NETWORK PROTOCOLS AND TIME SYNCHRONIZATION FOR UNDERWATER ACOUSTIC NETWORKS

NITTHITA CHIRDCHOO

NATIONAL UNIVERSITY OF SINGAPORE

2011

NETWORK PROTOCOLS AND TIME SYNCHRONIZATION FOR UNDERWATER ACOUSTIC NETWORKS

NITTHITA CHIRDCHOO

(*B. Eng. (Hons.), M. Sc.*)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2011

To my grand parents

and

my beloved family

Acknowledgment

I hereby would like to express my special gratitude to my supervisors, Assistant Professor Soh Wee Seng and Professor Chua Kee Chaing, who always give me the stimulating suggestions and encouragement. With their guidance and merciful supports at every step along the way of my study gave me the possibility to complete this thesis. Their research intuitions have made them not only the great resources of initiative new ideas but also the inspiration that leads me to the wish to become a researcher like them some day. What I have written here can never express my deepest gratitude I have towards them. However, it will remain in my heart forever. I have furthermore to thank all my former teachers and lecturers who taught and led me to be what I am today.

I am deeply indebted to National University of Singapore (NUS) for the research scholarship and the fundings all along the years of my study, research and conferences. I also obliged to Nakhon Pathom Rajabhat University for the opportunity to take a study leave and for handling the administrative issues during my absence. To Communication Society of the IEEE, I am thankful for a travel grant awarded to me for my paper presentation in the INFOCOM'07, Alaska, USA.

I am bound to Dr. Vikram Srinivasan, Dr. Mehul Motani, Dr. Mandar Chitre, Shiraz Shahabudeen, MingSheng Gao and Ng Hai Heng for their valuable comments, encouraging discussions and resourceful suggestions during our brainstorming sessions for STARFISH project. To everybody in the STARFISH project and Acoustic Research Lab (ARL) at NUS, it is a pleasure to have the collaboration from all of you.

I would also like to give my special thanks to my friends, Nok and her family, Pee Moy, Pee Boom, Kung and Pee Pu. Without their sincere friendship,

delightful moments and all those delicious Thai foods and parties we had together, life in Singapore would not have been this cheerful. I have furthermore to thank Teerachai Munjitjantra (Tee) for his worthy advice on C++ programing. For Mahtab A. K. M. Hossain and He Rong, I will always remember a great time I spend with them doing the class projects and the touching and long lasting friendship they had for me. To my dearest friend, Tor, my deepest thank can never be enough for his help, support, understanding, advice, comments and many many more.

Most of all, I would like to thank my parents, my brother and my sister who have always listened and been very supportive through my entire life, making me realize how sweet family can be so thriving to me. Thanks to all my little nieces who cherish my soul and my spirit with their youth and their innocence. I also would like to express my best regards and my cordial thanks to my grandparents for the wonderful memory of my child life staying with them although they are not on earth with me anymore.

Lastly, I would like to offer my best regards and holy blessings to all of those people who gave me precious supports in any respect during the completion of this thesis.

Table of Contents

Abstract **ix**

List of Figures **x**

List of Tables **xvi**

List of Abbreviations **xvii**

1 Introduction **1**

1.1 Background and Motivation 1

1.1.1 Communication Channel 2

1.1.2 Node Mobility 4

1.1.3 Node Density 5

1.1.4 Other Issues 6

1.2 Thesis Objectives 7

1.3 Main Contributions 7

1.4 Organization of the Thesis 9

2 Literature Survey **10**

2.1 MAC Protocols in UWA Networks 10

2.1.1 Applicability of Terrestrial MAC Techniques 10

2.1.2 UWA MAC protocols 15

2.2 Routing in UWA Networks 20

2.2.1 Data-centric Routing 20

2.2.2 Hierarchical-based Routing 22

2.2.3 Geographical-based Routing 22

2.3 Time Synchronization in UWA Networks 24

3 Aloha-based MAC Protocols with Collision Avoidance **28**

3.1 Aloha-based Variants and the Usefulness of Information Overheard 29

3.2 Aloha-CA 31

3.3 Aloha-AN 37

3.4 Normalized Throughput Analysis 40

3.4.1 Pure Aloha and Slotted Aloha 41

3.4.2 NP-CSMA and Aloha-HD 44

3.4.3 Aloha-CA 46

3.5 Simulation Model 53

3.6 Simulation Results 55

3.6.1 Verifying Aloha-CA's Analytical Normalized Throughput Lower-bound 55

3.6.2 Performance of Aloha-HD and NP-CSMA 57

3.6.3 Performance of Aloha-CA and Its Enhanced Variant 60
3.6.4 Performance of Aloha-AN 62
3.6.5 Effects of Inter-nodal Propagation Delay Estimation Errors 65
3.7 Conclusion 66

4 Handshaking-based MAC protocols for UWA networks 69
4.1 Introduction 69
4.2 RIPT 71
4.2.1 Overview of RIPT 71
4.2.2 Details of RIPT 74
4.2.3 Adaptive Train Size 82
4.2.4 When to Initiate an RTR Packet 83
4.3 Simulation Model 83
4.4 Simulation Results 86
4.4.1 Factors Affecting the RIPT's Performance 87
4.4.2 Performance Comparison Against Aloha-AN and MACA . 89
4.5 Discussion 92
4.6 Conclusion 95

5 Routing in UWA Networks 97
5.1 Introduction 97
5.2 SBR 99
5.2.1 Overview of SBR 99
5.2.2 Details of SBR 101
5.2.3 Finding the Next Relay Node 101

5.2.4 Implicit/Explicit Acknowledgments 103
5.2.5 The Analysis of SBR 104
5.3 SBR-DLP 105
5.3.1 Destination Location Prediction 106
5.4 Simulation Model 107
5.5 Simulation Results 109
5.5.1 Performance of SBR in Mobile Networks with Static Sink . 109
5.5.2 Performance of SBR and SBR-DLP in Fully Mobile Networks 116
5.6 Conclusion 120

6 Time Synchronization in UWA Networks 122

6.1 Introduction 122
6.2 The Causes of Error in Time Synchronization 124
6.2.1 Errors Caused by Uncertainty of Message Delivery Time . 125
6.2.2 Other Causes of Synchronization Errors 126
6.3 Clock Drift Modeling 126
6.4 MU-Sync 128
6.4.1 Overview of MU-Sync 128
6.4.2 Details of MU-Sync 128
6.4.3 Error Analysis of Propagation Delay Estimation 130
6.5 Simulation Model 133
6.6 Simulation Results 135
6.7 Discussion 139
6.8 Conclusion 141

7 Conclusions and Directions for Future Work **143**

7.1 Research Contributions 143

7.2 Future Work 147

7.2.1 Future Study 147

7.2.2 Interesting Research Problems for Investigation 148

Appendices **152**

A The Derivation of P_{k,B_2} 152

B Derivation of Backoff Probability $P_{bo}(\lambda'_j)$ 153

C Simulation Environment 155

C1 Aloha-CA and Aloha-AN 155

C2 RIPT 156

C3 SBR and SBR-DLP 157

C4 MU-Sync 159

D Protocol Stack 160

D1 Aloha-CA 160

D2 Aloha-AN 164

D3 RIPT 168

D4 SBR and SBR-DLP 174

D5 MU-Sync 179

List of Publications **192**

Name: Nitthita Chirdchoo

Degree: Doctor of Philosophy

Department: Electrical and Computer Engineering
National University of Singapore

Thesis Title: Network Protocols and Time Synchronization for Underwater Acoustic Networks

Abstract

Unlike terrestrial networks that mainly rely on radio waves for communications, underwater acoustic (UWA) networks utilize acoustic waves, which poses a new research challenge in the networking area. In this dissertation, we address three important networking problems which are: (1) how to improve the normalized throughput via MAC (Medium Access Control) design; (2) how to enhance the packet delivery ratio (PDR) when packets are routed in mobile UWA networks; and (3) how to reduce the time synchronization error in mobile UWA networks.

To address the first problem, we propose two simple random access MAC protocols which are suitable for small UWA networks. The protocols attempt to avoid collisions in a distributed manner, by making use of the information that it overhears, as well as the knowledge of inter-nodal propagation delays. The normalized throughput performance has been evaluated in both simulations and theoretical framework against the terrestrial-designed random access MAC protocols.

For larger networks, we propose a handshaking-based MAC protocol for multi-hop UWA networks. The design addresses the channel's long propagation delay characteristic by utilizing receiver-initiated reservations, as well as by

coordinating packets from multiple neighboring nodes to arrive in a packet train manner at the receiver.

As for the second problem, we introduce a location-based routing protocol that only assumes rough knowledge of the node's own location and the destination's location (e.g. sink) which enables the node to select the next best forwarder efficiently when it is combined with our sector-based routing mechanism. Furthermore, the need for precise knowledge in locating the destination node can be relaxed when the protocol is coupled with the location prediction mechanism.

In addressing the last problem, we develop a cluster-based synchronization algorithm for mobile UWA networks. Our design is the first to take both the long and time-varying propagation delays into account in the skew and offset estimation process. As a result, it can achieve a significant improvement in reducing the synchronization errors of mobile UWA networks.

Keywords: underwater acoustic network, MAC protocol, location-based routing, long propagation delay network, time synchronization.

List of Figures

2.1 Packets sent at the same time may or may not collide at receiver. 13

2.2 Channel listening may back off a packet unnecessarily. 14

3.1 An example of how Aloha-HD can miss an opportunity to avoid a collision. In this example, assuming that p_{12}, p_{13}, p_{23} are 1.5, 2 and 3.5 s, when p_{xy} denotes the inter-nodal propagation between Node x and Node y. Also, the packet transmission time is assumed to be 1 s. Furthermore, the packets generated at Node 3 and Node 1 are at the time $t = 0$ and 2.5 s, respectively. 31

3.2 An example of the database table at a node in Aloha-CA. 33

3.3 Useful region in which overhearing can help avoid collision. 35

3.4 The vulnerable periods of Pure Aloha in high latency networks; they are staggered with respect to each other depending on inter-nodal distances. 42

3.5 The vulnerable periods of Slotted Aloha in high latency networks. 43

3.6 The vulnerable periods of NP-CSMA when $a > 1$. 45

3.7 (a) Busy period of type B_1, which ends without any collision. Here, there are $m = 4$ successful packet transmissions. (b) Busy period of type B_2, which ends with a collision between at least two packets. Here, there are $k = 3$ successful packet transmissions. 49

3.8 Verification of Aloha-CA's analytical normalized throughput lower-bound via comparison with simulation results. Note that all inter-nodal propagation delays are set to $p_{\max}$ in the simulation network. 56

3.9 Approximating Aloha-CA's average normalized throughput by substituting $p_{\max}$ with p in the analytical normalized throughput lower-bound. 57

3.10 Comparing the normalized throughput of Pure Aloha, Aloha-HD, and NP-CSMA when the packet size is 2400 bits. 58

3.11 Effects of different packet lengths on NP-CSMA for 4-node network. 59

3.12 Maximum normalized throughput vs. pT-ratio for NP-CSMA and Aloha-CA. Note that "maximum normalized throughput" refers to the peak point in the respective normalized throughput vs. G' plots, and the peaks may occur at different G' for different pT-ratios. 60

3.13 Comparing the normalized throughput of NP-CSMA, Aloha-CA, and an enhanced Aloha-CA scheme for the 4-node network. 61

3.14 Normalized throughput of Aloha-AN in the 4-node network. . . . 63

3.15 Normalized throughput of Enhanced Aloha-CA and Aloha-AN with fixed delay estimation errors of 0, 33 ms and 133 ms for the 4-node network with 2400-bit packets. 65

4.1 4-way handshaking with multiple-node polling. 74

4.2 Our simulation network topology. Note that the nodes are not placed precisely at the grid intersection points. Also, the arrows in the figure only show the routes between a single node and its 16 two-hop neighbors. 84

4.3 Effects of $M_{\text{train,max}}$ and T_{avg} on normalized throughput. 86

4.4 Comparing the througput of RIPT, MACA, MACA-PT, Aloha-AN (grid spacing = 700 m). 90

4.5 Comparing the number of DATA packet transmissions and the number of DATA packet collisions of RIPT, MACA, MACA-PT and Aloha-AN. Note that the simulation duration is 8×10^5 s for every point. 91

4.6 Comparing the packet delays of RIPT, MACA, MACA-PT and Aloha-AN. 92

5.1 Forwarder selection at the sender. 102

5.2 Effects of sector size on the PDR. 110

5.3 Effects of sector size on the delay. 111

5.4 Effects of node density on the PDR. 112

5.5 Effects of node density on the delay. 113

5.6 Effects of node's speed on the PDR. 114

5.7 Effects of node's speed on the delay. 115

5.8 Effects of number of sectors in a 30-node network, for SBR (dashed) and SBR-DLP (solid). 117

5.9 Effects of number of nodes in a 6-sector network, for SBR (dashed) and SBR-DLP (solid). 118

5.10 Effects of node speed in a 30-node and 6-sector network, for SBR (dashed) and SBR-DLP (solid). 119

6.1 Illustration of clock drifts. 127

6.2 Phase 1: Skew and offset acquisition phase. Note that the propagation delay can vary during the REF packet exchange. 130

6.3 Linear regression at the cluster head with a total of 10 reference packet responses from node y. 130

6.4 Phase 2: Synchronization phase. 131

6.5 The error in time estimation VS the time elapsed since synchronization. 135

6.6 Effects of clock skew. 136

6.7 Effects of changing the number of REF beacons. 137

6.8 Effects of (t_3 - t_2). 138

6.9 Effects of the average direction change interval. 139

6.10 Effects of V_{max}. 140

D1 Packet generation in Aloha-CA. 160

D2 Flowchart of packet transmission. 162

D3 Flowchart of how the node reacts when a header is received. . . . 162

D4 Flowchart of how the node reacts when the data segment is received. 163

D5 Packet generation in Aloha-AN 164

D6 Flowchart of how the node decides whether to transmit the NTF packet. 165

D7 Flowchart of how the node reacts when an NTF packet is received. 166

D8 Flowchart of how the node decides whether to transmit a DATA packet. 167

D9 The start of RIPT. 168

D10 Flowchart of an RTR initiation. 168

D11 Flowchart of how the node reacts when the RTR packet is received. 169

D12 Flowchart of how the node reacts when the SIZE packet is received. 170

D13 Flowchart of how the node reacts after acquiring the SIZE packets from all of its neighbors. 171

D14 Flowchart of how the node reacts when ORDER packet is received. 172

D15 Flowchart of how the node reacts when the DATA packet is received. 173

D16 Flowchart of how SBR and SBR-DLP starts to route a packet. . . 174

D17 Flowchart of how the node in SBR reacts when Chk_Ngb packet is received. 175

D18 Flowchart of how the node in SBR-DLP reacts when Chk_Ngb packet is received. 176

D19 Flowchart of how the node chooses the next-hop forwarder. 177

D20 Flowchart of how the node react when ACK is received. 178

D21 Flowchart of the DLP algorithm. 178

D22 Flowchart of how the synchronizing node initiate the synchronization process. 179

D23 Flowchart of how the synchronizing node transmit the REF packet. 179

D24 Flowchart of how the synchronized node reacts when the REF packet is received. 180

D25 Flowchart of how the synchronizing node reacts when it receives RSP packet. 181

D26 Flowchart of how the synchronizing node checks for the RSP packet from the synchronized node. 181

List of Tables

3.1 Notations used for explaining useful region. 35

3.2 Notations used for evaluating backoff probability 54

4.1 Notations used for explaining the RIPT protocol. 75

4.2 An example illustrating the slot assignment strategy, where M_{train} = 4. 80

5.1 How Node S picks its next relay node. 103

6.1 A performance comparison of MU-Sync and TSHL when the simulation parameters are set according to the setting presented in Section 6.5. Note that the overhead cost is computed based on j synchronized nodes and n REF packets while the resynchronization is triggered when the error exceeds 100 ms. 140

D1 Notations used in explaining the protocol stack. 161

List of Abbreviations

ACK	Acknowledgment packet
ARQ	Automatic Repeat Request
AUV	Autonomous Underwater Vehicle
BER	Bit-Error-Rate
BEB	Binary Exponential Backoff
CDMA	Code Division Multiple Access
CSMA	Carrier Sense Multiple Access
CTS	Clear-To-Send
FDMA	Frequency Division Multiple Access
LHS	Left-Hand-Side
MAC	Medium Access Control
NP-CSMA	Non-persistent Carrier Sense Multiple Access
NTF	Notification packet

PDR	Packet Delivery Ratio
p.d.f.	probability density function
RHS	Right-Hand-Side
RTR	Ready-To-be-Receiver
RTS	Request-To-Send
RTT	Round-Trip-Time
SNR	Signal-to-Noise Ratio
TDMA	Time Division Multiple Access
UTC	Universal Time Coordinated
UWA	Underwater Acoustic

Chapter 1

Introduction

In recent years, there has been increasing interest in real-time exploration of underwater environments around the world [1] for diverse applications in environmental monitoring, natural resource exploration, military surveillance, and tsunami warning, etc. For these applications, it will be attractive to deploy networks of underwater sensors (either static or mobile, such as autonomous underwater vehicles (AUVs)) that could sense and relay data in real-time.

Unlike the terrestrial wireless networks that the technology has already been well-established, underwater acoustic (UWA) networks are still in its infancy stage and many areas have not yet been adequately explored. This dissertation deals with the protocol design for UWA networks. This includes the issues of medium access control (MAC), routing, and time synchronization.

1.1 Background and Motivation

In this section, we shall describe some background information on the nature and characteristics of underwater networks as well as the motivation of our dissertation work.

1.1.1 Communication Channel

Unlike the terrestrial wireless sensor networks that rely mainly on radio waves for communications, underwater networks utilize acoustic waves. This is because, radio waves can only propagate for a very short range, in the order of several meters in underwater environments, due to high attenuation (e.g., 6-10 m at 1-8 bps with the carrier frequency of 122 kHz [2]). Although radio waves can still be used for long range communication at very low frequencies (30 Hz-300 Hz), it is impractical to do so because of the need for large antennas and high transmission powers [3].

Optical signals are not suitable for long-range communication because light is highly scattered and absorbed underwater. For example, in the very clear water, it is possible to transmit with data rates up to several Mbps at the ranges of up to 100 m [4]. Hence, optical signals are being considered only for low-cost, short-range (1-2 m) and low-rate (57.6 kbps) data communication [2, 5].

As another alternative, acoustic waves appear to be a good choice for underwater communications because of their lower loss when compared to radio waves. As discussed in [6], the three main contributors for acoustic signal loss in underwater are spreading loss, absorption loss, and scattering loss. Spreading loss is distance dependent and can be either cylindrical or spherical. Cylindrical spreading happens in shallow water whereby the wave can only spread out to the limit of the upper and lower boundaries of the ocean (e.g. ocean surface and sea floor) while spherical spreading, on the other hand, occurs in deep water. Between the two, spherical spreading is a more severe loss because the signal energy decays at a higher loss exponent (the loss exponent for spherical spreading and cylindrical spreading are 2 and 1, respectively). Absorption loss is the amount of signal energy that converts into heat. It is frequency dependent and increases

with increasing frequency [7]. Scattering loss is a portion of the wave that is reflected away from the receiver and is a result of the dynamic and roughness of the sea surface. The experimental study in [8] shows that the time spread of the path taken by each of the scattered signal increases as a function of range, frequency and wind speed. The high delay spread can result in rapidly fluctuating of the channel and causes difficulties in channel estimation [6].

Although it is clear that acoustic communication is the most practical means for long-range communication underwater, it has some certain undesirable characteristics that need to be addressed when designing UWA networks. First, it is characterized by its narrow available bandwidth, which leads to low data rate. Specifically, the amount of available bandwidth depends on both the communication range and the carrier frequency; a long-range system that operates over several tens of kilometers may have a bandwidth of only a few kilohertz, while a short-range system operating over several tens of meters may have more than a hundred kilohertz of bandwidth [9]. As a result, the current range-rate product of the acoustic communications is limited, by the physical properties of the channel, to approximately 40 km·bps for vertical channels in deep water. This performance can dramatically degrade for horizontal channels in shallow-water [10]. Moreover, within this small available bandwidth, UWA channel experiences a time-varying multi-path, resulting in inter-symbol interference (ISI), Doppler shifts and spreads, that are more severe than that encountered in radio channels used in terrestrial wireless networks.

Second, acoustic waves travel slowly with the speed varying between 1433 and 1554 m/s, depending on the properties of sea water (e.g., temperature, pressure and salinity). The variation of the sound speed introduces the propagation delay variance which leads to the difficulty in time synchronization between nodes.

In general, we assume that the speed is approximately 1500 m/s, which is five orders of magnitude slower than radio waves. The low speed introduces a large propagation delay (6.7 s/km), which can lead to the low normalized throughput and the difficulty in sensing the channel's status.

While the time-varying multi-path, signal absorption, frequency-selective absorption, temporal spreading, Doppler induced frequency distortion are being addressed as a problem at the physical layer, the narrow available bandwidth of the UWA channel and the slow speed of the acoustic waves remain a big challenge in the data-link and the network layer.

1.1.2 Node Mobility

Underwater sensor nodes tend to exhibit some degree of mobility due to wind and ocean current, even if they are designed to be "static" nodes without any self-propelling capability. In addition, a group of AUVs may also be deployed for sensing tasks [11]. Node mobility often leads to dynamic network which can be very difficult to deal with when designing network protocols. For example, the dynamic nature of network makes the routing problem very challenging because route stability and route management become important issues in addition to energy and bandwidth efficiency [12]. Moreover, for routing protocols that assume the knowledge of network topology, the long propagation delay makes it highly likely that this information becomes stale by the time it reaches the intended node. From empirical observations, the ocean current typically moves at the rate of 3 - 6 km/hr (around 0.83 - 1.67 m/s) [13], while existing AUVs typically move at a rate of up to 2 m/s.

1.1.3 Node Density

In contrast to terrestrial sensor networks that generally assume fairly dense and fully connected coverage using small and inexpensive sensor nodes, nodes in UWA networks tend to be sparsely deployed [9], in the order of several kilometers apart, and network disconnectivity can be quite common. The reasons for low node density in UWA networks are discussed as follows. Firstly, the high cost of underwater sensor nodes limits the size of most UWA networks. Secondly, although less significant in shallow-water networks, in order to cover the whole water column in an area of interest, nodes in underwater networks are usually deployed in different depths, resulting in a 3D-network. According to [14], to achieve the same node density in a 2D square area of a^2 and a 3D cube volume of a^3 while having the same communication radius of r, a 3D network requires $\frac{3a}{4r}$ more nodes than a 2D network.

Node density has significant effect on network protocol design, especially in the area of routing. Specifically, some of the protocols proposed for a 2D network cannot be easily extended to work well in a 3D system. For example, in the Blind Geographic Routing (BGR [15]), a node specifies a forwarding area (e.g., sector, circle) using the location information (of the sender and of itself). Only nodes within this forwarding area are allowed to relay the packet. However, when the node cannot find any relay node within its current forwarding area, it tilts its forwarding area 60° in an arbitrary direction and rebroadcasts the packet. It is shown in [14] that in 3D, the forwarding volume only covers half of the communication volume of the corresponding forwarding areas in 2D, resulting in a higher number of packets sent and a low packet delivery ratio (PDR). Note that the low density of sensor nodes coupled with a node's mobility makes it highly likely to have a disconnected (network's void) network. The disconnectivity and

the delay problem make it a major challenge when designing network protocols for UWA networks.

1.1.4 Other Issues

Although sensor nodes in both terrestrial and underwater networks are battery-operated, it is much more difficult to recharge the battery for an underwater sensor once it has been deployed. Unlike a terrestrial sensor node that may utilize a solar cell to recharge its battery, manually replacing the battery is more common for an underwater sensor node. In order to reduce the cost of network maintenance, network protocols designed for underwater networks need to focus on reducing energy consumption, e.g., by focusing on schemes such as collision avoidance and link-layer acknowledgment.

It is also interesting to note that the relative ratios of power consumption in underwater acoustic modems among the different operating modes are very different from those of terrestrial radio-based modems. In general, a terrestrial radio-based modem in either receive or idle mode consumes approximately half of the amount of energy that it consumes in transmit mode [16]. On the other hand, an UWA modem, such as the one used in WHOI [17], consumes less than 1% of the transmit power when it is in either receive or idle mode. The significant difference on power consumption between different modes of the modem implies that the trade-off criteria among the different modes of operation in UWA modems are likely to be different from those used in terrestrial RF modems [16].

Other than the energy efficiency issue, underwater sensor nodes are prone to failure due to fouling and corrosion from sea water, resulting in even higher of network dynamic, due to topology changes (e.g., nodes leave the network). Network protocols must also adapt quickly to the change of network parameters (e.g.

node density, topology) due to these effects.

1.2 Thesis Objectives

The following summarizes the thesis objectives of this dissertation:

1. We would like to enhance the normalized throughput while reducing the energy consumption of UWA networks via the MAC design. This is to be achieved by designing the MAC protocols that allow sensor nodes to efficiently avoid collisions. The collision avoidance mechanism is designed by taking the channel's unique characteristics into account along with the cross-layer design technique.

2. We would like to improve the PDR when packets are routed in mobile UWA networks. We propose the design to improve the existing UWA routings by focusing on the channel's long propagation delay and the node mobility. The location-based routing and location prediction mechanism will be considered in this study.

3. We would like to achieve smaller errors in time synchronization of mobile UWA networks. This is possible by allowing the node to take the long and time-varying propagation delay into account during the process of the skew and the offset estimation.

1.3 Main Contributions

The main research contributions of this dissertation are as follows:

1. The study of Aloha-based variant protocols in both theoretical and simulation frameworks is carried out, in order to understand the behavior of each protocols in the long propagation delay networks. From this study, we realized that with the knowledge of inter-nodal propagation delays, a node could help avoid collisions in a distributed manner by utilizing the information that it picks up from overheard packets. This only arises in a long propagation delay environment such as UWA networks.

   Therefore, the design of two novel but yet simple random access MAC protocols that adopt the idea of using the overheard information to help the node in avoiding collisions are proposed. The usefulness of overheard information is also discussed and the theoretical derivation of the normalized throughput of one of the proposed protocols is also provided. From extensive simulations, both schemes have shown that the collision avoidance mechanism that uses the overheard information is simple but very efficient, leading to the enhancement of the normalized throughput.

2. The design of a handshaking-based MAC protocol that is suitable for multihop UWA networks. The scheme utilizes a packet train technique to overcome the low normalized throughput which results from the long propagation delay. Unlike other existing works that the packet train can only be formed from a single node, our packet train is formed from multiple neighboring nodes. This helps to improve the normalized throughput even when the network is operating at low load. In addition to the use of packet train, our design utilizes a receiver-initiated reservation approach which can greatly help the node to avoid collisions.

3. The development of a location-based routing scheme that can work well even when the network is encountering network disconnectivity, node mobility

and high bit error rate (BER) experienced in mobile UWA networks. The proposed routing protocol is able to achieve high PDR while maintaining low overhead. It is also shown that the assumption that the destination node's location is known can be relaxed when the protocol is coupled with a simple destination location prediction scheme.

4. The development of the first synchronization scheme for mobile UWA networks. Our scheme takes node mobility into account in the skew and the offset estimation process, resulting in the more accurate value of both the estimated skew and the estimated offset. This results in smaller time synchronization errors when compared to the previously proposed schemes.

1.4 Organization of the Thesis

The remaining of this dissertation is organized as follows. Chapter 2 presents the survey of the literature in the fields of MAC, routing, and time synchronization that are designed for UWA networks. Chapter 3 provides the detailed study of Aloha-based variants and the design of two novel random access MAC protocols that utilize channel's long propagation delay to be their advantage, resulting in the enhancement of their normalized throughput. Chapter 4 introduces a design and performance evaluation of the handshaking-based protocol that is suitable for multi-hop UWA networks. Chapter 5 deals with the location-based routing in mobile UWA networks. Chapter 6 describes the design of time synchronization scheme that can tolerate long and time-varying propagation delay in mobile UWA networks. Finally, Chapter 7 provides a summary of our research contributions, and describes some possible future research directions.

Chapter 2

Literature Survey

Although there are many differences between terrestrial wireless and UWA networks as we have previously discussed in the last chapter, both networks do share some similarities and common requirements (e.g., energy efficiency, bandwidth limitation, etc.). As such, we shall see that the design of existing UWA network protocols are heavily influenced by the already well-established technologies of the terrestrial wireless networks. In this chapter, we discuss the related works in the fields of UWA networks, focusing in three major areas: (1) MAC protocol, (2) routing, and (3) time synchronization.

2.1 MAC Protocols in UWA Networks

2.1.1 Applicability of Terrestrial MAC Techniques

Many MAC protocols have been proposed for terrestrial wireless networks with the goal of achieving high normalized throughput and low collision rate. Some recent schemes have also added features such as sleep-wake scheduling, topology management, and power control, etc. Although the goals of both underwater and

terrestrial MAC protocols may be the same, the high performance of the latter could be severely degraded when they are applied blindly in underwater, without considering the high propagation delay and low bit rate of the UWA channel.

In the following, we provide some insights into the performance of terrestrial MAC techniques in the underwater environment, and explain why new techniques are necessary.

2.1.1.1 Deterministic MAC Techniques

1. **Frequency Division Multiple Access (FDMA)**: The splitting of the available frequency band into multiple channels requires the node to be equipped with multiple transceivers, in order to support multiple frequency channels. This consequently will increase the system cost.

2. **Time Division Multiple Access (TDMA)**: TDMA's performance relies heavily on clock synchronization. In terrestrial networks where synchronization is relatively cheap, TDMA can achieve high normalized throughput by proper scheduling of data transmission by a node in the network. However, in underwater applications, there are two main reasons why TDMA will not work well. First, underwater nodes usually need to operate for long periods without any maintenance. Even if these nodes are perfectly synchronized at the time of initial deployment, they will eventually lose their synchronization, especially if inexpensive clocks are used. As a result, periodically resynchronization is required. Second, owing to the long propagation delay in UWA networks, the slot lengths need to be long (the slot length must be long enough to accommodate both the DATA packet transmission time and the maximum inter-nodal propagation delay), thus resulting in

a low normalized throughput. Specifically, the maximum achievable normalized throughput of TDMA depends on the ratio of the DATA packet transmission time to the maximum inter-nodal propagation delay. For example, in a long-range communication (e.g. 10 km transmission range), the maximum inter-nodal propagation delay can be as high as 6.67 s. If the DATA packet is 1 s-long, the maximum achievable normalized throughput is approximately only 13%.

3. **Code Division Multiple Access (CDMA)**: In order to differentiate between different senders, CDMA uses a different pseudo-noise code for each sender. The study in [18] reveals that CDMA's normalized throughput in UWA networks is actually lower than that of TDMA.

2.1.1.2 Non-deterministic MAC Techniques

1. **Polling**: The advantage of using polling is that each node has guaranteed channel access without the need for contention, thus allowing packet delays to be guaranteed as well. However, in underwater networks with long propagation delays, the need to poll each node before it is allowed to transmit a data packet is inefficient. In particular, a node that is polled but has no data to transmit still introduces a potentially long round-trip delay, thus resulting in under-utilization of the channel. In addition, the polling node also makes the network susceptible to single node failures.

2. **Channel Listening**: The channel listening technique, as employed by the Carrier Sense Multiple Access (CSMA) protocol, avoids collisions by requiring each node to listen to the channel before transmitting a packet. However, this does not work well in underwater networks due to the high

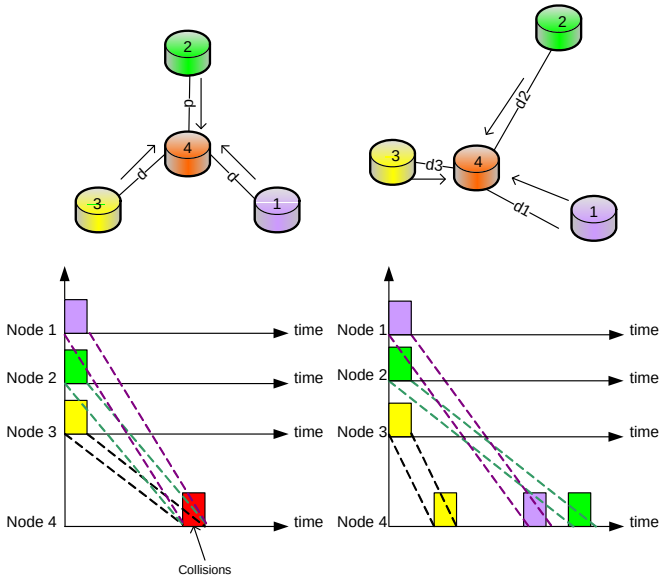

Figure 2.1: Packets sent at the same time may or may not collide at receiver.

propagation delay. The underwater channel may well be perceived to be idle by a node, say, A, even though a neighboring node, B, has already started transmitting for some time. Nevertheless, even if A subsequently decides to transmit a packet, it may or may not result in collisions at their respective intended receivers. Note that a "collision" that leads to data corruption only occurs when packets overlap at an intended receiver, but not when the packets pass each other in the transmission medium. As we will see in the examples shown in Fig. 2.1, a node's failure to sense an ongoing transmission before transmitting its own packet does not necessarily result in collisions, even if all the nodes concerned are within each other's transmission range. Suppose nodes 1, 2 and 3 coincidentally decide to send packets to node 4 at the same instant. In the network on the left, nodes 1, 2 and 3 are equidistant from node 4, thus all the three packets collide at node 4. In the network on the right, however, none of the packets collides at node 4, because of the differences in propagation delays.

Similarly, when a node senses that there is an ongoing transmission in the

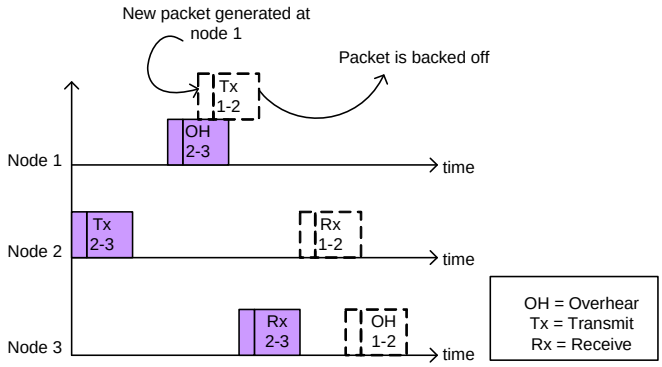

Figure 2.2: Channel listening may back off a packet unnecessarily.

channel, it does not necessarily mean that it cannot transmit a packet. This is because its packet may not overlap with the previously sensed packet at the intended receiver. Fig. 2.2 shows an example in which a packet may be backed off unnecessarily when it is alright to transmit[1]. Here, node 2 transmits a packet to node 3, which is overheard by node 1. Suppose node 1 generates a packet that it wishes to send to node 2 at the time when it overhears the packet "2-3". Based on pure channel listening without any additional information, node 1 may back off from transmitting the packet "1-2", although it will not have collided with packet "2-3" at both nodes 2 and 3. Thus, from the above examples, we see that there are certain circumstances in which pure channel listening in underwater networks not only does not help to avoid collision, but may also result in low normalized throughput because it is over-conservative.

3. **Handshaking**: The handshaking technique exchanges short control messages, namely, Request-To-Send (RTS) and Clear-To-Send (CTS), prior to transmitting each data packet. In terrestrial networks, the RTS/CTS frames also help alleviate the "hidden" and "exposed" terminal problems

[1]The number pair inside the packet, e.g., "2-3", denotes that the packet's sender is node 2, and its intended receiver is node 3.

in multi-hop networks. However, when implemented in UWA networks, the exchange of multiple (albeit short) control messages between the sender and the receiver is very expensive. The slow speed of sound in underwater, compounded by the sparse deployment of underwater nodes (typically hundreds to thousands of meters apart), introduce long delays that seriously impair the normalized throughput. In fact, both [19] and [20] have shown that the handshaking technique offers a lower normalized throughput than Aloha in the case of single-hop UWA networks.

4. **Aloha**: As one of the earliest proposed random access MAC protocols, Aloha is seldom chosen for terrestrial networks when normalized throughput is of main concern. It is well-known that the pure Aloha protocol can only achieve a maximum normalized throughput of 18% in terrestrial wireless networks when no synchronization is present, and 36% with its slotted variant. These normalized throughput levels have always been regarded as too low for terrestrial network applications. However, the Aloha concept should not be overlooked when designing underwater MAC protocols. Firstly, it has been shown in [19] that pure Aloha still maintains its maximum normalized throughput of 18% underwater, while retaining its simplicity. More importantly, as has been discussed above, the more complex schemes previously designed for terrestrial networks do not necessarily offer better normalized throughput than Aloha when they are applied to the underwater environment.

2.1.2 UWA MAC protocols

Currently, research efforts in UWA MAC protocols are still relatively young. Some work in the literature, such as [21], has adopted a centralized control approach

in which a master node performs the access scheduling of its slaves. While this is simpler, the distributed control approach in which each node schedules its own channel access is preferred because it is more scalable, responds faster to topology changes, has lower overheads in the number of control messages, and does not suffer from a single node (master node) failure.

In [22], Rodoplu and Park propose a MAC protocol that achieves energy efficiency by reducing the number of collisions. Each node schedules its own time to transmit the next packet, and broadcasts this information by attaching it to the current data packet. Upon hearing the broadcast, other nodes will know when to wake up for the subsequent packet, and they may go to sleep at other times. However, in order to maintain a low collision rate, each node must operate with a small duty cycle, which makes it difficult to achieve a high normalized throughput performance.

In [18], Morns *et al.* propose two scheduling protocols to control data packet transmission and arrival times. One protocol is based on CDMA, while the other one is based on TDMA. However, both protocols require clock synchronization between all the nodes. Also, the time slot allocation for individual nodes becomes hard to manage when the number of nodes grow. Guo *et al.* introduce the propagation-delay-tolerant collision avoidance protocol (PCAP) in [19], which is a handshaking-based protocol. It also requires clock synchronization between neighboring nodes, similar to [18]. Its uniqueness is that it allows a sender to perform other actions during the long wait between the RTS and the CTS frames. Although its maximum normalized throughput is 20%, which is higher than what a typical RTS-CTS handshaking-based protocol such as MACA [23] can achieve underwater, this is merely comparable to Aloha's normalized throughput. Its enhanced version, the adaptive-PCAP (APCAP), is recently proposed in [24, 25].

By allowing the destination node to negotiate with the source node on when to transmit the CTS, the normalized throughput can be greatly improved. However, clock synchronization is still required. Molins and Stojanovic propose in [26] a slotted random access MAC protocol, which also requires clock synchronization. It is also handshaking-based, but an RTS or CTS frame can only be transmitted at the beginning of each time slot. Although the protocol achieves guaranteed collision avoidance for its data packets, its normalized throughput is considerably lower due to both the long slot length requirement and the long delay introduced by the handshaking mechanism. Peleato and Stojanovic [27] improve the normalized throughput efficiency of the handshaking-based protocol by allowing a node to use different handshake lengths in which are determined by the inter-nodal distance. In addition, collision avoidance mechanism is also implemented such that the node sends the warning packet to defer the data transmission form the current handshaking node if there is possible collision.

Sozer *et al.* propose, in [28], another handshaking-based protocol that combines the capability of error detection using a Stop-and-Wait Automatic Repeat Request (ARQ). By adding a WAIT packet, it is possible to reduce the repetitive transmission request by telling the source node to wait while the destination node is busy and cannot immediately respond to the transmission request.

In [29], the adaptation of the original MACA to be used in a large propagation delay environment is studied. The authors propose the state transition which is suitable for MACA that is implemented in a high propagation delay environments. The derivation of the theoretical upper and the lower bound of MACA's normalized throughput in a single-hop network is also given in their study. The modification of the terrestrial MACAW [30] to be implemented in underwater is suggested by Foo *et al.* in [31]. The authors claim that different

priority level should be given at each node. This way, when two nodes happen to transmit the RTS to each other, a node with higher priority will eventually win the channel. Similarly, in [32], Ng *et al.* propose that in order to enhance the normalized throughput of MACA in underwater, modifications in three potential areas should be considered: state transition rules, packet forwarding strategy, and backoff algorithm. The authors then propose MACA-U, a handshaking-based protocol that can alleviate the problem of being over-conservative encountered in the original MACA by allowing the node to persistently wait for the CTS or DATA packet, instead of transiting itself to a silent state, even if it receives or overhears another RTS or CTS packet from other nodes. In addition, MACA-U gives higher priority to the node that are relaying a packet over the one that sending a newly generated at the node itself. As a result, MACA-U is less likely to have packet drop at the intermediate node. For the backoff, the original Binary Exponential Backoff (BEB) appears to generate aggressive contention which may result in higher collision. To solve the problem, the authors suggest that the long propagation delay should be considered in the determination of the BEB backoff interval. Specifically, the backoff interval should be set as $b \cdot (T_{\text{ctrl}} + p_{\max})$, where b is the number chosen randomly in the backoff window size W and T_{ctrl} is the transmission time of the control (e.g, RTS, CTS) packet. In [33], Kebkal *et al.* achieve normalized throughput efficiency with the adaptation of MACAW. This is possible by scheduling the data transmission in parallel with the acknowledgment (ACK). The data chunk is divided into two portions such that the transmission time of the second portion can be scheduled to transmit in parallel with the ACK without collision.

The analytical study of the suitability of both random access and handshaking-based protocol in single-hop underwater networks are carried out in [34]. In [35],

Kredo and Mohapatra propose a hybrid protocol that bridges the TDMA with the random access scheme, in order to achieve benefits of both schemes.

