UNIVERSITY OF PADOVA

FACULTY OF ENGINEERING

DOCTORAL SCHOOL OF INFORMATION ENGINEERING

XXV CICLO

MEDIUM ACCESS CONTROL, ERROR CONTROL AND ROUTING IN UNDERWATER ACOUSTIC NETWORKS: A DISCUSSION ON PROTOCOL DESIGN AND IMPLEMENTATION

A THESIS SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN INFORMATION ENGINEERING

BY

MD. SAIFUL AZAD

CO-SUPERVISOR

DR. PAOLO CASARI

SUPERVISOR

PROF. MICHELE ZORZI

Academic Year 2012/2013

You don't choose your family.
They are God's gift to you,
as you are to them.
- Desmond Tutu

Acknowledgments

First and foremost, I would like to thank my supervisor, Professor Michele Zorzi, for believing in me, for giving me the freedom to explore and for his valuable guidance and support.

To Dr. Paolo Casari, my co-supervisor, my mentor and guide who teaches me a lot about research. He gave me the opportunity to begin this adventure, he led me through all these years, yet always helping when I needed it. Our friendship will last forever.

To my wife and my son for their sacrifice and the support they had given me throughout my dissertation. Thank you for letting me live every day in your words and thoughts, just as much as you live in mine.

I am also grateful to my parents, my brothers and my sisters. I would especially like to thank my father who always supports and encourages me to pursue my higher study.

To everyone in SIGNET lab, with them I have lots of memorable moments. In particular, a warm thanks to Matteow, Riccardo, Federico, Beatrice, and others with whom I worked on various projects.

Contents

List of Figures

List of Tables

Abstract

The journey of underwater communication which began from Leonardo's era took four and a half centuries to find practical applications for military purposes during World War II. However, over the last three decades, underwater acoustic communications witnessed a massive development due to the advancements in the design of underwater communicating peripherals and their supporting protocols. Successively, doors are opened for a wide range of applications to employ in the underwater environment, such as oceanography, pollution monitoring, offshore exploration, disaster prevention, navigation assistance, monitoring, coastal patrol and surveillance. Different applications may have different characteristics and hence, may require different network architectures. For instance, routing protocols designed for unpartitioned multi-hop networks are not suitable for Delay-Tolerant Networks. Furthermore, single-hop networks do not need routing protocols at all. Therefore, before developing a protocol one must study the network architecture properly and design it accordingly.

There are several other factors which should also be considered with the network architecture while designing an efficient protocol for underwater networks, such as long propagation delay, limited bandwidth, limited battery power, high bit error rate of the channel and several other adverse properties of the channel, such as, multi-path, fading and refractive behaviors. Moreover, the environment also has an impact on the performance of the protocols designed for underwater networks. Even temperature changes in a single day have an impact on the performance of the protocols. A good protocol designed for any network should consider some or all of these characteristics to achieve better performance.

In this thesis, we first discuss the impact of the environment on the performance of MAC and routing protocols. From our investigation, we discover that even temperature changes

within a day may affect the sound speed profile and hence, the channel changes and the protocol performance vary. After that we discuss several protocols which are specifically designed for underwater acoustic networks to serve different purposes and for different network architectures. *Underwater Selective Repeat (USR)* is an error control protocol designed to assure reliable data transmission in the MAC layer. One may suspect that employing an error control technique over a channel which already suffers from long propagation delays is a burden. However, USR utilizes long propagation by transmitting multiple packets in a single RTT using an interlacing technique. After USR, a routing protocol for surveillance networks is discussed where some sensors are laid down at the bottom of the sea and some sinks are placed outside the area. If a sensor detects an asset within its detection range, it announces the presence of intruders by transmitting packets to the sinks. It may happen that the discovered asset is an enemy ship or an enemy submarine which creates noise to jam the network. Therefore, in surveillance networks, it is necessary that the protocols have jamming resistance capabilities. Moreover, since the network supports multiple sinks with similar anycast address, we propose a Jamming Resistance multi-path *Multi-Sink Routing Protocol (MSRP)* using a source routing technique. However, the problem of source routing is that it suffers from large overhead (every packet includes the whole path information) with respect to other routing techniques, and also suffers from the unidirectional link problem. Therefore, another routing protocol based on a distance vector technique, called *Multi-path Routing with Limited Cross-Path Interference (L-CROP)* protocol is proposed, which employs a neighbor-aware multi-path discovery algorithm to support low interference multiple paths between each *source-destination* pair. Following that, another routing protocol is discussed for next generation coastal patrol and surveillance network, called *Underwater Delay-Tolerant Network (UDTN)* routing where some AUVs carry out the patrolling work of a given area and report to a shore based control-center. Since the area to be patrolled is large, AUVs experience intermittent connectivity. In our proposed protocol, two nodes that understand to be in contact with each other calculate and divide their contact duration equally so that every node gets a fair share of the contact duration to exchange data. Moreover, a probabilistic spray technique is employed to restrict the number of packet transmissions and for error correction a modified version of *USR* is employed.

In the appendix, we discuss a framework which was designed by our research group to

realize underwater communication through simulation which is used in most of the simulations in this thesis, called DESERT Underwater (short for DEsign, Simulate, Emulate and Realize Test-beds for Underwater network protocols). It is an underwater extension of the NS-Miracle simulator to support the design and implementation of underwater network protocols. Its creation assists the researchers in to utilizing the same codes designed for the simulator to employ in actual hardware devices and test in the real underwater scenario.

Sommario

Il viaggio delle comunicazioni acustiche sottomarine che cominció nell'era di Leonardo, é durato quattro secoli e mezzo prima di vedere messe in pratica le prime applicazioni per la Seconda Guerra Mondiale. Comunque, nelle ultime tre decadi le comunicazioni sottomarine sono state protagoniste di un massiccio avanzamento a causa dell'avanzamento della tecnologia costruttiva dei modem, delle periferiche e dei protocolli soprastanti. Inoltre, si sono aperte le porte a una grande varietá di applicazioni che possono utilizzare le comunicazioni sottomarine, come per esempio l'oceanografia, il monitoraggio dell'inquinamento, sorveglianza delle coste, assistenza alla navigazione, esplorazione al largo e prevenzioni di disastri che possono provenire dal mare. Diverse applicazioni potrebbero avere differenti richieste e caratteristiche, quindi la struttura di rete richiesta potrebbe differire tra le varie applicazioni. Per esempio, i protocolli di routing per reti multi-hop non connesse non sono adatti a reti Delay-Tolerant. Inoltre, le reti single-hop non necessitano di un protocollo di routing. Per questi motivi, prima di progettare un protocollo, bisogna studiare l'architettura di rete e progettare i relativi protocolli concordemente.

Ci sono numerosi altri fattori che dovrebbero essere presi in considerazione nell'architettura di rete nella progettazione di un protocollo di rete, come lunghi tempi di propagazione, banda limitata, durata limitata della batteria dei modem, alto Bit Error Rate nel canale e molte altre proprietá avverse del canale come multipath fading e un elevato grado di rifrazione. Inoltre, l'ambiente ha un impatto sulle performance dei protocolli progettati per le reti underwater. Per esempio, anche la variazione della temperatura in un singolo giorno ha un impatto sulle prestazioni dei protocolli. Un buon protocollo dovrebbe tenere conto di queste caratteristiche per raggiungere buone prestazioni.

In questa tesi, innanzitutto discutiamo l'impatto dell'ambiente sulle prestazioni di pro-

tocolli MAC e routing. Dalle nostre sperimentazioni, abbiamo scoperto che anche i cambiamenti di temperatura in un giorno provocano la variazione dell'SSP, quindi il canale varia e anche le prestazioni dei protocolli. Successivamente, abbiamo discusso numerosi protocolli progettati per varie applicazioni di comunicazione sottomarina con varie architetture di rete. Underwater Selective Repeat (USR) é un protocollo per il controllo d'errore progettato per assicurare trasmissioni affidabili di dati a livello MAC. Si potrebbe pensare che adottare un protocollo per il controllo d'errore su un canale giá affetto da grandi ritardi di propagazione provochi un carico eccessivo di traffico. Tuttavia, USR utilizza costruttivamente i grandi tempi di propagazione per trasmettere piú pacchetti in un Round Trip Time utiiizzando una tecnica ad interlacciamento, quindi, diminuendo il carico di dati. Dopo USR, un protocollo di routing per reti di sorveglianza dove alcuni nodi sono piazzati nel fondale e alcuni sono fuori dall'acqua é stato discusso. Se un sensore intercetta una attivitá nel suo raggio di copertura, annuncia la presenza di un intruso attraverso un messaggio al SINK. Potrebbe succedere che l'intruso sia una nave nemica o un sottomarino nemico che creano disturbi alla rete acustica sottomarina. Quindi, nelle reti di sorveglianza é importante che i protocolli di rete abbiano una resistenza ai disturbi. Inoltre, siccome la rete supporta una modalitá multisink con simile indirizzo anycast proponiamo un protocollo di routing multicast resistente al jamming, Multi-Sink Routing Protocol (MSRP), che utilizza tecniche di source-routing. Tuttavia, uno dei maggiori problemi delle tecniche di routing basate su source-routing, é un grande over-head di dati (ogni pacchetto include tutti i dati del percorso) e,inoltre,il problema dell' "unidirectional link". Quindi, un altro protocollo basato sulla tecnica Distance-Vector chiamato L-CROP (Multi-path Routing with Limited Cross-Path Interference) é stato proposto, che impiega un algoritmo neighbour-aware per instaurare un percorso multi-path a bassa interferenza tra mittente e destinatario del messaggio. A seguire un altro protocollo di routing é stato discusso per le reti di sorveglianza e di pattugliamento costiero di prossima generazione, chiamato UDTN (Undertwater Delay-Tolerant Networks), dove alcuni AUV svolgono il lavoro di pattugliamento e riportano i dati a un centro di controllo sulla costa. Siccome l'area da pattugliare é vasta, l' AUV avrá connessione intermittente con la base. Nel protocollo progettato, i nodi preposti a contattare la base calcolano e dividono la durata delle loro connessioni in maniera equa, cosícché ogni nodo abbia la stessa durata di connessione per scambiare dati. Inoltre viene impiegata, una

tecnica ”probabilistic spray” per restringere il numero di trasmissioni. Per quanto riguarda la correzione d’errore, una versione modificata di USR é stata adottata.

Nell’appendice, abbiamo presentato un simulatore che abbiamo progettato per realizzare la maggior parte delle simulazioni presenti in questa tesi, chiamato DESERT (DEsign, Simulate, Emulate and Realize Test-beds for Underwater Networks). É una estensione del simulatore NS-Miracle progettato per supportare simulazioni di protocolli per reti acustiche sottomarine. Questo simulatore assite il ricercatore nell’utilizzo di hardware e nel test dei protocolli in uno scenario reale.

Chapter 1

Introduction

Contents

1.1 Motivation

The concept of retrieving information by observing water started back from the time of Leonardo Da Vinci who discovered the possibility of detecting a distant ship by listening to a long tube submerged under the sea [1]. That journey of underwater communication which began from Leonardo's era took four and a half centuries to find practical applications for military purposes during World War II. However, over the last three decades, underwater acoustic communications witnessed a massive development due to the advancements in the design of underwater communicating peripherals and their supporting protocols [1]. Curiosity, necessity and security, these are the three principal aspects which are attracting researchers, militaries and industries to spend valuable time and money in this field. Successively, doors are opened for a wide range of applications to employ in the underwater environment. Some possible applications of underwater communication are described below:

• **Oceanography:** Underwater Networks can assist in the study of the deep sea and shallow

coastal oceans. Sensors and AUVs can be used for collecting long-term scientific data from the ocean. The Autonomous Ocean Sampling Network (AOSN) is one such kind of network in Monterey Bay which employs AUVs for monitoring large areas of the coastal ocean. Three field experiments were performed in 2000, 2003 and 2006, where various of methods for monitoring were tested [2,4]. These experiments demonstrated the advantages of employing sophisticated underwater devices to improve the observation ability and prediction capabilities of the oceanic environments. Another example of a sensor network designed for such kind of application is proposed in [5] which can be used to detect extreme temperature gradients that is considered as a suitable breeding ground for some micro-organisms.

- **Underwater explorations and monitoring:** Exploration of underwater oil, gas and valuable mineral reservoirs can advance the economy of a country. Let's consider the case study of Brazil. At the end of 2007, Brazil discovered a huge underwater oil reservoir which is going to establish them as the next major oil exporter in the world in the future [6]. There are still many valuable minerals, gas and oil reservoirs under the sea which are yet to be discovered. Underwater networks can help discovering those materials. It also can be employed to control and monitor the equipments used to collect minerals at low cost and with more flexibility.

- **Environmental pollution monitoring:** Another major application of underwater communication is the pollution monitoring of the environmental systems. Lake, river or sea water can be polluted in many ways like industrial and sewage waste, bilge bottom waste, dumps, water transports, etc. Underwater sensor networks can be deployed to detect such pollution by detailing the chemical slurry of the water [7].

- **Coastal patrol and surveillance:** Underwater networks comprising of sensors and/or AUVs, like the one in Figure 1.1, can reduce the cost of coastal patrol and surveillance missions for which the governments (that own water territory) spend a lot of money. An example of sensor network for surveillance is proposed in [4] where sensors (which are planted at the bottom of the sea) inform the shore-based control center after detecting an intruder. Another underwater network is proposed in [1] for coastal patrolling and surveillance system, where AUVs patrol an area of interest. Whenever an AUV

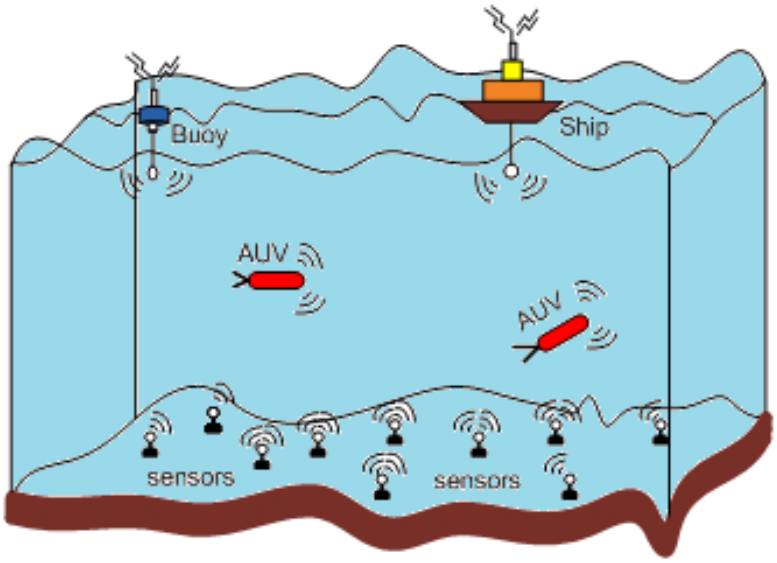


Figure 1.1. *A coastal patrol and surveillance network comprising of sensors, AUVs, a sink and a ship.*

detects an intruder, it starts following the asset and sends related information to the shore-based control center to take appropriate action.

- **Disaster prevention:** Underwater sensor networks can be deployed to measure seismic activity of the oceans. These types of networks can detect and provide tsunami warnings to the coastal areas [10]. They also can be utilized to study submarine earthquakes [2].

- **Mine reconnaissance:** Deepwater drones or AUVs can be used to detect mines planted at the bottom of the sea. According to a news published in [11], the US Army has already developed underwater mine-detecting drones which can detect mines, but cannot defuse them. It would be challenging to build robots which not only detect the mine, but also defuse them without any risk.

It can be observed from the previous discussion that different applications have different characteristics and hence, may require different network architectures. For instance, the coastal patrol and surveillance network proposed in [1] experiences intermittent connectivity and hence, DTN architecture is employed; where as, pollution monitoring and disaster

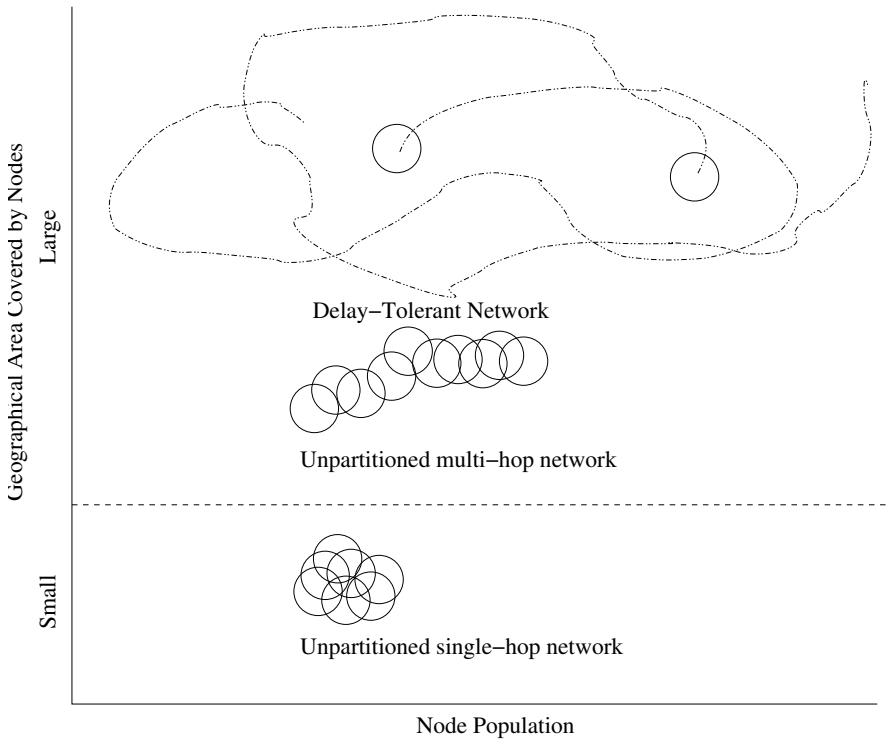


Figure 1.2. *Networks classified with respect to spatial coverage.*

prevention applications suggest a fixed deployment of the sensors. According to spatial coverage we can divide these networks mainly in three types which are illustrated in Figure 1.2

- **Single-hop networks:** In these networks, all nodes remain in contact with each other. This class can be further divided in centralized and distributed single-hop networks. If all nodes deliver data to a centralized station also known as sink, it is called centralized; whereas, in distributed, all nodes have sending and receiving capability. No special routing protocol is necessary for the communication.

- **Unpartitioned Multi-hop networks:** When a network has to cover a large area where single-hop is not possible and packets travel multiple hops to reach the destination, multi-hop networks are employed. These types of networks require routing protocols to deliver the data to the right destination. Every node must establish a path before transmitting a packet.

- **Delay-tolerant networks:** There are some applications in underwater networks where it is not possible to establish a path between the source and the destination, which implies

that unpartitioned multi-hop networks are not suitable in these applications. Usually routing protocols used in this kind of network architecture have *store and forward* capability. A node stores the data it receives from the upper layer or from the other node(s) and forwards it to the other node(s) with the hope that the receiving node is the destination or at least it can deliver the packet to the destination at a later time.

Since different networks have different characteristics, protocols designed for a specific network architecture must take its characteristics into account. For instance, routing protocols designed for unpartitioned multi-hop networks are not suitable for Delay-Tolerant Networks. Furthermore, single-hop networks do not need routing protocols at all. Therefore, before developing a protocol one must study the network architecture properly and design it accordingly. Beside network architecture, there are several other factors which should also be considered when designing an efficient protocol for underwater networks; described in the following:

- Usually in water, radio wave signals can travel long distances at extra low frequencies (30Hz-300Hz) which however require large antennas and high transmitter power, and therefore, are not preferable for underwater communication. On the other hand, optical waves do not suffer so much from attenuation, but scattering. Therefore, acoustic signals are preferred which can cover large communication areas. However, the propagation speed of acoustic signals is five orders of magnitude lower than that of radio signals; hence, they suffer large propagation delays.
- The underwater channel faces several challenges due to multi-path, fading and refractive properties of the sound channel.
- The available bandwidth of the underwater channel is severely limited.
- Underwater channel experiences high bit error rates.
- Battery power is limited and most of the time batteries cannot be recharged since solar energy cannot be exploited.
- Underwater channels can experience shadow zones which are spatial regions where almost no acoustic signals exist and cause connectivity dropouts.

- Environment also has an impact on the performance of the protocols designed for underwater networks [12]. Not only do seasonal changes affect the performance of the protocols, but also temperature changes in a single day have an impact on the performance of the protocols.

A protocol designed for any underwater network should utilize these factors of the condition to achieve better performance. The above described factors also make it different from other existing wireless terrestrial networks. Therefore, protocols designed for terrestrial networks are not suitable for underwater communication in general.

In this thesis, we describe several protocols which are specifically designed for underwater networks for various applications and for various network architectures. For instance, Underwater Selective Repeat (USR) is an error control protocol designed to assure reliable data transmission in the MAC layer, since underwater channels are vulnerable to error. USR also utilizes long propagation by transmitting multiple packets in a single RTT using an interlacing technique. The routing protocols proposed in this thesis employ a multipath technique, which assists them in achieving higher packet delivery, jamming resistance and fault tolerance capability. The DTNs routing protocols proposed, employ a modified version of the USR technique for error correction and for utilizing long propagation delays. Moreover, two nodes that understand to be in contact with each other calculate and divide their contact duration[1] equally so that every node gets a fair share of the contact duration to exchange data.

1.2 Discussion and Organization of this Thesis

The rest of the thesis is subdivided into several chapters, each containing a specific topic and the corresponding results, so that each chapter can almost be read separately.

In Chapter 2, we discuss the impact of the environment on the performance of MAC and routing protocols. From our investigation, we discover that even temperature changes within a day may affect the sound speed profile and hence, the channel changes and the protocol performance varies.

[1]Unlike terrestrial networks, contact duration in underwater networks is usually long since the acoustic signals can cover a longer area and the mobile devices move more slowly than in terrestrial networks.

In Chapter 3, an error control protocol, named Underwater Selective Repeat (USR), is discussed which is designed to assure reliable data transmission at the MAC layer. One may suspect that employing an error control technique over a channel which already suffers from long propagation delay is a burden. However, USR utilizes long propagation by transmitting multiple packets in a single RTT using an interlacing technique.

In Chapter 4, a routing protocol for surveillance networks is discussed where some sensors are laid down at the bottom of the sea and some sinks are placed outside the area. If a sensor detects an asset within its detection range, it announces the presence of intruders by transmitting packets to the sinks. It may happen that the discovered asset is an enemy ship or an enemy submarine which creates noise to jam the network. Therefore, in surveillance networks, it is necessary that the protocols have jamming resistance capabilities. Moreover, since the network supports multiple sinks with similar anycast address, we propose a Jamming Resistance multi-path Multi-Sink Routing Protocol (MSRP) using a source routing technique. However, the problem of source routing is that it suffers from large overhead (every packet includes the whole path information) with respect to other routing techniques, and also suffers from the unidirectional link problem. Therefore, another routing protocol based on a distance vector technique, called Multi-path Routing with Limited Cross-Path Interference (L-CROP) protocol is proposed, which employs a neighbor-aware multi-path discovery algorithm to support low interference multiple paths between each *source-destination* pair.

In Chapter 5, we discuss two routing protocols for DTNs, called Underwater DTN (UDTN) and Underwater DTN with Probabilistic spray (UDTN-Prob) routing protocol. In both protocols, two nodes that understand to be in contact with each other calculate and divide their contact duration equally so that every node gets a fair share of the contact duration to exchange data. In addition, for error correction a modified version of the USR is employed. The major difference between UDTN and UDTN-Prob protocols is the utilization of the probabilistic spray technique to restrict the number of packet transmissions.

Furthermore, in Appendix A, we discuss a framework which was designed in our group to realize underwater communication through simulation which is used in most of the simulations in this thesis, called DESERT Underwater (short for DEsign, Simulate, Emulate and Realize Test-beds for Underwater network protocols). It is an underwater extension of the

NS-Miracle simulator to support the design and implementation of underwater network protocols. Its creation assists the researchers in utilizing the same codes designed for the simulator in actual hardware devices and test in the real underwater scenario.

References

[1] M. Stojanovic, "Acoustic (underwater) communications," *entry in Encyclopedia of Electrical and Electronics Engineering, John G. Proakis, Ed., John Wiley & Sons*, vol. 22, pp. 688–698, 1999.

[2] M. Chitre, S. Shahabudeen, and M. Stojanovic, "Underwater acoustic communications and networking: Recent advances and future challenges," *Marine Tech. Soc. Journal*, vol. 42, no. 1, pp. 103–116, spring 2008.

[3] I. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: research challenges," *Elsevier's Ad Hoc Networks*, vol. 3, no. 3, 2005.

[4] "The autonomous ocean sampling network." [Online]. Available: http://www.mbari.org/twenty/aosn.htm

[5] B. Zhang, G. S. Sukhatme, and A. A. G. Requicha, "Adaptive sampling for marine microorganism monitoring," in *Proc. of IEEE/RSJ Internatioanl Conference on Intelligent Robots and Systems*, Sendal, Japan, Oct. 2004.

[6] "Underwater oil discovery to transform brazil into a major exporter." [Online]. Available: http://www.nytimes.com/2008/01/11/business/worldbusiness/11iht-oil.1.9147825.html

[7] X. Yang, K. G. Ong, W. R. Dreschel, K. Zeng, C. S. Mungle, and C. A. Grimes, "Design of a wireless sensor network for long-term, in-situ monitoring of an acquistic environment," *Sensors*, vol. 2, pp. 455–472, 2002.

[8] M. Goetz, S. Azad, P. Casari, I. Nissen, and M. Zorzi, "Jamming-resistant multi-path routing for reliable intruder detection in underwater networks," in *Proc. of ACM WUWNet*, Seattle, Washington, USA, Dec. 2011.

[9] S. Azad, P. Casari, and M. Zorzi, "Coastal patrol and surveillance networks using AUVs and delay-tolerant networking," in *Proc. of MTS/IEEE OCEANS*, Yeosu, South Africa, May 2012.

[10] N. N. Soreide, C. E. Woody, and S. M. Holt, "Overview of ocean based buoys and drifters: Present applications and future needs," in *Proc. of MTS/IEEE OCEANS*, Honolulu, HI, 2001.

[11] "Underwater mine-detecting robots developed by us army." [Online]. Available: http://www.smartplanet.com/blog/smart-takes/underwater-mine-detecting-robots-developed-by-us-army/26358

[12] S. Azad, C. P. Paolo Casari, R. Petroccia, and M. Zorzi, "On the impact of the environment on mac and routing in shallow water scenarios," in *Proc. of IEEE/OES OCEANS*, Santander, Spain, 2011.

Chapter 2

Impact of the Environment on MAC and Routing in Shallow Water Scenarios

Contents

2.1 Overview

The recent advances in underwater acoustic modem technology have fostered the interest in undersea exploration using autonomous fixed and mobile objects, and motivate the investigation of new protocols and applications for underwater networking. In particular, considerable attention is being given to the feasibility of underwater wireless sensor networks (UWSNs).

The CoLlAborative eMbedded networks for submarine surveillance (CLAM) project,

funded by the European Commission under the 7th Framework Programme, has the objective to reduce the complexity and cost of pervasive and fine-grained underwater surveillance. This problem, inherently a formidable task due to the harshness of the environment and to the limited technology we can count on today, will be tackled via remotely controlled and cooperating underwater sensor nodes. These nodes will be networked via acoustic communications over multi-hop topologies. Information flows are foreseen to be bi-directional: from a control center to the nodes (to change configuration parameters, perform task allocation, control node operation) and from the nodes to the control center (to provide reports on the sensed data). For this purpose new protocol solution will be proposed with particular attention to Medium Access Control (MAC) and routing protocols.

Recently, many solutions for underwater MAC and Routing have been proposed, mostly based on variants of typical terrestrial radio approaches.These methods rely on simple modifications of the ALOHA scheme [1], of Carrier-Sense Multiple Access (CSMA) without [3,21] and with [16] collision avoidance, of TDMA [5,6], and CDMA [7]. Hybrid schemes have also been proposed [8,9]. The performance of some of these schemes has also been compared by means of simulations [10]. Similarly, several routing solutions have been proposed, for both fixed and mobile networks (see, e.g., [11] for an overview).

For both routing and MAC protocols, most of the study and evaluation phases have been carried out via simulations. However, every network simulation software needs to approximate the behavior of sound propagation in the water: therefore, the reproduction of actual channel variability will also be approximated (see, e.g., the discussion related to the Bellhop ray tracing package in [12]). In turn the behavior of the protocols simulated on top of approximated channel behaviors may be different than what would happen in practice. Other than the error related to our knowledge of the environmental data and to the way we exploit it to model sound propagation, simulation packages may also be subject to other systematic errors, if they do not typically reflect the dynamics of the environment. In fact, e.g., the sound speed profile changes over time, the profile of surface waves evolves, and these changes can affect the accuracy of the simulation results. This is especially true for shallow water scenarios, where the channel dynamics strongly affect the quality of network links over time.

Because typical scenarios in the CLAM project entail shallow water monitoring, the first contribution of this work is to improve the network simulation tool of choice in what follows, namely ns2-Miracle [20] connected with the Bellhop propagation simulator [22] via the WOSS extensions [21], to make it capable to track environmental changes over time. This helps mitigate the problem introduced above, and makes the results of the simulations more realistic, provided that actual data retrieved *in situ* is available, e.g., in the form of SSP samples taken at fixed time intervals during one or more days. In particular, this feature is key to run evaluations that cover large spans of simulated time. In addition, we also consider the reproduction of surface wave patterns according to standard surface wave spectrum functions: this also makes the computation of propagation patterns more realistic. Ultimately, this whole effort improves the convenience of simulation as a tool to experiment what would be too difficult to test in real life.

It is worth noting that some works considered the behavior of networking protocols in the presence of different environmental data. Among these, [15] focuses on the simulation of MAC protocols using WOSS [21], using environmental data from both the summer and the winter seasons. However, the authors do not consider changes over a shorter time scale and their effects on protocols, as we do instead in the present investigation.

The second contribution of our work is a study of how the time-varying behavior of the underwater acoustic channel (as accounted for in simulations) impacts the performance of network protocols over time. We perform this evaluation over scenarios of practical interest for the CLAM system.

We study the effects of different environmental conditions on the performance of network protocols, with focus on MAC and routing. We also show that it is convenient to let the network adapt to changing environmental conditions, e.g., by recalculating the routes packets should follow on the way towards the sink over time. Our results show that this is actually beneficial not only in terms of typical metrics such as throughput and packet delivery ratio (which improve by about 10 to 20%), but also in terms of transmission efficiency, as the shorter routes devised by allowing adaptation to the environment help avoid useless data replication over multiple hops.

