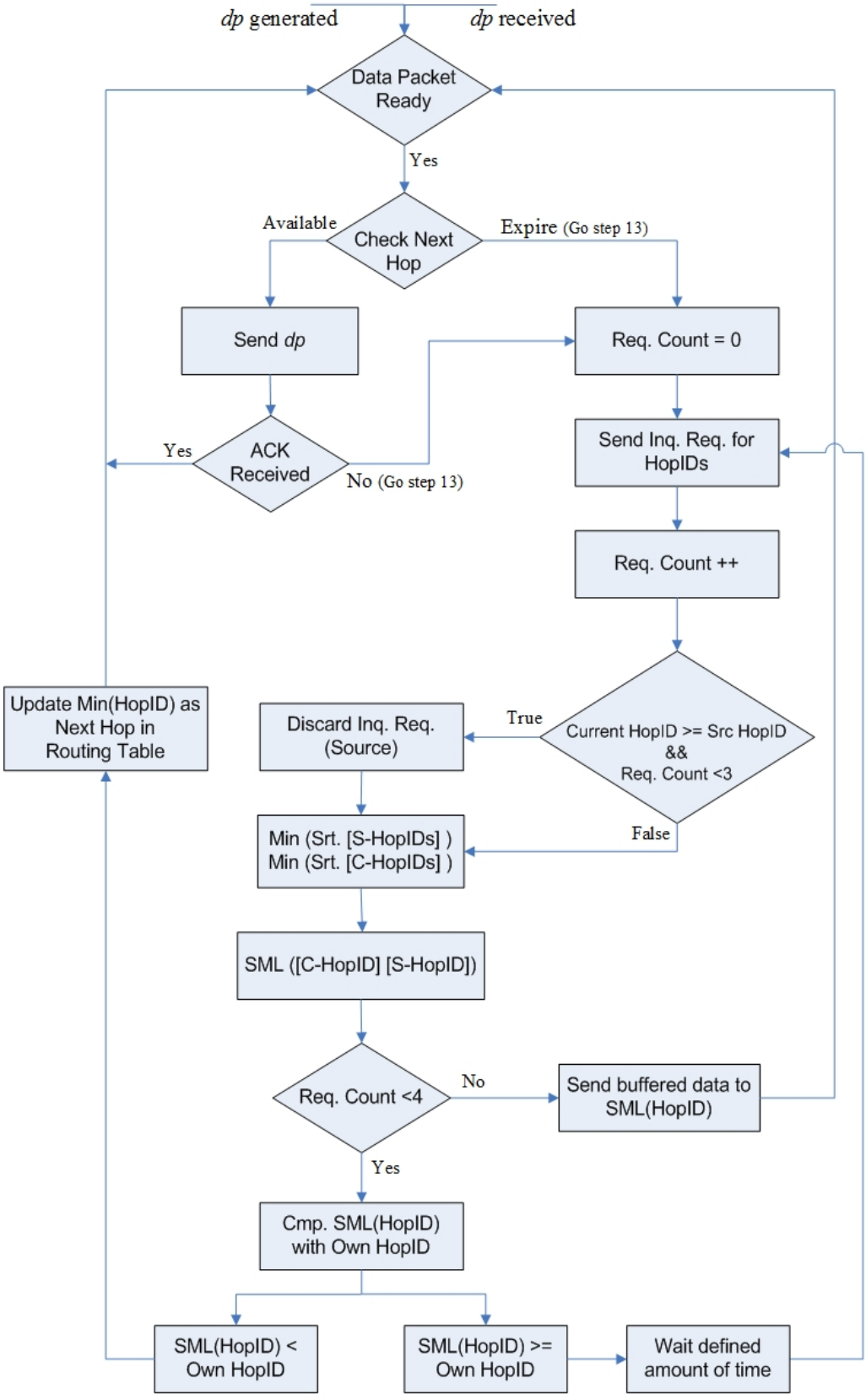

Figure 4.6 Forwarding data packets when courier nodes are available

4.3 Reliability Model for H2-DAB

As described earlier, it is unfeasible to achieve end-to-end reliability due to frequent network partitioning of UWSNs. We have to focus on the hop-by-hop reliability in order to make it more responsible for these environments. In typical Hop-by-Hop ACK (HbH-ACK) scheme; only two nodes are involved, as receiving node will reply the ACK when it receives an error free packet successfully. When the sending node receives the ACK, it can discard the current packet and continue to process the next available data packet. For stable environments like wired networks, this HbH-ACK has no problems, but for the unstable environments like underwater, where nodes can die or get lost due to many reasons, this traditional ACK method becomes less suitable. The receiving node is the only node in the network, which has the current data packet because the sending node will discard it after receiving the ACK. For UWSNs, due to continuous node movements and sparseness, it is possible that, the receiving node can not find the next hop for long intervals in order to reach the destination and during this, it can die due to limited power or any failure can occur due to fouling and corrosion problems. In such cases all the packets held by the current node will be lost permanently because none of the other node maintains the backup of these lost data packets. This section includes the algorithm of 2H-ACK reliability model proposed for the H2-DAB in order to increase its reliability.

4.3.1 2H-ACK Reliability Model for H2-DAB

Current node which has data packet sends an inquiry request to its neighbor nodes for their HopIDs. This request for HopIDs is sent at a predefined bit rate (R). When it receives the Inquiry replies from the neighbor nodes, all the replied HopIDs are sorted out and current node will reach smallest HopIDs (SML. HopID). After finding this SML. HopID, current node will compare this HopID with own HopID. Further, here can be two possibilities (1) First it is considered the possibility when the SML. HopID is less than own HopID, then in this case it will reply an ACK towards the source node from where it has received the current data packet. After replying the ACK, it starts to calculate the best possible size of data packet in current situation (packetsize). In order to find an optimized data packet size according to different conditions,

current node will follow the steps presented in line 7. After calculating a suitable packetsize, current node will assemble the data packet *dp* and forward this optimized packet towards the minimum HopID. While in second case when current node itself is the source node for current data packet, it will start to immediately calculate packetsize and will forward it towards the SML. HopID without replying any ACK. In worst situation, when it can not find a suitable next hop with less HopID than own, it will wait for a defined amount of time and restart this procedure from step 1.

4.3.2 Algorithm for 2H-ACK Reliability Model.

/*Three data sets denoted as F_3, F_4, and F_5 are obtained from Fig.3, Fig.4, and Fig.5 respectively */
/* Source node and sink node are of homogeneous type */
/* Data packet (*dp*) ready to send */
1. Source node: send (*request HopID*) to the neighbors with predefined bit rate (*R*)
2. Neighbors: ACKs and return (*HopIDs*)
3. Source: with returned *HopIDs*
   Sort_out and get Smallest HopID *(SML. HopID)*
4. ***If*** *SML. HopID < Own HopID* ***Then***
5. ***If*** *Current node is not Source node* ***Then***
6. *send ACK to Previous Hop*
7. {
   BER (ρ);
   distance (*d*);
   with ρ indexed into F_3 to acquire $N_{opt\ 1}$;
   with *dR* product indexed into F_4 to acquire $N_{opt\ 2}$;
   N_{opt} := average($N_{opt\ 1}$, $N_{opt\ 2}$);
   with N_{opt} indexed into F_5 to acquire the energy efficiency (η);
   check: *difference* between η and η_{opt} from F_5;
   If (*difference*) < (5%) then
   packetsize := N_{opt}
   Else
   with ρ indexed into F_5 to obtain packet size (*N*) corresponds to max η ;
   packetsize := average(*N*, N_{opt});
   End If
   }
8. Assemble *dp* with *packetsize*
9. Forward *dp* to *SML. HopID*
10. ***Else***
11. Assemble *dp* with *packetsize*
12. Forward *dp to SML. HopID*
13. ***End If***
14. ***Else***
15. Wait defined amount of time
16. Go to step 1
17. ***End If***

4.3.3 Flow Chart of 2H-ACK Reliability Model

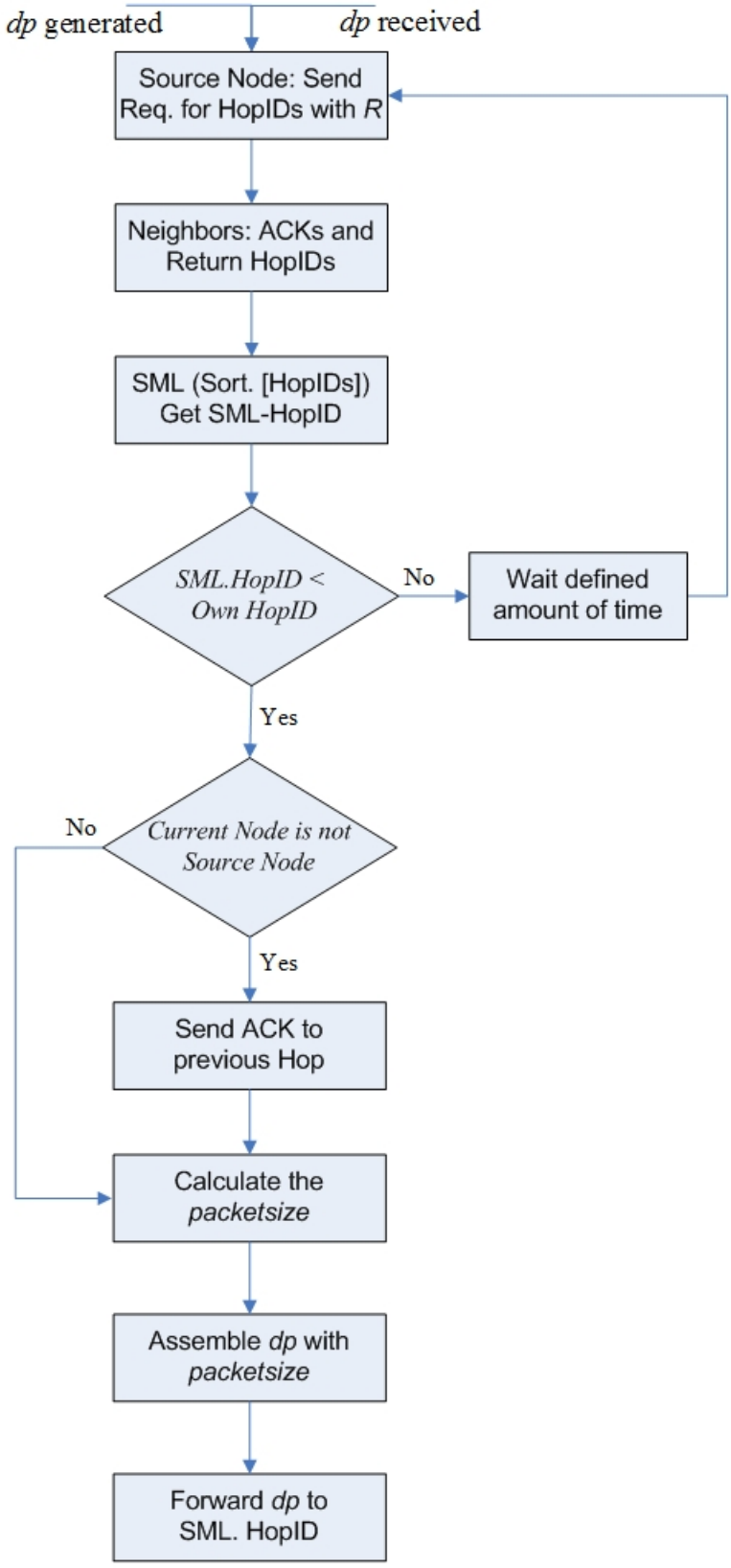

Figure 4.7 Flow chart for 2H-ACK reliability model.

CHAPTER 5

PERFORMANCE EVALUATION OF H2-DAB

In this chapter, simulation results are presented in order to evaluate the performance of H2-DAB and 2H-ACK algorithms, discussed earlier in chapters 3 and 4. We begin this chapter by describing the simulation environment and evaluation criteria. In section 2, we evaluate our proposed routing scheme according to different parameters including node mobility, interval life and courier nodes as well as with different offered loads. Further in sections 3 and 4, we provide the comparisons of H2-DAB with DBR, and with VBF. Section 5 elaborates the effect of dynamic packet size with different parameters. Lastly, section 6 presents the performance evaluation of proposed 2H-ACK reliability model with general hop-by-hop acknowledgment technique.

For performance evaluation purpose, Aqua-Sim [95] a NS-2 based network simulator is used, developed especially for the underwater environments. It supports three-dimensional deployment as well as continuous mobility according to the underwater environment. Moreover, it can not only simulate packet collisions and acoustic signal attenuation, but it also supports a complete protocol stack from physical layer to application layer.

5.1 Simulation Settings

In this section, we evaluated the performance of the H2-DAB through extensive simulations. We deployed 350 ordinary sensor nodes (some anchored at the bottom) in an area of 1500×800 m^2. The process of data delivery was completed with the help of 8 layers from bottom to surface. The transmission range of sensor nodes was varied from 100 m to 150 m, where the average depth and width of every layer were defined

at 100 m, while surface sinks were also deployed at a distance of 100 m.

The deployment of Courier nodes is optional as our routing protocol can complete its task without their presence. However, for better resource usage, we used a small number of Courier nodes, either 1% or 2 % of all the sensor nodes. Ordinary sensor nodes can move freely with the water currents in the horizontal direction at 2 and 4 m/sec and will return back after reaching the defined boundary, while vertical movements were not considered during the simulation. Any node in the network can generate data packet, but the nodes anchored at the bottom will generate half of the total data packets generated in the network. The size of the Hello packets was assumed to be very small compared to the data packets. Every Hello packet will consume 1% of the network resources as compared to every data packet.

Power consumption varies for different routing events; in this simulation we assumed the consumption of 1 energy unit during transmission and 0.02 units for receiving the data packet. In order to prevent data collisions, a node can transmit data when no other transmission is detected in its collision domain. The medium access control (MAC) protocol is based on the IEEE 802.11 and traffic sources are Constant Bit Rate (CBR) with 512 bytes per packet. A total of 500 data packets were generated at a rate of 1 packet per second. Among these, 250 packets were generated from the nodes anchored at the bottom and the remaining half was generated randomly from the floating nodes.

5.1.1 Performance Metrics

Delivery ratios, end to end delays and energy consumption are considered as the metrics to check the performance of the proposed scheme. *Delivery ratio* is the total number of data packets received successfully at all the sinks. *End to end delays* is defined as the average time taken for a data packet in order to reach the surface sink from the source node. *Energy consumption* is defined as the total energy consumed throughout the network during all the routing processes and states like sending, receiving and idling.

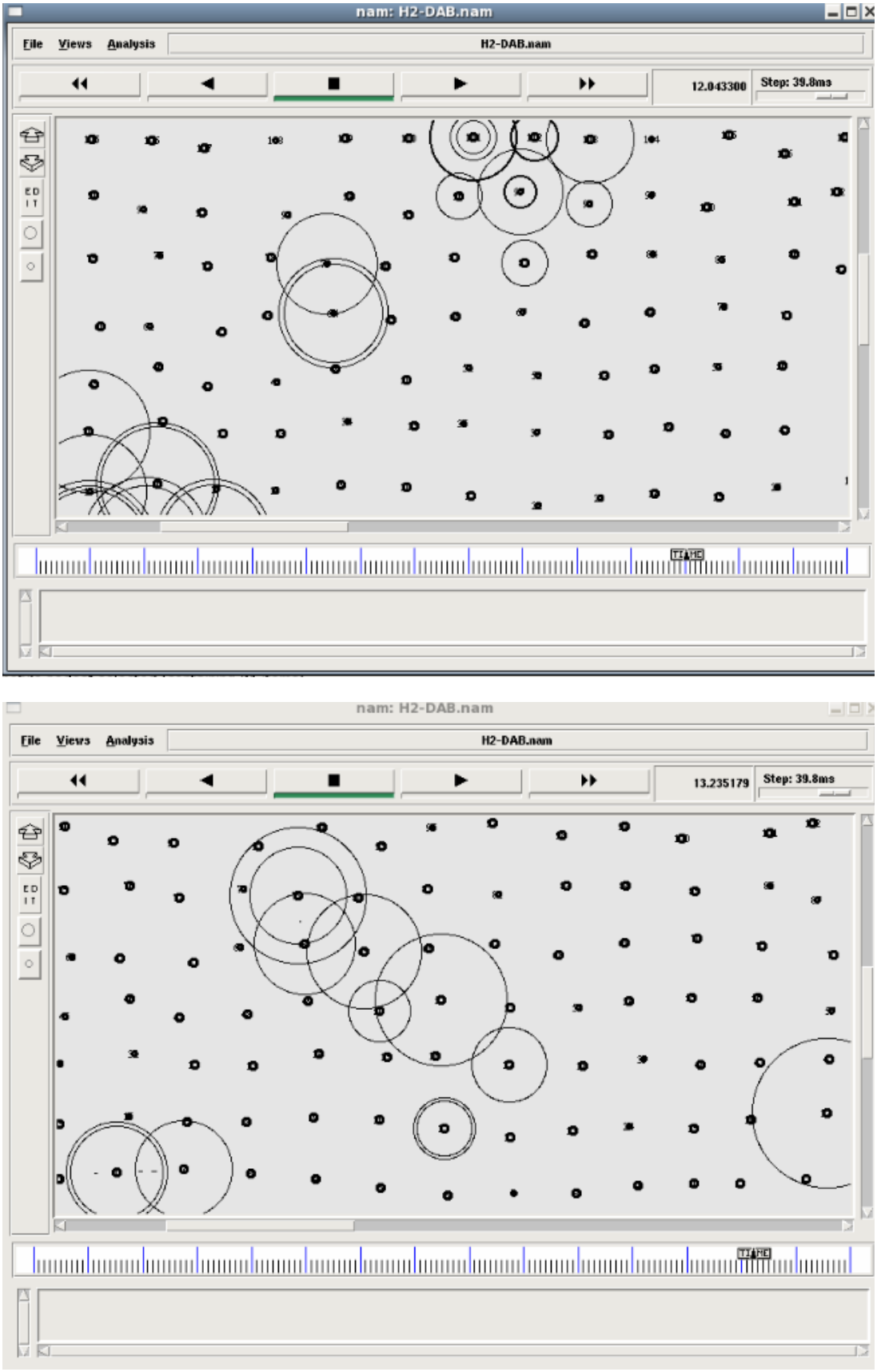

Figure 5.1 Screen shots of name animator during the H2-DAB simulations

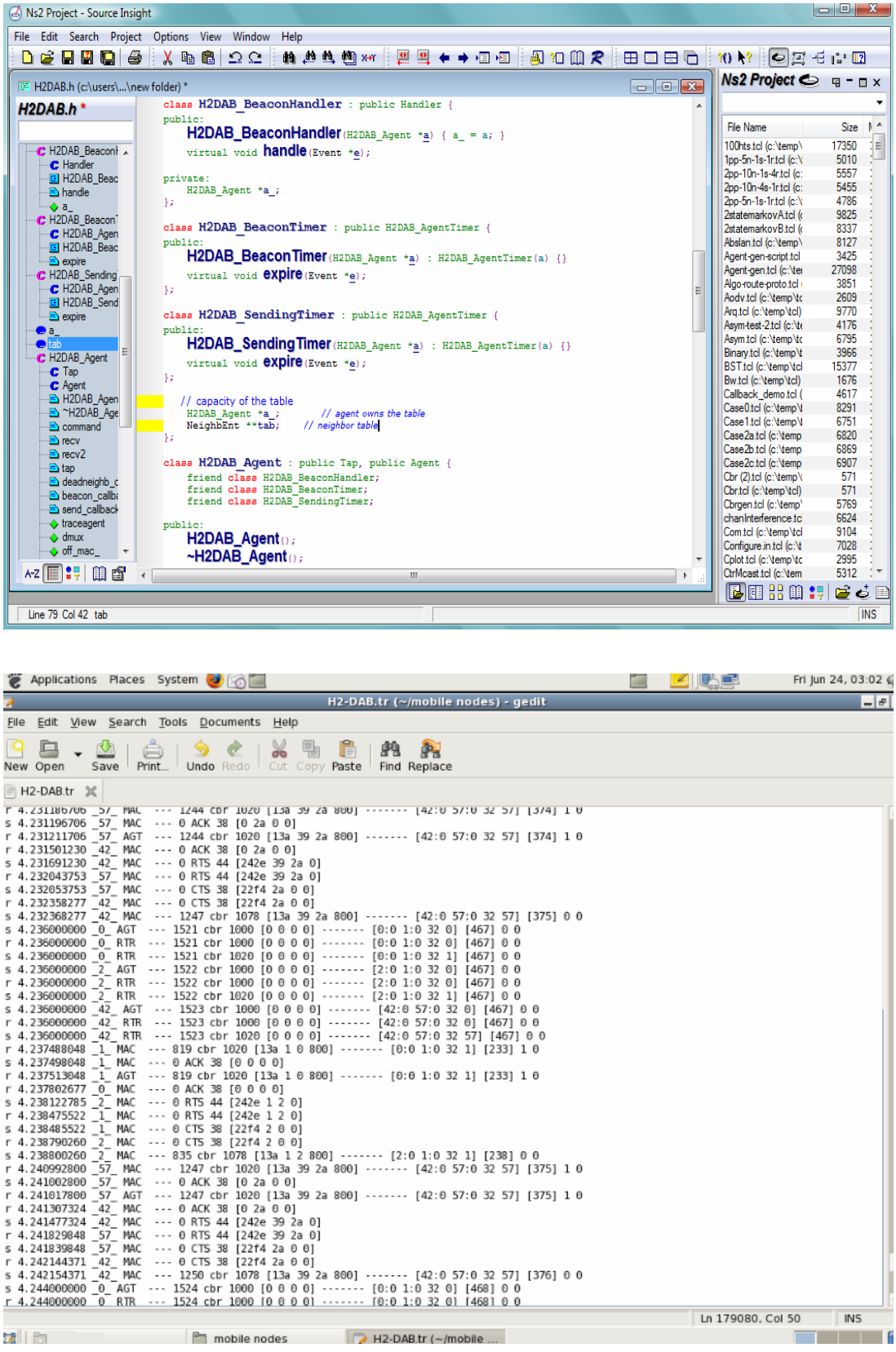

Figure 5.2 Screen shots of a trace file and Source-Insight programming tool during the H2-DAB simulations

5.2 Results and Analysis

In this section, we evaluate H2-DAB with different parameters including node mobility, different number of courier nodes, variations in interval life and with different offered loads.

5.2.1 Node Mobility

Figure 5.3 explains the data delivery ratios at different speed of node movements; three Courier nodes were used during the simulation setup. As shown in the figure, the data delivery ratios are 100% with the suggested number of nodes in the network. These delivery ratios are not seriously affected if the node density starts to decrease, we can still achieve around 95% delivery ratio if 25-35% nodes are not available. If we look at the delivery ratios in the sparse areas, where 50% nodes are not available, we can still receive around 85% data packets at the average node movements. The effect of node movement on end to end delays and energy consumption is shown as in Figure 5.4 and Figure 5.5, where some difference with node mobility can be observed as compared to static nodes. In the beginning, with dense deployment, this difference is minor and then it starts to increase when the nodes start to decrease; but these differences are still not considered high until more than 50% nodes become part of the network.