Recently, Aloha-based protocols have also gained attention due to their simplicity. The normalized throughput analyses of both pure and slotted Aloha in underwater are recently presented in [36] and [37]. Syed *et al.* [37] also discuss the effects of space-time uncertainty on the performance of high latency networks, such as underwater networks. Unlike terrestrial wireless networks in which space uncertainty has negligible effects (due to the extremely small propagation delay), the performance of high latency networks is affected by both space and time uncertainties. In particular, space uncertainty is caused by the nodes' locations, which result in different propagation delays, while time uncertainty is caused by the randomness of packet transmissions. Since both effects are not negligible in high latency networks, it may not provide any benefit to take care of just one of the uncertainties in a MAC protocol design; it has been shown in both [36] and [37] that the classic slotted Aloha degrades to pure Aloha in high latency networks. The authors also propose a propagation delay tolerant Aloha (PDT-Aloha), a protocol that can enhance the normalized throughput of the traditional slotted Aloha by simply adding a proper-size guard band in the transmission slot. Later, Ahn and Krishnamachari show the performance analysis of PDT-Aloha in [38]. This gain of interest shows that Aloha should not be overlooked due to its low normalized throughput experienced in wireless terrestrial networks.

Other research concentrates on the optimization techniques to increase the channel efficiency. Stojanovic [39] shows that in order to maximize the efficiency of the normalized throughput performance in a stop-and-wait automatic repeat request (ARQ) system, selecting an optimal packet size is essential. This optimal packet size is a function of link parameters such as range, rate, and error

probability. Pompilini *et al.* [40] propose a closed-loop distributed algorithm to set the optimal transmit power and code length in a CDMA system. The work by Harris *et al.* [41] suggests that normalized throughput can be maximized by considering the packet train length adaptation, the forward error correction, and the packet size adaptation in associated with distance between the sender and the receiver.

2.2 Routing in UWA Networks

In this section, we review existing works on UWA routing, according to their classification. Because routing algorithms are application- and network architecture-oriented, there are numerous ways to classify them. Here, we classify them into three main categories: data-centric, hierarchical-based and geographical-based. Only a short description of each category is given here. More details can be found in [42, 12, 43].

2.2.1 Data-centric Routing

With data-centric routing, due to the lack of global addressing, data is usually transmitted from multiple sensors within the region of an event. Data aggregation and in-network processing is commonly utilized so that the number of data transmissions can be reduced.

In [44], Lee *et al.* propose a multi-hop ad hoc routing and in-network processing protocol for time-critical aquatic applications, namely, *"Under-Water Diffusion (UWD)"*. The UWD aims to minimize the number of packet transmissions so that channel contentions can be greatly reduced. Although the UWD adopts data diffusion and in-network data aggregation techniques that are widely used

in terrestrial sensor network, it tries to limit the number of transmissions by reducing the number of flooding packets. While direct diffusion needs to periodically flood interests and route status (e.g., link failure has occurred) to handle topology changes due to node mobility, the UWD only floods the interest once in the initialization phase. When the sink wants to modify its interest, instead of flooding, it piggybacks the new interest on another packet.

Nevertheless, significant flooding is still required when applying the UWD in a mobile environment. This is because flooding is the only way to find the path from a node (that the event occurs) to the destination. While the UWD can perform well in static networks, its performance can be degraded significantly if the network is highly dynamic. This is because when the network is dynamic, topology is expected to be changed often.

In [45], Seah and Tan propose a virtual sink architecture that exploits the multipath nature of broadcasting communication to achieve a robust and energy efficient routing protocol. When multipath data delivery is combined with multiple virtual sinks, the scheme can overcome the problem of high traffic contention occurring near the sink, which is a common problem encountered by typical multipath-based routing protocols. This results in a significant enhancement in the data packet deliver ratio. In contrast to the typical multipath routing scheme that aims to set up multiple routes between a pair of nodes [46], the virtual sink architecture maintains multiple spatial routes to different sinks, which are connected to the actual sink via a high speed communication link to avoid contention and collisions, thus achieving high reliability.

However, the virtual sink architecture assumes that the sensor nodes are densely deployed (e.g., 100 nodes in the area of 2.5 km by 2.5 km) which may not be practical due to the high cost of the underwater sensor nodes. Despite the

fact that the sender utilizes a hop count mechanism to find the diverse path back to the sinks, the issues of reacting to a dynamic network have not been addressed in the paper. Furthermore, the network is assumed to be fully connected for the whole simulation duration, which ignores the possibility of network voids forming.

2.2.2 Hierarchical-based Routing

Another approach in data aggregation is using hierarchical-based routing which usually assumes that there are special nodes (e.g. cluster-head, master node) that have higher capability in terms of processor, memory or available energy. These special nodes are responsible for data aggregation and route decision making. To date, the hierarchical-based approach has not gained much interest from the underwater routing community due to its high overheads required in cluster formation and management. A delay-sensitive routing protocol has been proposed in [47]. The authors suggest a centralized control approach in which a node relies on the topology information gathered from the surface station which decides on the optimal path to route a packet.

2.2.3 Geographical-based Routing

Recently, geographical or location-based routing has attracted significant interest from the underwater routing community due to its suitability for mobile underwater sensor networks. Because network topology is unavailable in these networks, non-geographic routing algorithms need to find suitable routes to the destination node through flooding. However, flooding should be avoided because it is very expensive in terms of bandwidth and energy utilization. Geographical routing eliminates the need for flooding by utilizing the available location information (assuming that all nodes are aware of their own and the destination's positions), to

limit the number of nodes that can direct a packet toward the destination.

The first routing algorithm designed for mobile underwater sensor networks, the *vector-based forwarding (VBF)*, is proposed in [48]. In VBF, each of a sender's neighboring nodes determine its candidacy to be the next relay node by first computing the distance between itself and a virtual vector from the sender (S) to the destination (D), denoted as a "*routing vector* $\overline{SD}$". The predefined radius from the routing vector forms a "*routing pipe*". If the node is located within this routing pipe, it is a candidate to be the next relay node. Multiple candidates compete among themselves to be the next relay node using a *desirableness factor*. More specifically, each candidate calculates its desirableness factor which tells the node how long it should hold the packet before attempting to relay the packet. Because the desirableness factor favors the node that is located nearest to the destination, thus giving it a higher priority to be a next relay (holding the packet shorter than others). The VBF is extended to the HH-VBF (hop-by-hop VBF) in [49] in order to overcome two problems encountered in the VBF: small data delivery ratio in sparse networks and sensitivity of the routing pipe radius. Specifically, instead of using a single routing pipe from the source to the destination, the HH-VBF forms the routing pipe in a hop-by-hop fashion. As a result, HH-VBF can improve the PDR of VBF.

Although the HH-VBF can achieve better PDR than VBF, the sensitivity of the PDR to the routing width still exists in a sparse network, especially when it is often that there is no sensor node lying within the routing pipe.

Focused Beam Routing (FBR) has been recently proposed in [50]. It is a geographical routing algorithm integrated with an open loop power control mechanism that allows the sender to select its transmit power levels from P_1 to P_n. The protocol starts with a sender node (S) broadcasting a request-to-send (RTS)

to its neighbors using a certain power level (e.g., say P_i). A node hearing this RTS packet will first determine its relative distance from a virtual line drawn from the sender to the destination (D), denoted as $\overline{SD}$, by using the sender's and destination's locations included in the RTS packet and its current position. The node becomes a candidate relay node only if it is positioned within the cone of angle $\pm\theta$, referred as a "transmitting cone". Only nodes that are candidate relay nodes respond with the clear-to-send (CTS) packet including the their locations. Next, among all the candidate relay nodes, the sender selects the next relay node that is located nearest to the destination. If there is no node within the transmitting cone, the sender increases its transmit power level from P_i to P_{i+1} where $i + 1 \leq n$. In the case that there is still no neighbor node even after the sender has exhausted with its maximum power level, P_n, it will shift its cone in either right or left direction to cover the entire vicinity.

Similarly to the VBF and the HH-VBF, the performance of the FBR is highly sensitive to the size of its transmitting cone, especially in a sparse network.

2.3 Time Synchronization in UWA Networks

Since there are very limited work on time synchronization proposed for UWA networks, we shall also discuss some of interesting approaches in time synchronization proposed for terrestrial wireless sensor networks that could be adapted for implementation in underwater. The previously proposed synchronization schemes found in the literature can be divided into two categories: the receiver–receiver based approach, and the sender–receiver based approach. An example of the receiver-receiver based approach is the Reference Broadcast Synchronization (RBS) scheme [51]. In RBS, a node sends a beacon pulse to its neighbors. Upon receiving the pulse, the receiver marks its local time at which it receives the

beacon pulse. By assuming that the propagation delay is negligible, each of the receivers is assumed to have received the beacon pulse at the same instance. Next, each pair of the receivers simply exchange their pulse receiving time to calculate the difference in their offsets. For higher accuracy, the scheme can be extended to use multiple beacon pulses to determine both the clock skew and the clock offset by using linear regression and statistical technique, respectively. As we can see, the RBS does not rely on the time stamp from the sender side; this reduces the sender's nondeterministic effect. Although the authors claim that RBS can achieve very high accuracy (1.6 μs using 30 beacon pulses), its performance can degrade drastically when operating in mobile UWA networks, in which long and varying propagation delay is dominant. Moreover, with the use of unidirectional beacon pulse, it is impossible to compute and compensate for the propagation delay, thus leading to large synchronization error.

On the other hand, instead of using unidirectional message exchange as in the RBS, the sender-receiver approach combines bidirectional message exchange with local time stamping, in order to retrieve the node's offset and skew. Ganeriwal *et al.* proposed TPSN [52], a two-phase network-wide synchronization algorithm for wireless sensor networks. The TPSN first uses a *level discovery phase* to define the hierarchical topology of the whole network. Then, in the *synchronization phase*, a pair of nodes can learn their clock drift using bidirectional message exchange, with the sending node inserting its local time stamp on each message. The drift and propagation delay can be extracted from the time stamp collected from the two-way message exchange. The main drawback of TPSN is that it computes the clock drift by only estimating the offset. Without correcting the clock's skew, it is obvious that the TPSN will need frequent re-synchronization.

Although there has been extensive study in the field of UWA networks from

all around the world, the work in the area of underwater synchronization is still very limited. Syed and Heidemann recently propose the TSHL [53], which is a time synchronization algorithm designed for high latency networks. The design tries to minimize the synchronization error by estimating and compensating both the clock skew and offset, utilizing MAC-layer time stamping and bidirectional message exchange. While assuming that all the nodes in the network are static, the TSHL takes into account the long propagation delay when determining the clock offset. The clock skew estimation can be achieved by applying linear regression over multiple two-way reference packet exchanges. The major drawback in TSHL is that it assumes constant propagation delay during the n reference packet exchanges. This assumption is no longer valid in mobile networks.

Recently, Tian *et al.* proposed a localization and synchronization scheme for 3D UWA networks in [54], using atomic multilateration and iterative multilateration techniques. The scheme utilizes external anchor nodes that are located on the surface of the ocean, which already know their locations and time without error. The synchronization packet broadcasted from the anchor i includes the current location (x_i, y_i, z_i) of the anchor, and the time of transmitting the packet (t_i). In order to compute its location, each node needs to hear from at least five anchor nodes. Upon receiving enough synchronization packets, the node performs multilateration in order to obtain its location. Next, the node learns the drift between itself and the anchor by comparing its local time of receiving the packet with ideal time (t_i plus the propagation delay). The main drawback of this scheme is that it may not always be practical to have anchor nodes floating on the surface of the ocean, due to security reason. Moreover, the algorithm utilizes a hierarchical approach for network-wide synchronization (multi-hop networks). This means that, in order to synchronize a node, it needs to have at least five

neighboring nodes of a higher hierarchical level, acting as beacon nodes. This may be quite difficult to achieve in UWA networks since the number of sensor nodes are typically limited due to economical reasons. The most serious drawback is that the scheme only aims to estimate the offset. Without estimating the clock skew, frequent resynchronization is expected.

Chapter 3

Aloha-based MAC Protocols with Collision Avoidance

From Chapter 2, we find that the protocols that employ handshaking techniques can inevitably amplify the effects of the long propagation delay which limits the normalized throughput. On the other hand, those that rely on time slot allocation generally require slot lengths that are longer than the maximum propagation delay, which not only limits normalized throughput but incurs synchronization problems due to clock drift. These lead us to consider whether simpler MAC protocols may, in fact be more capable of achieving high normalized throughput and low collision rate for underwater networks.

In this chapter, we study Aloha-based protocols and show that their performance in underwater networks can be quite attractive, especially when enhanced with modifications that take into consideration the long propagation delays. Specifically, we propose two Aloha-based random access MAC protocols, namely, Aloha with collision avoidance (Aloha-CA) and Aloha with advanced notification (Aloha-AN), for single-hop UWA networks. In both schemes, each node avoids collisions in a distributed manner by utilizing information that it

overhears as well as the knowledge of inter-nodal propagation delays.

The remainder of this chapter is organized as follows. In Section 3.1, we describe two simple Aloha-based variants, before presenting our proposed schemes in Section 3.2 and Section 3.3, respectively. In Section 3.4, we provide a theoretical analysis of the Aloha-based protocols. We then describe in Section 3.5, the simulations that we have carried out to compare the performance of the proposed scheme with several others. Section 3.6 shows the simulation results of the proposed protocols. Finally, we conclude in Section 3.7.

3.1 Aloha-based Variants and the Usefulness of Information Overheard

Ideally, MAC protocols for UWA networks need to tolerate high propagation delays while offering high normalized throughput and low collision rates. They must also be simple because underwater nodes offer limited resources in terms of memory, CPU speed and power (batteries cannot be easily replaced or recharged). The Aloha protocol is one of the simplest MAC protocols, therefore, we have chosen to study Aloha-based variants in this work. In the following, we first describe an intuitive variant that can improve performance over pure Aloha. We then illustrate with an example how this variant could have avoided collisions if it were to utilize some information extracted from overheard packets. This forms the motivation for our proposed Aloha-CA, which will be described in the next section.

In pure Aloha, a node simply transmits a packet whenever it has anything to send, regardless of whether it is currently receiving a packet. This is very inefficient since the packet being received will definitely be discarded, resulting

in a lower normalized throughput and higher energy wastage. A possible variant to pure Aloha, which we shall call "Aloha half-duplex (Aloha-HD)", removes this inefficiency by having a node that is receiving a packet defer sending a new packet. Hence, any new packet generated while the node is receiving a packet will be backed off. Note that the above decision requires the node to decode the packet's header for the destination address.

Aloha-HD can still transmit a packet indiscriminately so long as it is not the intended receiver for the packet it currently overhears. Specifically, its transmitted packet may collide with the overheard packet at the former's or the latter's intended receiver, or both. An example is illustrated in Fig. 3.1, where node 3 sends a packet intended for node 2. Although node 1 overhears the packet "3-2", it transmits a packet to node 2 because the overheard packet is not meant for it. Packet "1-2" thus subsequently collides with packet "3-2" at node 2. This example depicts the worst case, in which both packets "3-2" and "1-2" are corrupted at their common intended receiver, i.e., node 2. Note, however, that if node 1's packet were intended for node 3, then only packet "3-2" would have been destroyed at node 2, while node 3 will receive the packet from node 1 successfully.

From the above example, we observe that each node can help avoid collisions by using the "sender-receiver" information that it picks up from overheard packets to make more informed decisions on when to avoid transmitting a packet. This opportunity only arises in a long propagation delay environment such as underwater networks, as opposed to typical terrestrial wireless networks in which concurrent transmissions by more than one node will almost always result in a collision.

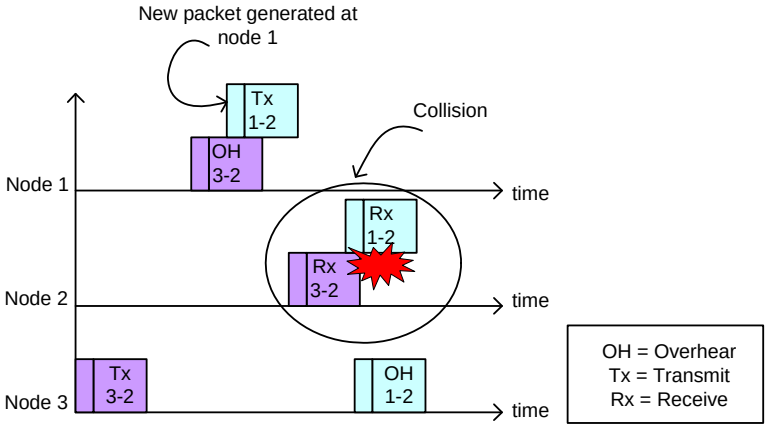

Figure 3.1: An example of how Aloha-HD can miss an opportunity to avoid a collision. In this example, assuming that p_{12}, p_{13}, p_{23} are 1.5, 2 and 3.5 s, when p_{xy} denotes the inter-nodal propagation between Node x and Node y. Also, the packet transmission time is assumed to be 1 s. Furthermore, the packets generated at Node 3 and Node 1 are at the time $t = 0$ and 2.5 s, respectively.

3.2 Aloha-CA

In this section, we propose a distributed random access MAC protocol, known as Aloha-CA, for single-hop UWA networks. The protocol takes advantage of the information overheard to help avoid collisions. In addition to "sender-receiver" information that can be picked up from overheard packets, Aloha-CA requires knowledge of the propagation delays between every node pair in the network ($O(N^2)$, where N is the total number of nodes). In an underwater network with static nodes, this information can be obtained (and subsequently distributed to all nodes) during initialization by having every node in the network exchange "hello" messages. For a network with mobile nodes, if each node knows its own location (e.g., through its navigation system), the nodes can periodically broadcast their location information, enabling other nodes to estimate the relevant inter-nodal propagation delays. Furthermore, for slow moving nodes, which are quite common in underwater networks (existing AUVs move at a rate of up to 2 m/s), their positions could be predicted based on their trajectories and bearings. It is

important to note that we do not require precise information about the nodes' positions, since a ten-meter positioning error only translates into approximately 6.7 ms of error in propagation delay estimation, which is relatively small compared to the typical inter-nodal propagation delays (in the order of seconds). We will illustrate in Section 3.6.5 the effects of positioning errors on Aloha-CA's normalized throughput.

For Aloha-CA to work, each packet must have two distinct segments, namely, a *header* segment and a *data* segment. The header segment must be decodable independently[1], and must be kept short so that the information it carries can be extracted and utilized as soon as possible by those nodes that overhear it. Specifically, the information is only useful to an overhearing node from the time the header segment is decoded, till the end of the data segment, because any packet transmitted by the overhearing node after this time will never collide with the overheard packet. Hence, the header segment should contain only essential information such as the sender's ID, receiver's ID, packet size (if variable) and header checksum or error correcting bits. Upon hearing a packet, a node refrains from transmitting any packet until the packet's header segment can be decoded. If it is the intended receiver, it simply receives the packet; otherwise, it computes the busy duration of every neighboring node as a result of the overheard packet, along with indications of whether these busy states are caused by *transmitting*, *receiving*, or *overhearing* the packet. Fig. 3.2 shows an example of the database table stored at a node that overhears a packet. In this particular example, node 1 transmits a packet to node 4 at time $t = 0$ s, and the transmission time of the packet and the header are 1 s and 0.1 s, respectively. Based on the busy durations and the inter-nodal propagation delays, the node can compute the constraints

[1]Although there is some inefficiency arising from the need to decode the header separately, and the actual throughput in practical implementations may be slightly lower. However, this is a tradeoff to make so as to reduce packet collisions.

Node ID	Cause of busy state	Busy duration	Constraint on the time Node 2 can transmit
1	Transmitting	t = [0,1] s	–
3	Overhearing	t = [1.5,2.5] s	$t \geq 0.9$ s
4	Receiving	t = [2,3] s	$t \geq 1.5$ s

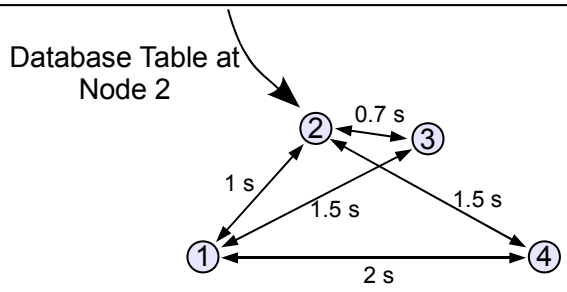

Figure 3.2: An example of the database table at a node in Aloha-CA.

caused by every other node on the time it can transmit. When it has a packet to transmit, besides making sure that it is not currently receiving a packet itself, it also checks its database table to ensure that doing so at this instant does not result in a collision at any other neighboring node. Specifically, it must not disrupt the reception of the overheard packet at the latter's receiver. Also, its own packet's intended receiver must not be busy by the time the packet arrives. In our example, to avoid interfering with node 4's reception, node 2 must not transmit before $t = 1.5$ s, regardless of its packet's intended receiver. In addition, if node 2 intends to transmit to node 3, it can only do so after $t = 0.9$ s. If any of the above constraints cannot be satisfied, the packet transmission will be postponed using a random backoff technique.

It is important to note that collisions are still possible in Aloha-CA nevertheless because the database table is maintained only based on the information that the node has already overheard and this may not be a complete representation of transmission activities in the network. When a collision[2] occurs, a node will by

[2]Note that in practice, packet collisions may be detected by underwater acoustic modems via preamble detection, especially if each source node could be assigned a different preamble.

default wait until the channel clears before it attempts to transmit a packet. We will describe in Section 3.6.3 a different variant which allows a node to transmit even before the channel clears.

In order for a node that overhears an ongoing transmission to be able to utilize the extracted information for collision avoidance, two conditions must be achieved: (1) the header must be decoded correctly and (2) the node must be within a bounded region that is determined by the relative positions of the sender and the receiver, the packet's transmission time, as well as the length of the packet's header. We shall now analyze this useful region. The notations used are shown in Table 3.1. Referring to Fig. 3.3, suppose node S sends a packet that is destined for node R at time $t = 0$, and node O overhears the packet. Assuming an error-free transmission and that the time taken to process the header information is negligible, node O will be aware of the sender-receiver information at time $t = p_{\mathrm{SO}} + T_{\mathrm{h}}$. On the other hand, the packet will be received by node R from time $t = p_{\mathrm{SR}}$ to $t = p_{\mathrm{SR}} + T$. Therefore, node O can utilize the information overheard to help avoid a collision if it refrains from transmitting any packet from $t = p_{\mathrm{SO}} + T_{\mathrm{h}}$ to $t = p_{\mathrm{SR}} + T - p_{\mathrm{OR}}$. It can be observed that the information overheard will only be useful if the following holds:

$$p_{\mathrm{SO}} + T_{\mathrm{h}} < p_{\mathrm{SR}} + T - p_{\mathrm{OR}}. \tag{3.1}$$

After some manipulation, we obtain

$$d_{\mathrm{SO}} + d_{\mathrm{OR}} < d_{\mathrm{SR}} + v(T - T_{\mathrm{h}}). \tag{3.2}$$

For a particular S-R pair, an overhearing node O can utilize the information overheard so long as the topology satisfies (3.2). Therefore, the useful region is

Table 3.1: Notations used for explaining useful region.

Notation	Meaning
d_{xy}	Distance between nodes x and y.
p_{xy}	Propagation delay between nodes x and y.
T	Transmission time of an entire packet.
T_{h}	Transmission time of a packet's header.
v	Speed of acoustic wave in underwater, assumed to be constant within the transmission range.

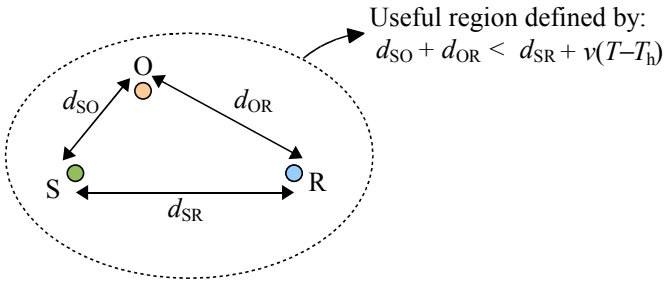

Figure 3.3: Useful region in which overhearing can help avoid collision.

defined by all possible locations of node O around the S-R pair that satisfies (3.2). By varying the position of node O while keeping the rest of (3.2) constant, we observe that the RHS of (3.2) is a constant. The boundary of the useful region is thus an ellipse, with nodes S and R being the foci. Note that for 3-D deployment, where the nodes can be placed at different depths, the useful region becomes a spheroid; Fig. 3.3 then becomes its cross-section that cuts through nodes S, R, and the particular node O being examined.

We can make two important observations from (3.2). First, we see that the longer the packet (directly proportional to T), the larger the useful region. This is intuitive because a long packet increases the likelihood that even by the time a distant node overhears the transmission, the receiver still has a long time to go before it finishes receiving the entire packet. The second observation is that,

the shorter the header (directly proportional to T_{h}), the larger the useful region. Here, if the header can be fully received sooner, then those nodes that are farther away will also be able to extract the information before it expires.

Although any node within the useful region that overhears a packet will be able to help avoid collision, the usefulness of the overheard information is strongly dependent on the time window from the moment the information is extracted, to the moment it expires. This time window can be normalized by the packet transmission time, and we shall call it the "usefulness index (U)":

$$U = (p_{\mathrm{SR}} + T - p_{\mathrm{OR}} - p_{\mathrm{SO}} - T_{\mathrm{h}})/T, \tag{3.3}$$

$$= (p_{\mathrm{SR}} - p_{\mathrm{OR}} - p_{\mathrm{SO}})/T + (T - T_{\mathrm{h}})/T. \tag{3.4}$$

For a node O that resides within the useful region, U ranges from 0 to $(T - T_{\mathrm{h}})/T$, or approximately 1 if $T_{\mathrm{h}} \ll T$. Note that U is at its maximum if node O falls directly between the sender and the receiver. For a node O that falls outside the useful region, U will be negative. By considering all node pairs acting as a sender-receiver(for an N-node network, there are the total of $j = \binom{N}{2}$ sender-receiver node pairs) with equal probability, while all other nodes acting overhearing nodes, the average U (U_{avg}) across the entire network can be obtained as a sender as

$$U_{\mathrm{avg}} = U_1 + U_2 + ... + U_j, \tag{3.5}$$

where U_j denotes the useful index of node-pair j^{th}. (3.5) can be manipulated such that we can obtain

$$U_{\mathrm{avg}} = -p/T + (T - T_{\mathrm{h}})/T, \tag{3.6}$$

where p is the average inter-nodal propagation delay. The ratio p:T, which we shall call the "pT-ratio", thus has an important significance in Aloha-CA's performance. That is, in order for Aloha-CA to work well (assuming that all nodes have the same probability to act as the sender and the receiver), the average U must be positive; this implies that the pT-ratio must be smaller than $(T - T_{\mathrm{h}})/T$, which is approximately 1.

The above analysis can also be extended to CSMA in general. Since a node exercising CSMA will refrain from transmitting its own packet so long as it overhears an ongoing transmission (i.e., without the need to decode the packet's header), its U_{avg} is simply

$$-p/T + 1, \tag{3.7}$$

meaning that CSMA can work well only if the pT-ratio is smaller than 1.

3.3 Aloha-AN

In Aloha-CA, because the data segment is sent out right after the header, a node that has successfully overheard a packet's header will not be able to overhear a subsequent packet's header until the previous overheard packet ends, due to a collision between the data segment and the new header. Thus, the node loses the opportunity to overhear more packets during this time. In addition, just like the problem encountered by NP-CSMA [55], sometimes the information may already be obsolete when the node overhears it. As we can see, the table entries in Aloha-CA are indeed a very small subset of the overall picture. Also, since Aloha-CA assumes that the header is decoded independently from the rest of the packet, this may lead to an inefficiency in error correction code which usually works better over a longer block length. To overcome such problems, we propose

in this section Aloha-AN.

The Aloha-AN is built upon the similar idea as Aloha-CA, that the information overheard by a node may sometimes help to reduce collisions. However, it goes one step further by providing the potentially useful information much earlier, and hence its name "Aloha with *advance notification*". Specifically, a small advance notification packet (NTF), which contains similar information as a normal header segment, will be transmitted first. The sender will then wait for a period of time, called the *lag-time*, before sending out the actual DATA packet. As the lag-time will be set as a network parameter, every node in the network that hears the NTF packet will know when to expect the associated DATA packet. The main advantage of having a lag-time between the NTF and the DATA packets is that it is now possible for a node to receive multiple NTF packets from its neighbors. This gives the node a bigger subset of the overall picture compared to Aloha-CA, thus allowing it to make better decisions in trying to avoid collisions.

Similarly to Aloha-CA, the Aloha-AN requires each node to maintain its own database table to monitor the busy durations of every neighboring node. Each entry in the table identifies which node is making the particular neighboring node busy, and whether the busy state is caused by *transmitting*, *receiving*, or *overhearing* a packet. Every time when a node successfully receives an NTF packet, it calculates the busy duration caused by the associated DATA packet at every node, including itself. However, before it inserts the entry into the table, it needs to check whether the associated DATA packet will cause any conflict with its own scheduled DATA packet transmissions. If there is no conflict, the entry will be inserted without further test. However, if there is a conflict, a resolution mechanism will be invoked before deciding whether the entry should still be inserted. Here, a conflict may arise if the impending DATA packet associated

with the NTF overheard appears to collide with the node's scheduled DATA packet at the intended receiver, or when the node itself is the intended receiver but it is scheduled to transmit a DATA packet during this time. The conflict-resolution mechanism checks to see which node among the two that cause the conflict will transmit first. If the current node loses, it will refrain from sending its own DATA packet by applying random backoff, and inserts the entry into the table. Otherwise, the entry will be discarded.

Whenever a node has a packet to transmit, it will check its database table to ensure that the packet does not result in a collision at any other neighboring nodes. While this test is similar to that of Aloha-CA, an important difference is that the node also needs to make sure that the new DATA packet's schedule does not overlap with the other DATA packet transmissions already scheduled in the pipeline. If the node decides not to transmit after these tests, it applies random backoff to the packet concerned[3]. In Aloha-AN, nodes are allowed to drop packets that have been backed off a specific number of times (e.g., 10). A node that has dropped a significant number of packets inherently learns that the network is busy, and will then try to alleviate the problem, such as reducing its own packet generation rate temporarily.

Although it appears that the decision on whether to transmit the DATA in both Aloha-CA and Aloha-AN depends highly on the nodes' position, this will not cause a fairness problem because packet arrivals among all nodes are distributed randomly and thus the transmission attempts are also random among all nodes.

[3]Deadlock on DATA packet transmission will not happen in Aloha-AN since the node decides to hold its DATA only if it finds that another node has transmitted the NTF prior its NTF transmission. Note that the time at which each node transmits its NTF can be computed at the other node, by using the arrival time of other nodes' NTF and the inter-nodal propagation delay between the two nodes. Although the decision on whether the node should transmit DATA is computed locally at each node and it is possible that the node misses other nodes' NTF, this again will not cause a deadlock but could affect the accuracy on the node's decision making.

The fairness problem will arise only when the same nodes repeatedly attempt to transmit their DATA at the same instance, which is unlikely to occur if the random backoff is applied after each of the unsuccessful transmission attempt.

3.4 Normalized Throughput Analysis

In this section, our aim is to understand the normalized throughput performance of different protocols analytically using a mathematical model. Specifically, we analyze the normalized throughput of the terrestrial random access MAC protocols (e.g., pure Aloha, Slotted Aloha, NP-CSMA), in the presence of high propagation delay (such as in underwater acoustic networks), as well as Aloha-HD and Aloha-CA.

In the analytical model, we assume that there are an infinite number of nodes in the network, each of which is independent of the other nodes. The aggregate packet arrivals in the network are assumed to be Poisson with a rate of λ' packets/s, and are distributed evenly among all the nodes in the network while the rate of aggregate packet attempts to transmit in the network is λ packets/s, also assumed to be Poisson. Here, note that λ' includes only those packets arriving from the network layer, while λ includes transmission attempts from both packets arriving from network layer and previously backed off packets at the MAC layer. We also define G' as the total number of packets arriving from the network layer per packet transmission time, while G (also called *offered load*) is defined as the total number of packets attempting transmission at the MAC layer per packet transmission time. Thus, we have $G' = \lambda' T$ and $G = \lambda T$ where T is the transmission time of the fixed length packet (L bits). The maximum internodal propagation delay is denoted as $p_{\max}$. Finally, the channel is assumed to be error-free so that any collision that arises is solely due to the MAC protocol's

behavior.

3.4.1 Pure Aloha and Slotted Aloha

In Pure Aloha, each node transmits its packets in an unsynchronized fashion. When a node has a packet to transmit, it switches to transmitting mode regardless of whether it is currently receiving a packet. As a result, the normalized throughput, S, of Pure Aloha is

$$S_{\text{Pure Aloha}}(\lambda) = GP_{\text{s}}, \tag{3.8}$$

where P_{s} denotes the probability of a successful transmission (for a given λ), which occurs when the received packet does not overlap with any other packet at the receiver. In order for this to happen, every other node (except the transmitter) has a period of $2T$ over which it must not have transmitted a packet. This period is called the "*silent period*". Since Pure Aloha transmits a packet when it arrives from the network layer, the time interval when there must not be any packet arrival from the network layer—commonly known as the "*vulnerable period*"—coincides exactly with the silent period. Note that, in the presence of non-negligible propagation delays, the vulnerable periods of the different nodes are staggered with respect to each other, as shown in Fig. 3.4. Nevertheless, they all have the same duration of $2T$. Hence, we can calculate P_{s} using

$$P_{\text{s}}(\lambda) = e^{-2\lambda T} = e^{-2G}. \tag{3.9}$$

By combining (3.8) and (3.9), we obtain the normalized throughput of Pure Aloha as

$$S_{\text{Pure Aloha}}(\lambda) = Ge^{-2G}. \tag{3.10}$$

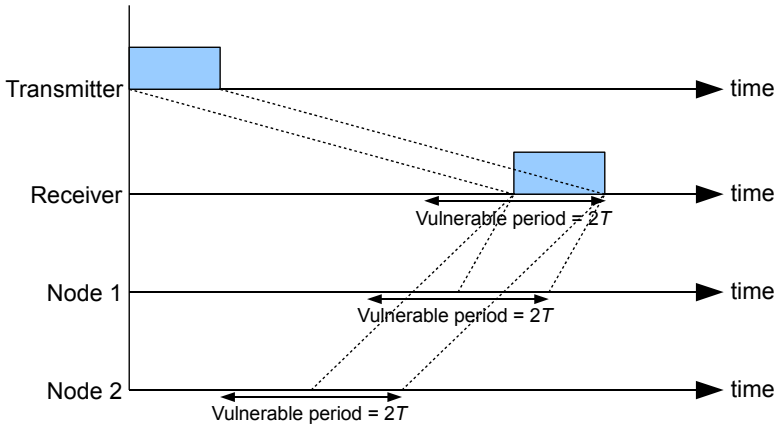

Figure 3.4: The vulnerable periods of Pure Aloha in high latency networks; they are staggered with respect to each other depending on inter-nodal distances.

Since the normalized throughput of Pure Aloha in underwater as shown in (3.10) matches exactly with its well-known normalized throughput obtained for terrestrial networks, we can see that the long propagation delays in underwater acoustic networks have no effect on the normalized throughput of Pure Aloha. This is consistent with results in [37].

Slotted Aloha introduces discrete timeslots and a node can only transmit a packet at the beginning of a timeslot. In terrestrial networks, it is generally assumed that the timeslot duration is equal to the transmission time of a single packet, i.e., T [37, 55]. Based on this assumption, both [36] and [37] have shown that Slotted Aloha degrades to Pure Aloha in high latency networks. Fig. 3.5 shows the timing diagram of Slotted Aloha in the presence of long propagation delays. Since it will be rare for a packet to be perfectly aligned with a timeslot when it arrives at the receiver, the receiver usually needs to remain silent over two timeslots (which we shall refer to as "*silent slots*") in order to avoid a collision, instead of one. This gives rise to a vulnerable period of $2T$ at the receiver in most cases. For every other node, the number of silent slots is usually two as well, and the corresponding vulnerable period is $2T$. Exceptions to this, for which

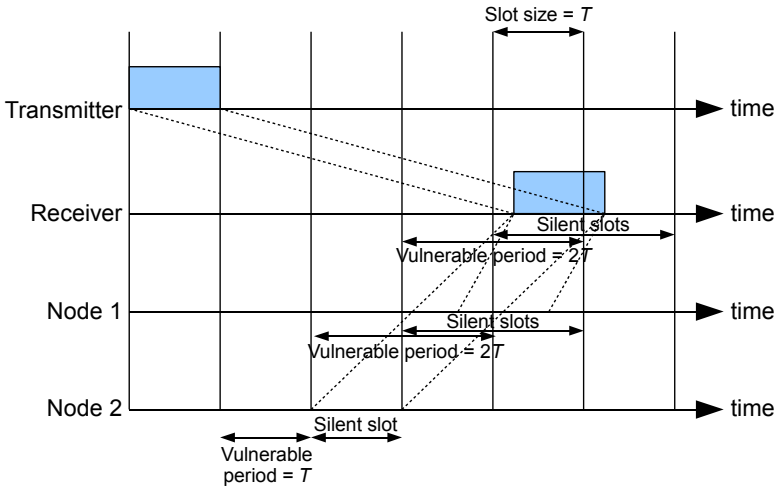

Figure 3.5: The vulnerable periods of Slotted Aloha in high latency networks.

the vulnerable period is reduced to T, are if the propagation delay between the node concerned and the receiver is equal to the propagation delay between the transmitter and the receiver, or if these two delays differ by an exact multiple of T (e.g., node 2's delay from the receiver is shorter than the transmitter-receiver delay by $2T$ in Fig. 3.5). Since these exceptions are rare, the average vulnerable period of Slotted Aloha in high latency networks is close to that of Pure Aloha (i.e., $2T$). This explains why Slotted Aloha degrades to Pure Aloha in high latency networks.

In order to reinstate the advantages of synchronization in Slotted Aloha, Syed *et al.* [37] propose the addition of guard bands to the timeslots. Specifically, the slot duration is increased to $T + \beta \cdot p_{\max}$, where $0 < \beta \leq 1$. We now examine the enhanced Slotted Aloha's normalized throughput for the special case where $\beta = 1$, which ensures that only packets that are transmitted within the same timeslot could collide at the receiver. With this, the vulnerable period at each node is reduced to the duration of a single timeslot again. However, the timeslot

duration is now $T + p_{\max}$. Therefore, the normalized throughput is given by

$$S_{\text{Slotted Aloha }(\beta=1)} = Ge^{-\lambda(T+p_{\max})} = Ge^{-G(1+a)}, \tag{3.11}$$

where $a = p_{\max}/T$. Note that the above equation is only valid for $a \leq 1$. For the case where $a > 1$, more than one packet can actually fit within a single timeslot without any overlap. This implies that the normalized throughput will be higher than (3.11). Nevertheless, it has been observed in [37] that the normalized throughput actually falls below that of Pure Aloha when $a > 1$. For the case where $a \leq 1$ over which (3.11) is valid, we can see from the equation that it outperforms Pure Aloha when $a < 1$ and is identical to Pure Aloha when $a = 1$.

3.4.2 NP-CSMA and Aloha-HD

In [55], it is shown that the normalized throughput of NP-CSMA can be expressed in terms of a and G as follows:

$$S_{\text{NP-CSMA}}(\lambda) = \frac{Ge^{-aG}}{G(1+2a) + e^{-aG}}. \tag{3.12}$$

However, it can be observed through their analysis that their approach does not hold when $a > 1$, which can be quite common in long-range underwater acoustic networks. When $a > 1$, Equation (3.12) suggests that the normalized throughput will keep decreasing until it reaches zero as $a \to \infty$, which is untrue. In the following, we show that the normalized throughput of NP-CSMA is always equal or better than that of Pure Aloha.

Let us examine the vulnerable periods at the respective nodes when a transmitter sends a packet to its receiver, using Fig. 3.6 as an illustration. As shown in the figure, the vulnerable period at the receiver is T, because NP-CSMA does not

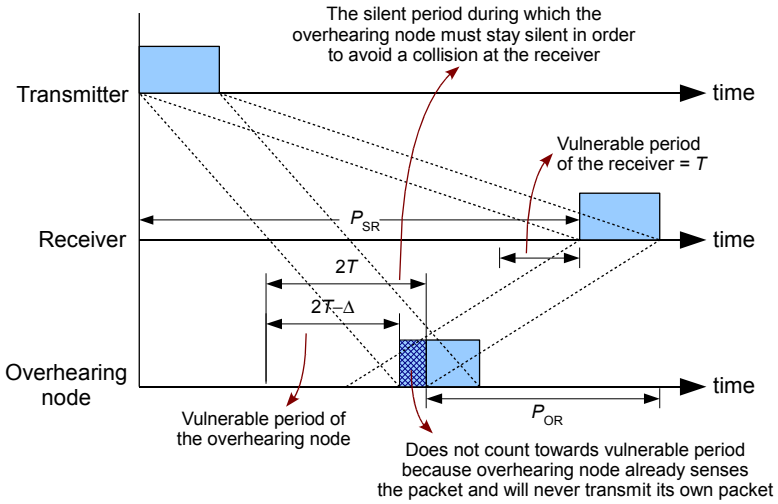

Figure 3.6: The vulnerable periods of NP-CSMA when $a > 1$.

allow a node to switch to transmit mode when it is receiving a packet. This is in contrast to Pure Aloha's vulnerable period of $2T$ at the receiver. For each of the remaining nodes that overhears the transmission (referred to as the "overhearing nodes"), its silent period during which there must not be any transmission is $2T$. However, its corresponding vulnerable period is not $2T$, but $2T - \Delta$, where $0 \leq \Delta \leq T$. The value of Δ is determined by the relative locations of the transmitter, the receiver, and the particular overhearing node. Specifically, Δ is the overlap duration between an overhearing node's silent period and the time over which it overhears the packet. This overlap duration does not count toward the vulnerable period of the overhearing node, because it already senses the packet and will not transmit its own packet. Thus, $\Delta = 0$ when there is no overlap between the silent period and the instant the packet is overheard, and $\Delta = T$ when the overheard packet falls entirely within the silent period. Since the vulnerable period of an overhearing node is as large as $2T$ only in the worst-case, and the receiver's vulnerable period is T, the normalized throughput of NP-CSMA can never be lower than that of Pure Aloha.