2.2 Considered Protocols

For the forthcoming comparison, we consider two different MAC protocols (CSMA-ALOHA and DACAP), that have been implemented in our framework.

CSMA–ALOHA is a form of ALOHA with a very short channel sensing phase before a packet is actually transmitted. Given that the typical balance between the packet transmission time and the propagation delay in underwater networks makes CSMA inefficient, the short sensing introduced here serves the only purpose to avoid a trivial collision scenario (i.e., where the current node starts its own transmission while a different signal is propagating in the same area where the node is located, resulting in a likely collision at the intended receiver of such signal). The sensing time is randomized to avoid the synchronization of channel access attempts and repeated collisions. In case the channel is found busy, the transmitter employs a standard binary exponential backoff scheme. This protocol is a mixed form of both ALOHA and CSMA, hence the name.

The **Distance Aware Collision Avoidance Protocol** (DACAP) [16] uses an extended version of a Request-To-Send/Clear-To-Send (RTS/CTS) handshake for reserving the channel before packet transmission. More specifically, when a node has a packet to send, it senses the channel: if the channel is idle, the node transmits an RTS. Upon receiving the RTS, the destination replies with a CTS and then waits for the data packet. With respect to the usual CSMA/CA scheme, DACAP adapts to the underwater channel characteristics by using the following mechanism. If, while waiting for a data packet, a destination node overhears a control packet for some other node, it sends a short warning packet to its transmitter. In addition, every sender defers the data packet transmission for a prescribed amount of time, after receiving a CTS. If it overhears another control packet or receives a warning packet from the destination during this period, the sender aborts the current transmission attempt. The length of the defer time depends on the distance between the source and its destination, which the sender can estimate using the RTS/CTS round-trip time.When the receiver overhears an RTS and sends a warning, it does not know whether the warning will reach the sender on time to make it abort the transmission. Since a data packet can still arrive, the receiver must continue listening to the channel even after a warning has been sent. For this reason, the defer time is defined as the minimum waiting period between receiving the CTS

and sending the data that guarantees the absence of harmful collisions.

2.3 Extensions to WOSS and Scenario Description

The World Ocean Simulation System (WOSS) [21] employs oceanographic databases for environmental parameters in order to bring realistic acoustic propagation patterns into ns2-Miracle using the Bellhop ray tracing software [22]. In particular, WOSS has been interfaced with public databases such as the World Ocean Database for SSPs [20], the General Bathymetric Chart of the Oceans [21] for bathymetry and the National Geographic Data Center's Deck41 Database [18] for bottom sediments. Furthermore, it also supports data retrieval from the user-custom datasets.

We extended the capabilities of WOSS to support the processing of time-varying environmental conditions. For example, this allows to implement SSP changes over the duration of a day. However, we note that SSP measurements are typically available only at certain time epochs throughout a day, whereas the simulator should model the transition between the current and the next SSP samples as well. This would include a much more detailed representation of the environment (e.g., considering all phenomena that cause sudden temperature changes such as currents and internal waves). In order not to put too much complexity in the simulator, we model the transition from one SSP sample to the following one via a simple convex combination as follows:

$$\mathrm{SSP}(t) = \mathrm{SSP}(t_i)\frac{t_{i+1} - t}{t_{i+1} - t_i} + \mathrm{SSP}(t_{i+1})\frac{t - t_i}{t_{i+1} - t_i} \tag{2.1}$$

where t_i and t_{i+1} are the time epochs when the ith and $(i+1)$th samples of the SSP have been measured, respectively, and t is the current time epoch, $t_i \leq t < t_{i+1}$. The user is allowed to specify as many SSPs as needed, possibly spanning more days. In addition, we implemented a random generation of surface wave profiles, which have been set to obey a Bretschneider 2-parameter spectrum [19].[1]

The SSP samples we use in this investigation have been collected during the SubNet'09 sea trials, which took place off the eastern shore of the Pianosa Island, Italy ($42.585°$N,

[1]The characteristic height has been set to $\bar{H}_{char} = 1.5\,\mathrm{m}$ and the average wave period to $\bar{T} = 3\,\mathrm{s}$. This yields surface wave realizations matching a windy day.

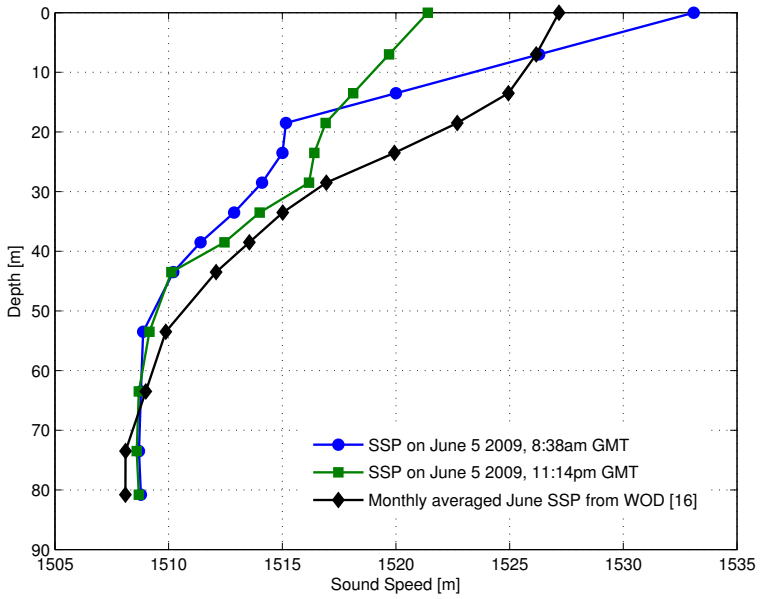
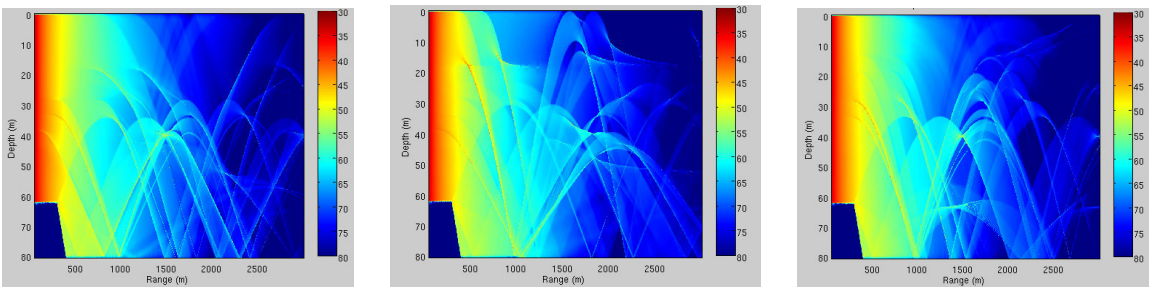


Figure 2.1. *Sound speed profile realizations taken during the SubNet'09 campaign off the island of Pianosa, Italy, between June 5 and 6, 2009. A monthly average of the SSP across the month of June, taken from the WOD 2009 database [20], has been included for comparison*

10.1°E) [12]. This location has therefore been chosen as the simulation area. From the SubNet'09 dataset, we pick 8 distinct SSP samples spanning the duration of one day on June 5th, 2009. Some SSP realizations used in the simulation are shown in Fig. 2.1: note that the database samples do not include sound speed measurements at very shallow depths: therefore, we linearly extrapolated the SSP starting from the last two known speed samples (located at a depth of 18 and 13 m, respectively). For comparison, we also depict in Fig. 2.1 the monthly average of the SSPs realizations in the WOD 2009 database, i.e., the realization which WOSS would automatically retrieve if not supplied with external data. We observe that the measurements and the monthly average are in fact similar, with the exception that the latter yields slightly higher sound speed values at low depth. This difference is amplified by the fact that the SubNet'09 SSPs we used have been taken on the 5th of June, whereas the monthly SSP average contains a significant contribution from later June days, when surface waters are typically warmer, yielding higher sound speed.

These variations can substantially affect the way sounds travel through water. In fact, the change of sound speed across the water column causes the trajectory of sound waves to



(a) Nighttime SSP (b) Daytime SSP (c) June SSP from WOD 2009

Figure 2.2. *Attenuation profiles obtained with Bellhop using (a) a nighttime SSP (b) a daytime SSP and (c) the monthly averaged June SSP automatically retrieved by WOSS from the World Ocean Database 2009 [20]. In the bottom-left corner of the pictures we observe the bathymetry profile as extracted by WOSS from General Bathymetric Chart of the Oceans [21].*

bend, and can thus change the level of the sound pressure in time at any point in the network area. To exemplify this phenomenon, two channel realizations obtained via Bellhop[2] are shown in Figs. 2.2(a) and 2.2(b), using a nighttime and a daytime SSP, respectively. For comparison, a realization generated using the June SSP from WOD 2009 has been shown in Fig. 2.2(c). From these patterns we observe that the attenuation of a signal at a given point in space can vary significantly due to the varying refraction effects caused by different temperature gradients throughout the day. For example, acoustic rays bent downward relatively sharply in the daytime, whereas during nighttime this refraction effect is less pronounced, which makes the insonification more uniform and able to reach slightly larger distances. For example, consider the attenuation in Figs. 2.2(a) and 2.2(b) at a depth of 0 to $15\,\mathrm{m}$ from a distance of about $2000\,\mathrm{m}$ from the source (located on the left in the pictures at a depth of $40\,\mathrm{m}$): during daytime, the attenuation is very high and almost no sound reaches the area due to strong downward refraction; conversely, the attenuation is estimated to about $70\,\mathrm{dB}$ during nighttime, thanks to a more uniform SSP and to the bottom bounces which reach the top of the water column thanks to milder downward refraction. For comparison, the WOD 2009 SSP in Fig. 2.2 shows an even larger shadow zone with little sound propagation extending deeper than in Fig 2.2(b), whereas the insonification level before a distance of $2000\,\mathrm{m}$ is comparable to that in Figs. 2.2(a) and 2.2(b). Such changes may have a significant impact on the performance of networking protocols, as will be discussed in Subsections 2.4.1 and 2.4.2.

[2]We consider here the "incoherent" propagation profile calculation option.

In the following evaluation, we assume that all nodes transmit using a Binary Phase Shift Keying (BPSK) modulation technique at a bit rate of $4800\,\mathrm{bps}$ at a central frequency of $25\,\mathrm{kHz}$, similar to one of the possible configurations of the WHOI micromodem [20]. The length of data and ACK packets is fixed at $L_D = 125\,\mathrm{Bytes}$ and $L_A = 10\,\mathrm{Bytes}$, respectively. The traffic is generated randomly according to a Poisson process of fixed rate λ packets per second in the whole network. We recall that the network area chosen for the simulations is off the eastern coast of the Pianosa Island, in Italy, at about $42.585°\mathrm{N}$, $10.1°\mathrm{E}$. All nodes are randomly placed within the network area except the sink, which is centrally placed. The channel is periodically re-computed according to the new propagation conditions dictated by the SSP changes as per Eq. 2.1.

We will now consider two different sets of network results: the first focuses on MAC performance, whereas the second one focuses on routing performance. In the MAC test scenario, we consider a network of 19 nodes plus one sink, centrally placed at a depth of $3\,\mathrm{m}$. The network is put under stress by considering a relatively small area of $3\,\mathrm{km} \times 1.5\,\mathrm{km} \times 80\,\mathrm{m}$, where however the nodes located farthest from the sink still have a fair probability of delivering a packet correctly (albeit this probability may decrease according to SSP changes). In the routing test scenario, we consider a larger network area of $5\,\mathrm{km} \times 5\,\mathrm{km} \times 80\,\mathrm{m}$, in order to increase the average distance between the nodes, their neighbors, and the sink. In this case, the farthest nodes are not guaranteed any consistently good-quality link towards the sink, and must therefore resort to routing in order to have their own packets delivered. All simulation results are averaged over 25 different rounds of MAC experiments and 45 different runs for routing experiments.

2.4 Simulation Results

2.4.1 MAC results

We start our comparison by considering the MAC-level performance of static topologies. We consider first the CSMA-ALOHA MAC protocol described in Sec. 2.2. Figs. 2.3 and 2.4 depict the normalized throughput (i.e., the average number of packets delivered to their intended destination per packet transmission time) and the packet delivery ratio (i.e., the ratio of the correctly received packets over all sent packets) for a network of 19 nodes and

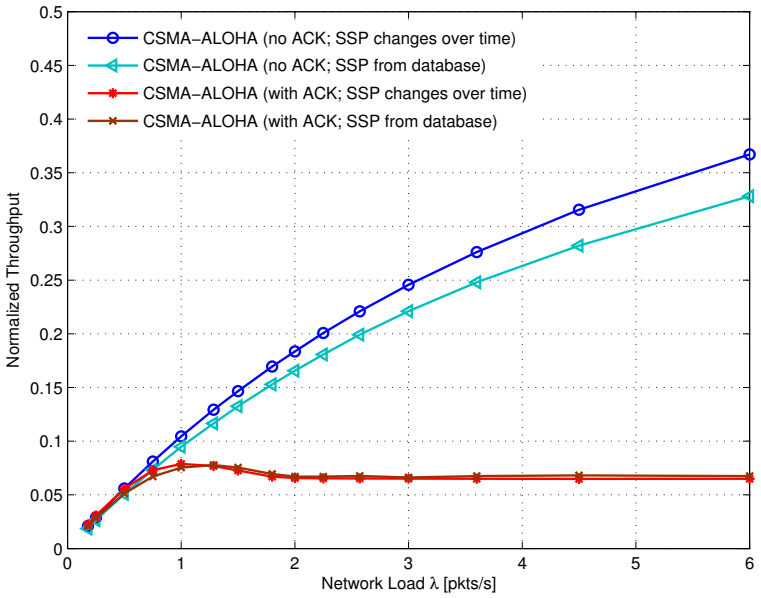


Figure 2.3. *Normalized throughput as a function of the traffic generation rate in the network using CSMA-ALOHA.*

one sink. Each figure contains four curves, corresponding to two different versions of the protocol, with and without Stop-and-Wait Automatic-Repeat reQuest, S&W ARQ: in the latter case, the transmitter sends a packet and waits for the confirmation of correct reception from the sender in the form of an acknowledgment message (ACK). The two versions have been evaluated both under changing SSP conditions and under the same (fixed) SSP that WOSS would retrieve from the WOD database.

By comparing the curves for fixed and varying SSP in Figs. 2.3 and 2.4, we have indeed observed different performance, mainly due to the stronger downward refraction caused by the averaged SSP in the WOD 2009 database, which deviates most of the sound pressure towards the bottom and does not favor long-range transmissions, and in turn causes packet reception errors. This effect is similar to that observed, e.g., in Fig. 2.2(b). Notice that, while negligible in terms of absolute values, the difference between the two no-ACK CSMA curves is about 10% and increasing for higher traffic, meaning that as the network approaches the point of maximum throughput, the value obtained using WOD's SSP is quite lower than what would be obtained in a real scenario, and makes the usage of more frequent environmental data samples worth whenever possible. However, when a more detailed data set is

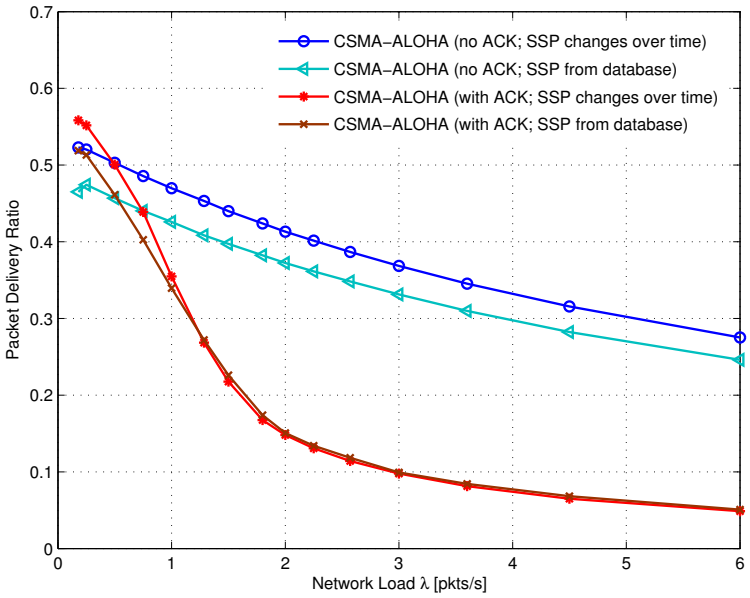


Figure 2.4. *Packet delivery ratio as a function of the traffic generation rate in the network using CSMA-ALOHA.*

not available, the curves of the protocols under the monthly averaged SSP show the same trend as those obtained with a varying SSP. Hence, higher-level performance indications, such as the network load for which the maximum throughput is reached, are similar. For example this is the case for the throughput and packet delivery ratio of CSMA-ALOHA with S&W ARQ in Figs. 2.3 and 2.4, respectively. In fact, the throughput curves show a very similar trend, and their difference is further mitigated by the inherently higher waiting times caused by S&W, which translate into lower throughput in both the fixed and the varying SSP cases. On the other hand, the slight difference between the two maxima of the throughput curves in Fig. 2.3 can be explained via the higher delivery ratio obtained in the varying SSP case (Fig. 2.4).

Similar observations apply to Figs. 2.5 and 2.6, which present normalized throughput and delivery ratio results for DACAP. In this case the throughput is comparatively lower than CSMA-ALOHA's, due to the requirement to perform 3-way and 4-way handshakes in the no-ACK and ACK cases, respectively. In addition, hidden terminal effects and collisions between control and data packets [10] adversely affect the capability to set up links, and decrease the throughput of the DACAP protocol.

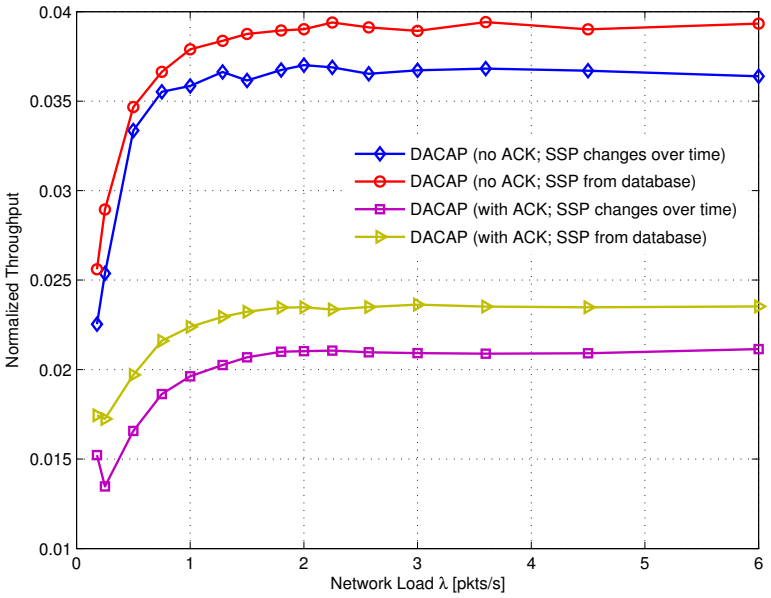


Figure 2.5. *Throughput as a function of the traffic generation rate in the network using DACAP.*

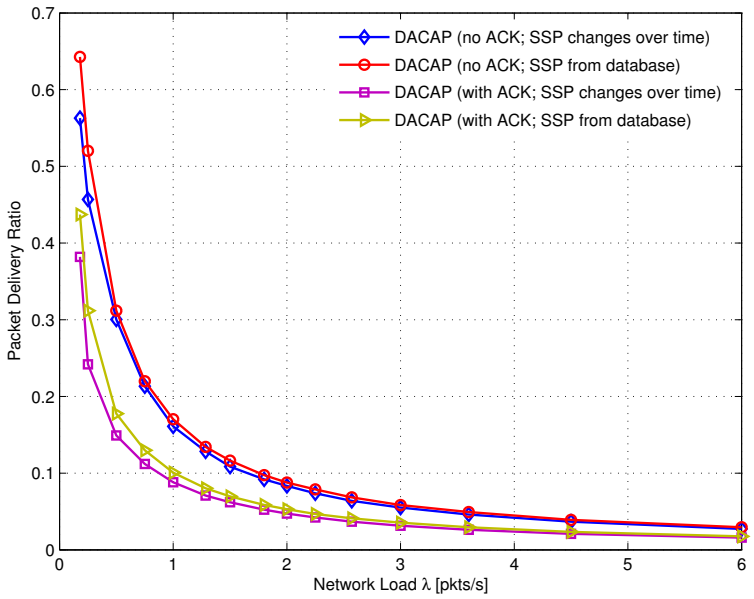
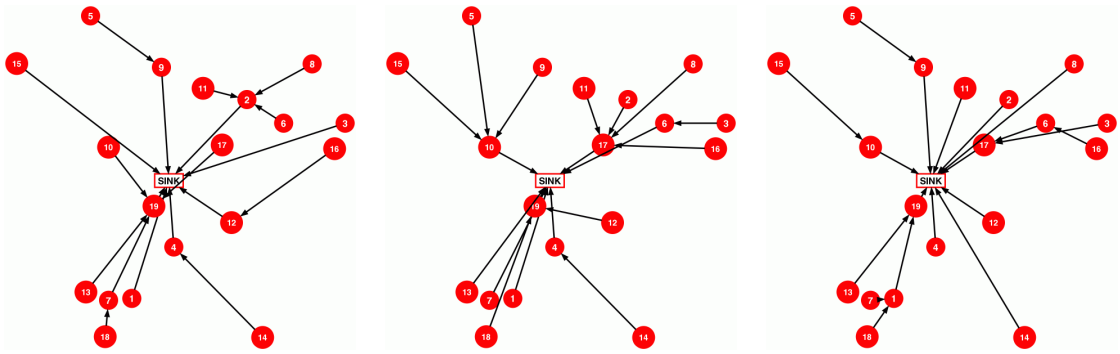


Figure 2.6. *Packet delivery ratio as a function of the traffic generation rate in the network using DACAP.*

2.4.2 Routing results

As the variation of the sound propagation pattern over time has an impact on the performance of MAC protocols, likewise such change affects the capability of a network to route



(a) After 3:20 hours (b) After 10:00 hours (c) After 20:00 hours

Figure 2.7. *Example of how optimal routes evolve as the SSP changes at different times during a day. The deployment is actually three-dimensional, but is seen from above and therefore depicted as two-dimensional for convenience.*

data. In this section, we present some results to show this phenomenon. We consider static topologies where 19 nodes are deployed in the same network area described above, with a sink centrally placed at a depth of $3\,\mathrm{m}$. In this evaluation, the network must convey (or converge-cast) the data generated by all nodes to the sink. Given the better performance of CSMA-ALOHA in the previous evaluation, we employ it as the medium access control technique in what follows. For simplicity, the network maintains static routes. This means that each node knows exactly what the following neighbor is on the path toward the sink. The objective of the study is to show that recalculating such routes once every so often yields performance improvements both in terms of the maximum data throughput and in terms of the length of the routes: these tend to be shorter, which avoids useless replication of transmitted packets over multiple hops.

In more detail, we take 8 SSP samples from the SubNet'09 data set, which span the duration of one day between June 5th and June 6th, 2009. For each of these samples, we run Bellhop to get a measure of the attenuation over the link between all pairs of nodes. We then calculate shortest paths to the sink via a simple Dijkstra algorithm, with the constraint that the link between nearest neighbors should span the longest distance and guarantee a Signal-to-Noise Ratio (SNR) of at least $15\,\mathrm{dB}$.[3] Network simulations are then performed by letting

[3] An even more realistic approach would require to perform multiple Bellhop runs for each link, each featuring a small randomization of the environmental parameters. The links to form the routes could then be constrained to have an *average* SNR (over all realizations) exceeding some threshold. However, this would sub-

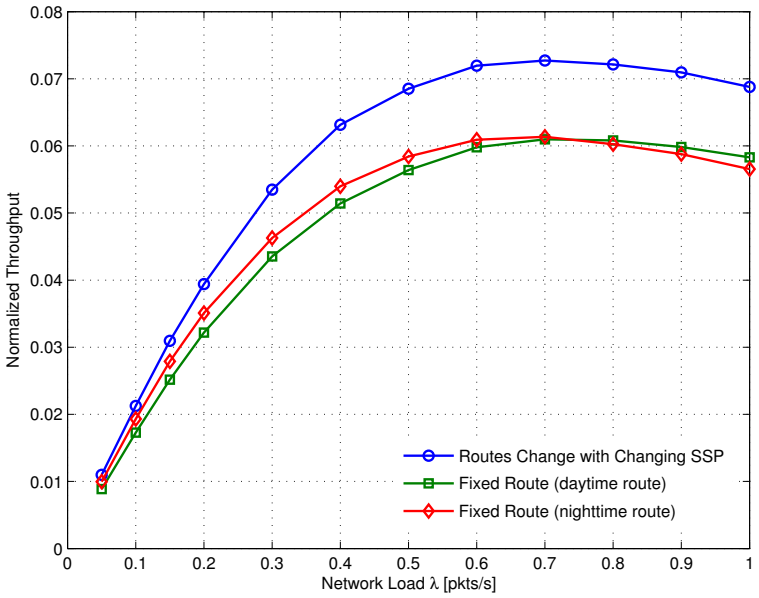


Figure 2.8. *Normalized throughput as a function of the traffic generation rate in the network using fixed and adaptive routes.*

the SSP change over time according to the discussion in Sec. 2.3. However, at each time epoch corresponding to one of the 8 SSP measurements taken from the SubNet'09 dataset, we recalculate all the routes to match the new propagation conditions. Figures 2.7(a)–(c) give an example of how such routes change over time at different epochs throughout the simulation. Note that we do not explicitly implement a route dissemination protocol: the study of a suitable approach for this is left as a future extension.

Figs. 2.8 and 2.9 show the throughput and packet delivery ratio as a function of the traffic generation rate in the network. This time, throughput is defined as the number of packets correctly delivered to the sink per packet transmission time. For each traffic rate, simulations are run for about 24 hours of simulated time, spanning all considered SSP samples. Along with the adaptive route case we will also evaluate a scenario with two different fixed routes, one computed using a daytime SSP, and a second one using a nighttime SSP. From Fig. 2.8, we observe that route changes cause a throughput increase of roughly 20% over fixed routes. This improvement is mainly due to the better packet delivery ratio (Fig. 2.9), consistently

stantially increase the computational burden. We therefore decided to perform only one realization per link, and compensated for small-scale changes by imposing a quite high threshold of $15\,\mathrm{dB}$.

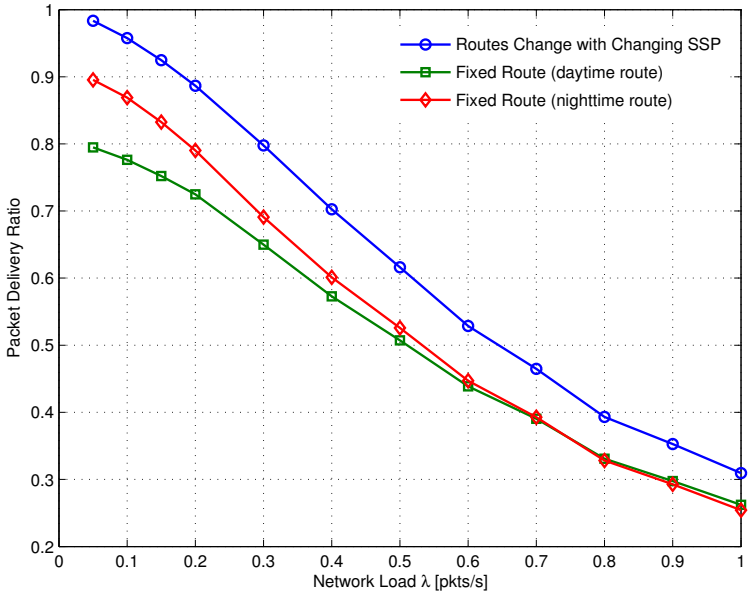


Figure 2.9. *Packet delivery ratio as a function of the traffic generation rate in the network using fixed and adaptive routes.*

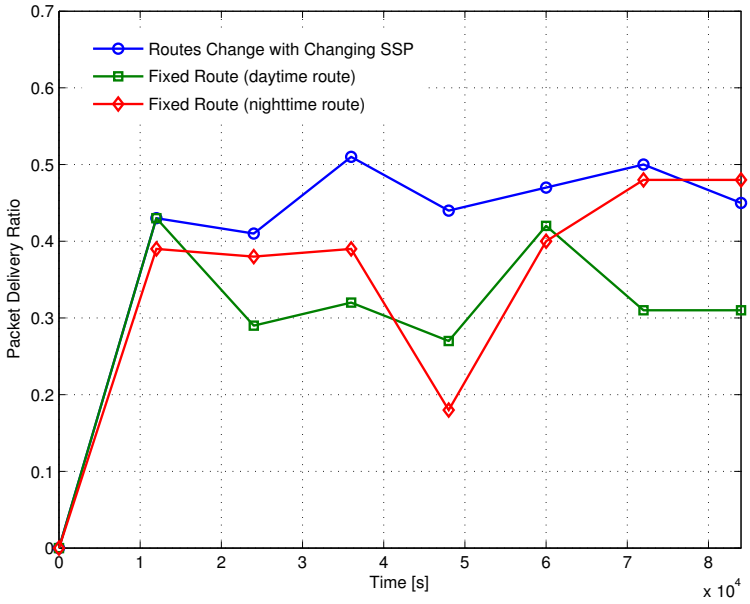


Figure 2.10. *Packet delivery ratio experienced over time by a specific node, using adaptive and fixed routes.*

10% better than both fixed route cases.