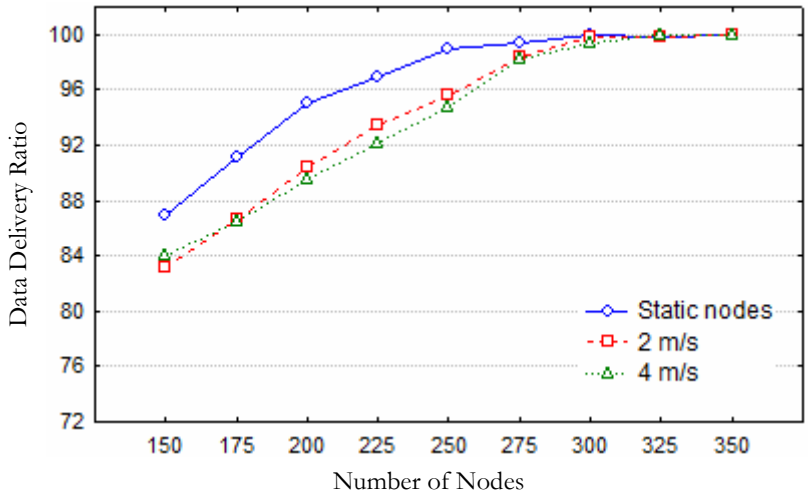

Figure 5.3 The effect of node movements on H2-DAB (data delivery ratio)

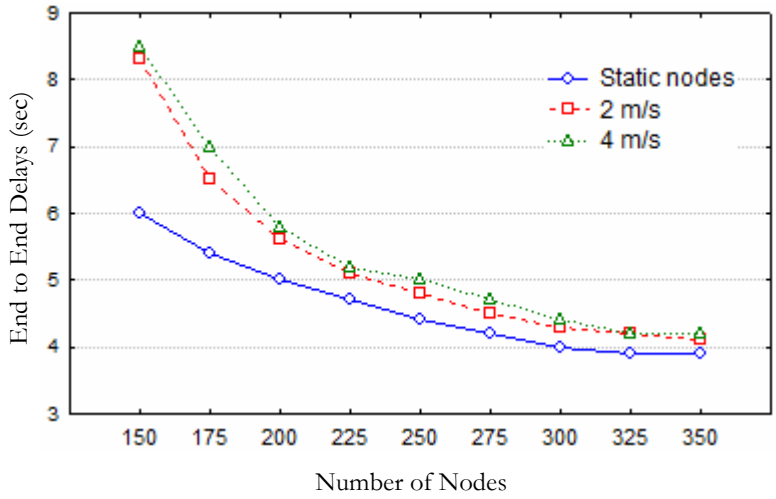

Figure 5.4 The effect of node movements on H2-DAB (end-to-end delay)

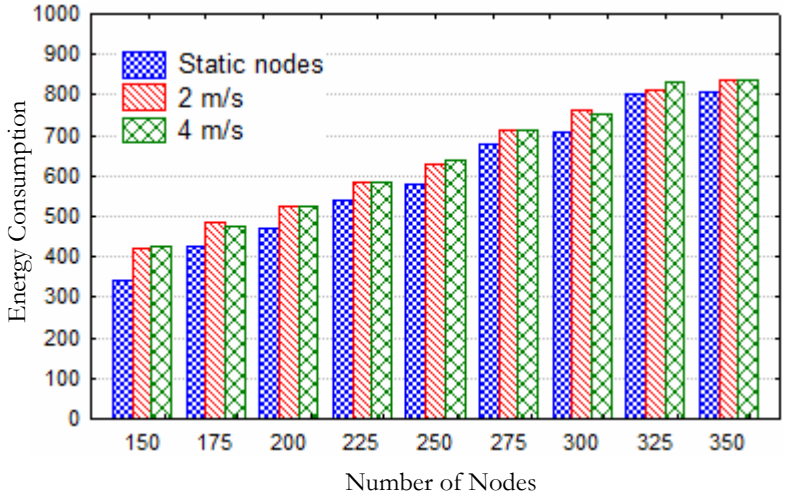

Figure 5.5 The effect of node movements on H2-DAB (energy consumption)

In general, node mobility has no serious effect on data delivery and energy consumption, as the result differences are minor at different node speeds. It is due to that, no complex routing tables need to be maintained according to the location information of the sensor nodes. So, there is no need to manipulate them before any routing decision, even when the node has changed its position. Every node can use its *HopID*, no matter how far it has moved vertically. The movement of nodes can affect the average end to end delays to some extent, but this only occurs in sparse areas. In dense areas, all the metrics generate almost similar results even with different node movements.

5.2.2 Courier Nodes

As mentioned earlier H2-DAB can complete its task without the presence of any Courier node, but for reliability concern and better resource utilization we are suggesting that a small number of Courier nodes to be used. The following figures show how different number of Courier nodes helps to increase the overall performance of the routing protocol. In Figure 5.5, from the delivery ratios we can see that, although in dense areas we can achieve high delivery ratios even without Courier nodes, but in sparse areas the presence of Courier nodes can increase these ratios. In such situations, these Courier nodes easily accommodate the deficiency of ordinary sensor nodes. From these results, it is clear that only 1-2% of Courier nodes can provide more than 90% delivery ratios with 50% less ordinary sensor nodes

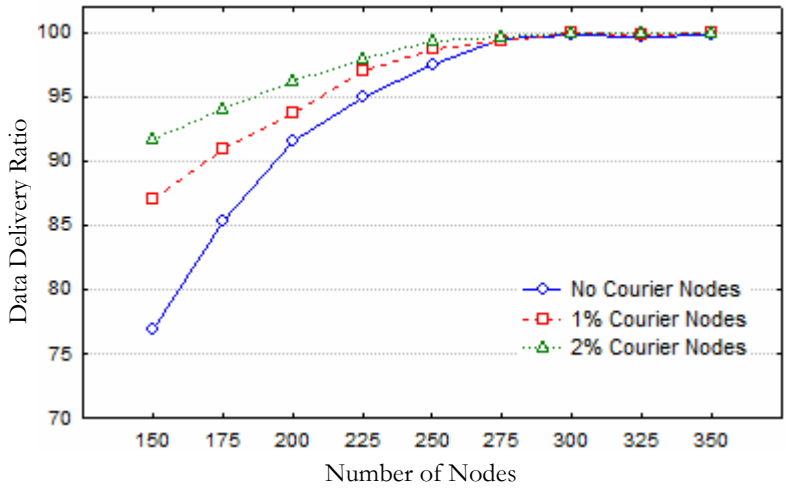

Figure 5.6 The effect of courier nodes on H2-DAB (data delivery ratio)

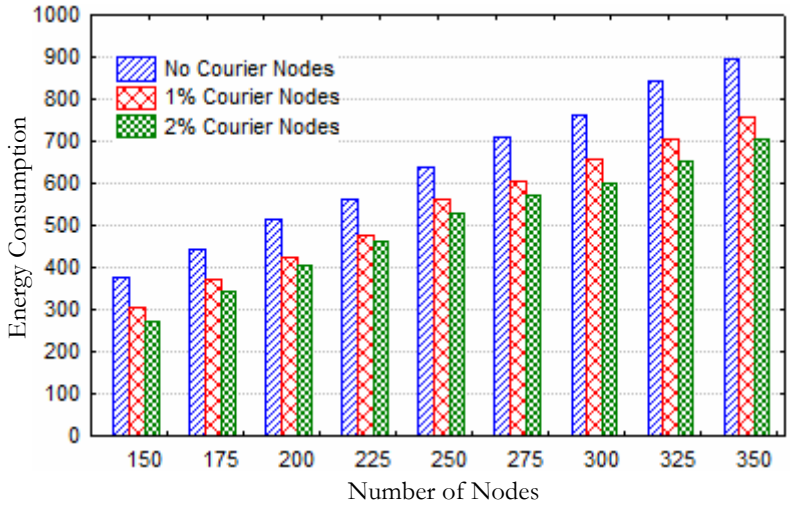

Figure 5.7 The effect of courier nodes on H2-DAB (energy consumption)

Not only increasing delivery ratios, but Courier nodes also help to reduce the overall energy consumption of ordinary sensor nodes which is shown in Figure 5.7. These Courier nodes collect the data packets from the bottom layers and move physically in order to deliver these packets to the surface sinks directly. By doing so, the involvement of ordinary sensor nodes decreases, so the cost per packet delivery is decreased which ultimately increases the life of the network.

5.2.3 Interval Life

This section explains the effect of interval life on the performance of H2-DAB. The life of every interval depends on the conditions of its environment and situations, and we can adjust them according to the conditions.

Consider a 6-hour result where each interval has a life of 2.5 hours and the network consists of 300 ordinary and 4 Courier nodes. So, *HopIDs* will remain valid for 2.5 hours and after every 2.5 hours new Hello packets will broadcast and new *HopIDs* will be assigned. In Figure 5.8 which presents the results of data delivery ratios, it can be observed that these ratios are around 100% at the start of the half time; then during the last half or one hour these ratios start to decrease by approximately 1% or less, but at the beginning of the next interval they start to increase again, approaching 100%. Next from Figure 5.9, which shows the end to end delays at different hours of one interval, we can see that these can vary from 5-8 seconds. At the start of each interval, these are small and then before the start of the next interval these can increase to 2-3 seconds. Therefore, it is the only metric where results can vary during different parts of an interval.

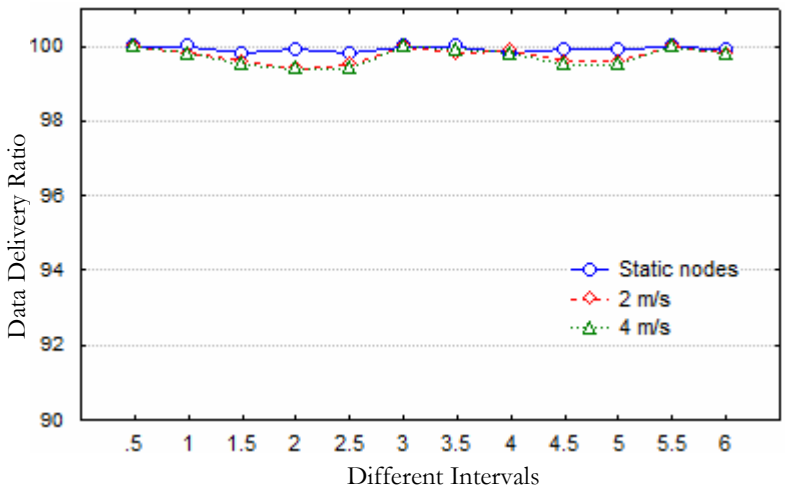

Figure 5.8 The effect of interval life on H2-DAB (data delivery ratio)

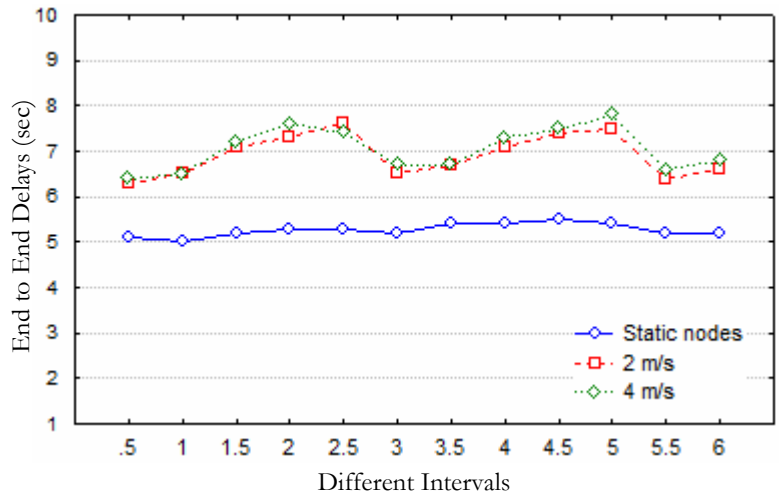

Figure 5.9 The effect of interval life on H2-DAB (end to end delay)

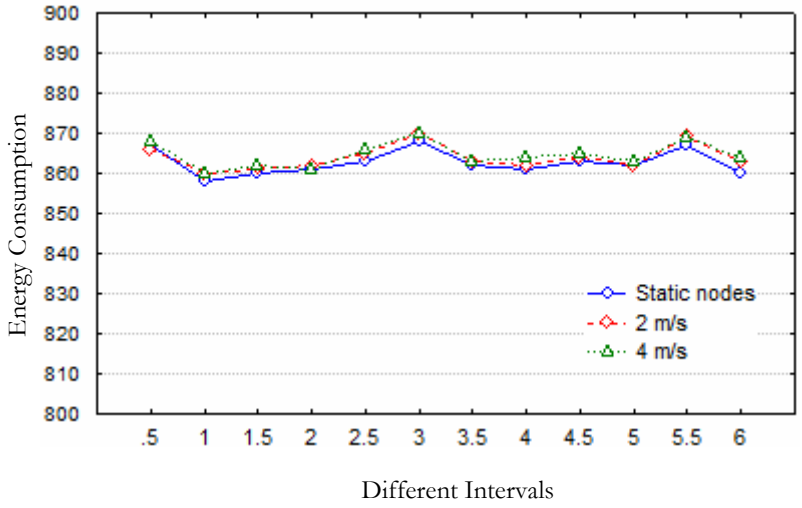

Figure 5.10 The effect of interval life on H2-DAB (energy consumption)

In Figure 5.10 which presents the consumption of energy, again we can see almost the same pattern of energy consumption, not only at different hours but also with different node mobility. The only observable variation is at the start of every interval when Hello packets start to broadcast, which uses some extra energy as compared to the remaining life of the same interval.

5.2.4 Offered Load

In order to check the performance of H2DAB with different offered load, we analyzed the delivery ratios and end-to-end delays by producing more data packets in the network. In normal case, a network generates 1 packet per second, but here we consider the cases where first the network generates 3 packets every 2 seconds and then 2 packets per second. The delivery ratio with different offered load is presented in Figure 5.11. It shows that with dense nodes, the delivery ratios are almost the same, and they start to differ when the number of nodes starts to become sparse. At high offered load and with fewer nodes in the network, sometimes a node cannot find the next hop and the packets start to increase in the buffer which results in the network discarding them.

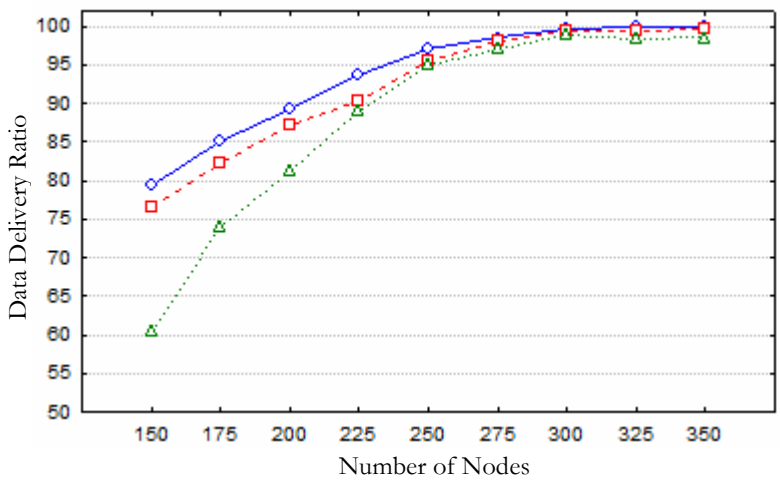

Figure 5.11 The effect of different offered loads on H2-DAB (data delivery ratio)

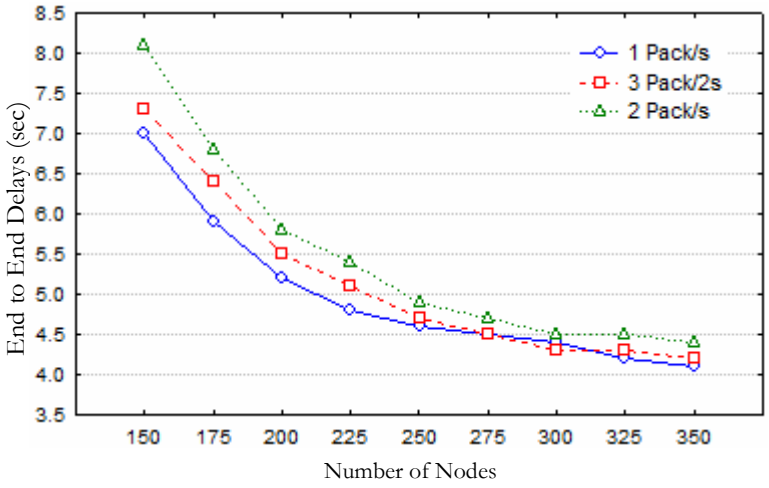

Figure 5.12 The effect of different offered loads on H2-DAB (end to end delay)

Figure 5.12 shows the variations in end-to-end delays when the number of packets in the network is increased. It shows that, the network can easily handle even when 50% more packets becomes part of the network; even these delays are affordable when double packets are generated in the network.

5.3 Comparison with VBF

VBF [10] is the most commonly used routing technique for evaluating the performance of routing techniques and in this work, it is used to compare the performance of the new routing techniques proposed for the underwater sensor networks. It is a position based routing approach in order to handle the issue of continuous node movements. As it is a position based approach, state information of the sensor nodes is not required and only a small number of nodes are involved during the packet routing. Data packets are forwarded along redundant and interleaved paths from the source to sink, which helps to handle the problem of packet losses and node failures. It is assumed that every node already knows its location, and each packet carries the location of all the nodes involved including the source, forwarding nodes and final destination. Here, the idea of a vector like a virtual routing pipe is proposed and all the packets are forwarded through this pipe from the source to the destination.

Only the nodes closer to this pipe or “vector” from their source to the destination, can forward the messages. By using this idea, not only the network traffic can be reduced significantly, but also it is easy to manage the dynamic topology.

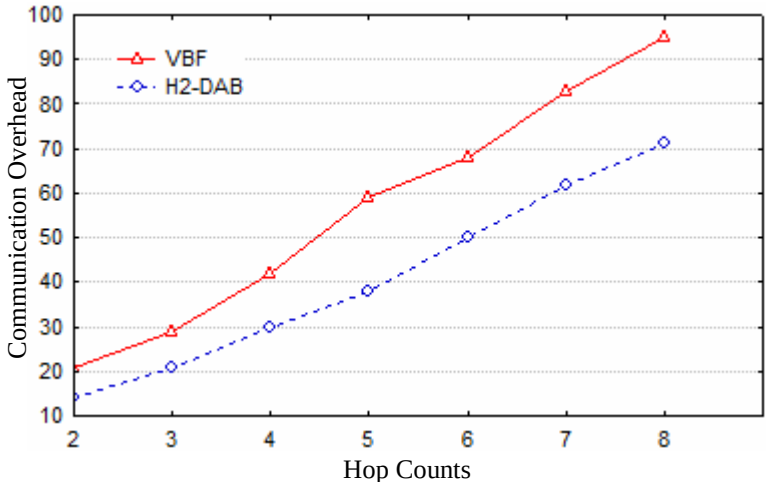

Figure 5.13 H2-DAB vs VBF (communication overhead)

5.3.1 Communication Overhead

First we examine the effect of communication overhead that both approaches face during the data forwarding process and the results are shown in Figure 5.13. When we talk about VBF, it requires a 3-way handshaking before the actual transmission of the data packets can occur. When a source has a data packet ready to be sent, first it sends a DATA_READY message towards the sink. After receiving this DATA_READY message, the sink replies with an INTEREST message to the source node. When the source node receives this INTEREST message then it sends the data packet towards the sink using this established path. Following that, all the data packets are flooded along this routing vector.

On the other hand, H2-DAB does not follow any 3-way hand shake process before starting data packet forwarding. Dynamic addresses assigned during the address assigning phase are used during the data forwarding phase.

Moreover, H2-DAB does not follow the packet flooding as only one copy of each data packet is forwarded by each sensor node taking part in the data forwarding. As a result, H2-DAB has a significantly low communication overhead than VBF regardless of any environment factors like node movements.

In underwater environments, node movements are common due to water currents; hence the effect of node mobility on different performance metrics is very important. Due to these movements, uniform distribution of sensor nodes is strongly affected; which as a result nodes start to become dense as well as sparse in some areas. Not only this, but the nodes can leave the monitoring areas which also make the network sparser. Here, we explore the effects of node mobility on both of the routing techniques. In Figure 5.14, we can observe that the performance of VBF is seriously affected by node densities, where delivery ratios are degraded due to possible occurrences of void regions. In sparse areas, it is possible that no node is available in the routing pipe where nodes are responsible for packet forwarding. On the other hand, these movements have small effect on H2-DAB as delivery ratios are acceptable in sparse areas even when only half of the nodes are available in the network. The main reason for this is that, H2-DAB follows a recovery mode where a node can pass data packets towards the lower layers when it cannot find a proper next hop in sparse areas.

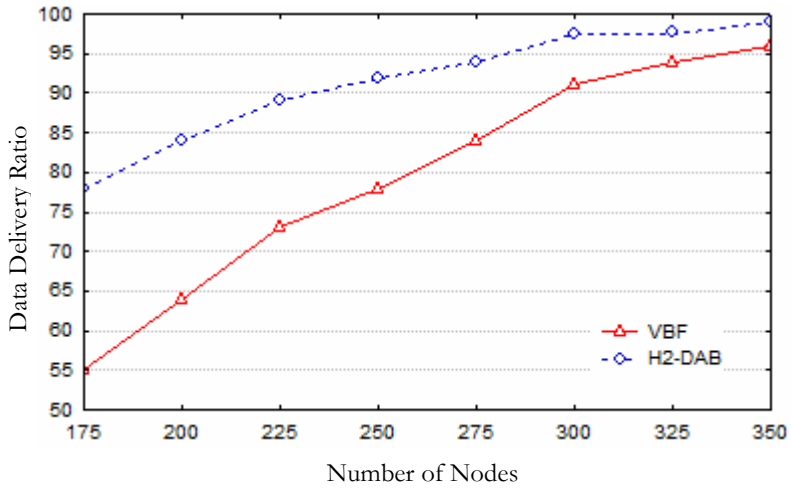

Figure 5.14 H2-DAB vs VBF (data delivery ratio)

5.3.2 Delivery Ratios

In VBF, ratio of data delivery is strongly dependent on the routing pipe as data packets can be forwarded only along static routing vector. In some areas, if nodes are deployed sparsely or become sparser due to some water movements, then it is possible that very few or even no node will lie within that virtual pipe which is responsible for data forwarding; it is also possible that some paths may exist outside the pipe which ultimately decreases the delivery ratios as shown in Figure 5.14. Moreover, VBF is very sensitive to the routing pipe radius threshold; this threshold can affect the routing performance significantly. If this threshold is too short then there is a strong possibility that no forwarding node is available in this pipe, while if it is too large then more and more nodes can take part in this process which ultimately increases the communication overhead.

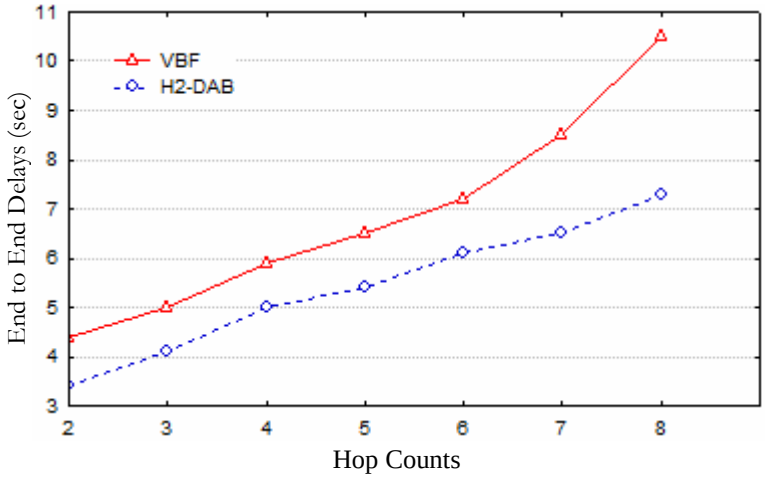

Figure 5.15 H2-DAB vs VBF (end to end delays)

5.3.3 End-to-End Delay

Next, we measured the end-to-end delay with different hop counts from source to sink (3 to 7). As we have mentioned earlier, VBF follow a 3-way handshake procedure before transmitting the data packets, which leads to a larger end-to-end delays. Further, larger number of hops makes this difference more significant as the control packets are required to exchange at every hop. In contrast, with H2-DAB, end-to-end delay increases linearly according to the number of hops as shown in Figure 5.15.