For Aloha-HD, the difference from Pure Aloha is that, a receiver will not switch to transmit mode once it realizes that the current packet is intended for itself. Since the receiver only finds this out after receiving the packet's entire header, its vulnerable period is hence $T + T_{\mathrm{h}}$. For all other overhearing nodes, their respective vulnerable periods remain as $2T$. Thus, Aloha-HD will always perform better than Pure Aloha, but worse than NP-CSMA:

$$S_{\text{Pure Aloha}} < S_{\text{Aloha-HD}} < S_{\text{NP-CSMA}}.$$

Also, since Aloha-HD's sole advantage over Pure Aloha is that its receiver has a vulnerable period of $T + T_{\mathrm{h}}$ instead of $2T$, we would expect its gain over Pure Aloha to be significant only when the number of nodes is small. As the number of nodes grows, its gain over Pure Aloha diminishes, and in the extreme case, it is expected that $S_{\text{Aloha-HD}} \approx S_{\text{Pure Aloha}}$. Nevertheless, it works well for underwater acoustic networks because the number of nodes deployed is usually small.

3.4.3 Aloha-CA

Since Aloha-CA works well when the pT-ratio is smaller than 1, our derivation in the following will be under this condition. Specifically, we will derive the lower-bound of Aloha-CA's normalized throughput by assuming that the inter-nodal propagation delay between any pair of nodes is always $p_{\max}$. This configuration represents the worst-case scenario because the average U (i.e., usefulness index) across the entire network as given by (3.6) will be at its minimum when $p = p_{\max}$. We also assume that $T_{\mathrm{h}} \ll p_{\max}$, because the packet's header segment is generally kept small as explained in Section 3.2.

The channel alternates between a busy state (B) and an idle state (I), with

random durations t_{B} and t_{I}, respectively. A busy period is defined as the time over which the channel is being utilized, while an idle period is defined as the time interval between two consecutive busy periods. If $\bar{t}_{\mathrm{B}}$ is the expected duration of a busy period, $\bar{t}_{\mathrm{I}}$ the expected duration of an idle period, and $\bar{t}_{\mathrm{U}}$ the expected duration within a busy period that the channel is used for transmitting data successfully, we can obtain Aloha-CA's normalized throughput lower-bound using

$$S_{\text{Aloha-CA\_LB}}(\lambda) = \frac{\bar{t}_{\mathrm{U}}}{\bar{t}_{\mathrm{B}} + \bar{t}_{\mathrm{I}}}. \tag{3.13}$$

The channel is considered to be in a busy state so long as there is at least one packet propagating in the medium. Hence, if a packet's transmission starts at time $t = 0$, the channel will remain in a busy state till at least $t = T + p_{\max}$ as a result. A busy period can be sustained over multiple packet transmissions by different nodes without encountering any collision (which we shall refer to as a "packet train" in the sequel). Although each packet's channel occupancy time is $T + p_{\max}$, the busy period resulting from m successful packet transmissions within a packet train can be less than $m(T + p_{\max})$. This is because there can be partial overlap in time between successive packets' channel occupancy (by up to $p_{\max}$), and yet these packets do not collide at their respective receivers, as explained previously in Section 3.1. In fact, so long as a node is not the intended receiver for the previous packet, it is allowed to start transmitting its own packet during the final $p_{\max}$ of the previous packet's channel occupancy.

A busy period can end either because no other node transmits a packet during the final $p_{\max}$ of the last packet's channel occupancy, or because a collision has occurred between two or more packets, causing all other nodes to wait until the channel clears. Here, we have assumed in our analysis that a collision between

multiple packets can be detected by all other nodes[4]. Thus, a busy state can be divided into two different types as illustrated in Fig. 3.7(a) and 3.7(b), labeled as B_1 and B_2 with corresponding random durations t_{B_1} and t_{B_2}, respectively. Specifically, a busy state of type B_1 consists entirely of successful packet transmissions, with m (where $m \geq 1$) packets within a packet train; it ends because no other node transmits before the m^{th} packet clears the channel. On the other hand, a busy state of type B_2 consists of k successful packet transmissions (where $k \geq 0$), but always ends with a collision. A collision occurs when one or more other nodes transmit during a packet's vulnerable period. Since we have assumed in our analysis that the inter-nodal propagation delay between any pair of nodes is always $p_{\max}$, the vulnerable period of each packet is simply the initial $p_{\max}$, beyond which all other nodes would have overheard the packet and would back off their own packets during the next $T - p_{\max}$. Note that k can be 0, in which case the entire busy state of type B_2 consists of only failed transmissions.

Let $\bar{t}_{U_1}$ and $\bar{t}_{U_2}$ respectively denote the expected durations in type B_1 and B_2 busy periods that the channel carries successful transmissions, and let $\bar{t}_{B_1}$ and $\bar{t}_{B_2}$ respectively denote the expected durations of the busy periods t_{B_1} and t_{B_2}. We can then rewrite (3.13) as

$$S_{\text{Aloha-CA\_LB}}(\lambda) = \frac{\bar{t}_{U_1} + \bar{t}_{U_2}}{\bar{t}_I + \bar{t}_{B_1} + \bar{t}_{B_2}}. \tag{3.14}$$

We note from Fig. 3.7 that the durations of t_{B_1}, t_{B_2}, t_{U_1} and t_{U_2} depend on the total number of consecutive packets that are successfully received in each of the busy periods (i.e., m for type B_1 and k for type B_2). In the following, we first derive the expressions for $\bar{t}_{U_1}$ and $\bar{t}_{B_1}$. We will then derive $\bar{t}_{U_2}$ and $\bar{t}_{B_2}$ using a

[4]Note that in practice, packet collisions may be detected by underwater acoustic modems via preamble detection, especially if each source node could be assigned a different preamble.

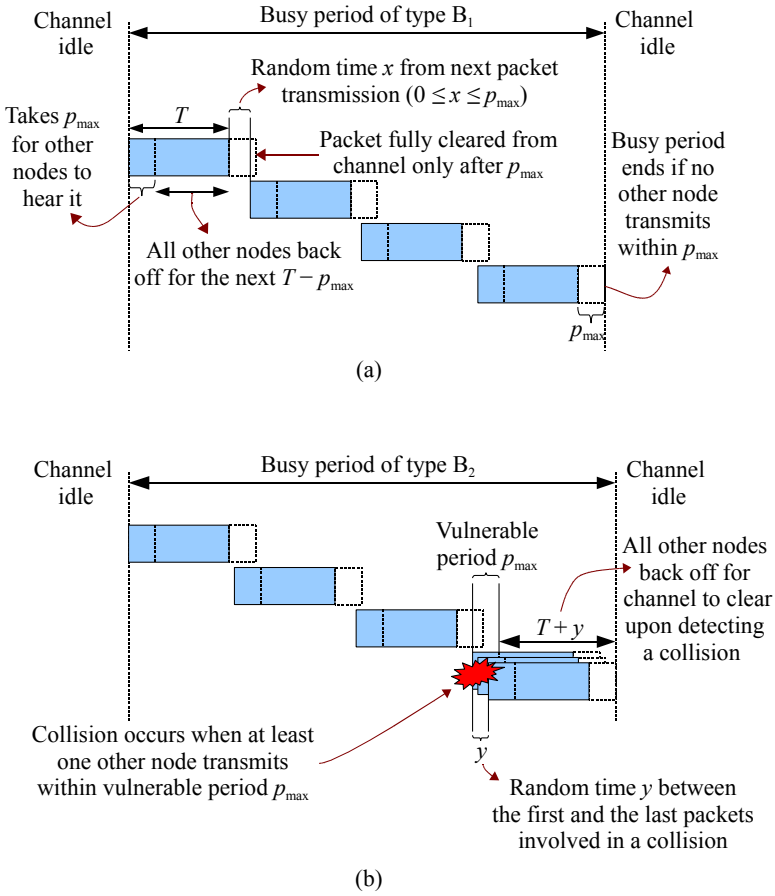

Figure 3.7: (a) Busy period of type B_1, which ends without any collision. Here, there are $m = 4$ successful packet transmissions. (b) Busy period of type B_2, which ends with a collision between at least two packets. Here, there are $k = 3$ successful packet transmissions.

similar approach. From Fig. 3.7, we obtain,

$$t_{U_1}(m) = mT, \tag{3.15}$$

$$t_{B_1}(m) = mT + \sum_{i=1}^{m-1} x_i + p_{\max}, \tag{3.16}$$

where x_i is the i^{th} instance of x, which is the random duration from the time the previous successful packet completely leaves its source node, to the time the next packet starts its transmission. We also note that, $0 \leq x \leq p_{\max}$, which is the criterion for the packet train (and hence the busy period) to continue. The

random duration x has the following distributions:

$$\begin{aligned} F(x) &= P\{\text{no arrival during } x \,|\, 0 \le x \le p_{\max}\}, \\ f(x) &= \frac{\lambda e^{-\lambda x}}{(1 - e^{-\lambda p_{\max}})}, \text{where } 0 \le x \le p_{\max}. \end{aligned} \tag{3.17}$$

Hence, the expected value of x for a given λ is,

$$\bar{x} = \int_0^{p_{\max}} x f(x)\, dx = \frac{1}{\lambda} - \frac{p_{\max} e^{-\lambda p_{\max}}}{(1 - e^{-\lambda p_{\max}})}. \tag{3.18}$$

Next, we compute the probability of a busy period being of type B_1 and contains m successful packets, denoted as P_{m,B_1}. For the case of $m = 1$ in B_1, there must be no other transmission during the vulnerable period $p_{\max}$ of the one and only packet. In addition, there must be no other packet arrival during the final $p_{\max}$ of the packet's channel occupancy in order for the busy period to end. Thus,

$$\begin{aligned} P_{m=1,\mathrm{B}_1} &= (P\{\text{no arrival during } p_{\max}\})^2 \\ &= e^{-2\lambda p_{\max}}. \end{aligned}$$

For the case of $m = 2$ in B_1, a second packet must arrive during the final $p_{\max}$ of the first packet's channel occupancy. There must also be no other arrival during both the first and the second packets' vulnerable periods. Finally, there must be no other packet arrival during the final $p_{\max}$ of the second packet in order for the busy period to end. Thus,

$$\begin{aligned} P_{m=2,\mathrm{B}_1} &= (P\{\text{no arrival during } p_{\max}\})^3 \cdot P\{x \le p_{\max}\} \\ &= e^{-3\lambda p_{\max}}(1 - e^{-\lambda p_{\max}}). \end{aligned}$$

Following the same reasoning as the above, we can obtain the general expression for P_{m,B_1} for any $m \geq 1$ as,

$$P_{m,\mathrm{B}_1} = e^{-2\lambda p_{\max}} \left[(1 - e^{-\lambda p_{\max}})e^{-\lambda p_{\max}}\right]^{(m-1)}. \tag{3.19}$$

From (3.19) and letting $A = (1 - e^{-\lambda p_{\max}})e^{-\lambda p_{\max}}$, we can obtain $\bar{t}_{\mathrm{U}_1}$ and $\bar{t}_{\mathrm{B}_1}$ as follows:

$$\bar{t}_{\mathrm{U}_1} = \sum_{m=1}^{\infty} mT \cdot P_{m,\mathrm{B}_1} = \frac{Te^{-2\lambda p_{\max}}}{(1-A)^2} \tag{3.20}$$

$$\begin{aligned} \bar{t}_{\mathrm{B}_1} &= \sum_{m=1}^{\infty} [mT + (m-1)\bar{x} + p_{\max}] \cdot P_{m,\mathrm{B}_1} \\ &= \frac{(T + \bar{x})e^{-2\lambda p_{\max}}}{(1-A)^2} + \frac{(p_{\max} - \bar{x})e^{-2\lambda p_{\max}}}{(1-A)} \end{aligned} \tag{3.21}$$

Next, we proceed to derive the expressions for $\bar{t}_{\mathrm{U}_2}$ and $\bar{t}_{\mathrm{B}_2}$. From Fig. 3.7, we obtain,

$$t_{\mathrm{U}_2}(k) = kT \tag{3.22}$$

$$t_{\mathrm{B}_2}(k) = kT + \sum_{i=1}^{k} x_i + y + T + p_{\max}, \tag{3.23}$$

where y is the random time interval between the first and the last packets involved in a collision. The variable y is constrained by the vulnerable period $p_{\max}$, beyond which all other nodes would have overheard the ongoing transmissions, and back off their own packets. Since there is no packet arrival from the time the last packet starts its transmission till the end of the vulnerable period, the distributions of y can be obtained as follows:

$$F(y) = P\{\text{no arrival during } p_{\max} - y \,|\, 0 \leq y \leq p_{\max}\},$$

$$f(y) = \frac{\lambda e^{-\lambda(p\max - y)}}{(1 - e^{-\lambda p\max})}, \text{where } 0 \leq y \leq p_{\max}. \tag{3.24}$$

Hence, the expected value of y for a given λ is

$$\bar{y} = \int_0^{p\max} y f(y)\, dy = \frac{p_{\max}}{(1 - e^{-\lambda p\max})} - \frac{1}{\lambda}. \tag{3.25}$$

In order to derive $\bar{t}_{\mathrm{U}_2}$ and $\bar{t}_{\mathrm{B}_2}$, we first need to derive the probability of a busy period being of type B_2 and contains k successful packets (denoted as P_{k,B_2}), which is presented in Appendix A. We then obtain,

$$\bar{t}_{\mathrm{U}_2} = \sum_{k=0}^{\infty} kT \cdot P_{k,\mathrm{B}_2} = \frac{T(1 - e^{-\lambda p\max})A}{(1 - A)^2} \tag{3.26}$$

$$\begin{aligned} \bar{t}_{\mathrm{B}_2} &= \sum_{k=0}^{\infty} [k(T + \bar{x}) + \bar{y} + T + p_{\max}] \cdot P_{k,\mathrm{B}_2} \\ &= \frac{(T + \bar{x})(1 - e^{-\lambda p\max})A}{(1 - A)^2} + \\ &\quad \frac{(\bar{y} + T + p_{\max})(1 - e^{-\lambda p\max})}{(1 - A)}. \end{aligned} \tag{3.27}$$

Finally, the average idle period is simply equal to the average inter-arrival time of packets, which is

$$\bar{t}_{\mathrm{I}} = \frac{1}{\lambda}. \tag{3.28}$$

By substituting (3.20), (3.21), (3.26), (3.27) and (3.28) into (3.14), we can obtain Aloha-CA's normalized throughput lower-bound.

The above analysis expresses Aloha-CA's normalized throughput as a function of λ, which is the aggregate packet arrival rate seen at the MAC layer. Since λ includes the effects of a MAC protocol's backoff strategy, it is often more useful to express normalized throughput in terms of λ', the arrival rate of packets from the network layer; this yields a fairer comparison between different MAC

protocols' performance. For a given λ', we can obtain λ by using the following recursive relationship,

$$\lambda'_{j+1} = \lambda' + \lambda'_j P_{\text{bo}}(\lambda'_j), \text{for } j \geq 1, \tag{3.29}$$

where $\lambda'_1 = \lambda'$, $\lambda = \lambda'_\infty$, and $P_{\text{bo}}(\lambda'_j)$ is the probability that a packet upon arrival at the MAC layer gets backed off when the aggregate arrival rate is λ'_j. From (3.14) and (3.29), $S_{\text{Aloha-CA\_LB}}(\lambda')$ can be obtained. Note that (3.29) typically converges after a small number of iterations, such that $\lambda'_{j+1} \approx \lambda'_j$.

In order to obtain λ using (3.29), we need to evaluate $P_{\text{bo}}(\lambda'_j)$. Specifically,

$$\begin{aligned} P_{\text{bo}}(\lambda'_j) &= P_{\text{bo}|\text{B}_1} \cdot P_{\text{B}_1} + P_{\text{bo}|\text{B}_2} \cdot P_{\text{B}_2} \\ &= P_{\text{B}_1} \left[\sum_{m=1}^{\infty} P_{\text{bo}|(m,\text{B}_1)} \cdot P_{m|\text{B}_1} \right] + \\ &\quad P_{\text{B}_2} \left[\sum_{k=0}^{\infty} P_{\text{bo}|(k,\text{B}_2)} \cdot P_{k|\text{B}_2} \right], \end{aligned} \tag{3.30}$$

where the notations used are summarized in Table 3.2, and (λ'_j) has been dropped from the RHS terms for simplicity. The derivations of these probabilities are presented in Appendix B. We will also show in Section 3.6.1 that our analytical normalized throughput lower-bound for Aloha-CA as a function of G' (where $G' = \lambda' T$) matches our simulation results very well.

3.5 Simulation Model

In this section, we describe the simulations performed to verify our findings from the previous section, as well as to investigate the performance of the Aloha-CA protocol relative to the other single-hop MAC protocols discussed above. We

Table 3.2: Notations used for evaluating backoff probability

Notation	Description
P_{B_i}	Probability that a packet, upon arrival, finds itself in a type B_i busy period
$P_{\mathrm{bo}\|\mathrm{B}_i}$	Probability that a packet is backed off, given that it arrives in a type B_i busy period
$P_{\mathrm{bo}\|(n,\mathrm{B}_i)}$	Probability that a packet is backed off, given that it arrives in a type B_i busy period with n successful packets
$P_{n\|\mathrm{B}_i}$	Probability that there are n successful packets in the busy period, given that it is a type B_i busy period

consider two single-hop network topologies, namely, a small network with only four nodes, and a large network with 100 nodes. The results obtained from the 100-node network approximate what one would expect from a network with infinite number of nodes, which corresponds to the assumption made in our earlier analysis. On the other hand, the 4-node network resembles a more practical scenario whereby underwater sensor nodes are typically deployed in a sparse manner due to the high cost of each node. All the nodes are assumed to be static and randomly deployed with a uniform distribution, within a deployment area of 1000 m by 1000 m. Although the assumed topologies are two-dimensional, all the MAC protocols studied here are expected to exhibit similar behavior when they are applied to three-dimensional network topologies with similar inter-nodal distances. The average inter-nodal distances are approximately 474 m and 539 m while the maximum inter-nodal distances are 604 m and 1292 m, in the 4-node and 100-node networks, respectively. All the nodes are assumed to be equipped with half-duplex and omnidirectional modems, with a fixed data rate of 2400 bps. The speed of underwater acoustic waves is assumed to be fixed at 1500 m/s. The packet generation at each node is assumed to be Poisson with rate λ' packets/s, and each packet's intended receiver is randomly chosen with equal probability.

We have also used a number of different packet lengths to examine their effects on the MAC protocols. Regardless of the packet length (L), we assume that each packet includes a 32-bit header. Since we focus on single-hop networks in this chapter, we assume that all nodes are within each other's range. Also, we assume that the channel is error-free, so that packet losses are only caused by collisions at the receivers. Finally, we do not consider any kind of packet retransmission when a collision occurs.

For each scheme that we have simulated, we are interested in its normalized throughput, S, versus the load from the network layer, G'. Specifically, we compute S as follows:

$$S = \frac{\text{No. of packets received} \times \text{Packet length } (L)}{\text{Data rate} \times \text{Simulation duration}} \tag{3.31}$$

3.6 Simulation Results

3.6.1 Verifying Aloha-CA's Analytical Normalized Throughput Lower-bound

In our earlier analysis of Aloha-CA, we have derived its normalized throughput lower-bound by assuming that the inter-nodal propagation delay between any pair of nodes is always $p_{\max}$. Here, we verify our derivation by comparing the analytical results with simulation results for the 100-node network. In this part of our simulations, we artificially set the inter-nodal propagation delay between any pair of nodes to be $p_{\max}$, where $p_{\max} = 0.861$ s (corresponding to 1292 m). Fig. 3.8 shows the results obtained for different packet lengths. As can be seen, the simulation results match the analytical results very well, thus validating the accuracy of our previous analysis.

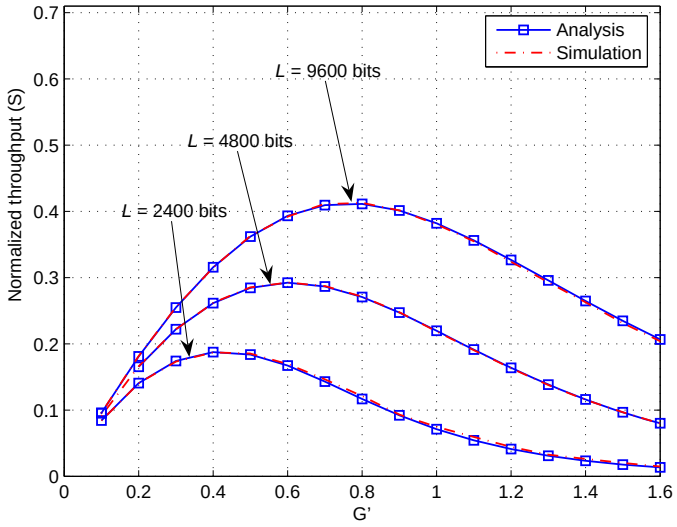

Figure 3.8: Verification of Aloha-CA's analytical normalized throughput lower-bound via comparison with simulation results. Note that all inter-nodal propagation delays are set to $p_{\max}$ in the simulation network.

We note that, in a realistic network, many of the node pairs would have inter-nodal propagation delays that are much smaller than $p_{\max}$. Therefore, we would expect the analytical normalized throughput lower-bound to be loose compared to the average normalized throughput. In an attempt to better approximate Aloha-CA's average normalized throughput, we consider substituting the value of $p_{\max}$ in the analytical results with the value of average inter-nodal propagation delay, p, where $p = 0.359$ s (corresponding to 539 m). Fig. 3.9 compares the resulting analytical results after the substitution of $p_{\max}$ with p against the simulation results from the same 100-node network with the actual inter-nodal propagation delays retained. Interestingly, we see that the results are quite close to each other, especially when the load G' is not too large. This shows that our earlier analysis of Aloha-CA's normalized throughput lower-bound could also be used to approximate Aloha-CA's average normalized throughput by substituting $p_{\max}$ with p.

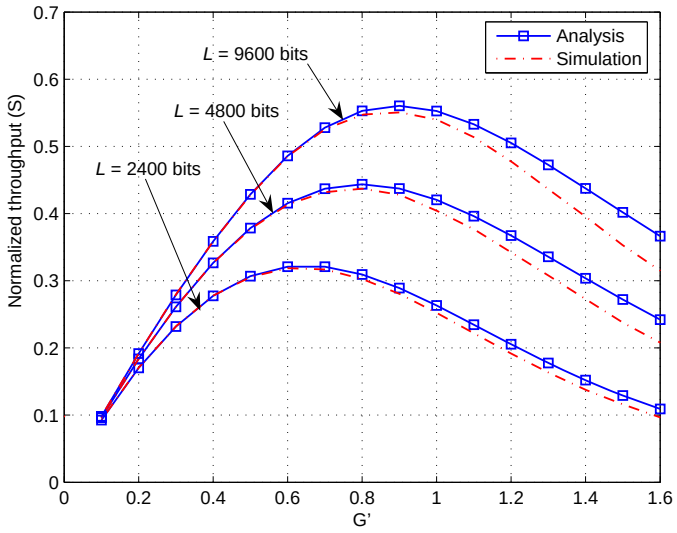

Figure 3.9: Approximating Aloha-CA's average normalized throughput by substituting $p_{\max}$ with p in the analytical normalized throughput lower-bound.

3.6.2 Performance of Aloha-HD and NP-CSMA

Fig. 3.10 compares the normalized throughput of Pure Aloha, Aloha-HD, and NP-CSMA in underwater networks when the packet size is 2400 bits. As can be seen, the results satisfy our earlier analysis that their relative normalized throughput conforms to the relationship, $S_{\text{Pure Aloha}} < S_{\text{Aloha-HD}} < S_{\text{NP-CSMA}}$. From the figure, we also observe that Aloha-HD's improvement over Pure Aloha becomes insignificant for the 100-node network. This agrees with our earlier analysis in Section 3.4.2 that Aloha-HD's normalized throughput diminishes to that of Pure Aloha as the number of nodes becomes large.

Another observation that can be made from Fig. 3.10 is that the normalized throughput of each scheme for the 4-node network is higher than that of the 100-node network. This can be explained as follows. Suppose we have a finite network with N nodes. When a node transmits a packet, it cannot transmit another packet concurrently that will collide with the prior packet. Consequently, the aggregate load that could result in a collision can only come from the remaining $N-1$ nodes.

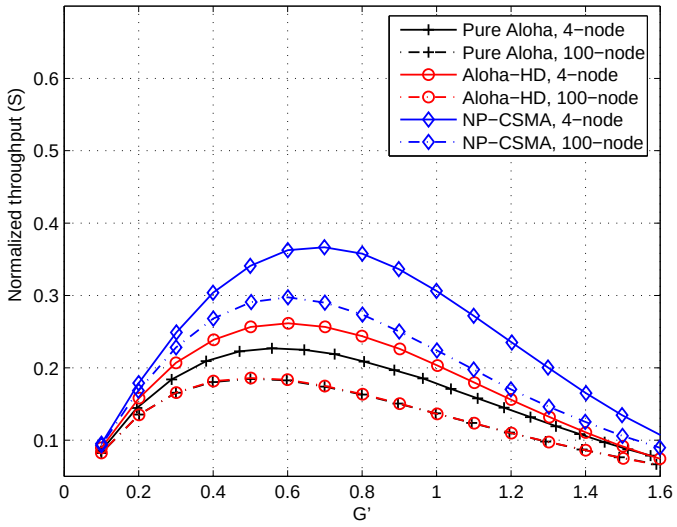

Figure 3.10: Comparing the normalized throughput of Pure Aloha, Aloha-HD, and NP-CSMA when the packet size is 2400 bits.

With $1/N$ of the network load not contributing towards collision probability, the smaller the value of N, the lower would be the collision probability, and hence the larger the normalized throughput.

In Fig. 3.11, we examine the effects of different packet lengths on NP-CSMA for the 4-node network. As can be seen, increasing the packet length from 200 bits to 800 bits does not have much effect on the normalized throughput. However, for longer packets, we see that increasing the packet length also leads to a corresponding increase in the normalized throughput. In order to understand this, we focus our attention on the pT-ratio, previously defined in Section 3.2 as the ratio of the average inter-nodal propagation delay to the packet's transmission time. Fig. 3.12 shows the maximum possible normalized throughput corresponding to different pT-ratios for both NP-CSMA and Aloha-CA (to be discussed next). As can be seen, when the pT-ratio is less than 1, the maximum normalized throughput increases dramatically as the pT-ratio decreases. On the other hand, when the pT-ratio is greater than 1, the maximum normalized throughput

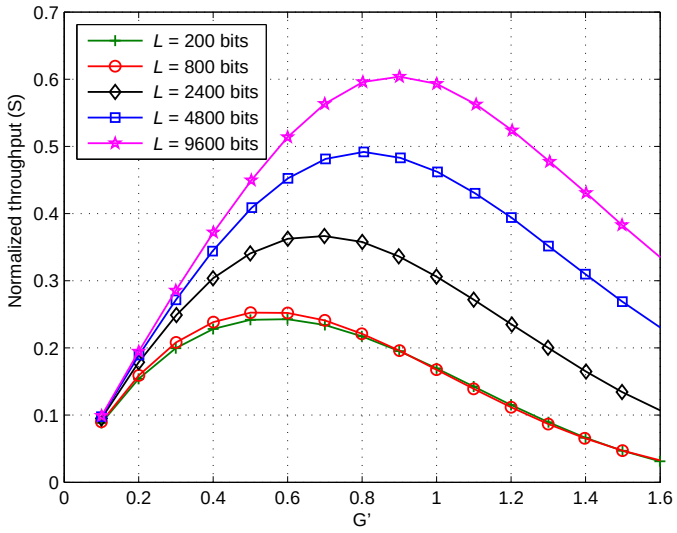

Figure 3.11: Effects of different packet lengths on NP-CSMA for 4-node network.

stays low. In our simulation model used for generating Fig. 3.11, since the data rate is 2400 bps and the average inter-nodal distance in the 4-node network is around 474 m, the pT-ratio is equal to 1 when the packet length is approximately 758 bits. This explains why we have observed in Fig. 3.11 that NP-CSMA's normalized throughput starts improving significantly when the packet lengths are longer than 800 bits. The above observation also agrees with (3.7) that the pT-ratio threshold for NP-CSMA to perform well is approximately 1.

Another observation that we can make from Fig. 3.12 is that, even for very large values of pT-ratio, the normalized throughput of NP-CSMA is still better than that of Pure Aloha. This also agrees with our analysis in Section 3.4.2 that the normalized throughput of NP-CSMA can never be lower than that of Pure Aloha.

Although it may seem from Fig. 3.11 that the normalized throughput of NP-CSMA can continue to grow with longer packets, this is not practical because longer packets will also be more susceptible to corruptions due to bit errors. In

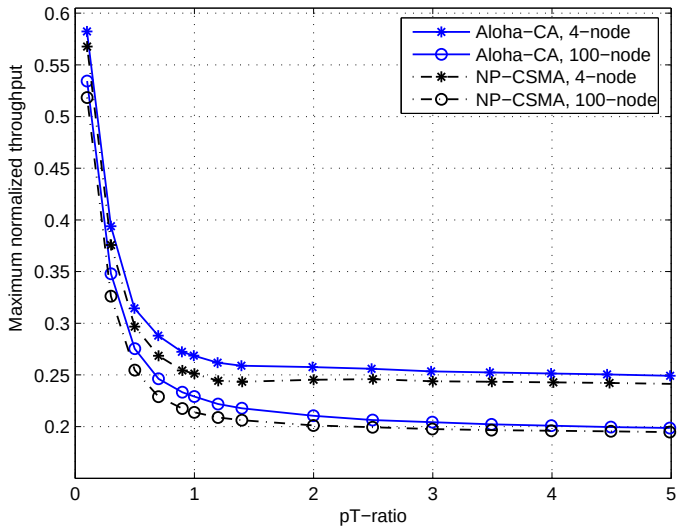

Figure 3.12: Maximum normalized throughput vs. pT-ratio for NP-CSMA and Aloha-CA. Note that "maximum normalized throughput" refers to the peak point in the respective normalized throughput vs. G' plots, and the peaks may occur at different G' for different pT-ratios.

our simulations, we have assumed that the channel is error-free. If bit errors were introduced, we expect the normalized throughput to grow with increasing packet length only up to a certain level, beyond which the normalized throughput will drop due to higher likelihood of packets being corrupted by channel errors.

3.6.3 Performance of Aloha-CA and Its Enhanced Variant

We now examine the normalized throughput of Aloha-CA. As can be seen from Fig. 3.12, the maximum normalized throughput of Aloha-CA is consistently better than that of NP-CSMA across all pT-ratios. Also, its maximum normalized throughput increases dramatically as the pT-ratio decreases below 1. This agrees with (3.6) that the pT-ratio threshold for Aloha-CA to perform well is approximately 1.

In Fig. 3.13, we take a closer look at the relative performance between

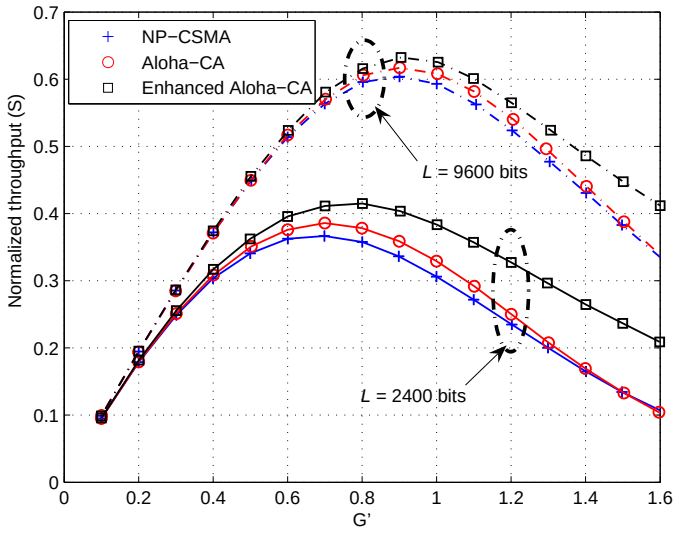

Figure 3.13: Comparing the normalized throughput of NP-CSMA, Aloha-CA, and an enhanced Aloha-CA scheme for the 4-node network.

Aloha-CA and NP-CSMA for the 4-node network across different loads. As can be seen, Aloha-CA outperforms NP-CSMA slightly when the load G' is in the moderate range, but their normalized throughputs converge when G' is large. The convergence at high load can be explained as follows. When the load is high, there is a higher tendency for packets to collide not only at the intended receiver, but also at other overhearing nodes. Since the overhearing nodes cannot extract the header information from the collided packets, they will not have any useful information to help them decide whether they should avoid transmitting their own packets. The default mechanism of Aloha-CA requires such nodes to refrain from transmitting any packet until the corrupted packets can no longer be sensed. This behavior mimics that of NP-CSMA's carrier sensing, and thus explains why the normalized throughputs of Aloha-CA and NP-CSMA converge at high load.

In an attempt to boost Aloha-CA's normalized throughput, we consider a possible variant, which we shall call "Enhanced Aloha-CA". Recall from Section 3.2 that when a collision occurs, a node will by default wait until the channel

clears before it attempts to transmit a packet. In Enhanced Aloha-CA, a node is now allowed to transmit its own packet when it realizes that the packet it is currently overhearing/receiving has been corrupted by another incoming packet. Note that its own transmission is still subject to the constraints imposed by its database table entries that doing so does not result in a collision at some other nodes. As can be seen in Fig. 3.13, the Enhanced Aloha-CA has higher normalized throughput than the original Aloha-CA, especially in the high load region.

It should be noted that both variants of Aloha-CA perform better than NP-CSMA because they allow a node to transmit a packet even when it overhears another packet if it determines that doing so will not result in a collision at either packet's intended receiver. This is based on the information the node collects from the overheard packet, which in turn leverages on the long propagation delay in underwater acoustic networks. Although the performance gains of both variants over NP-CSMA diminishes as the packet length becomes longer, the packet lengths used in underwater acoustic networks are typically short due to the high bit error rates.

3.6.4 Performance of Aloha-AN

In the following, we study the performance of Aloha-AN. When comparing Fig. 3.13 with Fig. 3.14, we see that Aloha-AN offers even better results than Enhanced Aloha-CA. The normalized throughput is now much higher, along with better stability in the high load region. At the low load region when the offered load ranges from 0.1 to 0.4 (look at the plots of lag-time = 0.403 s), the normalized throughput is almost as high as it could get. On the other hand, in the high load region, even when the offered load goes above 1, the normalized throughput does not fall steeply. Note that the normalized throughput will

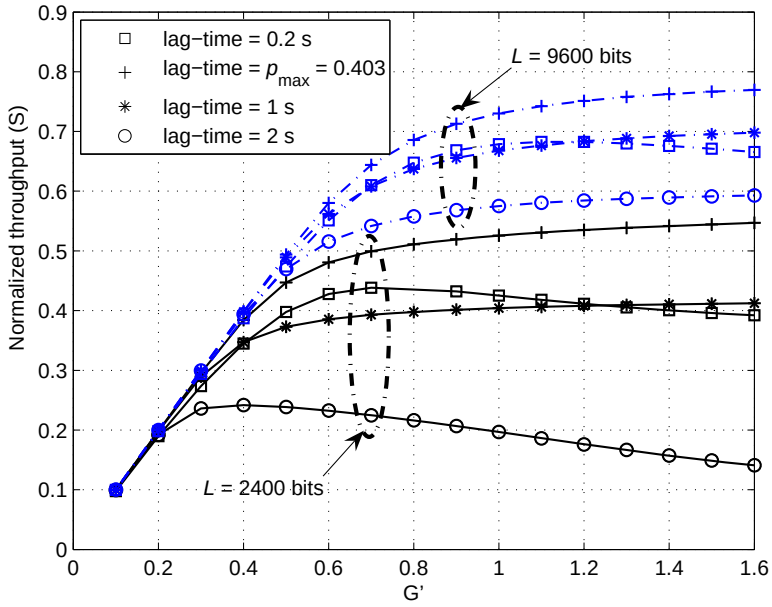

Figure 3.14: Normalized throughput of Aloha-AN in the 4-node network.

always be smaller than the offered load for our Aloha-AN, because we do not count the NTF packets towards the normalized throughput, as they are overhead incurred.

In Fig. 3.14, we also observe that different choices of *lag-time* will give us significantly different normalized throughput and stability levels. Now, let us consider the plots of $L = 2400$ bits in Fig. 3.14. When the lag time is too small (lag-time = 0.2 s), the normalized throughput is low and decreases at high load because the nodes do not have sufficient windows to acquire enough NTF packets from their neighbors, which subsequently degrade their ability to make informed decisions about packet transmissions. This in turn leads to a higher number of collisions. On the other hand, when the lag time is too long (lag-time = 2 s), the normalized throughput will again become lower, because each node will spend a lot of time listening for NTF packets, such that the channel bandwidth becomes under-utilized. When lag-time is set to 1 s, the normalized throughput is stable but still lower than that in the case of lag-time equals to

$p_{\max}$. We observe that a lag-time of $p_{\max}$ provides the best normalized throughput for both packet lengths.

Our Aloha-AN can be viewed as an extension of our Aloha-CA. Since the Aloha-CA works well only if the packet is long enough, in the Aloha-AN, we make the packet artificially "long" by introducing a lag time between the NTF packet (like a header segment) and the remaining DATA packet. This not only improves normalized throughput performance, but is also less susceptible to bit errors compared to a truly long packet. Ideally, the Aloha-AN's DATA packet size should be as large as possible to improve normalized throughput performance. At the minimum, its transmission time should be at least equal to the lag time, else the normalized throughput will be very low. It is also important to note that, by allowing packets to be dropped if they have been repeatedly backed off, the protocol (when the suitable lag-time is chosen) is very stable even in the face of high traffic load.

We now give some insights about how the lag time can be picked. Ideally, it should be just long enough for a source node to be able to hear the NTF packets from all its neighbors whose transmissions of the associated DATA packets could potentially collide with the source node's own scheduled DATA packet. This helps the source node to better determine whether it needs to cancel the transmission of its scheduled DATA packet. Thus, it appears that the lag time should be set as the maximum propagation delay in the network. In a real implementation, this implies that users should select the packet length to be longer than the parameter $p_{\max}$. If the packet length is fixed and smaller than the value of $p_{\max}$, the series of packets can be transmitted in a form of a packet train, in order to solve the problem.

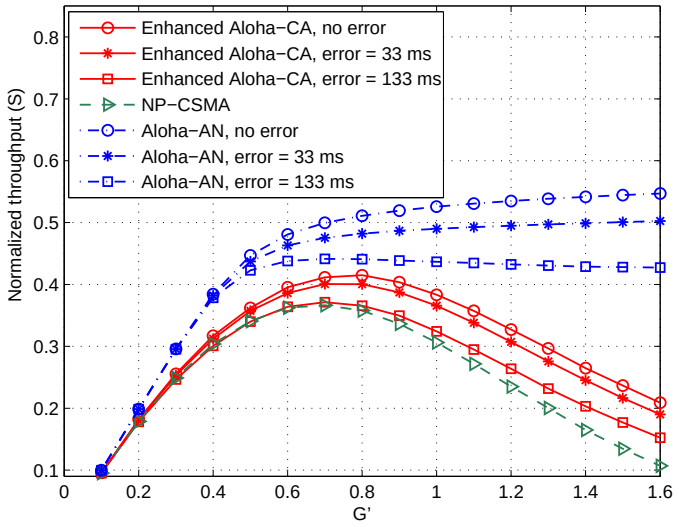

Figure 3.15: Normalized throughput of Enhanced Aloha-CA and Aloha-AN with fixed delay estimation errors of 0, 33 ms and 133 ms for the 4-node network with 2400-bit packets.

3.6.5 Effects of Inter-nodal Propagation Delay Estimation Errors

Since both Aloha-CA and Aloha-AN require the knowledge of inter-nodal propagation delays to help avoid collision, their performances are dependent on the accuracy of the delay estimates. In static networks, the inter-nodal propagation delay between any pair of nodes could be estimated reasonably well by measuring the round-trip time between them. It should be noted that even when the nodes are anchored, there could still be some sway movements due to underwater currents, and hence the delay estimations can never be perfectly accurate. For networks with mobile nodes, the round-trip delay measurement technique could be augmented with readings from the individual nodes' navigation systems, as well as mobility prediction algorithms; nevertheless, the delay estimation errors in mobile networks are usually more significant than those of static networks.

We now examine the effects of inter-nodal propagation delay estimation errors on the performance of the Enhanced Aloha-CA and Aloha-AN. Fig. 3.15 shows their normalized throughput for the 4-node network with 2400-bit packet length, for the cases of (i) no error, (ii) delay estimation error of 33 ms (50 m error in distance estimation), and (iii) delay estimation error of 133 ms (200 m error in distance estimation).

As can be seen, although the normalized throughput of Enhanced Aloha-CA deteriorates when the error increases, it can tolerate significant amount of delay estimation errors (133 ms in this case) and yet still outperform the NP-CSMA scheme.

Similarly to Enhanced Aloha-CA, the normalized throughput of Aloha-AN also decreases as the error increases. For the same error, we can observe that the amount of normalized throughput decrement in Aloha-AN is higher than that of Enhanced Aloha-CA. However, Fig. 3.15 shows that even for the error of as large as 133 ms (200 m error in distance estimation), Aloha-AN is still able to maintain stable normalized throughput and perform better than Enhanced Aloha-CA.