In spite of the relatively limited difference between the adaptive and fixed route cases,

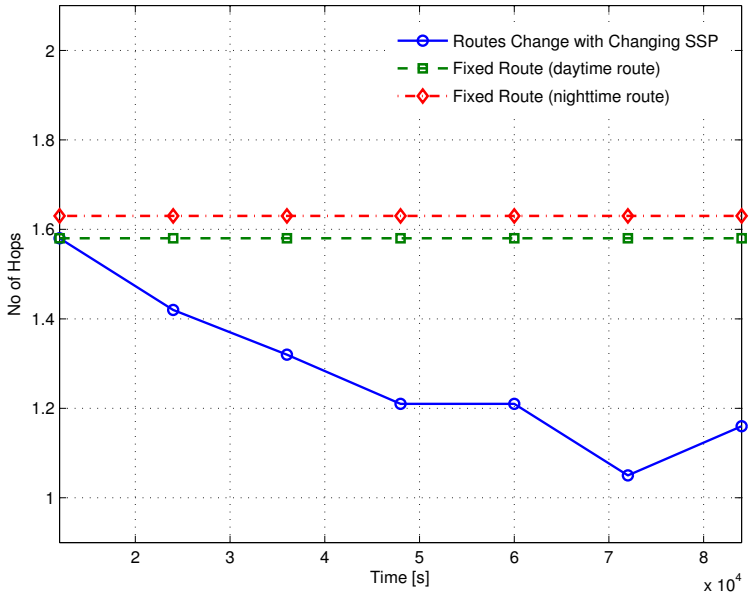


Figure 2.11. *Average route length in the network over time using adaptive and fixed routes.*

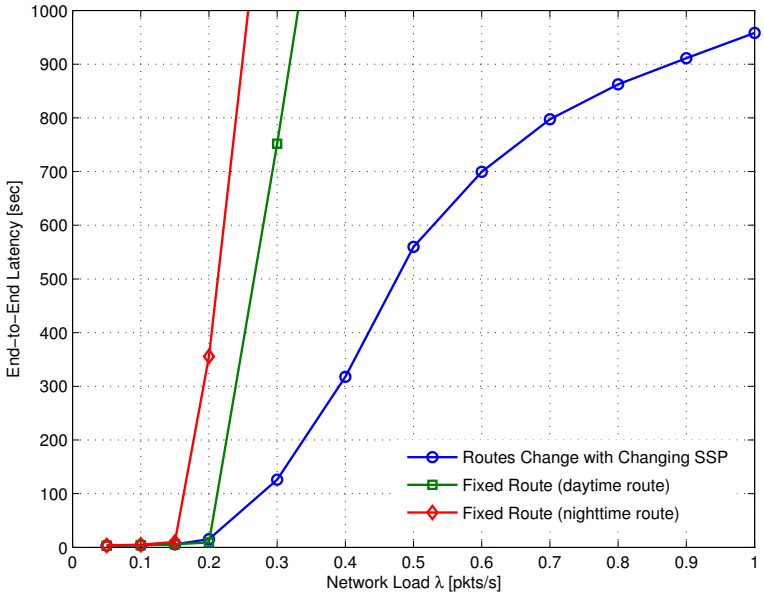


Figure 2.12. *Delivery delay as a function of the traffic generation rate in the network using fixed and adaptive routes.*

single nodes can suffer from worse performance and affect the network as a consequence. For example, consider Fig. 2.10, which depicts the packet delivery ratio experienced by a

specific node as a function of time, as it forwards its packets to the next hop towards the sink. Each point corresponds to the packet delivery ratio measured over the preceding 3 hours. As environmental conditions change with the time of day, the link between the two nodes can become quite unreliable. The usage of adaptive routes allows to choose next hops according to how much the link towards them is reliable, and therefore leads to a better and more stable performance. In addition, adapting routes allows to keep them short, avoiding useless packet replication through multiple nodes. This effect can be seen in Fig. 2.11, where we observe that the average route length decreases significantly with adaptive routes: this makes the network almost single-hop towards the end of the simulation. These results explain the quite sharp advantage gained by adaptive routes in terms of average delivery delay, defined as the time elapsed between the generation of a packet and its delivery to the sink. Fig. 2.12 details the comparison among the delivery delays as a function of traffic for the adaptive routes and the fixed daytime and nighttime routes. The curves show that both fixed routes lead to very long delays at low traffic; on the contrary, adaptive routes keep the delay within more acceptable levels, and help reduce the rate at which delay increases with traffic.

2.5 Conclusions

In this chapter, we have evaluated the effects of changing environmental conditions on the performance of MAC and routing protocols in underwater acoustic networks. To this end, we employ a modified version of the WOSS simulator where the sound speed profile over the water column is allowed to change in time. Our results show that MAC protocols are in fact impacted by the SSP employed to simulate, even though the performance achieved using measured environmental data and that measured using the sound speed retrieved from the free World Ocean Database 2009 are relatively similar, and in any event show the same trend.

We then showed that it is not convenient to set up static routes in a multihop network, as SSP changes would make such routes suboptimal over time. We argued that even a route update as rare as once every three hours achieves much better performance, especially in terms of delivery delay and of the average length of the routes.

Future work on this topic includes a more detailed study of the optimal route update frequency and the implementation of a route dissemination and update protocol.

Acknowledgment

This work has been supported in part by the European Commission under the 7th Framework Programme (grant agreement no. 258359 – CLAM).

The authors gratefully acknowledge the Nato Undersea Research Centre, La Spezia, Italy, for providing the samples of the sound speed profile from the SubNet'09 dataset.

References

[1] N. Chirdchoo, W.-S. Soh, and K. C. Chua, "ALOHA-based MAC protocols with collision avoidance for underwater acoustic networks," in *Proc. of IEEE INFOCOM*, Anchorage, AK, May 2007.

[2] F. Guerra, P. Casari, and M. Zorzi, "World Ocean Simulation System (WOSS): a simulation tool for underwater networks with realistic propagation modeling," in *Proc. of ACM WUWNet 2009*, Berkeley, CA, Nov. 2009.

[3] S. Basagni, C. Petrioli, R. Petroccia, and M. Stojanovic, "Choosing the packet size in multi-hop underwater networks," in *Proceedings of IEEE OCEANS*, Sydney, Australia, May 2010.

[4] B. Peleato and M. Stojanovic, "Distance aware collision avoidance protocol for ad hoc underwater acoustic sensor networks," *IEEE Commun. Lett.*, vol. 11, no. 12, pp. 1025–1027, Dec. 2007.

[5] A. Syed, W. Ye, and J. Heidemann, "Understanding spatio-temporal uncertainty in medium access with ALOHA protocols," in *Proc. of ACM WUWNet*, Montréal, Canada, Sep. 2007.

[6] J. Yackoski and C. Shen, "UW-FLASHR: Achieving high channel utilization in a time-based acoustic MAC protocol," in *Proc. of ACM WUWNet*, San Francisco, CA, Sep. 2008.

[7] D. Pompili, T. Melodia, and I. Akyildiz, "A CDMA-based medium access control for underwater acoustic sensor networks," *IEEE Trans. Wireless Commun.*, vol. 8, no. 4, pp. 1899–1909, Apr. 2009.

[8] K. B. Kredo II and P. Mohapatra, "A hybrid medium access control protocol for underwater wireless networks," in *Proc. ACM WUWNet*, Montreal, Quebec, Canada, Sep. 2007.

[9] R. Diamant and L. Lampe, "A hybrid spatial reuse MAC protocol for ad-hoc underwater acoustic communication networks," in *Proc. of IEEE ICC*, Cape Town, South Africa, May 2010.

[10] C. Petrioli, R. Petroccia, and M. Stojanovic, "A comparative performance evaluation of MAC protocols for underwater sensor networks." in *Proc. of MTS/IEEE OCEANS*, Québec City, Canada, Sep. 2008.

[11] D. Pompili and I. Akyildiz, "Overview of networking protocols for underwater wireless communications," *IEEE Commun. Mag.*, vol. 47, no. 1, pp. 97–102, Jan. 2009.

[12] B. Tomasi, G. Zappa, K. McCoy, P. Casari, and M. Zorzi, "Experimental study of the space-time properties of acoustic channels for underwater communications," in *Proc. IEEE/OES Oceans*, Sydney, Australia, May 2010.

[13] N. Baldo, F. Maguolo, M. Miozzo, M. Rossi, and M. Zorzi, "NS2-MIRACLE: a modular framework for multi-technology and cross-layer support in network simulator 2," in *Proc. of ACM NSTools*, Nantes, France, Oct. 2007.

[14] M. Porter *et al.*, "Bellhop code." [Online]. Available: http://oalib.hlsresearch.com/Rays/index.html

[15] F. Guerra, P. Casari, and M. Zorzi, "A performance comparison of MAC protocols for underwater networks using a realistic channel simulator," in *Proc. of MTS/IEEE Oceans*, Biloxi, MS, Oct. 2009, to appear.

[16] "World ocean atlas." [Online]. Available: www.nodc.noaa.gov/OC5/WOA05/pr_woa05.html

[17] "General bathymetric chart of the oceans." [Online]. Available: www.gebco.net

[18] "National geophysical data center, seafloor surficial sediment descriptions." [Online]. Available: http://www.ngdc.noaa.gov/mgg/geology/deck41.html

[19] C. L. Bretschneider, "A one dimensional gravity wave spectrum," in *Proc. of the Conference on Ocean Wave Spectra*. Prentice Hall, 1963.

[20] L. Freitag, M. Grund, S. Singh, J. Partan, P. Koski, and K. Ball, "The WHOI Micro-Modem: An Acoustic Communications and Navigation System for Multiple Platforms," http://www.whoi.edu, 2005.

Chapter 3

Error Control Protocols

Contents

3.1 Overview

Error control techniques are of prominent importance to counter reception errors in underwater (UW) acoustic communications [1]. Usually, a Forward Error Correction (FEC) code is natively implemented in the physical layer (PHY) of commercial UW communication devices. Nevertheless, movement- or environment-induced fluctuations of the signal power at the receiver, as well as the effect of multiple access interference in multiuser networks, can

generate transmission errors that exceed the correction capabilities of the PHY-level FEC. In this case, Automatic Repeat reQuest (ARQ) policies are needed to recover packet errors. Such event is quite likely in real scenarios: for this reason, pioneering UW networks such as SeaWeb [2] incorporate some form of ARQ.

Generally speaking, ARQ schemes prescribe that the receiver sends one acknowledgment packet (ACK) back to the transmitter for every data packet correctly received. In case of errors (e.g. as detected via a failed CRC check) the receiver transmits one not-Acknowledged (NACK) packet, or remains silent. This allows the sender to detect the reception error and to take action for retransmitting erroneous packets. The policy most typically employed to administer retransmissions is a form of Stop-and-Wait (S&W) ARQ [3], both for simplicity and because S&W can be straightforwardly employed over half-duplex media such as the UW acoustic channel. S&W ARQ prescribes that the sender transmits a packet and waits for the corresponding ACK/NACK packet, before sending the next data packet. If no ACK is received within a timeout period, or a NACK packet is received, the corresponding data packet is transmitted again. This choice is inherently inefficient for UWANs, as it requires a sender to remain idle for one whole round-trip time (RTT). As the propagation speed of the acoustic waves underwater is low, the throughput achieved by S&W is limited [4]. Therefore, several protocols are designed to perform multiple packet transmissions back-to-back [4, 5], hence achieving a larger throughput via a more intense channel utilization. However, this strategy requires prolonged channel usage, which would be unfair in multiuser networks with random access techniques [6].

If full-duplex communications can be achieved by means of, e.g., time-division or frequency-division duplexing (TDD and FDD, respectively), more effective techniques based on Selective Repeat (SR) ARQ can be employed. With SR-ARQ a whole sequence (or window) of up to M consecutive packets can be transmitted before the sender stops to wait for an ACK; moreover, retransmissions are limited to erroneous packets, at the price of a re-scheduling buffer at the receiver to cope with out-of-order receptions. In any event, we note that such buffer is well within the capabilities of any existing hardware to date.[1]

[1]The Go-Back-N (GBN) ARQ technique [3] was also introduced in terrestrial networks to avoid the acknowledgement of out-of-order receptions, and thus save the storage space required for a re-sequencing buffer at the receiver. GBN is known to be outperformed by SR whenever the re-sequencing buffer can be afforded [3], hence

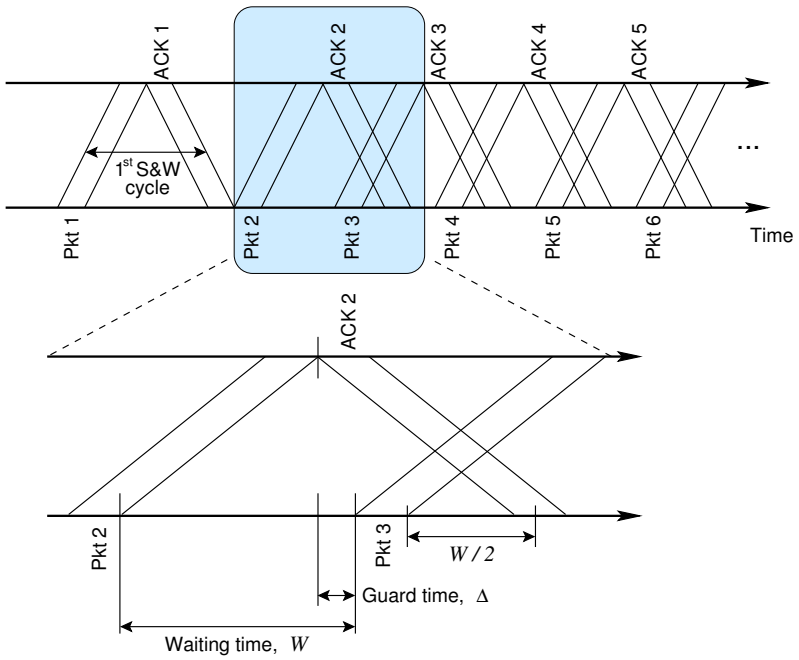


Figure 3.1. *Example of Underwater Selective Repeat (USR) in operation, for a transmit window $M = 2$. Relevant timings are highlighted. (Adapted from [6].)*

To realize ARQ schemes based on SR techniques, a natural approach is to obtain a time-division duplex channel by leveraging on the propagation delays experienced by underwater sound. In typical shallow water network scenarios [7], the distance between neighboring nodes can be of the order of one or more kilometers, which turns into propagation delays of 1 s to 2 s.[2] The packet transmission times obtained with typical modem hardware in many underwater scenarios are shorter than these values [11], therefore employing S&W would turn into very low throughput.

In this work, we jointly address the above issues by proposing and discussing Underwater Selective Repeat (USR), a SR-ARQ scheme which employs a form of TDD, and works well in combination with MAC protocols based on random access. The idea behind USR is that the typically long UW propagation delays should be exploited to pack several data packet transmissions within the same RTT, while keeping the receiver silent when it is expected to receive ACKs. This requires the transmitter to estimate the RTT. Note that, in the

it will not be considered here.

[2]Many modems available to date support transmissions over such distances. For instance, AquaComm supports ranges of 3 km [12], the LinkQuest UWM2000 and UWM3000 modems support 1.5 km and 3 km, respectively [13], and the Evologics S2C R 48/78 modem supports 3.2 km [14].

following, we define the RTT as the delay between the end of a data packet transmission and the beginning of the reception of the corresponding ACK, with reference to the case where both the data and the ACK transmissions are successful.[3] An estimate of the RTT can be obtained by starting USR in a S&W mode, and by measuring the RTT from the timing of the data/ACK exchange. If the RTT is sufficiently large, any further packets for the same receiver can be sent sequentially: however, the timing of subsequent data transmissions must be adjusted so that the transmitter is never deaf to incoming ACKs. Note that the transmission of data packets and the reception of ACKs are interlaced in time, thereby avoiding the need to split data communications and feedback over separate channels.

While the RTT can be easily estimated with a preliminary S&W cycle as detailed above, other design choices are required. For example, it has to be determined how many packets should fit within one RTT. Intuitively, there is a tradeoff between a higher point-to-point throughput if transmissions are packed tightly, and a better capability to avoid multiple-access interference when transmissions are separated by longer waiting times. This tradeoff will be detailed in Section 3.2.

The use of some form of TDD to support multiple packet transmission within a single RTT has been also considered in [12–14]. In [12], the authors propose to share the RTT equally between two communicating nodes, so that they may transfer an equal number of packets. However, this technique requires perfect time synchronization between the nodes, something that may be difficult to achieve even in static networks, and hinders the extension of the technique to multiuser networks. Another TDD-like technique, named Juggling-like-Stop-and-Wait (JSW), is proposed in [13]. With JSW, the sender transmits a fixed number of data packets as specified by a pre-calculated window size, and then it waits for the related ACK/NACK. This does not allow the implementation of plain S&W cycles in the presence of low traffic (e.g., when a node has only one packet to transmit). Moreover, if the network is randomly deployed, it is suboptimal to choose the same window size for all nodes, as the RTT is generally different. Finally, the protocol in [14] is designed to reduce the packet delivery delay over a single link, not in multi-user scenarios.

[3] We remark that this definition is slightly different from what is used in some other contexts, where the RTT is defined as the time interval from the beginning of a data packet transmission to the end of the reception of the corresponding ACK.

Unlike the approaches above, the scheme we propose in this work adapts to the distance between any two communicating nodes, hence to the value of the RTT over the respective link. This adaptation does not have the objective to optimize the transfer of data over a single link: instead, it seeks the optimization of the network performance as a whole. Such result is achieved by avoiding to pack the largest possible number of data packet transmissions within the same RTT, and by choosing transmission timings properly, i.e., in a way that exploits the underlying Medium Access Control (MAC) protocol to avoid collisions. Moreover, our proposed protocol does not require any time synchronization, and can be applied to any randomly deployed network (unlike, e.g., [12]) with any number of nodes (unlike, e.g., [13]). An option to make it more robust in the presence of mobility is described in Section 3.3.3.

3.2 Underwater Selective Repeat (USR)

The USR protocol has been designed according to two guidelines: i) the underwater propagation delays are typically long with respect to the packet transmission time, and should be exploited for enabling a SR-ARQ scheme; ii) the scheme should be designed in a way that makes it suitable to multiuser networks.

USR works as follows. Whenever a node has one or more packets for a certain destination, it first sends one packet using a common S&W scheme: namely, it performs the channel access procedures required by the Medium Access Control (MAC) scheme in use, it sends one data packet, and then waits for the corresponding ACK. At the reception of this ACK, the node estimates the RTT between itself and the destination. If this time is too short to allow the transmission of multiple packets within one RTT, while still ensuring that data packet transmissions and the ACK receptions will not collide, the transmitter falls back to S&W. The same action is taken if there is only one packet to send. The knowledge of the RTT allows the transmitter to estimate the length of the transmit window, i.e., the maximum number of packets that can be transmitted before waiting for ACKs.

With reference to Fig. 3.1, call τ the propagation delay, and T_D and T_A the transmission time of a data packet and of an ACK packet, respectively. The window size M can be

computed as

$$M = \max\left(1, \left\lfloor \frac{k\tau}{T_D + T_A + \Delta} \right\rfloor\right) . \tag{3.1}$$

where Δ is a guard time (required to compensate for changes in the RTT during the transmission of data packets and/or for changes in the speed of the nodes in case of mobility), and k is an adaptation factor which limits the portion of the RTT to be considered in the computation of M, $0 < k \leq 2$. In the limit, if $k = 2$, the packets will be transmitted with the minimum possible spacing, whereas if $k = 0$ the window M will be lower-bounded by 1, which corresponds to falling back to S&W. The same happens if the communicating nodes are very close: in this case, the expression $\left\lfloor \frac{k\tau}{T_D+T_A+\Delta} \right\rfloor$ in (5.9) is rounded to 0, but M would be set to 1 as per the definition in (5.9). Note that, in a multiuser network, the RTT between different pairs of nodes may be different, and in general this reflects on different values of the maximum window size M. The estimate of the RTT performed during the first S&W cycle makes USR adapt to these differences.

Whenever $M > 1$, the sender waits for a fixed time W before sending the next packet of the window, in order to avoid receiving an ACK while transmitting a data packet. In the following, we compute W in such a way that that the ACK reception takes place in the middle of the waiting time, i.e., it is centered $W/2$ after the end of the reception of the previous data packet, if the RTT is constant. The waiting time W is defined by the following relationship

$$T_D + \frac{T_A}{2} + 2\tau = MT_D + (M-1)W + \frac{W}{2} , \tag{3.2}$$

where the left-hand side models the time that elapses between the beginning of a data packet transmission and the middle of the reception of the corresponding ACK, whereas the right-hand side stands for the fact that this reception should take place in the middle of the corresponding waiting time. By solving for W we get

$$W = \frac{T_A + 4\tau - 2(M-1)T_D}{2M-1} . \tag{3.3}$$

We remark that W depends on M, which in turn is chosen such that there is always at least one data packet and one ACK transmission within a time interval of duration $k\tau$, $0 < k \leq 2$. Since W is computed so that an ACK is received in the middle of the interval of duration W (in case both the receiver and the transmitter are stationary and the propagation delay is constant), a smaller window M always results in longer waiting times between subsequent

data packet transmissions. In any event, we observe that by imposing that $W \geq T_A$ in (5.10), we get $M - 1 \leq 2\tau/(T_A + T_D)$, which is verified as per the expression in (5.9).

In the following, we consider two versions of USR: in USR-SLiding Window (USR-SLW), a sender keeps sending packets whenever fewer than M packets are still to be acknowledged; in USR-FiXed Window (USR-FXW), the sender transmits M packets and then waits until all the corresponding ACKs have been received; only after that does it perform a further transmission of a window of M packets. We finally note that the different behavior of the two USR versions requires different channel access patterns: in particular, the USR-FXW technique performs the access procedure before transmitting every window of M packets; on the contrary, USR-SLW accesses the channel only before the transmission of the first packet to a given node. For this reason, USR-SLW strikes a different balance between how many transmissions are pushed towards the receiver per unit time and how much interference is generated in the area nearby the transmitter and the receiver.

3.2.1 Remarks on the backoff algorithm

Backoff is commonly seen as part of the MAC scheme in use. However, USR's timings may be broken if backoff events and times are chosen in a way that is completely oblivious of the ARQ scheme. Hence, in this work we prefer to let USR manage backoff events. The event that triggers backoff is the loss of an ACK. By virtue of (3.2), the ACK related to a certain packet is expected 2τ after the packet transmission, and around the middle of a time window of length W. Hence, W also represents the timeout for the ACK reception. If the ACK is missing, the sender refrains from further transmissions using a standard binary exponential backoff scheme. In particular, the window of the backoff time is doubled at every failed ACK reception, and reset at the first successful one. After a backoff timer expires, a node always performs a fresh channel access attempt before transmitting again. In any event, the backoff time is never less than the time required to receive all pending ACKs.[4]

[4]This choice proved to offer the best performance in all simulations, the main reason being that no two nodes seize the channel for themselves for long periods of time.

3.2.2 Accounting for mobility in USR

Normally, the setting of M and W in (5.9) and (5.10), respectively, guarantees that the transmitter is never deaf to ACKs from the receiver. However, if the transmitter and the receiver are mobile, the RTT may vary over time, and possibly lead to the superposition of data transmissions and ACK receptions. This effect can be compensated for by reducing the transmission window M via a different estimation of the propagation delay. In the following, we employ a conservative approach: first, when we compute M in (5.9), we assume that the transmitter and the receiver move toward each other, and therefore that the propagation delay will decrease over time. This has the net effect to yield a lower value of M than in the static case. Second, when computing W in (5.10), we still employ the actual estimate of the RTT yielded by USR's first S&W exchange. This translates into larger values of W, hence larger spacing between subsequent packet transmissions, and ultimately longer guard times for accommodating mobility.

In more detail, assume that node A transmits to node B at time t_1 using USR, and infers the propagation delay τ_{AB} between the two nodes using the timing of USR's first S&W exchange. This estimate of τ_{AB} will be employed in (5.10) to compute W. If A and B are mobile, at a later time t_2, A will compute M by employing the propagation delay estimate $\tau'_{AB} = d'_{AB}/c$, where c is the speed of sound (approximated as a fixed value equal to 1500 m/s for the purposes of this computation), and

$$d'_{AB} = d_{AB} - (t_2 - t_1)v\,. \tag{3.4}$$

In (3.4), v is the maximum relative velocity of the nodes, computed as the sum of the speeds of A and B (B can make A aware of its speed by piggybacking the corresponding value in the ACK of USR's first S&W exchange). Unlike in the static case, here d'_{AB} is a worst-case estimate of the distance between the moving nodes, and hence leads to computing a lower bound of the propagation delay. We remark that the nodes need not know their position or direction of movement.

3.3 Simulation Results

This section presents the performance evaluation of the USR protocol in several scenarios. First, we provide a description of the scenario and of the system parameters in Section 3.3.1; we illustrate the simulation results in static networks in Section 3.3.2 and proceed to the results related to mobile networks in Section 3.3.3; in Section 3.3.4, we show the impact of the parameter k introduced in (5.9). Based on the considerations provided in the latter section, an adaptive version of USR is proposed in Section 3.3.5.

3.3.1 Scenario definition and common parameters

The USR protocol is designed to complement a Medium Access Control (MAC) protocol with error control capabilities. In this work, we stack USR on top of a 1-persistent Carrier-Sense Multiple Access (CSMA) scheme tested in underwater networks in [21]. This instance of CSMA is described as follows: a node that has a packet to transmit senses the channel first. The sensing time is random, and very short with respect both to the transmission time of a data packet and to the maximum RTT in the network. Moreover, the random length of the sensing time makes it possible to avoid the synchronization of channel access attempts performed by different nodes. If the channel is found busy, the node immediately performs a second sensing phase, again of random length, and keeps repeating this until the channel is eventually sensed idle. At this point the node transmits. CSMA's preliminary carrier-sensing period makes it possible to avoid some typical collision events via a less aggressive access behavior (unlike what would happen, e.g., with ALOHA). In addition, CSMA has been found to provide good performance [21] with respect to a scheme based on collision avoidance [16] and to a contention-based scheme relying on wakeup tones [17]: this motivates its adoption for evaluating USR in this work.

In this work, the USR-CSMA pair will be compared against CSMA plus a plain Stop&Wait (S&W) ARQ scheme, and against an ALOHA channel access protocol (whereby a node transmits a packet as soon as it is generated, unless the node is already engaged in another packet transmission), also coupled to S&W ARQ. For all protocols, the maximum number of retransmissions allowed for a packet is limited to 5. If the reception of an ACK fails for any reason, the transmitter resorts to exponential backoff. As for USR, the backoff window

is doubled upon successive failed transmissions, and reset upon the reception of an ACK (see Section 3.2.1).

We carry out the simulations using the ns2-Miracle framework [20] along with the World Ocean Simulation System (WOSS) [21]. For producing realistic acoustic propagation via the Bellhop ray tracing software [22], WOSS retrieves the geographical coordinates of the nodes from ns2-Miracle and queries oceanographic databases for environmental data measured nearby the network deployment area. (In particular, a typical July SSP retrieved from WOD [20] is employed throughout our simulation campaign.) Realistic channel profiles are then obtained via Bellhop [22].

In all simulations, the network area is a square of side 2500 m located in the Mediterranean Sea, near the Corsica Island, France. The upper left corner of this area is set at $(43.0625°N, 9.3095°E)$. All nodes are randomly deployed within the area. The nodes communicate using a Binary Phase Shift Keying (BPSK) modulation technique at a bit rate of 4500 bps at a center frequency of 25 kHz. The system bandwidth is set to 9 kHz and the transmit power has been set to 200 dB re μPa. This configuration makes the network fully connected, which in turn helps put the error control protocols under stress. We note that the probability of error over each link may still vary, due to the different channel realizations obtained from Bellhop.

The data packet and the ACK packet sizes are fixed to $L_D = 125$ Bytes and $L_A = 10$ Bytes, respectively. In every simulation run, each node generates packets according to a Poisson process of normalized rate λ packets per packet transmission time. Every node randomly chooses a destination, and transmits all its packets to that destination throughout the whole simulation period. All nodes are both sources of packets and potential destinations for the packets of other nodes. The simulations have been performed for several values of λ, for several network area sizes, and for different numbers of nodes. The performance of the protocols is evaluated in both static and mobile networks. The simulation results are averaged over a total of 25 simulation runs, which yields sufficient statistical confidence.

3.3.2 Static network

We start by considering the case of a static network with a fixed number of nodes. For clarity, from now on we will refer to the compared protocols with a shorthand name that

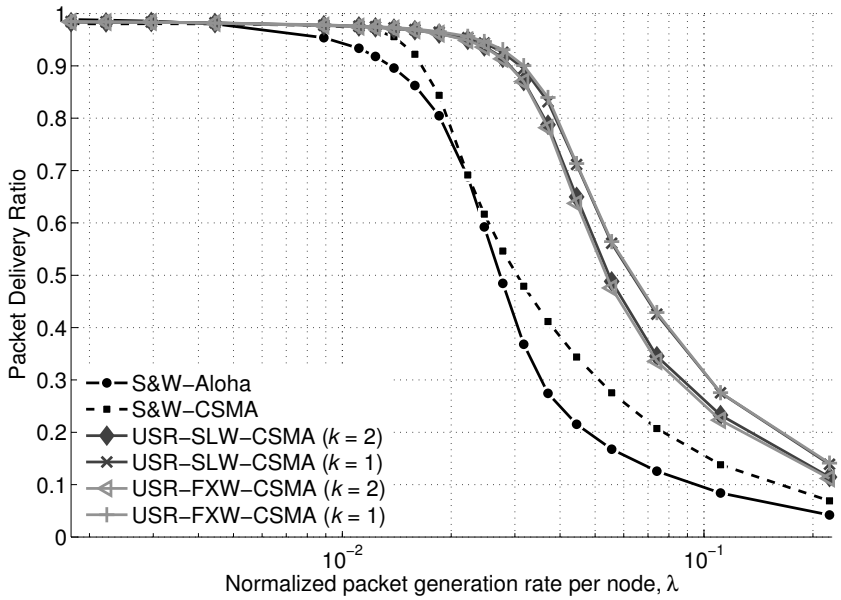


Figure 3.2. *PDR vs. λ for all protocols in a static network of 5 nodes.*

includes both the ARQ scheme in use and the channel access protocol: S&W-ALOHA and S&W-CSMA indicate the use of a S&W ARQ scheme on top of ALOHA and CSMA, respectively, whereas USR-FXW-CSMA and USR-SLW-CSMA denote the use of either version of USR along with the CSMA access protocol.

Figs. 3.2 and 3.3 show the average Packet Delivery Ratio (PDR) (defined as the ratio of the number of packets correctly received by the intended destinations to the number of generated packets[5]) and the normalized throughput (defined as the number of packets correctly delivered in the network per packet transmission time) in a static network of 5 nodes. All protocols perform similarly for small values of λ, corresponding to a light network load and a low probability of collision. As λ increases, S&W-CSMA's channel sensing procedure prevents some collisions, hence the protocol achieves better PDR and throughput than ALOHA. In this scenario, the greatest throughput achieved by both S&W-ALOHA and S&W-CSMA is around 0.08. However, S&W-CSMA's sensing procedure makes it more robust in the face of heavy traffic, hence its throughput curve remains stable at a value close to its maximum, whereas S&W-ALOHA's decreases.