5.4 Comparison with DBR

Further, we checked the performance of H2-DAB by comparing with DBR [6]. The proposed algorithm provides better results in most of the situations and with different parameters.

5.4.1 DBR

For location based routing schemes, most of the protocols require and manage full-dimensional location information of the sensor nodes in the network which itself is a challenge left to be solved for UWSNs. Instead of requiring complete localized information, DBR [6] needs only the depth information of sensor node. In order to obtain the depth of current node, the authors suggested to equipping every sensor node with an inexpensive depth sensor. In their architecture, multiple data sinks placed on the water surface are used to collect the data packets from the sensor nodes. DBR takes decision on the depth information, and forwards the data packets from the higher to lower depth sensor nodes. When a node has a data packet to be sent, it will sense its current depth position relative to the surface and place this value in the packet header field “Depth” and then broadcast it. The receiving node will calculate its current depth position and can only forward this packet if its depth is smaller than the value embedded in the packet, otherwise it will simply discard the packet. This process will be repeated until the data packet reaches at any of the data sink. Packets received at any of the data sink are considered as successful delivery at the final destination as these data sinks can communicate efficiently with much higher bandwidth through radio channel.

5.4.2 Data Delivery Ratios

First, we compare the delivery ratios with single and multiple sinks as shown in Figure 5.16. When we used multiple sinks, it was found that both algorithms provided similar results with dense node deployments. However, when the nodes started to decrease then the delivery ratios for DBR started to decrease as well, while the effect on H2-DAB delivery ratios is less.

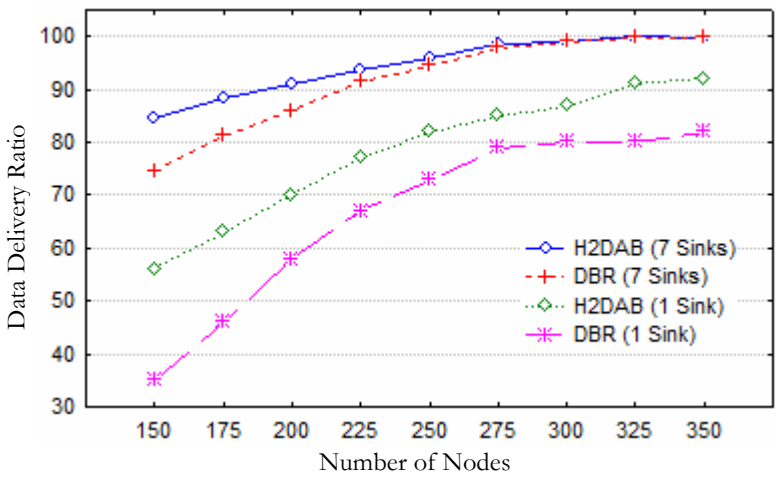

Figure 5.16 H2-DAB vs DBR (data delivery ratio)

The main reason is that, DBR uses only the greedy mode for data forwarding; sometimes when no node at lesser depth is found then it cannot forward even though some nodes at higher depth are available in the communication range. . When we used a single sink which was placed at the centre of the surface, a clearer difference in delivery ratios was found. Again due to the greedy mode of DBR, the sensor nodes tried to send towards the water surface but not towards the sink which was placed at the centre of the deployment area. Here no recovery method was available for DBR and when the nodes in the upper layers broadcast the data packets, no node was able to accept the packets because no nodes was available at smaller depth. At the same time the sinks were not available at those locations. Due to this, the data packets were discarded which consequently decreased the delivery ratios.

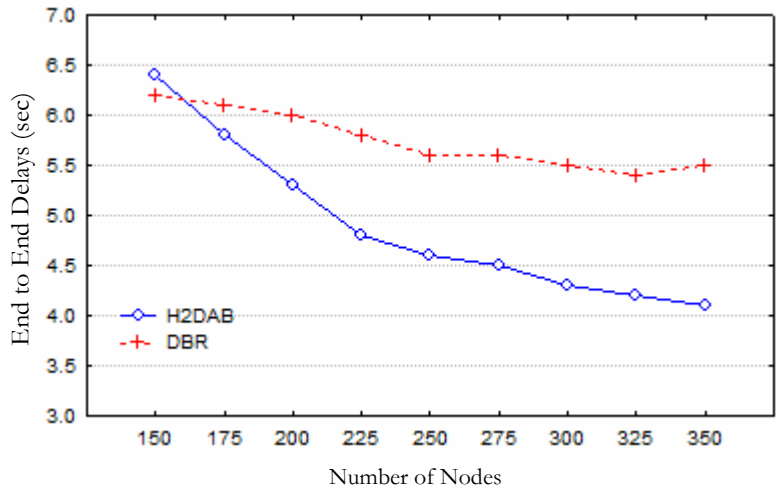

Figure 5.17 H2-DAB vs DBR(dnd to dnd delay)

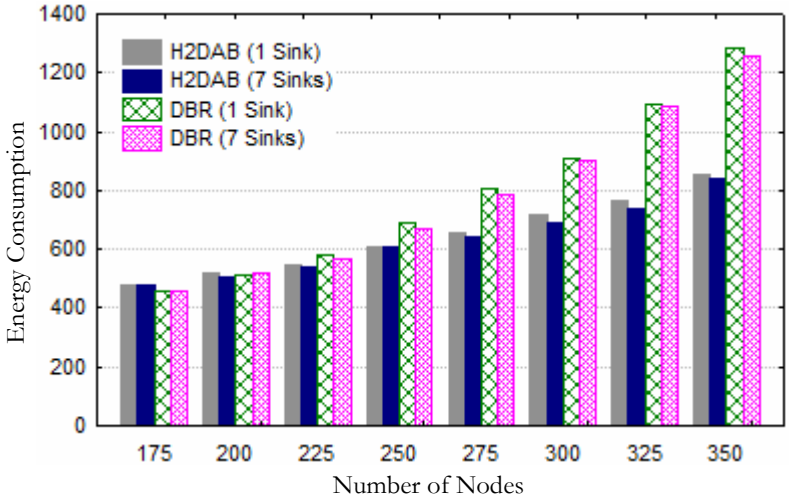

Figure 5.18 H2-DAB vs DBR (energy consumption)

5.4.3 End-to-End Delays

Referring to end-to-end delays in Figure 5.17; here H2DAB delivers data packets with less end-to-end delays when reasonable sensor nodes are available in the deployed area. Based on our understanding, this is due to the holding time used in DBR where all the nodes that have received the data packet wait, instead of immediately forwarding the packet to check whether their neighbouring nodes are also going to forward the same data packet or not.

On the other hand, in H2DAB when the nodes start to become sparser then delays start to increase. This occurs because with small neighbour node when a node cannot find a forwarding node with smaller *HopID* then it waits for a defined amount of time before going for 2nd or 3rd try. In such a situation, it can send the data packets to nodes at higher depths even in much sparse areas. Even though it may result in higher delays when nodes become much sparser, but ultimately better delivery ratios can be achieved.

5.4.4 Energy Consumption

Next, Figure 5.18 shows the comparison of energy consumption with different number of sinks. The comparison shows that the energy consumption with smaller number of nodes is almost similar for both schemes. The energy consumption starts to differ when the number of nodes starts to increase. For DBR, it happens due to two reasons. First, DBR uses broadcast for every single data packet in dense areas after receiving it, then more and more nodes go for broadcasting of the same data packet. Secondly, in such areas, when more nodes receive the data packet then every receiving node will check its depth every time. Due to the presence of large number of neighbouring nodes, the current node receives a burst of data packets and in order to decide whether to accept or discard it, each receiving node will check its depth every time which ultimately drain more energy from the network.

On the contrary in H2DAB, it consults the neighbours only when the “Next Hop” expires or not acknowledged. DBR faces problems in both situations. When nodes start to increase the energy consumption is high, and when nodes start to decrease then the delivery ratios are affected by this sparseness. On the other hand, H2DAB maintains good delivery ratios with small number of nodes, and starts to improve with controlled energy consumption when nodes start to increase in the area.

5.5 Dynamic Packet Size

The general scenario of the underwater environment set up is shown in Fig 5.19. A cluster of 100 nodes is placed in the middle of a body of water with a dimension of 2km x 2km x 200m. This is to avoid reflection effects near the water surface and the water bottom. The depth of 200m is chosen to simulate the shallow water environment. One sink is placed roughly at the centre of the cluster to collect data packets from other nodes. The distance range between the sink and a source node is 100m to 1km. The maximum transmission range of the nodes is to be 1km. In the simulation, two nodes were created (one transmitter and one receiver/sink) at any one time for a one hop data packet relay with one constant bit rate (CBR) module per layer. A unidirectional Module/Link connects the two nodes. The packet flow is in accordance to the ns-2 MIRACLE layered framework.

The transmitter CBR module, acting as an agent, generates data packet of the required size. The MIRACLE physical layer (MPHY) uses binary phase shift keying (BPSK) modulation to send the data packet over the underwater channel to the receiver. The underwater channel is configured with Shannon channel characteristics. Our simulation has adopted the energy efficiency definition from the work of [141]. Some essential parameters used in the simulation are listed in Table 5.1.

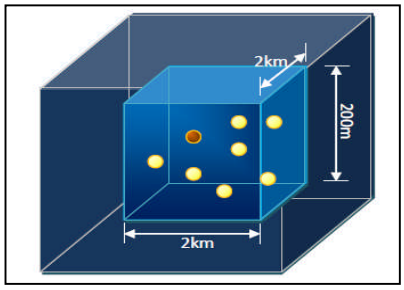

Figure 5.19 The general node deployment scenario

Table 5.1 Essential simulation parameters

Parameter	**Setting**
Payload length	10 – 1000 bits
Header length	10, 40, 160 bits
Distance	100m – 1000m
Frequency	8.2 KHz
Bandwidth	6 KHz
Protocol	ALOHA

5.5.1 Data Packet size and BER

When ARQ protocol is used in relatively high BER links the communication performance is sensitive to the packet size. In our simulation, we used the k_{opt} as defined in equation (1) below which was adopted from [142, 143].

$$k_{opt} = \frac{-h \ln(1-\rho) - \sqrt{-4h \ln(1-\rho) + h^2 \ln(1-\rho^2)}}{2 \ln(1-\rho)} \quad (1)$$

This equation shows that the optimal packet size k_{opt} is a function of BER, ρ and packet header length, h.

Fig. 5.20 shows a set of graphs relating packet size to different BERs with different header length. This is one of the set of graphs to be used in the proposed optimization algorithm. Do take note that a header length of 160 bits is the standard length used in the RTS data packet for stop-and-wait ARQ protocol. It is understood that under this stop-and-wait protocol the source node will transmit an RTS packet to the sink node to establish the link between them before packets transmission. In the proposed algorithm, this RTS packet will double its function as a test data packet for the source node to compute the quality of the link, thus obtaining the link BER. The top most graph/line in Figure 5.20 is to be used as the reference graph for the proposed algorithm. The rest of the plots are used for comparative studies purpose.

A simplified data set can be obtained from Figure 5.20. For example, with a header length of 40 bits, the simplified data set is obtained as in Table 5.2. This simplified data set stores BERs in an incremental step of a decade. These increment steps make BER computation practically faster. For practical implementation, the packet size to be composed in actual transmission can be the truncated value or a round-up value if truncation is not preferred.

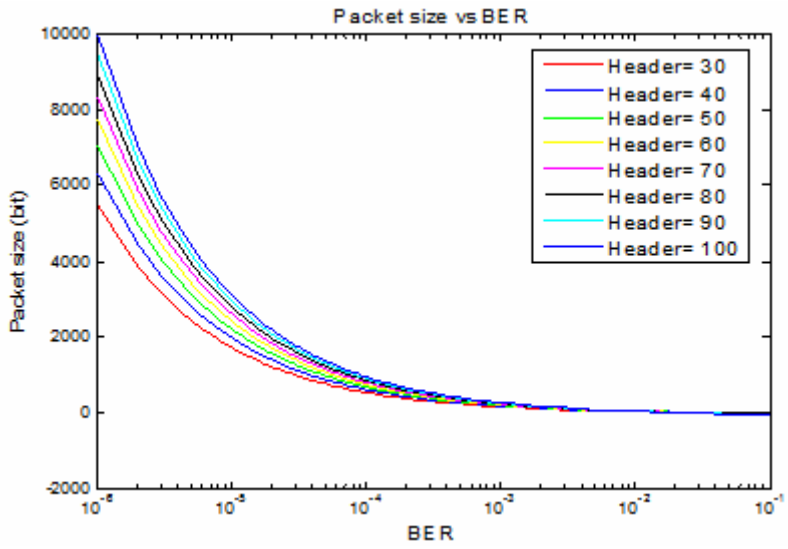

Figure 5.20 Packet size vs BERs

Table 5.2 Simplified data set

BER	k_{opt}	Truncate
10^{-2}	39.86605	39
10^{-3}	178.9482	178
10^{-4}	612.1234	612
10^{-5}	1979.8950	1979
10^{-6}	6304.5221	6304

5.5.2 Data Packet Size and Throughput Efficiency

In stop-and-wait ARQ protocol, its throughput efficiency is defined as the ratio of useful packet time and the total time spent on the average for a successful packet transmission. The average time is taken over the number of retransmissions. With a probability of packet error given as ρ, the average time needed to transmit 1 packet successfully is given by [144] as,

$$T_1 = \frac{1}{1-\rho} T(1) \tag{2}$$

With this the efficiency for transmitting a group of g successful packets can be expressed as,

$$\eta = \frac{g\, N_l\, T}{T_g} = (1-\rho)\frac{g\, N_l\, T}{T(g)} \tag{3}$$

Where, N_l is the payload length, and T is the bit duration. So, with a given set of physical layer parameters (ρ, R, d) where ρ is the probability of packet error, R is the bit rate, and d is the distance between transmitter and the receiver; the throughput efficiency can be written in the form of,

$$\eta = (1-\rho)^{N_l+N_{ok}} + \frac{N_l}{N_l+\mu} \tag{4}$$

$$\mu = N_{oh} + \frac{T_w R}{g} N_{oh} + \frac{2}{gc} dR \tag{5}$$

Where, Tw is the total waiting time in the stop-and-wait protocol, c is the nominal underwater acoustic sound speed of 1500 m/s, and N_{oh} is the header length. The optimal packet size can now be evaluated by differentiating η with respect to Nl and equating it to zero. From which the optimal packet size, N_{opt} is given by,

$$N_{opt} = \frac{\mu}{2}\left[\sqrt{1+\frac{4}{\mu\rho}} - 1\right] \tag{6}$$

With N_{opt} evaluated, the optimal throughput efficiency can be written as,

$$\eta_{opt} = (1-\rho)^{N_{opt}+N_{ok}} + \left(\frac{N_{opt}}{N_{opt}+\mu}\right) \tag{7}$$

Take note that μ is related to dR (range-rate) product, where d denotes the distance in meter between a source-sink pair and R is the data transmission rate in bps. It is explicit that N_{opt} is a function of range-rate product (dR) and BER (ρ) of the communication link. Some of the crucial parameters used in our simulation include,

Table 5.3 Simulation parameter for packet size and throughput efficiency

Link delay	:	0.01s
BER (ρ)	:	10^{-3} , 10^{-4}
Distance	:	500 m to 5 km
Rate (R)	:	100 bps to 1000 bps
Header (N_{oh})	:	160 bits
No. of group (g)	:	1

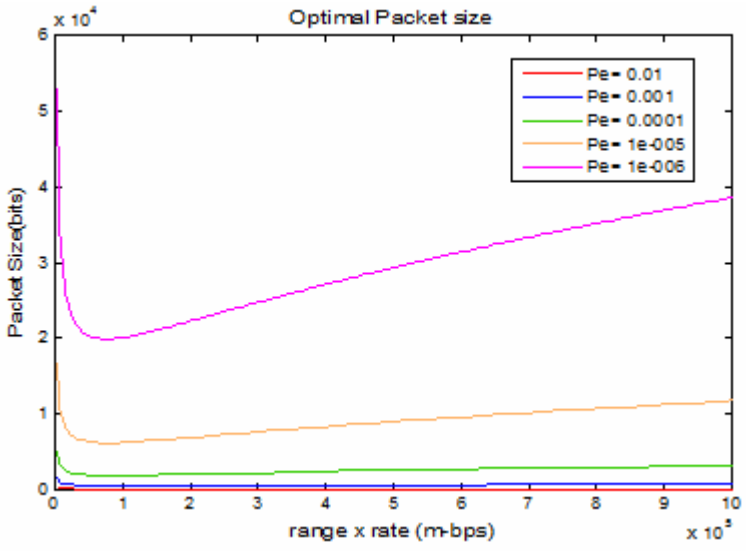

Figure 5.21 Packet size vs. range rate with different BER

The simulation of this N_{opt} resulted in a set of graphs shown in Figure 5.21. It can be seen here that low quality link does not permit large packet size. By keeping the distance d between the source-sink pair constant e.g. static nodes deployment, and for a certain BER, the packet size seems to be increasing fairly linearly with an increasing R. However, the packet size increases at a faster rate if the link ρ is low.

5.5.3 Data Packet Size and Energy Efficiency

In data communication systems, energy efficiency can be defined as the ratio of the amount of data transmitted and the energy consumed for that operation. Thus, minimizing the total amount of energy spent on its operations is an important factor for an energy efficient system. The underwater wireless channel, being time-varying and noisy in nature, dictates the possibility of data corruption causing packet losses (discarded) at the sink which demands retransmissions of the packets resulting in a waste of valuable energy. In actual fact, a well known primary cause of energy wastage is in the retransmissions of data packets.

Our investigation is focused on the physical layers (PHY), and it is assumed that nodes are able to discover each other and self-organize into a communication network

with peer-to-peer communication between any pair of neighboring nodes. In this context, the energy efficiency equation (Equation 10) by [145] is adopted and would be the main reference for the simulation works on finding the relationship between energy efficiency and packet sizes. This equation is a function of packet length l and BER link, ρ. k1 and k2 are transmitter/receiver equipment constant with α the header length. Implicitly, it involves the energy per useful bit (EPUB) element.

$$\eta = \frac{k_1 l}{k_1(l+\alpha) + k_2}(1-\rho)^{l+\alpha} \tag{8}$$

In our simulation it is assumed that the source and the sink are of homogeneous type, therefore they have the same equipment constants i.e. k1 = k2. So the energy efficiency term in the η equation i.e. the term

$$\frac{k_1 l}{k_1(l+\alpha) + k_2}$$

can be approximated to l/(l + α) for (l + α) >> 1. This is acceptable since in most of the practical applications packet length is more than hundreds of bits. This is also in line with the basic definition of energy efficiency. In simulating this energy efficiency, some of the essential parameters are listed here:

Table 5.4 Essential simulation parameters

Parameters		Values
Link delay	:	0.01s
BER (ρ)	:	10^{-2} , 10^{-3} , 10^{-4}
Header (α)	:	40 bits
Length (l)	:	0 – 1000 bits

Our simulation has adopted the energy efficiency definition from the work of [10, 13]. A database was constructed from the outcomes of the simulation, from which the graph of packet size against energy efficiency under different link bit error rate (BER) is plotted as in Figure 5.22 below.

The graph strongly depicts high energy efficiency for low BER. The energy efficiency for link with BER of 10^{-4} is almost two fold than those with BER of 10^{-2}. The efficiency drops very sharply for high BER when the packet length is increased beyond the peak energy efficiency. This is practically true because the probability of packets being corrupted is high and therefore the demand for retransmission increases and more energy is thus wasted. Therefore it is not surprising to observe that the energy efficiency tapered off more gently beyond the peak performance for links with low BERs. The consequence is large packet length/size in good quality link is able to attain higher energy efficiency than links with poor quality.

It is obvious from this plot that an optimal packet size N can be obtained from each BER. That is, N can be easily determined by choosing the point of maximum energy efficiency from the graph. For example, the optimal packet size for a link quality of 0.001 is given by 100 bits with the maximum energy efficiency of 84%.

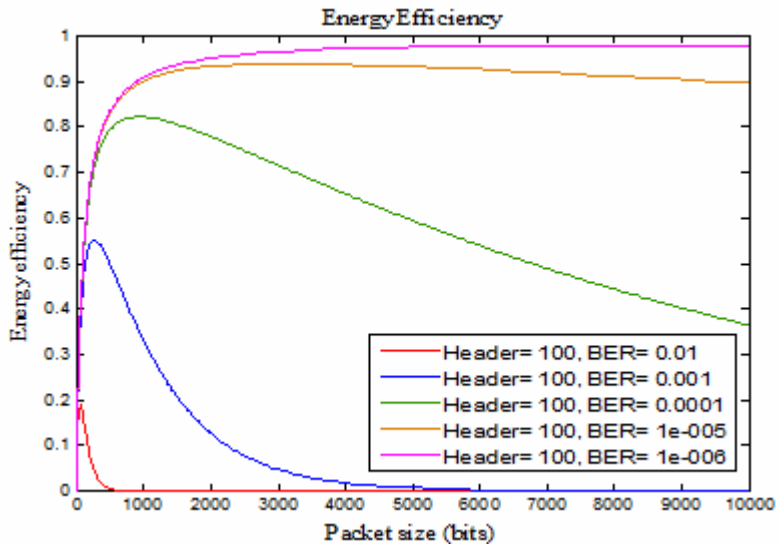

Figure 5.22 Energy efficiency vs packet size under different BERs

It can be seen from the plot in Figure 5.22 that the energy efficiency decreases with increasing BER, denoting that the more unreliable the channel is, the more energy is wasted. This phenomenon can be explained in the sense that when the link quality deteriorates, more data packets would be corrupted. As a result, the demand for packet retransmissions increases thus resulting in more energy being consumed for

these retransmissions. It is interesting to observe that the energy efficiency for link with low BER drops more gently after the peak than link with high BER. It brings out a point here that energy efficiency may not suffer much deterioration under good link quality even with a large packet size. For instance, with a BER of 0.0001 the optimal packet length can be varied practically from 150 bits to 900 bits with the energy efficiency maintained at/or above 90%. This, in turn, may help to produce a higher throughput efficiency with the opportunity to load the transmitted packets with larger payload. A snapshot of the database structure constructed from the outcomes of the simulation and which was used to plot the graph of Figure 5.22 is given in Table 5.5.