3.7 Conclusion

In this chapter, we have studied the normalized throughput of different Aloha-based MAC protocol variants in single-hop UWA networks via both theoretical analysis and simulations. We have also demonstrated that with the knowledge of inter-nodal propagation delays, a node could help avoid collisions in a distributed manner by utilizing the "sender-receiver" information that it picks up from overheard packets, provided it is within a bounded region defined by the

relative positions of the sender and the receiver, as well as the packet's transmission time. This opportunity only arises in a long propagation delay environment such as underwater acoustic networks. Based on this strategy, we propose two Aloha-based random access MAC protocol, namely, Aloha-CA and Aloha-AN, for UWA networks. Like pure Aloha, both protocols are simple, and do not require any handshaking or clock synchronization.

To understand the performance of Aloha-CA, we have provided a theoretical analysis of its normalized throughput lower-bound, which is verified using discrete event-driven simulations. Our simulations also show that Aloha-CA, and especially its enhanced variant, is able to outperform NP-CSMA and other Aloha-based variants. The enhanced Aloha-CA is able to maintain its advantage over NP-CSMA even in the presence of considerable errors in the inter-nodal propagation delay estimates.

The performance of Aloha-AN has been studied through simulations which reveal that Aloha-AN outperforms both the Aloha-CA and the enhanced Aloha-CA significantly, when a suitable lag-time is chosen. Specifically, the lag time should be set as the maximum propagation delay in the network while keeping the packet's transmission time longer than the maximum propagation delay.

Between our two protocols, Aloha-CA is simpler and more scalable, as it only needs a small amount of memory, and does not rely on additional control messages. Aloha-AN, on the other hand, requires the use of additional NTF packets, which serve as advance notification to neighboring nodes, so that they can avoid transmitting packets that could result in collisions. The Aloha-AN needs to collect and store more information, therefore it requires more resources than Aloha-CA. Due to the need to select a suitable lag time for a given network setting, the scheme is less scalable as it needs to check if its lag time is still

appropriate whenever there are any significant topology changes. However, the extra cost allows the Aloha-AN to achieve much better normalized throughput and collision avoidance. In addition, Aloha-AN is more suitable than Aloha-CA for a system that the decoding time of the header packet (NTF packet) is substantial. This is because the lag-time in Aloha-AN can be adjusted according to the length of the decoding time, which is deterministic [56], such that the overheard information is still useful in collision avoidance.

Chapter 4

Handshaking-based MAC protocols for UWA networks

4.1 Introduction

Although the normalized throughput of random access MAC protocols in a single-hop UWA network can be improved by simply implementing a collision avoidance mechanism, as being done in Aloha-CA and Aloha-AN in the last chapter, this enhancement is diminished when the protocols are to be implemented in a multi-hop network. In such a network, it is widely known that hidden and exposed terminal problems are the main causes of low normalized throughput. The hidden-terminal problem causes high collision rate, while the exposed-terminal problem causes a node to become over-conservative when transmitting packets. These problems tend to result in under-utilization of the channel.

In terrestrial wireless networks, there are two main approaches in MAC protocol designs to alleviate the abovementioned problems. The first approach is the use of a busy signal to inform the hidden node about an ongoing transmission.

Upon hearing the busy signal, the hidden node will avoid accessing the channel until the busy signal ends. This could help resolve the hidden and exposed terminal problems to some extent, depending on the variation of those techniques that fall within this category. Examples of such protocols are BTMA [57], RI-BTMA [58], and DBTMA [59]. In order to utilize the busy signal approach, every node needs to be equipped with more than one transceiver. This may not be feasible for some applications such as sensor networks, in which cost is a major concern. The second approach, on the other hand, uses a handshaking mechanism to reduce the hidden and exposed terminal problems without requiring any additional hardware. "Handshaking" refers to the exchange of multiple small control packets prior to transmitting a longer data packet. This approach has been studied extensively, and many ad-hoc MAC protocols are designed based on this idea. MACA [23] was the first MAC protocol that uses the handshaking mechanism. Some other examples of handshaking-based MAC protocols are MACAW [30], MACA-BI [60] and the widely used IEEE 802.11 protocol.

For UWA networks, the narrow available bandwidth implies that it may not be practical to set aside a separate frequency band for transmitting busy signals. The long propagation delay, on the other hand, makes it very expensive to transmit multiple control packets (e.g., RTS/CTS frames) before *every* data packet transmission. In fact, both [19] and [20] have shown that such a technique offers a lower normalized throughput than the well-known Aloha protocol when applied in UWA networks.

The high latency overhead introduced by the control packets of handshaking-based protocols implies that the channel's utilization may be improved if multiple data packets in the form of a packet train can be transmitted for every set of handshaking. This is one of the key motivations underlying our proposed MAC

protocol, which we call the "Receiver-initiated Packet Train" (RIPT) protocol. While the RIPT protocol is also handshaking-based, the key difference here is that the reservations are receiver-initiated. As will be explained in Section 4.2 later on, the use of receiver-initiated reservations is crucial in reducing data packet collisions in the presence of long propagation delays. Another novel concept of the RIPT protocol is that the "packet train" that arrives at the receiver after each set of handshaking is actually formed by transmissions from multiple neighboring nodes. This is built on the assumption that every node knows the inter-nodal propagation delay between itself and each of its immediate neighbors, so that it can schedule its transmissions accordingly to ensure that a packet train can be formed at the receiver. This design results in high normalized throughput, as well as low data packet collisions.

The remainder of this chapter is organized as follows. We present in Section 4.2 the RIPT protocol that we propose for UWA networks with distributed topology. Section 4.3 and Section 4.4 describe the simulations that were carried out to compare the performance of the proposed scheme with several others. In Section 4.5, we provide further insights into our proposed scheme, and finally, we give our conclusions in Section 4.6.

4.2 RIPT

4.2.1 Overview of RIPT

Although the RTS/CTS mechanism is widely used for alleviating the hidden and exposed terminal problems in terrestrial multi-hop networks, they suffer from two main drawbacks when they are applied in UWA networks. Firstly, the need for at least one full round-trip exchange of control packets prior to sending every

data packet introduces considerable latency, due to the long propagation delay. This leads to under-utilization of the channel, and low normalized throughput. Secondly, the long propagation delay also seriously impacts the ability of the RTS/CTS handshake mechanism to resolve the hidden terminal problem, because it now takes much longer for a node to receive RTS and CTS packets from its neighbors, which extends the vulnerable period. This leads to higher collision rate, and again, low normalized throughput. For the first drawback, some previously proposed protocols [26, 39] attempt to increase the normalized throughput by sending a train of packets after each successful handshake. Note that the packet train concept has also been proposed for terrestrial wireless networks in [61]. For the second drawback, it appears that receiver-initiated reservations are better at avoiding collisions in the presence of long propagation delay, since the receiver has accurate information on its own current state.

The important observations above lead us to propose the RIPT protocol. While it also seeks to alleviate the hidden and exposed terminal problems through a handshaking mechanism, it does a better job at avoiding collisions by utilizing receiver-initiated reservations. In order to improve channel efficiency, we propose the idea of "*multiple-node polling*", in which multiple nodes are allowed to transmit data packets to a single receiver within each round of handshake. By assuming that every node knows the propagation delays between itself and its neighboring nodes, the transmissions can be scheduled in such a way that the data packets will be received by the receiver in the form of a packet train. This is different from the packet train approach in [26, 61, 39], in which the train of packets that a receiver receives are sent by a single transmitter.

We now give more insights into the RIPT protocol's design. As discussed in [37], the performance of high latency networks, such as underwater networks, is

affected by both space and time uncertainty. The space uncertainty is caused by the nodes' locations, which result in different propagation delays, while the time uncertainty is caused by the randomness of packet arrivals. The RIPT overcomes the space uncertainty by carefully scheduling the data packet transmissions to avoid collision at the receiver, using the knowledge of inter-nodal propagation delays. For the time uncertainty, we realize that those techniques widely used in terrestrial MAC (e.g., synchronizing transmission, sensing channel [37]) are not appropriate in underwater since they only remove the uncertainty at the transmitter, but not at the receiver. Moreover, unlike other transmitter-initiated protocols that encounter two types of data packet collision, namely, "transmit-receive collision" and "receive-receive collision", a receiver-initiated approach only experiences receive-receive collision. This is because a receiver knows exactly when the current handshake will end, and how long it should defer its own transmission in order to avoid a transmit-receive collision. The above reasonings explain why we have adopted a receiver-initiated approach in designing the RIPT protocol. Note that transmit-receive collision refers to the scenario whereby an incoming packet arrives at a node while it is transmitting. In this case, the incoming packet will not be heard. On the other hand, receive-receive collision occurs when two or more packets arrive at a receiver simultaneously, causing all packets to be corrupted.

Instead of the typical 3-way (RTS/CTS/DATA) handshake found in protocols such as MACA, the RIPT protocol utilizes a receiver-initiated 4-way handshake (RTR/SIZE/ORDER/DATA). The RTR (Ready-To-Receive) packet serves to inform all of the initiating receiver's neighbors that the receiver node is ready to act as a receiver for a certain duration of time. A series of SIZE packets will then be sent by the receiver's neighbors to inform the receiver about the number

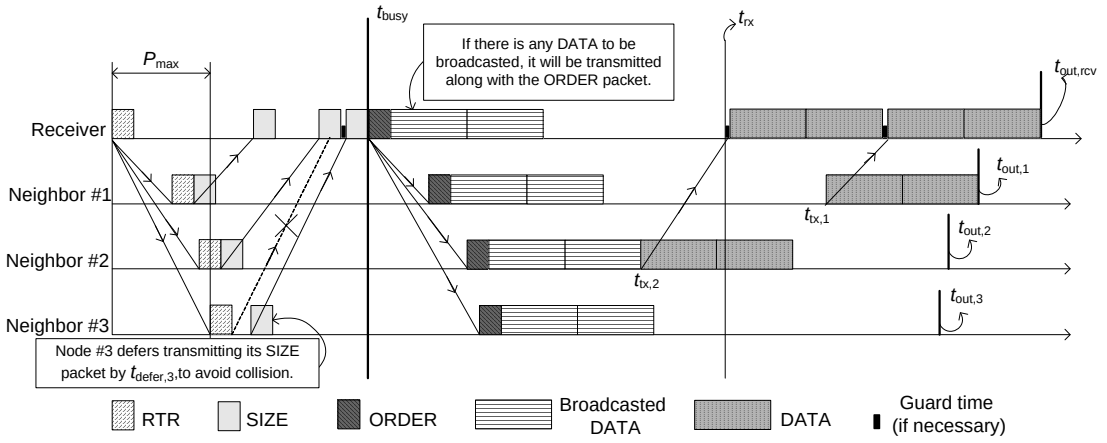

Figure 4.1: 4-way handshaking with multiple-node polling.

of packets that each neighbor wishes to send to it. The receiver then sends an ORDER packet, which informs its neighbors the relative order to transmit their data packets, and how many packets they are allowed to transmit. Finally, the respective neighbors transmit their DATA packets.

4.2.2 Details of RIPT

As mentioned earlier, the RIPT protocol requires every node to know the inter-nodal propagation delay between itself and each of its immediate neighbors. Therefore, the protocol works best in either a static network, or one with limited mobility but every node can determine its own position. For static networks, such inter-nodal propagation delays can be estimated during initialization, in which all nodes take turns to broadcast some control packets to its neighbors. Upon hearing such a packet from one of its neighbors, a node can calculate its propagation delay by comparing the timestamp on the packet with its local clock. Although this procedure requires time synchronization among all the nodes, the assumption is quite reasonable because the initialization stage is short, and thus any

Table 4.1: Notations used for explaining the RIPT protocol.

Notation	Description
t_j	Time at which the neighbor of order j finishes receiving the RTR packet
$t_{\mathrm{SIZE},j}$	Time at which the neighbor of order j starts transmitting its SIZE packet
t_{busy}	Time at which receiver finishes receiving last SIZE packet
$t_{\mathrm{out,rcv}}$	Timeout at receiver
$t_{\mathrm{out},x}$	Timeout at node x
$t_{\mathrm{tx},x}$	Time at which node x starts transmitting DATA packet
$t_{\mathrm{rx},x}$	Time at which node x's DATA packet first arrives at receiver
t_{rx}	Time at which first DATA packet within the packet train may arrive at receiver
M_{train}	Number of DATA slots currently reserved at receiver
$M_{\mathrm{train,max}}$	Maximum allowable value for M_{train}
N_{b}	Number of broadcast packets (if any) from the receiver
$N_{\mathrm{slots},i}$	Number of DATA slots allocated to the i^{th} node to transmit
p_x	Propagation delay between node x and the receiver
$p_{x(j)}$	Propagation delay between the receiver and node $x(j)$ which has order j
p_{max}	Maximum p_x among all first-hop neighbors of the receiver
$p_{x,\,y}$	Propagation delay between node x and node y
n	Average number of first-hop neighbors per node
n_{hidden}	Average number of hidden terminals per node
T_{RTR}	Transmission time of each fixed-length RTR packet
T_{ORDER}	Transmission time of each fixed-length ORDER packet
T_{SIZE}	Transmission time of each fixed-length SIZE packet
T_{DATA}	Transmission time of each fixed-length DATA packet
T_{avg}	Average time interval between initiating RTRs at a node
T_{guard}	Guard time to protect against any estimation error in the inter-nodal propagation delays

clock drift will be negligible if the synchronization is carried out right before deployment. Note that the RIPT protocol no longer requires time synchronization beyond the initialization stage. For the case where each node has positioning capability, messages could be exchanged between neighboring nodes to update each other about their locations, which can then be used for computing the inter-nodal propagation delays.

We now explain how the RIPT protocol works. Table 4.1 shows the notations that are used, while Fig. 4.1 shows an example of how the 4-way handshake is carried out.

4.2.2.1 4-way handshake initiation by the RTR packet

When an idle node wishes to become a receiver, it initiates the 4-way handshake by broadcasting an RTR packet. In order to avoid any confusion, we shall clarify that the terms "receiver", and "senders", refer to the initiating node that intends to be a receiver, and its immediate neighbors that have packets to send to it, respectively. The RTR packet contains the initiating receiver's node ID, the number of DATA slots reserved at the receiver (M_{train}), and the inter-nodal propagation delay from itself to each of its neighbors, if necessary; for the case of a static network, the inter-nodal propagation delay information can be exchanged during initialization, and does not need to be retransmitted with every RTR packet. Note that these inter-nodal propagation delays will be used by each neighboring node to compute the time at which it needs to send its SIZE packet, which will be explained later in Section 4.2.2.2. In order to accommodate the need to broadcast DATA packets, the RTR packet also includes a flag to indicate whether the receiver has any DATA packet to broadcast, as well as a field that indicates the total number of DATA packets it will broadcast (N_{b}). As mentioned

earlier, the RTR packet serves to inform all of the receiver's neighbors that the former is ready to act as a receiver for a certain duration of time.

4.2.2.2 Transmission slot request using the SIZE packet

When a neighboring node hears the RTR packet, it needs to respond with a SIZE packet. The rule of thumb is to transmit the SIZE packet immediately upon receiving the RTR packet, subject to the condition that it will not collide with another node's SIZE packet at the receiver. Any such collision will be costly because the receiver will not allocate any DATA slot to a neighboring node if it does not hear the latter's SIZE packet, and will result in low normalized throughput. Fortunately, such collisions can be easily avoided if the inter-nodal propagation delay between the receiver and each of its neighbors are known to all of these neighbors. Note that the overhead incurred to maintain this information is of the order of $O(n^2)$ per node (where n is the average number of first-hop neighbors per node), if it is statically maintained at each node. If this information is provided by the RTR packet instead, then the overhead is in the order of $O(n)$. The information allows each neighboring node to compute the time at which it is supposed to transmit its SIZE packet without colliding with other SIZE packets at the receiver. The node first arranges the inter-nodal propagation delays between the receiver and each of the neighboring nodes in ascending order. If there are multiple nodes having the same propagation delay, the conflict is resolved by granting priority to the node with the smaller node ID. Suppose the node finds that it has the order j, and t_j is the time at which it finishes receiving the RTR packet. The time at which it should transmit its SIZE packet is given by

$$t_{\mathrm{SIZE},j} = \max[t_j, (t_{\mathrm{SIZE},j-1} + p_{x(j-1)} + T_{\mathrm{guard}} + T_{\mathrm{SIZE}} - p_{x(j)})], \qquad (4.1)$$

where $t_{\text{SIZE},1} = t_1$, $p_{x(j)}$ is the propagation delay between the receiver and the neighboring node $x(j)$ that has order j, and T_{guard} is a small guard time that can be inserted to protect against any estimation error in the inter-nodal propagation delays. We will discuss more about how the value of T_{guard} may be chosen in Section 4.5, but it should be noted for now that the amount of T_{guard} required is usually very small compared to the DATA packet's transmission time.

In order to better understand the above algorithms, we shall look at the example in Fig. 4.1. As can be seen, if both neighboring nodes #2 and #3 respond with their SIZE packets immediately upon hearing the RTR packet, their SIZE packets will collide at the receiver. Here, neighboring node #3 defers transmitting its SIZE packet, so as to ensure that it will only arrive at the receiver after neighboring node #2's SIZE packet has been completely received.

Having resolved the time to transmit its SIZE packet, the neighboring node will also compute the busy duration at the receiver that will be caused by all the SIZE packets sent from the receiver's neighbors. The end of this busy duration is denoted by t_{busy} (see Fig. 4.1), which is the time at which the receiver finishes receiving the SIZE packet sent from its most distant neighbor of order n, where n is the number of first-hop neighbors that the receiver has. Every first-hop neighboring node can then calculate t_{busy} locally as follows:

$$t_{\text{busy}} = t_{\text{SIZE},n} + p_{x(n)} + T_{\text{SIZE}}. \tag{4.2}$$

The value of t_{busy} will then be used to compute $t_{\text{out,rcv}}$, which is the time at which the receiver is expected to finish receiving the entire packet train. Specifically,

$$\begin{aligned} t_{\text{out,rcv}} = t_{\text{busy}} + 2p_{\max} + T_{\text{ORDER}} + (N_{\text{b}} \cdot T_{\text{DATA}}) \\ +(M_{\text{train}} \cdot T_{\text{DATA}}) + n \cdot T_{\text{guard}}. \end{aligned} \tag{4.3}$$

We now describe the information contained within a neighboring node's SIZE packet. It contains the number of relay DATA packets, as well as the number of its own DATA packets that it wishes to transmit to the receiver. It also contains its own timeout, calculated as

$$t_{\text{out},x} = t_{\text{out,rcv}} - p_x, \tag{4.4}$$

where p_x is the propagation delay between the receiver and the node itself. Note that $t_{\text{out},x}$ is the timeout that node x sets to release itself from the current handshaking loop. This timeout needs to be large enough to allow the receiver to finish receiving all the DATA packets in the current handshaking loop. However, it does not need to be as large as $t_{\text{out,rcv}}$, because of the propagation delay p_x between the receiver and itself. It simply needs to be large enough such that any transmission from this node beyond the timeout will not interfere with the receiver.

The SIZE packet serves two purposes. Besides informing the receiver about the number of relay and new DATA packets to be transmitted, it also informs each of the receiver's second-hop neighbors (i.e., its hidden nodes) to avoid initiating an RTR handshake until a certain timeout. For a second-hop neighbor (say, node y), upon receiving the SIZE packet sent by the first-hop neighbor (say, node x), its timeout is

$$t_{\text{out},y} = t_{\text{out},x} + p_{x,y}, \tag{4.5}$$

where $p_{x,y}$ is the propagation delay between node x and node y. The need for the second-hop neighbors to avoid becoming receivers is key for achieving a stable normalized throughput. If the second-hop neighbors were to act as receivers, they may lose some DATA packets due to collisions arising from the first-hop

Table 4.2: An example illustrating the slot assignment strategy, where $M_{\text{train}} = 4$.

Priority	Node ID	Relay packets	New packets	Slots assigned
1	Neighbor #2	0	3	2
2	Neighbor #1	2	1	2
3	Neighbor #3	0	3	0
		Total number of slots assigned		4

neighbors' transmissions.

It is also important to note that the RIPT protocol still functions properly even when some of the receiver's neighbors miss the RTR broadcast. When such a case arises, the only impact on RIPT is that the particular neighbor will not respond with a SIZE packet, and subsequently, it will not be allocated any DATA slot for the current round of handshake.

4.2.2.3 Transmission order broadcast through the ORDER packet

After the receiver has acquired all the SIZE packets from its neighbors, it allocates its available DATA slots (i.e., M_{train}) using a simple strategy. The rule of thumb is to prioritize all relay DATA packets over new DATA packets, because the relay packets have already consumed channel resources to reach the intermediate nodes, and it would be wasteful if they were to be discarded due to buffer overflow. We now explain the assignment strategy using the example shown in Table 4.2, where $M_{\text{train}} = 4$. First, each of the neighboring nodes is assigned a unique priority randomly. Their requirements are then sorted according to decreasing node priority. Next, the receiver runs through the "relay packets" column according to the node priorities, and accommodate as many relay packets as possible. If there are still available DATA slots after considering the relay packets, the node will then run through the "new packets" column, and assign

the remaining DATA slots accordingly. Upon completing the slot assignment, the receiver then transmits the ORDER packet, which contains the total number of DATA slots assigned to each neighboring node, the order of transmission, the broadcast flag, as well as the number of DATA packets to broadcast. Notice that the receiver resends the information on broadcast packets so as to improve the chances of its neighbors to be ready for them. Immediately after transmitting the ORDER packet, the receiver transmits its broadcast packets, if any.

4.2.2.4 DATA Train Transmission

Upon hearing the ORDER packet, a node that has been allocated at least one DATA slot must compute the time at which it shall start its DATA transmission, so that its packets will form a packet train at the receiver with the other senders' packets. Note that the transmission start time must take into account the propagation delay to the receiver. For instance, if node x's DATA packet is expected to reach the receiver at time $t_{\mathrm{rx},x}$, it shall start transmitting the packet at

$$t_{\mathrm{tx},x} = t_{\mathrm{rx},x} - p_x. \tag{4.6}$$

Suppose node x is assigned by the receiver as the l^{th} node to transmit, we can obtain $t_{\mathrm{rx},x}$ as

$$\begin{aligned} t_{\mathrm{rx},x} = t_{\mathrm{busy}} + 2p_{\max} + T_{\mathrm{ORDER}} + (N_{\mathrm{b}} \cdot T_{\mathrm{DATA}}) \\ +(l-1) \cdot T_{\mathrm{guard}} + \sum_{i=1}^{l-1} N_{\mathrm{slots},i} \cdot T_{\mathrm{DATA}}, \end{aligned} \tag{4.7}$$

where $N_{\mathrm{slots},i}$ is the number of slots allocated to the i^{th} node.

4.2.3 Adaptive Train Size

In actual implementation, the packet train size M_{train} for each handshaking loop should not be held constant, because the offered load would fluctuate with time. When the load is low, only a few neighbors may have DATA packets to transmit, or, in the worst case, no neighbor has any DATA packet to transmit. On the other hand, when the load is high, many neighbors may wish to transmit DATA packets. A self-adaptive algorithm would allow each node to adapt the M_{train} parameter according to the current load observed. In particular, the total number of DATA packets that all neighbors wish to transmit during the current handshaking loop could be used to predict a suitable M_{train} value for the future handshaking loop.

We now describe a possible approach as follows. If the receiver finds that its M_{train} is not large enough to accommodate all the slot requests from its neighbors, it increases M_{train} by 2 for the next round. If it finds that there are insufficient slot requests to fill up its M_{train}, it decreases M_{train} by 1. The main reason why the algorithm is more conservative when decreasing M_{train} is due to the relative reliability of the above two triggers. To understand this, we need to be aware that the sum of slot requests computed from those SIZE packets that it receives does not always reflect the true number of packets that its neighbors wish to send. The inaccuracy may arise because some neighbors' SIZE packets might have been corrupted, or it may be because some neighbors are required to remain silent as they are currently involved in other handshaking loops. Although the computed sum may not be accurate, if it happens to be higher than the current M_{train}, there is no ambiguity that the current M_{train} is indeed too small. On the other hand, if the sum is less than M_{train}, the receiver cannot be sure whether its M_{train} is indeed too large, because there might be missing information. Note that the change in M_{train} only affects the next round of handshake. Also, there

should be a maximum limit for M_{train}, so as to avoid any receiver from capturing the channel for too long.

4.2.4 When to Initiate an RTR Packet

Because the RIPT protocol requires a node that wishes to act as a receiver to initiate the handshaking loop by broadcasting an RTR packet, the timing of initiating RTR packets is an important issue. Although a traffic prediction scheme might be useful for helping a node to schedule the proper time to initiate the handshaking loop, it is beyond the scope of our study. Here, we simply pick the exponential distribution for the time between RTR-initiations, with an average of T_{avg}.

In order to avoid the same node from acting as a receiver successively before other neighboring nodes have a chance at playing the role, we make use of a "fairness bit" at each node. If a node has just been released from a handshaking loop while acting as a receiver, it will set this bit to '0'. While in this state, it will not initiate any RTR packet. The fairness bit can only be reset once the node has served as a sender in any subsequent handshaking loop. However, if the node's fairness bit has been set to '0' for longer than a threshold time t_{limit}, it will reset the fairness bit back to '1' to avoid any deadlock.

4.3 Simulation Model

Our simulation model consists of 36 static nodes arranged in a grid topology, as shown in Fig. 4.2. However, instead of precisely placing each node at a grid intersection point, we introduce some degree of randomness by allowing each node to deviate from the grid intersection point by a maximum of 10% of its grid

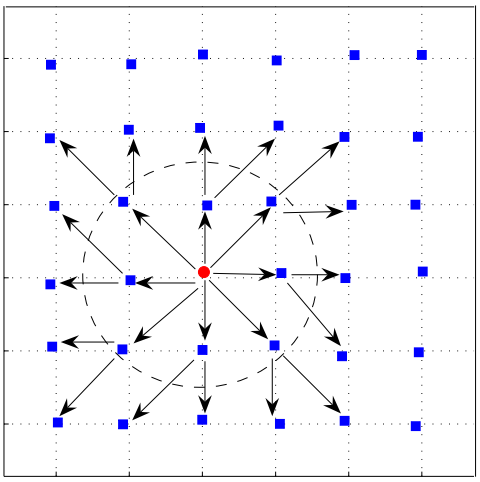

Figure 4.2: Our simulation network topology. Note that the nodes are not placed precisely at the grid intersection points. Also, the arrows in the figure only show the routes between a single node and its 16 two-hop neighbors.

spacing, in both the x and y directions. The deviations from the grid intersection points are introduced here in order to ensure that the network topology resembles a real scenario, whereby the nodes are usually non-equidistantly placed. Note, however, that the RIPT still works even if the neighboring nodes are equidistant. The transmission range of each node is assumed to be 1.75 times the grid spacing, such that each node has exactly eight neighbors within its range. In order to avoid edge effects, we have adopted the wraparound strategy, such that even the nodes at the boundaries will have eight one-hop neighbors. Note that, in a real scenario where edge effects exist, we expect the normalized throughput to be higher than our simulation results; this is because the nodes at the network edge usually have lower number of hidden and exposed nodes, thus resulting in lower number of collisions.

We assume that the traffic load is divided evenly among all nodes according to the Poisson distribution. For routing, in order to make it easier to interpret the results, we consider two-hop routes only, rather than varying number of hops.

For each packet that is generated by a node, we randomly pick its destination to be any of the node's 16 two-hop neighbors with equal probability. Also, we apply static routing here. The arrows in Fig. 4.2 show the routes originating from *one* particular node (the round node) to each of its 16 two-hop neighbors. We do not show the two-hop routes for each of the remaining 35 nodes when they behave as source nodes, but their two-hop routes have exactly the same pattern.

We also assume that all the nodes are equipped with half-duplex, omnidirectional modems, with a fixed data rate of 2400 bps. The acoustic propagation speed is assumed to be 1500 m/s. Since the RIPT is designed to perform independently of the physical layer, and our simulation study only focuses on the RIPT's performance in the MAC layer, we do not specify the modulation scheme used. Here, the channel is also assumed to be error-free, so that all packet losses are purely due to the MAC protocol's performance. We also do not implement ACK for any of the schemes simulated, thus there is no retransmission for lost packets. All control packets (i.e., RTR, SIZE, and ORDER) have the same size of 100 bits, while DATA packets are 2400-bit long. The buffer size for both new packets and relayed packets are set to 100 each, and the parameter M_{train} is initialized to 1. We choose to benchmark our protocol with three other schemes, namely, Aloha-AN [62] and MACA [23] and MACA with packet train (MACA-PT). In these three schemes, we set the control packet length (i.e., NTF packet for Aloha-AN, and RTS/CTS packets for MACA and MACA-PT) to 64 bits, while keeping all other parameters the same. Note that MACA-PT is very similar to MACA but its DATA transmission is in a form of a packet train. Similarly to RIPT, the size of the packet train (M_{train}) in MACA-PT is adaptive within the range between 1 and $M_{\text{train,max}}$.

Note that, all the protocols in our simulation study are random access MAC

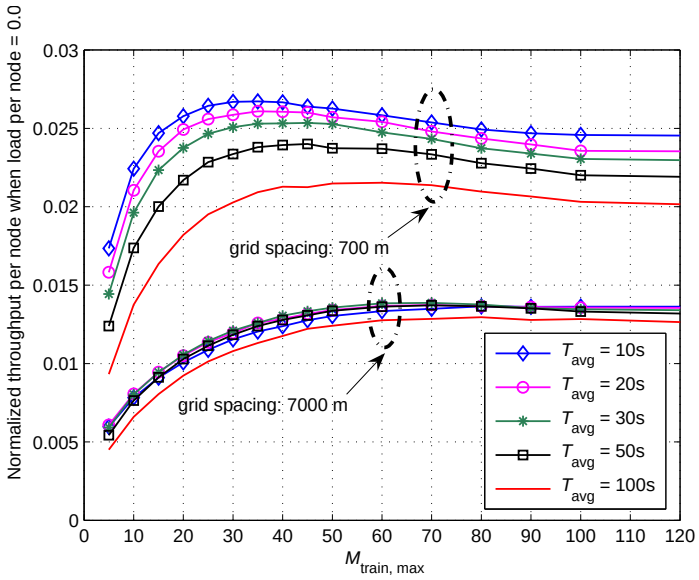

Figure 4.3: Effects of $M_{\text{train,max}}$ and T_{avg} on normalized throughput.

protocols that do not require any time synchronization. We also investigate both 700 m and 7000 m grid spacings to evaluate the performance of the protocols under different average propagation delays.

4.4 Simulation Results

Here, we adopt the definition of "normalized throughput" from [20], and define "normalized throughput per node" as the average normalized throughput over 36 nodes as follows:

$$\text{Normalized throughput per node} = \frac{1}{36}\left[\frac{\text{No. of Packets Received/Simulation Time}}{\text{Data Rate/Packet Length}}\right] \tag{4.8}$$

4.4.1 Factors Affecting the RIPT's Performance

- **$M_{\text{train,max}}$ and T_{avg}:** Fig. 4.3 shows how the parameters $M_{\text{train,max}}$ and T_{avg} affect the RIPT's normalized throughput when the offered load per node is 0.07. Note that this is the offered load that is high enough to cause the RIPT's normalized throughput to saturate. From the figure, we can observe that when T_{avg} becomes large, the normalized throughput actually decreases. Ideally, T_{avg} should be as small as possible, in order to reduce the packet delays. For the case where the grid spacing is 700 m, we observe that the suitable range of $M_{\text{train,max}}$ varies with T_{avg}. For example, when T_{avg} is 10 s, the suitable range of $M_{\text{train,max}}$ would be approximately [25,40]. However, when T_{avg} increases to 100 s, the suitable range of $M_{\text{train,max}}$ also increases to [35,70]. This is intuitive since a larger T_{avg} would imply that there are more DATA packets waiting to be transmitted in each handshake. When $M_{\text{train,max}}$ is outside these suitable ranges, we see that the RIPT's normalized throughput deteriorates. Furthermore, for any T_{avg}, the normalized throughput initially increases as $M_{\text{train,max}}$ increases, but begins to decrease when $M_{\text{train,max}}$ is too large. This can be explained as follows. If $M_{\text{train,max}}$ is too small, the network actually spends more time exchanging control packets rather than transmitting DATA packets, which results in low normalized throughput. On the other hand, if $M_{\text{train,max}}$ is too large, the normalized throughput may also be low due to the higher chances of collisions, and also due to more unutilized reserved slots. Note that, despite using a receiver-initiated handshaking approach, the RIPT still encounters collisions, just like other transmitter-initiated handshaking approaches. Collisions can occur between the various combinations of control packets and DATA packets. When a node transmits longer train of

packets, there is a higher chance that it may miss the control packets from its other neighbors, thus losing the opportunity to keep an accurate view of its neighbors' status (e.g., when they are acting as receivers); this in turn reduces the node's capability to avoid collisions. In addition, if a neighboring node misses an RTR packet, it will not be able to transmit DATA packets to the receiver, even if it has many packets to send; this may result in unutilized reserved slots at the receiver. Another point worth mentioning about $M_{\text{train,max}}$ is its effect on packet delay. Although not shown here, we have found via simulations that the packet delay tends to increase when $M_{\text{train,max}}$ becomes larger.

- **Inter-nodal Propagation Delay:** When the grid spacing is increased from 700 m to 7000 m, we can see from Fig. 4.3 that the normalized throughput becomes less sensitive to the variation in T_{avg}. However, changes in $M_{\text{train,max}}$ still produce significant changes in the normalized throughput. The suitable ranges of $M_{\text{train,max}}$ have also increased compared to the previous case. This is intuitive since more DATA packets should be transmitted in each round of handshake in the presence of longer propagation delay, in order to stay efficient. It should also be mentioned that the overall normalized throughput has also dropped significantly compared to the previous case. This is a common observation among handshaking-based MAC protocols.

- **Packet Length:** We have also performed simulations for the case where the packet length is increased to 4800 bits. However, we do not show the results here because similar conclusions can be made as above. Nevertheless, it should be mentioned that a larger DATA packet length is observed to reduce the effects of $M_{\text{train,max}}$ on normalized throughput.

4.4.2 Performance Comparison Against Aloha-AN and MACA

For a better understanding of the RIPT protocol's performance when compared against Aloha-AN, MACA and MACA-PT, we use four metrics as our performance measure, as follows:

- **Normalized Throughput:** As can be seen in Fig. 4.4, the RIPT outperforms all other schemes significantly as the load increases. The normalized throughput of Aloha-AN, when implemented in the current multi-hop network setting, becomes lower than that of MACA and MACA-PT at high load, although the latter are designed for terrestrial networks. This is because Aloha-AN does not address the hidden terminal issue in its design, which becomes worse when the load is high. The results obtained from the study of MACA and MACA-PT has proven that handshaking-based schemes could help reduce collisions in multi-hop underwater networks by alleviating the hidden terminal problem. They also guarantee a stable normalized throughput at high load. However, as seen in Fig. 4.4, their normalized throughputs are significantly lower when compared to our RIPT protocol. This is largely due to MACA's inefficiency in underwater since it only transmits a single data packet per round of handshake, which suffers from under-utilization of the channel when the propagation delay is high. Although the transmission of the DATA packets in a form of packet train is allowed in MACA-PT, its normalized throughput is still lower than RIPT since the handshake in MACA-PT is only involves with only a single neighboring node. In contrast, our RIPT protocol improves normalized throughput by forming a packet train for multiple neighboring nodes for each round of handshake.

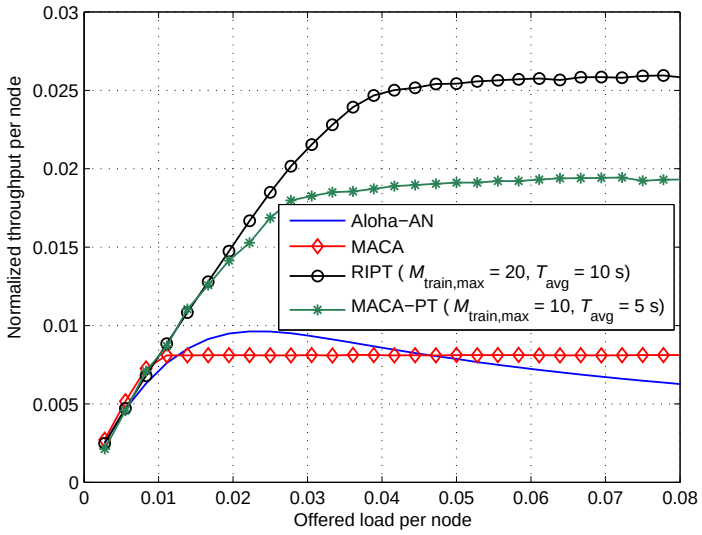

Figure 4.4: Comparing the througput of RIPT, MACA, MACA-PT, Aloha-AN (grid spacing = 700 m).

- **Number of DATA Packet Transmissions and Collisions:** In Fig. 4.5, it is shown that the number of transmissions and collisions of the three protocols do not increase (beyond a certain value) even when the load increases. This is because at this point the channel has been saturated. Increasing the load (or increasing a number of generated packets) beyond this saturation point will only result in a higher number of backoffs but not a higher number of packet transmissions. Consequently, the number of collisions is also constant beyond this point. We also notice that the RIPT transmits approximately as many DATA packets as the Aloha-AN, while having less number of collisions. This confirms that the RIPT avoids collisions by maintaining more accurate information about the receiver compared to Aloha-AN. When comparing the RIPT against MACA and MACA-PT, the RIPT transmits much more packets. This arises from our technique of using multiple neighbors to form a packet train at the receiver, which is much more efficient than MACA and MACA-PT. Despite being able

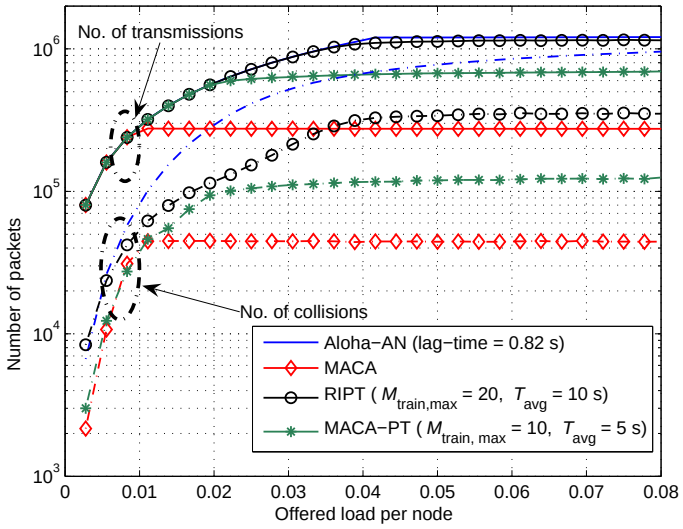

Figure 4.5: Comparing the number of DATA packet transmissions and the number of DATA packet collisions of RIPT, MACA, MACA-PT and Aloha-AN. Note that the simulation duration is 8×10^5 s for every point.

to offer a high and stable normalized throughput, the RIPT suffers from much higher number of collisions than MACA and MACA-PT. Although the RIPT's handshaking-based mechanism can greatly alleviate the hidden terminal problem in multi-hop networks, it cannot resolve the problem completely. Thus, whenever a collision occurs, a large number of DATA packets within a packet train may be corrupted. In contrast, MACA and MACA-PT transmit less number of DATA packets during each round of handshake, and hence, they lose less packets in a collision.

- **Delay:** Fig. 4.6 shows the delay performance of all schemes. At very low load (below 0.01), the RIPT has the worst delay performance. This is because of its receiver-initiated approach, whereby a sender needs to wait until there is a handshake initiated by the receiver before it can attempt transmit a DATA packet to the latter. Moreover, its packet train tends to be very short when the load is low, thus making the overhead of its

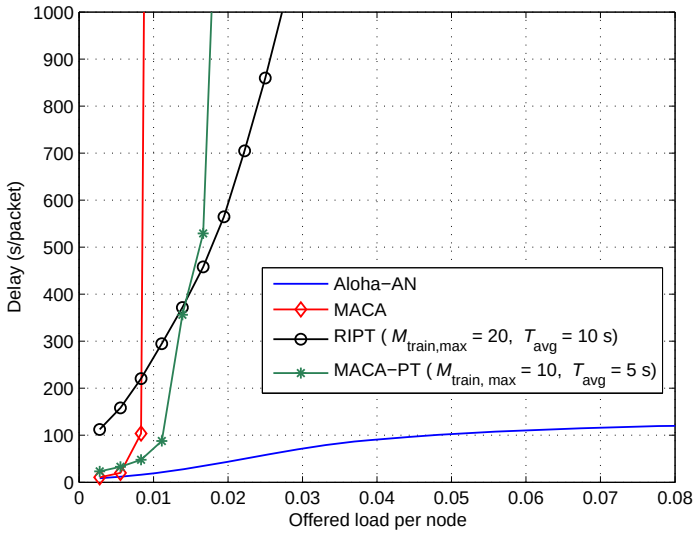

Figure 4.6: Comparing the packet delays of RIPT, MACA, MACA-PT and Aloha-AN.

4-way handshake mechanism more significant. However, beyond a load of 0.01 and 0.015, its delay becomes shorter than the MACA and MACA-PT, respectively. This is the point where the average packet train size has grown large enough to overcome the overheads incurred by both the 4-way handshake, and the average waiting time for the handshake initiation by the receiver. The Aloha-AN is seen to have the best delay performance among all the three schemes. This is due to the fact that it is not a handshaking-based protocol, and only uses a one-way notification mechanism.

4.5 Discussion

An important point that we would like to stress about the RIPT protocol is that, a neighboring node can compute the time at which it needs to transmit its SIZE packet and its DATA packet train, based on the time at which it receives the RTR packet; in other words, there is no need for absolute clock synchronization.