For all values of λ, both versions of USR achieve a better PDR than the other protocol stacks. This is due both to the use of a TDD scheme for duplexing data packets and ACKs,

[5] All packets left in the buffer of a node at the end of a simulation are counted as lost packets.

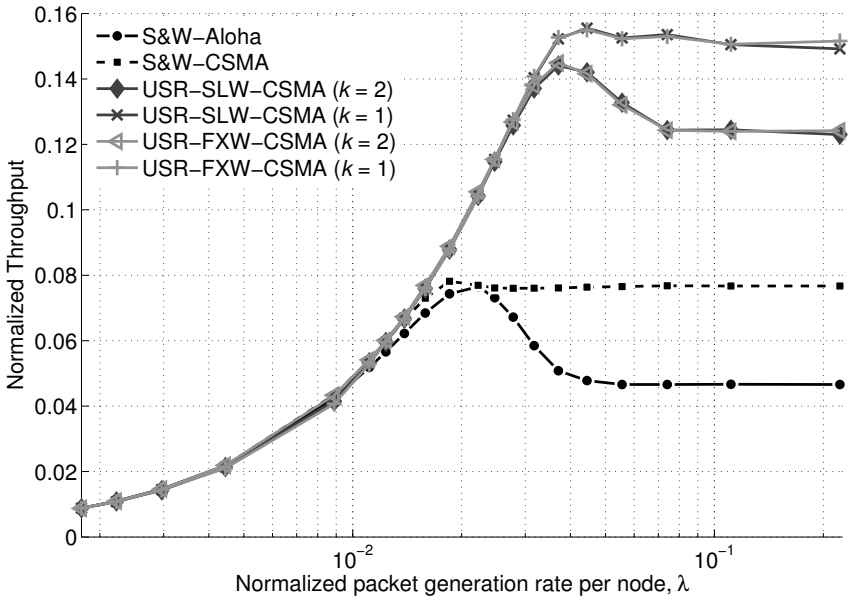


Figure 3.3. *Normalized throughput vs. λ for all protocols in a static network of 5 nodes.*

and to the transmission of multiple packets within one RTT; altogether, these features translate into a higher normalized throughput. In Section 3.3.4 we will discuss the impact of k in more detail: at this time, we simply note that $k = 1$ leads to better performance than $k = 2$. Intuitively, the reason is that a lower value of k leaves longer silence periods between any pair of subsequent packets. In turn, this makes it less likely to experience collisions generated by the hidden terminal effect, or by the fact that transmitters are deaf to packets meant for them. On the contrary, $k = 2$ yields a higher number of transmissions per unit time: this would be optimal for a single link, but translates into greater interference in a multiuser network. In turn, this originates more collisions, hence more backoff events, and results in a loss of throughput due both to the transmission errors and to the silence periods that ensue.

Figs. 3.4 and 3.5 show the PDR and throughput for all protocols in a network of 10 nodes. As λ is the packet generation rate per node, having 10 nodes effectively doubles the network load with respect to the case of Figs. 3.2 and 3.3. The heavier contention that results highlights the difference between S&W-ALOHA (which performs better at low traffic) and S&W-CSMA (which outperforms S&W-ALOHA for $\lambda > 0.022$). The difference between the $k = 1$ and the $k = 2$ cases still remains, and is explained in the same way as in the 5-node scenario: $k = 1$ leaves longer periods of silence between subsequent data transmissions, hence reducing the probability of collisions among different pairs of nodes, with beneficial

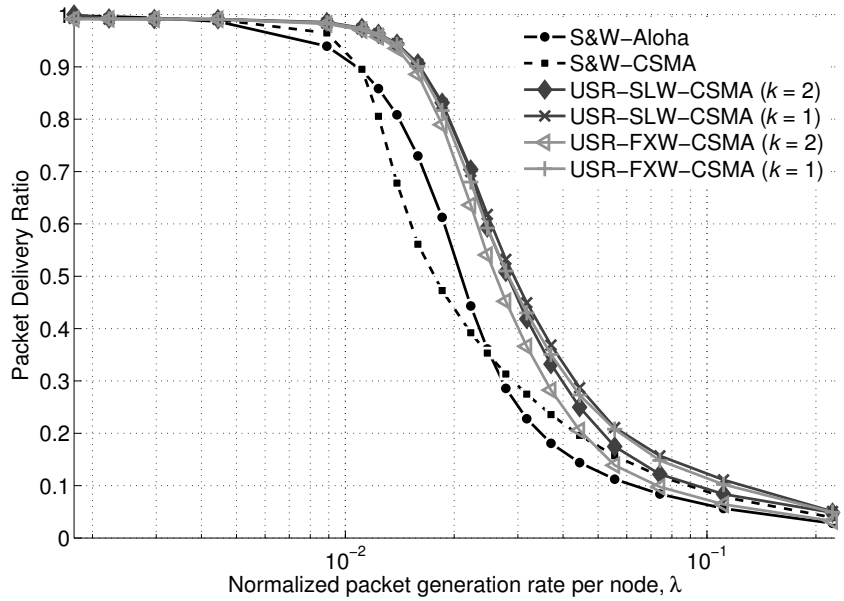


Figure 3.4. *PDR vs. λ for all protocols in a static network of 10 nodes.*

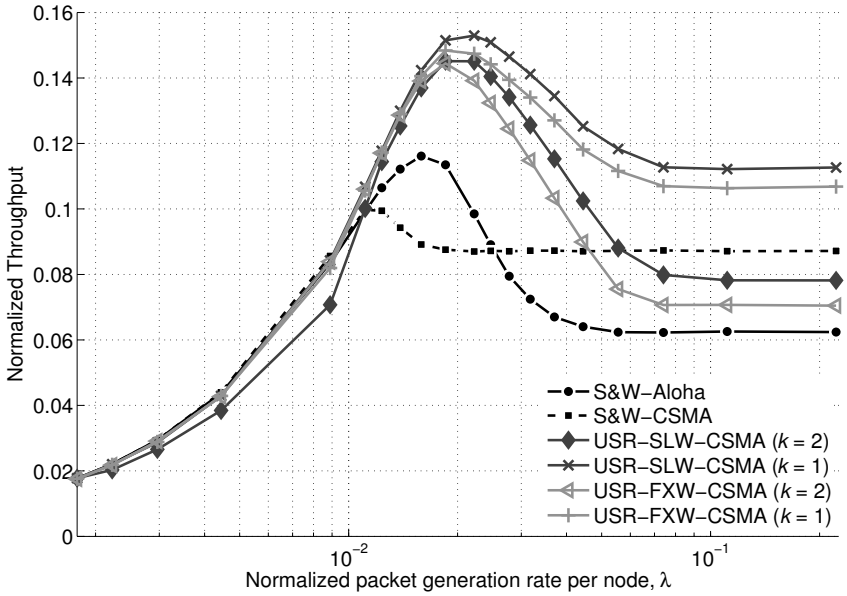


Figure 3.5. *Normalized throughput vs. λ for all protocols in a static network of 10 nodes.*

effects on both PDR and throughput.

We note that both versions of USR perform well: the difference between USR-SLW-CSMA and USR-FXW-CSMA for high values of λ is explained by recalling that the latter sends a train of packets and then waits for all their corresponding ACKs. This forces deterministic silence periods for all transmitters and leads to a lower throughput.

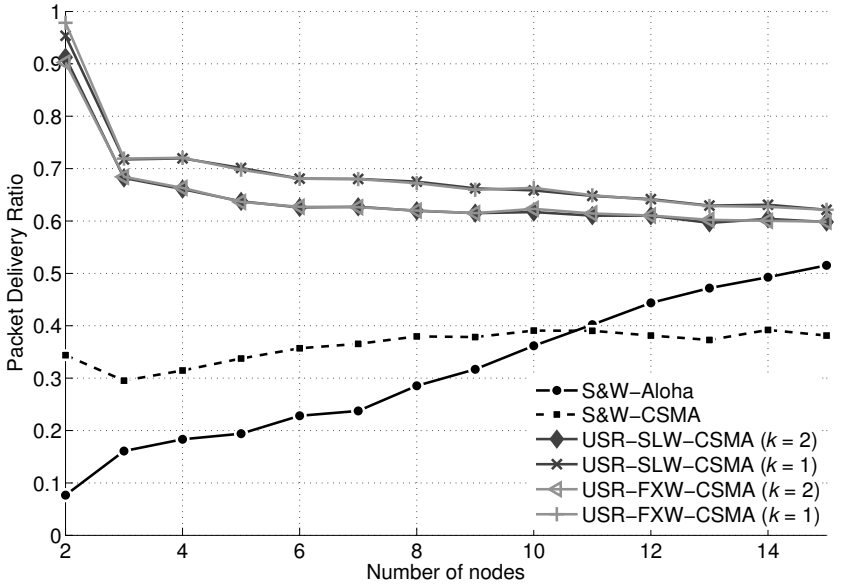


Figure 3.6. *PDR vs. the number of nodes in the network for all protocols,* $\lambda = 0.22$.

It is interesting to note that, for $k = 2$, both versions of USR perform worse than S&W-CSMA in the scenario with 10 nodes for $\lambda > 0.06$. This further supports our observation above, as S&W-CSMA employs a S&W ARQ scheme, which inherently gives rise to silence periods longer than those employed by USR. In other words, the transmission of multiple packets within the same RTT does not always result in good performance in multiuser networks: tuning USR's k parameter as a function of the total network traffic is therefore key to making USR outperform the S&W approaches.

Figs. 3.6 and 3.7 detail the PDR for varying number of nodes and for a varying network area side length, respectively. In both cases, we fix $\lambda = 0.22$. From Fig. 3.6, we observe that, when the network is sparse, the PDR of S&W-ALOHA is lower than that of S&W-CSMA. This is expected, as the nodes do not coordinate their transmissions, increasing deafness and hidden terminal effects. However, when the network is denser, the chance that some transmitter-receiver pairs are very close increases: for these pairs, it hence becomes more likely to capture even interfered packets. The merit of S&W-CSMA, in this case, is that its PDR remains almost constant for every number of nodes.

Both USR-SLW-CSMA and USR-FXW-CSMA outperform S&W-based protocol stacks. The PDR of both USR versions is comparable, whereas we notice the same difference between $k = 1$ and $k = 2$ observed in Figs. 3.2 and 3.4.

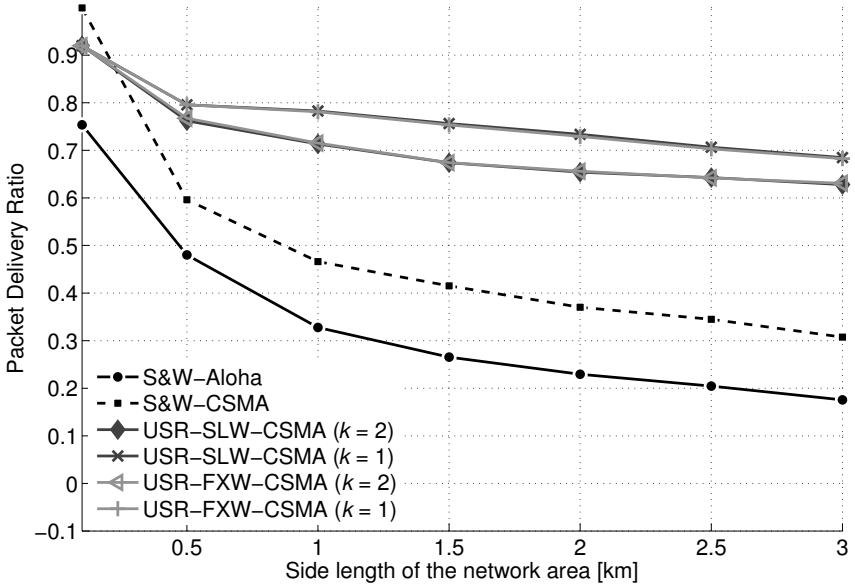


Figure 3.7. *PDR vs. the length of the side of the network area for all protocols,* $\lambda = 0.22$.

It should be noted that USR inherently increases the level of coordination among the nodes: for instance, the channel is not released after USR's first S&W exchange, unlike in S&W-CSMA: on the contrary, the receiver stays silent and waits for further packet transmissions from the sender. This makes the occurrence of deafness events and collisions less likely, and leads to a PDR of nearly 100% in the presence of 2 nodes.

In Fig. 3.7, we consider a network of 5 nodes and increase the length of the side of the network area from 0.1 km to 3 km. When nodes are very close to each other, the channel sensing gives benefit to S&W-CSMA and to both USR versions, which are also coupled with CSMA. However, the average RTT is lower, hence USR rarely has a chance to interlace data packet transmissions and ACK receptions; as a consequence, the performance improvement offered by USR is limited. The opposite occurs for larger areas up to 3 km of side: in this case, the average distance between a sender and its receiver is higher, and the longer propagation delays that result are better exploited by USR than by S&W-ALOHA and S&W-CSMA. As a result, the PDR of the two USR versions is still between 60% and 70%, whereas that of S&W-ALOHA and S&W-CSMA drops to 30% or less.

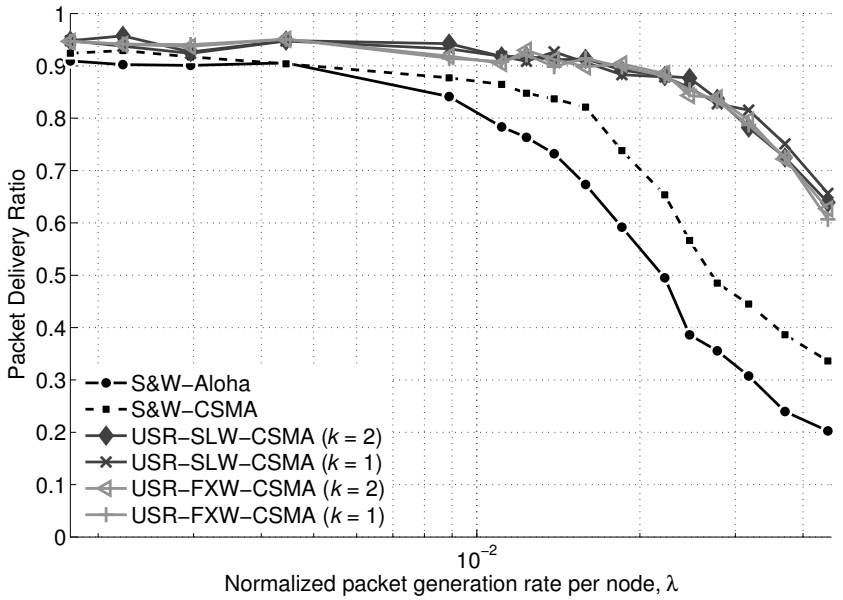


Figure 3.8. *PDR vs. λ for all protocols in a mobile network of 5 nodes.*

3.3.3 Mobile network

We now consider a mobile network of 5 nodes. Each node starts from a random position and depth, and moves within the network area according to a Gauss-Markov mobility model [23] with fixed correlation parameter $\alpha = 0.8$. Note that according to this model, each node periodically chooses a new random velocity vector which will be kept constant for a random amount of time, after which a new vector is generated. Hitting the boundaries of the network area causes the nodes to bounce back. The depth of each node is kept constant throughout each simulation run.

Figs. 3.8 and 3.9 respectively show the PDR and the normalized throughput for all protocols in a mobile network of 5 nodes. We recall that the maximum number of retransmissions in case of errors is limited to 5: this limit is sometimes exceeded, leading to packet drops, and explaining why the delivery ratio of the protocols never reaches 100%. We also note that there is little difference between the PDR and throughput of the different versions of USR. The dominating factor, in this case, is that USR adapts the transmission window M to the RTT measured during its first S&W exchange; since such RTT varies over time with the same statistics for all USR versions, the average number of packets sent per unit time is the same for all versions of USR. In addition, we recall that USR always assumes that the transmitter

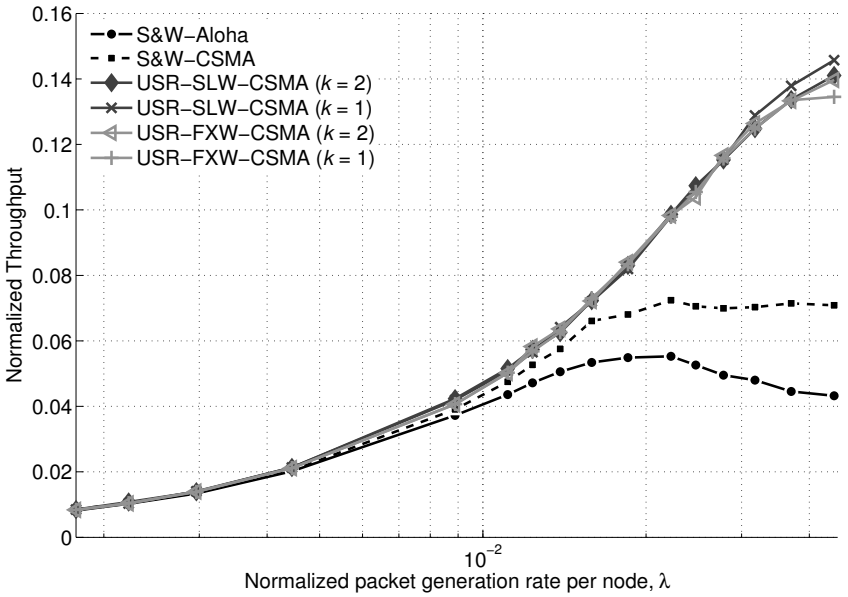


Figure 3.9. *Normalized throughput vs. λ for all protocols in a mobile network of 5 nodes.*

and its receiver move away from each other, hence the ARQ scheme employed eventually falls back to S&W. This happens for both USR-SLW and USR-FXW, and tends to reduce the difference between the protocols even further. In any event, the highest throughput is achieved by USR-SLW-CSMA when $k = 1$, and is twice as high as that of S&W-CSMA.

3.3.4 The impact of k

In the previous subsections, we have observed that k can be used to adapt the behavior of USR: in particular, k tunes the number of transmissions performed within one RTT, and therefore the timings these transmissions are subject to. We have intuitively explained that $k = 2$ is a good setting for the optimization of point-to-point scenarios, whereas any value $k < 2$ reduces the frequency of transmissions in time, and therefore leads to a lower probability of losing packets because of collisions. This suggests that $k < 2$ is a good choice in multiuser networks. However, decreasing k too much would be detrimental in terms of throughput, especially if the RTT between the transmitter and the receiver is very high.

In this section, we explain this tradeoff in more detail, and give some guidelines for the choice of k as a function of the scenario parameters and of the metrics to be optimized. For consistency with Subsection 3.3.5 and because USR-SLW-CSMA and USR-FXW-CSMA lead

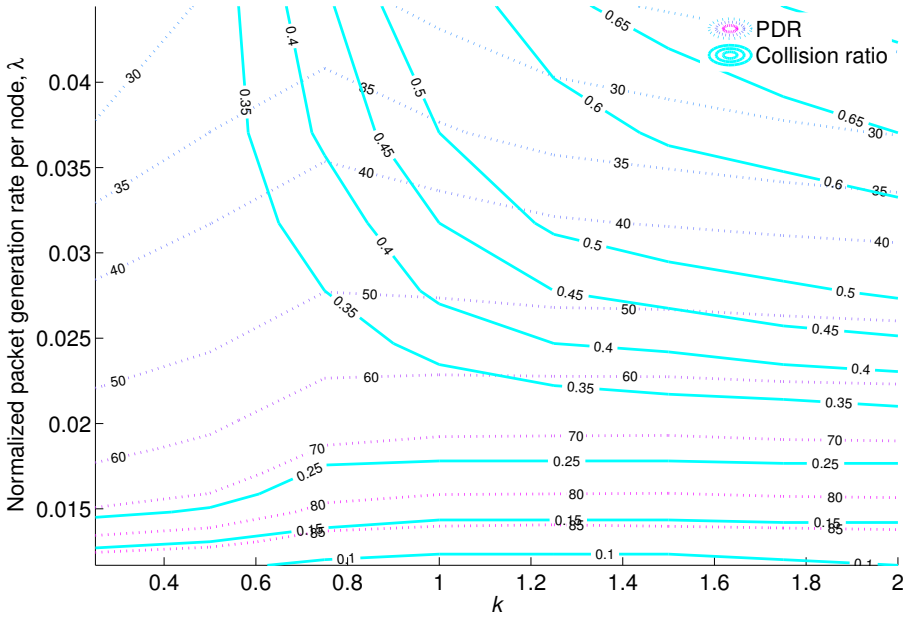


Figure 3.10. *Contour curves of the PDR and of the collision ratio for USR-FXW-CSMA in a static network of 10 nodes. Curves are plotted as a function of k and λ. Each contour curve is obtained as the intersection of the PDR or collision ratio surfaces as a function of k and λ with a horizontal plane corresponding to the PDR or the collision ratio value indicated by the label on each curve.*

to similar PDR and throughput for a fixed value of k, we only consider USR-FXW-CSMA in this subsection. We also focus on the case of a 10-node network: the conclusions for a 5-node network are entirely analogous.

We start with Fig. 3.10, which depicts the PDR and the collision ratio (i.e., the fraction of packets lost due to collisions) as a function of k and λ. The dependence of these metrics on such parameters is shown by means of contour plots, where each curve is obtained as the intersection of the PDR and collision ratio surfaces with a horizontal plane corresponding to the value indicated by the label on the curve. First, we observe that for $\lambda < 0.022$, increasing k improves the PDR by allowing the nodes to resort more often to large transmission windows ($M > 1$ in 5.9); in turn, this reduces the chance that there are still packets in the queue of the nodes at the end of a simulation, and that these packets are counted as lost. Note that this does not increase the number of collisions, as the traffic generated by the nodes is still very low. For higher values of λ, increasing k improves the PDR for the same reasons above, but only roughly until $k < 0.75$, after which the PDR starts decreasing for increasing k. The reason is the higher chance that two transmissions collide: in fact, the collision ratio also

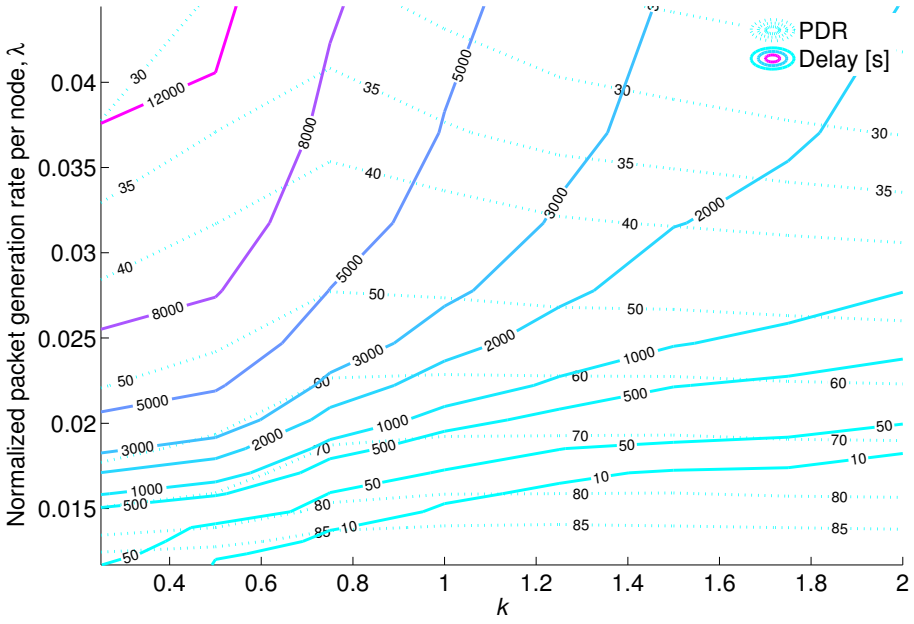


Figure 3.11. *Contour curves of the PDR and of the delivery delay for USR-FXW-CSMA in a static network of 10 nodes. Curves are plotted as a function of k and λ. Each contour curve is obtained as the intersection of the PDR or delivery delay surfaces as a function of k and λ with a horizontal plane corresponding to the PDR or the collision ratio value indicated by the label on each curve.*

increases.

A similar analysis can be applied to the comparison of PDR and delivery delay, defined as the average time required to deliver a packet to its destination node. Such analysis is presented in Fig. 3.11. The figure shows that increasing k always leads to a lower delay, as a consequence of the denser packing of packet transmissions within the same RTT. In turn, if $\lambda > 0.022$, this leads to a lower PDR. For a given value of λ, Fig. 3.11 helps choose the best value of k that achieves the desired PDR and delay, and also highlights which values of these metrics are not achievable. For instance, $k = 1$ achieves a PDR of less than 60% and a delay of about 2000 s for $\lambda = 100$ bps/node. Increasing k helps reduce the delay while maintaining the PDR almost constant. However, it is impossible to achieve, e.g., a PDR of 80% and a delay of less than 10 s for $\lambda = 0.022$. However, if $\lambda \approx 0.015$, such constraints can be jointly achieved for any $k \geq 1.16$.

3.3.5 USR Additive Increase–Multiplicative Decrease (USR-AIMD)

The previous subsection explains that k can be tuned in order to achieve a given set of constraints on such network metrics as the PDR, the collision ratio and the delivery delay. Which value should be chosen depends on, e.g., the amount of traffic generated per unit time in the network. As these conditions may be subject to change over time (for example due to local congestion events), in this section we design an algorithm that adapts the value of k depending on the capability of a transmitter to deliver packets without errors. We name this technique USR-Additive Increase Multiplicative Decrease (USR-AIMD), as it is inspired to the well-known window adaptation mechanism of the Transport Control Protocol (TCP)'s congestion avoidance mode. USR-AIMD is implemented as an extension of USR-FXW (which sends a given number of packets within one RTT and then waits for all ACKs to be received). This way, the updates of the window length occur only after both the transmission of a window of packets and the reception of the corresponding ACKs have been completed (or, in case of errors, after the ACK timeout has fired).

As the name suggests, for each successful transmission, the window size M increases until any packet loss occurs, and a retransmission must be performed. Specifically, focus on the link between a source A and its destination B. We increase the window size as follows

$$M' = \min\left(M_{AB}, M + 1\right) , \tag{3.5}$$

where M_{AB} is the maximum window size computed for A and B and M is the previous window size. If any packet is lost (the corresponding ACK is not received), the window is simply decreased according to the factor $0 \leq \alpha < 1$.

$$M' = \max\left(1, \lfloor \alpha M \rfloor\right) . \tag{3.6}$$

Figs. 3.12 and 3.13 show the throughput achieved by all versions of USR, including USR-AIMD, in a network of 5 nodes and in a network of 10 nodes, respectively. We excluded S&W-ALOHA and S&W-CSMA from this comparison, since they are consistently outperformed by USR in terms of throughput. Note that we omit to mention CSMA in the keys, as all versions of USR build on top of CSMA. In both figures, USR-AIMD shows the same performance as the other USR versions at low traffic, and outperforms them when the traffic is sufficiently high ($\lambda > 0.07$ in the 5-node network and $\lambda > 0.04$ in the 10-node network).

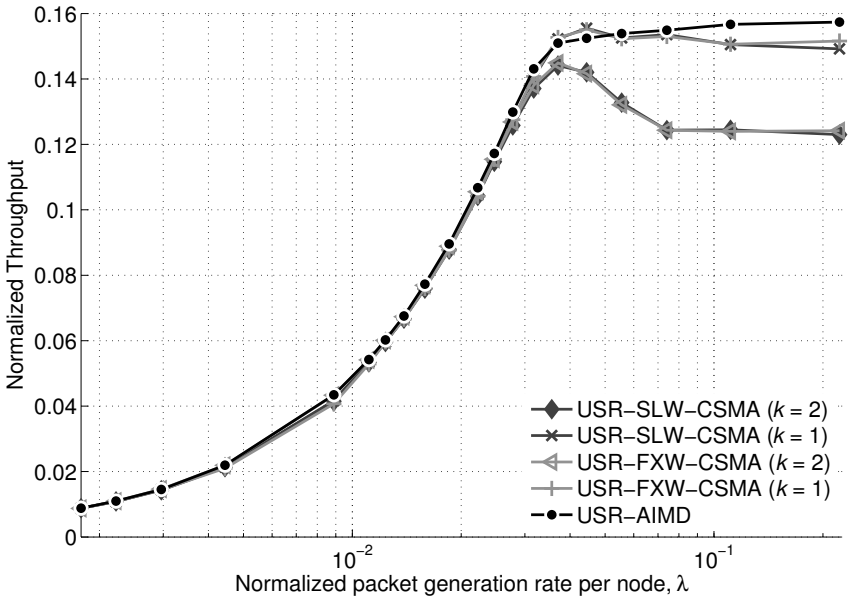


Figure 3.12. *Normalized throughput vs. λ for all versions of USR in a static network of 5 nodes.*

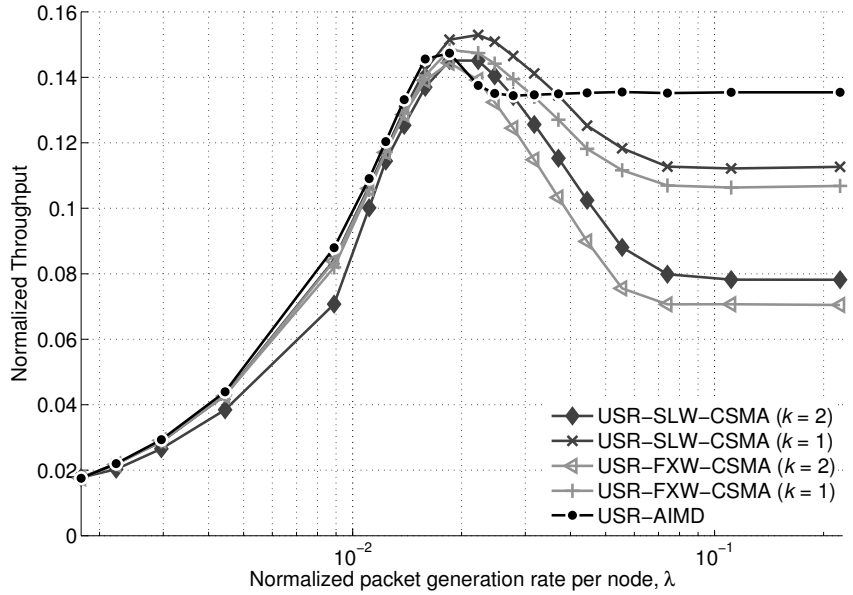


Figure 3.13. *Normalized throughput vs. λ for all versions of USR in a static network of 10 nodes.*

We notice that in the intermediate traffic regime, USR-AIMD is outperformed by the other versions. This is due to the multiplicative decrease of the window length, which makes USR-AIMD refrain from performing too many data transmissions, in case some packets do not get through. The same feature also allows USR-AIMD to keep the throughput stable at high traffic, reaching a larger value than what achieved by the other USR versions.