Table 5.5 Snapshot of database structure energy efficiency

Pckt Size (bits)	**EPUB (mJ/bit)**	**BER**	**PER**	**Energy Efficiency**
16	1.9668	0.01 0.001 0.0001	0.1485 0.0159 0.0016	0.4257 0.4921 0.4992
96	1.0728	0.01 0.001 0.0001	0.6190 0.0916 0.0096	0.3493 0.8327 0.9079
176	1.0302	0.01 0.001 0.0001	0.8295 0.1615 0.0174	0.1628 0.8004 0.9379
256	1.0151	0.01 0.001 0.0001	0.9237 0.2260 0.0253	0.0739 0.7499 0.9443
336	1.0074	0.01 0.001 0.0001	0.9658 0.2855 0.0330	0.0333 0.6975 0.9439
416	1.0027	0.01 0.001 0.0001	0.9847 0.3405 0.0407	0.0150 0.6469 0.9408

5.6 2H-ACK Reliability Model

In this section, we present the simulation results of our proposed 2H-ACK scheme and compare these results with the results obtained by general HbH-ACK method. Our proposed scheme generated better results when the number of nodes starts to decrease in the network. This can be observed from Figure 5.23; with different number of nodes, delivery ratios drop with pace when HbH-ACK is used, but these ratios are less affected when 2H-ACK scheme is applied. As UWSNs are error prone and nodes can die or leave the network, which results in sparseness of the network, so 2H-ACK provides better results in such situations with small densities.

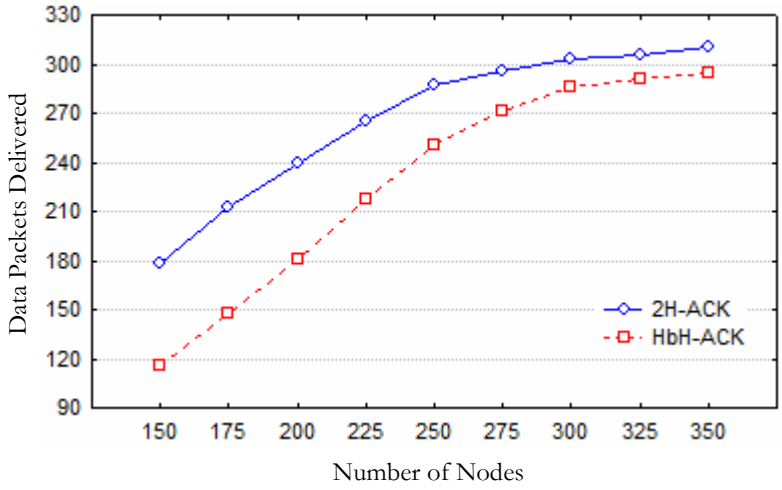

Figure 5.23 2H-ACK vs HbH-ACK
(delivery ratios with different number of nodes)

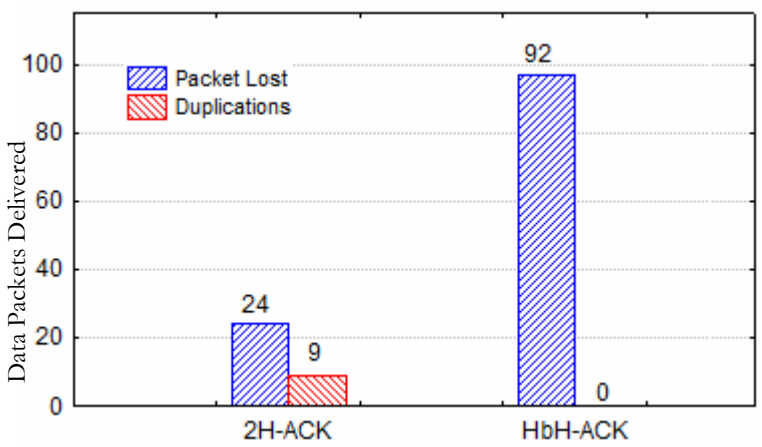

Figure 5.24 2H-ACK vs HbH-ACK (packet losses and duplications)

2H-ACK provides reliability by maintaining two copies of the same data packet by different nodes. Although, more than one copy of the same data packet can be received at the destination, but it happens with low probability especially when we compare them with data packets losses. This can be observed clearly in Figure 5.24 that shows the comparison of data packet duplications with the average number of data packet losses. Both the number of duplicate data packets and amount of packet losses are small when 2H-ACK is used. On the other hand, the results from HbH-ACK show that no duplicate packets are received as in this scenario only one node has the data packet in the network, but at the same time the amount of lost data packets is very high. These high data packet losses are due to node failure. As in both cases, when a node cannot communicate with any other nodes then all the packets residing in its buffer will fail to reach the destination.

CHAPTER 6

CASE STUDIES

This chapter describes a possible implementation of the proposed routing technique in different ocean monitoring applications. For this purpse, four case case studies are discussed: the first case study is regarding the monitoring of oil and gas field reservoirs; the second case is concerning marine pollution, the third case is about marine biology while in the fourth case study we will discuss the different issues related to port security. This chapter is divided into four main sections. Each section starts with a brief description of the case study followed by a short discussion on the possible implementation of UWSNs for these applications and what benefits can be achieved.

6.1 Importance of Oceans

Throughout history, oceans have had a big influence on humans both directly or indirectly. They take up about 99% of the habitat, 97% of the water, and 71% of the surface on the planet, but we still don't consider oceans as important as land or air. This is true even though more than half of our planet's population lives within 60 miles of the oceans, and ocean critters generate a good deal of the oxygen we breathe.

Ocean waters serve a variety of human needs: as a source for food and valuable minerals, a vast highway for commerce, and a place for both recreation and waste disposal. More and more, people are turning to the oceans for their food supply, either for direct consumption or by indirect means through consumption of meat, from livestock which have been fed with food made from harvested and processed fish.

This being said, the potential for food-production of the oceans is only partially realized. Furthermore, food is not the only product sourced from the oceans for

commercial use; other biological products of the oceans are also commercially used. Examples of these are pearls used in jewellery which are taken from oysters as well as shells and coral which have been widely used as a source of building material. Even the rock strata beneath the ocean floor are an important source of petroleum and natural gas.

Moreover, the very water itself is also a valuable resource. Seawater is not only a source of sodium chloride and other salts but also, by removing the salt, it can be used for drinking and for agricultural and industrial purposes. The ocean and its currents are also influential to the climate of many areas. It is well known that the ocean is important, but not many of us realise HOW important it actually is. Researchers do not know as much about the oceans as they do about land. As a consequence, we believe that this is something that we should be concerned about as the oceans have been in existence much longer than we have.

It is, in no doubt, a daunting task to monitor the aquatic environment and dynamical changes of the oceans. However, if we want to conserve marine resources while obtaining a sustainable development, changes occurring in the marine environment have to be monitored effectively. Oceanic lives and their ecosystems may be threaten by the climatic changes and increased water-born activities. A rapid change in the marine environment may have a tremendous influence on terrestrial life and the environment. One example of this is the effect of greenhouse gases and an increased surface temperature, leading to the melting of sea ice which in turn raises the sea level.

3.1 Underwater Oil and Gas Reservoirs

The ocean floor habitat is not as well known as coral reefs or coastal areas, but it is very important to all the organisms that live on the bottom (**benthic organisms**), as well as being commercially important. Many important minerals as well as oil and natural gas are found on the continental shelves and ocean floor. This natural gas and oil play a major role in meeting the world's energy needs. Although the outer continental shelf contains more than 50% of the remaining undiscovered natural gas

and oil resources, natural gas production in this area was enough to meet the needs of half of the world's population in the late 90's.

The world energy use is expected to increase by 45% between 2008 and 2030, and as the demand for oil and gas outstrips supply, increased pressure has been put on offshore operators. About 58 billion barrels of oil equivalent total resources have been discovered in deep water from 18 basins on six continents. However, only 25 percent of the total resources are either developed or currently being developed and less than 5 percent have been produced, which illustrates the immaturity of deepwater exploration and production. Thus they are pressured not only to go deeper in order to discover new reserves, but existing fields also must be made to continually operate at their peak potential. Many of the operations taking place offshore operate against the backdrop of continued economic uncertainty, fluctuating oil and gas prices, and frozen capital markets. It is estimated that currently more than $8 trillion in offshore reserves exist today. The focus is now on reducing costs while managing risk. Therefore, the development of effective production strategies is important in order to accelerate cash flow, and to ensure each well operates at the peak of its capabilities for efficient recovery of these reserves. This can best be done with modern monitoring techniques.

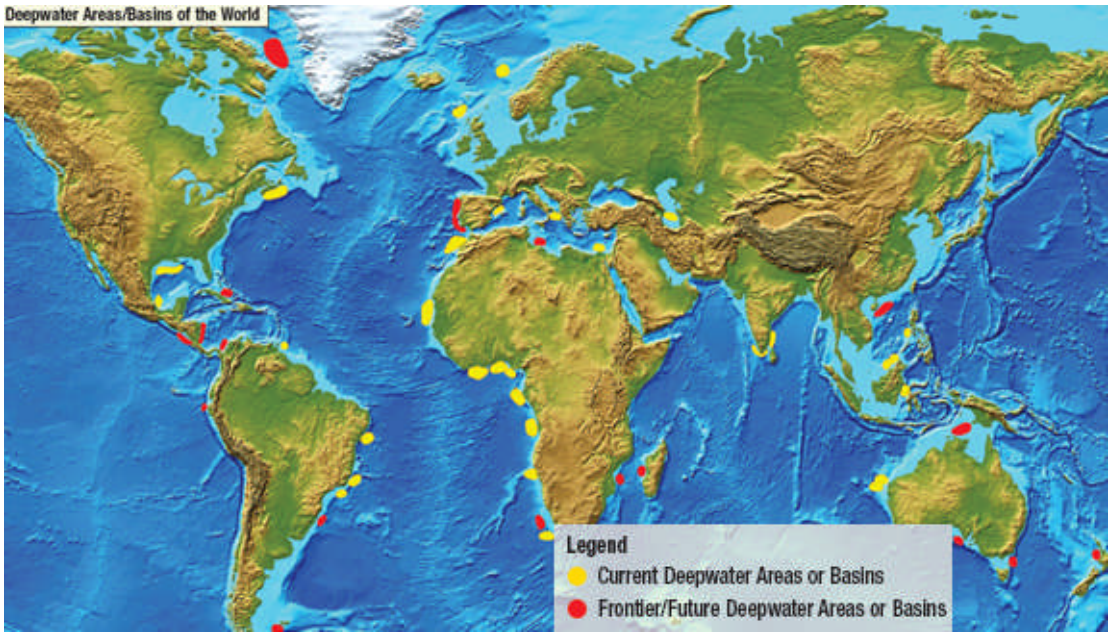

Figure 6.1 Current and future deepwater areas/basins of the world

During the last couple of years, the world's deepwater reserves have more than doubled. The growth within deepwater oil and gas production is expected to grow substantially during the next few years. Deepwater resources are currently one of the main areas of new offshore exploration efforts, as well as main areas for offshore production growth over the next few years. Offshore oil production is still dominated by benign and shallow water resources. Oil production from harsh environment areas currently represents approximately 7% of the global production, while deepwater areas represent only 5% of the production. A breakdown of the world oil supply is shown in Figure 6.2

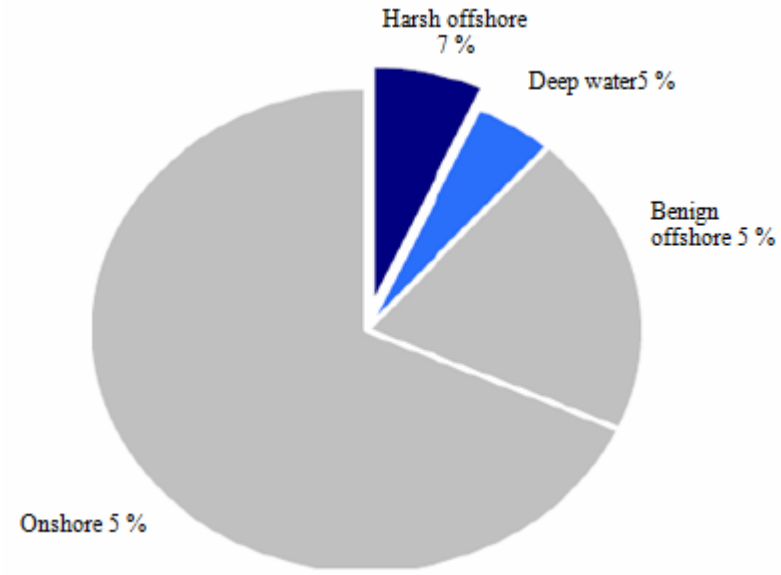

Figure 6.2 World oil supply. source: Pareto, DTI, NPD and Douglas Westwood

6.2.1 Monitoring Offshore Oil & Gas Reservoirs

Wireless communication has been proven to be a powerful technology and network tool for terrestrial applications. Implementing the wireless approach utilizes highly sophisticated subsea equipment, which removes the physical cabling requirement normally associated with subsea equipment. This reduction, subsequently, helps to reduce the total cost; ultimately, it is more practical to deploy and maintain a far greater number of sensor nodes. Furthermore, the benefits of temporary setups during the installation and workover operations are offered by adopting this technology.

It is important to ensure that the method used for implementation supports the instrumentation required when attempting to develop a wireless network method for enhanced subsea production monitoring systems. Such instrumentation may include a range of devices with a variety of control and output data requirements. This could be anything from low data rate temperature sensors to high data rate video systems. For this reason, it is necessary to include a wide communication capability. Besides the device data requirements, short-range wireless communications for on-system applications are also required. These could include a subsea tree or longer range communications for pan-field applications like monitoring along risers.

There are many challenges for wireless systems ranging from the complexity of the metal structures of subsea wellheads and trees to the associated flow lines, pumps, and valves. In developing a wireless network for subsea production systems, it is important to account for the complexity of the metal structures and to ensure that the wireless technology used, as well as the designed network, minimize potential problems like background noise and interference reflection. Due consideration should also be given to data rate requirements for each instrument; this is to minimize the power consumption during communication.

Efficiency is increased while costs are reduced for offshore operators by utilizing wireless technologies. Smart wireless applications allow operators to add monitoring points throughout offshore facilities at a fraction of the cost of wired instrumentation. Such applications also reduce the weight and footprints on platforms while the visibility of the operators' assets and overall operations is increased. The cost saving of this wireless technology is compelling. Emerson, in a recent study of an actual offshore platform with about 4,000 I/O, found that a savings of up to 70% in expenses can be achieved when wireless along with other technologies is installed in the process control system. . Moreover, eliminating some 800 more wired points also means a weight savings of up to 35 metric tons (38.5 tons) as well as a reduction of up to 129 cu m (168.7 cu yd) of the required deck space for cabling, cable trays, junction boxes and cabinets.

Another common challenge in the oil and gas industry is the need for gathering measurements of temperature, pressure and flow in remote and often unsafe locations.

As the industry grows and technology advances, there is an increased demand for real time measuring, recording, and transmitting of data. Underwater sensor technology can do this without cables and the problems commonly associated with them. A variety of systems is involved in the instrumentation of an offshore drilling or wellhead monitoring application. Measurement devices are often situated in remote locations and, therefore, wired sensors and equipment require electrical power, cables, and a conduit in order to be connected. This can be costly, inconvenient, and often impossible. Another factor is the manual labor associated with the installation, monitoring, recording, and data processing; human error is always a possibility. With that being said, one of the biggest concerns is when the operation is in risky and extreme offshore conditions.

All of the above challenges can be solved by adopting the sensing technology in an offshore environment. Among the possible benefits are:

- work remotely from anywhere over the Internet,
- work reliably without user error, as can happen with manual measurements and recordings,
- ensure immediate action by making real-time decisions,
- work smarter with efficient and timely condition-based maintenance and early detection,
- save money and space without cable runs to each sensor,
- work with confidence in dirty, harsh and hazardous conditions,
- reduce installation and maintenance costs.

Tremendous effort has gone into the development of wireless acoustic networks for use underwater. However, the most common commercial acoustic modems are designed mainly for point-to-point communication rather than for network operations. In order to produce practical, low cost ad hoc network systems, several challenges must be addressed. These include high bit error rates due to multi-path scenarios, path redundancy, complex architecture and increased network functionality. In addition to those, a limited bandwidth may be a critical issue when a large number of sensors are in use. Advanced processing and system designs are being developed, which will enable higher data rates and more robust links; they will also address latency,

reflections, and ray divergence. Consequently and not surprisingly, alternative subsea wireless technologies for subsea sensor networks has attracted the interest of researchers in the field; a particular interest has been shown for highly instrumented subsea production equipment.

6.2.2 Implementing the UWSNs

The properties of the world's oceans include swift temporal and spatial fluctuations which require new measurement techniques in order to increase resolution in oceanographic measurements. An increase in today's conventional methods for oceanic sampling would be both time consuming and cost ineffective; it would also demand large amounts of resources. Changes in ocean temperatures and currents are bringing to reality what was before just a threat of climate changes caused by greenhouse gases. Early prediction of these effects is vital, and an increase in synoptic sampling of oceanic properties is demanded. Considering these conditions, underwater wireless sensor networks can be the best choice for a rapid environmental assessment as well as for increasing the resolution of oceanographic measurements in space and time [122].

An efficient means for adaptive sampling of enormous areas in a fairly short timeframe has been provided for by the development of Autonomous Underwater Vehicles (AUVs) for scientific use. AUVs may be equipped with a variety of sensors and passive sampling devices; this gives them a strong position as a convenient platform [123-125]. Another advantage with AUVs are the potential to sample e.g. oxygen, Conductivity, Temperature, Depth (CTD) and current properties in a three dimensional space. Although there are some great advantages with AUVs, it is clear that they also have disadvantages. The facts that they may malfunction and be lost during a mission, or something could go wrong and the data obtained in between surface might be lost are the two major disadvantages. Others arethat, they are rather expensive even though the technology is improving and they usually require a fairly large amount of manpower for deployment/retrievement, service and monitoring.

Intelligent subsea wells require continuous real-time reservoir surveillance, often through downhole controls, to evaluate production variations and to track wellbore performance; however measurements such as temperature, flow rate, pressure, and chemical properties can all be made. Moving multiphase metering technology downhole enables operators to monitor and control flow behavior more closely in wells with multiple producing zones. A higher daily output without risk to production from, for example, excessive sand or water is the result. The combined effect is a better reservoir understanding; ultimately, recovery is increased.

New oil and gas field developments are increasingly more advanced and more often than not include subsea installations, satellite wells, or subsea-to-beach applications. Deepwater and harsh environment areas which may hold significant oil and gas reserves offer major growth prospects for producers to meet future oil and gas demands. However, these inhospitable offshore conditions pose different operational challenges; one of the most critical is ensuring efficient flow of oil and gas production. To find a solution to this challenge, an online or real time system for monitoring subsea equipment and well production is necessary in order for operators to receive vital information about the production content. This flow assurance challenge increases as the flowline length and water depth increases. Moreover, other critical situations such as equipment failure, wearing down of the choke, and leakage or blockage of pipelines can also arise. Adverse situations like flow rate issues or equipment malfunction, not only cost the operator time and money, but also often result in an environmental catastrophe. If changes in normal conditions are detected early, unplanned shut-downs of wells may be avoided.

UWSNs employ different number of sensor nodes; in this way costs are restricted while a high spatial and temporal sampling of limited areas is achieved. It is believed that networks of these types would make use of generic sensors of a limited size and consequently, a small battery pack in order to achieve easy deployment strategies with some buoyancy control, and utilization of fast moving vessels. A dense deployment with an internode distance below 1 km would be encouraged as a result of the limited battery and general complexity of the acoustic communication. Utilizing UAN presupposes good localization algorithms to achieve the required geographical

metadata that is necessary in oceanographic measurements. The greatest advantage of UAN might just be its ability for almost real time monitoring by acoustic communication links, making it far more reliable than moorings.

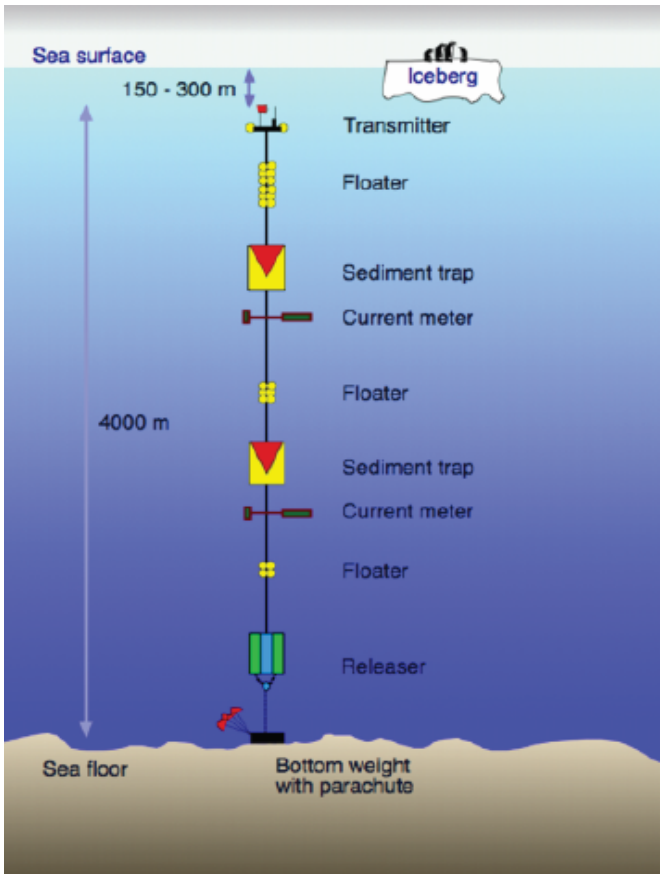

Figure 6.3 An example of deep sea mooring [126]

UWSNs can be beneficial for the continuous monitoring of the production performance parameters for each individually perforated zone in a multilayer well. Measurements such as temperature, flow rate, pressure, fluid fraction and chemical properties can all be collected with their help. Monitoring of temperature, pressure, and water cut, as well as gas fraction, sand rate, and flow velocity can be accomplished by placing the sensors between each production zone; thereby allowing the sensors to monitor production performance parameters continuously in each individually perforated zone of a multilayer well. This intelligent downhole sensor network reduces uncertainty by increasing reservoir knowledge. Furthermore, these sensors can also detect any unwanted activity and in some cases instructions can be

given to prevent any further increase in the trouble. For example, if unwanted water or gas enters the wellbore, the sensors can detect the change in the multiphase composition at the subsea wellhead, providing the operator with real-time information from the downhole pressure and temperature gauges for examination which eventually can lead to identification of the location of the problem. Decisions such as choke setting and artificial lifts, for example, can also be made when maximum amount of information is provided.

Expensive and inconvenient conduits can be eliminated by wireless sensing systems; accurate collection of measurement data can be carried out in real-time for faster response and decision-making without any loss in system integrity or availability. Furthermore, the number of personnel once needed to manually perform duties can be minimized by employing wireless systems. These wireless sensing systems offer an affordable choice for offshore production platforms. The cost to equip an entire platform with wireless networks fits well within the budget of most offshore operations and in the future it will be even cheaper. Oil and gas companies can now afford to integrate sensors at process points which have been unavailable in traditional wired networks.

To be precise, wireless sensing methods can provide the following readings during offshore oil & gas monitoring activities,

1. Operating pressure of the fluid being measured.
2. Internal pressure of the sensor's body cavity.
3. Temperature of the fluid being measured.
4. Temperature of the electronics inside the sensor.
5. Expected remaining battery life.