In a way, this bears some similarity with the MAC protocol proposed in [22], in which the neighboring nodes can achieve a locally synchronized schedule without absolute clock synchronization. Therefore, the purpose of the guard times in the RIPT protocol is merely to buffer any error in the inter-nodal propagation delays that were previously estimated. These small guard times are required at strategic instances within the 4-way handshake. It is important to note that we do not require the guard time to be inserted between every DATA packet that is sent; it is only inserted between the string of packets that are transmitted from different neighbors. Also, the guard time only needs to be as large as the maximum expected error in the inter-nodal propagation delay estimation, which may be in the order of tens of milliseconds. Note, also, that it is possible for the RIPT protocol to correct any error in the delay estimates through the following enhancement. A receiver can examine the timings at which the SIZE packets are arriving from its neighbors, and calculate their deviations from the expected arrival times. The receiver can then include the timing corrections the next time it broadcasts an RTR packet to these neighbors.

Earlier in Section 4.4, we have seen that if $M_{\text{train,max}}$ is too large, it may result in low normalized throughput as well. In the following, we provide some guidelines for selecting an appropriate $M_{\text{train,max}}$. As the receiver and the senders need to exchange control packets before the DATA packet train can be transmitted, the inter-nodal propagation delays thus introduce some amount of fixed-cost (C_{fix}) in each handshake. Assuming that the receiver does not have any packet to broadcast, and also assuming the worst-case scenario whereby all the n first-hop neighbors are located at $p_{\max}$ from the receiver, the 4-way handshake incurs a fixed-cost overhead of

$$C_{\text{fix}} = 4p_{\max} + T_{\text{RTR}} + nT_{\text{SIZE}} + T_{\text{ORDER}}. \tag{4.9}$$

Thus, the total transmission time of the DATA packet train must be longer than C_{fix}, in order to justify this overhead. Besides the above constraint, $M_{\text{train,max}}$ should also be able to accommodate all the DATA packets waiting to be transmitted in each of the receiver's neighbors. Assuming that the network is operating in the high load region, at which all nodes are backlogged. Thus, in each handshake, $M_{\text{train,max}}$ should ideally be large enough to accommodate $(n_{\text{hidden}} + n)$ DATA packets, where n is the average number of first-hop neighbors per node, and n_{hidden} is the average number of hidden terminals per node. Thus, the size of $M_{\text{train,max}}$ can be calculated as

$$M_{\text{train,max}} = \left\lceil \max\left(\frac{C_{\text{fix}}}{T_{\text{DATA}}}, n_{\text{hidden}} + n\right)\right\rceil . \tag{4.10}$$

So far, we have assumed a scenario whereby all nodes are statically deployed, and set up at the same time. We now discuss how the RIPT can be modified to handle network dynamics caused by new nodes joining an existing network. Suppose a new node, Node y, wishes to be considered by a receiver as one of its possible transmitting neighbors. In order to cope with this, a receiver's 4-way handshake can include an additional listening interval, T_{join}, right after the time at which it expects the last bit of the last SIZE packet, which we denote by $T_{\text{SIZE\_end}}$. In order to declare its presence, Node y first listens to the receiver's RTR packet, and calculates the value of $T_{\text{SIZE\_end}}$ from the inter-nodal propagation delay information attached within the RTR packet. Next, it transmits a short JOIN packet that will be received at the receiver some time within the T_{join} interval. This can be ensured if T_{join} is larger than the maximum propagation delay between one-hop neighbors, which depends on the transmission range. Since clock synchronization is no longer available, Node y must try to estimate the propagation delay between itself and the receiver as half of the round-trip

time (RTT) instead. The RTT can be obtained using a technique that exchanges time-stamped messages (as nicely described in [53]); in our case, the ORDER packet is used to piggyback the time-stamped message in the reverse direction. Once Node y obtains the propagation delay to the receiver, it transmits this information to the receiver during the next round of handshake, to be received within the receiver's T_{join} interval again. It can then be formally included as one of the receiver's neighbors. Note that a receiver does not need to include T_{join} in every handshake; it can use a flag within its RTR packet to indicate whether the current 4-way handshake includes the interval T_{join}. In this way, the receiver can control the overhead incurred, which is a tradeoff with how soon a new node can join as its neighbor.

Although we have omitted the effects of channel packet losses (e.g., due to bit errors) in our discussion so far, we will now discuss them briefly. Note that, a neighboring node can calculate when to transmit its SIZE packet and DATA train, so long as it has received the RTR packet and the ORDER packet correctly. If these packets were corrupted, the neighboring node simply does not get to transmit any DATA train in the current round of handshake, which causes the normalized throughput to drop. Nevertheless, the other neighboring nodes can still proceed without any timing conflict. Conversely, if a receiver does not receive a neighbor's SIZE packet correctly, it will not allocate any DATA slot for that neighbor, which also reduces the normalized throughput performance. This, however, only has an isolated effect on that neighbor, but not other neighbors.

4.6 Conclusion

In this chapter, we have proposed and studied a MAC protocol for multi-hop UWA networks – RIPT. The RIPT protocol is a handshaking-based protocol

that addresses the channel's long propagation delay characteristics by utilizing receiver-initiated reservations, as well as by coordinating packets from multiple neighboring nodes to arrive at the receiver node in a packet train fashion. The utilization of receiver-initiated reservations helps to combat the effect of the space and the time uncertainties. These uncertainties usually lead to the high number of packet collisions. The coordination of packets (e.g., packet train), on the other hand, helps to enhance the low normalized throughput that is caused by the use of handshaking mechanism in a long propagation delay environment.

We have confirmed through simulations that the RIPT can achieve high and stable normalized throughput with proper values of packet train size, M_{train}, as well as average time between handshake initiations, T_{avg}.

Overall, the RIPT is suitable for delay-tolerant underwater applications that are required to operate at high load, such as undersea exploration and data collection. In particular, it is efficient for networks in which every node has a large number of neighbors. This is because the RIPT allows multiple neighbors to transmit to a receiver at one go to form a packet train, in contrast to other handshaking-based protocols that would require every neighbor to perform dedicated handshake with the receiver. Furthermore, due to its receiver-initiated handshaking nature, the RIPT would be more appropriate for applications in which the offered load does not fluctuate too rapidly. Otherwise, the system parameter M_{train} may not adapt fast enough, which leads to inefficiency.

Chapter 5

Routing in UWA Networks

5.1 Introduction

In the previous two chapters, we have been focusing on the design of MAC protocols for UWA networks. Next, we will turn our interest to routing, as it is also one of the integral factors to achieving the desired network performance. Given the numerous routing algorithms available, ones would expect that we could easily adopt some of these protocols for UWA networks since such networks have many similarities with terrestrial ad-hoc networks such as the absence of infrastructure, energy constraints and contentions between sensors, etc. However, due to the differences between the two systems as discussed in Chapter 1, extra attention must be given to these differences when design routings for UWA networks. More specifically, the challenges in routing in mobile UWA networks arises from:

- **The use of an acoustic channel:** the low bandwidth and high propagation delay characteristics of the channel make it highly likely that the topology information becomes stale by the time it reaches the intended node. Without the topology information, the problem in routing becomes very interesting. Moreover, disconnectivity can be experienced more often

in UWA networks, due to the high error rate of the channel.

- **Node mobility:** network dynamic makes the routing problem very challenging because route stability and route management become important issues in addition to energy and bandwidth efficiency.

- **The low density of sensor nodes:** when coupled with a node's mobility makes it highly likely to have a disconnected (networks void) network.

From the literature review given in Chapter 2, we find that location-based scheme is an efficient approach for UWA routing, as it requires small amount of overheads and low power consumption when routing a packet. Hence, we propose in this chapter a routing protocol for mobile UWA network, called, "Sector-based Routing Protocol (SBR)". It is a location-based routing protocol that is designed to maximize the packet delivery ratio (PDR) in both low and high node density conditions. As other location-based routings, SBR assumes a knowledge of a node's own and a destination's location (e.g. the sink is static and its location is known to all the nodes) to enable the node to efficiently select the next best forwarding node based on a sector-based routing mechanism. Such mechanism takes long propagation delay and node movement into account, resulting in the enhancement of the PDR.

In order to work well in networks with mobile sink, we propose the "SBR-DLP". It is the SBR that couples with the Destination Location Prediction scheme. Specifically, the node in SBR-DLP utilizes the destination's pre-planed path and the simple location prediction mechanism, in order to estimate the position of the mobile destination node. This estimated location is then used in the finding of the node's next best forwarder to route the packet.

The remainder of this chapter is organized as follows. We present the proposed routing protocols, SBR and SBR-DLP, in Section 5.2 and Section 5.3, respectively. Section 5.4 describes the detail of the simulations that were carried out to evaluate the protocol and the simulation results are discussed in Section 5.5. Finally, we conclude this chapter in Section 5.6.

5.2 SBR

5.2.1 Overview of SBR

The SBR is a location-based routing protocol in which each of the sensor nodes does not carry any information of its neighboring nodes or the network topology. However, each node knows its current and the destination node's positions. In practice, on-board navigation equipment allows each AUV to know its current location. While such positional information may not be very accurate, the SBR does not require exact knowledge of a node's location to perform well. For the destination's location, we assume that the SINK is fixed, thus its location can be made available prior to node deployment.

The main objective in designing the SBR is to enhance the PDR in a sparse network such as a network that is purely established with a fleet of costly AUVs. Typically, in a single-sector scheme (e.g., the FBR [50]) the PDR is very sensitive to the chosen sector size. For example, by setting the sector size small, a single routing sector scheme can achieve good PDR; however its performance can be significantly degraded in a sparse network. The SBR overcomes this problem by considering not only a single sector, but multiple sectors simultaneously, in order to cover the whole communication area, which makes much more sense in a sparse network. Thus, it is expected to be less sensitive to sector size.

To address the problem of the long propagation delay, the SBR utilizes hop-by-hop routing which means that a node only tries to find the best relay node locally (among its neighbors), instead of finding the complete path from the sender to the destination. Although finding a complete path before actually transmitting a packet can potentially prevent a routing loop, which can be very harmful in a resource constraint system such as an UWA network, the node's mobility and the long propagation delay may render the selected path stale.

At this point, it is important to note that SBR is different from both the VBF [48] and the HH-VBF [49] in that instead of allowing each candidate node to decide whether it should relay the packet, the SBR lets the sender determine its next hop using the information given from all the candidate nodes. This eliminates the problem of having multiple nodes acting as relay nodes which is encountered in both the VBF and the HH-VBF. Moreover, because of the constant radius of the routing pipe in both VBF and HH-VBF, it is possible that there is no node within the routing pipe.

When compared with the FBR, although the SBR adopts a similar idea to the FBR (e.g. both protocols let the sender decide its next relay node), instead of using a single transmitting cone which covers only a fraction of the communication area, the SBR considers the whole communication area to locate the candidate relay nodes. While the FBR needs to rebroadcast the RTS packet every time it cannot find a candidate node within its transmitting cone, the SBR can handle this situation without the need for any retransmission of the RTS packet. Note that even if the FBR extends its transmitting cone to have its angle width of 360°, due to the lack of a collision avoidance mechanism, collisions from the CTSs coming from different neighbors may occur, resulting in a degraded performance. Such a problem is highly pronounced in a dense network. In the SBR, this

problem has been addressed in its design which will be discussed in the following section.

5.2.2 Details of SBR

5.2.3 Finding the Next Relay Node

When a node, S, wishes to send a packet (either a new or relay packet) to the destination node D, it finds its next relay node by broadcasting a *Chk_Ngb* packet, which includes the sender's current position and the packet ID. Upon hearing the *Chk_Ngb*, each neighboring node x checks whether it is nearer to Node D than the distance between Nodes S and D (this is possible since all nodes assume the knowledge of the destination node). If the condition is met, Node x will have to respond to node S by transmitting a *Chk_Ngb_Reply* packet.

In order to reduce possible collisions at node S among the *Chk_Ngb_Reply* responses, each neighboring node first determines the sector that it is in, and then schedules the transmission time of its *Chk_Ngb_Reply* accordingly. For a given k-sector system, the node starts to locate the first sector by ensuring that the sector is bisected by the virtual vector $\overline{SD}$. The subsequent sectors are then labeled according to their priorities, which are determined using their angular differences from $\overline{SD}$. Figure 5.1 illustrates how a four-sector system is labeled. After determining the sector that it is in (say, j), a neighboring node writes into its *Chk_Ngb_Reply* the sector number j, its node ID, and its estimated distance from the predicted destination location. It then schedules the transmission to occur after an offset given by

$$t_{\text{offset},j} = \alpha(j-1)P_{\max}, \tag{5.1}$$

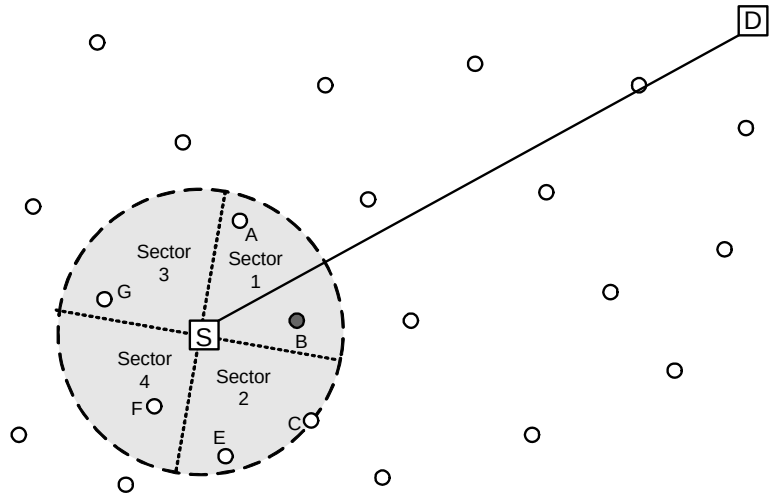

Figure 5.1: Forwarder selection at the sender.

where α ($0 \leq \alpha \leq 1$) and $P_{\max}$ are a parameter used to alleviate the chance of collision caused by the transmissions from different sectors and the maximum propagation delay, respectively. As a general guideline, α can be selected based on the number of sectors, k, such that if k is large, a small α would suffice because the probability of collisions from different sectors would also be small. This also helps shorten the duration that Node S needs to wait before acquiring all the responses from all the k sectors.

After gathering all the *Chk_Ngb_Reply* from its candidate neighbors, node S filters out those nodes that might travel out of its range before being able to acknowledge the receipt of its packet. This is estimated using its propagation delay from each candidate node, the time at which it receives the *Chk_Ngb_Reply*, and the maximum possible relative velocity. Note that the filtering is necessary because the change in their relative distance may be quite significant over the long delay incurred, as a result of the slow propagation speed of UWA waves. Also, if a more accurate estimation of the candidate node's movement is desired, one may consider including the node's direction and speed in the *Chk_Ngb_Reply*

Table 5.1: 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

to the sender. After performing the filtering, the remaining candidates are sorted according to their sector priorities. If there is more than one candidate having the same priority at the top of the list, the one that has the closest predicted distance to Node D will be picked. Table 5.1 illustrates how Node S picks Node B to be its relay node, based on the topology shown in Figure 5.1. After selecting its relay node, Node S transmits its data packet to this node. The relay node then acts as a sender using the same procedure above, until the packet reaches its destination.

Now, suppose that there is no response from any of the sender's neighboring nodes. The sender shall wait for a time interval of T_{wait} for the topology to change, before making another attempt. If the sender fails to find any neighbor for a number of n_{discard} times, it discards the packet.

5.2.4 Implicit/Explicit Acknowledgments

For a high error rate channel such as the UWA channel, the acknowledgment is preferably done in a hop-by-hop fashion, while leaving the end-to-end acknowledgment for the higher layers. Moreover, to achieve energy efficiency, the acknowledgment in the SBR-DLP is done implicitly through overhearing whenever possible. The sender assumes that the packet is successfully received if it overhears the *Chk_Ngb* packet from its relay node while the latter is trying to find

the next relay node. An explicit acknowledgment (ACK) packet will be used in the case where the packet is received by its destination node. If a sender does not receive any acknowledgment after a certain timeout, it makes another transmission attempt by broadcasting the *Chk_Ngb* again. For the case where a relay node has successfully received the data packet but its acknowledgment fails to reach the sender, it will reply to the sender with an explicit ACK when it hears the sender's subsequent *Chk_Ngb* for the same packet ID.

5.2.5 The Analysis of SBR

Theorem 5.1. Given the same sector size with an angle of θ_k, and $\alpha = 1$ for the SBR, if a packet is routable in the single-sector scheme, it is also routable in the SBR.

Proof. : Assuming that the sets of nodes, $\mathcal{N}$ and $\mathcal{M}$, exist in the routing sector of the single-sector scheme and Sector 1 of the SBR, respectively. For the same network topology, because the alignment of Sector 1 in the SBR is exactly the same as the routing sector in the single-sector scheme, we thus have $\mathcal{N} = \mathcal{M}$. Now, $\alpha = 1$ ensures that there will be no collision between responses coming from different sectors in the SBR. Thus, if a single-sector scheme picks Node p as the relay node, where $p \in \mathcal{N}$, the SBR will also pick Node p. □

Theorem 5.2. Given the same sector size with an angle of θ_k, and $\alpha = 1$ for the SBR, a packet that cannot be routed using the single-sector scheme may still be possibly routed by the SBR.

Proof. : Again, since $\alpha = 1$, there is no collision between responses coming from different sectors in the SBR. We prove this lemma by examining the total routing angle of the two schemes. More specifically, the larger the total routing angle

that a scheme has, the higher the chance that a node could find a relay node. In the single-sector scheme, it is obvious that the total routing angle is the same as its routing sector's angle, which spans from $-\theta_k/2$ to $\theta_k/2$. On the other hand, due to the criterion of $l_{x\mathrm{D}} < l_{\mathrm{SD}}$, only those nodes that are nearer to the SINK (node D) than the sender (node S) can respond to the sender and become candidate relay nodes. Hence, the total routing angle in the SBR ranges from -90° to 90°. Because the criterion of $l_{x\mathrm{D}} < l_{\mathrm{SD}}$ also applies to the single-sector scheme, the total routing angle in a single-sector scheme can never be larger than that of the SBR. □

The two lemmas discussed above lead to the conclusion that the PDR of the SBR is lower-bounded by the PDR of the single-sector scheme.

5.3 SBR-DLP

In this section, we present an enhanced version of the SBR which we shall call "SBR-DLP". The SBR-DLP is designed to relax the need of precise knowledge of the destination's location in the SBR. As a result, broader range of UWA applications are possible. In SBR-DLP, each node assumes the knowledge of its own position, and the destination node's pre-planned movements. Here, we are more interested in applications such as sea exploration and monitoring, etc., that may require the destination node to move along with the mobile network in order to cover the entire exploration/monitoring area. For such applications, the destination node acts as a moving sink, and its movement is usually predefined prior to launching the network.

Although the SBR-DLP tackles the mobility issue of the destination node

by assuming that its pre-planned movements (e.g., its waypoints and their corresponding schedule) are made known to all other nodes before launching, it is important to note that the destination node may still deviate from its schedule due to the ocean current. Also, the SBR-DLP does not assume the knowledge of all other nodes' movements.

In detail, when there is a packet to be routed, the node in the SBR-DLP adopts the exact operations (e.g., finding candidates, choosing the next relay node, acknowledging the received packet) as if it were implemented with the SBR. The only difference between the two protocols is how the sender node acquires the destination's location. While SBR assumes that the destination's location is available accurately (e.g., static sink) at all time, SBR-DLP relies on the "destination location prediction" scheme, in order to predict such information. In the following sub-section, we discuss the destination location prediction scheme that is implemented in the SBR-DLP in detail.

5.3.1 Destination Location Prediction

In order to help predict the destination location, the SBR-DLP requires the destination node to periodically broadcast a "*Notification (NTF)*" packet to notify its one-hop neighbors if it deviates from its schedule significantly. We choose to notify only its one-hop neighbors, rather than the entire network, because the long propagation delay can cause the *NTF* packet to become stale by the time it reaches a node that is several hops away. In addition, since the destination node itself is mobile, other nodes within the network may also hear its *NTF* packet at a different time. In order to trigger the *NTF* packet, the destination node checks if it has deviated from its schedule every time when it reaches a predefined waypoint. If it finds that the difference (Δ) between the current time (t_{NTF}) and the

scheduled time (t_{expect}) is greater than a threshold ($\Delta_{\text{threshold}}$), it will broadcast the *NTF* packet, which contains the parameters t_{NTF} and Δ. Upon hearing the *NTF* packet, a node stores these parameters for later use.

Now, suppose that a node has just heard the *Chk_Ngb* packet from a sender at time t_{now}. It first checks if it has previously heard the *NTF* packet. If so, it will estimate the current location of the destination by looking at the destination's predefined movement at the time that is offset by $\hat{\Delta}$ from its schedule, where $\hat{\Delta}$ is the estimated time difference from the predefined schedule. The node uses the parameters t_{NTF}, Δ, and t_{now} to compute $\hat{\Delta}$ using

$$\hat{\Delta} = \frac{\Delta \cdot t_{\text{now}}}{t_{\text{NTF}}}. \tag{5.2}$$

5.4 Simulation Model

In our simulation setup, there are N sensor nodes moving randomly within a 2D network of 1000 m by 1000 m. All nodes (including the SINK) are equipped with half-duplex and omnidirectional modems, which operate at a fixed data rate of 2400 bps, and a communication range of 300 m. We assume that the speed of sound in underwater is constant at 1500 m/s, while the maximum speed (V_{max}) of the sensor node is 2 m/s unless specified otherwise. The speed and direction of each sensor node are randomly picked from the range of $[0, V_{\text{max}}]$ and $[-45°, 45°]$, respectively, according to uniform distribution. These remain constant for an exponentially distributed period of time with an average interval of 300 s, before they are randomly picked again. If the node reaches the boundary of the testing area, it is reflected by the boundary, so that it goes back within the testing area. The *DATA* and control packets (e.g., *Chk_Ngb, Chk_Ngb_Reply, NTF and ACK*) are 4800-bit and 32-bit long, respectively, while the other parameters used are:

$n_{\text{discard}} = 3$, $\alpha = 0.5$, $\Delta_{\text{threshold}} = 30$ s and $T_{\text{wait}}(n_{\text{fail}}) = 30n_{\text{fail}}$ s, where n_{fail} is the number of failed attempts to transmit a packet and $1 \leq n_{\text{fail}} < n_{\text{discard}}$.

There is only one destination node, referred to as a SINK. While the SINK is fixed and located at the point (1000, 1000) for the study of the mobile network with a static sink, it is constantly moving with a pre-planed path in the fully mobile network. The movement pattern of the SINK is replica to the mobile node in which described in the previous paragraph and this pre-planned path is stored in each of the sensor nodes. However, due to the ocean current, the SINK deviates from its pre-planned path in such a way that, for any given pair of consecutive waypoints, X and Y, instead of moving from X to Y directly, the SINK travels from X to Z to Y. The position of point Z is $d_{\max}$ away perpendicularly from the midpoint of $\overline{XY}$. Although the deviation path is assumed to be linear, the important factor used in deviation calculation is actually the duration that the SINK has deviated from its preplanned path not the actual position. Also, since the direction of deviation can be changed randomly as discussed above, the deviation path can be considered as a random walk.

In order to easily interpret and understand the behavior of the protocols under different settings, we eliminate the effects of the MAC layer by allowing only one packet in the network at any instant. For each packet, the source node is selected randomly among the N nodes.

In our simulations, we evaluate the protocols' performance by varying the following parameters: sector size, node density, and node's speed. The routing performance's metrics that we have used are: the PDR, which is defined as the ratio of the number of unique *DATA* packets that are successfully received at the SINK to the total number of *DATA* packet transmissions; and the average packet delay, which is defined as the average duration that a packet takes to travel from

the source to the SINK.

5.5 Simulation Results

5.5.1 Performance of SBR in Mobile Networks with Static Sink

In this sub-section, we choose to benchmark the SBR with a single-sector routing scheme, so that we can evaluate both schemes under the same parameter settings. Specifically, the single-sector scheme is a replica of the SBR, except that it allows a sender node to pick its next relay node from one sector only (Sector 1 in Figure 5.1). In order to make the comparison as fair as possible, the sector size used in both schemes are the same, i.e., each sector has an angle of $\theta_k = 360°/k$, where k is the number of sectors that the SBR uses.

5.5.1.1 The Effects of Sector Size

As can be seen in Figure 5.2, in a sparse network (where $N = 10$ and 20), when the number of sectors is varied from 3 to 10 in which the sector size θ_k decreases from 120° to 36°, both schemes have better PDR when the sector size is large. In addition, the SBR achieves a higher and much more stable PDR when compared with the single-sector scheme. It is quite intuitive for the single-sector scheme to have higher PDR when the sector size is large because this can enhance the chances of having neighbors to relay the packet. For the SBR, although one might think that the sector size does not have any impact on the PDR since the total area covered by the communication range remains the same regardless of the sector size, it should be realized that the sector size could actually affect the choice of the relay node. Recall that the SBR gives higher priority to a sector

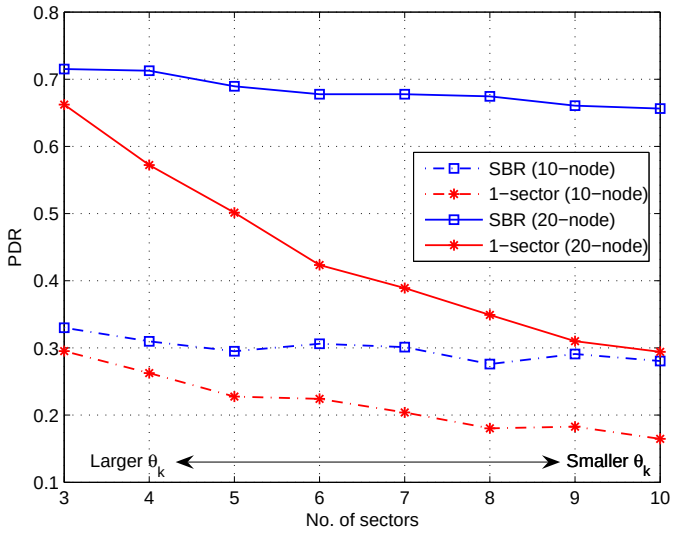

Figure 5.2: Effects of sector size on the PDR.

that has a smaller index. Hence, if there is at least one candidate node within Sector j that meets the criteria described in Section 5.2.3, it will be picked as the relay node, even if there were another node within Sector $j+1$ that is closer to the SINK. However, as the sector size becomes larger, there will be lower chances of picking a suboptimal node as the relay node arising from sector prioritization, because there are now more nodes with the same sector priority. This explains why the SBR also sees a higher PDR in general when the sector size is large.

Figure 5.3 shows the average packet delays of both schemes. In general, the delay increases when the sector size decreases, except for the single-sector scheme corresponding to the 10-node case, which will be explained at the end of this sub-section. For the single-sector scheme corresponding to the 20-node case, the average delay increases rapidly as the sector size decreases. This is because a smaller sector size reduces the likelihood that a suitable relay node could be found within the sector, which in turn causes the single-sector scheme to have to frequently wait for the topology to change before making another attempt. For

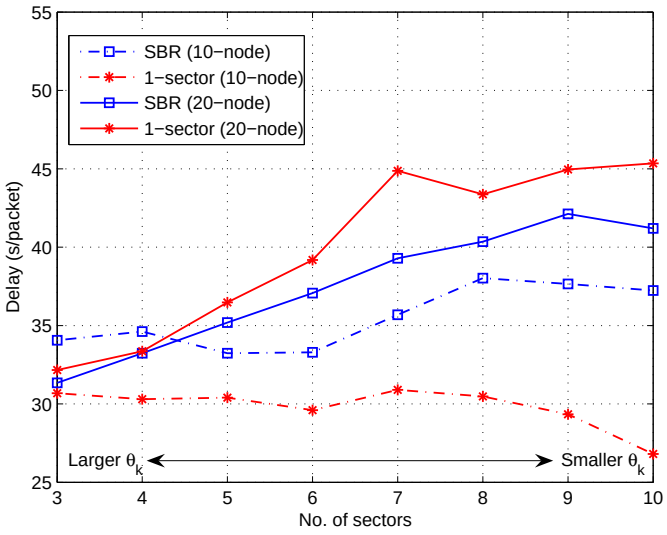

Figure 5.3: Effects of sector size on the delay.

the SBR, the reasoning behind the observed trend is different; the relay node is picked from the entire communication range, hence a smaller sector size does not reduce the likelihood that a suitable relay node could be found. Here, the delay increases when the sector size decreases mainly because the sender node takes longer time to collect all the *Chk_Ngh_Reply* responses from its neighbors when there are more sectors.

As mentioned earlier, the delay behavior of the single-sector scheme corresponding to the 10-node case deviates from the general trend. This can be explained as follows. The combination of a sparse network with small sector size results in very low PDR (as seen in Figure 5.2); consequently, most of the packets that are successfully routed to the SINK are those that were generated very close to the SINK, which lead to very short average delays.

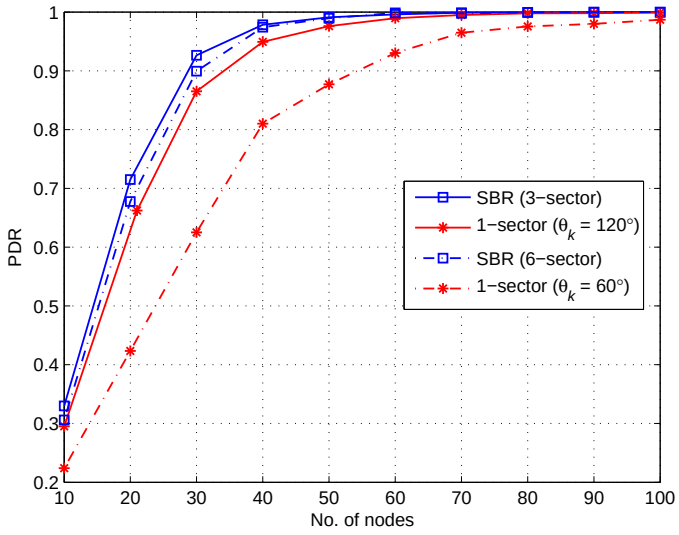

Figure 5.4: Effects of node density on the PDR.

5.5.1.2 The Effects of Node Density

Unsurprisingly, Figure 5.4 illustrates that a higher node density results in a higher PDR. This is because of the higher probability that there would be a potential candidate node within the sender's communication range for the case of the SBR, or within the sector of interest for the case of the single-sector scheme. When the sector size is small, the SBR (6-sector) performs significantly better than the single-sector scheme ($\theta_k = 60°$) when N ranges from 10 to 70, while both schemes have comparable PDR in a denser network where N is more than 70. A similar trend is also observed when we compare the results for the larger sector size, albeit not as significant as the previous case.

Figure 5.5 shows that the average packet delay decreases in general as the node density increases. In the SBR, a higher node density leads to a higher chance that Sector 1 contains a suitable relay node, which is more desirable. When the packets are routed more often through relay nodes that are chosen from Sector 1, the paths would be generally shorter, which in turn result in smaller delays.

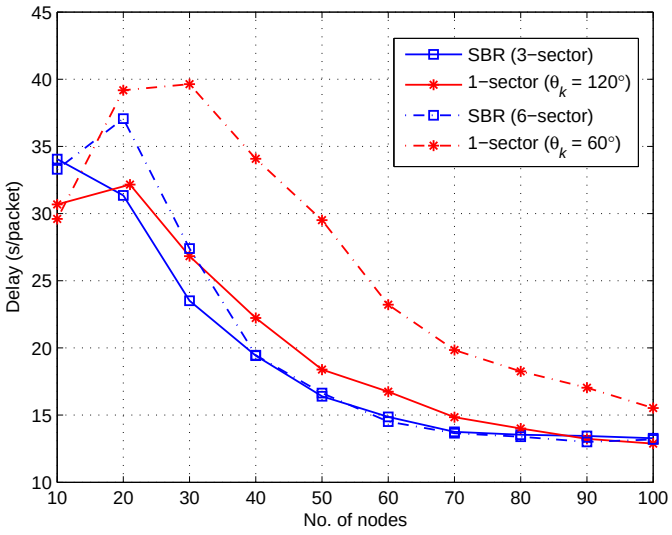

Figure 5.5: Effects of node density on the delay.

Moreover, a higher node density also implies that it is more likely to find a relay node that is closer to the SINK. In order to confirm both claims, let us look at the plots of the SBR for both 3-sector and 6-sector cases. When the number of nodes increases from 20 to 40, the delay of both cases decreases, but the delay in the 3-sector case is smaller due to the larger size of its Sector 1. Beyond N=40, the delay of both cases merge due to the second reason. These explanations also apply to the improvement of the single-sector scheme's delay performance as the node density increases.

Notice that when N=10, both schemes have smaller delay than when N=20. This deviates from the general trend that we have discussed above. This is because when the network is very sparse, those packets that are generated several hops away from the SINK have high probability of being discarded eventually and do not contribute towards the average delay calculation. Thus, most of the packets that are routed successfully to the SINK are those that were generated nearby, and would naturally experience lower delay.

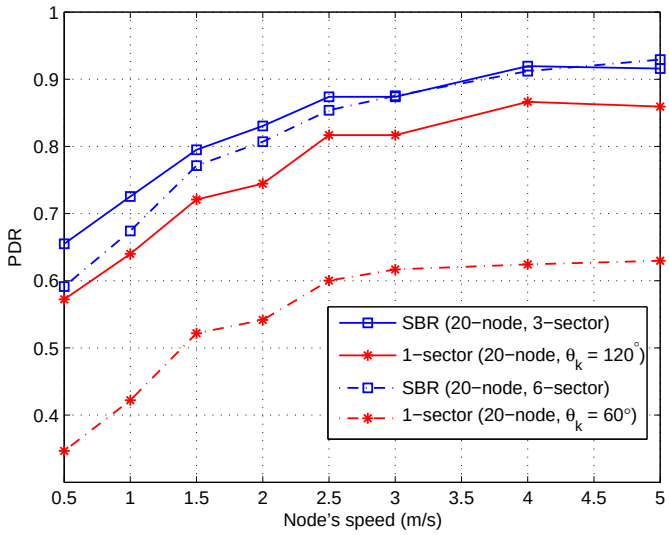

Figure 5.6: Effects of node's speed on the PDR.

5.5.1.3 The Effects of Node Mobility

Next, we study the effects of node mobility by varying the node's speed from 1-5 m/s. For a chosen speed, all the nodes move with the same speed throughout, while their directions are still kept random.

We can see from Figure 5.6 that node mobility actually helps to enhance routing in sparse networks for both schemes, as the PDR increases when node's speed increases. In general, node mobility can be either an advantage or a disadvantage for routing protocols, depending on their respective designs. On one hand, the change in topology caused by node mobility may harmfully cause the network to be disconnected; on the other hand, it may beneficially allow the network to be reconnected. For those protocols that do not address node mobility adequately, they experience both effects, and the end result depends on which factor plays a more significant role. In our case, the SBR takes node mobility into account during the process of finding its potential candidate. Thus, it only benefits from the change in topology caused by node mobility.

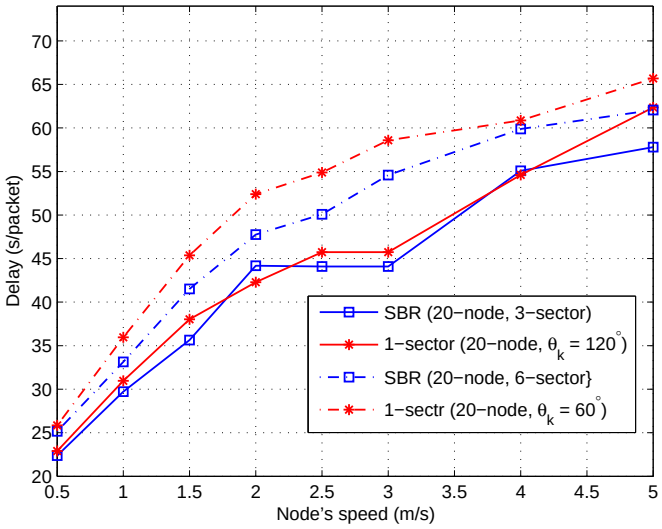

Figure 5.7: Effects of node's speed on the delay.

Figure 5.7 shows that the packet delay increases as the node's speed increases for both the SBR and the single-sector scheme. In order to understand this better, recall from Figure 5.6 that more packets are routed successfully to the SINK as the node's speed increases. In other words, those packets that cannot be routed in a low-mobility network can now be routed in a high-mobility network. Because we have assumed that the channel is error-free, the main reason that a packet is not successfully routed to the SINK is that it gets dropped by an intermediate node when the latter cannot find any relay node after n_{discard} attempts. For a sparse network, this problem can be alleviated by the change of network topology caused by node mobility. However, when such packets are successfully routed to the SINK, their packet delays also include the time spent by the intermediate relay nodes waiting for the topology to change. These longer delays now contribute towards the average delay calculation.

5.5.2 Performance of SBR and SBR-DLP in Fully Mobile Networks

In this study, in order to obtain the location of the SINK, SBR relies solely on the original pre-planned path of the SINK while SBR-DLP utilizes both the destination's pre-planed path and the destination location prediction scheme. We keep all other parameters the same for both schemes. The results in this study provide us the understanding the gain in performance resulting from the use of destination location prediction.

5.5.2.1 The Effects of Sector Size

Figure 5.8 shows that the performance of both the SBR-DLP and the SBR are rather independent of the number of sectors, as their PDR are quite stable with respect to the number of sectors. Note that the SBR-DLP and the SBR are equivalent when $d_{\max}$ is 0, because the destination node always conforms to its pre-planned path. From the figure, we can also see the improvement in PDR when location prediction is introduced. By comparing the plots from the SBR-DLP with the ones from the SBR, we can see that when the maximum deviation $d_{\max}$ is increased from 10 m to 100 m, the use of location prediction helps raise the PDR significantly for all $d_{\max}$. An interesting observation from the SBR's plots is that, although one may expect that a higher deviation $d_{\max}$ would result in a lower PDR, it is noted that the SBR's performance does not decrease further when $d_{\max}$ changes from 50 m to 100 m. This can be explained by focusing our attention on the movement of the SINK. Keeping in mind that $d_{\max}$ is the amount of the SINK's deviation from its pre-planned path. When the deviation is large enough to cause routing failure, increasing the deviation further would still result in the same routing failure, without causing much change to the PDR.

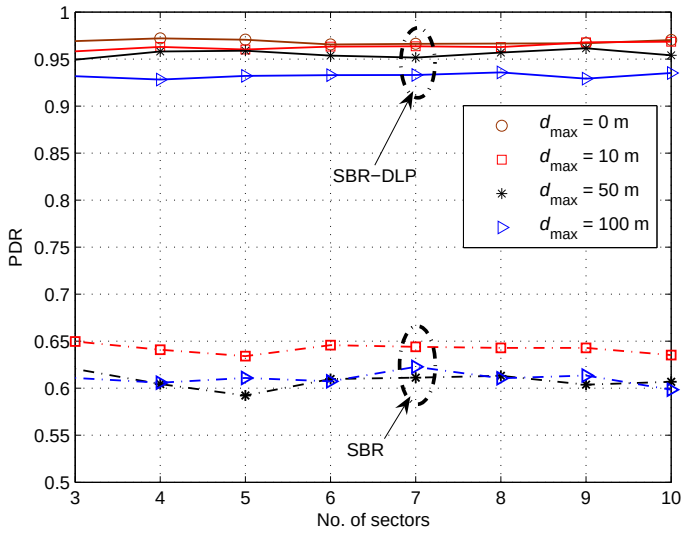

Figure 5.8: Effects of number of sectors in a 30-node network, for SBR (dashed) and SBR-DLP (solid).

Note that the PDR decreases with the increasing sector size when the SINK is static (look at Fig. 5.6) while the PDR remains almost constant regardless of the number of sectors when the SINK is mobile (as shown is Fig. 5.8). This two opposing results arise from the difference in node density used in the two networks. The result shown in Fig. 5.6 is obtained from a 20-node network while the result shown in Fig. 5.8 is obtained from a 30-node network. With low node density, increasing the sector size leads to a better PDR as discussed in Section 5.5.1.1. However, when node density is high, increasing the sector size does not impact the PDR since there is always a route to the SINK found within the first sector.

5.5.2.2 The Effects of Node Density

Next, we study the effects of node density on both the SBR-DLP and the SBR. As shown in Fig. 5.9, both algorithms exhibit similar trends when the number of nodes in the network is varied from 10 to 40 nodes. Unsurprisingly, we notice

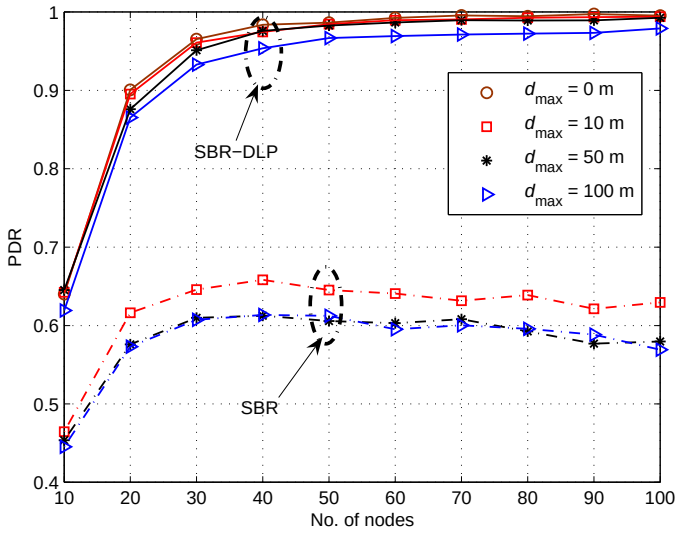

Figure 5.9: Effects of number of nodes in a 6-sector network, for SBR (dashed) and SBR-DLP (solid).

that the PDR increases dramatically as the number of nodes increases in this range. For the SBR-DLP, the PDR starts to get saturated when the number of nodes exceeds 40 nodes, likely because network disconnectivity has now become rare.