3.4 Conclusions

In this work, we proposed Underwater Selective Repeat (USR), an Automatic Repeat reQuest (ARQ) scheme for multiuser underwater acoustic networks. The scheme relies on time division in order to set up a duplex channel between the transmitter and the receiver, so that the transmission of data packets can be interlaced with the reception of the corresponding acknowledgements (ACKs). We evaluated the performance of USR by means of simulation in static and mobile networks, and showed that USR outperforms other protocol stacks relying on plain Stop&Wait. We then observed that by limiting the ARQ window length (i.e., the number of data packet transmissions that can be performed before receiving an ACK) via the k parameter, the network achieves better delivery ratio and throughput; in addition, we commented on the relationship between the traffic generation rate, k and such network metrics as the packet delivery ratio and the delivery delay, showing how k should be tuned in order to achieve a given set of constraints on these metrics. We finally designed USR-AIMD, an adaptive version of USR which automatically adapts the window length and achieves a stable throughput at high traffic. We endorse the latter as a good candidate for implementation in real underwater multiuser networks.

Acknowledgment

This work was supported in part by Johns Hopkins University, Applied Physics Laboratorys internal research and development funds, by the European Commission under the Seventh Framework Programme (Grant Agreement 258359 CLAM), and by the Italian Institute of Technology within the Project SEED framework (NAUTILUS project)

References

[1] A. Valera, P. W. Q. Lee, H.-P. Tan, H. Liang, and W. K. G. Seah, "Implementation and evaluation of multihop ARQ for reliable communications in underwater acoustic networks," in *Proc. of IEEE/OES Oceans*, Bremen, Germany, May 2009.

[2] R. K. Creber, J. A. Rice, P. A. Baxley, and C. L. Fletcher, "Performance of undersea acoustic networking using RTS/CTS handshaking and ARQ retransmission," in *Proc. of MTS/IEEE Oceans*, Honolulu, HI , USA, Nov. 2011.

[3] B. A. Forouzan and S. C. Fegan, *Data Communications and Networking*. McGraw-Hill, 2003.

[4] M. Stojanovic, "Optimization of a data link protocol for an underwater acoustic channel," in *Proc. of IEEE Oceans*, Brest, France, Jun. 2005, pp. 68–73.

[5] N. Chirdchoo, W. Soh, and K. C. Chua, "MACA-MN: A MACA-based MAC protocol for underwater acoustic networks with packet train for multiple neighbors," in *Proc. of IEEE VTC Spring*, Singapore, May 2008.

[6] S. Azad, P. Casari, and M. Zorzi, "On ARQ strategies over random access protocols in underwater acoustic networks," in *Proc. of IEEE/OES Oceans*, Santader, Spain, May 2011.

[7] J. G. Proakis, E. M. Sozer, J. A. Rice, and M. Stojanovic, "Shallow water acoustic networks," *IEEE Commun. Mag.*, vol. 39, no. 11, pp. 114–119, Nov. 2001.

[8] "Aquacomm: Underwater wireless modem." [Online]. Available: http://www.dspcomm.com/products_aquacomm.html

[9] "Underwater acoustic modem models." [Online]. Available: http://www.link-quest.com/html/models1.html

[10] "S2C R 48/78 underwater acoustic modem." [Online]. Available: http://www.evologics.de/en/products/acoustics/s2cr_48_78.html

[11] P. Casari and M. Zorzi, "Protocol design issues in underwater acoustic networks," *Elsevier Computer Communications*, vol. 34, pp. 2013–2025, Jun. 2011.

[12] Y. Xiao, Ed., *Underwater Acoustic Sensor Networks*. Auerbach publications, 2008, ch. MAC Protocol Design for Underwater Networks: Challenges and New Directions, by V. Rodoplu and A. A. Gohari.

[13] M. Gao, W.-S. Soh, and M. Tao, "A transmission scheme for continuous ARQ protocols over underwater acoustic channels," in *Proc. of IEEE ICC*, Dresden, Germany, Jun. 2009.

[14] L. Badia, P. Casari, M. Levorato, and M. Zorzi, "Analysis of an automatic repeat request scheme addressing long delay channels," in *Proc. of AINA*, Bradford, UK, May 2009.

[15] F. Guerra, P. Casari, and M. Zorzi, "World Ocean Simulation System (WOSS): a simulation tool for underwater networks with realistic propagation modeling," in *Proc. of ACM WUWNet*, Berkeley, CA, Nov. 2009.

[16] B. Peleato and M. Stojanovic, "Distance aware collision avoidance protocol for ad hoc underwater acoustic sensor networks," *IEEE Commun. Lett.*, vol. 11, no. 12, pp. 1025–1027, Dec. 2007.

[17] A. Syed, W. Ye, and J. Heidemann, "Comparison and evaluation of the T-Lohi MAC for underwater acoustic sensor networks," *IEEE J. Select. Areas Commun.*, vol. 26, pp. 1731–1743, Dec. 2008.

[18] N. Baldo, M. Miozzo, F. Guerra, M. Rossi, and M. Zorzi, "MIRACLE: The Multi-Interface Cross-Layer Extension of ns2," *EURASIP Journal on Wireless Communications and Networking*, Jan. 2010. [Online]. Available: http://www.hindawi.com/journals/wcn/2010/761792/cta/

[19] M. Porter *et al.*, "Bellhop code." [Online]. Available: http://oalib.hlsresearch.com/Rays/index.html

[20] "World ocean database." [Online]. Available: http://www.nodc.noaa.gov/OC5/WOD09/pr_wod09.html

[21] B. Liang and Z. J. Haas, "Predictive distance-based mobility management for PCS networks," in *Proc. of IEEE INFOCOM*, New York, NY, Mar. 1999, pp. 1377–1384.

Chapter 4

Routing Protocols

Contents

4.1 Overview

Multipath routing protocols trade off some network resources to convey data through multiple available routes, and thereby achieves a variety of benefits such as fault tolerance, improved delivery ratio and end-to-end delay. In the specific case of underwater acoustic networks, multipath routing provides a good means of compensating for the typically high packet error rates experienced [1, 2] and for the rapid channel dynamics that are typically difficult to be modeled, predicted and compensated for in advance.

Furthermore, several applications may explicitly require multipath routing in order to achieve robustness in a specific scenario. For instance, in [4], the authors investigated an underwater movement detection network where several sensors are deployed on the sea bottom off a coastal area to be surveiled, and two sinks are placed on the surface on the other side of the network, further offshore. Whenever a sensor detects the movement of a passing ship, it generates data to be delivered to either sink through multihop routes. As the movement of a ship generates acoustic noise in the communications band, the network resorts to multipath routing to increase the robustness of data relaying.

Most multipath routing algorithms can discover link- or node-disjoint routes by means of some local or global knowledge of the network topology graph [4]. However, administering the transmission of packets through these paths is as important as ensuring some type of disjointness. For example, two transmissions over parallel paths located physically near to each other would likely lead to mutual interference. A well designed multipath routing protocol should not only identify multiple alternative paths, but also practically employ a set of paths that avoids mutual interference. This specifically applies to multipath routing in underwater acoustic networks, where the slow propagation speed of sound waves makes interference endure over a longer period of time and over areas of significant size.

Several multipath routing protocols are proposed in the literature [5–7] for terrestrial networks. For instance, Multipath-DSR (M-DSR) [5], extends the Dynamic Source Routing (DSR) [8] protocol by selecting multiple routes by employing multipath link-disjoint algorithm. Since in this protocol only link disjointness is the selection parameter of the alternative links, therefore, there exists a higher number of parallel paths which are located close to each other and hence, experience higher interferences. Moreover, in MDSR, dropping of duplicate RREQs at intermediate nodes hamper the discovery of disjoint paths. Split Multipath Routing (SMR) [6] counters this problem by introducing a different route discovery mechanism which requires more control packets. For underwater networks, such a high control overhead would not be feasible. Multipath AODV (MAODV) [7] is another distance vector based multipath routing protocol which utilizes link-disjoint algorithm to discover multiple alternative paths. Alike MDSR, MAODV also suffers from higher interferences due to the transmissions trough multiple link disjoint paths in underwater networks. A different approach is taken in the Graph-based Multipath Routing (GMR) [9] protocol. GMR

includes graph information in the control packets during route discovery, in order to build a more complete network graph at the destination. A local graph search algorithm is used to find disjoint paths. GMR will finally select only one path, while retaining the alternatives for use in case of link breakage on the current path. This reduces the delivery delay, as no route rediscovery is necessary. However, it requires a huge control packets to be exchanged to build a complete network graph which again increases with increasing number of nodes in the network. Moreover, alike other source based routing protocol, GMR also includes a considerable amount of extra bytes with the payload to incorporate the full path description in the packet. Moreover, architectural differences between the terrestrial networks and the underwater networks also play a vital rule to be not chosen a protocol which is designed for one architecture to be employed in another.

Therefore, there are a few multipath routing protocols which are proposed for underwater communication. For example, in [10], authors propose a multipath protocol which employs Multipath Power-control Transmission scheme, (MPT), for time-critical applications in underwater sensor networks. In MPT, node-disjoint algorithm is employed to discover multiple paths. When a node has to transmit a packet, it transmits the same packet through multiple paths to the same destination and combined at the destination upon receptions. It also combines multipath schemes with a power control technique at the physical layer. Since, in node-disjoint technique, there is no common node between any alternative links. Therefore, it suffers lower interferences than link-disjoint algorithm, but still significant enough than the neighbor-aware multipath discovery algorithm which we describe in Subsection 4.4.1. Neighbor-aware multipath discovery algorithm is designed to find paths that create limited interference to the other paths connecting the same *source-destination* pair and ensures node-disjointness among these paths. Moreover, the algorithm eliminates unidirectional links in the selected paths. Since we assume no a-priori knowledge of the network topology, the paths are to be discovered on demand: the mechanism to do so is designed in such a way that it reduces the packet overhead in the network, but provides sufficient discovery information to select suitable paths.

Another important difference between our proposed multipath protocols and other multipath protocols is that, they support multiple sinks along with the single sink in the network unlike general-purpose communication networks.

In this chapter, we discuss two multipath routing protocols which are designed specifically for underwater acoustic networks; they are: Multi-Sink Routing Protocol (MSRP) and Multi-path Routing with Limited Cross-Path Interference (L-CORP) protocol. Among them, MSRP is a source based routing protocol which employs graph based technique to discover multiple alternative paths; whereas L-CORP is designed to overcome the limitations of the MSRP protocol. This chapter is organized as follows: in Section 4.2, we state the problem and the objective behind designing both the protocols. After that we describe and evaluate the performance of the MSRP protocol in Section 4.3. Following that in Section 4.4, a details description and the performance evaluation of the L-CORP protocol is enunciated.

4.2 Problem Statement

The development and experimentation of collaborative strategies are turning modern navies into international cooperating forces, where vessels from different nations may collaborate to accomplish a common objective, often in international waters. In such a scenario, communications and situational awareness are of primary importance; this includes the surveillance not only of the sea surface, but also of the submarine environment. To establish and maintain a safe operating area, the use of autonomous sensors on the surface and the seafloor is envisioned [2]. These networks may detect relevant information such as movement via, e.g., magnetic or acoustic sensors; in addition, their acoustic communications equipment makes it possible to transmit such information to data-collecting endpoints in contact with the rest of the fleet.

Acoustic communications, in this case, also obey a practical constraint, i.e., to deploy the network rapidly without elaborated wiring. The nodes will be connected via acoustic links, and build a self-configuring underwater network. For this purpose, a project with name *Robust Acoustic Communication in Underwater Networks* (RACUN) led by Atlas Elektronik, Germany, was started in 2010 in the framework of the *European Defense Agency* (EDA), funded by and in collaboration with the Governments/MoDs of Italy, Germany, Norway, Sweden and The Netherlands. RACUN has the objective to develop and demonstrate the capability to establish a robust ad hoc underwater acoustic network for multiple purposes, using both mobile and stationary nodes. Among the purposes of the network, the project targets gen-

eral support for surface vessel operations via data collection from underwater nodes. This is also the case we consider in the simulations.

In more detail, we focus on an underwater intrusion detection network, deployed at the entrance of a harbor in order to monitor outbound surface boats (also called "intruders"). The presence of the network is unknown to the boats. In order to maintain this status, the nodes forward data to collaborating surface vessels by means of acoustic communications only. Also, no gateway buoys are placed very close to the harbor to act as surface sinks and gateways. This measure avoids that such equipment may be detected and stolen, or tampered with.

A further design objective regards the coverage of acoustic communications, which should allow the network to monitor a sufficiently large area with only a few nodes. In turn, this calls for a multihop configuration, where each hop spans 5 to $10\,\mathrm{km}$, so that a line of nodes can monitor a wide portion of the coast, while at the same time being able to haul the data out to a sea base at a safe distance, several hops away. A basic design guideline stemming from the sonar and empirical noise power spectral density equations in [11,12] suggests that acoustic communications over such distances should be operated in the 4 to $8\,\mathrm{kHz}$ band, in different shipping and wind conditions. The main concern with this band, however, is that it is highly affected by the noise generated by boat propellers, especially by those of speedboats [13, 14], which are the main target of the movement detection network. This noise may disrupt communications. In order to cope with this problem, we propose to exploit the redundancy of multipath routing to increase the probability that detection data is correctly delivered. In other words, multiple routes allow data to escape jamming,[1] in that at least one route is hopefully sufficiently free of the propeller noise generated by the boats. Note that for this detection application it is not necessary that 100% of the detection data reaches the sink: a subset of the generated packets would suffice to reconstruct the movement of the boat to a rougher degree of accuracy: however the number of detections should not be too low, otherwise the received data may be interpreted as a false alarm, and therefore neglected.

[1] the word jamming describes possibly unintentional interference coming from a source other than the network nodes.

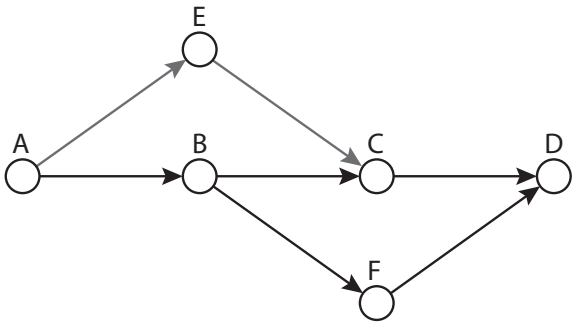


Figure 4.1. *Example of route establishment in MSRP.*

4.3 Multi-sink routing protocol (MSRP)

MSRP is a proactive based routing protocol, i.e., they exchange routing information and find paths before actual network operations, which is feasible for the static surveillance networks. Therefore, the delivery of data packets takes less time. An on-demand route discovery process would take too long on a network with such long distances and propagation delays, and in addition it may also be interfered by the jamming noise.

In MSRP, the route establishment phase employs a similar approach described in [9] for the graph-based multipath routing where it is necessary to construct a topology graph, which in turn requires that each signaling packet stores the identities of the nodes that relay it downstream. Though this approach gives rise to the additional overhead during the route establishment, but allows all bottom nodes to find disjoint paths to the different sinks.

A complete flooding is carried out in the route establishment phase to select multiple suitable alternative paths which covers all the nodes and collect a sufficient number of alternative routes involving potentially any neighbors. This way, there is a greater probability that one of the known routes still works, even in the presence of a node generating jamming noise. Every intermediate nodes wait a few seconds after receiving a control packet before forwarding it to collect additional information about the topology. During this waiting period, the node collects duplicates of the control packets transmitted via other routes. The path information in each packet is merged, and a more complete topology sub-graph is transmitted when forwarding the control packet further. Figure 4.1 shows how an additional waiting period can lead to additional routes. If every node forwarded the control

packet directly, node D would only learn the routes A→B→C→D and A→B→F→D, which are not disjoint. The route A→E→C→D (which is disjoint from A→B→F→D) would not be found, because C would forward the packet from B and therefore drop the packet coming later from E. This problem can be mitigated by waiting before propagating signaling messages, thereby merging route information at each intermediate node.

The route establishment in this protocol is initiated and conducted by the sinks. Each sink broadcasts a control packet every 5 minutes, which includes the sink address, a sequence number and a hop count field. Using a shortened 2-byte network address, the size of this packet is 6 bytes (2 bytes for each field). Every bottom node which receives the message adds the sink to its routing table, and stores the hop distance and the last hop address, which is included in the MAC header. If a node receives the control packet for the first time, it increments the hop distance by one and rebroadcasts it. Duplicates received from other neighbors are not discarded, but stored to have alternative routes available if the links break due, e.g., to temporary interference or node failures. The MSRP employs source routing approach to guarantee that the packets are routed along the chosen paths, i.e., every node includes the full path description with every data packet injected into the network.

Routing entries are declared outdated i) after a period twice as long as the broadcast interval of the message from the sinks, or ii) if a packet with a higher sequence number and the same originating sink and the same last hop address as the current entry is received.

In practice, we consider a network of two sinks and therefore, two paths are selected from the graph, one for each sink, possibly involving disjoint nodes. Whenever a node detects any intruder passing by, it generates detection packets and transmits them through the selected multiple paths to the multiple sinks.

4.3.1 Scenario and system parameters

With reference to Fig. 4.2, we assume that an underwater acoustic network is deployed in the proximity of a harbor to be surveilled. All nodes are bottom-mounted and organized in subsequent lines, or barriers. The first barrier is placed in front of the harbor, and is composed of 5 nodes, in order to obtain good coverage along the coast. The distance between nearest neighbors within a barrier is $3\,\mathrm{km}$. The sensing range is $2\,\mathrm{km}$. Every $8\,\mathrm{km}$ comes another barrier which can sense movement as well as relay data, and has one node less than

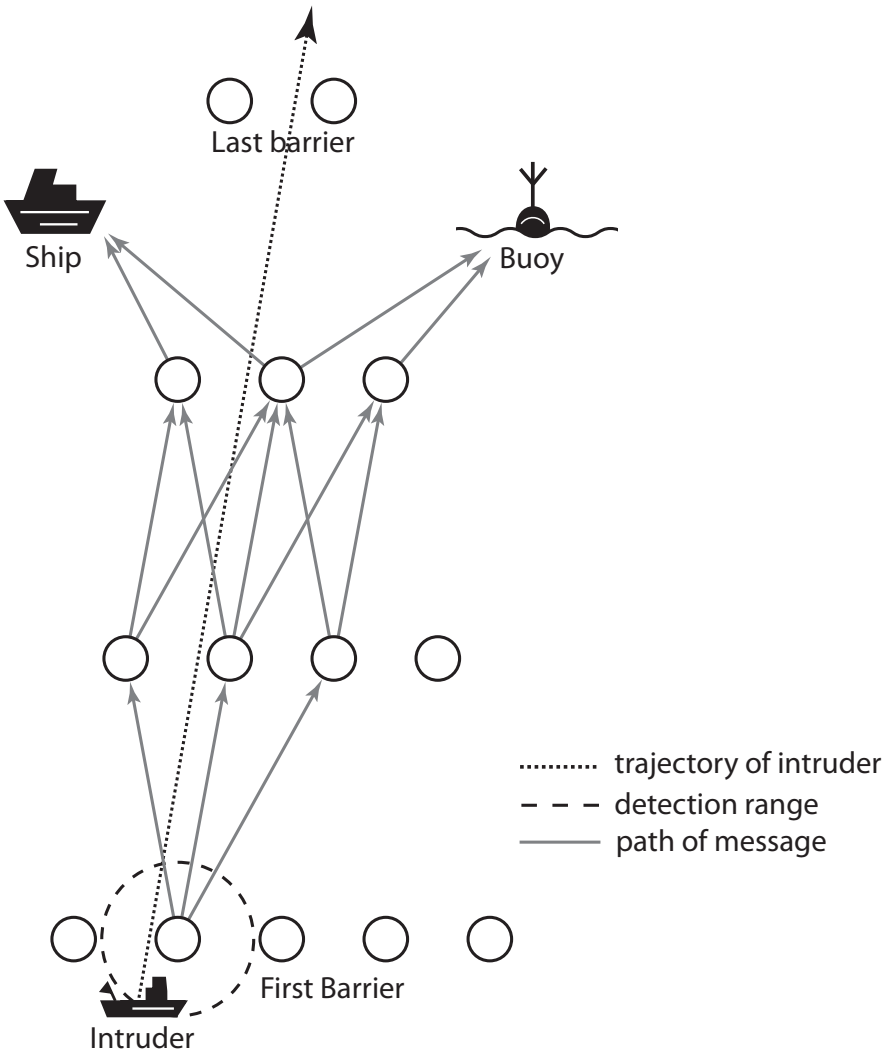


Figure 4.2. *Harbor surveillance network, with 14 bottom nodes organized in 4 parallel barriers. A ship and a buoy act as sea-borne sinks and gateways. Grey links show the forwarding paths obtained using the RF strategy.*

the previous one, so that 2 nodes constitute the fourth and last barrier. Again, this reflects the need to provide finer movement readings near the coast. The network covers a total area of $16\,\mathrm{km} \times 32\,\mathrm{km}$. The intended maximum transmission range of a node in RACUN amounts to about $10\,\mathrm{km}$, hence adjacent barriers are typically in range of each other. We have finally assumed that a ship and a gateway buoy are deployed close to the last barriers to act as sea-borne sinks and gateways towards the sea base. More details about this scenario can be found in [15].

The traffic generation pattern in this scenario is inherently event-based. In a real appli-

cation, the message generation frequency could also be tuned to provide finer knowledge of the node movements: one of our objectives is to evaluate how this affects the network performance. All messages will be relayed to the sink using one of the routing strategies described in the next section. The size of a detection message is set to 16 bytes, according to the Generic Underwater Application Language (GUWAL) [15].

Although the number of nodes is reduced after each barrier, the network features high connectivity, and multiple paths exist between the nodes. This makes the network robust against node failures, as well as against link breakage caused by jamming. In addition, it makes no difference which sink first receives a given data packet, as they are assumed to be connected to each other and to the sea base via other radio (terrestrial or satellite) links.

The evaluation of the network performance has been carried out using the nsMiracle simulator [20]. The scenario reflects the topology of Fig. 4.2. All nodes are deployed at an average depth of about $1000\,\mathrm{m}$, and each has a detection range of about $2\,\mathrm{km}$. There are two static sinks, one on the left side and a second one on the right side of the network. The boat to be detected (or "intruder") leaves the shore and enters the detection range of the nodes in the first barrier. As long as the intruder is within the detection range of a node, that node will generate detection packets, which are to be routed to either sink. These packets are $16\,\mathrm{Bytes}$ long, and are generated at a fixed rate as long as the boat is within the detection range. We test the network performance over typical packet generation rates of interest in RACUN, from 1 to 6 packets per minute. We simulate imperfect detections by applying a 5% chance that a detection fails: in this case, the corresponding packet is not generated. Control packets are $6\,\mathrm{Bytes}$ long, plus an additional $2\,\mathrm{Bytes}$ for every source routing entry in MSRP.

The nodes communicate with the RACUN band, from 4 to $8\,\mathrm{kHz}$. The transmission bit rate is $256\,\mathrm{bps}$, which corresponds to a transmission time of $0.5\,\mathrm{s}$ for a detection packet. The transmit power of each node is $157.3\ \mathrm{dB\ re}\ \mu\mathrm{Pa}$, and has been set so that two subsequent barriers can communicate, but no node can reach two barriers away. The attenuation of the acoustic signals is computed via the link budget model in [11, 12]. The noise in this case is generated both by the environment and by the engines of the intruder, which acts as a de-facto jammer. In order to test the network performance under different jamming power, in our results we also vary such power from 120 to $180\ \mathrm{dB\ re}\ \mu\mathrm{Pa}$.

The simulation results are averaged over several runs, each featuring a different, random

position of the nodes within a circular area of radius $500\,\mathrm{m}$ around their nominal location. The trajectory of the node exiting the harbor is always a straight line, whose starting point and direction are also randomized at each run. The speed of the intruder is fixed to $10\,\mathrm{knots}$. The background noise level, inferred from the empirical equations in [11], is $34\ \mathrm{dB\ re}\ \mu\mathrm{Pa}$.

Before proceeding, we stress that since we focus only on the routing, therefore, we employ a medium access control protocol as simple as ALOHA. This separates the routing performance results from the performance of lower-level protocols. In addition, ALOHA is a feasible choice in large multihop networks as discussed in [17]: this is true also in our scenario, where smaller packet size and large propagation delay to make it unlikely that many collisions take place, and jamming noise is the major source of packet losses.

We evaluate the performance of the MSRP protocol with two other protocols, they are: Singel-Path (SP) and Restricted-Flooding (RF) routing protocol. Brief descriptions of both the protocols are given below:

4.3.1.1 Single-path routing (SP)

In this case, whenever a node wants to transmit data, it searches in its routing table for the sink reachable with the smallest number of hops. Then, the node sends the data packet by specifying a next hop field, so that only the node addressed in such field will receive the packet and forward it further with the same technique. With this technique, the packet is routed back to the closest sink on the shortest reverse path. If a node notices that the next hop is not responding, it can select another neighbor from the routing table. Also, if routing entries become obsolete the node can directly select another routing entry from the list without being forced to wait until the next control packets are broadcast.

The advantage of this strategy is its low overhead, which sums up to only one address field; moreover, no unnecessary duplicates are sent. However, this also makes the protocol more vulnerable to jamming and broken links.

4.3.1.2 Restricted Flooding (RF)

The restricted flooding strategy (or selective flooding) [18] takes advantage of the fact that all nodes know the minimum hop distance to a sink. If a node has data to send, it adds only a time-to-live (TTL) field of the packet, before broadcasting it. The TTL field is set to the

hop distance of the closest sink. All nodes which receive this packet look into their routing tables and forward the packet only if they know a route to that sink with shorter or equal TTL.

This strategy is more robust than single-path routing, but such robustness comes at the price of the additional overhead due to the possibly many duplicates generated. The robustness can be improved (at the price of an increased overhead) by increasing the start value of the TTL field. For example, if the TTL value is increased by one, all nodes in the same barrier will also forward the data packet.

4.3.2 Simulation results

Figs. 4.3 to 4.6, depict the average packet delivery ratio (PDR), the packet overhead, the average delivery delay and the number of hops traveled per packet, respectively. All metrics have been calculated based only on the first copy that reaches either sink: the PDR is the average fraction of generated packets that reach either sink; the delivery delay is the average time elapsed from the generation of the original packet to when the first copy reaches either sink; the number of traveled hops is similarly defined as the average route length incurred by the first packet copy that correctly reaches either sink; finally, the packet overhead is the average ratio of the number of duplicate packets that reach the sinks to the number of generated packets.

In Fig. 4.3 we plot the average packet delivery ratio against the jamming noise power caused by the intruder. The generation rate of detection packets is fixed to $6\,\mathrm{pkt/min}$. As expected, the single-path (SP) protocol fails to deliver 100% of the transmitted packets, even in the presence of a low-power jammer. This is due to errors caused by noise and collisions over the long-haul links in Fig. 4.2. (Recall that we focus on routing approaches, and thus do not consider specific medium access protocols or error control schemes.) Multipath approaches may exploit the redundancy offered by multiple transmissions at the price of the greater overhead, whereas the SP protocol cannot: an erroneous transmission over any link of the only route leading from the source to the sink would result in a lost packet. For low jamming power, both the restricted flooding (RF) algorithm and the multi-sink routing protocol (MSRP) achieve near-100% delivery ratios. On the contrary, for a high jamming MSRP performs worse, as it saves on the number of replicas via a form of source routing, which

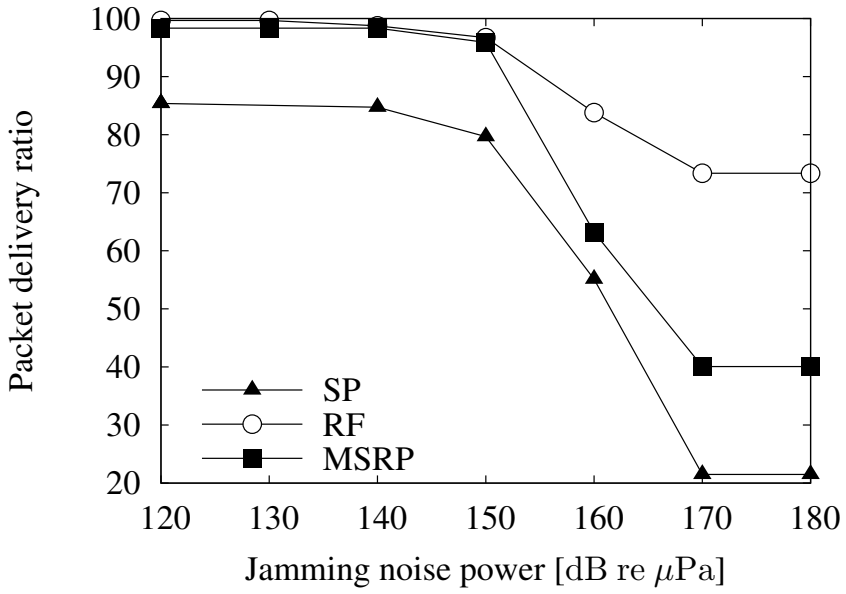


Figure 4.3. *Packet delivery ratio as a function of the jamming noise power for a packet generation rate of* 6 *pkt/min.*

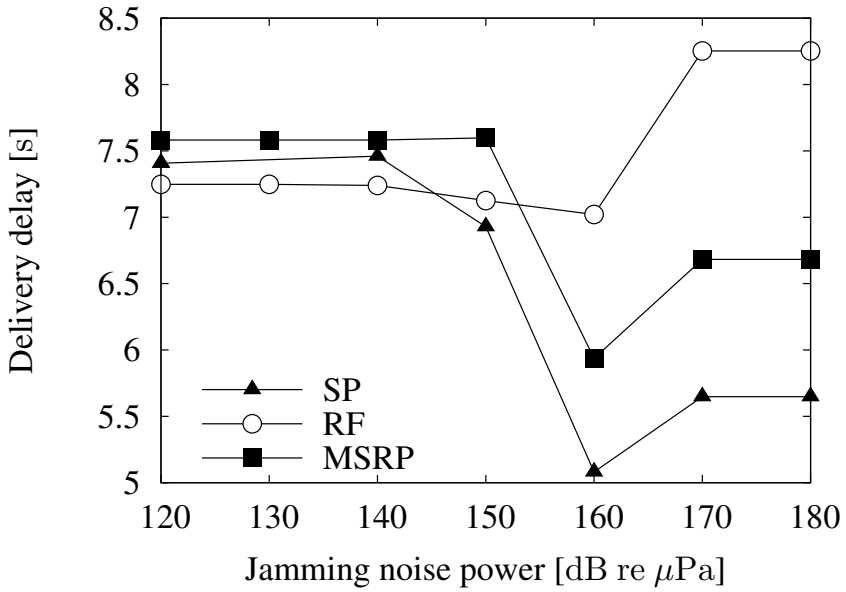


Figure 4.4. *Delivery delay as a function of the jamming noise power for a packet generation rate of* 6 *pkt/min.*

requires updating before a new route can be established. This procedure is also subject to errors as route update packets, albeit very short and frequently sent, may be corrupted by the jamming noise. However, the multipath behavior still gives MSRP an advantage over

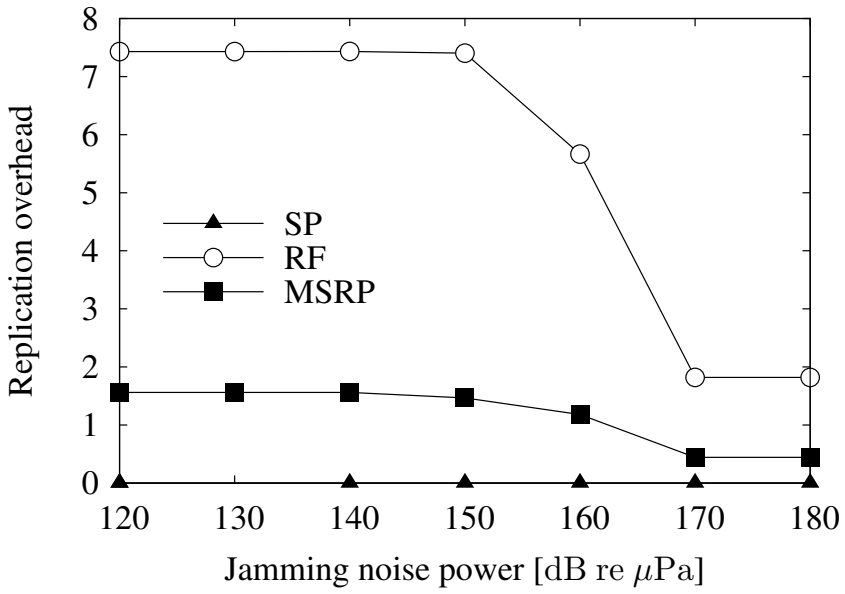


Figure 4.5. *Packet overhead as a function of the jamming noise power for a packet generation rate of* 6 *pkt/min.*

SP. RF works best thanks to local flooding in the direction of the sinks, which gives rise to more packet replicas.