6.3 Marine Pollution

It is sad but true that the ocean has increasingly become the end station for large amounts of land based refuse. For the most part, marine pollution originates from land based sources which include household waste, agricultural runoffs, sewage and different toxic chemicals. Toxins may have their origin in, e.g. wind carried pesticide

or polluted rivers. In the ocean, these toxins are broken down into many small particles which are absorbed by benthos and plankton. Once in the ecosystem, the pollution concentrates within the food chain and all parts of sea life are contaminated. Many chemical particles carried along by rivers and streams may combine chemically in a way that makes estuaries anoxic and cause the extermination of all aquatic life. Since marine resources are often used for food, heavy metals and dangerous chemicals may eventually end being consumed by humans or land animals which are then contaminated. In addition to the spread of toxins and mortality, this contamination may be the cause of mutations or changes in biochemistry, tissue matter or even an inability for reproduction in both humans and animals [127].

Of great concern is the increase in ship traffic and deep sea oil & gas extraction. It appears that the majority of pollution within the ocean has its origin in operative discharge from ships as well as accidents on oil tankers and platforms. The amount of oil spilled annually worldwide is estimated at more than 4.5 million tons; this is equivalent to two super tanker accidents weekly. Moreover, water pollution is an even more predominant problem in the Third World; where the main source of water for drinking and sanitation comes from unprotected streams and ponds that are contaminated with human waste. Millions of people use this water every day and it is this type of contamination which is estimated to cause more than 3 million deaths from diarrhoea, annually; these are mostly children.

6.3.1 Implementing the UWSNs

Vast adaptive sampling of pollutants in specific maritime environments is definitely of vital importance today; however, there is quite a lack of effective approaches to be utilized for detection, monitoring, analyzing and prediction of marine pollution. The methodology presently applied whereby active and passive sampling is carried out may not be suitable for monitoring the continued spreading of marine pollutants now and in the future. To assess the near future threat of marine pollution, it is imperative to develop a system for vast adaptive sampling.

Monitoring of pollutants in the ocean is no less a challenge than that of monitoring physical oceanic parameters. The platforms available for on-site sensing of pollutants are very similar to those for oceanographic use. Many pollutants may only be detected when utilizing analytical chemistry which is quite a large drawback in regards to pollution monitoring; consequently their presence will possibly be undetected by sensors deployed or mounted on AUVs.

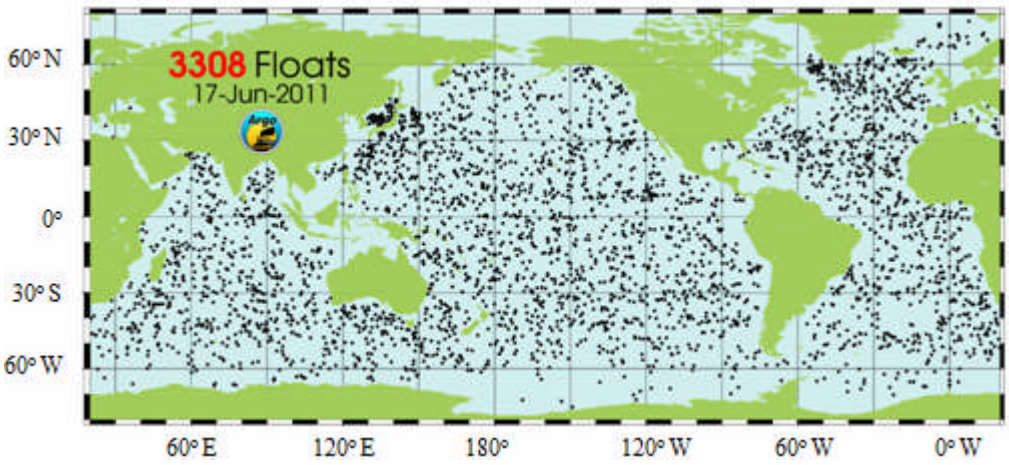

Figure 6.4 Latest positions of the floats (used to sense and deliver data) [128]

Utilizing a nephelometer to measure turbidity and oceanographic sensors for oxygen levels are possibly still applicable for a general impression of the quantity of contaminants in an environment. Measurement of oxygen levels is rather useful especially for monitoring rising levels of nutrients which might make the water volume hypoxic. Underwater wireless sensor networks provide a possible effective solution for monitoring limited coastal areas and estuaries. UWSNs are expected to work best with short nodal spacing and rather simple sensing equipment; this would make them a promising solution for measuring turbidity or oxygen levels in smaller areas. Equipped with a custom designed AUV, an underwater sensor can be deployed quickly in case of an emergency to be used for tracking oil spills. Another application which is quite suitable is the monitoring of discharge from offshore installations or waterside facilities. The capability of UWSNs of carrying out synoptic sampling in a limited environment for the purpose almost real-time monitoring represents a great resource marine pollution investigation in high risk environments. The use of UWSNs for the purpose of pollution monitoring is discussed to some extent in [129].

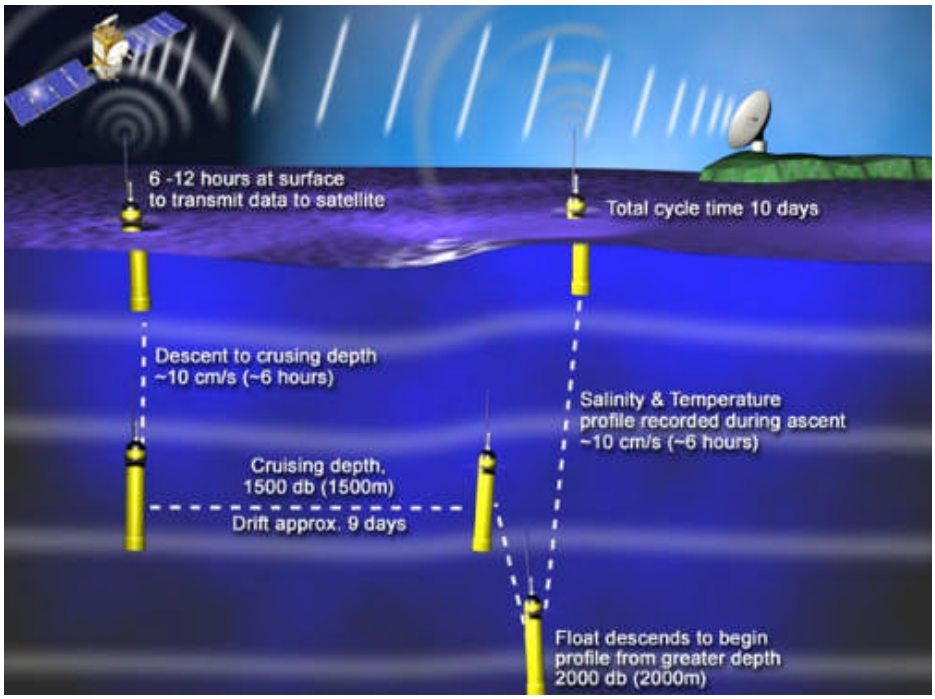

Figure 6.5 Task completing cycle of the Argo drifter [128]

6.4 Marine Biology

Marine biology is the study of living organisms in the sea or other marine environments. Marine biologists not only study the variety of organisms which make up life in the sea, but also the effect different substances have on that sea life. It is closely related to oceanography and, consequently, the study of the temperature, salinity and dynamic changes of the sea.

Like oceanography, marine biology is a relatively new science and has its origin in the study of land creatures. The hypothesis that life started in the sea is the main motivation for studying sea life. Marine biology covers the study of one cell organisms and bacteria via photosynthesis in deep water to the migration of large marine mammals. It is believed by scientists that sea life can tell us much about the evolution of the Earth, and it gives probable indications regarding changes which occurred in the past as well as changes occurring now. However, life in the sea is related to the dramatic changes happening in the atmosphere and, in turn, to life on land is still to be fully understood. Since vast areas of the ocean have yet to be

unexplored, it is unknown as to what new information about evolution as well as future resources might be contained therein.

Marine resources are tremendously significant for our lifestyle as well as for life on Earth, in general. We are provided with food, medicine and minerals from resources found in the oceans and seas; furthermore, even tourism which affects our economy is influenced by them. Consequently, marine life influences all aspects of the biology on Earth and also affects the climate and the oxygen cycle. A vital part of biology is the study of habitats and ecosystems. The change in physical or chemical properties in a specified environment could have a vital impact on the quantity and quality of life there. Monitoring of a habitat allows scientists to obtain relevant data in regards to the abundance of species and the conditions surrounding their life or even reproduction. Habitats to be monitored include the layers of surface water which suffer from rapid changes in properties, reefs with are a rich environment containing hundreds of species, and deep sea trenches at depths of several thousand meters where no sunlight is available [130].

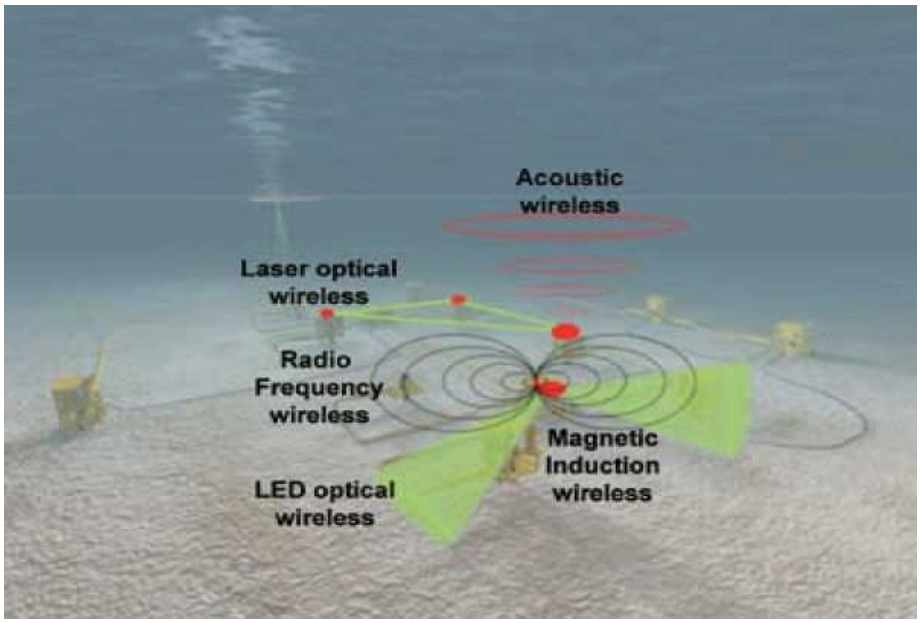

Figure 6.6 Schematic of an integrated subsea wireless system comprising acoustic, optical, and magnetic induction systems [131]

6.4.1 Implementing the UWSNs

Techniques used to monitor a specific habitat in the ocean today consist of active sampling and field work in order to decide on the biodiversity and species abundance available in that given area. For habitats on land, this work is partly taken over by distributed sensor networks [132, 133]. This approach can, to a certain degree, also be used in underwater habitats, as reported by [134] and [135]. Utilizing passive acoustic detection with a UWSN may simplify marine habitat monitoring as well as limit the costs involved. The fact that oceanic habitats are limited in size makes them a proper platform for UWSNs. Passive acoustic detection has been found to be less energy demanding; moreover, reducing the temporal sampling points can result in even less power consumption. One can imagine different sensors sampling at different intervals and submitting raw data to the control centre for assessment. A small reef, lagoon or other habitat under consideration for monitoring need not be larger than enough to be covered by 10 - 100 sensor nodes. Consequently, the network would be quite easy to manage and control. However, the life-time of such systems depends on the temporal sampling as well as the power needed for recording, processing and communicating. It's hard to imagine that underwater sensor networks for habitat monitoring could last longer than a full year without being recharged or redeployed. For sensor network systems, there are still many variables to be determined in regards to battery capacity and the energy requirements of passive acoustic sensors, which are not discussed here.

USWNs may also make use of conventional acoustic remote sensing equipment in a configuration where several receiver nodes record backscattered signals, and one or more source nodes initiate an event. This approach can supply the same sort of data achieved by remote sensing done with modern applications today.

6.5 Waterside Security

Rick Benavidez, Director of Security and Safety at Port Freeport, reported that with the deployment of the Ciber and Sonardyne system, 'We now have landside waterside and underwater detection systems. The Port's financial investment in these security initiatives shows support for a "safe and secure Port".' Further, he pointed out that it

makes good commercial sense to have good waterside security in today's economic climate: 'We feel our customers do look at those ports which can protect their investments, i.e., their cargo and vessels while at the dock, while at the same time facilitating commerce.'

Technology is moving in the direction of more unmanned self operating subsea structures in the area of offshore exploitation of natural resources as well as other kinds of maritime infrastructure like fish farms and wave power plants. These types of installations are very complex, consisting of a variety of constructions spread over a vast area underwater. In today's society, the security situation leaves those rigs in a high position of vulnerability in regards to attacks in a variety of ways as well as unconventional warfare. Busy harbours are another potential target where hundreds of shipping vessels are being taken care on a daily basis. A strike against any such marine infrastructure may have a large economic and environmental impact. This means that the area of waterside security (WSS) is not the unique concern of the military any longer [136].

Divers with/without scooters, speed boats, unmanned underwater vehicles and AUVs are all potential threats having different signatures and possible approach trajectories; therefore an advanced system to detect all kinds of threats is vital [136]. One of the greatest challenges is the ability to identify covert threats hidden within the normal daily activities; these threats have the potential of being disguised as a small cargo vessel. Protection of sea infrastructure is further complicated by the relatively easy access provided to the public.

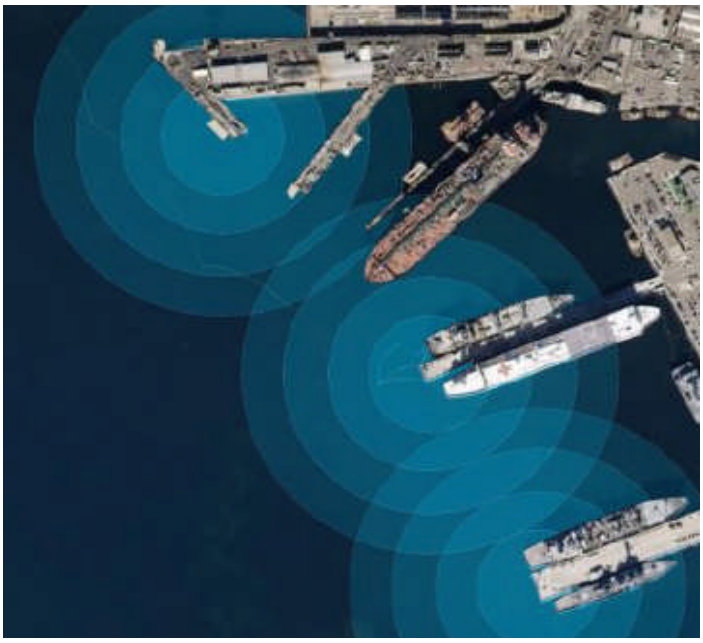

Figure 6.7 Probable diver detection range at a port [137]

Harbour areas have the advantage of being situated on the shore. Consequently, they have easy access to security personnel and other land resources. Installations far at sea, on the other hand, have limited resources which make the security situation more complicated and yet this challenge has not been addressed seriously. Civilian marine security systems appear to spend their time and energy focused mostly on harbour protection, inspired by military applications for the same purpose. All the while, there is a continuous and increasing demand for hydrocarbons which is in no way leading towards a decrease in offshore activity. Furthermore, the technology for utilizing renewable energy offshore is being developed steadily. These are not the only areas in need of better security systems. Wave energy plants and possible offshore windmills are other examples of applications which demand that waterside security systems be developed for future use.

6.5.1 Implementing the UWSNs

Military installations generally have high end security and are quite protected both under and over the surface. Furthermore, port security has received a lot of attention

recently and with the AUVs being introduced for commercial use cost effective underwater patrolling of large harbour areas has been achieved. [138] presented several concepts of AUVs for waterside security use which when utilized together with coast guard patrols and strict routines, has made the security situation within a port facility manageable. However, the task of providing complete coverage of all arriving and departing traffic as well as isolating any security threat is still extremely difficult, and a vast adaptive sampling of the total area under protection and short reaction time for countermeasures are vital for security.

For protection of waterside and offshore installations, the surveillance system used is commonly based on an integrated method [139]. This involves the use of several sensing systems and sensor platforms. For surveillance above the surface, it is common practice to utilize radar and patrol boats, but various sensing systems can be employed under the surface as well. Often a combination of surveillance systems with floating barriers and control points are used for prevention purposes. In this situation, acoustic sensors cab be installed on existing constructions, such as piers or platforms, or positioned at predetermined points on specific structures to obtain a much wider coverage [139]. In order to inspect the hull, an effective method is the use of sensors positioned on the seabed.

If the areas to be monitored are high risk, then an ad hoc network of sensor nodes can offer a good coverage for marine surveillance. The sensor network consists of a number of receiver nodes which are able to analyze acoustic signals, some source nodes with acoustic transmitters, and a few gateway nodes with satellite links. The nodes are presented with some form of buoyancy element containing a short distance radio for internal communication, an acoustic receiver and a processing unit for analyzing received signals. When there is a detection noted, a report is sent to an onshore control centre to analyze the data, and determine the type of alarm and response needed.

When utilizing an UAN, a similar method can be used, only the floating sensor nodes are replaced with subsurface sensor nodes. The advantage of installing the network on the bottom of the ocean is that it is not in the way of ship traffic; moreover, it could also be used for hull inspection in order to detect reflections of

passing ships emitted by a central source. However, the possible disadvantage for these scenarios is related to battery consumption; positioning on the seabed means that changing the battery may be quite a challenge compared to floating nodes. On the other hand, having a system which is deployed close to some particular infrastructure, and possesses a mechanism whereby the node could pop up to the surface when it reaches a critical battery level is a possible solution to this challenge. In this way, a node which requires a battery change could be located quickly by use of a radio beacon; the battery could be easily changed and the node redeployed.

In the case of active sonar systems where the sensor node works on a regular basis, depending on the shot intervals of the source to analyze received signals, a passive acoustic solution may be more feasible in order to lengthen the lifetime of the network. This would result in the node only reacting when distinct acoustic signatures are detected, thereby saving processing energy. . Another way of going at this challenge is to avoid onboard processing and instead send the data directly to onshore control centre; however, this would necessitate having a fairly high data rate which might not be available in an UAN. In regards to altering the position in relation to its battery life, [140] proposes a sensor network for passive diver detection, and gives a good discussion on sensor placement versus probability of detection. This network configuration presupposes local processing and, in the case of intruder detection, the initiation of an alarm. The power required to do this processing is not stated, and a trade-off between probability of detection and energy consumption is highly expected. UWSNs can help to increase the waterside security in many situations such as follows:

- Port protection.
- Protection of critical infrastructure.
- Protection of high-profile events.
- Small boat monitoring and stopping.
- Protection of ships and energy platforms.
- Prevention of underwater intruders.
- Maritime interdiction, maritime surveillance and information sharing.

CHAPTER 7

CONCLUSION

An overview of the conducted research along with contribution and work considered to be necessary as a future work are discussed in this chapter.

7.1 Overview

The goal of this thesis is to explore the problem of data gathering in the inhospitable underwater environment and a solution is provided in the form of a distributed routing protocol. According to our best knowledge, H2-DAB is first addressing based routing approach for underwater sensor networks and not only has it helped to choose the routing path faster but also efficiently enabled a recovery procedure in case of smooth forwarding failure. Novelty of this protocol is that it does not require full dimensional location information as well as there is no need to maintain the complex routing tables. We attempt to keep the routing overheads minimum as available data rates are extremely low for the UWSN. In this research, the idea of *per-contact routing*, instead of *source routing* or *per-hop routing* according to the underwater requirements is used. An important fact about the H2-DAB is that, the delivery ratios are not seriously based on the density or sparseness of sensor nodes. Node mobility due to water currents is a challenge for underwater routing but easily handled with the proposed protocol. The problem of node failure - a major threat and possibility for UWSN - is not a serious problem for H2-DAB. New nodes can be added at any time and at any location, and these new nodes can easily configure during next interval. The hop-by-hop nature of the proposed scheme makes it robust where intermediate nodes can respond more appropriately to rapid changes. The exchange of control packets remains small as intermediate nodes does not maintain global knowledge of whole network. Proposed protocol is suitable for long term applications even covering vast

areas with large number of nodes, because every node obtain its routing address despite any effect of network size. The proposed technique shown is to achieve performance targets by means of simulation and the publication with these results is accepted with minor revision in Elsevier's *Computer Communications*.

For UWSNs, due to continuous node movements and sparseness, it is possible that, the receiving node can not find the next hop for long intervals in order to reach the destination and during this, it can die due to limited power or any failure can occur due to fouling and corrosion problems. In such cases all the packets held by the current node will be lost permanently because none of the other node maintains the backup of these lost data packets. In these situations, traditional Hop-by-Hop acknowledgement (HbH-ACK) techniques are not suitable as single node is responsible for current data packets. In order to handle this challenge, H2-DAB provide two hop acknowledgement (2H-ACK) reliability model where two copies of the same data packet are maintained in the network without extra serious burden on the available network resources.

7.2 Summary of Contribution

Besides the advantages such as low cost and requiring no dimensional location information for task completion, the following are some important aspects of H2-DAB

Node Address Expiry: H2-DAB allows an auto update of its *HopID* upon receiving new Hello packets and then forwards the buffered packets in the same way just like before the expiry of the old *HopID.* Thus, old packets need not be discarded when the address of a node expires.

Loop Free Routing: The occurrence of loops during the routing decisions, especially during address assignment is common for dynamic addressing based protocols. However, H2-DAB is sensitive and intelligent enough to avoid the occurrence of such routing loops.

Nodes Lost and Found: In H2-DAB, every node has only one entry in each of their routing table. Hence, to add or to delete entries when nodes are lost or added in the network is unnecessary. Newly deployed nodes will obtain their *HopIDs* at the next interval and automatically become a part of the network. H2-DAB is highly adaptive to network dynamics such as nodes joining and leaving the network for its reactive hop-by-hop packet routing mechanism.

Node Movements are easily handled: Vertical node movements are very common for these networks and as a result, a node can change its neighbors both with upper and lower layers. Even the neighbors are continued to change, these addresses can still be used as address of any node that will remain smaller from the addresses of lower nodes and larger from the upper nodes.

Problem of Table Size: The growth of the routing table is a serious problem for dynamic addressing based networks. For H2-DAB, the size of routing table is not affected by the network size as it will remain of the same size and every node maintains a table of one entry even when the network consists of a large number of nodes.

Problem of Address Space Exhaustion: Most of the dynamic address assignment schemes used for the ad hoc networks face the issue of address space exhaustion, but not in H2-DAB, as addresses will remain of 2 digits per HopID as well as multiple nodes that can use the same address without any problem during data deliveries.

Destination Address Changes: Final destination address for all the surface sinks is similar and static, so the problem of destination address change during routing decisions does not occur. Additionally, packets can be delivered to any nearest surface sink.

Destination Movement Flexibility: Some protocols assume that destination is fixed and unable to change its location. However, it seems not to be always true due to the water currents. While some other like [11] assumes that destination movement is predefined and already known to all the sensor nodes before launching, For H2-DAB, no such assumption is made. All sinks can move and can still receive data packets easily.

Routing Decisions without Maintaining Global Knowledge: intermediate nodes forward data packets without maintaining global knowledge of whole network which strongly helps to decrease the communication of the control packets.

Monitoring areas with normal depths: In most of the applications, the monitored areas are with a depth of not more than 1 km, H2DAB in such environments provides even better results (with 4-5 hops).

7.3 Open Issues and Future Work

Throughout the course of this thesis, several issues related to the UWSN have been outlined. H2-DAB in turn is the answer for many of them. Several avenues exist and further research is suggested in the following directions in order to improve the performance of the proposed scheme.