On the other hand, the SBR's performance degrades slightly when the number of nodes exceeds 40 nodes. Without destination location prediction, the SBR always relies on the SINK's pre-planned path, which is no longer accurate. However, a low density network may not be affected as much by the deviations in the SINK locations, compared to a high density network. This is because, for higher density networks, there are also higher chances that a sender would pick a relay node that is much closer to both the destination and the virtual vector $\overline{SD}$. Thus, for such networks, the effects of inaccurate SINK locations are naturally more pronounced. To support this claim, let us compare the SBR with the SBR-DLP for both low and high density regions in Figure 5.9. As can be seen, the

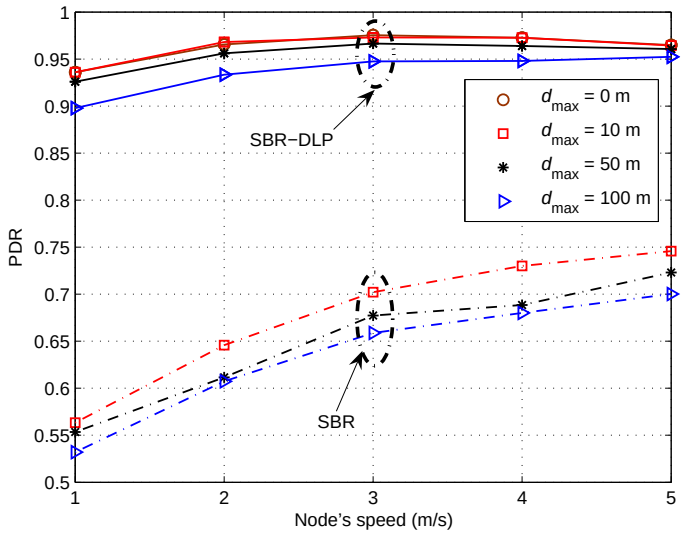

Figure 5.10: Effects of node speed in a 30-node and 6-sector network, for SBR (dashed) and SBR-DLP (solid).

performance gain from the use of destination location prediction is much more significant in the higher density region.

5.5.2.3 The Effects of Node Mobility

Now, let us look at Figure 5.10 to examine the effects of node speed on the performance of both the SBR-DLP and the SBR. Here, we only focus on the 30-node network, since it has been shown to be dense enough to illustrate the significant gains brought by destination location prediction. As expected, the SBR-DLP outperforms the SBR significantly in all cases. It can also be seen that the PDR of both the SBR-DLP and the SBR improves as the node speed increases. In general, node mobility can be both advantageous and disadvantageous to routing protocols. On the one hand, the change in topology caused by node mobility may harmfully cause the network to be disconnected; on the other hand, it may beneficially allow the network to be reconnected. For those protocols that take node

mobility into account adequately, they can make the advantages outweigh the disadvantages. In our case, both the SBR-DLP and the SBR take node mobility into account during the process of finding the next relay node. Thus, they benefit more from the change in topology caused by node mobility, which explains why their PDR increases with node speed.

Although the results shown above correspond to the use of the SBR and the SBR-DLP in 2D networks, similar trends also apply if they were to be implemented in 3D networks. In a 3D network, the communication circle and the sector become a communication sphere (assuming omni-directional antenna) and a spherical wedge, respectively. As long as a node can locate itself within 3D space, there is no burden scaling from 2D to 3D networks for these protocols.

5.6 Conclusion

In this chapter, we present two location-based routing protocols for mobile UWA networks, the SBR and the SBR-DLP. Both protocols are suitable for both sparse and dense networks. The designs take into account the unique characteristics of such networks, namely, long propagation delay, node mobility, and low data rate. In addition, both protocols consider multiple sectors, at the same time, in the process of finding its relay node which is more efficient than considering only a single sector at a time. Also, SBR and SBR-DLP are highly adaptive to network dynamics, such as nodes joining and leaving the network, because a node find its next relay node reactively on a hop-by-hop basis. For fully mobile networks, our studies show that it is essential to account for the mobility of the destination node. However, even a simple location prediction mechanism could help improve the packet delivery ratio significantly.

For applications that have a static sink, such as ocean data collection in a fixed area, SBR is more suitable than SBR-DLP since it can achieve the high PDR while keeping the number of overhead small. On the other hand, SBR-DLP is more efficient, although at the cost of higher overheads, for fully mobile UWA networks where the destination nodes can also move along with other nodes in the network such as in the application of undersea animal tracking.

Chapter 6

Time Synchronization in UWA Networks

6.1 Introduction

The clock synchronization problem has drawn considerable attention from researchers in the past few decades, especially in the area of wireless sensor networks, due to its wide variety of possible applications in which it can be integrated such as environmental monitoring, target tracking, security surveillance, and many more. In wireless sensor networks, each node performs its task (e.g., sensing the environment) in a distributed manner; the often time-sensitive data from multiple sensor nodes are then aggregated and converted to more meaningful information by using techniques such as data fusion. For example, in a target tracking application, while the interest object is moving, sensors in different areas sense the object and report the presence of the object in its local vicinity. These distributed reports can then be fused to extract information such as the speed and the direction of the moving object.

Most of the applications in sensor networks require that all sensor nodes have a common time (e.g., all synchronized), so that they can coordinate and collaborate with each other in order to accomplish their tasks. Basically, we

can classify these applications into three categories based on the synchronization level required [52, 63]. Some applications merely require the order of the event occurrences, while there may be other applications that require the time interval of each of the event occurrences. Yet, there may also be applications that require the absolute time at which each event occurs. In addition, not only would the application layer find time-synchronization useful; other layers, such as MAC and networking layers, may also benefit from time synchronization. For example, PCAP [19], APCAP [24] and the Slotted FAMA [26] are examples of MAC protocols that require time synchronization.

Because of the usefulness of time synchronization, numerous synchronization algorithms have been recently proposed for terrestrial wireless sensor networks; however, none of these can be directly applied to UWA networks for several reasons. Firstly, the previously proposed algorithms are designed for high speed radio communication, and they typically assume that the propagation delay is negligible. In contrast, underwater communication mainly uses acoustic channel with a low propagation speed, thus resulting in significantly longer propagation delay [62]. Secondly, the terrestrial synchronization algorithms typically do not worry much about the re-synchronization frequency. In contrast, synchronization overhead is an important issue in UWA networks, due to its low data rate resulting from its narrow available bandwidth. Thus, the synchronization algorithm should be able to maintain a certain accuracy without the need for frequent re-synchronization, in order to avoid excessive consumption of the traffic capacity. Furthermore, when re-synchronization is required, the overhead incurred should not degrade the system performance.

Moreover, the nodes in underwater sensor networks tend to exhibit some degree of mobility. Thus, the synchronization algorithm must be able to cope

with the sensors' movement, which introduces time-varying delay.

In this chapter, we propose a cluster-based synchronization algorithm for mobile UWA networks, called "MU-Sync". Our design avoids frequent re-synchronization by estimating both the clock skew and the offset. As underwater mobile networks experience both time-varying and long propagation delay, we estimate the skew and offset by performing least square error linear regression [64] twice over a set of local time information gathered through message exchanges. With the help of MAC-level time stamping, we can further reduce the nondeterministic errors that are commonly encountered by those synchronization algorithms that rely on message exchanges.

The remainder of this chapter is organized as follows. In Section 6.2, we discuss possible causes of error typically found in time synchronization. We then describe the model of the clock in Section 6.3. In Section 6.4, we provide the details of the MU-Sync, as well as an analysis of the possible synchronization error. Next, Section 6.5 describes the detail of the simulations that were carried out to compare the performance of the proposed schemes with several others and the simulation results are shown and discussed in Section 6.6. Section 6.7 then provides a discussion of the algorithm, and finally, we give our conclusions in Section 6.8.

6.2 The Causes of Error in Time Synchronization

In any clock synchronization algorithm, an error may still exist even at the instance immediately after the synchronization. As time progresses, this error grows with time, and re-synchronization is hence required. In order to avoid frequent

re-synchronization, the error should be minimized. In this section, we explore the possible causes of error, and ways to reduce it. Specifically, we can divide them into two categories as will be discussed next.

6.2.1 Errors Caused by Uncertainty of Message Delivery Time

Many existing synchronization algorithms often utilize the technique of message exchange between synchronizing nodes, in order to acquire their local clock drift. By utilizing the message exchange technique, the common sources of error for clock synchronization, (first introduced by Koeptz and Schwab [65, 66]), come from the uncertainty of the following:

- **Send time**: The time used to construct the message and send the request to the MAC layer. It is non-deterministic and also dependent on the current load as well as the operating system. The error arising from the send time can be minimized by utilizing MAC-layer time stamping at the sender side.

- **Access time**: The delay incurred while waiting to access the channel until the transmission begins. The amount of access time depends on the current network traffic and the nature of the running MAC protocol. Physical layer time stamping can be used to eliminate this error.

- **Propagation time**: The time it takes to transmit the message from the sender to the receiver. The propagation time is highly deterministic depending on the distance between the sender and the receiver. In terrestrial sensor networks, it is often considered as a negligible contribution of synchronization error due to the high speed of radio wave. However, when

dealing with UWA networks, this becomes a major cause of synchronization error, as will be discussed in Section 6.4.3.

- **Receive time**: The time it takes to process the notification of an incoming message.

- **Encoding and Decoding time**: The time it takes it needs to encode/decode the packet at the physical layer. This is deterministic[1] and is in the order of hundred of microseconds [56].

6.2.2 Other Causes of Synchronization Errors

Even when the clocks of two nodes are perfectly synchronized at the beginning, their clocks may drift if inconsistency occurs due to changes of the surrounding environment, such as when the nodes experience changes in temperature, pressure, battery voltage, etc. As underwater sensor nodes typically exhibit some mobility, it is highly likely that they encounter changes in the abovementioned parameters, and hence require re-synchronization more often than static terrestrial sensor networks. Moreover, the clock can also be affected by the interaction of other components of the sensor system. For example, the sensor may miss an interrupt while busy transmitting or receiving a packet, as described in [67].

6.3 Clock Drift Modeling

The timer inside each clock usually uses a crystal oscillator operating at a certain angular frequency, which determines the rate at which the clock runs [63]; this is also widely referred to as the clock's "skew". Typically, each clock may have

[1] To estimate encoding and decoding time accurately, one needs hardware support and appropriate time-stamping mechanisms (including sending time stamp of earlier transmitted packets in later packets, after the time of sending is accurately known)

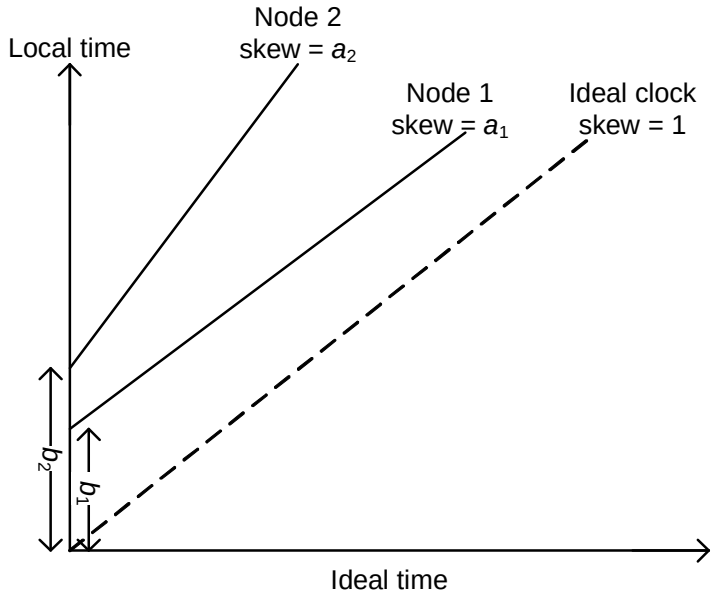

Figure 6.1: Illustration of clock drifts.

slight differences in frequency due to the manufacturing process. These skew differences cause the drift among the sensor nodes. In general, we often model the local time of node i using two parameters, namely, its skew and its offset, as follows:

$$T_i(t) = a_i t + b_i, \tag{6.1}$$

where a_i and b_i are the skew and the offset of node i, and t is the ideal time or *Universal Time Coordinated* (UTC). (6.1) can also be written as $T_i(t) = a_i(t)t + b_i$ for a clock that has a time varying skew. However, skew variation is assumed to be small in this study, thus, we use (6.1) as our clock model. The offset arises when each sensor node has a different starting time. Fig. 6.1 shows the drift of two clocks having different skews and offsets. Note that, while the offset causes constant error independent of time, the skew can cause increasing error as time progresses. Thus, in order to avoid the need for frequent resynchronization, the synchronization algorithm must be able to accurately estimate both the clock

skew and the offset.

6.4 MU-Sync

6.4.1 Overview of MU-Sync

The MU-Sync is designed to minimize the drift between nodes by estimating and compensating both the skew and the offset using a two-phase operation, namely, the *skew and offset acquisition phase*, and *the synchronization phase*. In the first phase, the clock skew and offset is estimated by applying linear regression twice over a set of n reference beacons. While all of the existing synchronization algorithms perform linear regression only once to retrieve the estimated skew, the MU-Sync performs it twice. The first regression allows the cluster head to extract the amount of propagation delay that each reference (REF) packet encounters. After adjusting the REF beacons' timings with their respective propagation delays, a second linear regression is performed over this new set of points, from which the estimated skew $(\hat{\hat{a}}_y)$ and offset $(\hat{\hat{b}}_y)$ of node y can be obtained.

Since the MU-Sync is cluster-based, it can easily be applied to mobile multi-hop UWA networks. In contrast to the TSHL [53], in which each node computes its own skew and offset, the cluster head in the MU-Sync takes the responsibility to start the synchronization process and calculate its neighbors' skew and offset.

6.4.2 Details of MU-Sync

In Phase 1, the synchronization process starts with the cluster head (also referred to as Node x throughout the rest of this section) broadcasting the i^{th} REF packet to its neighbors at time $T_{1,i}$, as shown in Fig. 6.2. Upon receiving the REF packet,

the neighboring node marks its local time as $T_{2,i}$ and responds to the cluster head at time $T_{3,i}$, informing it about the $T_{2,i}$ and $T_{3,i}$ timestamps. Note that, in order to reduce the chances of a collision, each neighboring node may introduce some small random interval before responding to the REF packet. When the cluster head receives the response from its neighboring node y at time $T_{4,i}$, it waits for some duration denoted as REF_TX_INT before transmitting the $(i+1)^{\text{th}}$ REF packet, and continues the same procedure until it has reached the number of required REF packets, n, or until the error of the linear regression is below a certain threshold. Next, the cluster head performs the first linear regression over the set of local times reported by node y to obtain its first estimated skew $(\hat{a_y})^2$, as illustrated in Fig. 6.3. The value of $\hat{a_y}$ is then used to compute the amount of one-way propagation delay that each REF packet has encountered, using the local time stamps $T_{1,i}$, $T_{2,i}$, $T_{3,i}$ and $T_{4,i}$:

$$\hat{p}^{x\to y}_{t_1\to t_2,i} = \frac{1}{2}\left[T_{4,i} - T_{1,i} + \frac{T_{2,i} - T_{3,i}}{\hat{a_y}}\right] \tag{6.2}$$

where $p^{x\to y}_{t_m\to t_n,i}$ denotes the propagation delay between node x's location at time t_m and node y's location at time t_n of the i^{th} REF packet, where $1 \le i \le n$. We next subtract the estimated propagation delay corresponding to each of the data points to obtain a new set of data points. The cluster head then runs the second linear regression to obtain the final estimated skew and offset of neighboring node y, denoted by $\hat{\hat{a}}_y$ and $\hat{\hat{b}}_y$, respectively.

In the synchronization phase shown in Fig. 6.4, the cluster head broadcast all neighbors' $\hat{\hat{a}}_y$ and $\hat{\hat{b}}_y$, so that every neighbor can keep track of these parameters. When every node in the cluster knows the skew and the offset of every other node in the cluster, we can claim that cluster-wide synchronization has been achieved.

[2] An example of how linear least squares regression is computed can be found at [64]

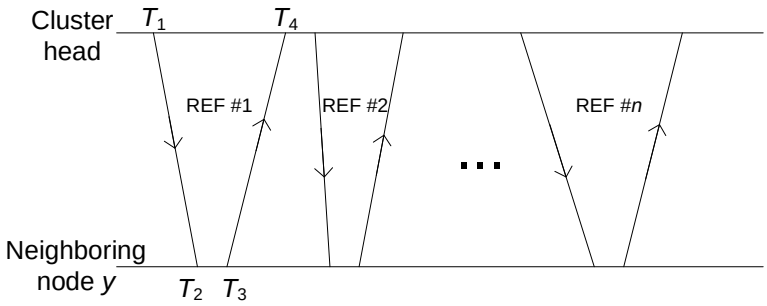

Figure 6.2: Phase 1: Skew and offset acquisition phase. Note that the propagation delay can vary during the REF packet exchange.

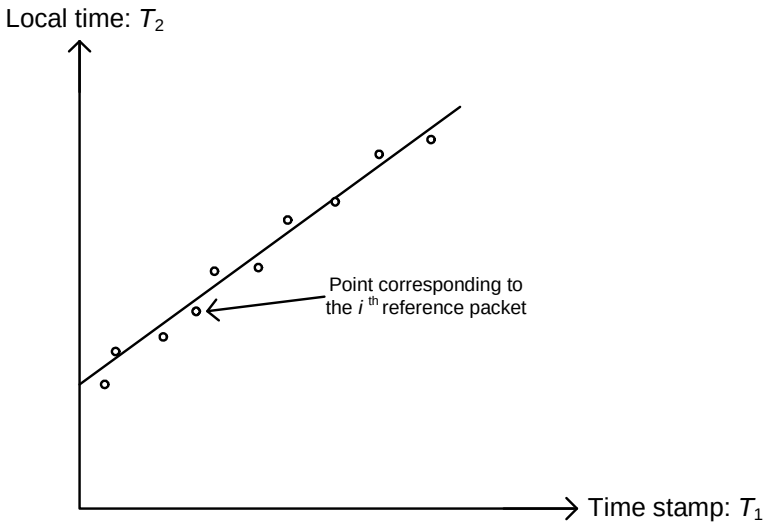

Figure 6.3: Linear regression at the cluster head with a total of 10 reference packet responses from node y.

6.4.3 Error Analysis of Propagation Delay Estimation

Now, let us assume that the clock of node x can be modeled by using its skew and offset relative to an ideal clock in a similar form as (6.1). The following set

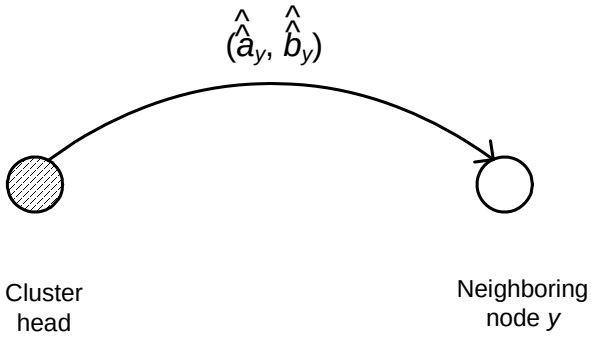

Figure 6.4: Phase 2: Synchronization phase.

of equations of the node's local time can hence be derived:

$$T_1 = a_x t_1 + b_x \tag{6.3}$$

$$T_2 = a_y t_2 + b_y \tag{6.4}$$

$$T_3 = a_y t_3 + b_y \tag{6.5}$$

$$T_4 = a_x t_4 + b_x \tag{6.6}$$

The ideal time of t_2 and t_4 can also be written as

$$t_2 = t_1 + p^{x \to y}_{t_1 \to t_2}, \tag{6.7}$$

and

$$t_4 = t_3 + p^{y \to x}_{t_3 \to t_4}. \tag{6.8}$$

Our objective is to estimate the one-way propagation delay $p_{t_1 \to t_2}^{x \to y}$ from the round trip time[3], which can be written as

$$p_{t_1 \to t_2}^{x \to y} = \frac{(t_2 - t_1) + (t_4 - t_3)}{2}, \tag{6.9}$$

By substituting (6.7) and (6.8) into (6.2), the estimated propagation delay $\hat{p}_{t_1 \to t_2}^{x \to y}$ computed at Node x is

$$\hat{p}_{t_1 \to t_2}^{x \to y} = \frac{1}{2}\left[a_x(t_4 - t_1) - \frac{a_y}{\hat{a}_y}(t_3 - t_2)\right]. \tag{6.10}$$

Since we are interested in the relative drift of node y compared to node x, we let $a_x = 1$ which results in

$$\hat{p}_{t_1 \to t_2}^{x \to y} = \frac{1}{2}\left[(t_4 - t_1) - \frac{a_y}{\hat{a}_y}(t_3 - t_2)\right] \tag{6.11}$$

$$= \frac{1}{2}\left[(1 - \frac{a_y}{\hat{a}_y})(t_3 - t_1) + p_{t_3 \to t_4}^{y \to x} + \frac{a_y}{\hat{a}_y} p_{t_1 \to t_2}^{x \to y}\right] \tag{6.12}$$

As $p_{t_1 \to t_2}^{x \to y}$ is an average of the propagation delay obtained from $t_4 - t_3$ and $t_2 - t_1$, while the actual delay is $t_4 - t_3$, we can calculate the error of propagation delay estimation (Δ) as

$$\Delta = \left|\hat{p}_{t_3 \to t_4}^{y \to x} - p_{t_3 \to t_4}^{y \to x}\right| \tag{6.13}$$

$$= \frac{1}{2}\left|(1 - \frac{a_y}{\hat{a}_y})(t_3 - t_1) - p_{t_3 \to t_4}^{y \to x} + \frac{a_y}{\hat{a}_y} p_{t_1 \to t_2}^{x \to y}\right|. \tag{6.14}$$

Equation (6.14) indicates that the error of the propagation delay estimation depends on three parameters:

[3]Note that MU-Sync assumes that there is no motion of mobile nodes during the exchange of the REF packet. However, motion of mobile nodes for multiple REF exchanges is taken into account by calculating $p_{t_1 \to t_2}^{x \to y}$ for each of every data point obtained from each REF packet exchanges.

1. The relative drift between the estimated and the real skew: $\frac{a_y}{\hat{a_y}}$
2. The time interval $t_3 - t_1$
3. The value of $\left| p^{y \to x}_{t_3 \to t_4} - \frac{a_y}{\hat{a_y}} p^{x \to y}_{t_1 \to t_2} \right|$

We can rewrite (6.14) as

$$\Delta = \frac{1}{2} \left| (1 - \frac{a_y}{\hat{a_y}})(t_3 - t_1) - \frac{v_{\text{node}}(t_4 - t_3)}{v_{\text{s}}} + \frac{a_y}{\hat{a_y}} \frac{v_{\text{node}}(t_2 - t_1)}{v_{\text{s}}} \right|, \qquad (6.15)$$

where v_{node} and v_{s} are the relative speed between node x and node y and the speed of sound in underwater, respectively. Currently, mobile nodes (e.g., AUVs) can travel at the maximum velocity of v_{node}= 2 m/s and the typical value for v_{s} is 1500 m/s [13]. After obtaining the propagation delay, the cluster head is now able to estimate node y's offset, $\hat{b_y}$, by first deducting the propagation delay effect and running the linear regression again to obtain both $\hat{\hat{a_y}}$ and $\hat{\hat{b_y}}$.

6.5 Simulation Model

In our simulation setup, the sensor nodes are allowed to move randomly within an area of 1000 m by 1000 m. We assume that the speed of sound in underwater is constant at 1500 m/s, and there is no skew variation arising from a change in the environment as previously discussed in Section 6.2.2, so that we can concentrate solely on the effect of the parameters that we are interested in. The nondeterministic errors encountered during the message exchange is modeled using Gaussian distribution, as suggested by Elson and Estrin [51]. Unless specified otherwise, we use the following set of parameters for our simulations:

- Maximum speed of the sensor node (V_{max}) is 2 m/s.

- Clock skew is 40 ppm.
- Clock offset is 10 ppm.
- The number of beacons used to perform linear regression is 25.
- The duration $t_3 - t_2$ is 0 s.
- The time interval between two successive reference packets is 5 s.
- The sensor nodes change its speed randomly within the range of $[0, V_{\max}]$, with an average interval of 600 s.
- The sensor nodes change its direction randomly within the range of $[-45°, 45°]$, with an average interval of 600 s.
- Clock granularity is 1 μs.
- Receive jitter is 15 μs.

We study the following parameters in order to investigate their effects on the MU-Sync's performance:

1. The node's initial skew
2. The number of beacons
3. The duration of $t_3 - t_2$
4. The frequency at which the sensors change direction
5. The speed of the sensors

We choose to benchmark our scheme with the TSHL [53], as well as a network that does not undergo any synchronization. Although the TSHL is designed for

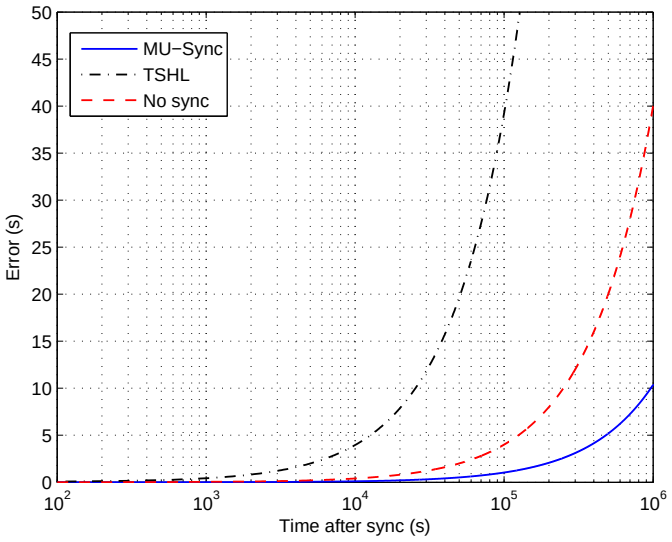

Figure 6.5: The error in time estimation VS the time elapsed since synchronization.

static underwater sensor networks, which assume long but constant propagation delay, it is the closest form of underwater synchronization scheme available so far.

6.6 Simulation Results

In all the results shown in this section, each data point is obtained from the average of 1000 simulation runs. The error bars in the figures represent the standard deviations. Fig. 6.5 shows how the error in time estimation grows for each scheme as time elapses since the last synchronization. As can be seen, the MU-Sync perform better than the TSHL significantly. In fact, the TSHL is even worse than the case where no synchronization is performed. Its poor performance arises from its poor accuracy in estimating the skew. This is due to its assumption that the inter-nodal propagation delay is constant during the skew estimation process. When a node moves, the propagation delay varies with time; hence, if

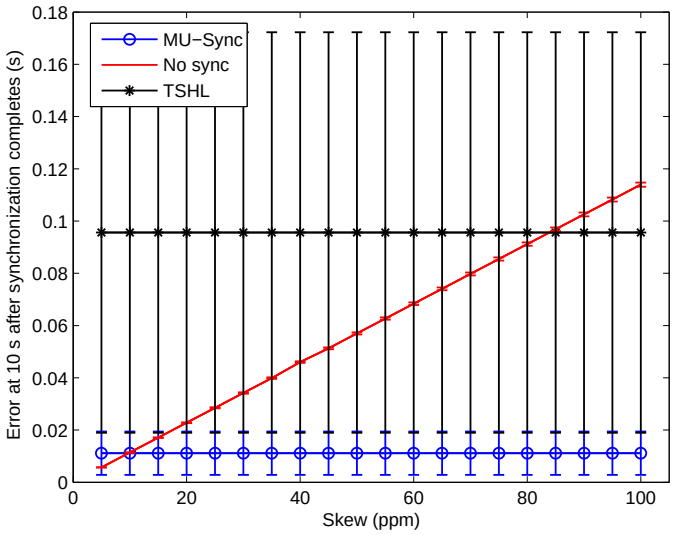

Figure 6.6: Effects of clock skew.

the linear regression is applied blindly without taking this into consideration, it causes inaccurate skew estimation. The small error in skew estimation can cause severe drift as time progresses. For example, the error can grow as large as 8 s within a day of operation even with a skew error that is as small as 0.0001.

When we vary the neighboring nodes' clock skew from 5-100 ppm, Fig. 6.6 shows that the MU-Sync's performance is independent of the node's initial skew. However, when the initial skew error is less than 10 ppm, the MU-Sync's performance is worse than the unsynchronized one. We can also see in Fig. 6.6 that both the TSHL and the MU-Sync achieve a constant average error regardless of the neighboring node's initial skew, because both algorithms estimate the skew and try to compensate for it. However, the TSHL's performance is much worse than even the unsynchronized version. The high standard deviation noticed from the error bars for the TSHL is due to the high variation of the propagation delay arising from node mobility. This happens when the linear regression is applied to estimate the skew over the set of points $\{T_{1,i}, T_{2,i}\}$ where $1 \leq i \leq n$. For the

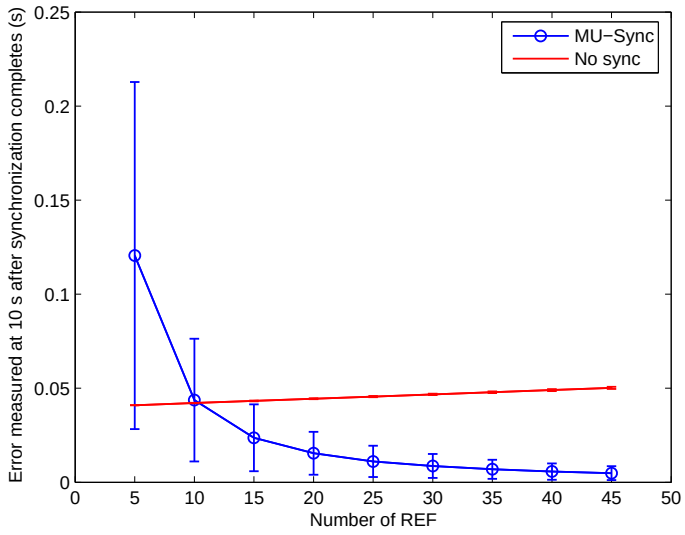

Figure 6.7: Effects of changing the number of REF beacons.

rest of this section, we decide to continue our study without the TSHL as thus far its performance is poorer than the unsynchronized version.

We next study the effect of the number of beacons on the synchronization error. It is obvious that a higher number of reference beacons used for linear regression will result in a lower error in estimation. Fig. 6.7 illustrates that a finer synchronization can be achieved by adjusting the number of reference beacons fed into the linear regression. With our cluster head synchronization approach, the MU-Sync can easily adjust the number of beacons used adaptively. Note that the larger the number of beacons to be collected for linear regression, the longer it takes to finish the synchronization process; this also explains why the slope of the unsynchronized version is linearly increasing as seen in Fig. 6.7.

The varying of the time interval $t_3 - t_2$ also affects the synchronization error. As shown in Fig. 6.8, when we vary the duration of $t_3 - t_2$ from 0 s (the neighboring node responds to the REF beacon right after it finishes receiving the packet) to 25 s (the neighboring node responds 25 s after it receives the REF),

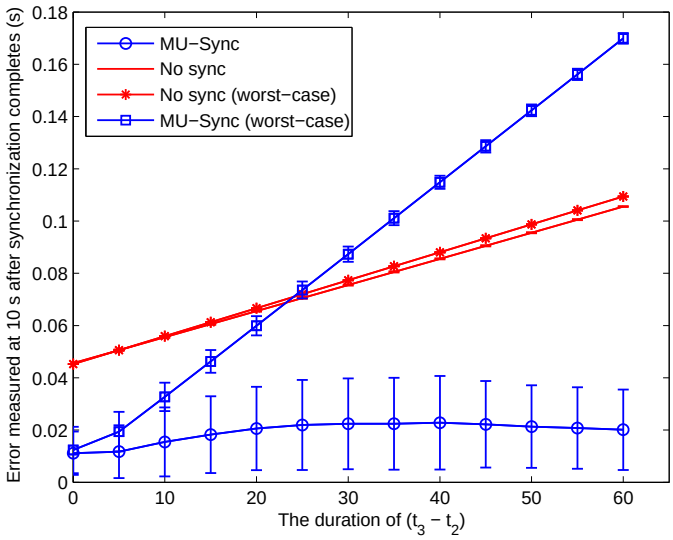

Figure 6.8: Effects of (t_3 - t_2).

the synchronization error of the MU-Sync increases with the duration of $t_3 - t_2$. However, when the interval of $t_3 - t_2$ goes beyond 25 s, the error tends to stabilize, or even decrease. The explanation is that the parameter $t_3 - t_2$ does not directly affect the synchronization error; instead, the relative distance between the cluster head and the neighboring node at time $t_1 \rightarrow t_2$ and $t_3 \rightarrow t_4$ plays a more important role. Since we are currently using half of the round trip time to estimate the one-way propagation delay, the estimation error depends on the value of $\left| p_{t_1 \rightarrow t_2}^{x \rightarrow y} - p_{t_3 \rightarrow t_4}^{y \rightarrow x} \right|$. We verify our claim by examining at the worst-case plot. Here, we notice that the error increases steeply as the duration of $t_3 - t_2$ increases. This effect is less significant when the nodes move in a more random manner.

Fig. 6.9 shows that changing the direction of the nodes frequently does not significantly affect the synchronization error. Here, we vary the average direction change interval from 10 s to 400 s.

Fig. 6.10 shows the impact of the sensor's speed on the synchronization error.

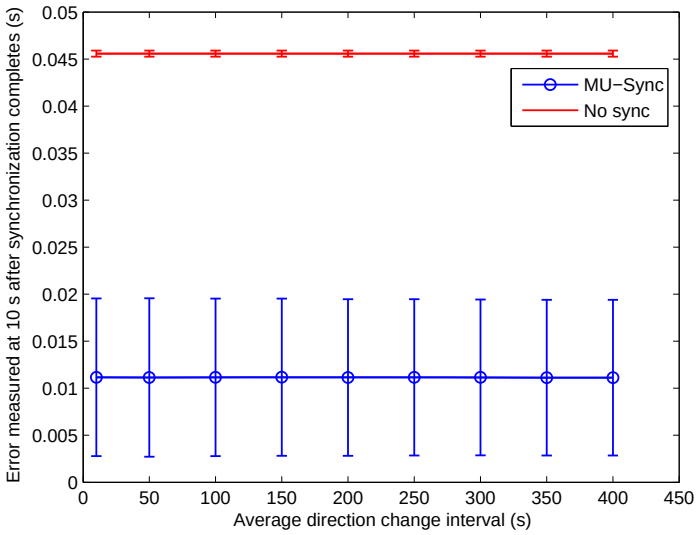

Figure 6.9: Effects of the average direction change interval.

As can be seen, the synchronization error increases with the parameter $V_{\max}$ (look at the worst-case MU-Sync plot). This is because, when the sensor is allowed to move very fast, the value of $\left|p_{t_1 \to t_2}^{x \to y} - p_{t_3 \to t_4}^{y \to x}\right|$ can be so large that using (6.12) to estimate the propagation delay is no longer accurate enough. Fortunately, in most networks, we would expect the nodes to undergo speed and direction changes over time, rather than persisting in the worst-case setting all the time. Therefore, the effect of $V_{\max}$ is, on the average, much less significant on the synchronization error (as indicated by the average MU-Sync plot).

6.7 Discussion

In this section, we discuss the applicability of the MU-Sync, and also how it may be improved. The comparison of the key performances between MU-Sync and TSHL is also given in Table 6.1.

- Although the MU-Sync seems to have a higher overhead than the TSHL, as

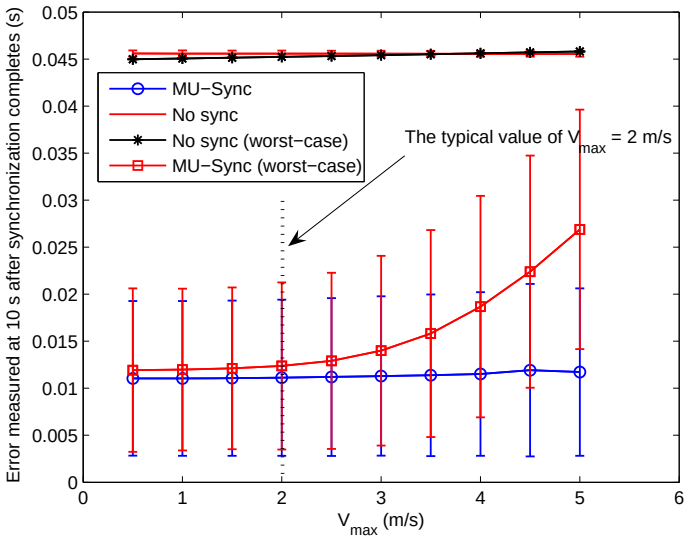

Figure 6.10: Effects of V_{max}.

Table 6.1: A performance comparison of MU-Sync and TSHL when the simulation parameters are set according to the setting presented in Section 6.5. Note that the overhead cost is computed based on j synchronized nodes and n REF packets while the resynchronization is triggered when the error exceeds 100 ms.

Key performance	MU-Sync	THSL
Achievable error	0.868 ms	38.7 ms
Overhead (no. of control packets)	$n + j + 1$	$n(1 + j) + 1$
Resynchronization frequency	every 9600 s	every 200 s

it requires the neighboring node to send a response for every REF packet received, it can easily be integrated with existing handshaking MAC protocols such as MACA [23], Slotted FAMA [26], MACA-MN [68], PCAP [19], etc., by piggybacking the REF and its response within the RTS/CTS packets. While the MU-Sync could achieve finer synchronization by having a higher number of beacons messages, the TSHL may not benefit from this approach when the nodes are mobile, since it does not account for time-varying propagation delay. In fact, since a higher number of beacon messages also

require a longer duration to finish collecting the beacons, the TSHL may become even more vulnerable to violating the constant propagation delay assumption.

- From our simulation, Fig. 6.8 (look at the worst-case) shows that the MU-Sync cannot cope when the duration of $t_3 - t_2$ is longer than approximately 25 s. The major factor that causes the error comes from the technique used to estimate the one-way propagation delay. Although (6.12) is widely used to calculate the one-way propagation delay from the round trip time, it may not be suited in our scenario. Since the nodes keep moving during the interval (t_2, t_3), significant error may be introduced when the propagation delay is estimated as half the round-trip time. A better method is hence needed in order to achieve higher accuracy.

- One advantage that the MU-Sync has over the TSHL is that, when the cluster head broadcasts its neighboring nodes' estimated skew and offset, every node learns the estimated skew and offset of all other nodes in the same cluster, instead of just the relative parameters between the cluster head and a particular node.

- Although we do not discuss how a cluster selects its cluster head in this chapter, the task can be achieved by applying an existing cluster head selection algorithm as presented in [69, 70, 71].

6.8 Conclusion

In this chapter, we proposed a cluster-based synchronization algorithm for UWA mobile networks, called "MU-Sync". Our design avoids frequent re-synchronization by estimating both the clock skew and offset. As underwater mobile networks

experience both time-varying and long propagation delay, previous works that estimate the clock skew using a single least square error linear regression tend to be inaccurate. In the MU-Sync, there are two phases of operation; (1) the *skew and offset acquisition phase*, and (2) *the synchronization phase*. In the first phase, the clock skew is estimated by performing the linear regression twice over a set of local time information gathered through message exchanges. The first linear regression enables the cluster head to offset the effect of long and varying propagation delay; the second regression in turn obtains the estimated skew and offset. With the help of MAC-level time stamping, we can further reduce the nondeterministic errors that are commonly encountered by those synchronization algorithms that rely on message exchanges. After obtaining the estimated skew and offset, the protocol precedes to the second phase in which the skew and offset distribution takes place.

We evaluate the performance of MU-Sync, comparatively to two other benchmarking schemes: TSHL and a network that does not undergo any synchronization, by varying the node's initial skew, the number of beacons, the duration of $t_3 - t_2$, the frequency at which the sensors change direction and the speed of the sensors. From the simulation results, we can conclude that time synchronizations that utilize message exchange technique must consider the time-varying propagation delay into account at the skew and the offset estimation process, in order to avoid large synchronization errors which lead to frequent re-synchronization.

Chapter 7

Conclusions and Directions for Future Work

This chapter is the dissertation conclusion with a review of the main research contributions and proposed directions for future researches.

7.1 Research Contributions

This dissertation addresses three important problems in UWA networks: (1) how to improve the normalized throughput performance via MAC design; (2) how to enhance the PDR when packets are routed in mobile UWA networks; and (3) how to reduce the time synchronization error in mobile UWA networks. We summarize our main research contributions as follows:

- In Chapter 3, we have studied the normalized throughput performance of different Aloha-based MAC protocol variants in single-hop underwater acoustic networks via both theoretical analysis and simulations. From this study, we found that the normalized throughput of Pure Aloha is unaffected by the long propagation delay in UWA networks. On the other hand, the

normalized throughput of Slotted Aloha, that usually performs much better than Pure Aloha in terrestrial networks, degrades to Pure Aloha in high latency networks. The Aloha-HD is shown to be able to outperform the Pure Aloha when the number of sensor nodes in the network are kept small (e.g., 4 nodes). For a very large number of nodes (e.g., 100 nodes), Aloha-HD's performance is comparable to that of Pure Aloha. Unsurprisingly, the study of NP-CSMA shows that its performance is better than that of Pure Aloha. However, the information obtained from channel sensing in the long propagation delay networks may not be as useful as it is in the terrestrial networks, in helping the node to avoid collision.