The latter statement is proven by Fig. 4.5, which shows the much larger overhead induced by RF with respect to MSRP. (The overhead of SP is 0.) At high jamming power, around 2 of the many replicas generated by RF survive, whence its higher PDR.

The delivery delay and number of hops traveled by a packet are very similar for all policies (see Figs. 4.4 and 4.6 respectively). We observe that for low jamming power, the delay is about $7.5\,\mathrm{s}$, as the average is taken over both short and longer routes. As the jamming noise increases, the detection packets traveling the longest routes get corrupted and are sometimes lost, whence the decreasing PDR in Fig. 4.3. Conversely, the packets traversing a lower number of hops reach the sink with high probability. As the average delay is computed only over correctly received packets, its value decreases. When the jamming power increases beyond $160\ \mathrm{dB\ re\ }\mu\mathrm{Pa}$, the interference considerably affects even shorter routes, and the delay increases again. The greater number of replicas created by RF (responsible of the better PDR) comes also at the price of the longer routes, which also leads to longer delays. We infer that RF's performance would get worse if the routes got longer: a possible countermeasure would be to increase the number of sinks if the number of nodes increases.

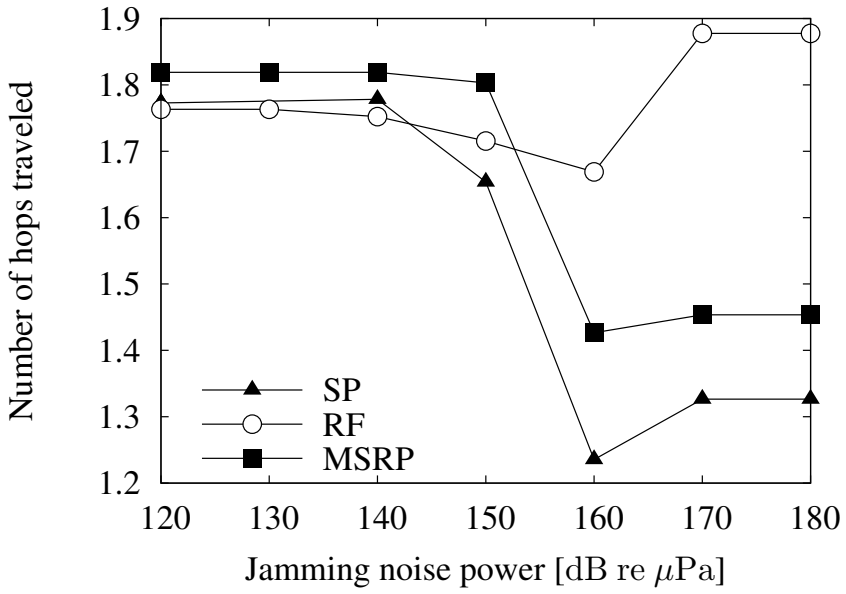


Figure 4.6. *Number of hops traveled per packet as a function of the jamming noise power for a packet generation rate of* 6 *pkt/min.*

As a final note on this first set of results, the average number of hops required to reach a sink is always on the order of 1 to 2: this is a consequence of the deployment of the network in barriers, and of the placement of the sinks on the left and right of the third barrier. In fact, the packets generated by the first barrier require about 3 hops to reach one of the sinks (see also Fig. 4.2), those generated by the second barrier require about 2 hops, whereas the last two barriers make it to the closest sink in one hop.

We conclude our evaluation of the SP, RF and MSRP routing protocols by briefly noting that their packet delivery ratio is almost constant as a function of the number of packets generated by the bottom nodes. This is a consequence of two facts: i) the detection packets are short, which helps keep them separated in time, and iii) the traffic generation pattern is strongly event-based, which limits the amount of generated traffic. The same observations apply also in the absence of the jammer. With respect to this case, the drop observed when the jammer is present is about 15% for RF, and about 30% for SP and MSRP. The other metrics considered in the first part of this section are also quite insensitive to the packet generation rates.

4.4 Multi-path Routing with Limited Cross-Path Interference (L-CROP)

One of the major drawbacks of the MSRP (which is described in Section 4.3) is that every data packet carries along the description of its own path, which causes significant overhead. Moreover, the protocol also assumes bidirectional link, and may become inefficient in the presence of significant link asymmetries. Therefore, we designed another efficient proactive routing protocol which resolves the problems of the MSRP protocol, named Multi-path Routing with Limited Cross-Path Interference (L-CORP) protocol. Moreover, a new *neighbor-aware* multipath discovery algorithm is employed in the L-CORP protocol to discover limited interference paths between the *source-destination* pair and hence, limited interference paths for all the nodes in the network. The proposed multipath discovery algorithm assures node-disjointness as well as link-disjointness among the paths. Moreover, the *request-relay* based route discovery process assists the protocol to eliminate unidirectional links in the selected paths. Furthermore, its rebroadcasting technique is designed in such a way that it reduces the packet overhead in the network, but provides sufficient discovery information to select suitable paths.

Like in AODV and DSR [19], we resort to a source-initiated path discovery procedure started with the source through the transmission of a *Path Discovery* packet. A source that needs to transmit data to a given sink(s) and know no route towards that sink transmits a *Path Discovery* packet and waits for a fixed time period to receive *Path Reply* packets. In order to limit the flooding of the discovery packet, the sender assigns a Time To Live (TTL) field value equal to the maximum expected number of hops the packet should travel to reach a sink.[2] The TTL decreases at each hop; when it reaches zero, the packet is dropped. The sender also incorporates a list of the last nodes: every node rebroadcasting the packet puts its own address. Unlike in [7], which keeps track only of the address of the second-to-last hop[3] traveled by the *Path Discovery* packet to discover link-disjoint alternative paths. However, we can argue that second-to-last hop information not always can assure link

[2]The diameter can be computed by knowing the size of the network deployment area and an estimate of the transmission range of the acoustic modems in use.

[3]The last hop is the current forwarder, and its address is always present in the forwarded replica of the *Path Discovery* packet.

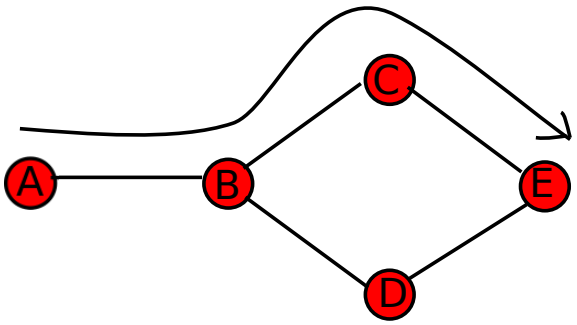


Figure 4.7. *Example of network where keeping track of the second-to-last hop information visited by path discovery packets discovers link-disjoint paths.*

disjointness between the selected links. This fact is exemplified in Fig. 4.7 and Fig. 4.8. In Fig. 4.7, two replicas of the *Path Discovery* reach the destination E, one through route $p_1 = \{A, B, C, E\}$ and one through route $p_2 = \{A, B, D, E\}$. For both, the hop traveled before the last forwarder is B as they reach node E. Therefore E selects p_1 and discards p_2. However, in Fig. 4.8, E also receives two path discovery packets, one through route $p_1 = \{A, B, C, F, E\}$ and one through route $p_2 = \{A, B, D, E\}$, which show a different second-to-last hop. In this case, E wrongly selects both the paths as link-disjoint path since their second-to-last hop is not similar. Therefore, L-CORP keeps track of its full path description. Although this approach increases the size of the packet, but it makes it possible to safely achieve link disjointness between the selected alternative paths.

Whenever an intermediate node receives a *Path Discovery* packet, it does not immediately rebroadcast the packet; instead, it takes the decision whether to rebroadcast using one of the following techniques. One simple technique can be, a node can rebroadcasts only one *Path Discovery* packet from the same *source-destination* pair and same *sequence number*. Through this technique reduces the number of rebroadcasting packets in a considerable margin, but it provides less options to the destination to select a suitable path. In other words, the destination may receive only those packets which have lower *hop count*. However, a path with higher *hop count* with better SNR may provide higher throughput than a path with lower *hop count*. The intermediate hops between the source and the destination do not necessarily forward all *Path Discovery* packets they receive. In another technique, a node maintains a

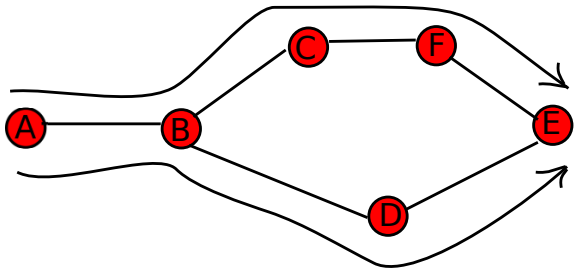


Figure 4.8. *Example of network where keeping track of the second-to-last hop information visited by path discovery packets is not sufficient to discover link-disjoint paths.*

table for the rebroadcasting packets where it stores the information of the *last hops* in the *Path Discovery* packets it rebroadcasts. After receiving a *Path Discovery* packet, an intermediate node checks whether it has any common link with respect to the already rebroadcasted packets of the same *sequence number* like link-disjoint algorithm. Packet is not rebroadcasted and dropped if there exist any common link. This technique does not reduce rebroadcasting packets compare to the previous technique, but it provides sufficient information to the destination to select a suitable path. Therefore, the latter technique is incorporated into our proposed protocol. Before rebroadcasting, the intermediate node employs a *cross-layer* message to find out the SNR of the received signal and add this information in the packet header with other SNRs.

When a *Path Discovery* packet reaches its intended destination, the destination notes down the source, starts a timer, and keeps collecting *path discovery* packets from the same source until the timer expires. At this point, the destination sorts all paths found in order of decreasing average SNR, which is computed via the SNR information contained in the *Path Discovery*. The destination then finds the node-disjoint paths through a standard search algorithm [4] and, for each of them, it sends a *Path Reply* through the reverse of the route stored in the respective *Path Discovery* packets. Note that *Path Reply* packets have the double function to communicate valid paths to the source and at the same time eliminate those paths which include unidirectional links (as the *Path Reply* would not be reliably returned to the source through such paths).

Any intermediate node that receives a *Path Reply* packet, follows the *neighbor-aware* mul-

tipath discovery algorithm before transmitting the *Path Reply* packet, which is described in Subsection 4.4.1 in details. In this way, the paths which have the possibility to interfere with each other are removed from the path list. The intermediate node only puts the information of those paths in the routing table which it transmits. In the routing table it stores the information of the *next hop*, the *source address*, the *destination address*, *hop count* and *expire time* of the path. Here, we store all the paths using both source address and destination address, because, the proposed path discovery technique allows different *source-destination* pairs to select different *next hops*. The source only stores the information of those paths through which it receives *Path Reply* packets.

For discovering neighbor and maintaining the routing table, a node periodically transmits *Hello* packets. A node presumes a neighbor is unavailable if it does not receive any *Hello* packet for that neighbor for a fixed time period which is set equal to three consecutive *Hello* transmits time. After discovering that one of its neighbors is not available, the node announces this information by sending a *Neighbor Unavailable* message. Every node that receives this message eliminates the routes containing this node from its routing table and rebroadcasts the message once for the same *source address* and sequence number and initiates a path discovery process after a random time.

4.4.1 Neighbor-aware Multiple path Discovery Algorithm

Our neighbor-aware multiple path discovery algorithm is basically a refinement over node-disjoint path discovery procedures. It is a simple but effective algorithm to reduce interference while transmitting a data packet simultaneously through multiple alternative paths.

We start by recalling the definition of link- and node-disjointness. Assume that a sender S has data to send to a destination D, and knows two different paths $p_1 = \{x_1, x_2, ..., x_n\}$ and $p_2 = \{y_1, y_2, ..., y_m\}$, where both paths are represented as the ordered sets of the relays x_i, $i = 1, \ldots, n$ and y_j, $j = 1, \ldots, m$ that are traversed to reach node D. Let us indicate a link along either path with an ordered pair (x_i, x_{i+1}), and (y_j, y_{j+1}), respectively. The paths p_1 and p_2 are said to be link-disjoint if $(x_i, x_{i+1}) \neq (y_j, y_{j+1})\ \forall\, i = 1, \ldots, n-1$ and $\forall\, j = 1, \ldots, m-1$. They are said to be node-disjoint if $x_i \neq y_j\ \forall\, i, j$. Node-disjointness implies link-disjointness. Note that even if two paths are node-disjoint, one or more nodes

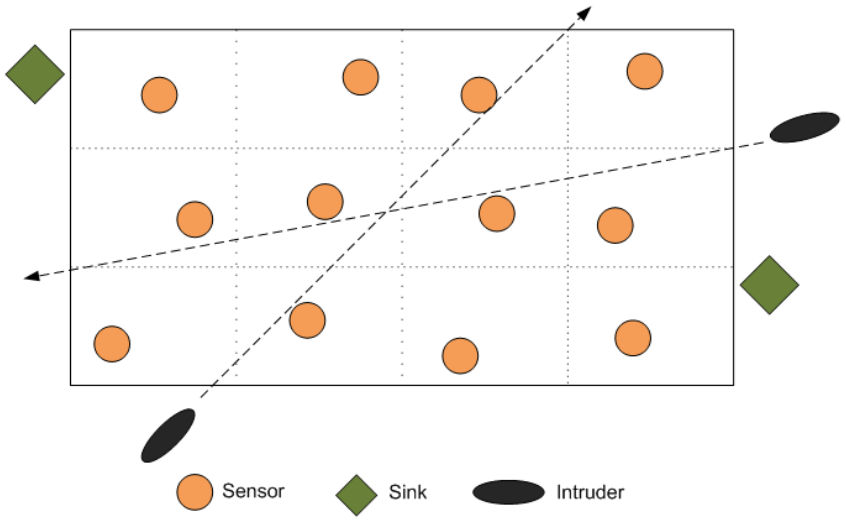


Figure 4.9. *A network with 12 nodes which are randomly distributed within the area, 2 sinks which are also placed randomly outside the area and 2 intruders trespassing the area following a random trajectory*

across either path can still be neighbors, or anyways located sufficiently close to one another. This means that two node-disjoint paths can still cause significant interference to each other. Our neighbor-aware multiple path discovery algorithm tries to mitigate such interference by imposing that no nodes can be neighbors if they belong to two different paths connecting the same source and destination, i.e., $\forall\, x_i \in p_1$ and $\forall\, y_j \in p_2$, the links (x_i, y_j) and (y_j, x_i) must not exist. Generally, a node can choose those paths only if it has the information about the whole networks, which requires huge data exchange among the nodes. Therefore, we take the help from the intermediate nodes and *request-reply* based path discovery process to find out those paths. In this algorithm, after receiving discovery packets, a node selects paths using node-disjoint technique and transmits reply packet(s) to the source. Any intermediate node transmits this packet if and only if none of its neighbors had already transmitted the reply for the same *source-destination* pair and same *sequence number*. A node overhears the transmission of other nodes to learn their reply packet reception/transmission status. It also learns about its neighbors from periodic beacon and irregular control packets.

4.4.2 Simulation Scenarios and Results

We simulate all protocols using the ns2/MIRACLE-based DESERT framework [25], along with the WOSS libraries [21], which enable the use of realistic underwater channel realizations in our simulation software. We consider a network of 12 bottom-mounted nodes deployed in an area of $9\,\mathrm{km} \times 16\,\mathrm{km}$, located near the coordinates $(55.51°\mathrm{N}, 6.14°\mathrm{E})$. The depth of the area is extracted [21] from the GEBCO database [21]. The area is divided into 12 cells; one node is placed at random within each cell. Two sinks are located at opposite sides of the network area, at a random location along one of the short sections. Both sinks are assigned the same anycast address, so that one path discovery process can discover routes towards both sinks.

We consider two use cases for this network. In *Case 1*, the network is run in environmental monitoring mode, and the nodes generate packets according to a Poisson process of given rate. In *Case 2*, the network is run in event mode: two "intruders" cross the network area following a linear trajectory with a random direction, and in doing so they generate noise of given power within the communications band. An intruder is detected by a network node whenever it enters a detection range of $2\,\mathrm{km}$ from that node. For each network node, the packet generation rate is computed as 0.5 packets per minute times the number of intruders located within the node's detection range. The payload of all data packets is 512 bits. A packet is considered delivered if any sink receives it correctly. The nodes communicate in the 4-$8\,\mathrm{kHz}$ band, at a bit rate of $256\,\mathrm{bps}$. The results presented in the following are obtained by averaging over 100 realizations of the network topology. The transmit power is set to $180\,\mathrm{dB}$ re $\mu\mathrm{Pa}$.

We start our performance evaluation from *Case 1*. Figs. 4.10 and 4.11 respectively show the packet delivery ratio (PDR, defined as the number of unique packets delivered to any sink divided by the total number of generated packets), and the packets dropped for interference ratio (PDIR, defined as the number of packets lost due to interference from concurrent transmissions divided by the total number of generated packets). Unlike the PDR, the PDIR nominator includes relayed packets. All multipath protocols described in Section 4.4 are considered. The graphs are drawn as a function of the packet generation rate in packets per minute per node. L-CROP outperforms both MLD and MND with respect to both performance metrics. This is due to its capability to single out a few routes that do not interfere

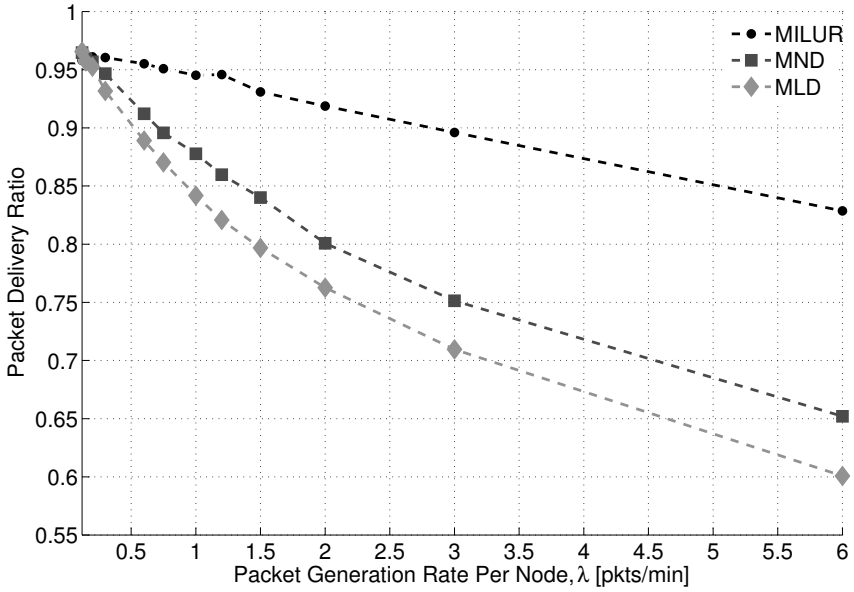


Figure 4.10. *Packet delivery ratio as a function of packet generation rate per node, λ in packets per minute for a grid network with 2 sinks.*

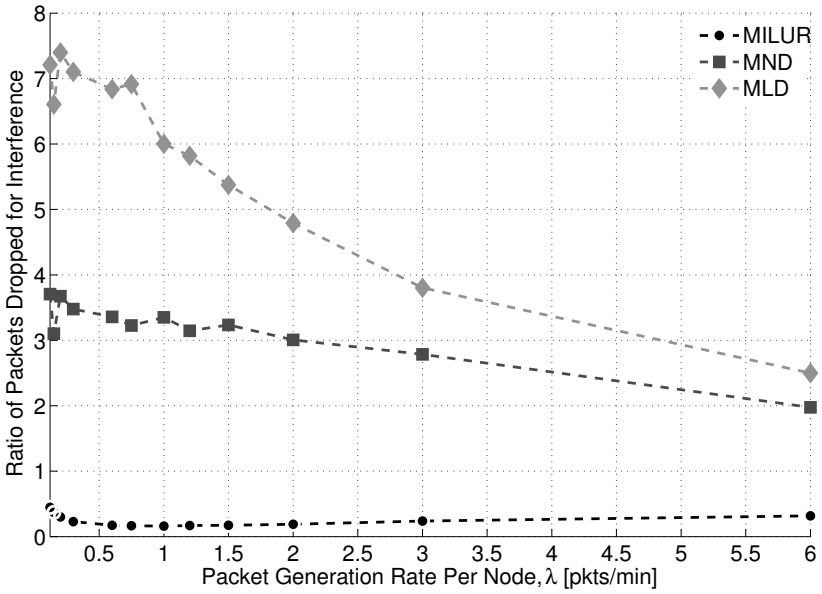


Figure 4.11. *Packets dropped for interference as a function of various packet generation rate per node, λ in packets per minute for a grid network with 2 sinks.*

too much with one another. In fact, the protocols rank depending on the number of routes obtained during the path discovery process: MLD discovers the largest number of routes, which would potentially offer the highest reliability, as every data packet is always sent

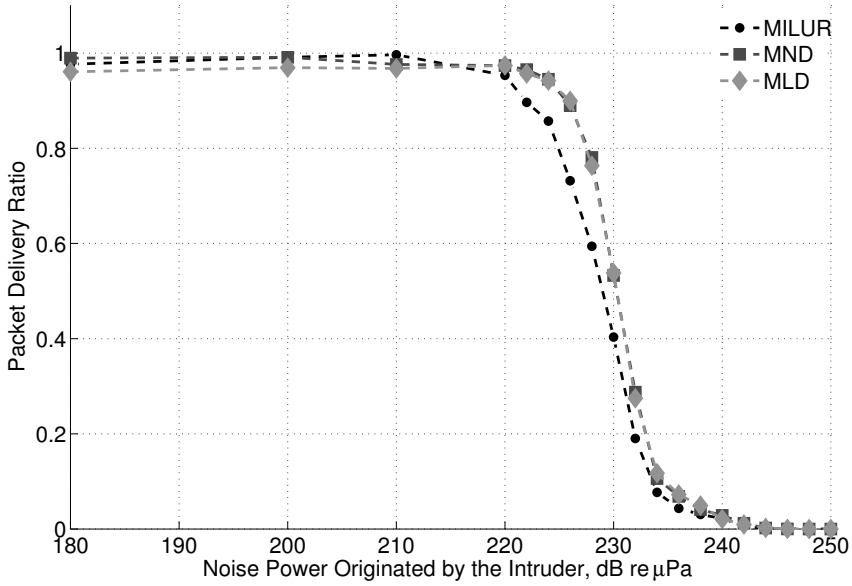


Figure 4.12. *Packet delivery ratio as a function of the noise power of intruders for a grid scenario with 2 sinks and 2 intruders trespassing the area.*

through all known routes. However, these transmissions create significant interference to one another (Fig. 4.11), and the overall effect is a low PDR (Fig. 4.10). Hence, MND (which discovers fewer routes) outperforms MLD, and L-CROP, which poses further constraints on cross-path interference, outperforms both. Note that the PDIR decreases for increasing packet generation rate, because deafness to incoming transmissions (e.g., because the desired receiver is also transmitting) becomes a significant reasons for packet loss as well.

We now turn to *Case 2*. Figs. 4.12 and 4.13 respectively show the PDR and PDIR of the three multipath routing algorithms. Unlike in *Case 1*, the performance metrics are plotted as a function of the power of the noise originated by one intruder within the communications band (we recall that the packet generation rate is not controllable in *Case 2*). When such noise power is sufficiently low, the PDR of all protocols is around 1: this is due to the event-based packet generation, which makes transmissions more erratic: hence, each transmission is typically subject to negligible interference. For the same reason, the PDR achieved by L-CROP is in line with that of MLD and MND. In the presence of significant intruder noise, around 240 dB re μPa, the latter two protocols perform slightly better, as they allow multi-path transmissions over more routes than L-CROP. Nevertheless, the cross-path interference among these routes is quite high, as confirmed by Fig. 4.13, where L-CROP is again shown

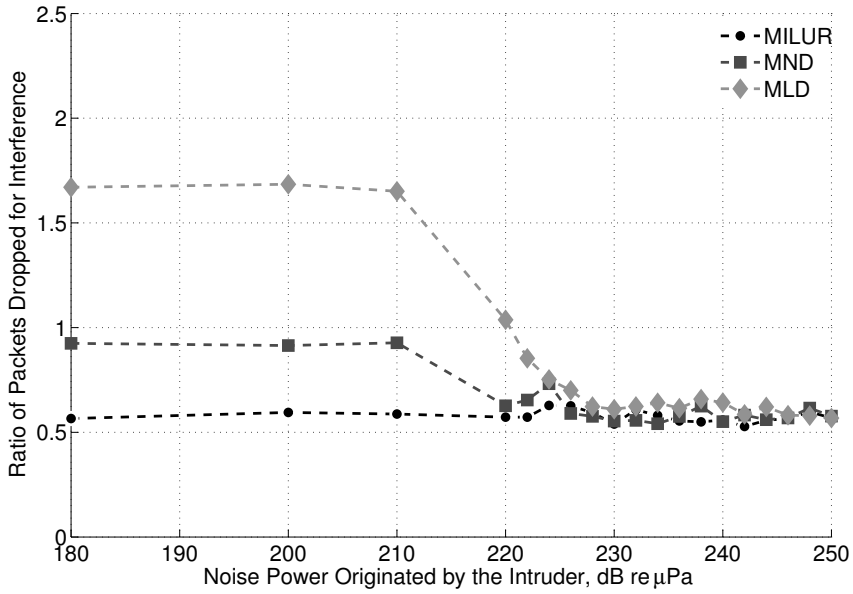


Figure 4.13. *Packet dropped for interference as a function of the noise power of intruders for a grid scenario with 2 sinks and 2 intruders trespassing the area.*

to lead to very few interference-induced packet losses for comparable PDR.

4.5 Conclusions

In this chapter, we discussed two multipath routing protocols which are specifically designed for underwater communication. Among them, MSRP is a source routing protocol which discovers multiple alternative paths using the graph based technique. We compared an MSRP protocol with single-path (SP) and multipath routing protocols for underwater acoustic networks in terms of resilience against in-band jamming noise. The two multipath protocols achieve jamming resistance via restricted flooding (RF) or via an adaptive source routing (MSRP). Overall, we concluded that the best protocol in terms of PDR is RF, whereas SP is the worst. However, the absence of multipath routing overhead in SP can make it a good candidate whenever the power of jamming noise is known to be significantly lower than the power of received signals. MSRP is a protocol with intermediate performance, and represents a good tradeoff between the requirements of high PDR and limited overhead, even though its PDR may suffer in the presence of very high jamming noise.

To overcome the limitations of the MSRP, we proposed L-CORP protocol which ex-

periences less interference while transmitting packet through multiple alternative paths. L-CORP employs a new *neighbor-aware* multipath discovery algorithm which assists it to avoid interference because of concurrent transmission/reception from neighbors. In addition, *reques-reply* based route discovery process of L-CORP helps to avoid unidirectional links and hence, avoid dropping packets due to link absence in a path. To avoid the huge rebroadcasting of *path discovery* packet, it employs a link-disjoint algorithm which restrict a node to rebroadcast a *path discovery* packet only if it receives through link-disjoint path. All the *path discovery* packets receive at the destination are prioritized using average SNR metrics and selected accordingly, which assists the protocol to select a higher throughput path.

The performance of the proposed L-CORP protocol is compared with other two multipath protocols, MLD and MND. The simulation results demonstrate that L-CORP outperforms other protocols in terms of packets dropped due to interference ratio and also exhibits higher or at least comparable performance in terms of packet delivery ratio.

Acknowledgment

This work has been partially supported by the Italian Institute of Technology under the project SEED framework (NAUTILUS project) and "Robust Acoustic Communications in Underwater Networks" (RACUN) project under the EDA Project Arrangement No. B 0386 ESM1 GC. The partners of the RACUN's are: Atlas Elektronik (Germany), WTD71-FWG (Germany), L-3 Communications ELAC Nautik (Germany), TNO (The Netherlands), Köngsberg Maritime (Norway), FFI (Norway), FOI (Sweden), SUS (Sweden), WASS (Italy) and CETENA (Italy).