1. Deployment model*:* As previously mentioned, a proposed scheme is based on hop-by-hop nature so the problem of multi-hop routing exist in which, for being used more frequently, nodes near the sink drain more energy . Although H2-DAB, due to its multi-sink architecture, has reduced this effect in that the burden of few nodes is distributed on whole layer as we have also proved with the analytical model, this problem still requires more attention. Research is needed to include a deployment model where it is possible to adjust the distance of sensor nodes between inter layers and intra layers to balance the energy consumption and fairness among sensor nodes in whole network.

2. Acknowledgements and Recovery of Lost Data Packets: H2-DAB contains two algorithms, (1) the one used for the dynamic address assignment and (2) the one to help to deliver the data packets at one of the surface sinks with best effort approach. Although a 2H-Ack reliability model is proposed as a part of thesis; further research is suggested in order to improve the reliability by providing end-to-end ACKs and recovery of lost data packets as a result of node failures.

3. Optimizing the Courier Node Utilization: As previously mentioned, H2-DAB can complete its task without introducing even a single courier node, however here these special nodes to decrease the energy consumption are suggested. During the simulations in this thesis, 2% to 4% courier nodes of all the sensor nodes was used and further research is required to optimize their requirement. Utilizing the courier nodes with optimum not only can help to reduce the cost of network but also will ultimately improve the life of the network.

4. Integration with Underwater MAC protocols: In this thesis, a general MAC/802.11 is used with H2-DAB as a MAC layer protocol. A possible future work is to integrate H2-DAB with specialized underwater MAC protocols such as R-MAC [146] in order to investigate its relative performance.

REFERENCES

1. Akyildiz, I.F., D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: research challenges," *Ad Hoc Networks*, 3(3): 2005, p. 257-279.

2. Heidemann, J., et al., "Research challenges and applications for underwater sensor networking," *Wireless Communications and Networking Conference, 2006.* WCNC 2006. IEEE. 2006.

3. Jiang, Z., "Underwater Acoustic Networks- Issues and Solutions," *International Journal of Intelligent control and systems*, 2008. 13: p. 152-161.

4. Partan, J., J. Kurose, and B.N. Levine, "A survey of practical issues in underwater networks," *Proceedings of the 1st ACM international workshop on Underwater networks*. 2006, ACM: Los Angeles, CA, USA.

5. Chitre, M., et al., "Underwater Acoustic Communications and Networking: Recent Advances and Future Challenges," *OCEANS 2008*.

6. Yan, H., Z.J. Shi, and J.-H. Cui, "DBR: depth-based routing for underwater sensor networks," *Proceedings of the 7th international IFIP-TC6 networking conference on AdHoc and sensor networks, wireless networks, next generation internet*. 2008, Springer-Verlag: Singapore.

7. Kai Chen, Y.Z., Jianhua He, "A Localization Scheme for Underwater Wireless Sensor Networks," *International Journal of Advanced Science and Technology*, 2009. Vol. 4.

8. Erol, M. and S. Oktug, "A localization and routing framework for mobile underwater sensor networks,". *INFOCOM Workshops 2008*, IEEE. 2008.

9. Jornet, J.M., M. Stojanovic, and M. Zorzi, "Focused beam routing protocol for underwater acoustic networks," *Proceedings of the third ACM international workshop on Underwater Networks*. 2008, ACM: San Francisco, California, USA.

10. Xie, P., J.-H. Cui, and L. Lao, "VBF: Vector-Based Forwarding Protocol for Underwater Sensor Networks," *Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems*. 2006, Springer Berlin / Heidelberg. p. 1216-1221.

11. Chirdchoo, N., S. Wee-Seng, and C. Kee Chaing, "Sector-Based Routing with Destination Location Prediction for Underwater Mobile Networks," *Advanced Information Networking and Applications Workshops, 2009.* WAINA '09.

12. Jinming, C., W. Xiaobing, and C. Guihai, "REBAR: A Reliable and Energy Balanced Routing Algorithm for UWSNs," *Grid and Cooperative Computing, 2008.* GCC '08.

13. Daeyoup, H. and K. Dongkyun, "DFR: Directional flooding-based routing protocol for underwater sensor networks," *OCEANS 2008*.

14. Carlson, E.A., P.P. Beaujean, and E. An., "An Improved Location-Aware Routing Protocol for Mobile Underwater Acoustic Networks," *OCEANS 2007*.

15. Seah, W.K.G. and H.P. Tan, "Multipath virtual sink architecture for wireless sensor networks in harsh environments," *Proceedings of the first international conference on Integrated internet ad hoc and sensor networks, 2006,* ACM: Nice, France.

16. Ali, K. and H. Hassanein, "Underwater Wireless Hybrid Sensor Networks," *Computers and Communications, 2008.* IEEE Symposium. 2008.

17. Dario Pompili, T.M., Ian F. Akyildiz., "A Resilient Routing Algorithm for Long-term applications in Underwater Sensor Networks," *MedHocNet*, 2006.

18. Bin, Z., G.S. Sukhatme, and A.A. Requicha, "Adaptive sampling for marine microorganism monitoring," *Intelligent Robots and Systems, 2004. (IROS 2004),* IEEE/RSJ International Conference, 2004.

19. S1510, "Underwater Radio Modem," 2007; Available from: http://www.wirelessfibre.co.uk/Content/press/press%20releases/UnderWaterRadioModemS1510.pdf

20. Quazi, A. and W. Konrad, "Underwater acoustic communications," *Communications Magazine, IEEE*, 1982. 20(2): p. 24-30.

21. Farr, N., et al., "Optical modem technology for seafloor observatories," *OCEANS, 2005,* Proceedings of MTS/IEEE. 2005.

22. Sozer, E.M., M. Stojanovic, and J.G. Proakis, "Underwater acoustic networks," *IEEE Journal of Oceanic Engineering,* 2000. 25(1): p. 72-83.

23. Stojanovic, M., "On the relationship between capacity and distance in an underwater acoustic communication channel," *Proceedings of the 1st ACM international workshop on Underwater networks*. 2006, ACM: Los Angeles, CA, USA.

24. Lanbo Liu, S.Z., and Jun-Hong Cui, "Prospects and Problems of Wireless Communications for Underwater Sensor Networks," *Wiley's Wireless Communications and Mobile Computing*, (Special Issue on Underwater Sensor Networks), May 2008.

25. Olariu, S. and J.V. Nickerson, "Protecting with sensor networks: perimeters and axes," *Military Communications Conference, 2005.* IEEE. 2005.

26. Megerian, S., et al., "Worst and best-case coverage in sensor networks," *IEEE Transactions on Mobile Computing*, 2005. 4(1): p. 84-92.

27. Jain, E. and L. Qilian, "Sensor placement and lifetime of wireless sensor networks: theory and performance analysis," *Global Telecommunications Conference, GLOBECOM '05*, IEEE. 2005.

28. Pompili, D., T. Melodia, and I.F. Akyildiz, "Routing algorithms for delay-insensitive and delay-sensitive applications in underwater sensor networks," *Proceedings of the 12th annual international conference on Mobile computing and networking*, 2006, ACM: Los Angeles, CA, USA.

29. Aitsaadi, N., et al., "Differentiated Underwater Sensor Network Deployment," *OCEANS 2007* - Europe. 2007.

30. Ibrahim, S., J.H. Cui, and R. Ammar, "Efficient surface gateway deployment for underwater sensor networks," *Computers and Communications, ISCC 2008. IEEE Symposium*, 2008.

31. Alsalih, W., H. Hassanein, and S. Akl, "Placement of multiple mobile data collectors in underwater acoustic sensor networks," *Wiley's Wireless Communication and Mobile Computing*, 2008. 8(8): p. 1011-1022.

32. Jun-Hong, C., et al., "The challenges of building mobile underwater wireless networks for aquatic applications," *Network, IEEE*, 2006. 20(3): p. 12-18.

33. Erol, M., L.F.M. Vieira, and M. Gerla, "Localization with Dive'N'Rise (DNR) beacons for underwater acoustic sensor networks," *Proceedings of the second workshop on Underwater networks*. 2007, ACM: Montreal, Quebec, Canada.

34. Peng, S., W.K.G. Seah, and P.W.Q. Lee, "Efficient Data Delivery with Packet Cloning for Underwater Sensor Networks," *Symposium on Underwater Technology and Workshop on Scientific Use of Submarine Cables and Related Technologies*, 2007.

35. Holland, G. and N. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. 1999, ACM: Seattle, Washington, United States.

36. Scheuermann, B., C. Lochert, and M. Mauve, "Implicit hop-by-hop congestion control in wireless multihop networks," Ad Hoc Network, 2008. 6(2): p. 260-286.

37. Peng, W.B.a.T.-H., "Tracers in the sea," *ed. Eldigio Press*. 1982, NY: Lamont Doherty Earth Observatory of Columbia University,

38. Ayaz, M. and A. Abdullah. "Hop-by-Hop Dynamic Addressing Based (H2-DAB) Routing Protocol for Underwater Wireless Sensor Networks," *International Conference on Information and Multimedia Technology, 2009*, ICIMT '09.

39. Xie, P., et al., "SDRT: A reliable data transport protocol for underwater sensor networks," *Ad Hoc Networks*, 2006. 8(7): p. 708-722.

40. Peng, X., et al., "Efficient Vector-Based Forwarding for Underwater Sensor Networks," *Hindawi Publishing Corporation* 2010.

41. Yarvis, M., et al., "Exploiting heterogeneity in sensor networks," *INFOCOM 2005*, 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE. 2005.

42. Akkaya, K. and M. Younis, "An Energy-Aware QoS Routing Protocol for Wireless Sensor Networks," *Proceedings of the 23rd International Conference on Distributed Computing Systems '03*, IEEE Computer Society 2003,.

43. Zhou, Z., et al., "Efficient multipath communication for time-critical applications in underwater acoustic sensor networks," *IEEE/ACM Transaction Netwworks*, 2011. 19(1): p. 28-41.

44. Stojanovic, M., "Underwater acoustic communication" *Wiley Encyclopedia of Electrical and Electronics Engineering*, 22, 1999.

45. Lysanov, L.B.a.Y., "Fundamentals of Ocean Acoustics," Vol. 8. 1982, New York: Springer Series.

46. Coates, R., "Underwater Acoustic Systems," 1989, New York: Wiley.

47. Ayaz, M. and A. Abdullah, "Underwater wireless sensor networks: routing issues and future challenges," *Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia*. 2009, ACM: Kuala Lumpur, Malaysia.

48. Domingo, M.C. and R. Prior, "Energy analysis of routing protocols for underwater wireless sensor networks," *Computer Communications*, 2008. 31(6): p. 1227-1238.

49. Stojanovic, M., "Acoustic (Underwater) Communications," *Wiley Encyclopedia of Telecommunications*, (John G. Proakis, John Wiley & Sons, 2003).

50. Proakis, J.G., et al., "Shallow water acoustic networks," *Communications Magazine*, IEEE, 2001. 39(11): p. 114-119.

51. Ethem M. Sozer, M.S., and John G. Proakis, "Underwater Acoustic Networks," *IEEE JOURNAL OF OCEANIC ENGINEERING*, Vol 25, No. 1, Janaury, 2000.

52. Partan, J., J. Kurose, and B.N. Levine, "A survey of practical issues in underwater networks," *SIGMOBILE Mobile Computer Communication. Review*, 2007. 11(4): p. 23-33.

53. Akyildiz, I.F., D. Pompili, and T. Melodia, "State-of-the-art in protocol research for underwater acoustic sensor networks," *Proceedings of the 1st ACM international workshop on Underwater networks*. 2006, ACM: Los Angeles, CA, USA.

54. Ovaliadis K., N.S.a.V.K., "Energy Efficiency in Underwater Sensor Networks: a Research Review," *Journal of Engineering Science and Technology Review*, June 2010. 3(1): p. 151-156.

55. Pompili, D., T. Melodia, and I.F. Akyildiz, "Three-dimensional and two-dimensional deployment analysis for underwater acoustic sensor networks," *Ad Hoc Network*, 2009. **7**(4): p. 778-790.

56. Jiejun, K., et al., "Building underwater ad-hoc networks and sensor networks for large scale real-time aquatic applications," *Military Communications Conference, 2005.* MILCOM 2005. IEEE. 2005.

57. Lucani, D.E., et al., "Network coding schemes for underwater networks: the benefits of implicit acknowledgement," *Proceedings of the second workshop on Underwater networks*. 2007, ACM: Montreal, Quebec, Canada.

58. Casari, P., S. Marella, and M. Zorzi, "A Comparison of Multiple Access Techniques in Clustered Underwater Acoustic Networks," *OCEANS 2007* - Europe. 2007.

59. Frampton, K.D., "Acoustic self-localization in a distributed sensor network," *Sensors Journal*, IEEE, 2006. 6(1): p. 166-172.

60. Mahajan, A. and M. Walworth, "3D position sensing using the differences in the time-of-flights from a wave source to various receivers," *IEEE Transactions on Robotics and Automation*, 2001. 17(1): p. 91-94.

61. Bulusu, N., J. Heidemann, and D. Estrin, "GPS-less low-cost outdoor localization for very small devices," *Personal Communications*, IEEE, 2000. **7**(5): p. 28-34.

62. Nath, D.N.a.B., "DV based positioning in ad hoc networks," *Springer, Telecommunication Systems*, 2003: p. 267-280.

63. Zhou, Z., J.-H. Cui, and S. Zhou, "Efficient localization for large-scale underwater sensor networks," *Ad Hoc Network*. 8(3): p. 267-279.

64. Thomas, C.B.a.H., "GIB system: The underwater GPS solution," *Proceedings of 5th Europe Conference on Underwater Acoustics*. 2000.

65. Austin, T.C., R.P. Stokey, and K.M. Sharp, "PARADIGM: a buoy-based system for AUV navigation and tracking," *OCEANS 2000 MTS/IEEE Conference and Exhibition*. 2000.

66. Yuecheng, Z. and C. Liang, "A Distributed Protocol for Multi-hop Underwater Robot Positioning," *Robotics and Biomimetics, ROBIO 2004.* IEEE International Conference 2004.

67. Nicolaou, N., et al., "Improving the Robustness of Location-Based Routing for Underwater Sensor Networks," *OCEANS 2007* - Europe. 2007.

68. Johnson, D.B., D.A. Maltz, and J. Broch, "DSR: the dynamic source routing protocol for multihop wireless ad hoc networks," *Ad hoc networking*. 2001, Addison-Wesley Longman Publishing Co., Inc. p. 139-172.

69. Carlson, E.A., P.P. Beaujean, and E. An., "Location-Aware Routing Protocol for Underwater Acoustic Networks," *OCEANS,* 2006.

70. Domingo, M.C. and R. Prior, "A Distributed Clustering Scheme for Underwater Wireless Sensor Networks," *Personal, Indoor and Mobile Radio Communications, PIMRC 2007.* IEEE 18th International Symposium on. 2007.

71. Anupama, K.R., A. Sasidharan, and S. Vadlamani, "A location-based clustering algorithm for data gathering in 3D underwater Wireless Sensor Networks," *International Symposium on Telecommunications, IST 2008.*

72. Ayaz, M., A. Abdullah, and J. Low Tang, "Temporary cluster based routing for Underwater Wireless Sensor Networks," *International Symposium in Information Technology (ITSim),* 2010.

73. Tonghong, L., "Multi-sink opportunistic routing protocol for underwater mesh network," *International Conference on Communications, Circuits and Systems 2008,* ICCCAS 2008.

74. Stojanovic, M., J.A. Catipovic, and J.G. Proakis, "Phase-coherent digital communications for underwater acoustic channels," *IEEE Journal of Oceanic Engineering,* 1994. 19(1): p. 100-111.

75. Heinzelman, W.B., A.P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications,* 2002. 1(4): p. 660-670.

76. Younis, O. and S. Fahmy, "HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Transactions on Mobile Computing,* 2004. 3(4): p. 366-379.

77. Pu, W., L. Cheng, and Z. Jun, "Distributed Minimum-Cost Clustering Protocol for UnderWater Sensor Networks (UWSNs)," *IEEE International Conference on Communications, '07,* ICC. 2007.

78. Uichin, L., et al., "Pressure Routing for Underwater Sensor Networks," *INFOCOM, 2010, IEEE Proceedings*.

79. Wei, L., et al., "Information-Carrying Based Routing Protocol for Underwater Acoustic Sensor Network," *International Conference on.Mechatronics and Automation '07,* ICMA 2007.

80. Magistretti, E., et al., "A Mobile Delay-Tolerant Approach to Long-Term Energy-Efficient Underwater Sensor Networking," *IEEE Wireless Communications and Networking Conference,* 2007.

81. Zheng, G., et al., "Adaptive Routing in Underwater Delay/Disruption Tolerant Sensor Networks," *Fifth Annual Conference on Wireless on Demand Network Systems and Services '08.* WONS 2008.

82. Akyildiz, I.F., et al., "Wireless sensor networks: a survey," *Computer Networks,* 2002. 38(4): p. 393-422.

83. Xu, N., "A Survey of Sensor Network Applications," *IEEE Communications Magazine,* 2002. 40.

84. Underwater Sensor Network Lab, http://uwsn.engr.uconn.edu/index.html.

85. Peng, Z., et al., "An underwater network testbed: design, implementation and measurement," *Proceedings of the second workshop on Underwater networks*. 2007, ACM: Montreal, Quebec, Canada.

86. Shang, G.-y., Z.-p. Feng, and L. Lian, "A low-cost testbed of underwater mobile sensing network," *Journal of Shanghai Jiaotong University (Science),* 2010: p. 1-6.

87. Kaiser, C.L.a.J., "A Survey of Mobile Ad Hoc Network Routing Protocols," *Ulm University, Report Series,* 2005. Nr. 2003-08.

88. NS-2. http://www.isi.edu/nsnam/ns/.

89. Harris, A.R. and M. Zorzi, "On the Design of Energy-efficient Routing Protocols in Underwater Networks," *Sensor, Mesh and Ad Hoc Communications and Networks, 2007.* SECON '07. 4th Annual IEEE Communications Society Conference on. 2007.

90. Chun-Hao, Y. and S. Kuo-Feng, "An energy-efficient routing protocol in underwater sensor networks," *Sensing Technology, 2008.* ICST 2008. 3rd International Conference on. 2008.

91. Baldo, N., et al., "ns2-MIRACLE: a modular framework for multi-technology and cross-layer support in network simulator 2," *Proceedings of the 2nd international conference on Performance evaluation methodologies and tools*. 2007, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering): Nantes, France.

92. Underwater Channel, P.a.M.f.N.-M. http://www.dei.unipd.it/wdyn/?IDsezione=5216.

93. Jan Bauer, N.A., Raphael Ernst, "Optimizing ODMRP for Underwater Networks," *Military Communications Conference*, October 2010: San Jose, CA.

94. Zorzi, M., et al., "Energy-Efficient Routing Schemes for Underwater Acoustic Networks," *IEEE Journal on Selected Areas in Communications,* 2008. 26(9): p. 1754-1766.

95. Peng, X., et al., "Aqua-Sim: An NS-2 based simulator for underwater sensor networks," *OCEANS 2009, MTS/IEEE Biloxi - Marine Technology for Our Future:* Global and Local Challenges. 2009.

96. Tiansi, H. and F. Yunsi, "QELAR: A Machine-Learning-Based Adaptive Routing Protocol for Energy-Efficient and Lifetime-Extended Underwater Sensor Networks," *IEEE Transactions on Mobile Computing,* 9(6): p.796-809.

97. Opnet. http://www.opnet.com/solutions/network_rd/modeler_wireless.html.

98. Sözer M. , S., J. G. Proakis, "Initialization and routing optimization for ad-hoc underwater acoustic networks," *In Proceedings of Opnetwork.* 2000.

99. Xianhui, C., et al., "A Static Multi-hop Underwater Wireless Sensor Network Using RF Electromagnetic Communications," *29th IEEE International Conference on Distributed Computing Systems Workshops,* 2009.

100. The Qualnet Simulator, http://www.scalable-networks.com/.

101. AUVNetsim, http://users.ece.gatech.edu/jmjm3/publications/auvnetsim.pdf.

102. Liu, G. and Z. Li, "Depth-Based Multi-hop Routing protocol for Underwater Sensor Network," *2nd International Conference on Industrial Mechatronics and Automation (ICIMA), 2010*.

103. Choi, S.K., S.A. Menor, and J. Yuh., "Distributed virtual environment collaborative simulator for underwater robots," *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2000.* (IROS 2000). 2000.

104. Carlson, E.A., P.P. Beaujean, and E. An. "Simulating communication during multiple AUV operations," *Autonomous Underwater Vehicles, 2004* IEEE/OES. 2004.

105. Lee, S.-J., J.-I. Namgung, and S.-H. Park, "Efficient UDD Architecture for Underwater Wireless Acoustic Sensor Network," *Proceedings of the 2009 International Conference on Computational Science and Engineering,* IEEE Computer Society*, Volume 02*. 2009.

106. Glenn Elert, H.L. Depth of the Ocean, http://hypertextbook.com/facts/2006/HelenLi.shtml. 2006.

107. Xin, W., et al., "A Distributed Power Control Based MAC Protocol for Underwater Acoustic Sensor Networks," *4th IEEE International Conference on Circuits and Systems for Communications '08, ICCSC 2008.*

108. Jornet, J.M. and M. Stojanovic, "Distributed power control for underwater acoustic networks," *OCEANS* 2008.

109. Wills, J., W. Ye, and J. Heidemann, "Low-power acoustic modem for dense underwater sensor networks," *Proceedings of the 1st ACM international workshop on Underwater networks*. 2006, ACM: Los Angeles, CA, USA.

110. Pereira P. R., A.G., "End-To-End Reliability In Wireless Sensor Networks: Survey And Research Challenges," *Euro FGI workshop on IP QoS and Traffic Control*. 2007: Lisbon, Portugal.

111. Melodia, T., D. Pompili, and I.F. Akyldiz, "Handling Mobility in Wireless Sensor and Actor Networks," *IEEE Transactions on Mobile Computing*, 2010. 9(2): p. 160-173.

112. Creixell, W. and K. Sezaki, "Routing protocol for ad hoc mobile networks using mobility prediction," *Int. J. Ad Hoc Ubiquitous Compuing*, 2007. 2(3): p. 149-156.

113. Zhong, Z., C. Jun-Hong, and A. Bagtzoglou, "Scalable Localization with Mobility Prediction for Underwater Sensor Networks," *INFOCOM 2008. The 27th Conference on Computer Communications.* IEEE. 2008.

114. Caruso, A., et al., "The Meandering Current Mobility Model and its Impact on Underwater Mobile Sensor Networks," *INFOCOM 2008. The 27th Conference on Computer Communications.* IEEE. 2008.

115. Pompili, D. and I.F. Akyildiz, "A multimedia cross-layer protocol for underwater acoustic sensor networks," Transactions on Wireless Communications, 2010. 9(9): p. 2924-2933.

116. Pompili, D., "Efficient Communication protocols for Underwater Acoustic sensor networks," *School of Electrical and Computer Engineering*, 2007, Georgia Institute of Technology.

117. Stojanovic, M., "Optimization of a data link protocol for an underwater acoustic channel." *Oceans 2005* - Europe. 2005.

118. Basagni, S., et al., "Choosing the packet size in multi-hop underwater networks," *OCEANS 2010*, IEEE - Sydney.

119. Basagni, S., et al., "Optimizing network performance through packet fragmentation in multi-hop underwater communications," *OCEANS 2010* IEEE - Sydney.