We have also demonstrated that with the knowledge of inter-nodal propagation delays, a node could help avoid collisions in a distributed manner by utilizing the sender-receive information that it picks up from overheard packets, provided it is within a bounded region defined by the relative positions of the sender and the receiver, as well as the packet's transmission time. This opportunity only arises in a long propagation delay environment such as UWA networks.

Based on this strategy, we proposed two Aloha-based random access MAC protocols, namely, Aloha-CA and Aloha-AN, for UWA networks. Between the two protocols, Aloha-CA is simpler and more scalable, as it only needs a small amount of memory, and does not rely on additional control messages. Aloha-AN, on the other hand, requires the use of additional NTF packets, which serve as advanced notification to the neighboring nodes, so that they can avoid transmitting packets that could result in collisions. Due to the need to select a suitable lag time for a given network setting, the scheme is less scalable as it needs to check whether its lag time is still appropriate

whenever there are any significant topology changes. However, the extra cost allows the Aloha-AN to achieve much better normalized throughput and collision avoidance.

- In Chapter 4, we proposed an asynchronous, handshaking-based MAC protocols for multi-hop UWA networks–RIPT. In order to improve the normalized throughput performance, RIPT utilizes the concept of multiple-node polling in which multiple nodes are allowed to transmit data packets to a single receiver within each round of handshake. Also, RIPT makes use of a 4-way handshake (RTS/SIZE/ORDER/DATA) and takes a more aggressive approach in reducing the number of collisions by utilizing a receiver-initiated reservation approach. As such, by allowing the receiver to request for the data transmission, a number of collisions can be greatly reduced because the uncertainty of the receiver's status, which can be perceived at the transmitter, is completely eliminated. This is confirmed through simulations that RIPT can achieve high and stable normalized throughput when proper values of packet train size (M_{train}), as well as average time between handshake initiations (T_{avg}) are chosen.

- In Chapter 5, we proposed the SBR, a hop-by-hop, location-based routing protocol that is suitable for both sparse and dense mobile UWA networks. Its design takes into account the UWA characteristics, namely, long propagation delay, node mobility, and low data rate. In SBR, the node only assumes rough knowledge of its own location and the destination's location which enables the node to select the next best forwarder efficiently when it is combined with our sector-based routing mechanism. In addition, the transmitter in SBR always takes node mobility into account when it chooses its next forwarder. By doing so, the packet drops that could have caused

from the two nodes (the transmitter and the receiver) moving away from each other's transmission range is completely eliminated, resulting in a better PDR. In this chapter, we also proposed the enhanced version of the SBR, namely SBR-DLP. In SBR-DLP, the need for precise knowledge in locating the destination node can be relaxed due to the use of the location prediction mechanism.

We have shown through simulations that routing designs for fully mobile UWA networks need to account for the mobility of the destination node. However, even a simple location prediction mechanism could help improve the PDR significantly. In addition to their superior performance, the SBR and the SBR-DLP are highly adaptive to network dynamics, such as nodes joining and leaving the network, because of their reactive hop-by-hop packet routing mechanism.

- In Chapter 6, we have presented a cluster-based synchronization algorithm for mobile UWA networks, known as MU-Sync. In order to minimize the need of frequent re-synchronization, MU-Sync performs time synchronization by estimating both the clock's skew and the clock's offset. We also find that the major contributor to time synchronization error in UWA networks is the long and time-varying propagation delay. Hence, unlike those existing synchronization schemes that either treat the propagation delay as negligible or constant, MU-Sync takes into account both characteristics in the process of the skew and the offset estimation. This consequently leads to a more accurate estimation and smaller synchronization errors. The extensive simulation results show that MU-Sync is able to reduce synchronization errors and outperforms all other benchmarked schemes.

7.2 Future Work

As UWA research has just begun its major advancement for only the last few decades, many research topics and issues remain for further studies before UWA networks can fully be exploited in real applications. In the following, we discuss the possible directions for our future researches and the interesting research problems for further investigation.

7.2.1 Future Study

As for our future work, it is important to analyze the normalized throughput for an infinite number of nodes since the number of sensor node deployed in UWA networks usually small. Also, it is interesting to extend our analysis to understand the behaviour of the average waiting time of a packet before it can be successfully transmitted, in addition to the normalized throughput performance. This understanding is useful in determining the average packet delay which is one of key metric in MAC's performance evaluation. Moreover, we plan to examine our proposed MAC protocols in a more realistic network scenario, by relaxing our assumption of having no BER, no time-varying propagation delay, as well as no node mobility presence in the network model. Upon understanding how these factors affect the network performance of our proposed protocols, we shall concentrate on a design or the modification of MAC protocols for realistic mobile networks. Although the speed of the AUV is typically low, approximately 2 m/s, the lack of time synchronization and unavailable on-board positioning system make the MAC design a very challenging task.

Our future plan for the SBR-DLP is to integrate it with several UWA MAC protocols and investigate their relative performance. We also plan to relax the

assumption that the SINK can lock back to its intended path fairly quickly after a certain deviation, so as to create a more realistic mobility model. Moreover, in the design of SBR and SBR-DLP, the network relies on node mobility to overcome the problem of network disconnectivity. This may not be a good solution for a network that is delay-intolerant. Thus, the study on a suitability of relying on node mobility to overcome the problem of disconnectivity can be one of a future direction in this area.

For MU-Sync, as we are currently using half of the round-trip time as an estimation of the one-way propagation delay, this may result in low accuracy if the propagation delay varies very significantly within the round trip message exchange. In our future work, we plan to concentrate on how the varying propagation delay can be estimated more accurately, while still maintaining low overhead. More specifically, the assumption of constant propagation speed should be relaxed such that the the spatial and temporal variations (e.g., due to temperature and pressure variations) of the sound speed is taken into account.

7.2.2 Interesting Research Problems for Investigation

7.2.2.1 Cross-layer Design Optimization

In such a harsh environment such as UWA channels, a cross-layer design will continue to gain interest from researchers since it has proved, in both the existing and our proposed protocols, that network performances (e.g., normalized throughput, collision rate) can be enhanced by considering and associating the information available among different network layers. Based on our observations, the research direction of a network protocol design will be focusing more on the variety of Quality-of-Service (QoS) requirements, in order to ensure security, energy balance and network disruptive tolerance, etc. Besides the usual cross-layer

approach of sharing information across the traditional network layering architecture, the joint-optimization technique can provide benefits for underwater networks. Considering the dynamic nature of underwater networks, it is difficult to determine which optimal settings can achieve the desired QoS. Thus, the optimization scheme could be integrated with the cross-layer design to adaptively provide the optimal network parameters even when there are some unexpected events occur. For example, the available topology could be useful in determining a routing path that can minimize the delay while at the same time maintain the energy balance among sensor nodes. The right determination may also helps in avoiding the network disruption caused by the power depletion of just a few nodes.

7.2.2.2 Network Coding

Although coding was originally designed to be used at the physical layer, recent studies [72, 73, 74] show that coding in network layer could also offer benefits such as maximum rate transmission, load balancing, efficient bandwidth utilization and link failure recovery. Because of these benefits, network coding for UWA networks is expected to gain research interest in a near future.

One possible area in applying the network coding is to provide an efficient error recovery in the high error rate and long propagation delay UWA channel. Traditionally, there are two techniques to overcome the error in a channel: the ARQ-based schemes and the FEC (Forward Error Control). In the ARQ-based schemes, a sender retransmits a packet only if it detects that the packet has been lost. The detection of packet loss usually comes from the destination node. In a long propagation delay channel, this error recovery means induces a long delay before the packet can be retransmitted, exacerbating the long propagation

delay already experienced by a packet. The FEC, on the other hand, adds extra redundancy to the packet before transmission. With the bandwidth and energy constraints, the amount of redundancy that needs a carefully determination is very challenging in a highly dynamic UWA channel. Network coding, on the other hand, groups transmissions at the intermediate node into multiple packets to serve the objective that when receiving only some of the transmitted packets, the destination node can extract the original packets. In addition, network coding could be viewed as an efficient routing that guarantees the low delay but high PDR as well as the low power consumption.

Before network coding can be fully exploited and implemented underwater, the relationship of the total number of coded packets to be transmitted, the amount of energy saved in certain conditions of UWA channel, and the code design, etc. need to be furthermore investigated.

7.2.2.3 Self-reorganization of UWA Networks

For some certain types of application, which their networks require quite a long operational period, self-reorganization algorithm is of great importance. These types are such as environmental monitoring and security surveillance, etc. The self-reorganization allows networks to restore and recover itself even when they are encountering security threats or strong network dynamics (e.g., nodes leave or enter the network) that could lead to network failure. Without the need of re-deployment and re-initiation of the network, the maintenance cost can be greatly reduced. Specifically, self-reorganization in UWA networks can be achieved by applying power control, reallocating the cluster, or re-initiating the network, etc.

Although this topic has not yet been largely explored currently, we strongly believe that it is an essential part of the network stack that can enhance network

robustness which is certainly one of the most important objectives in network design.

Appendix

A The Derivation of $P_{k,\mathbf{B_2}}$

Here, we compute the probability that a busy period is of type B_2 and contains k successful packets, P_{k,B_2}, where $k \geq 0$. Consider the case where none of the packets transmitted in the busy period is successful, i.e., $k = 0$. This implies that there is at least one other packet arrival during the vulnerable period $p_{\max}$ of the first packet that started the busy period. Thus,

$$\begin{aligned} P_{k=0,B_2} &= P\{\text{at least one arrival during } p_{\max}\} \\ &= 1 - e^{-\lambda p \max}. \end{aligned} \tag{A1}$$

For the case of $k = 1$ in B_2, one packet is successfully received without any collision, followed by a collision between multiple packets that ends the busy period. Since the first packet is successful, there must be no arrival during its vulnerable period. Also, in order to continue the busy period beyond the first packet, there must be one packet arrival during the final $p_{\max}$ of the first packet's channel occupancy. Finally, in order for the second packet to be involved in a collision with one or more other packets, there is at least one other packet arrival during its vulnerable period. Thus,

$$\begin{aligned} P_{k=1,B_2} &= P\{\text{no arrival during } p_{\max}\} \cdot P\{x \leq p_{\max}\} \cdot \\ &\quad P\{\text{at least one arrival during } p_{\max}\} \\ &= e^{-\lambda p \max}(1 - e^{-\lambda p \max})(1 - e^{-\lambda p \max}) \\ &= A(1 - e^{-\lambda p \max}). \end{aligned} \tag{A2}$$

Following the same reasoning as the above, we can obtain the general expression for P_{k,B_2} for any $k \geq 0$ as,

$$P_{k,\mathrm{B}_2} = A^k(1 - e^{-\lambda p_{\max}}). \tag{A3}$$

B Derivation of Backoff Probability $P_{bo}(\lambda'_j)$

From (3.30), we see that $P_{\mathrm{bo}}(\lambda'_j)$ can be expressed in terms of P_{B_1}, P_{B_2}, $P_{\mathrm{bo}|(m,\mathrm{B}_1)}$, $P_{\mathrm{bo}|(k,\mathrm{B}_2)}$, $P_{m|\mathrm{B}_1}$, and $P_{k|\mathrm{B}_2}$ (we have dropped (λ'_j) from these notations for simplicity). Here, we provide their derivations.

First, let us derive $P_{\mathrm{bo}|(m,\mathrm{B}_1)}$. In a busy period of type B_1 that contains m successful packets, there are exactly m backoff periods, each with duration $(T - p_{\max})$. In addition, there are exactly m arrivals outside these backoff periods. Since $P_{\mathrm{bo}|(m,\mathrm{B}_1)}$ can be obtained as the ratio of the average number of arrivals that fall within the backoff periods to the average number of arrivals within the entire busy period, we have,

$$\begin{aligned} P_{\mathrm{bo}|(m,\mathrm{B}_1)} &= \frac{m\lambda'(T - p_{\max})}{m + m\lambda'(T - p_{\max})} \\ &= \frac{\lambda'(T - p_{\max})}{1 + \lambda'(T - p_{\max})}, \end{aligned} \tag{B1}$$

which is independent of m.

Next, we derive $P_{\mathrm{bo}|(k,\mathrm{B}_2)}$ in a similar manner. In a busy period of type B_2 that contains k successful packets, there are exactly $k+1$ backoff periods. Each of the first k backoff periods has a duration of $(T - p_{\max})$, while the final backoff period, as can be seen from Fig. 3.7, has an average duration of $T + \bar{y}$. Outside these backoff periods, we have $k+2$ confirmed packet arrivals, corresponding to the k successful packets, as well as the first and the last packets that are involved

in a collision. In addition, some packet arrivals may also occur between the first and the last packets involved in a collision, with an average of $\lambda'\bar{y}$. Hence, we have,

$$\begin{aligned} P_{\mathrm{bo}|(k,\mathrm{B}_2)} &= \frac{\lambda'[k(T-p_{\max})+(T+\bar{y})]}{k+2+\lambda'\bar{y}+\lambda'[k(T-p_{\max})+(T+\bar{y})]} \\ &= \frac{k\lambda'(T-p_{\max})+\lambda'(T+\bar{y})}{k[1+\lambda'(T-p_{\max})]+\lambda'(T+2\bar{y})+2}. \end{aligned} \tag{B2}$$

As for P_{B_1} and P_{B_2}, they can be obtained as follows,

$$P_{\mathrm{B}_1} = \frac{\bar{N}_1}{\bar{N}_1+\bar{N}_2}, \tag{B3}$$

$$P_{\mathrm{B}_2} = \frac{\bar{N}_2}{\bar{N}_1+\bar{N}_2}, \tag{B4}$$

where $\bar{N}_1$ and $\bar{N}_2$ are the average number of packet arrivals that occur in a type 1 and a type 2 busy period, respectively. Specifically, $\bar{N}_1$ and $\bar{N}_2$ can be obtained as,

$$\begin{aligned} \bar{N}_1 &= \sum_{m=1}^{\infty} P_{m,\mathrm{B}_1} \cdot m[1+\lambda'(T-p_{\max})] \\ &= \frac{e^{-2\lambda' p\max}[1+\lambda'(T-p_{\max})]}{(1-A)^2}, \end{aligned} \tag{B5}$$

$$\begin{aligned} \bar{N}_2 &= \sum_{k=0}^{\infty} P_{k,\mathrm{B}_2} \cdot \{k[1+\lambda'(T-p_{\max})]+\lambda'(T+2\bar{y})+2\} \\ &= (1-e^{-\lambda' p\max}) \cdot \\ &\quad \left[\frac{\lambda'(T+2\bar{y})+2}{(1-A)} + \frac{A[1+\lambda'(T-p_{\max})]}{(1-A)^2}\right]. \end{aligned} \tag{B6}$$

Next, by applying Bayes' theorem to both P_{m,B_1} in (3.19) and P_{k,B_2} in (A3),

we can derive $P_{m|\mathrm{B}_1}$ and $P_{k|\mathrm{B}_2}$ as follows:

$$P_{m|\mathrm{B}_1} = \frac{P_{m,\mathrm{B}_1}}{\sum_{m=1}^{\infty} P_{m,\mathrm{B}_1}} = (1-A)A^{m-1}, \tag{B7}$$

$$P_{k|\mathrm{B}_2} = \frac{P_{k,\mathrm{B}_2}}{\sum_{k=0}^{\infty} P_{k,\mathrm{B}_2}} = (1-A)A^{k}. \tag{B8}$$

Finally, by substituting (B1), (B2), (B3), (B4), (B7), and (B8) into (3.30), we can obtain $P_{\mathrm{bo}}(\lambda'_j)$.

C Simulation Environment

The results shown in thesis are simulated using our own event-driven simulator written in C++. We would like to devote this section to discuss on the simulation environment so that the same results can be generated using other simulators.

C1 Aloha-CA and Aloha-AN

- **Propagation delay:** Fixed at the rate of 1500 m/s
- **Channel error:** Currently, we assume that the channel is error-free which makes it easier in understanding and interpreting the behaviour of our protocol designs.
- **Network size and deployment(N):** 4 and 100 static nodes are randomly deployed with a uniform distribution
- **Area of interest:** 2-D network with the size of 1000 m by 1000 m
- **Communication range:** Single-hop network–all nodes can hear every other nodes in the network

- **Modem:** Half-duplex omnidirectional modem with the data rate of 2400 bps
- **Packet generation at each node:** The packet generation is assumed to be Poisson with rate λ'/N packet/s, and each packet's intended receiver is randomly chosen with equal probability.
- **Packet length:** L bits with 32-bit header within (Aloha-CA) and L bits with a separate 32-bit NTF packet (Aloha-AN)
- **Packet retransmission:** No packet retransmission
- **Random backoff:** The exponentially distributed backoff with the average duration of $1/\lambda'$.
- **Collision detection:** It is assumed that the collision between multiple packets can be detected by all other nodes.

C2 RIPT

- **Propagation delay:** Fixed at the rate of 1500 m/s
- **Channel error:** Currently, we assume that the channel is error-free which makes it easier in understanding and interpreting the behaviour of our protocol designs.
- **Grid spacing:** 700 and 7000 m
- **Network size and deployment:** 36 static nodes are arranged in a grid topology (See Fig. 4.2). However, instead of precisely placing each node at a grid intersection point, we introduce some degree of randomness by allowing

each node to deviate from the grid intersection point by a maximum of 10% of its grid spacing, in both the x and y directions.

- **Communication range:** 1.75 times the grid spacing
- **Modem:** Half-duplex omnidirectional modem with the data rate of 2400 bps
- **Packet generation at each node:** The packet generation is assumed to be Poisson with rate $\lambda'/36$ packet/s, and each packet's intended receiver is randomly chosen among its 16 two-hop neighbors with equal probability.
- **Packet length:** All control packets (i.e., RTR, SIZE and ORDER) have the same size of 100 bits, while DATA packets are 2400-bit long.
- **Routing:** Static routing as shown in Fig. 4.2
- **Buffer size at each node:** 2 buffers, with the size of 100 each, are used to separately store new packets and relayed packets
- $M_{\mathbf{train}}$**:** Initially set to 1
- **Packet retransmission:** No packet retransmission

C3 SBR and SBR-DLP

- **Propagation delay:** Fixed at the rate of 1500 m/s
- **Network size and deployment:** N mobile nodes moving randomly within a 2-D network with the size of 1000 m by 1000 m
- **SINK:** The SINK is fixed and located at the point (1000, 1000) for the study of the mobile network with a static sink, it is constantly moving with

a pre-planed path in the fully mobile network. However, due to the ocean current, the SINK deviates from its pre-planned path in such a way that, for any given pair of consecutive waypoints, X and Y, instead of moving from X to Y directly, the SINK travels from X to Z to Y. The position of point Z is $d_{\max}$ away perpendicularly from the midpoint of $\overline{XY}$.

- **Node mobility:** All mobile nodes has a maximum speed ($V_{\max}$) of 2 m/s unless specified otherwise. The speed and direction of each sensor node are randomly picked from the range of $[0, Vmax]$ and $[-45°, 45°]$, respectively, according to uniform distribution. These remain constant for an exponentially distributed period of time with an average interval of 300 s, before they are randomly picked again. If the node reaches the boundary of the testing area, it is reflected by the boundary, so that it goes back within the testing area.

- **Packet length:** The *DATA* and control packets (e.g., *Chk_Ngb, Chk_Ngb_Reply, NTF and ACK*) are 4800-bit and 32-bit long, respectively.

- **Other parameters:** $n_{\text{discard}} = 3$, $\alpha = 0.5$, $\Delta_{\text{threshold}} = 30$ s and $T_{\text{wait}}(n_{\text{fail}}) = 30n_{\text{fail}}$ s

- **Modem:** Half-duplex omnidirectional modem with the data rate of 2400 bps

- **Communication range:** 300 m

- **MAC protocol:** No MAC protocol–assuming there is only one packet to be routed at any instance

C4 MU-Sync

- **Propagation delay:** Fixed at the rate of 1500 m/s
- **Network size and deployment:** Two mobile sensor nodes move randomly within an area of 1000 m by 1000 m
- **Node mobility:** The maximum speed ($V_{\max}$) of a mobile node is 2 m/s unless specified otherwise. The speed and direction of each sensor node are randomly picked from the range of [0, $Vmax$] and $[-45°, 45°]$, respectively, according to uniform distribution. These remain constant for an exponentially distributed period of time with an average interval of 600 s, before they are randomly picked again. If the node reaches the boundary of the testing area, it is reflected by the boundary, so that it goes back within the testing area.
- **Clock skew:** 40 ppm and no skew variation
- **Clock offset:** 10 ppm
- **Number of beacons used to perform linear regression:** 25
- **Duration $t_3 - t_2$:** 0 s
- **Time interval between two successive reference packets:** 5 s
- **Clock granularity:** 1 μs
- **Receive jitter:** 15 μs

D Protocol Stack

This section presents our protocol designs in a flow-chart form which allows better understanding of how the node reacts when certain event has arrived. The notations used in the following flow charts are summarized in Table D1

D1 Aloha-CA

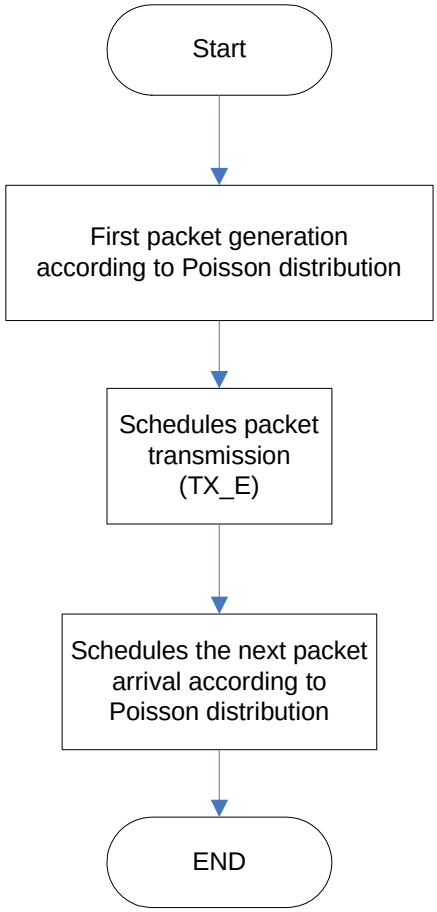

Figure D1: Packet generation in Aloha-CA.

Table D1: Notations used in explaining the protocol stack.

Notation	Description
TX_E	Transmit event
Gen_Pkt_E	Generate a new packet event
RX_Header_E	Header packet received event
RX_DATA_E	DATA received event
TX_NTF_E	Transmit NTF packet event
RTR_INT_E	RTR initiation event
TO_SIZE_E	Timeout for acquiring the SIZE packets event
RX_RTR_E	RTR packet received event
RX_SIZE_E	SIZE packet received event
RX_ORDER_E	ORDER packet received event
RX_Chk_Ngb_E	Chk_Ngb packet received event
Chk_Ngb_Reply_E	Chk_Ngb_Reply packet received event
TO_ACK_E	Timeout for acquiring the ACK packet event
TX_REF_E	Transmit REF packet event
TO_RSP_E	Timeout for acquiring the response packets (RSP_REF) event
RX_REF_E	REF packet received event
RX_RSP_E	RSP_REF packet received event
IDLE	The node is being idle
BUSY	The node is being busy
TX	The node is currently transmitting
RX_NTF	The node is receiving an NTF packet
RX_DATA	The node is receiving a DATA packet
CX	The node is currently under collision. This includes the transmit-receive and receive-receive collision

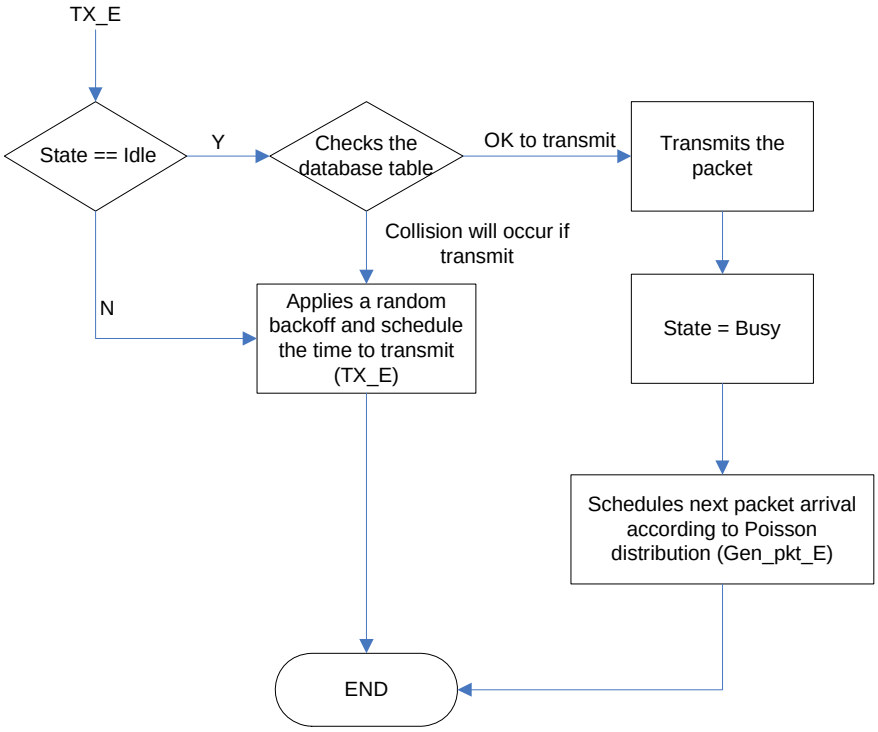

Figure D2: Flowchart of packet transmission.

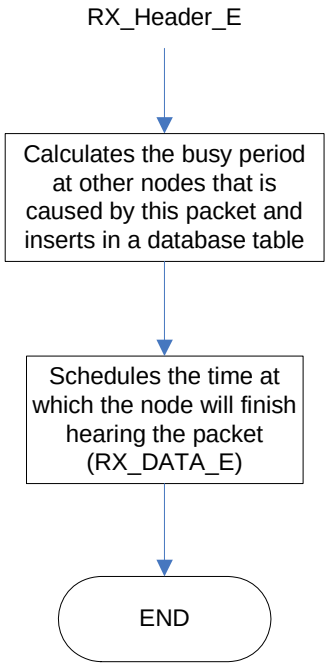

Figure D3: Flowchart of how the node reacts when a header is received.

Figure D4: Flowchart of how the node reacts when the data segment is received.

D2 Aloha-AN

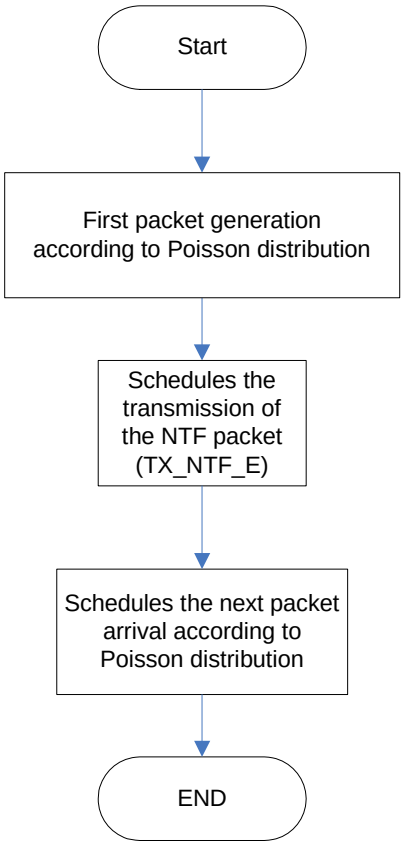

Figure D5: Packet generation in Aloha-AN

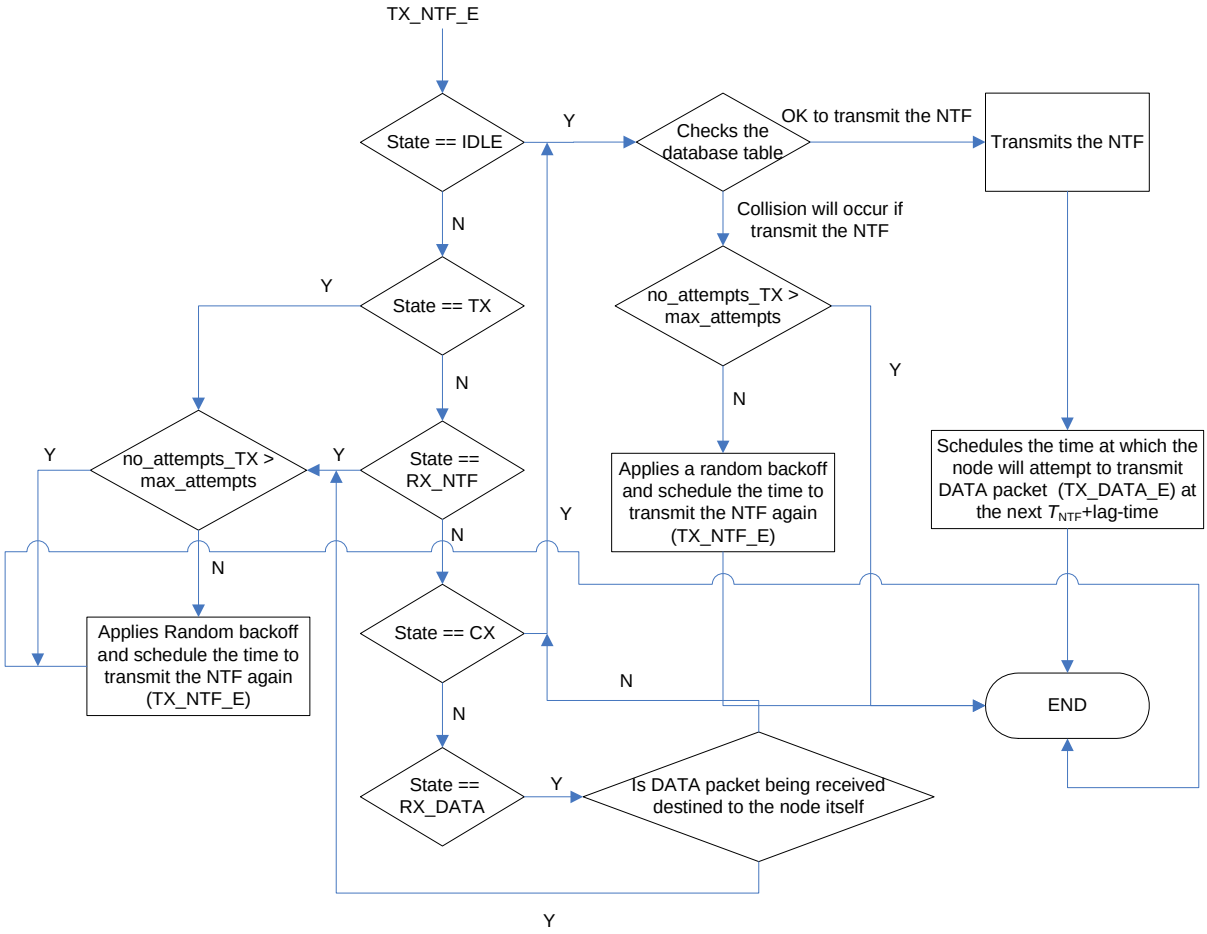

Figure D6: Flowchart of how the node decides whether to transmit the NTF packet.

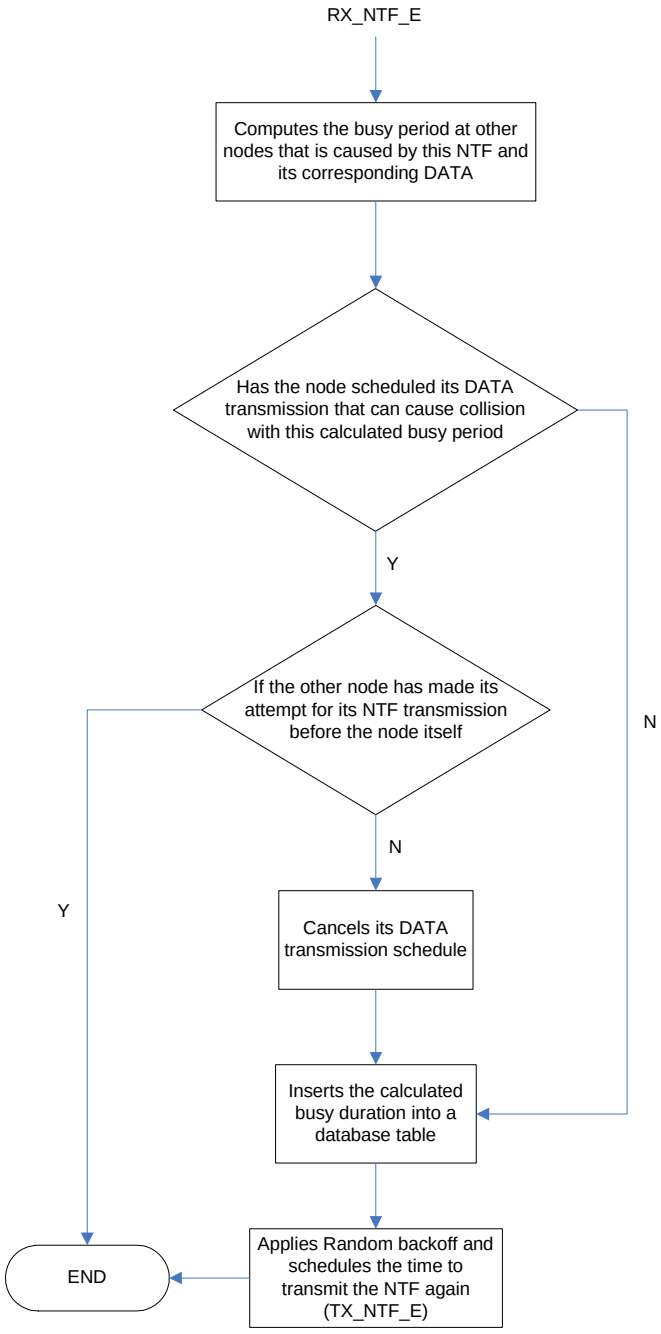

Figure D7: Flowchart of how the node reacts when an NTF packet is received.

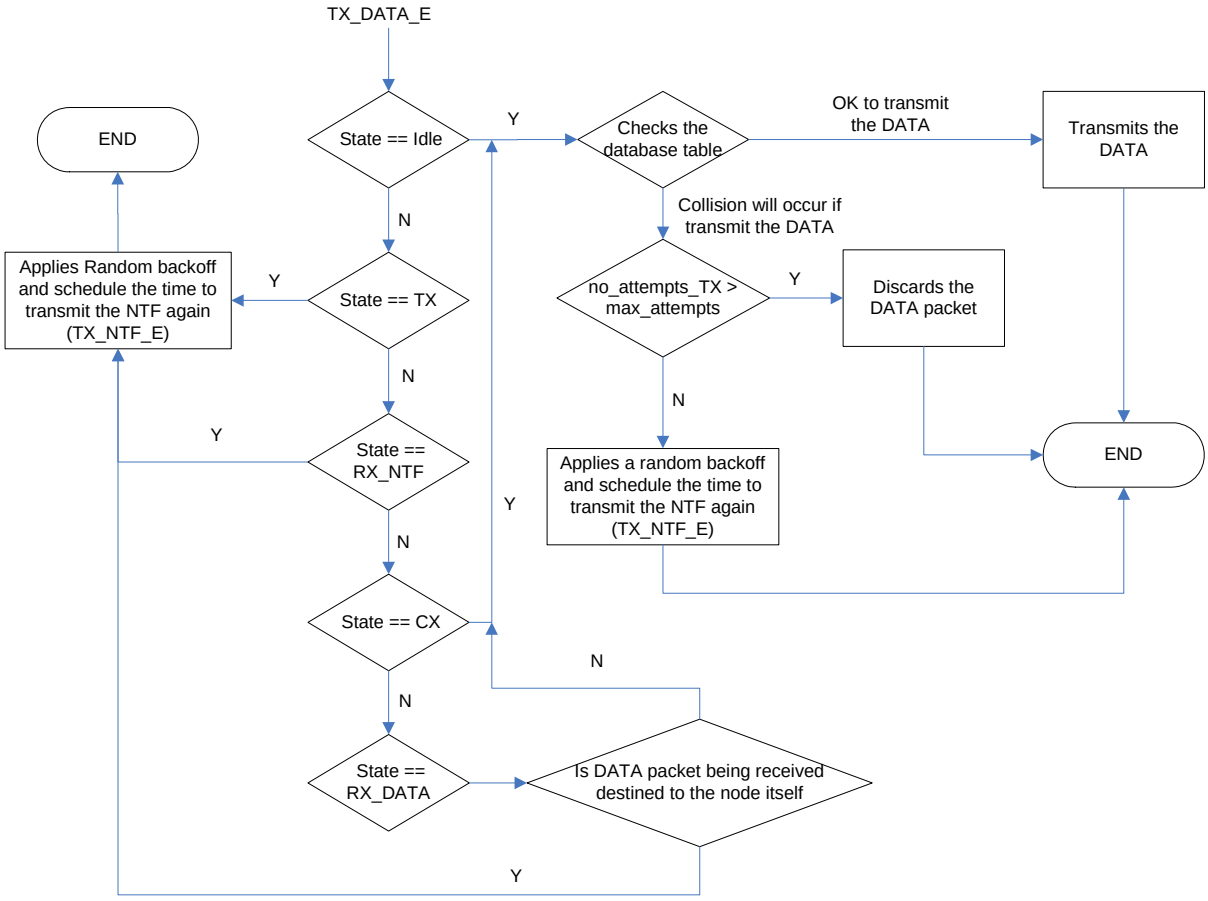

Figure D8: Flowchart of how the node decides whether to transmit a DATA packet.

D3 RIPT

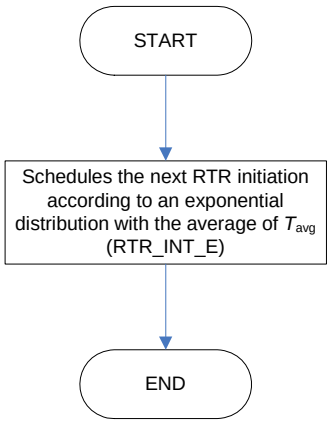

Figure D9: The start of RIPT.

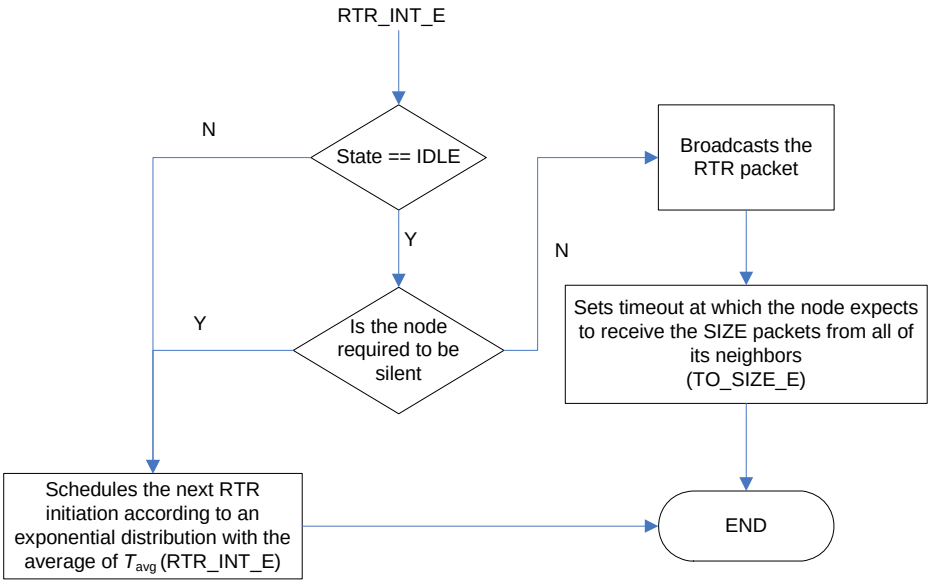

Figure D10: Flowchart of an RTR initiation.

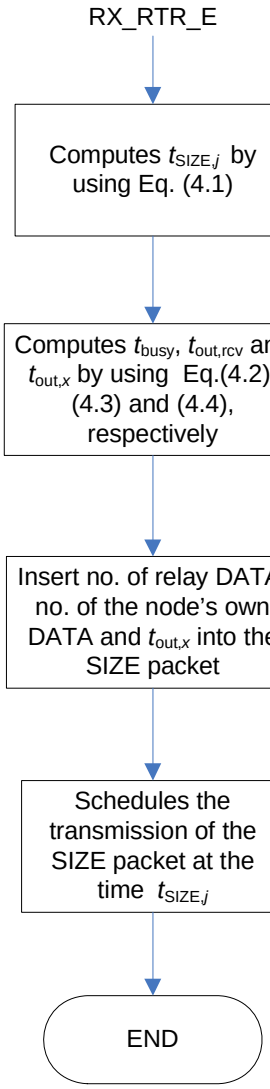

Figure D11: Flowchart of how the node reacts when the RTR packet is received.

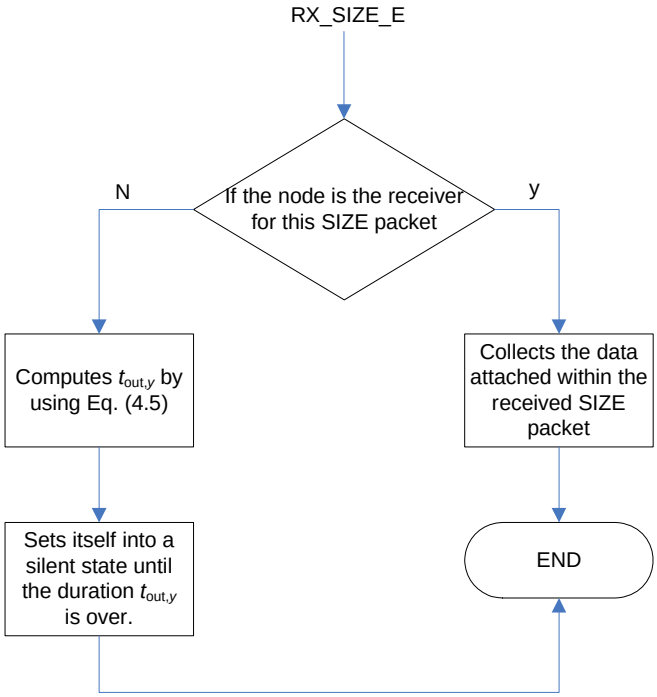

Figure D12: Flowchart of how the node reacts when the SIZE packet is received.