References

[1] M. Chitre, S. Shahabudeen, and M. Stojanovic, "Underwater acoustic communications and networking: Recent advances and future challenges," *Marine Tech. Soc. Journal*, vol. 42, no. 1, pp. 103–116, spring 2008.

[2] I. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: research challenges," *Elsevier's Ad Hoc Networks*, vol. 3, no. 3, 2005.

[3] M. Goetz, S. Azad, P. Casari, I. Nissen, and M. Zorzi, "Jamming-resistant multi-path routing for reliable intruder detection in underwater networks," in *Proc. of ACM WUWNet*, Seattle, Washington, USA, Dec. 2011.

[4] Y. Yang, "Routing permutaions with link-disjoint and node-disjoint paths in a class of self-routable networks," in *Proc. of International Conference on Parallel Processing*, Dec. 2002.

[5] A. Nasipuri, R. Castaneda, and S. R. Das, "Performance of multipath routing for on-demand protocols in mobile ad hoc networks," *ACM/Kluwer Mobile Networks and Applications*, vol. 6, no. 4, pp. 339–349, 2001.

[6] S. J. Lee and M. Gerla, "Split multipath routing with maximally disjoint paths in ad hoc networks," in *Proc. of ICC*, Jun. 2001.

[7] M. K. Marina and S. R. Das, "Ad hoc on-demand multipath distance vector routing," *WIRELESS COMMUNICATIONS AND MOBILE COMPUTING*, vol. 6, pp. 969–988, 2006.

[8] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," *Mobile Computing*, pp. 153–181, Feb. 1996.

[9] G. Koh, D. Oh, and H. Woo, "A graph-based approach to compute multiple paths in mobile ad hoc networks," in *Proc. of HSI*, Jun. 2003.

[10] Z. Zhou and J.-H. Cui, "Energy Efficient Multi-Path Communication for Time-Critical Applications in Underwater Sensor Networks," in *Proc. of MobiHoc*, Hong Kong SAR, China, May 2008.

[11] R. Urick, *Principles of Underwater Sound*. New York: McGraw-Hill, 1983.

[12] M. Stojanovic, "On the relationship between capacity and distance in an underwater acoustic communication channel," *ACM Mobile Computing and Communication Review*, vol. 11, pp. 34–43, Oct. 2007.

[13] B. Kipple and C. Gabriele, "Underwater noise from skiffs to ships," in *Proc. of Glacier Bay Science Symposium*, Oct. 2004.

[14] S. Amoser, L. E. Wysocki, and F. Ladichc, "Noise emission during the first powerboat race in an alpine lake and potential impact on fish communities," *J. Acoust. Soc. Am.*, vol. 116, no. 6, pp. 3789–3797, Dec. 2004.

[15] M. Goetz and I. Nissen, "Generic underwater application language (guwal) – a specification approach," FWG, Kiel, Germany, Tech. Rep. Tech. Rep. WTD71–0161/2010, Jun. 2010.

[16] N. Baldo, F. Maguolo, M. Miozzo, M. Rossi, and M. Zorzi, "NS2-MIRACLE: a modular framework for multi-technology and cross-layer support in network simulator 2," in *Proc. of ACM NSTools*, Nantes, France, Oct. 2007.

[17] M. Zorzi, P. Casari, N. Baldo, and A. F. Harris III, "Energy-efficient routing schemes for underwater acoustic networks," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 9, pp. 1754–1766, Dec. 2008.

[18] M. Y. Iu, "Selective flooding in ad hoc networks," http://etd.uwaterloo.ca/etd/my2iu2002.pdf, 2002.

[19] G. Tomar, T. Sharma, D. Bhattacharyya, and T.-H. Kim, "Performance comparision of AODV, DSR and DSDV under various network conditions: A survey," in *Proc. of Ubiquitous Computing and Multimedia Applications (UCMA)*, Daejeon, Korea, Apr. 2011.

[20] R. Masiero, S. Azad, F. Favaro, M. Petrani, G. Toso, F. Guerra, P. Casari, and M. Zorzi, "Desert underwater: an ns–miracle-based framework to design, simulate, emulate and realize test-beds for underwater network protocols," in *Proc. of MTS/IEEE OCEANS*, Yeosu, South Africa, May 2012.

[21] "General bathymetric chart of the oceans." [Online]. Available: www.gebco.net

Chapter 5

Delay-Tolerant Routing Protocols

Contents

5.1 Overview

In a Delay-Tolerant Network (DTN) architecture, the nodes involved in communication experience intermittent connectivity and long and variable propagation delays. This kind of network architecture is essential for several applications in underwater communication. For instance, the coastal patrol and surveillance system described in [1] has the characteristics of intermittent connectivity: where Autonomous Underwater Vehicles (AUVs) patrol an area of interest and inspect the surface ships or the underwater assets passing through the area, and a shore-based control center monitors the behaviors of the AUVs. When an asset enters into the surveilled area, one or more AUVs start following it. Since the area to be patrolled is quite large and the AUVs may remain out of range of the shore center most of the time, this type of network architecture can be considered as a DTN.

In DTNs, those routing protocols [2–5] which establish complete end-to-end routes before data transmission are not suitable due to the lack of connectivity; conversely, *store-and-forward* based routing protocols are suitable, since they store a packet and employ an opportunistic strategy to deliver the packet to the destination. In the latter approach, when a node receives a packet from the upper layer or from another node, it stores the packet until it gets any opportunity to forward the packet to the other node(s) with a hope that the receiving node is the destination or will at least forward the packet to the destination. A common practice which is observed among various *store-and-forward* based routing protocols is that they allow a node to replicate a packet several times in order to maximize the probability that the packet is successfully delivered. This kind of routing protocols can be further classified as *replication-based* routing protocols which achieve a higher delivery ratio by imposing higher replication overhead, and thereby possibly wasting the bandwidth. On the contrary, there are a limited number of routing protocols which do not replicate any packet, instead, they forward the same packet from one node to the other until it reaches its wanted destination. These routing protocols are known as *forwarding-based*: this technique reduces the probability of successful delivery, but injects lower traffic in the network. Usually, from a DTN routing protocol it is expected that it will provide a higher packet delivery ratio (like *replicaiton-based*) with lower packet overhead (like *forwarding-based*). An efficiently designed routing protocol should take these aspects into consideration. In addition, it should also take

into consideration that the underwater channel is vulnerable to errors, requiring some form of error control to be employed while developing an underwater DTN routing protocol.

In this chapter, we discuss two *replication-based* routing protocols for underwater DTNs, named Underwater DTN (UDTN) routing protocol and Underwater DTN with Probabilistic spray (UDTN-Prob) routing protocol. The UDTN routing protocol is designed specifically for coastal patrol and surveillance networks, whereas, the UDTN-Prob is designed for all other applications where the DTN architecture is essential. Both protocols employ an efficient data exchange technique to utilize the infrequent contact among the nodes. They estimate the contract duration among the two nodes that understand to be in contact with each other by exchanging information through control packets, so that both nodes in contact can have a fair share of the contact time to transmit their own data. It also incorporates a modified version of Underwater Selective Repeat (USR) [6] ARQ technique to deal with the erroneous characteristics of the underwater channel which is described in details in Chapter 3.

Though, both the protocols have similarities, but, since they are designed to serve different applications, they have several major differences. For instance, since UDTN is designed specifically for coastal patrol and surveillance network, one of the objectives of this protocol is to deliver the recent data about the assets being followed to the shore-based control center. Therefore, data packets are assigned a priority (the newest data packets generated are given the highest priority), in order to transmit the most important data first. When two nodes setup the contact to exchange data to each other, they exchange the summary of their buffer through the control packets so that other node can transmit packets accordingly. On the other hand, UDTN-Prob routing protocol does not assign any strict priority to the data packets and hence, does not exchange summary of the buffer. Another major difference between the two protocols is the replication strategy they follow. UDTN follows intensive replication strategy, i.e., a node replicates a packet until its lifetime expires. Although this strategy assists UDTN in achieving higher packet delivery ratio, it increases packet overhead. On the contrary, UDTN-Prob only transmits those packets which pass a given criteria and hence, limit number of packets injected into the network. When two nodes make contact, the UDTN-Prob employs a probabilistic approach and a binary spray technique to replicate only those packets which have a higher chance to be delivered to their

destination by the node that receives the replica. Details of the UDTN and the UDTN-Prob routing protocols are described in Section 5.3 and Section 5.4, respectively.

5.2 Related Works

DTN routing protocols can be generally classified into *repliaction-based* and *forwarding-based* protocols, as described in Section 5.1. Since the protocols presented in this chapter are replication based protocols, in this section, we are going to discuss only the existing replication based DTNs routing protocols.

Epidemic [7] routing is a replication based protocol where a node continuously replicates and transmits packets to newly discovered contacts that do not already own any copy of the same packets. Because of the flooding nature of the epidemic routing, it may achieve a very high successful delivery ratio, but it typically always exhausts the network resources and generates a very high amount of overhead.

To limit the replication overhead of flooding based protocols, other routing protocols have been proposed. For instance, in the *spray-and-wait (SAW)* [8] protocol, the replication of each packet is restricted to a fixed number of copies. In the *vanilla* version of the SAW protocol, the originator of a packet only can replicate that packet and all others can deliver the packet to the destination; on the contrary, the *binary* version allows also other nodes to replicate. In the latter approach, a node keeps half of the replicated copies of a packet and sends the other half to the node which is understood to be in contact if it has more than one copy of the packet. When it has a single copy of a packet, it tries to deliver that by itself. In [9], the authors demonstrate that the distribution nature of the binary technique helps it achieving higher successful delivery than that of the *vanilla* version, which inspires us to employ this technique in our proposed protocols.

PROPHET [10] is another protocol which also limits the replication by forwarding the replicated copies only to those neighbors which have a higher probability of contacting the packet's destination in a short time. The main difference between the *PROPHET* and our probabilistic spray approach is the *Delivery Predictability Function (DPF)*. In *PROPHET*, the DPF is computed based on the history of encounters for a mobility model where nodes move in a predictable fashion based on repeating behavioral patterns. In our proposed

technique, the assumption that the movement pattern is predictable is removed and the statics of the encounters are adaptively derived based on the history of previous encounters. In particular, the Cumulative Distribution Function (CDF) of the inter-contact duration is obtained from the inter-contact information extrapolated from long time observation from a mobility model where nodes move randomly, contacts are very infrequent and do not follow any repeating behavioral pattern; this CDF is stored in a table by every node.

Another such protocol that limits the replication is *RAPID* [11], which replicates only those packets that exhibit the highest calculated utility. Every node computes the utility of contacts based on a routing metric, such as average delay, missed deadlines, or maximum delay, to be optimized a specified by the network administrator. When a node gets an opportunity to transfer packets, it replicates or allocates bandwidth resources to a set of packets in its buffer in order to optimize the given routing metric. In our proposed protocol, we employ a different utility function, which is the probable time of meeting the destination directly or through other nodes by a given deadline. At each transfer opportunity, the nodes exchange only those packets which have higher lifetime than the deadline and hence limit the number of replicas injected in the network.

All these protocols described above are designed specifically for a terrestrial network where the signals can be considered instantaneous and the transmission range of a modem is limited to around one hundred meters. On the contrary, underwater networks experience higher propagation delays, because of the lower speed of the acoustic signal and typical underwater modems support ranges up to few kilometers [12–14] which is several times higher than terrestrial networks. The vehicles that travel underwater or on the water surface are slower than the vehicles traveling on land. Therefore, the duration of the contacts among the communicating nodes in terrestrial networks are usually short-lived; on the contrary, such duration is longer in underwater networks. Moreover, underwater networks experience more challenging erroneous channel condition than terrestrial channel.

Therefore, an efficiently designed DTN routing protocol for underwater networks should leverage on the above properties to its own advantage. Like DTN routing protocols for terrestrial networks, most of the proposed underwater DTN routing protocols [15, 16] also do not consider the properties of underwater networks. However, our proposed protocols are designed in such a way that they exploit long contact duration by fairly distributing the

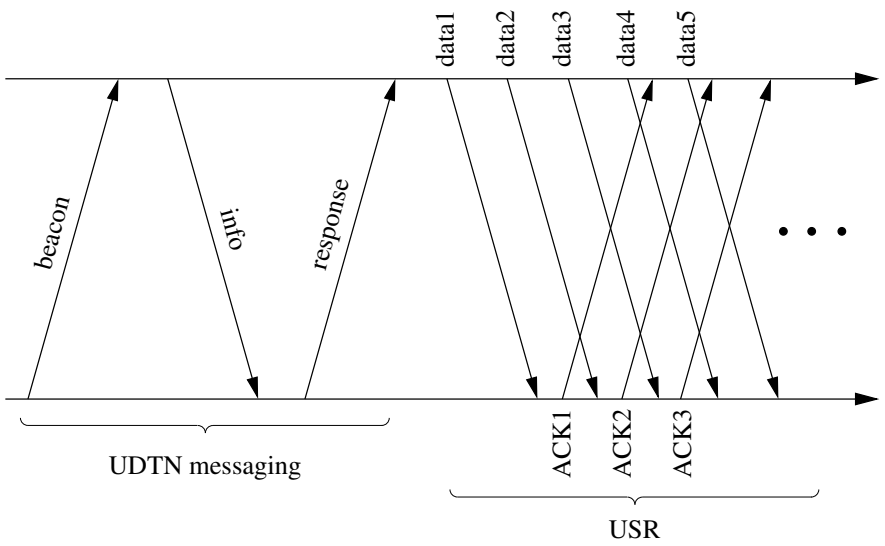


Figure 5.1. *Control and data packet exchange scheme of the proposed UDTN technique with the modified USR protocol.*

estimated contact duration between the communicating nodes. Moreover, modified USR technique assists them to deal with erroneous channel as well as helping exploit the long propagation delay by transmitting multiple packets in a single round trip time. Moreover, priority-based packet exchanges make the UDTN protocol suitable for coastal patrol and surveillance networks. On the other hand, the probabilistic spray and the binary spray techniques which are employed in the UDTN-Prob protocol to reduce replication overhead makes UDTN-Prob suitable for most of the underwater DTN applications.

5.3 Underwater Delay-Tolerant (UDTN) Network routing protocol

To explain both the protocols in more detail, consider a network with 4 nodes and call them A, B, C and D; where D is the destination and all the nodes have data to send to node D. The UDTN routing protocol is described in the next two subsections. Subsection 5.3.1 provides the details of the preliminary signaling phase employed to discover neighbors and organize the transfer of packets upon the occurrence of contacts, whereas Subsection 5.3.2 describes the ARQ technique employed for error control.

5.3.1 Preliminary UDTN messaging for neighbor discovery and contact setup

The messaging scheme of UDTN is illustrated in Fig. 5.1. In UDTN, every node periodically broadcasts beacon messages to discover its neighbors whenever it has data to deliver. Let us consider that node A has data to deliver to node D, therefore, it transmits a *beacon* message and waits for a sufficiently enough time to receive answers from possible neighbors. If no answer is received within the waiting period, A assumes that no node is located within its transmission range, and retransmits a new beacon after a random amount of time.

Let us consider again that node B receives the *beacon* message and it is going to reply with an *info* message. Node B includes its current position and velocity information in the *info* message, which will allow the beaconing node to estimate the relative velocity of the nodes, hence the contact duration. It also includes a summary of the contents of its buffer. The summary is strictly related to the application to be supported (coastal surveillance, in this work) where the network nodes are AUVs moving to track a certain asset. These AUVs store data packets containing samples of the position of the asset being followed. Given that the most recent packets have the highest priority, a good summary of B's buffer is composed of the most recent data packets available about every asset currently being followed in the network area.[1] This is accomplished by putting several (asset ID, timestamp) pairs in the info packet, one for each asset B stores information about. After sending the *info* message, node B waits for the respective *response* message by enabling a waiting time. If no response message is received within the timeline, B goes back to the idle state.

When A receives the *info* message from B, it estimates the contact duration as follows. Call $\vec{P_A}$ and $\vec{P_B}$ are the positions and $\vec{V_A}$ and $\vec{V_B}$ are the speed vectors (whose modulus is in m/s) of nodes A and B, respectively. Therefore, the relative position of A and B is $\vec{P_R} = (\vec{P_A} - \vec{P_B}) + \zeta_P$ and the relative velocity of those nodes is $\vec{V_R} = (\vec{V_A} - \vec{V_B}) + \zeta_V$. ζ_P models the position calculation error and ζ_V models the velocity calculation error. Using $\vec{P_R}$ and $\vec{V_R}$, the instantaneous distance $R(t)$ at time t can be found as

$$R(t) = \sqrt{\|V_R\|^2 t^2 + 2(\vec{P_R} \cdot \vec{V_R})t + \|P_R\|^2} \tag{5.1}$$

[1] We stress that the summary includes information about every asset, not just the asset followed by B. In fact, B may be storing data packets received from other nodes, which hence provide information about other assets. These packets should be also be transmitted to A, in order to improve the chance that the sink receives them.

where $\cdot$ denotes the inner product, and $\| \ \|$ is the 2-norm. The approximate contact duration can be computed as the time required for the nodes to exit the communication range of each other. Therefore, if we replace $R(t)$ with the transmission range T_R of the modem in use, we can compute the contact duration T_c. We define T_R as the range where a probability P_{tgt} of correct packet transmission is achieved.

To compute T_R, we assume that *BPSK* modulation technique is employed which transmits packets of the length L and the transmission power level is set such that it can achieve a target Packet Error Rate (PER), P_{tgt}. We can calculate target PER or P_{tgt} as [17]

$$P_{tgt} = 1 - \left(1 - \frac{1}{2} erfc \sqrt{\frac{SNR}{\psi}}\right)^{L} \tag{5.2}$$

where SNR is the signal-to-noise ratio at the receiver and ψ is SNR margin factor in *dB* at the receiver. From equation (5.2), the SNR_{tgt} can be found as

$$SNR_{tgt} = \left(erfc^{-1}(2 - 2(1 - P_{tgt})^{1/L})\right)^2 \times \psi \tag{5.3}$$

The SNR over an underwater acoustic channel access a distance R for a signal frequency f can be expressed as

$$SNR = \frac{P_{tx} \times (R^{-\varphi} \times a(f)^{-R})}{N} \tag{5.4}$$

where P_{tx} is the transmission power, φ is the spreading factor, $a(f)$ is the thorp absorption coefficient in kHz, R is in *meters* and N is the noise power which can be computed as a function of the frequency in kHz, as described in [18]. The spreading factor φ describes the geometry of the propagation, and $\varphi = 1.75$ provides a practical value for a typical underwater channel [11]. Details of the relationship absorption coefficient and frequency is demonstrated in [19].

Since at the edge of the transmission area, SNR and SNR_{tgt} are equal, i.e., $SNR = SNR_{tgt}$; therefore, by replacing SNR with SNR_{tgt} and solving (5.4), T_R can be found as

$$T_R = \frac{\varphi}{ln(a(f))} \times W\left(\left(\frac{SNR_{tgt} \times N}{P_{tx}}\right)^{-1/k} \times \frac{ln(a(f))}{\varphi}\right) \tag{5.5}$$

Where W (.) is the practical branch of the Lambert-W function which is the inverse relation of the function $x = we^w$ where e^w is the exponential function and w is any complex number.

Now, by replacing instantaneous distance, $R(t)$, of equation (5.1) with a calculated transmission range of a transducer, T_R, we get

$$\|V_R\|^2 T_c^2 + 2(\vec{P}_R \cdot \vec{V}_R)T_c + \|P_R\|^2 = T_R^2 \tag{5.6}$$

Solving Equation (5.6), the approximate contact duration can be found as

$$T_c = \frac{-(\vec{P}_R \cdot \vec{V}_R) + \sqrt{(\vec{P}_R \cdot \vec{V}_R)^2 - \|V_R\|^2(\|P_R\|^2 - T_R^2)}}{\|V_R\|^2} \tag{5.7}$$

After A finishes computing the approximate contact duration, it will prepare a *response* message for B only if the calculated contact duration is higher than a given threshold, ς. In addition, if A receives multiple *info* messages from multiple neighbors, the response message is sent to that node to which it can deliver the highest number of packets as the final destination. It may happen that among the nodes from which A receives the *info* messages, none is the destination, like in the example we are currently discussing. In such case, A sends a response message to a node with whom it has the highest contact duration which is also higher than ς. If no such node exists, A does not send any *response* message, goes silently to the idle state and retransmits the *beacon* message after a random time. Node A also includes a summary of its own buffer in the *response* message. Let us assume that the contact duration calculated between node A and node B is higher than ς; hence, A selects B to send the *response* message. In the message, A includes the share of the contact duration of B, which is $T_s = \eta T_c/2$.[2] Since the *response* message is transmitted to only one node, other *info* transmitters go back to the idle state.

When B receives the *response* message, it realizes the share of the time it acquires and from the summary of A's buffer, and it also realizes the recent information A is expecting from it. Node B then retrieves packets from its buffer to be transmitted and it is done in such a way that no packet is transmitted if A has no interest in it (referring again to our coastal surveillance application, no packet is transmitted to A about a given asset if A has more recent information about that asset). The packets are retrieved from the buffer so as to maximize the transfer of information to A regarding the status of the assets. Assuming that B is storing data about multiple assets, it will transmit one data packet per asset in a

[2]The only exception is when every node transmits *response* packet to the sink which is usually assumed to be fixed in underwater scenario, where A will be the only node to transmit, and hence, $T_s = \eta T_c$.

round-robin fashion, starting from the most recent packets (i.e., those with highest priority), until its transmission time is over.

After all packets are selected from the buffer, B enables a timer at the lower layer so that it can only transmit within its own share of the contact. After that all data packets are delivered to the lower layer, they waits for ACKs in accordance with the ARQ scheme implemented in UDTN. While the details of the scheme are given in Subsection 5.3.2, here we wish to highlight an inherently cross-layer behavior in UDTN. Namely, the ARQ scheme always informs the routing protocol about the correct reception of the packets by the current destination (in this case, by A) via cross-layer messages. This helps the routing layer understand which messages have already been delivered to a node (in order not transmit the same packets twice), and allows the nodes to remove the packets from the queue if they have been correctly delivered to the final destination.

After the transmission of B is over, A starts transmitting its own packets. It may happen that a node may not have enough data packets to transmit which can fill up the estimated transmission time. To avoid the time allocation loss, a flag, called *last packet*, is set in the header of the last packet which notifies the other node that reception time is over. Another notable benefit of this approach is that if the transmission of the first node finishes earlier than expected, the other node can transmit until the contact duration ends. Moreover, it also may happen that a node may not have any new packet to transmit. To inform this status to the other node, a node sends a *proxy* message and changes its state accordingly. When transmission and reception of both the nodes are over, they move back to the idle state and transmit beacon packets periodically after a random time.

One further control procedure is applied during the data transmission process. Namely, if the nodes sending data packets do not receive ACKs for a given time, they will assume that the receiver has moved out of range, and they will hence stop transmissions. The same happens if the receiver does not receive any data for a given time.

5.3.2 Modified Underwater Selective Repeat (USR)

Though error control techniques are necessary to deal with the erroneous channel conditions of underwater networks, one may assume that implementing the error control technique in underwater networks where propagation delay is already high enough is not ef-

ficient. However, *Underwater Selective Repeat (USR)* is designed in such a way that it can exploit the long propagation delay by transmitting multiple packets in a single *Round-Trip-Time (RTT)* and thus, performs comparable to non error control based protocol. Packet transmission in *USR* is spaced in such a way that the transmission of a *data* packet does not collide with the reception of any *ACK* of previously transmitted packets. *USR* is adaptable and can switch between the *Selective Repeat (SR)* and the *Stop & Wait (S&W)* ARQ protocols according to the necessity. As the name suggests, *USR* usually behaves according to the SR technique but reverts to plain S&W when, $i)$ there is only one packet in the buffer to transmit, $ii)$ the distance between the source and destination does not allow to transmit more than one packet, i.e., they are closer to each other and multiple packet transmission may not be possible, and $iii)$ a node is transmitting an initial data packet, which is used to determine whether to employ S&W or SR technique. From the reception of the *ACK* of the initial packet, a node calculates the RTT between the nodes in contact which is equal to $2 \times \tau_{AB}$ where τ_{AB} is the one-way propagation delay between node A and node B. Since the nodes are usually mobile in an underwater DTN, the estimation of RTT also should consider this aspect so that packet transmissions do not superimpose to *ACK* receptions as nodes move. To achieve this, we employ a conservative approach where we assume that the nodes move towards each other, i.e., the RTT is progressively reduced as time goes by. Let us assume that node A previously met with node B at time t_1. If node A and node B meet at any later time t_2, A can compute the updated approximate propagation delay, τ'_{AB}, as

$$\tau'_{AB} = \tau_{AB} - \frac{N_M(t_2 - t_1)v}{c} \tag{5.8}$$

where v is the maximum speed of the mobile node, c is the speed of the sound in the water and N_M can be 0 (if the sender and receiver both are static), or 1 (if one of the two move), or 2 (if both move). Node A updates τ'_{AB} for every packet receptions performed using USR, and thereby adapt to RTTs that vary over time.

If node A discovers that the RTT between itself and B is long enough to accommodate more than one packet transmission, it computes the number of packets that fit within the RTT and starts transmitting so that the reception of *ACKs* will be interlaced with time with the packet transmission, demonstrated in Fig. 5.2. The number of packets (or "window

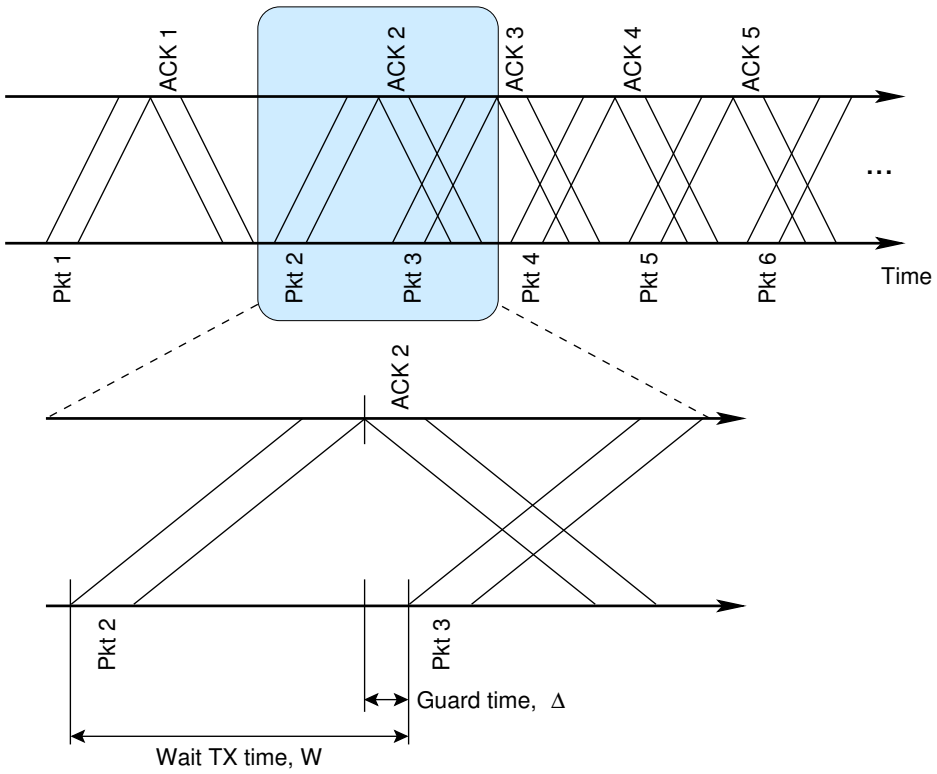


Figure 5.2. *Packet exchange scheme employed in the modified USR protocol (reproduced from [6]*

size") M that A can transmit to B within an RTT can be found as

$$M = \max\left(1, \left\lfloor \frac{k\tau'_{AB}}{T_D + T_A + \Delta} \right\rfloor\right) \tag{5.9}$$

where T_D and T_A are the transmission time of the *data* packet and of the *ACK* packet, respectively, Δ is a guard time, which is required to compensate for mobility-induced changes of the RTT and $0 \leq k \leq 2$ is a tunable parameter, which specifies the portion of τ'_{AB} to be considered. If $k = 0$, from (5.9), it can be observed that $M = 1$, therefore, USR employs a simple S&W ARQ; on the contrary, for $k = 2$ the whole RTT will be considered when computing the window size.

When a node is transmitting multiple packets within a single RTT, i.e., $M > 1$, it may happen that transmission of a packet may block the reception of an *ACK* and thus induces unnecessary retransmission in the network. To deal with this aspect, after transmitting one packet, node A must wait for a fixed waiting time W before transmitting the next packet. W is calculated in such a way that an *ACK* of a previously transmitted packet can be received

in the middle of W. Therefore, W can be expressed as

$$W = \frac{T_A + 4\tau' - 2(M-1)T_D}{2M-1}\,. \tag{5.10}$$

A detailed description of the USR protocol is given in Chapter 3.

For obtaining the best results from the lower layer protocols, it is necessary to adapt them according to the necessity of the upper layer. In our case, we modified the original USR according to the requirements of UDTN in such a way that we can get the best out of it. Unlike the original USR, the modified USR is aware of the duration of the transmission which is set using a *cross-layer* message from UDTN so that transmission of the both contacting nodes do not overlap with each other. Every time before transmitting a packet, a node checks whether within the remaining time period it can receive the ACK of this packet. A packet is transmitted only when the condition is true. When the timer expires, the sender deletes all the remaining packets in the queue. If one packet is not delivered correctly in a contact, the same packet is retransmitted again at the next meeting between the nodes. Another important modification to USR is that it also includes a *cross-layer* to notify about the correct delivery of a data packet, thus reducing unnecessary transmission of the *ACKs* at the routing layer. The reception of an ACK at the routing layer, assists a node to update the status of the packets in its buffer. Namely, the packet is removed from the buffer if it is correctly delivered to the destination; whereas, in other cases, the node information is stored against that packet so that the same packet is not transmitted to the same node at a later time.

5.3.3 Simulation scenario and settings

The performance of the UDTN protocol is evaluated using the ns2-Miracle framework [20], along with the World Ocean Simulation System (WOSS) package [21], which provides ns2-Miracle with an interface to the Bellhop ray tracing tool [22]. The location chosen for simulation is an area of $8000\,\mathrm{m} \times 4000\,\mathrm{m}$ in the Mediterranean Sea, whose upper-left corner is placed at $43.0625°N$, $9.3095°E$. Network operations are assumed to take place in July: the corresponding environmental properties are retrieved from the oceanographic databases employed by WOSS.