120. Vuran, M.C. and I.F. Akyildiz, "Cross-Layer Packet Size Optimization for Wireless Terrestrial, Underwater, and Underground Sensor Networks," *INFOCOM 2008. The 27th Conference on Computer Communications.* IEEE. 2008.

121. Haykin, S., "Communication Systems" 3rd ed. 1994: *Wiley Publishers*.

122. Wang, D., et al., "Acoustically focused adaptive sampling and on-board routing for marine rapid environmental assessment," *Journal of Marine Systems*, 2009. 78(Supplement 1): p. S393-S407.

123. Collar, P.G., "The AUTOSUB project-autonomous underwater vehicles for data collection in the deep ocean," *Monitoring the Sea*, IEE Colloquium on. 1990.

124. Sousa, J., et al., "Multiple AUVS for coastal oceanography," *OCEANS '97.* MTS/IEEE Conference Proceedings. 1997.

125. Wood, S., et al., "The Development of an Autonomous Underwater Powered Glider for Deep-Sea Biological, Chemical and Physical Oceanography," *OCEANS 2007* - Europe. 2007.

126. Tomczak, M. University of South Australia in Adelaide, http://www.es.flinders.edu.au/˜mattom/IntroOc/newstart.html. Read December 13th 2010.

127. Richard Owen, C.M., Cheryl Woodley, Hank Trapido-Rosenthal, Tamara Galloway, Michael Depledge, James Readman, Lucy Buxton, Samia Sarkis, Ross Jones, and Anthony Knap, *"A common sense approach for confronting coral reef decline associated with human activities," Marine Pollution Bulletin*, 2005(51(5-7)): p. 481 - 485.

128. The Argo Programme, http://www.argo.ucsd.edu/ Read July, 2011.

129. Khan, A. and L. Jenkins, "Undersea wireless sensor network for ocean pollution prevention," *3rd International Conference on Communication Systems Software and Middleware and Workshops, 2008.* COMSWARE 2008.

130. Wallace J., D.E.I.a.W., "Oceanography : an introduction," *Wadsworth, San Diego*. 1995.

131. John Mulholland, D.M., "Wireless communication enhances subsea production monitoring," *Offshore*, 2011.

132. Mainwaring, A., et al., "Wireless sensor networks for habitat monitoring," *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. 2002, ACM: Atlanta, Georgia, USA.

133. Cerpa, A., et al., "Habitat monitoring: application driver for wireless communications technology," *Workshop on Data communication in Latin America and the Caribbean*. 2001, ACM: San Jose, Costa Rica.

134. Stolkin, R., et al., "Passive acoustic detection of modulated underwater sounds from biological and anthropogenic sources," *OCEANS 2007*.

135. Rodney Rountree, F.J., and Cliff Goudey, "Listening to fish: Applications of passive acoustics to fisheries," *The Journal of the Acoustical Society of America*, 2006. 119(5): p. 3277-3277.

136. Caiti., A., "Underwater acoustic networks scenario description," *Preliminary draft from ISME to UAN partners*, 2009.

137. Sonardyne International Ltd, http://www.sonardyne.com/Industry/Defence/Maritime˙Security/index.html. 2011.

138. Bovio, E., "Autonomous underwater vehicles for port protection," *NATO Research and Technology Organisation,*. 2006.

139. Asada, A., et al., "Advanced Surveillance Technology for Underwater Security Sonar Systems," *OCEANS 2007* - Europe. 2007.

140. Stolkin, R. and I. Florescu, "Probability of Detection and Optimal Sensor Placement for Threshold Based Detection System," *Sensors Journal, IEEE*, 2009. 9(1): p. 57-60.

141. Inwhee, J., "Optimal packet length with energy efficiency for wireless sensor networks," *IEEE International Symposium on Circuits and Systems*, 2005.

142. Modiano E., "Data Link Protocols for LDR MILSTAR Communications," *Communications Division Internal Memorandum*, Lincoln Laboratory, October 1994.

143. Schwartz, M., "Telecommunication Networks: Protocols; Modeling and Analysis," *Addison-Wesley*: New York, 1987, p. 125-156.

144. Schwartz, M., "*Telecommunication Networks*," *Adison Wesley*, 1988,

145. Sankarasubramaniam, Y., I.F. Akyildiz, and S.W. McLaughlin, "Energy efficiency based packet size optimization in wireless sensor networks," *Sensor Network Protocols and Applications, 2003.* Proceedings of the First IEEE. 2003 IEEE International Workshop on. 2003.

146. Peng, X. and C. Jun-Hong, "R-MAC: An Energy-Efficient MAC Protocol for Underwater Sensor Networks," *International Conference on Wireless Algorithms, Systems and Applications, '07, WASA 2007.*

LIST OF PUBLICATIONS

1. **Muhammad Ayaz**, Imran Baig, Azween Abdullah, Ibrahima Faye, "*A survey on routing techniques in underwater wireless sensor networks*" **Elsevier's** Journal of Networks and Computer Applications (IF - 1.1), 2011, (Published)

2. **Muhammad Ayaz**, Azween Abdullah, Ibrahima Faye, Yasir Batira, "*An Efficient Dynamic Addressing Based Routing Protocol for Underwater Wireless Sensor Networks*" **Elsevier's** Computer Communications (IF-0.85), 2011. (Accepted with minor revision)

3. **Muhammad Ayaz**, Azween Abdullah, Low Tang Jung, "*Reliable Data Deliveries Using Packet Optimization in Multi-hop Underwater Sensor Networks*", Journal of Physical Sciences, ISI Indexed (IF- 0.54), Submitted and decision is expected in August 2011.

4. **Muhammad Ayaz**, Azween Abdullah. "*A Taxonomy of Routing Techniques in Underwater Wireless Sensor Networks*", Book chapter will be published by *IGI Global Publication*, USA. (Accepted, In press)

5. **Muhammad Ayaz**, Azween Abdullah and Ibrahima Faye, "*Hop-by-Hop Reliable Data Deliveries for Underwater Wireless Sensor Networks"*. In: Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010, Fukuoka, Japan. Published. (Published).

6. **Muhammad Ayaz** and Azween Abdullah, "*Temporary cluster based routing for Underwater Wireless Sensor Networks"* In: Information Technology (ITSim), 2010, Kuala Lumpur Malaysia. (Published).

7. **Muhammad Ayaz** and Azween Abdullah, "*Hop-by-Hop Dynamic Addressing Based (H2-DAB) Routing Protocol for Underwater Wireless Sensor Networks"* In: International Conference on Information and Multimedia Technology, 2009, Jeju Island, Korea. (Published).

8. **Muhammad Ayaz** and Azween Abdullah, *"Underwater Wireless Sensor Networks: Routing Issues and Future Challenges"*, In: 7th int'l conference on advances in mobile computing & multimedia (MoMM 2009), KL, Malaysia. (Published).

PATENT STATUS

Application Number	PI2010001424
Invention	Hop-by-Hop Dynamic Addressing Based Routing Protocol for Underwater Wireless Sensor Networks
Filing Date	31 Mar 2010
Current Status	Draft sent on 8 February 2011

APPENDEX A

SIMULATION CODING FOR H2-DAB ROUTING ALGORITHM

basic_simulation_parameters

```
#==================================
#     Simulation parameters setup
#==================================
set opt(chan)           Channel/UnderwaterChannel
set opt(prop)           Propagation/UnderwaterPropagation
set opt(netif)          Phy/UnderwaterPhy

set opt(mac)            Mac/UnderwaterMac/RMac
set opt(ifq)            Queue/DropTail

set opt(txpower)      1
set opt(rxpower)    0.1
set opt(ant)          Antenna/OmniAntenna

#==================================
#Specify the parameters for the R-MAC protocol
#==================================

Mac/UnderwaterMac set bit_rate_ 1.0e4
Mac/UnderwaterMac/RMac set ND_window_ 2
Mac/UnderwaterMac/RMac set ACKND_window_ 4
Mac/UnderwaterMac/RMac set PhaseOne_window_ 7
Mac/UnderwaterMac/RMac set PhaseTwo_window_ 2
Mac/UnderwaterMac/RMac set IntervalPhase2Phase3_ 2

#==================================
#Mac/UnderwaterMac/RMac set ACKRevInterval_ 0.1
#==================================
Mac/UnderwaterMac/RMac set duration_ 0.1
Mac/UnderwaterMac/RMac set PhyOverhead_ 8
Mac/UnderwaterMac/RMac set large_packet_size_ 560 ;# 70 bytes
Mac/UnderwaterMac/RMac set short_packet_size_ 80 ;# 10 bytes
Mac/UnderwaterMac/RMac set PhaseOne_cycle_ 1;#deleted later
Mac/UnderwaterMac/RMac set PeriodInterval_ 1
Mac/UnderwaterMac/RMac set transmission_time_error_ 0.0;

set chan_1_ [new $opt(chan)]
```

```
#===================================
#     Mobile node parameter setup
#===================================
$ns_ node-config -adhocRouting $opt(adhocRouting) \
-llType $opt(ll) \
-macType $opt(mac) \
-ifqType $opt(ifq) \
-ifqLen $opt(ifqlen) \
-antType $opt(ant) \
-propType $opt(prop) \
-phyType $opt(netif) \
-agentTrace OFF \
-routerTrace OFF \
-macTrace OFF\
-topoInstance $topo\
-energyModel $opt(energy)\
-txPower $opt(txpower)\
-rxPower $opt(rxpower)\
-initialEnergy $opt(initialenergy)\
-idlePower $opt(idlepower)\
-channel $chan_1_

$node_(0) set sinkStatus_ 1
$node_(0) set passive 1

#===================================
#        Nodes Definition
#===================================
#Create 350 nodes
set n0 [$ns node]
$n0 set X_ 901
$n0 set Y_ 397
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 1088
$n1 set Y_ 421
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 1309
$n2 set Y_ 431
$n2 set Z_ 0.0

//...
//...
//...

$ns initial_node_pos $n349 20
set n349 [$ns node]
$n349 set X_ 3812
$n349 set Y_ 1714
$n349 set Z_ 0.0
$ns initial_node_pos $n349 20
```

```
#==================================
#        Agents Definition
#==================================
#Setup a UDP connection
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

set udp1 [new Agent/UDP]
$ns attach-agent $n2 $udp1

set udp2 [new Agent/UDP]
$ns attach-agent $n42 $udp2
set null1 [new Agent/Null]
$ns attach-agent $n57 $null1
$ns connect $udp2 $null1
$udp0 set packetSize_ 1500

set udp3 [new Agent/UDP]
$ns attach-agent $n64 $udp3
set null1 [new Agent/Null]
$ns attach-agent $n79 $null1
$ns connect $udp3 $null1
$udp0 set packetSize_ 1500

set null1 [new Agent/Null]
$ns attach-agent $n1 $null1
$ns connect $udp0 $null1
$ns connect $udp1 $null1
$udp0 set packetSize_ 1500
$udp1 set packetSize_ 1500

#==================================
#        Applications Definition
#==================================
#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 1000
$cbr0 set rate_ 1.0Mb
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 10.0 "$cbr0 stop"

#Setup a CBR Application over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 1000
$cbr1 set rate_ 1.0Mb
$cbr1 set random_ null
$ns at 1.0 "$cbr1 start"
$ns at 10.0 "$cbr1 stop"
//...
//...
//...
```

```
#Setup a CBR Application over UDP connection
set cbr349 [new Application/Traffic/CBR]
$cbr349 attach-agent $udp349
$cbr349 set packetSize_ 1000
$cbr349 set rate_ 1.0Mb
$cbr349 set random_ null
$ns at 1.0 "$cbr349 start"
$ns at 10.0 "$cbr349 stop"

set a_(0) [new Agent/UWSink]
$ns_ attach-agent $node_(0) $a_(0)

$node_(0) set_next_hop 0

#==================================
#       Initialization
#==================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

#Open the NS trace file
set tracefile [open H2-DAB.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open H2-DAB.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];#Create wireless channel

#==================================
#Define a 'finish' procedure
#==================================
proc finish {} {
   global ns tracefile namfile
   $ns flush-trace
   close $tracefile
   close $namfile
   exec nam out.nam &
   exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
   $ns at $val(stop) "\$n$i reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
      $ns run
```

packet_cache_h

```
#define_PKT_CACHE_H_
#include <packet.h>

typedef int  PacketID;

class PktCache
{
public:
     PktCache();
     ~PktCache();

     int& size(void)
     { return size_; }

     int accessPacket(PacketID p);
     void addPacket(PacketID p);
     void deletePacket(PacketID p);
     void dump(void);

private:
     PacketID *pcache_;                    // packet cache
     int        size_;                              // cache size
     int max_size_;                        // max cache size
}
```

packet_queue_h

```
#include <deque>

class PacketQueue {
public:
     PacketQueue();
     ~PacketQueue();

     bool empty() { return dq_.empty(); }
     int size() { return dq_.size(); }

     void pop() { dq_.pop_front(); );
     Packet* front() { return dq_.front(); };
     void push(Packet* p);

private:
     deque<Packet*> dq_;
}
```

packet_cache_cc

```
#include <stdio.h>
#include "pkt_cache.h"

PktCache::
PktCache()
{
	int i;

	max_size_ = 1500;
	size_ = 0;

	pcache_ = new PacketID [1500];

	//for (i = 0; i < 100; i++)
	//	pcache_[i] = new Packet;
}

PktCache::
~PktCache()
{
	int i;

	//for (i = 0; i < max_size_; i++)
	//	delete pcache_[i];

	delete[ ] pcache_;
}

void
PktCache::
addPacket(PacketID p)
{
	if (size_ == max_size_) {
		printf("Cache is full!\n");
		return;
	}

	pcache_[size_] = p;
	size_++;

	return;
}

void
PktCache::
deletePacket(PacketID p)
{
}

void
PktCache::
```

```
dump(void)
{
	int i;

	for (i = 0; i < size_; i++)
		fprintf(stderr, "[%d]: %d\n", i, pcache_[i]);
}

if 0
// test main function
int main(void)
{
	PktCache pc;
	Packet p1, p2, p3;

	pc.addPacket(&p1);
	pc.addPacket(&p2);
	pc.addPacket(&p3);

	pc.dump();

	pc.accessPacket(&p1);
	pc.dump();

	return 0;
}
endif
```

h2dab_h

```
/**
 *  Hop-by-Hop Dynamic Addrssing Based routing alogrithm header file.
 */

#ifndef _H2DAB_H_
#define _H2DAB_H_

#include <deque>

#include <stdarg.h>

#include <object.h>
#include <agent.h>
#include <trace.h>
#include <packet.h>
#include <mac.h>
#include <mobilenode.h>
#include <classifier-port.h>

#include "pkt_cache.h"

#define H2DAB_PORT

#define H2DAB_BEACON_INT 10
#define JITTER 1

class H2DAB_Agent;
class MyPacketQueue;
class NeighbEnt;
class NeighbTable;
class hdr_h2dab;

#if	0
struct H2DABPacket {
	int dest;
	int src;
	Packet *pkt;	// inner NS packet

	H2DABPacket() : pkt(NULL) {}
	H2DABPacket(Packet *p, hdr_h2dab *h2dabh) :
			pkt(p) {}
};
#endif

class H2DAB_AgentTimer : public TimerHandler {
public:
	H2DAB_AgentTimer(H2DAB_Agent *a) { a_ = a; }
	virtual void expire(Event *e) = 0;

protected:
```

```
        H2DAB_Agent *a_;
};
    class H2DAB_BeaconHandler : public Handler {
    public:
        H2DAB_BeaconHandler(H2DAB_Agent *a) { a_ = a; }
        virtual void handle(Event *e);

    private:
        H2DAB_Agent *a_;
    };

    class H2DAB_BeaconTimer : public H2DAB_AgentTimer {
    public:
        H2DAB_BeaconTimer(H2DAB_Agent *a) : H2DAB_AgentTimer(a) {}
        virtual void expire(Event *e);
    };

    class H2DAB_SendingTimer : public H2DAB_AgentTimer {
    public:
        H2DAB_SendingTimer(H2DAB_Agent *a) : H2DAB_AgentTimer(a) {}
        virtual void expire(Event *e);
    };

    class QueueItem {
    public:
        QueueItem() : p_(0), send_time_(0) {}
        QueueItem(Packet *p, double t) : p_(p), send_time_(t) {}

        Packet *p_;             // pointer to the packet
        double send_time_;      // time to send the packet
    };

    class MyPacketQueue {
    public:
        MyPacketQueue() : dq_() {}
        ~MyPacketQueue() { dq_.clear(); }

        bool empty() { return dq_.empty(); }
        int size() { return dq_.size(); }
        void dump();

        void pop() { dq_.pop_front(); };
        QueueItem* front() { return dq_.front(); };
        void insert(QueueItem* q);
        bool update(Packet *p, double t);
        bool purge(Packet *p);

    private:
        deque<QueueItem*> dq_;
    };

    class NeighbEnt {
    public:
        NeighbEnt(H2DAB_Agent* ina) :
```

```
            x(0.0), y(0.0), z(0.0), routeFlag(0) {}
        // the agent is used for timer object

        double x, y, z;             // location of neighbor, actually we only need hop info
        nsaddr_t hp_id;    // IP of neighbor
        int routeFlag;              // indicates that a routing path exists

        // user timer
        //H2DAB_DeadNeighbTimer dnt;          // timer for expiration of neighbor

    };

    class NeighbTable {
    public:
        NeighbTable(H2DAB_Agent *a);
        ~NeighbTable();

        void dump(void);
        void ent_delete(const NeighbEnt *e);                // delete an neighbor
        NeighbEnt *ent_add(const NeighbEnt *e);         // add an neighbor
        NeighbEnt  *ent_find_shadowest(MobileNode  *mn);    // find the neighbor with
minimal hop count
        void updateRouteFlag(nsaddr_t, int);

    private:
        int num_ents;       // number of entries in use
        int max_ents;       // capacity of the table
        H2DAB_Agent *a_;       // agent owns the table
        NeighbEnt **tab;    // neighbor table
    };

    #define      H2DABH_DATA_GREEDY    0
    #define      H2DABH_DATA_RECOVER  1
    #define      H2DABH_BEACON              2

    class hdr_h2dab {
    public:
        static int offset_;                     // offset of this header
        static int& offset() { return offset_; }
        static hdr_h2dab* access(const Packet *p) {
                return (hdr_h2dab*)p->access(offset_);
        }

        double x, y, z;

        int& packetID() { return packetID_; }

        int& valid() { return valid_; }
        int& mode()  { return mode_; }
        int& nhops() { return nhops_; }
        nsaddr_t& prev_hop() { return prev_hop_; }
        nsaddr_t& owner() { return owner_; }

        double& hop() { return hop_; }
```

```
    // get the header size
    int size()
    {       int sz;
            sz = 4 * sizeof(int);
            sz += 3 * sizeof(double);
            return sz;
    }

private:
    int packetID_;          // unique id for packet

    int valid_;             // is this header in the packet?
    int mode_;              // routing mode: greedy | recovery
    int nhops_;             // max # of hops to broadcast
                            // in recovery mode
    nsaddr_t prev_hop_;     // the sender
    nsaddr_t owner_;        // from whom the sender got it

    double hop_;            // the hop count of last hop
};

class H2DAB_Agent : public Tap, public Agent {
    friend class H2DAB_BeaconHandler;
    friend class H2DAB_BeaconTimer;
    friend class H2DAB_SendingTimer;

public:
    H2DAB_Agent();
    ~H2DAB_Agent();

    virtual int command(int argc, const char*const* argv);
    virtual void recv(Packet *, Handler *);
    virtual void recv2(Packet *, Handler *);

    virtual void tap(const Packet *p);

    void deadneighb_callback(NeighbEnt *ne);
    void beacon_callback(void);
    void send_callback(void);

    //H2DAB_BeaconHandler bhdl;

#endif      /* _H2DAB_H_ */
```