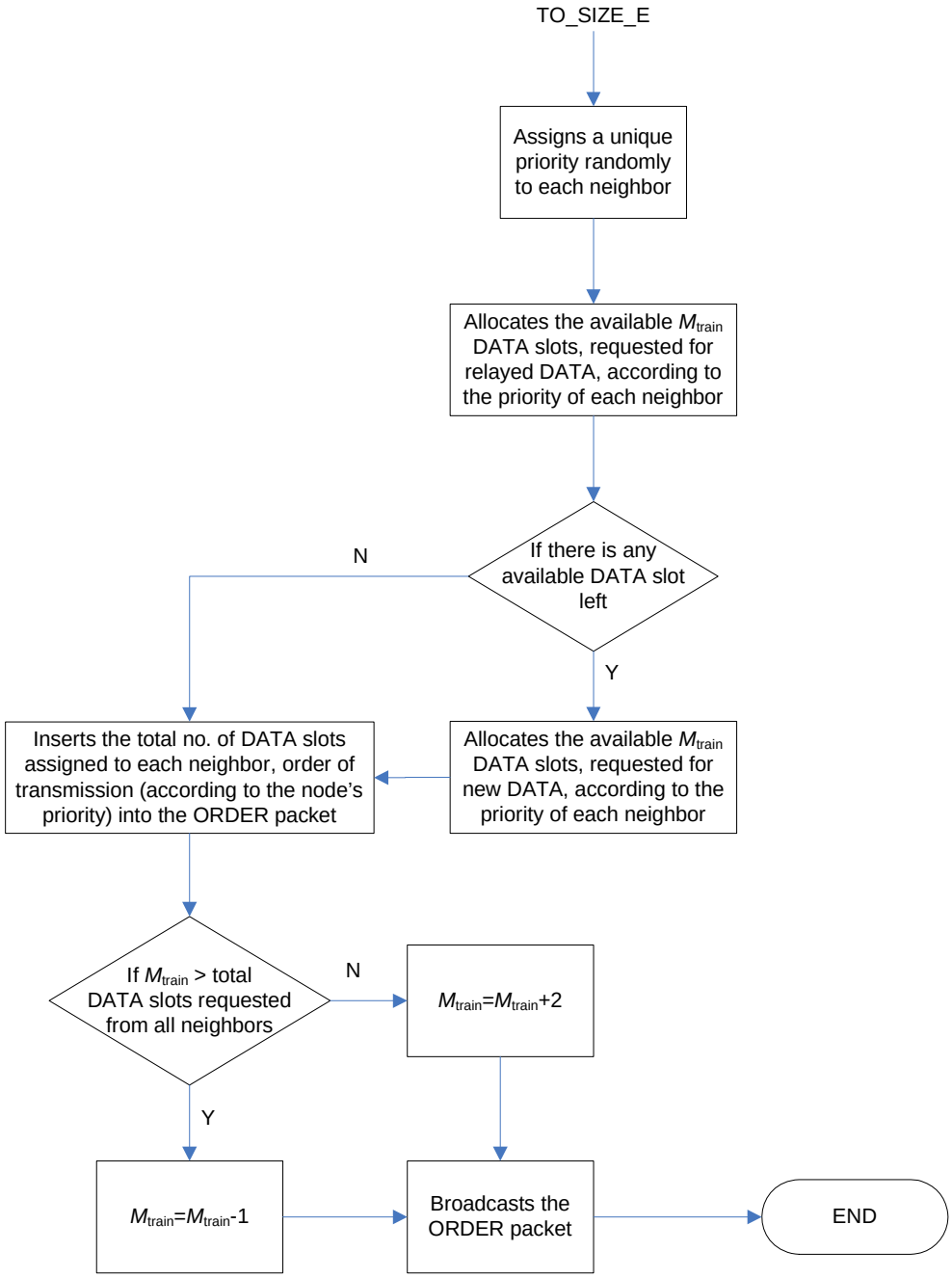

Figure D13: Flowchart of how the node reacts after acquiring the SIZE packets from all of its neighbors.

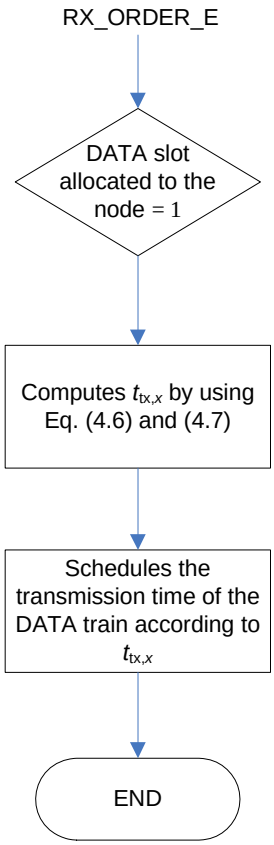

Figure D14: Flowchart of how the node reacts when ORDER packet is received.

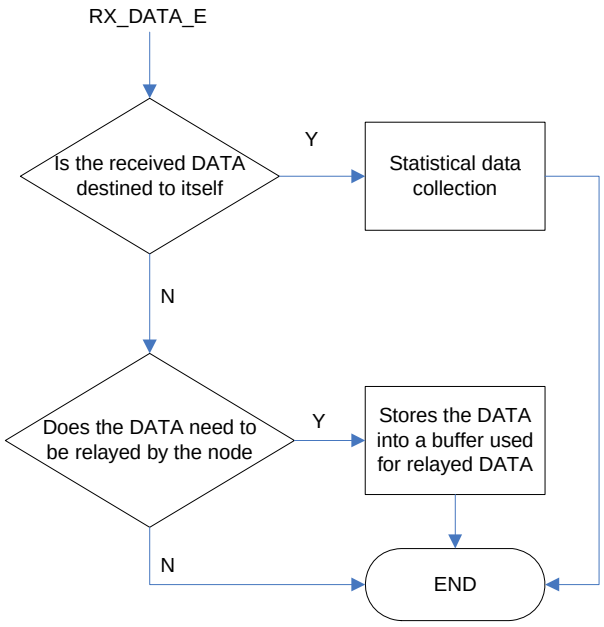

Figure D15: Flowchart of how the node reacts when the DATA packet is received.

D4 SBR and SBR-DLP

In the following, the protocol stacks shown are used for both SBR and SBR-DLP, if not specified otherwise.

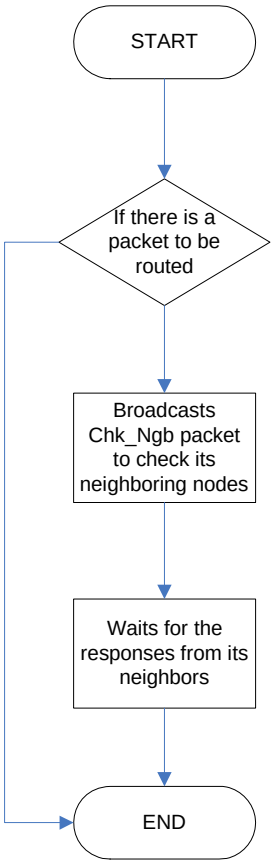

Figure D16: Flowchart of how SBR and SBR-DLP starts to route a packet.

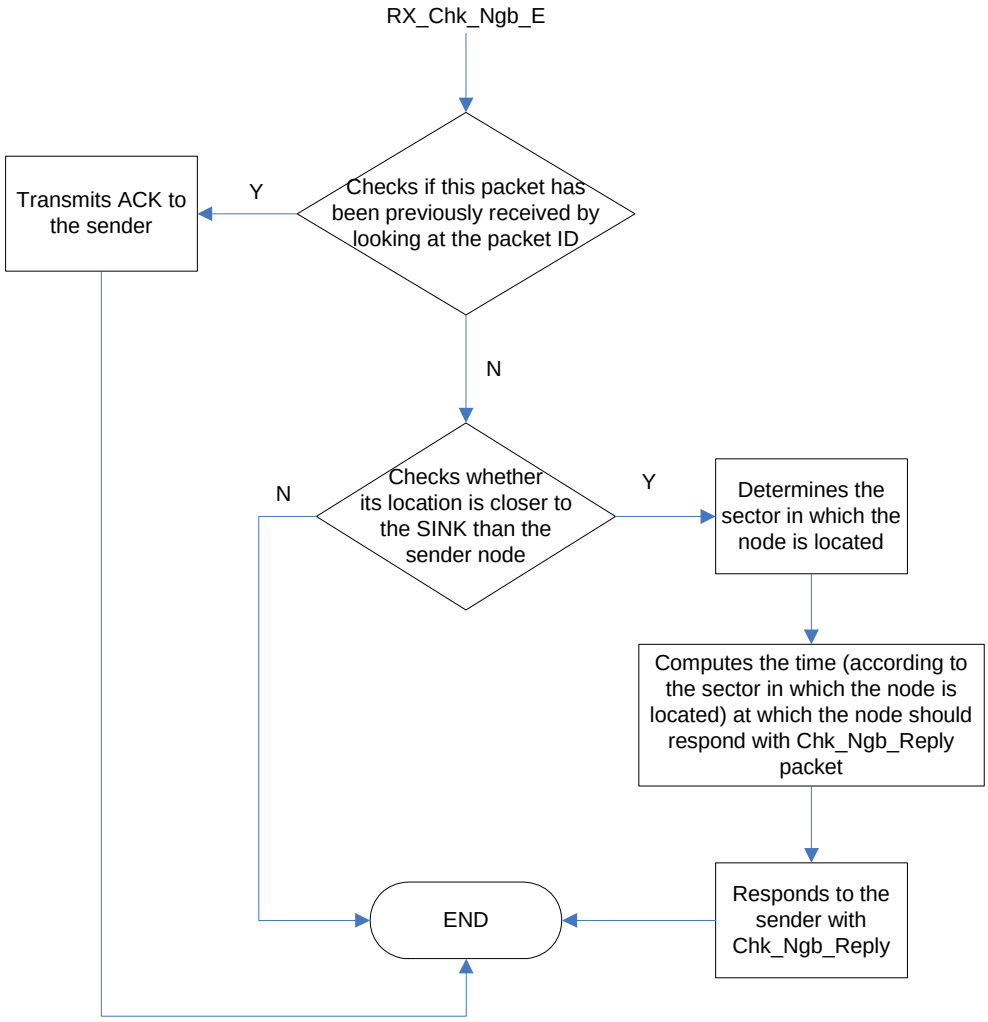

Figure D17: Flowchart of how the node in SBR reacts when Chk_Ngb packet is received.

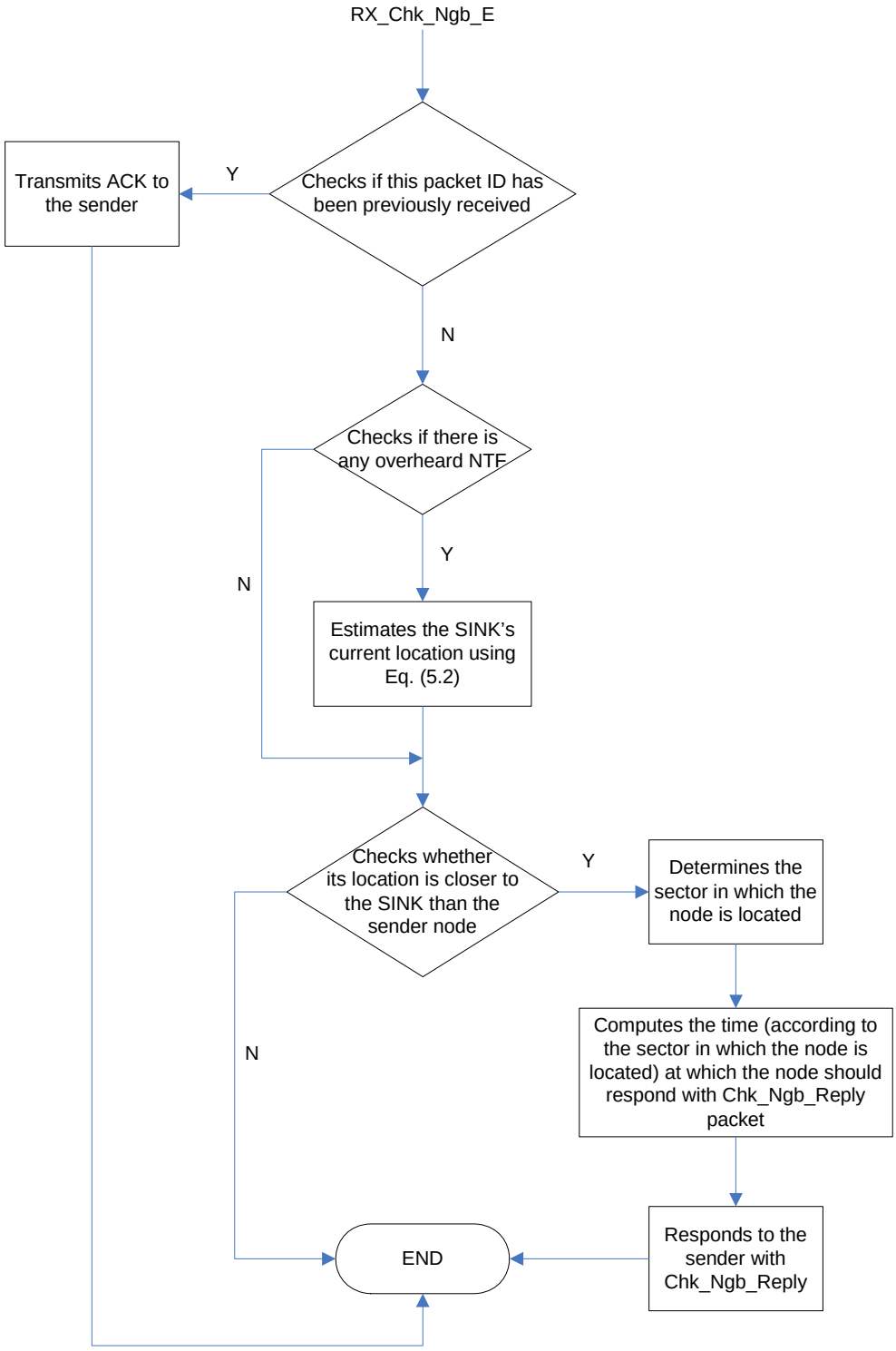

Figure D18: Flowchart of how the node in SBR-DLP reacts when Chk_Ngb packet is received.

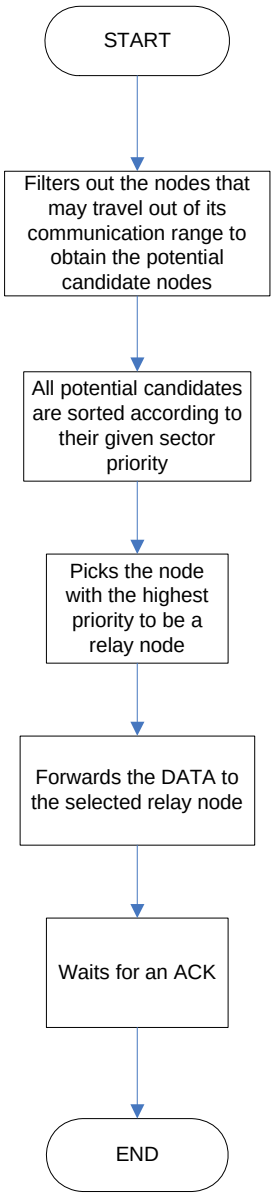

Figure D19: Flowchart of how the node chooses the next-hop forwarder.

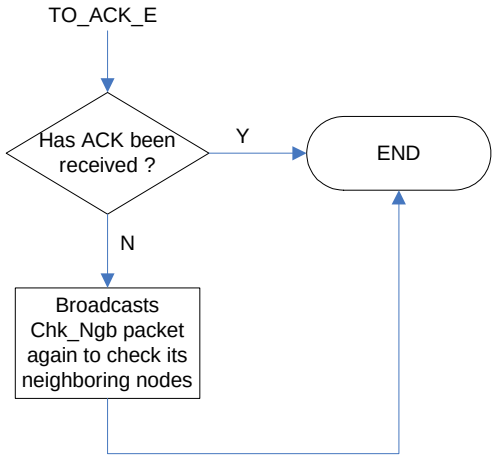

Figure D20: Flowchart of how the node react when ACK is received.

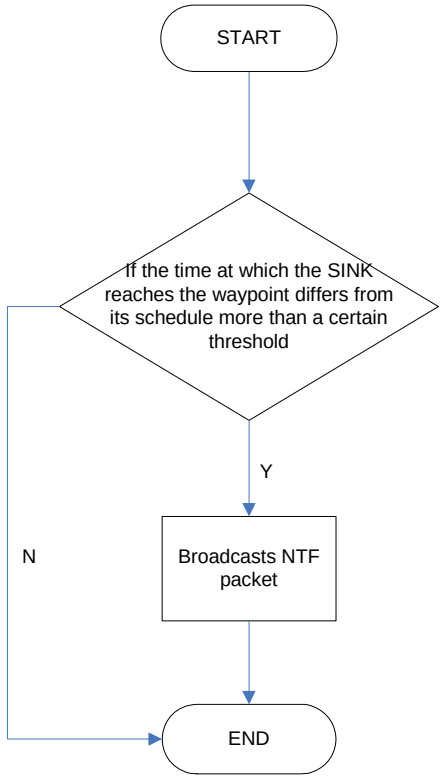

Figure D21: Flowchart of the DLP algorithm.

D5 MU-Sync

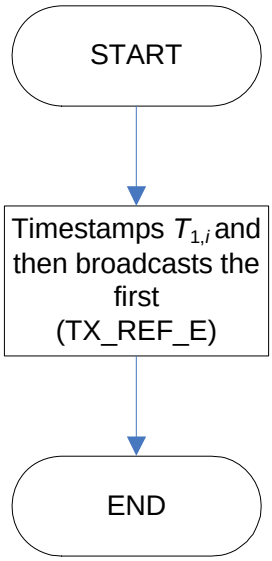

Figure D22: Flowchart of how the synchronizing node initiate the synchronization process.

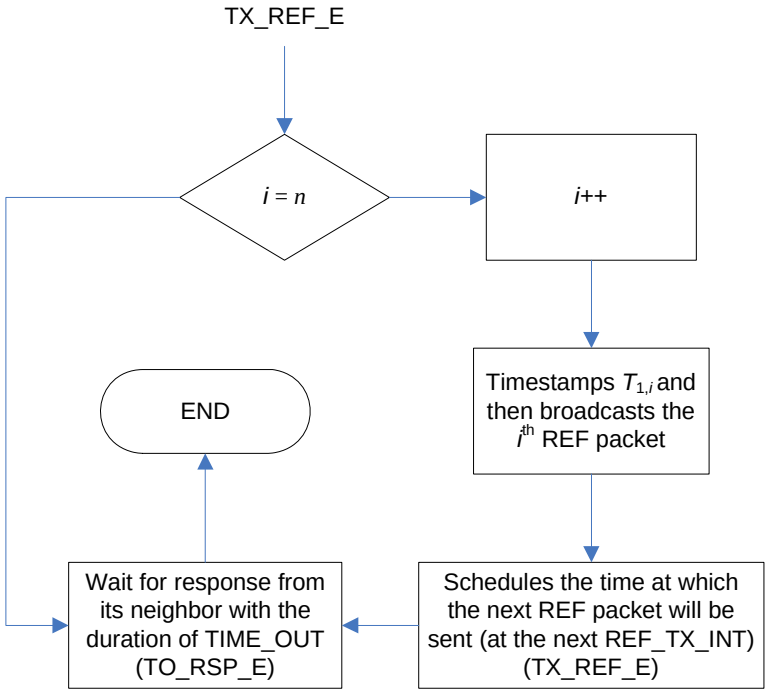

Figure D23: Flowchart of how the synchronizing node transmit the REF packet.

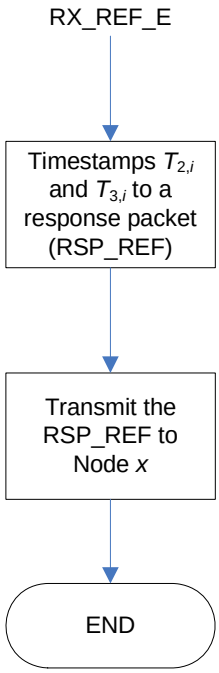

Figure D24: Flowchart of how the synchronized node reacts when the REF packet is received.

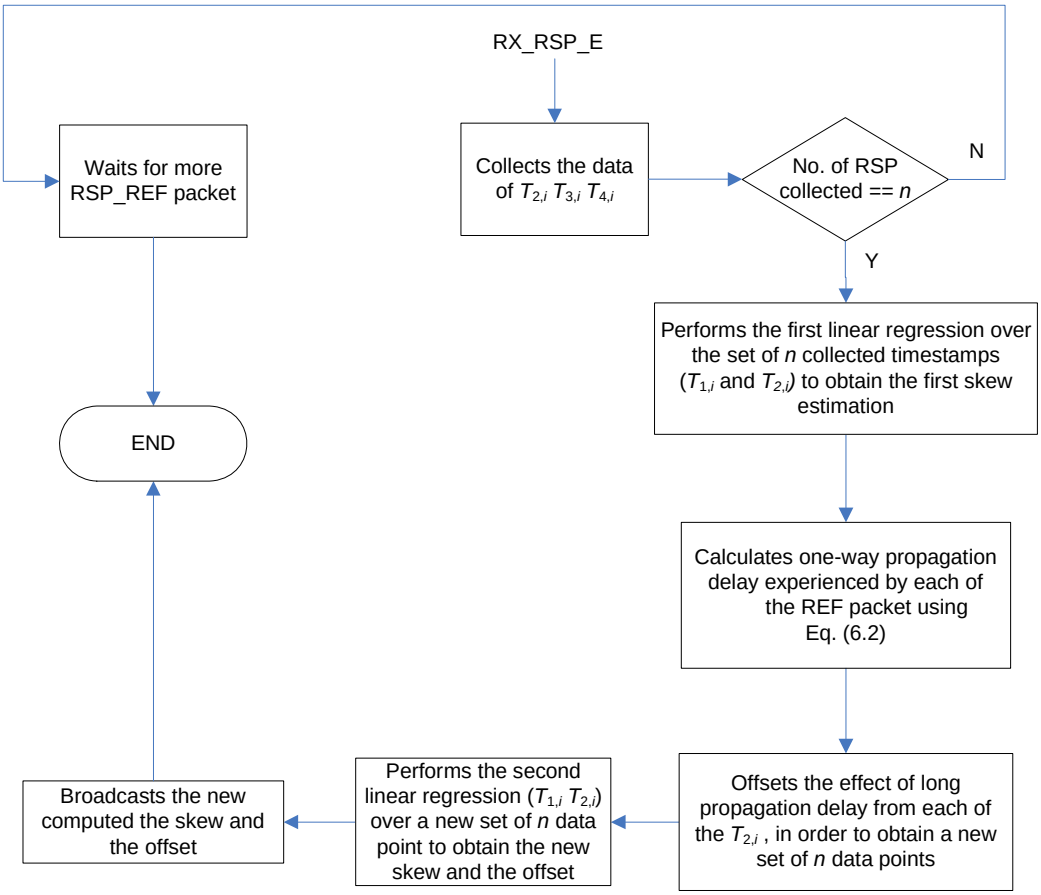

Figure D25: Flowchart of how the synchronizing node reacts when it receives RSP packet.

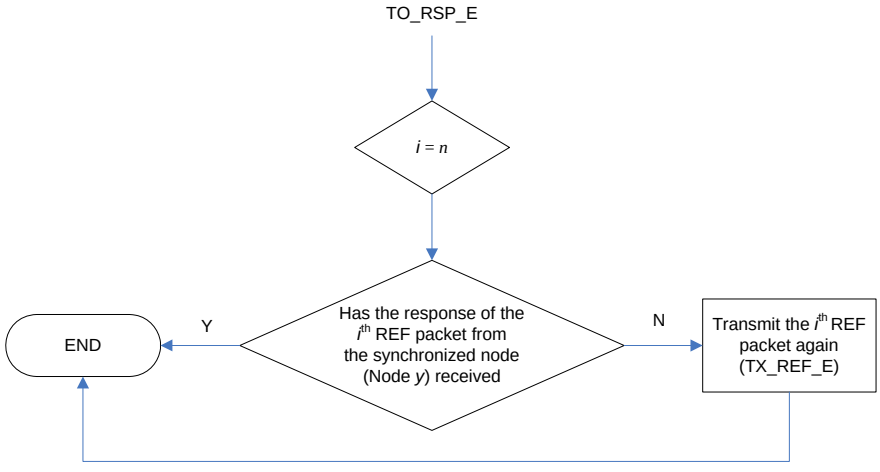

Figure D26: Flowchart of how the synchronizing node checks for the RSP packet from the synchronized node.

Bibliography

[1] J.-H. Cui, J. Kong, M. Gerla and S. Zhou, "The challenges of building mobile underwater wireless networks for aquatic applications," *IEEE Network*, vol. 20, no. 3, pp. 12–18, May 2006.

[2] F. Schill, U. R. Zimmer, and J. Trumpf, "Visible spectrum optical communication and distance sensing for underwater applications," in *Proc. ACRA*, Canberra, Australia, Dec. 2004.

[3] M. Stojanovic, "Underwater acoustic communications," *Encyclopedia of Electrical and Electronics Engineering, John G. Webster, Ed.*, pp. 688–698, 1999.

[4] N. Farr, "Optical modem technology for seafloor observatories," in *Proc. IEEE OCEANS'05*, Boston, MA, USA, Sept. 2005, pp. 1–6.

[5] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke, "Data collection, storage, and retrieval with an underwater sensor network," in *Proc. SenSys'05*, 2, Nov. 2005, pp. 216–221.

[6] J. Preisig, "Acoustic propagation considerations for underwater acoustic communications network development," in *Proc. ACM WUWNet'06*, Los Angeles, California, USA, Sept. 2006, pp. 1–5.

[7] C. S. Clay and H. Medwin, *Acoustical oceanography: Principles and Applications*. New York, NY: John Wiley & Sons, 1977.

[8] P. H. Dahl, "High-frequency forward scattering from the sea surface: The characteristic scales of time and angle spreading," *IEEE J. Oceanic Eng.*, vol. 26, no. 1, pp. 141–151, Jan. 2001.

[9] I. F. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: Research challenges," *Elsevier's Journal of Ad Hoc Networks*, vol. 3, no. 3, pp. 257–279, Jan. 2005.

[10] D. B. Kilfoyle and A. B. Baggeroer, "The state of the art in underwater acoustic telemetry," *IEEE J. Oceanic Eng.*, vol. 25, no. 1, pp. 4–27, Jan. 2000.

[11] N. Chirdchoo, W.-S. Soh, and K. C. Chua, "MU-Sync: A time synchronization protocol for underwater mobile networks," in *Proc. ACM WUWNet*, San Francisco, California, USA, Sept. 2008, pp. 35–42.

[12] K. Akkaya and M. Younis, "A survey of routing protocols in wireless sensor networks," *Elsevier Ad Hoc Network Journal*, vol. 3, no. 3, pp. 325–349, 2005.

[13] P. Xie and J.-H. Cui, "SDRT: A reliable data transport protocol for underwater sensor networks," *Technical Report: UbiNet-TR06-03*, Feb. 2006.

[14] M. Witt and V. Tuau, "Geographic routing in 3D," in *6. GI/ITG KuVS Fachgesprch Drahtlose Sensornetze,Technical Report, ISSN 0935-3232*, RWTH Aachen, Germany, July 2007.

[15] M. Witt and V. Turau, "BGR: Blind geographical routing for sensor networks," in *Proc. IEEE WISES*, Hamburg, Germany, May 2005, pp. 51–61.

[16] A. F. Harris III, M. Stojanovic and M. Zorzi, "When underwater acoustic nodes should sleep with one eye open: idle-time power management in underwater sensor networks," in *Proc. ACM WUWNet'06*, Los Angeles, California, Sept. 2006, pp. 105–108.

[17] L. Freitag, M. Grund, S. Singh, J. Partan, P. Koski, and K. Ball. (2005) The WHOI micro-modem: an acoustic communications and navigation system for multiple platforms. [Online]. Available: http://www.whoi.edu

[18] I. P. Morns, O. R. Hinton, A. E. Adams, and B. S. Sharif, "Protocol for subsea communication networks," in *Proc. MTS/IEEE OCEANS'01*, Honolulu, HI, USA, Nov. 2001, pp. 2076–2082.

[19] X. Guo, M. R. Frater, and M. J. Ryan, "A propagation-delay-tolerant collision avoidance protocol for underwater acoustic sensor networks," in *Proc. MTS/IEEE OCEANS'06*, Singapore, May 2006, pp. 1–6.

[20] S. Shahabudeen and M. A. Chitre, "Design of networking protocols for shallow water peer-to-peer acoustic networks," in *Proc. IEEE/OES OCEANS'05 Europe*, June 2005, pp. 628– 633.

[21] G. G. Xie and J. H. Gibson, "A network layer protocol for UANs to address propagation delay induced performance limitations," in *Proc. IEEE OCEAN'01*, Honolulu, HI, USA, Nov. 2001, pp. 2087–2094.

[22] V. Rodoplu and M. K. Park, "An energy-efficient MAC protocol for underwater wireless acoustic networks," in *Proc. MTS/IEEE OCEANS'05*, Washington, DC, USA, Sept. 2005, pp. 1198–1203.

[23] P. Karn, "MACA-A new channel access method for packet radio," in *Proc. ARRL/CRRL*, Sept. 1990, pp. 134–140.

[24] X. Guo, M. R. Frater, and M. J. Ryan, "An adaptive propagation-delay-tolerant MAC protocol for underwater acoustic sensor networks," in *Proc. IEEE OCEANS'07- Europe*, June 2007, pp. 1–5.

[25] ——, "Design of a propagation-delay-tolerant MAC protocol for underwater acoustic sensor networks," *IEEE J. Oceanic Eng.*, vol. 34, no. 2, pp. 170–180, April 2009.

[26] M. Molins, M. Stojanovic, "Slotted FAMA:A MAC protocol for underwater acoustic networks," in *Proc. MTS/IEEE OCEANS'06*, Singapore, May 2007, pp. 1–7.

[27] B. Peleato and M. Stojanovic, "A MAC protocol for ad-hoc underwater acoustic sensor networks," in *Proc. ACM WUWNet'06*, Sept. 2006, pp. 113–115.

[28] E. M. Sozer, M. Stojanovic, and J. G. Proakis, "Underwater acoustic networks," *IEEE Journal of Oceanic Engineering*, vol. 25, no. 1, pp. 72–83, Jan. 2000.

[29] H. Matsuno, H. Ishinaka and A. Hamanaga, "A simple modification for the drastic improvement of MACA in large propagation delay situation," in *Proc. WCNC 2000*, Chicago, IL, USA, Sept. 2000, pp. 865–869.

[30] V. Bhargavan, A. Demers, S. Shenker, L. Zhang, "MACAW-A media access protocol for wireless LAN's," in *Proc. ACM SIGCOMM*, London, United Kingdom, 1994, pp. 212–225.

[31] K. Y. Foo, P. R. Atkins, T. Collins, C. Morley, and J. Davies, "A routing and channel-access approach for an ad hoc underwater acoustic network," in *Proc. MTS/IEEE OCEANS'04*, Kobe, Japan, Nov. 2004, pp. 789 – 795.

[32] H.-H. Ng, W.-S. Soh, and M. Motani, "MACA-U: A media access protocol for underwater acoustic networks," in *Proc. IEEE Globecom*, New Orleans, Louisiana, USA, Dec. 2008, pp. 1–5.

[33] A. Kebkal, K. Kebkal, and M. Komar, "Data-link protocol for underwater acoustic networks," in *Proc. IEEE OCEANS'05*, June 2005, pp. 1174–1180.

[34] P. Xie and J.-H. Cui, "Exploring random access and handshaking techniques in large-scale underwater wireless acoustic sensor networks," in *Proc. IEEE OCEANS'06*, Boston, MA, USA, Sept. 2006, pp. 1–6.

[35] K. B. Kredo II and P. Mohapatra, "A hybrid medium access control protocol for underwater wireless networks," in *Proc. ACM WUWNet'07*, Sept. 2007, pp. 33–40.

[36] L. F. M. Vieira, J. Kong, U. Lee and M. Gerla, "Analysis of Aloha protocols for underwater acoustic sensor networks," in *Proc. ACM WUWNet'06*, Los Angeles, CA, USA, 2006, pp. 41–48.

[37] A. Syed, W. Ye, B. Krishnamachari, and J. Heidemann, "Understanding spatio-temporal uncertainty in medium access with ALOHA protocols," in *Proc. ACM WUWNet'07*, Montreal, Quebec, Canada, 2007, pp. 41–48.

[38] J. Ahn and B. Krishnamachari, "Performance of a propagation delay tolerant ALOHA protocol for underwater wireless networks," in *Proc. DCOSS*, Santorini Island, Greece, June 2008, pp. 1–16.

[39] M. Stojanovic, "Optimization of a data link protocol for an underwater acoustic channel," in *Proc. IEEE OCEANS'05-Europe*, Jun. 2005, pp. 68–73.

[40] D. Pompili, T. Melodia, and I. F. Akyildiz, "A distributed CDMA medium access control for underwater acoustic sensor networks," in *Proc. Med-Hoc-Net'07*, June 2007.

[41] A. F. Harris III, D. G. B. Meneghetti, and M. Zorzi, "Maximizing channel utilization for underwater acoustic links," in *Proc. IEEE OCEANS'07*, June 2007.

[42] E. M. Royer and C. K. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," *IEEE Personal Communications*, vol. 6, no. 2, pp. 46–55, April 1999.

[43] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: A survey," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 6–28, Dec. 2004.

[44] U. Lee, J. Kong, J.-S. Park, E. Magistretti, and M. Gerla, "Time-critical underwater sensor diffusion with no proactive exchanges and negligible reactive floods," *Elsevier Ad Hoc Netw.*, vol. 5, no. 6, pp. 943–958, 2007.

[45] W. K. G Seah and H.-X. Tan, "Multipath virtual sink architecture for underwater sensor networks," in *Proc. OCEANS'06-Asia Pacific*, Singapore, May 2007, pp. 1–6.

[46] S. Mueller, R. P. Tsang, and D. Ghosal, "Multipath routing in mobile ad hoc networks: Issues and challenges," *Lecture Notes in Computer Science*, vol. 2965, pp. 209–234, April 2004.

[47] D. Pompili and T. Melodia, "Three-dimensional routing in underwater acoustic sensor networks," in *Proc. ACM PE-WASUN'05*, Montreal, Quebec, Canada, Oct. 2005, pp. 214–221.

[48] P. Xie, J.-H. Cui, and L. Lao, "VBF: Vector-based forwarding protocol for underwater sensor networks," in *Proc. IFIP Networking*, Coimbra, Portugal, May 2006, pp. 1216–1221.

[49] N. Nicolaou, A. See, P. Xie, J.-H. Cui, and D. Maggiorini, "Improving the robustness of location-based routing for underwater sensor networks," in *Proc. OCEANS'07*, Aberdeen, June 2007, pp. 1–6.

[50] J. M. Jornet, M. Stojanovic, and M. Zorzi, "Focused beam routing protocol for underwater acoustic networks," in *Proc. ACM WUWNet*, San Francisco, California, USA, Sept. 2008, pp. 75–82.

[51] J. Elson, L. Girod, and D. Estrin, "Fine-grained time synchronization using reference broadcasts," in *Proc. 5th Symp. Op. Sys. Design and Implementation*, Boston, MA, USA, Dec. 2002, pp. 147–163.

[52] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. ACM 1st int. conf. on Embedded networked sensor systems*, Los Angeles, CA, USA, Sept. 2003, pp. 138–149.

[53] A. A. Syed and J. Heidemann, "Time synchronization for high latency acoustic networks," in *Proc. INFOCOM 2006*, Barcelona, Spain, April 2006, pp. 1–12.

[54] C Tian, W. Liu, J. Jin, J. W, and Y. Mo, "Localization and synchronization for 3D underwater acoustic sensor networks," in *Springer*, Berlin / Heidelberg, 2007, pp. 622–631.

[55] L. Kleinrock and F. Tobagi, "Packet switching in radio channels: Part I–carrier sense multiple access modes and their throughput delay characteristics," *IEEE Trans. Commun.*, vol. 23, no. 12, pp. 1400–1416, Dec. 1975.

[56] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "he flooding time synchronization protocol," in *Proc. ACM SenSys'04*, Baltimore, MD, USA, 2004, pp. 39–49.

[57] L. Kleinrock, F. A. Tobagi, "Packet switching in radio channels: Part I-carrier sense multiple access modes and their throughput-delay characteristics," *IEEE Trans. Commun.*, vol. 23, no. 12, pp. 1400–1416, Dec. 1975.

[58] C. Wu, V. O. K. Li, "Receiver-initiated busy-tone multiple access in packet radio networks," in *Proc. ACM SIGCOMM*, Stowe, Vermont, United States, 1987, pp. 336–342.

[59] Z. J. Hass, J. Deng, "Dual busy tone multiple access (DBTMA)-A multiple access control scheme for ad hoc networks," *IEEE Trans. Commun.*, vol. 50, no. 6, pp. 975–984, June 2002.

[60] F. Talucci and M. Gerla, "MACA-BI (MACA by invitation) A wireless MAC protocol for high speed ad hoc networking," in *Proc. ICUPC*, San Diego, CA, USA, Oct. 1997, pp. 913–917.

[61] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *Proc. IEEE INFOCOM 2002*, Jun. 2002, pp. 1567– 1576.

[62] N. Chirdchoo, W.-S. Soh, and K. C. Chua, "Aloha-based MAC protocols with collision avoidance for underwater acoustic networks," in *Proc. IEEE INFOCOM 2007*, Anchorage, Alaska, USA, May 2007, pp. 2271–2275.

[63] F. Sivrikay and B. Yener, "Time synchronization in sensor networks: A survey," *IEEE Network*, vol. 18, no. 4, pp. 45–50, July-Aug. 2004.

[64] StatTrek. Statistics Tutorial: Least Squares Linear Regression . [Online]. Available: http://stattrek.com/AP-Statistics-1/Regression.aspx?Tutorial=Stat

[65] H. Kopetz and W. Schwabl, "Global time in distributed real-time system," in *Technical Report 15/89*, Technische Universitat Wien, 1989.

[66] ——, "Clock synchronization in distributed real-time systems," *IEEE Transactions on Computers*, vol. 36, no. 8, pp. 933–939, Aug. 1987.

[67] Q. Li and D. Rus, "Global clock synchronization in sensor network," *IEEE Transactions on Computers*, vol. 55, no. 2, pp. 214–226, Feb. 2006.

[68] N. Chirdchoo, W.-S. Soh, and K. C. Chua, "MACA-MN: A MACA-based MAC protocol for underwater acoustic networks with packet train for multiple neighbors," in *Proc. IEEE VTC2008-Spring*, Singapore, May 2008, pp. 46–50.

[69] H. Chen and S. Megerian, "Cluster sizing and head selection for efficient data aggregation and routing in sensor networks," in *Proc. IEEE WCNC 2006*, Las Vegas, NV, April 2006, pp. 2318–2323.

[70] K. Dasgupta, K. Kalpakis, and P. Namjoshi, "An efficient clustering-based heuristic for data gathering and aggregation in sensor networks," in *Proc. IEEE WCNC 2003*, New Orleans, LA, USA, March 2003, pp. 1948–1953.

[71] G. Chopra, S. Srivastava, and A. Karandikar, "A novel clustering strategy for efficient routing in adhoc networks," in *Proc. IEEE ICPWC 2005*, Jan. 2005, pp. 67–71.

[72] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network information flow," *IEEE Tran. on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[73] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Proc. ISIT*, 29 June - 4 July 2003, p. 442.

[74] Z. Guo, P. Xie, J.-H. Cui, and B. Wang, "On applying network coding to underwater sensor networks," in *Proc. ACM WUWNet 2006*, Los Angeles, California, USA, Sept. 2006, pp. 109–112.

List of Publications

- N. Chirdchoo, W.-S. Soh and K. C. Chua, "Sector-based routing with destination location prediction for underwater mobile networks (invited paper)," *IEEE WUnderNet*, May 2009, pp. 1148-1153.

- N. Chirdchoo, W.-S. Soh and K. C. Chua, "RIPT: A Receiver-initiated reservation-based protocol for underwater acoustic networks," *IEEE Journal on Selected Areas in Communications (JSAC), Special Issue on Underwater Wireless Communications and Networks*, vol. 26, no. 9, pp. 1744-1753, Dec. 2008.

- N. Chirdchoo, W.-S. Soh and K. C. Chua, "MU-Sync: A time synchronization protocol for underwater mobile networks," *ACM WUWNet*, San Francisco, California, USA, Sept. 2008, pp. 35-42.

- N. Chirdchoo, W.-S. Soh and K. C. Chua,"MACA-MN: A MACA-based MAC protocol for underwater acoustic networks with packet train for multiple neighbors," IEEE VTC2008-Spring, Singapore, May 2008.

- N. Chirdchoo, W.-S. Soh and K. C. Chua, "Aloha-based MAC protocols

with collision avoidance for underwater acoustic networks," *IEEE INFOCOM*, Anchorage, Alaska, USA, May 2007, pp. 2271-2275.