h2dab_cc

```
#include <stdlib.h>
#include <stdio.h>
#include <object.h>
#include <agent.h>
#include <trace.h>
#include <hop_count.h>
#include <scheduler.h>
#include <random.h>

#include <mac.h>
#include <ll.h>
#include <cmu-trace.h>

#include "h2dab.h"

#define      BEACON_RESCHED

#define DEBUG_AGENT
//#define DEBUG_BEACON
#define DEBUG_PRINT
//#define DEBUG_BROADCASTING

// parameters to control the delay
#define H2DAB_MAX_DELAY     0.3      // maximal propagation delay for one hop
#define H2DAB_MAX_RANGE     150      // maximal transmmition range
#define H2DAB_MIN_BACKOFF  0.0       // minimal backoff time for the hello packet

#define H2DAB_USE_ROUTEFLAG
#define H2DAB_MAX_HOPS       10
#define H2DAB_HOP_COUNT 9

int hdr_h2dab::offset_;



static class H2DABAgentClass : public TclClass {
public:
     H2DABAgentClass() : TclClass("Agent/H2DAB") {}
     TclObject* create(int, const char* const*) {
               return (new H2DAB_Agent);
     }
} class_H2DABAgent;

class H2DABHeaderClass : public PacketHeaderClass {
public:
     H2DABHeaderClass() : PacketHeaderClass("PacketHeader/H2DAB",
                                            sizeof(hdr_h2dab)) {
               bind_offset(&hdr_h2dab::offset_);
     }
} class_h2dab_hdr;
```

```
void H2DAB_BeaconHandler::handle(Event *e)
{
    a_->forwardPacket((Packet*)e, 0);
}

void H2DAB_BeaconTimer::expire(Event *e)
{
    a_->beacon_callback();
}

void H2DAB_SendingTimer::expire(Event *e)
{
    a_->send_callback();
}

/*
void H2DAB_DeadNeighbTimer::expire(Event *e)
{
    a->deadneighb_callback(ne);
}
*/


void PacketQueue {
public:
    PacketQueue();
    ~PacketQueue();

    bool empty() { return dq_.empty(); }
    int size() { return dq_.size(); }

    void pop() { dq_.pop_front(); );
    Packet* front() { return dq_.front(); };
    void push(Packet* p);
}

void MyPacketQueue::insert(QueueItem *q)
{
    QueueItem *tmp;
    deque<QueueItem*>::iterator iter;

    // find the insert point
    iter = dq_.begin();
    while (iter != dq_.end())
    {
            tmp = *iter;
            if (tmp->send_time_ > q->send_time_)
            {
                    dq_.insert(iter, q);
                    return;
            }
            iter++;
    }
```

```
    // insert at the end of the queue
    dq_.push_back(q);
}
// construct the hello packet
Packet*
H2DAB_Agent::makeBeacon(void)
{
    Packet *p = allocpkt();
    assert(p != NULL);

    hdr_ip *iph = hdr_ip::access(p);
    hdr_cmn *cmh = hdr_cmn::access(p);
    hdr_h2dab *h2dabh = hdr_h2dab::access(p);

    // setup header
    cmh->next_hop_ = IP_BROADCAST;
    cmh->addr_type_ = AF_INET;
    cmh->ptype_ = PT_H2DAB;
    cmh->direction() = hdr_cmn::DOWN;
    cmh->size() = h2dabh->size() + IP_HDR_LEN;

    //iph->daddr() = IP_BROADCAST << Address::instance().nodeshift();
    iph->saddr() = mn_->address();
    iph->daddr() = IP_BROADCAST;
    iph->dport() = H2DAB_PORT;

    // fill in the location info
    mn_->getLoc(&(h2dabh->x), &(h2dabh->y), &(h2dabh->z));
    h2dabh->mode() = H2DABH_BEACON;
    h2dabh->nhops() = 1;

    return p;
}

// send beacon pkt only once at the beginning
void H2DAB_Agent::sendBeacon(void)
{
    Packet *p = makeBeacon();
    hdr_cmn *cmh = hdr_cmn::access(p);

    if (p)
    {
            assert(!HDR_CMN(p)->xmit_failure_);
            if (cmh->direction() == hdr_cmn::UP)
                    cmh->direction() = hdr_cmn::DOWN;
            Scheduler::instance().schedule(ll, p, 0);
    }
    else
    {
            fprintf(stderr, "ERROR: Can't make new beacon!\n");
            abort();
    }
}
```

```
// If such a item is found, return true, otherwise
// return false.
bool
MyPacketQueue::purge(Packet *p)
{
    Packet *pkt;
    int curID;
    deque<QueueItem*>::iterator iter;
    hdr_h2dab *h2dabh;

    // get current hello packet ID
    h2dabh = hdr_h2dab::access(p);
    curID = h2dabh->hello_packetID();

    // search the queue
    iter = dq_.begin();
    while (iter != dq_.end())
    {
            h2dabh = hdr_h2dab::access((*iter)->p_);
            if (h2dabh->hello_packetID() == curID)
            {
                    dq_.erase(iter);
                    return TRUE;
            }
            iter++;
    }

    return FALSE;
}

// Dump all the items in queue for debug
void MyPacketQueue::dump()
{
    deque<QueueItem*>::iterator iter;
    hdr_h2dab *h2dabh;
    int i = 0;

    iter = dq_.begin();
    while (iter != dq_.end())
    {
            h2dabh = hdr_h2dab::access((*iter)->p_);
            fprintf(stderr, "Queue[%d]: hello_packetID %d, send time %f\n",
                            i, h2dabh->hello_packetID(), (*iter)->send_time_);
            iter++;
            i++;
    }
}

NeighbTable::NeighbTable(H2DAB_Agent *a)
{
  int i;

  num_ents = 0;
  max_ents = 100;
```

```
    // create the default table with size 100
    tab = new NeighbEnt* [100];
    a_ = a;

    for (i = 0; i < 100; i++)
        tab[i] = new NeighbEnt(a);
}

NeighbTable::~NeighbTable()
{
    int i;

    for (i = 0; i < max_ents; i++)
        delete tab[i];

    delete[] tab;
}

static int neighbEntCmp(const void *a, const void *b)
{
    nsaddr_t ia = ((const NeighbEnt*)a)->hp_id;
    nsaddr_t ib = ((const NeighbEnt*)b)->hp_id;

    if (ia > ib) return 1;
    if (ia < ib) return -1;
    return 0;
}

void NeighbTable::dump(void)
{
    int i;

    for (i = 0; i < num_ents; i++)
        fprintf(stderr, "tab[%d]: %d x = %f, y = %f, z = %f\n",
                i, tab[i]->hp_id,
                tab[i]->x, tab[i]->y, tab[i]->z);
}

void
NeighbTable::ent_delete(const NeighbEnt *ne)
{
    int l, r, m;
    int i;
    NeighbEnt *owslot;

    // binary search
    l = 0; r = num_ents - 1;
    while (l <= r)
    {
        m = l + (r - l)/2;
        if (tab[m]->hp_id < ne->hp_id)
            l = m + 1;
        else if (tab[m]->hp_id > ne->hp_id)
            r = m - 1;
```

```
                else
                        // m is the entry to be deleted
                        break;
        }

        if (l > r)
                // no found!
                return;

        owslot = tab[m];

        // slide the entries
        i = m + 1;
        while (i < num_ents)
                tab[i - 1] = tab[i++];

        tab[num_ents-1] = owslot;
        num_ents--;
    }


    if (sHopID_r==0  &&  IDp != IDr)
         {
            cHopID_q=cHopID_p;
            cHopID_p=sHopID_r+1;
            IDp=IDr;                    //Hello Packet recieved directly from Sink

        //Updates occur, in primary hop
         }

         else if (sHopID_r != 0 && sHopID_r<cHopID_p && (p<sHopID_s ||
cHopID_p==sHopID_s))
            {
              cHopID_q=cHopID_p;
              cHopID_p=sHopID_r+1;
              IDp=IDr;              //Hello Packet recieved from an ordinary node

        //Updates occur, in primary hop
            }

            else if (sHopID_r != 0 && sHopID_r<cHopID_p && sHopID_s<cHopID_p)
               {
                 cHopID_p=sHopID_r+1;
                 cHopID_q=sHopID_s+1;
                 IDp=IDr;               //Hello Packet recieved from an ordinary node

       //Updates occur, in primary hop
                }

               else if (sHopID_r != 0 && (r==cHopID_p || sHopID_r>cHopID_p) &&
sHopID_s<cHopID_q)
                  {
                    sHopID_r=sHopID_r;
                    cHopID_q=sHopID_s+1;
```

```
                IDp=IDr;            //Hello Packet recieved from an ordinary node

        //Updates occur, but in backup hop only
                }

    NeighbTable::ent_add(const NeighbEnt *ne)
    {
        NeighbEnt **pte;
        NeighbEnt *pe;
        int i, j;
        int l, r, m;

        // find if the neighbor is already existing
        for (i = 0; i < num_ents; i++)
                if (tab[i]->hp_id == ne->hp_id)
                {
                        tab[i]->x = ne->x;
                        tab[i]->y = ne->y;
                        tab[i]->z = ne->z;

                        return tab[i];
                }


        // need we increase the size of table
        if (num_ents == max_ents)
        {
                NeighbEnt **tmp = tab;
                max_ents *= 2;                  // double the space
                tab = new NeighbEnt* [max_ents];
                bcopy(tmp, tab, num_ents*sizeof(NeighbEnt *));

                for (i = num_ents; i < max_ents; i++)
                        tab[i] = new NeighbEnt(a_);

                delete[ ] tmp;
        }

        // get the insert point


        if (SHopID_r==0  &&  IDp != IDr)
         {
            cHopID_q=cHopID_p;
            cHopID_p=sHopID_r+1;
            IDp=IDr;
            fprintf("\n\n\t\tHello Packet recieved directly from Sink");
            fprintf("\n\n\tUpdates occur, in primary hop");

         }

         else if (sHopID_r != 0 && sHopID_r<cHopID_p && (p<sHopID_s ||
cHopID_p==sHopID_s))
                {
```

```
                cHopID_q=cHopID_p;
                cHopID_p=sHopID_r+1;
                IDp=IDr;

                fprintf("\n\n\t\tHello Packet recieved from an ordinary node");
                fprintf("\n\n\tUpdates occur, in primary hop");

              }

              else if (sHopID_r != 0 && sHopID_r<cHopID_p && sHopID_s<cHopID_p)
                 {
                   cHopID_p=sHopID_r+1;
                   cHopID_q=sHopID_s+1;
                   IDp=IDr;

                   fprintf("\n\n\t\tHello Packet recieved from an ordinary node");
                   fprintf("\n\n\tUpdates occur, in primary hop");

                 }

                 else if (sHopID_r != 0 && (r==cHopID_p || sHopID_r>cHopID_p) &&
sHopID_s<cHopID_q)
                    {
                     sHopID_r=sHopID_r;
                     cHopID_q=sHopID_s+1;
                     IDp=IDr;
                     fprintf("\n\n\t\tHello Packet recieved from an ordinary node");
                     fprintf("\n\n\tUpdates occur, but in backup hop only");
                    }

                    else
                       {
                        fprintf("\n\n\t\tHello Packet discarded, no change in HopID");
                         HC=1;
                       }

          fprintf("\n\n\tCurrent HopID     = %d %d", p, q);
          fprintf("\n\tCurrent Sink ID   = %d", IDp);

              Hop_Count=Hop_Count-1;

         if (Hop_Count>0)
           {
             fprintf ("\n\n\t\tHello packet going to broadcast with Hop Count Value %d ", HC);
           }
          else
             {
              fprintf ("\n\n\t\tNo further Broadcast!");
             }

          return tab[i];
      }
      endif
```

```
void NeighbTable::updateRouteFlag(nsaddr_t addr, int val)
{
    int i;

    for (i = 0; i < num_ents; i++)
    {
            if (tab[i]->hp_id == addr)
            {
                    tab[i]->routeFlag = val;
                    return;
            }
    }
}


NeighbTable::ent_find_shadowest(MobileNode *mn)
{
    NeighbEnt *ne = 0;
    int i;
    double x, y, z, t;

    mn->getLoc(&x, &y, &z);
    t = z;

    #ifdef DEBUG_AGENT
    fprintf(stderr, "[%d]: %d neighbors.\n", mn->address(), num_ents);
    #endif

    for (i = 0; i < num_ents; i++)
    {
            #ifdef   DEBUG_PRINT
            fprintf(stderr, "%d[%d] x: %f, y: %f, z: %f\n",
                    mn->address(), tab[i]->hp_id, tab[i]->x, tab[i]->y, tab[i]->z);
            #endif




            if (tab[i]->routeFlag == 1)
            {
                    ne = tab[i];
                    return ne;
            }

            if (tab[i]->z > t)
            {
               t = tab[i]->z;
               ne = tab[i];
            }
    }
    return ne;
}

NeighbTable::ent_find_shadowest(MobileNode *mn)
```

```
    {
      NeighbEnt *ne = 0;
      int i;
      double x, y, z, t;

      mn->getLoc(&x, &y, &z);
      t = z;

      for (i = 0; i < num_ents; i++)
      {
        #ifdef   DEBUG_PRINT
        fprintf(stderr, "%d[%d] x: %f, y: %f, z: %f\n",
                mn->address(), tab[i]->hp_id, tab[i]->x, tab[i]->y, tab[i]->z);
        #endif

         if (tab[i]->z > t)
         {
            t = tab[i]->z;
            ne = tab[i];
         }
      }

      return ne;
    }
    #endif        // H2DAB_USE_ROUTEFLAG


    H2DAB_Agent::H2DAB_Agent() : Agent(PT_H2DAB),
        bint_(H2DAB_BEACON_INT),  bdesync_(H2DAB_BEACON_DESYNC),  mn_(0),
pkt_cnt_(0)
    {
        ntab_ = new NeighbTable(this);

        // create packet cache
        pc_ = new PktCache();
    }


    H2DAB_Agent::~H2DAB_Agent()
    {
        delete ntab_;

        // packet cache
        delete pc_;
    }

    // fetch the packet from the sending queue and broadcast.
    void H2DAB_Agent::send_callback(void)
    {
        QueueItem *q;
        hdr_h2dab *h2dabh;

        // we're done if there is no packet in queue
        if (pq_.empty())
```

```
                return;
        // send the first packet out
        q = pq_.front();
        pq_.pop();
        Scheduler::instance().schedule(ll, q->p_, 0);

        // put the packet into cache
        h2dabh = hdr_h2dab::access(q->p_);
        pc_->addPacket(h2dabh->hello_packetID());

        // reschedule the timer if there are
        // other packets in the queue
        if (!pq_.empty())
        {
                q = pq_.front();
                latest_ = q->send_time_;
                send_timer_->resched(latest_ - NOW);
        }
}

void H2DAB_Agent::beacon_callback(void)
{
        return;

        Packet *p = makeBeacon();

        hdr_cmn *cmh = hdr_cmn::access(p);
        hdr_ip *iph = hdr_ip::access(p);

        if (p)
        {
                assert(!HDR_CMN(p)->xmit_failure_);
                if (cmh->direction() == hdr_cmn::UP)
                        cmh->direction() = hdr_cmn::DOWN;

                //fprintf(stderr, "%d is broadcasting beacon pkt src: %d, dst: %d\n",
                //                mn_->address(), iph->saddr(), iph->daddr());

                Scheduler::instance().schedule(ll, p, Random::uniform()*JITTER);
        }
        else
        {
                fprintf(stderr, "ERROR: Can't make new beacon!\n");
                exit(-1);
        }

        //BEACON_RESCHED
}

void H2DAB_Agent::deadneighb_callback(NeighbEnt *ne)
{
        ntab_->ent_delete(ne);
}
```

```
void H2DAB_Agent::forwardPacket(Packet *p, int flag)
{
    hdr_ip *iph = hdr_ip::access(p);
    hdr_cmn *cmh = hdr_cmn::access(p);
    hdr_h2dab *h2dabh = hdr_h2dab::access(p);
    NeighbEnt *ne;

    double delay = 0.0;

    #ifdef   DEBUG_AGENT
    fprintf(stderr, "node %d is forwarding to %d\n",
                                mn_->address(),          Address::instance().get_nodeaddr(iph-
>daddr()));
    #endif

      if (r==0 && IDp != IDr)
     {
        q=p;
        p=r+1;
        IDp=IDr;

     }

     else if (r != 0 && r<p && (p<s || p==s))
        {
          q=p;
          p=r+1;
          IDp=IDr;
        }

        else if (r != 0 && r<p && s<p)
            {
              p=r+1;
              q=s+1;
              IDp=IDr;

            }

            else if (r != 0 && (r==p || r>p) && s<q)
                {
                  r=r;
                  q=s+1;
                  IDp=IDr;
                }

                else
                   {
                     HC=1;
                   }

        HC=HC-1;

    if (HC>0)
      {
```

```
        Hop Count Value %d , HC
      }
    else
       {
        No further Broadcast, discard
       }

#ifdef USE_FLOODING_ALG
void H2DAB_Agent::handlePktForward(Packet *p)
{
    hdr_ip *iph = hdr_ip::access(p);
    hdr_cmn *cmh = hdr_cmn::access(p);
    hdr_h2dab *h2dabh = hdr_h2dab::access(p);

    if (--iph->ttl_ == 0)
    {
            drop(p, DROP_RTR_TTL);
            return;
    }

    // Is this pkt recieved before?
    // each node only broadcasts same pkt once
    if (pc_->accessPacket(h2dabh->hello_packetID()))
    {
            drop(p, DROP_RTR_TTL);
            return;
    }
    else
            pc_->addPacket(h2dabh->hello_packetID());

    // common settings for forwarding
    cmh->num_forwards()++;
    cmh->direction() = hdr_cmn::DOWN;
    cmh->addr_type_ = AF_INET;
    cmh->ptype_ = PT_H2DAB;
    cmh->size() = h2dabh->size() + IP_HDR_LEN;
    cmh->next_hop() = IP_BROADCAST;

    // finally broadcasting it!
    assert(!HDR_CMN(p)->xmit_failure_);
    Scheduler::instance().schedule(ll, p, Random::uniform()*JITTER);
}
#else
// Forward the packet according to its mode
// There are two modes right now: GREEDY and RECOVERY
// The node will broadcast all RECOVERY pakets, but
// it will drop the GREEDY pakets from upper level.
void H2DAB_Agent::handlePktForward(Packet *p)
{
    hdr_ip *iph = hdr_ip::access(p);
    hdr_cmn *cmh = hdr_cmn::access(p);
    hdr_h2dab *h2dabh = hdr_h2dab::access(p);

    double delta;
```

```
	double delay = .0;

	double x, y, z;

	if (--iph->ttl_ == 0)
	{
		drop(p, DROP_RTR_TTL);
		return;
	}

	/*
	// dump the queue
	fprintf(stderr, "-------- Node id: %d --------\n", mn_->address());
	fprintf(stderr, " curID: %d\n", h2dabh->hello_packetID());
	pq_.dump();
	*/

#if 0
	// search sending queue for p
	if (pq_.purge(p))
	{
		drop(p, DROP_RTR_TTL);
		return;
	}
#endif

	// common settings for forwarding
	cmh->num_forwards()++;
	cmh->direction() = hdr_cmn::DOWN;
	cmh->addr_type_ = AF_INET;
	cmh->ptype_ = PT_H2DAB;
	cmh->size() = h2dabh->size() + IP_HDR_LEN;
	cmh->next_hop() = IP_BROADCAST;

	switch (h2dabh->mode())
	{
	case H2DABH_DATA_GREEDY:
		mn_->getLoc(&x, &y, &z);

		// compare the hop_count
		delta = z - h2dabh->hop_count();

		// only forward the packet from lower level
		if (delta < H2DAB_DEPTH_THRESHOLD)
		{
			pq_.purge(p);
			drop(p, DROP_RTR_TTL);
			return;
		}

		#ifdef DEBUG_NONE
		fprintf(stderr, "[%d]: z = %.3f, hop_count = %.3f, delta = %.3f\n",
			mn_->address(), z, h2dabh->hop_count(), delta);
		#endif
```

```
		// update current hop_count
		h2dabh->hop_count() = z;

		// compute the delay
		//delay = H2DAB_DEPTH_THRESHOLD / delta * H2DAB_SCALE;
		delta = 1.0 - delta / H2DAB_MAX_RANGE;
		delay = H2DAB_MIN_BACKOFF + 4.0 * delta * H2DAB_MAX_DELAY;

		// set time out for the packet

		break;
case H2DABH_DATA_RECOVER:
		if (h2dabh->nhops() <= 0)
		{
			#ifdef	DEBUG_PRINT
			fprintf(stderr, "[%d] drops pkt! (nhops < 0)\n", mn_->address());
			#endif

			drop(p, DROP_RTR_TTL);
			return;
		}
		h2dabh->nhops()--;
		break;
default:
		fprintf(stderr, "Unknown data type!]n");
		return;
}

// make up the H2DAB header
h2dabh->owner() = h2dabh->prev_hop();
h2dabh->prev_hop() = mn_->address();

#ifdef DEBUG_NONE
fprintf(stderr, "[%d]: delay %f before broacasting!\n",
			mn_->address(), delay);
#endif

if (pc_ == NULL) {
		fprintf(stderr, "packet cache pointer is null!\n");
		exit(-1);
}

// Is this pkt recieved before?
// each node only broadcasts the same pkt once
if (pc_->accessPacket(h2dabh->hello_packetID()))
{
		drop(p, DROP_RTR_TTL);
		return;
}
//else
//		pc_->addPacket(h2dabh->hello_packetID());
```

```
	// put the packet into sending queue
	double expected_send_time = NOW + delay;
	QueueItem *q = new QueueItem(p, expected_send_time);

	/*
	pq_.insert(q);
	QueueItem *qf;
	qf = pq_.front();
	send_timer_->resched(qf->send_time_ - NOW);
	*/

	if (pq_.empty())
	{
		pq_.insert(q);
		latest_ = expected_send_time;
		send_timer_->resched(delay);
	}
	else
	{
		if (pq_.update(p, expected_send_time))
		{
			pq_.insert(q);

			// update the sending timer
			if (expected_send_time < latest_)
			{
				latest_ = expected_send_time;
				send_timer_->resched(delay);
			}
		}
	}
}
endif		// end of USE_FLOODING_ALG

void H2DAB_Agent::recv(Packet *p, Handler *)
{
	hdr_ip *iph = hdr_ip::access(p);
	hdr_cmn *cmh = hdr_cmn::access(p);
	hdr_h2dab *h2dabh = hdr_h2dab::access(p);
	nsaddr_t src = Address::instance().get_nodeaddr(iph->saddr());
	nsaddr_t dst = Address::instance().get_nodeaddr(iph->daddr());

	#ifdef	DEBUG_PRINT
	//fprintf(stderr, "%d receives pkt from %d to %d\n",
	//		mn_->address(), src, dst);
	#endif

	if (mn_ == NULL)
	{
		fprintf(stderr, "ERROR: Pointer to node is null!\n");
		abort();
	}
```

```
if (h2dabh->mode() == H2DABH_BEACON)
{// hello packet
        // self is not one of the neighbors
        if (src != mn_->address())
                beaconIn(p);
        return;
}
else if ((src == mn_->address()) &&
        (cmh->num_forwards() == 0))
{// packet I'm originating

        #ifdef DEBUG_PRINT
        fprintf(stderr, "%d generates data packet.\n", src);
        #endif

        cmh->size() += IP_HDR_LEN + 8;
        cmh->direction() = hdr_cmn::DOWN;
        iph->ttl_ = 128;
        h2dabh->mode() = H2DABH_DATA_GREEDY;
        h2dabh->hello_packetID() = (int)mn_->address();
}
else if ((src == mn_->address()) &&
        (h2dabh->mode() == H2DABH_DATA_GREEDY))
{// duplicate packet, discard

        #ifdef    DEBUG_PRINT
        fprintf(stderr, "got the pkt I've sent\n");
        #endif

        drop(p, DROP_RTR_ROUTE_LOOP);
        return;
}
else if (dst == mn_->address())
{// packet is for me

        #ifdef DEBUG_BROADCASTING
        fprintf(stderr, "Packet is delivered!\n");
        fflush(stderr);
        #endif

}

else
{// packet I'm forwarding

        if (--iph->ttl_ == 0)
        {
                drop(p, DROP_RTR_TTL);
                return;
        }

        if((h2dabh->mode() == H2DABH_DATA_RECOVER) &&
                (h2dabh->owner() == mn_->address()))
```

```
{
                        //fprintf(stderr, "got the pkt I've sent\n");
                        drop(p, DROP_RTR_ROUTE_LOOP);
                        // it seems this neighbor couldn't find a greedy node
                        //ntab_->updateRouteFlag(h2dabh->prev_hop(), 0);
                        return;
                }
        }

        #ifdef  DEBUG_PRINT
        fprintf(stderr, "owner %d, prev-hop: %d, cur: %d\n",
                h2dabh->owner(), h2dabh->prev_hop(), mn_->address());
        #endif

        // it's time to forward the pkt now
        forwardPacket(p);
}
#endif

void H2DAB_Agent::recv2(Packet *p, Handler *)
{
        hdr_ip *iph = hdr_ip::access(p);
        hdr_cmn *cmh = hdr_cmn::access(p);
        hdr_h2dab *h2dabh = hdr_h2dab::access(p);

        nsaddr_t src = Address::instance().get_nodeaddr(iph->saddr());
        nsaddr_t dst = Address::instance().get_nodeaddr(iph->daddr());

        #ifdef  DEBUG_PRINT
        //fprintf(stderr, "%d receives pkt from %d to %d\n",
        //        mn_->address(), src, dst);
        #endif

        if (mn_ == NULL)
        {
                fprintf(stderr, "ERROR: Pointer to node is null!\n");
                abort();
        }

        if (h2dabh->mode() == H2DABH_BEACON)
        {// hello packet

                // self is not one of the neighbors
                if (src != mn_->address())
                        beaconIn(p);
                return;
        }
        else if ((src == mn_->address()) &&
                (cmh->num_forwards() == 0))
        {// packet I'm originating

                #ifdef DEBUG_PRINT
                fprintf(stderr, "%d generates data packet.\n", src);
                #endif
```

```
cmh->size() += IP_HDR_LEN + 8;
            cmh->direction() = hdr_cmn::DOWN;
            iph->ttl_ = 128;
            h2dabh->mode() = H2DABH_DATA_GREEDY;
            h2dabh->hello_packetID() = (int)mn_->address();
    }
    else if ((src == mn_->address()) &&
            (h2dabh->mode() == H2DABH_DATA_GREEDY))
    {// duplicate packet, discard

            #ifdef    DEBUG_PRINT
            fprintf(stderr, "got the pkt I've sent\n");
            #endif

            drop(p, DROP_RTR_ROUTE_LOOP);
            return;
    }
    else if (dst == mn_->address())
    {// packet is for me

            #ifdef DEBUG_BROADCASTING
            fprintf(stderr, "Packet is delivered!\n");
            fflush(stderr);
            #endif

            // we may need to send it to upper layer agent
            send_to_dmux(p, 0);

            return;
    }
```