All nodes transmit using a Binary Phase Shift Keying (BPSK) modulation, at a bit rate

of $4800\,\text{bps}$. The size of the *data* packet is fixed to $L_D = 125\,\text{Bytes}$, the ACK size is $L_A = 11\,\text{Bytes}$, the *beacon* size is $L_B = 1\,\text{Byte}$, the minimum length of an *info* packet is $L_I = 58\,\text{Bytes}$ and the minimum length of a *response* packet is $L_R = 11\,\text{Bytes}$. The simulations have been performed for various data generation rates per node, λ, from 0.18 to 6 packets per minute per node. We considered two different numbers of assets to be inspected (hence of AUVs in the DTN), 3 and 6. The nodes (both the assets and the inspecting AUVs) are initially deployed at random within the area. After this, the assets start moving freely around the area according to a Gauss-Markov mobility model [23]; the follower AUVs start moving to approach and follow the assets in accordance with the rules of the mobility model described in Subsection 5.3.3.1. A sink is placed near the shore, and represents the transceiver connected to the shore-based control center.

We compare the performance of UDTN against a modified version of the Spray-And-Wait (SAW) [8] protocol. As the original version of SAW [8] assumes a reliable packet transfer (which is quite unrealistic for underwater networks) we extended SAW to incorporate a S&W ARQ technique for error control. This is also required for a more fair comparison against the UDTN protocol, which implements ARQ via USR. The messaging pattern of the SAW protocol has also been modified to favor the exchange of fresh information among the nodes. To this end, all nodes periodically transmit beacon messages. Upon receiving a beacon, a node sends a query packet with 40 message ids related to packets that are still to be delivered to the sink. After receiving the query packet, the beacon sender transmits a response packet mentioning which ones of these 40 packets it would like to receive. The requested packets are then sent using a S&W technique. Note that, unlike in UDTN, the communication is not bidirectional, i.e., only the nodes that receive a beacon can transmit. In addition, no adaptation is performed to compensate for time-varying RTTs.

We set $\eta = 0.5$, $k = 2$ and $N_M = 2$. The results are averaged over 100 simulation runs. Each node stops generating packets after $86400\,\text{s}$ or $24\,\text{hours}$ and the simulation ends at $100000\,\text{s}$ or $27.78\,\text{hours}$.

5.3.3.1 Details on the mobility model

In order to simulate the trajectories of mobile assets within the network area, we employ the leader-follower paradigm (or "group mobility model") proposed in [24]. The model

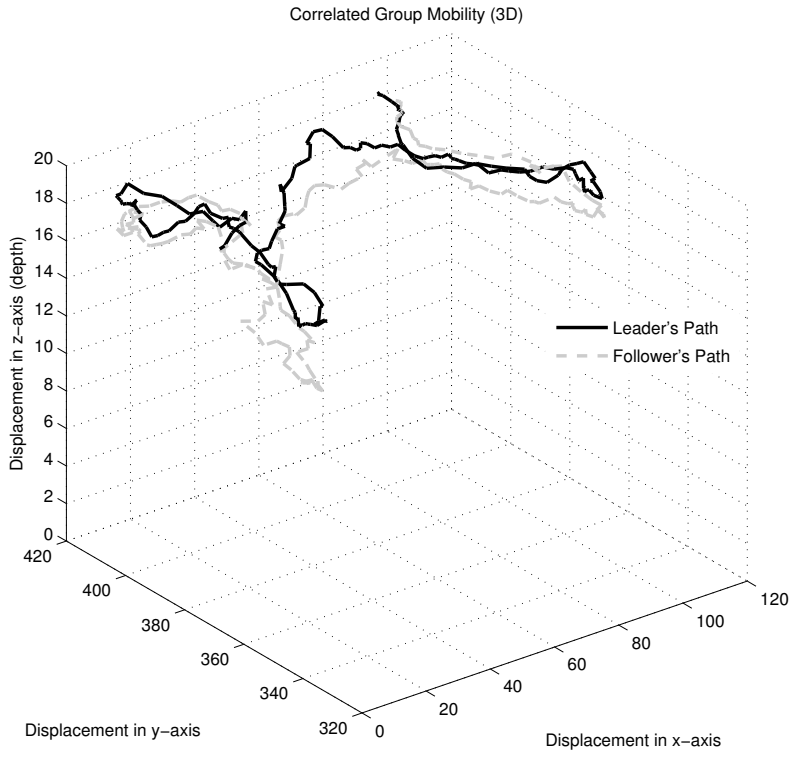


Figure 5.3. *Example of group mobility realization. A "leader node" (representing an asset to be inspected) is moving according to a Gauss-Markov model (black line), and a second node (representing an "inspector AUV") is following the leader (grey line).*

in [24] was designed for 2D terrestrial radio networks: for this investigation, we have extended the movement to take place in a 3D space and to realistically represent an AUV movement (e.g., by forbidding depth changes to be arbitrarily fast).

According to this model, a leader L_i moves either randomly or by following a predefined path, and each follower $F_{L_i,j}$ tunes its movement so that it approaches the route of the leader. In our scenario, the leader node represents any surface or underwater asset that enters the coastal area under surveillance, and thereby has to be inspected; followers represent instead the AUVs that approach these assets for performing the inspection. (In the following, we will use the above terms interchangeably.) Note that a follower can follow only one leader, whereas a leader may have several followers. From now on, we will assume that every leader has only one follower, i.e., an asset entering the patrolled area is inspected by only one AUV.

The movement of the followers consists of two components: i) a movement that attracts the follower towards the leader, and ii) a random movement. The attraction is obtained in a way that is similar to the attraction of electrical charges. Therefore, the mobility model has basically three parameters: the charge of the leader, C_L, the charge of the follower, C_F, and a tunable parameter α which is used to tune the intensity of the attraction field. In particular, a negative value of α attracts the follower towards the leader, whereas a positive value pushes the follower away. By considering the above parameters one can get the following expression for the attraction velocity $\vec{v}_a$ at time t_k [24]:

$$\vec{v}_a(t_k) = \beta(t_k)\frac{C_L C_F}{d^\alpha}\vec{u}_a \tag{5.11}$$

where $\vec{u}_a$ is the unit vector directed from the follower towards the leader and $\beta(t_k)$ is the modulus of the attraction speed at a distance $d = 1\ m$, which is calculated as [24]

$$\beta(t_k) = (1 - \zeta_a)\beta(t_{k-1}) + \zeta_a s_{a,k} \tag{5.12}$$

where $0 \leq \zeta_a \leq 1$ is a tunable variable, and $s_{a,k}$ is a Gaussian random variable with mean s_m and standard deviation s_v. Finally, the random velocity $\vec{v}_r(t)$ of the follower's movement is obtained through a Gauss-Markov mobility model [23]. Hence, the speed vector of a given follower at time t can be calculated as

$$\vec{v}(t) = \vec{v}_r(t) + \vec{v}_a(t) \tag{5.13}$$

Fig. 5.3 shows an example of leader-follower mobility pattern in a 3D area, obtained using the technique described above, with parameters $C_L = 2$, $C_F = 2$, $\zeta_a = 0.8$, $s_m = 0.02$ and $s_v = 0.005$. These parameters have also been used throughout the simulation campaign.

5.3.4 Simulation results

As a first comparison between UDTN and SAW, we show in Fig. 5.4 the Packet Delivery Ratio (PDR) of the UDTN and SAW protocols, defined as the ratio of the number of packets delivered to the sink to the number of packets generated in the network. The PDR is shown as a function of the data generation rate per node, λ, for a first configuration with 3 assets and 3 followers, and a second configuration with 6 assets and 6 followers.[3] We also recall

[3] From now on, we will refer to these configurations as the 3-node and the 6-node networks, respectively: this also reflects the fact that only the followers communicate.

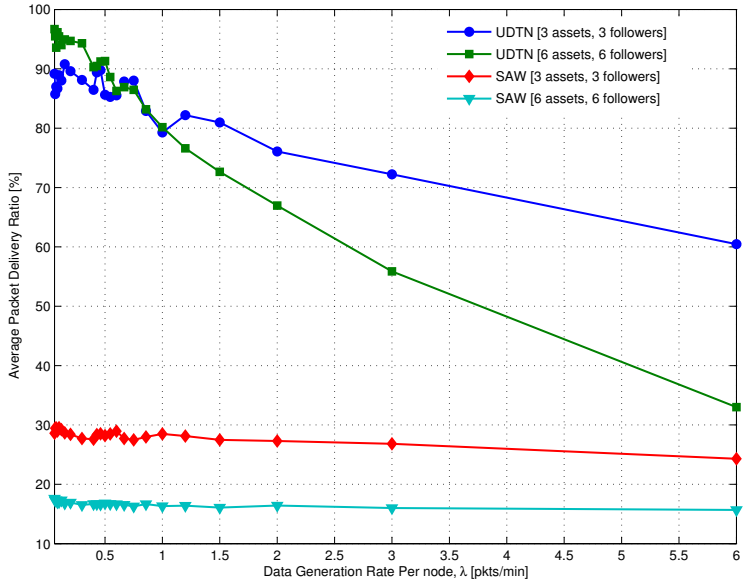


Figure 5.4. *Packet delivery ratio as a function of the data generation rate per node for UDTN and SAW.*

that a follower is always assigned to one and only one asset. The first observation is that UDTN outperforms SAW for all chosen values of λ. When the data generation rate is low, the 6-node network experiences a higher packet delivery ratio due to the denser deployment of the nodes, which increases the chance of contact. In turn, this favors the delivery of data to the sink, possibly across multiple intermediate store-and-forward steps. However, the performance of the 6-node network decreases due to greater interference and contention as the packet generation rate per node is increased. For higher values of λ, in fact, the 3-node network experiences a higher packet delivery ratio, which also decreases more smoothly due to the lower interference affecting the communications.

Similar observations hold for SAW, where however the PDR of the 3-node network is always greater than the PDR of the 6-node network. In any event, SAW is consistently outperformed by UDTN. There are several reasons for this. Namely, UDTN allows both nodes to transmit during a contact, whereas in SAW only one node can transmit. Furthermore, the S&W ARQ scheme in SAW introduces delays between subsequent packet transmissions, whereas the modified USR technique employed with UDTN exploits the long RTTs typical of underwater networks to improve the throughput of the data exchange [6]. Furthermore, the packet transmission between communicating nodes in UDTN is adapted based on the

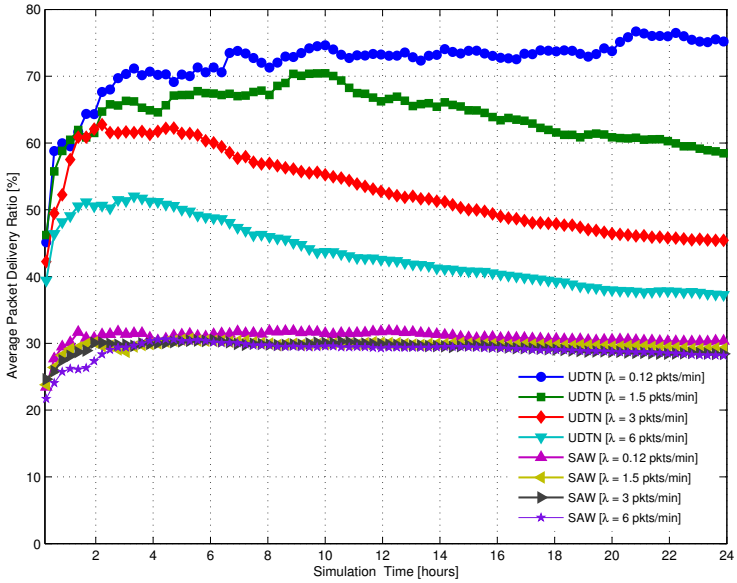


Figure 5.5. *Packet delivery ratio for one network node as a function of the simulation time for UDTN and SAW for a randomly selected node in a scenario with 3 assets and 3 followers, for several values of λ in packets per minute per node.*

computation of T_s, as described in Subsection 5.3.1, whereas no adaptation is performed in SAW.

The evolution of the PDR at different time epochs during the simulation is shown in Fig. 5.5 for one network node, and for different values of the packet generation rate per node. The PDR is initially low in all cases, as the few contacts that still occur are due to the random deployment of the nodes. The PDR starts increasing as time goes by and subsequent contacts allow the nodes to exchange packets and to deliver them to the sink using store-and-forward. For UDTN, in the presence of packet generation rates of 1.5 packets per minute per node and above, however, the number of generated packets exceeds the capability of the network to transport the packets to the sink, so that the node buffers build up: eventually, the nodes will drop old packets to free space for newly generated ones. As a result, the PDR decreases towards the end of the simulation. Notably, the PDR of SAW is consistently lower, at about 30% in all configurations, and is constant throughout the simulation. This is due to the lower number of transmissions that take place in SAW, due both to the absence of bidirectional communications within a contact and to the use of S&W. Fig. 5.5 can also

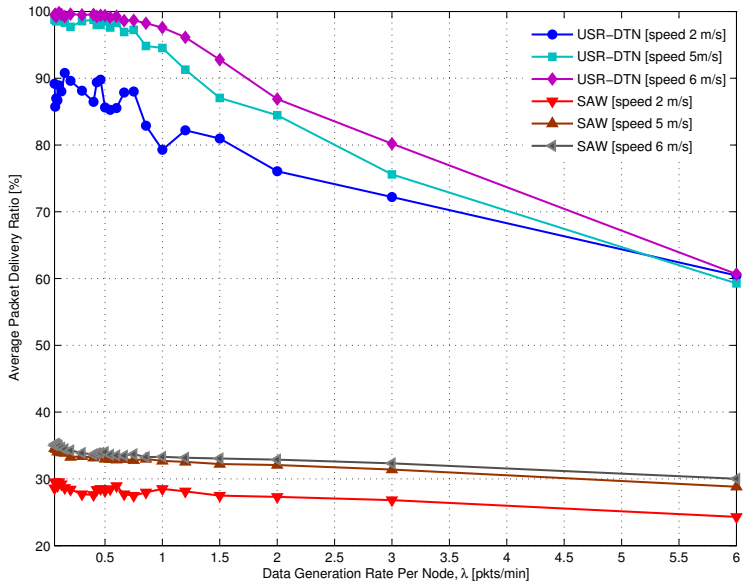


Figure 5.6. *Packet delivery ratio as a function of the packet generation rate per node, λ in packets per minute per node, for several values of the asset and follower speed, in a scenario with 3 assets and 3 followers.*

be used to understand the maximum packet generation rate per node that is sustainable in the network. Namely, if the curves keep increasing or are stable with the simulation time, we can infer that the network can correctly convey all generated traffic to the sink. In this 3-node network case, the maximum sustainable packet generation rate is between 0.12 and 1.5 packets per minute per node.

Fig. 5.6 shows the PDR of UDTN and SAW in a 3-node network for three values of the speed of the assets and of the follower AUVs. Since a higher speed translates into more frequent contacts, the PDR increases with increasing node speed for both protocols. In particular, for lower values of λ, the PDR of UDTN reaches a value close to 100% for speeds of 5 and 6 $\mathrm{m/s}$, which is around 10–15% higher than that the PDR at $2\ \mathrm{m/s}$. Some improvement in the PDR is also observed for intermediate values of λ. For higher values, the shorter duration of the contacts induced by the higher speeds (and the consequent lower number of exchanged packets per contact) dominates, making the PDR equivalent to that of the lowest speed case.

Finally, we recall that in the coastal surveillance application considered here, one of the

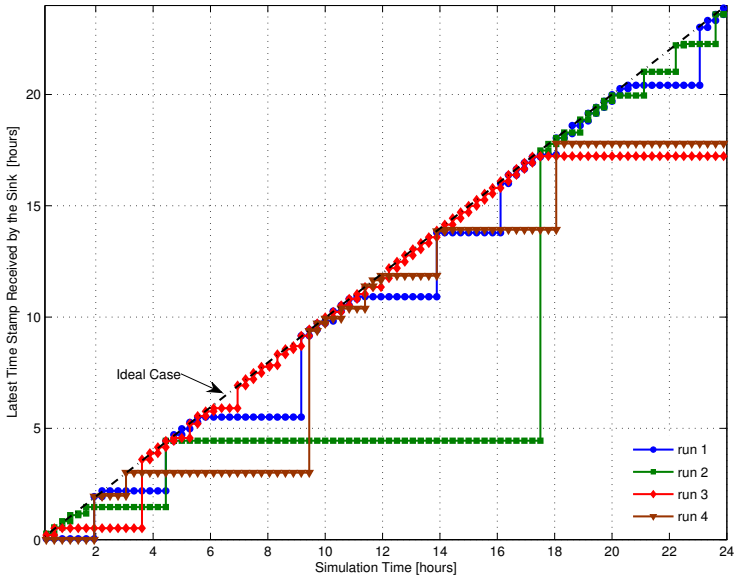


Figure 5.7. *Latest packet received by the sink as a function of the simulation time for UDTN in a scenario with 3 assets and 3 followers, and for $\lambda = 6$ packets per minute per node.*

objectives is to deliver timely information to the sink. In particular, we want to measure how often the sink is updated by the network with new data about the assets being followed. The evolution of the latest timestamp of the data delivered to the sink as a function of the simulation time is shown in Fig. 5.7. We set $\lambda = 3$ packets per minute per node

From the figure we can observe that the sink is typically up to date with new information. If long periods of "starvation" occur, it is a mobility issue (no node is in contact with the sink for a long time), not a problem of the UDTN protocol. On the contrary, the capability of UDTN to convey data to the sink via opportunistic transmissions, perhaps through multiple store-and-forward steps, effectively keeps the sink up to date when nodes are in range, as seen from the cases when the curves in Fig. 5.7 closely follow the ideal case line, and from the packet delivery ratio values in Fig. 5.5.

5.4 Underwater DTN routing protocol with Probabilistic Spray (UDTN-Prob)

One of the major drawbacks of the UDTN protocol is the replication overhead. Every node replicates packets as many times as it wants and hence, achieves a higher packet delivery ratio. However, a high number of replications imposes a high replication overhead, wasting necessary bandwidth. Therefore, it is necessary to reduce the replication overhead without affecting the delivery ratio. Therefore, we enhanced the UDTN protocol by employing a probabilistic binary spray technique. Probabilistic spray technique restricts every node to deliver only those packets which pass a given criterion and hence, help decrease the replication overhead. In this technique, when two nodes are in contact with each other, they also exchange the packet deadline information through control packets. As anticipated in Subsection 5.4.1, we assume that the nodes know the statistics of the contact durations and inter-contact times between themselves and any other node, in particular with the sink.Therefore, they only exchange those packets which have higher lifetime than the provided packet deadline. Thus, the proposed protocol decreases the number of packets injected into the network by allowing every node to exchange only those packets which the other node may be able to deliver in time to the destination by itself or through another node. Furthermore, when transmitting the packets, a binary spray technique employed in order to reduce the replication of the packets.

Since the preliminary messaging technique and modified USR employment is almost similar, therefore, in this section, we are going to discuss only the differences and enhancements made over the UDTN protocol. Keeping the descriptions of the UDTN in mind, we can further divide our descriptions in three Subsections. The details of the probability calculation and their arrangement in a table are described in Subsection 5.4.1. The minimum deadline computation procedure is illustrated in Subsection 5.4.2. In Subsection 5.4.3, we are going to describe the changes made in the preliminary signaling messages so that the UDTN-Prob can reduce replication overhead.

5.4.1 Details on the Probability Distribution Table

Every node which employs UDTN-Prob protocol stores a probability table. In the probability distribution table, the CDF of the inter-contact duration of different nodes are stored. We assume that this node is already available at everynode. In our implementation we derive it by running a *Gauss-Markov* mobility model [23] and by collecting the mobility traces of all the nodes for six months. From the mobility traces, we compute the number of contacts between different nodes, inter-contacts time between them, the contact durations and their statistics. We then compute the Probability Distribution Functions (PDF) for certain inter-contact time, t. Recall the example previously mentioned and assume that node A has several *inter-contacts times*, Δ_{iAB} with node B, where $i = 1, 2, 3, ..., n$. The number of contacts taking place at every t interval is δ_t where $\delta_t \leq n$. Using δ_t, we can compute PDF (f_{AB}) at t as

$$f_{AB}(t) = \frac{\delta_t}{n} \tag{5.14}$$

Now, the CDF for t can be computed as

$$F_{AB}(t) = \int_0^t f_{AB}(t')dt' \tag{5.15}$$

The computed CDF value is then stored in the probability distribution table for the time interval t. In practice, in the probability table, we store one value of the CDF for every $10\,\text{minutes}$ of inter-contact time.

5.4.2 Minimum Deadline Computation

In UDTN-Prob, the two nodes which are understood to be in contact with each other also compute exchange the minimum of all the packets delivery deadlines for the packets in their buffer, and exchange this information. This makes it possible to decrease the number of transmissions by restricting the sender to send only those packets which have a lifetime higher than the minimum deadline. Every node can compute this minimum deadline using any technique of the two techniques described below:

5.4.2.1 One hop forwarding

In this technique, the minimum deadline is compared equal to the probability that the contacted node meets the destination within a given time. Recall the example of 4 nodes and

let us assume that node A meets with node B. Node A and node B query their probability table to acquire the probability that it will meet the destination, D, before any given time. For any node i, CDF can be found in the following way

$$F_{iD}(t) = P[node\ i\ meets\ node\ D\ in\ less\ than\ t] \tag{5.16}$$

Assuming that the meeting time must be lower than a threshold, θ, the amount of time that satisfies this constraint is $T_i = {F_{iD}}^{-1}(\theta)$. Node A and node B then exchange their relative minimum deadlines, T_A and T_B, with each other through control packets. Since node A knows the minimum deadline of node B, it only transmits those packets which have a packet lifetime $\geq T_B$. Node B also applies the same condition in transmitting packets to node A.

5.4.2.2 Two hop forwarding

In this case, minimum deadline is not computed by considering only its own meeting time, but it also considers the probable meeting time of other nodes. Let us assume that any node j knows the function $F_{kD}(t)$ for all other nodes k, $1 \leq k \leq N, k \neq j\ and\ k \neq i$. Therefore, node j knows that it is going to meet the destination with probability θ within a time T_j. It also knows that it may be able to meet some other node who will then meet the destination, and that this 2-hop forwarding process may in fact take less than T_j. In other words, let us say that node j wants the packets to be forwarded to the destination within $L < T_j$ through another node k. This means that

$$F_{jkD}(L) = P[node\ j\ meets\ node\ k\ in\ less\ than\ t_{jk}\ AND\ node\ j\ meets\ node\ D\ in\ less\ than\ L - tjk] = \theta \tag{5.17}$$

Assuming independence and in correlation of the encounter processes over time we have

$$F_{jkD}(L) = \int_0^L F_{jk}(t_{jk}) F_{kD}(L_n - t_{jk}) dt_{jk} \tag{5.18}$$

and if $F_{jkD}(L) \geq \theta$, then node i can forward any packet with deadline greater than L to node j, who will relay it to the destination via node k.

In practice, node j can query the probability table to get the function $F_{jk}(t)$, but it cannot obtain the function $F_{kD}(t)$ from the probability table. Node k only knows about that

function and node j can realize this from node k. For this reason, when two nodes exchange control packets, they also incorporate a sub-sampled version of the CDF, computed with respect to various probabilities, e.g., in steps of $\theta = 0.05$. Thus, both nodes realize about a portion of the probability table of each other.

Recall the previous example where node A meets with node B and let's assume that both of them know the function $F_{CD}(T)$ of node C. Node A computes L_A such that $L_A = {F_{ACD}}^{-1}(\theta)$ and compare with T_A. If $L_A < T_A$, then the minimum deadline is considered equal to L_A, else, it is considered equal to T_A. Node B also follows the similar computational procedure to discover the minimum deadline.

5.4.3 Changes in the preliminary messages

Since UDTN-Prob routing protocol is not designed specifically for coastal patrol and surveillance networks, it is not necessary to include the summary information of the buffer in the control packets (both in the *info* and the *response* messages). Instead, the minimum deadline and the information about meeting the destination by itself with respect to various probabilities, θ, described in Subsection 5.4.2 are incorporated in the control packets (both in the *info* and the *response* messages) to reduce the replication of the packets.

Aline with previous descriptions, in UDTN-Prob, the selection of the packets from the buffer is performed according to the minimum deadline information received from the other node. In details, when a node queries the buffer with the minimum deadline information, the buffer primarily selects those packets which fulfill the minimum deadline criterion. Since UDTN-Prob also employs *binary spray* technique, it finally selects only those packets which have more than one replication copies. For every selected packet, the node keeps the half of the replicated copies and deliver the other half to the node with which it is exchanging data. Note that the packets with single replication copy will be selected only if the other node is the destination of that packet.

5.4.4 Simulation Scenario

The performance of the UDTN-Prob protocol is first compared with a modified version of SAW [8] protocol which is another viable candidate for underwater DTNs. The details of the modification of the SAW can be found in Subsection 5.3.3. After that, we further evalu-

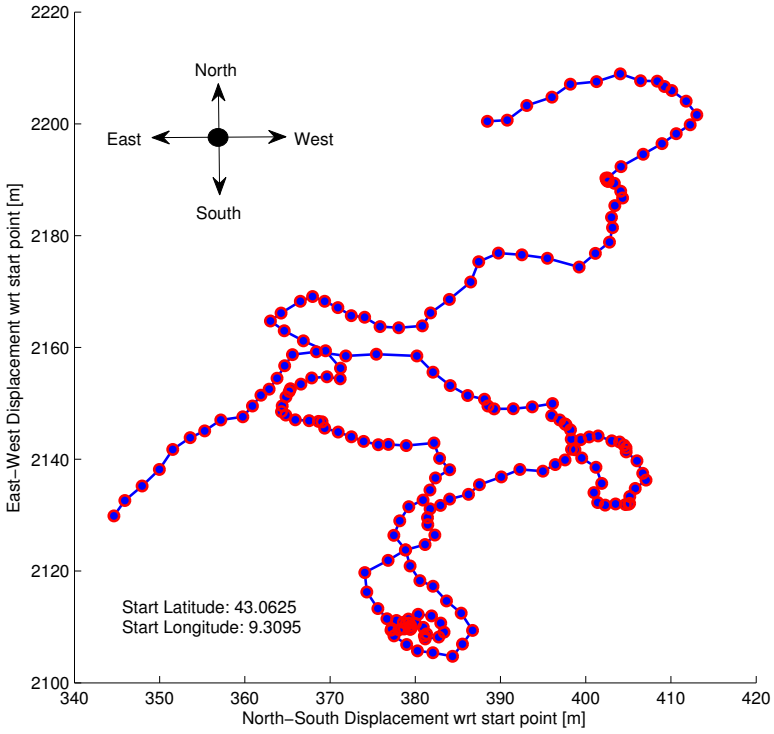


Figure 5.8. *Example of the Gauss-Markov mobility pattern with correlation parameter = 0.8. The blue-filled red circles correspond to the locations where the mobile randomly picks a new velocity vector.*

ate the performance of the UDTN-Prob protocol with respect to various parameters which are closely related to the protocol and uncover their relationships. The simulator we use to evaluate the protocol performance is the DESERT framework [25], which is an extension of the ns2-Miracle framework [20] for underwater communication. Network operations are assumed to take place in an area of $9000\,\mathrm{m} \times 5000\,\mathrm{m}$ whose upper left corner is placed at $39.97°N, 11.82°E$ in the Mediterranean sea in the month of July: the corresponding environmental properties are utilized.

All the nodes are communicating using a Binary Phase Shift Keying (BPSK) modulation technique at a bit rate of $4800\,\mathrm{bps}$ at a central frequency of $25\,\mathrm{kHz}$. The size of the *hello* packet is fixed to $L_H = 10\,\mathrm{Bytes}$, the *info* packet size is $L_I = 127\,\mathrm{Bytes}$, the *response* packet is $L_R = 115\,\mathrm{Bytes}$, the *proxy* packet is $L_{PX} = 10\,\mathrm{Bytes}$, the *data* packet size is $L_D = 125\,\mathrm{Bytes}$ and the *ACK* size is $L_A = 11\,\mathrm{Bytes}$. Traffics are generated according to a Poisson process of rate λ, from 0.12 to 3 packets per minute per node in the network. All the AUVs are

first deployed randomly within the area. After that, they start moving freely around the area using *Gauss-Markov* mobility model [23], with fixed correlation parameter $\alpha = 0.8$, as illustrated in Fig. 5.8. A sink is placed near the shore which is the destination of all the generated packets in the simulation.

Each AUV generates packets for a whole day, i.e., $24\,\text{hours}$ and the simulation stops after $27.78\,\text{hours}$. We set $\psi = 10\,\text{sec}$, $\eta = 0.5$, $k = 2$, $N_M = 2$, $v = 2\,\text{m/s}$, $queue\ size = 1000$ and $packet\ lifetime = 3\,\text{hours}$ throughout the whole simulation campaign unless otherwise mentioned. The results are taken as the average over 50 simulation runs with error bars showing 95% confidence intervals.

5.4.5 Results and Analysis

The performance of the UDTN-Prob and the SAW are compared for various numbers of nodes in the network and for various traffic generation rates, λ. The packet delivery ratio (PDR) of both the protocols is depicted in Fig. 5.9 and Fig. 5.10. The delivery ratio of the UDTN-Prob is further shown for 5 different θ values: $0, 0.25, 0.5, 0.75, 1$. When $\theta = 0$, UDTN-Prob protocol acts like restricted flooding[4] and hence, injects a lots of traffic in the network and achieves higher PDR. For increasing θ, the traffic in the network decrease, and hence, the PDR alo decrease in both networking scenarios. With respect to various numbers of nodes in the network, the PDR of the UDTN-Prob is higher for higher number of nodes increases the contact possibility. In both the scenarios, the SAW demonstrates lower performance than the UDTN-Prob for every θ considered. There are many reasons which influences these performance differences, first of all, the UDTN-Prob allows both the nodes to transmit during a contact, whereas in the SAW only one node can transmit. Moreover, thanks to the probabilistic spray technique of the UDTN-Prob which injects lower but adequate number of packets in the network. Furthermore, the *Stop-&-Wait (S&W)* ARQ scheme in the SAW introduces delays between subsequent packet transmissions, whereas the modified USR ARQ employed with the UDTN-Prob exploits long propagation delays by transmitting multiple packets in a single RTT. Therefore, the UDTN-Prob experiences lower delay and increases throughput of the data exchanged [6]. Again, the packet transmission

[4]This technique is named as restricted flooding since the packet spraying is restricted by a fixed value and a single packet is delivered only once to any node

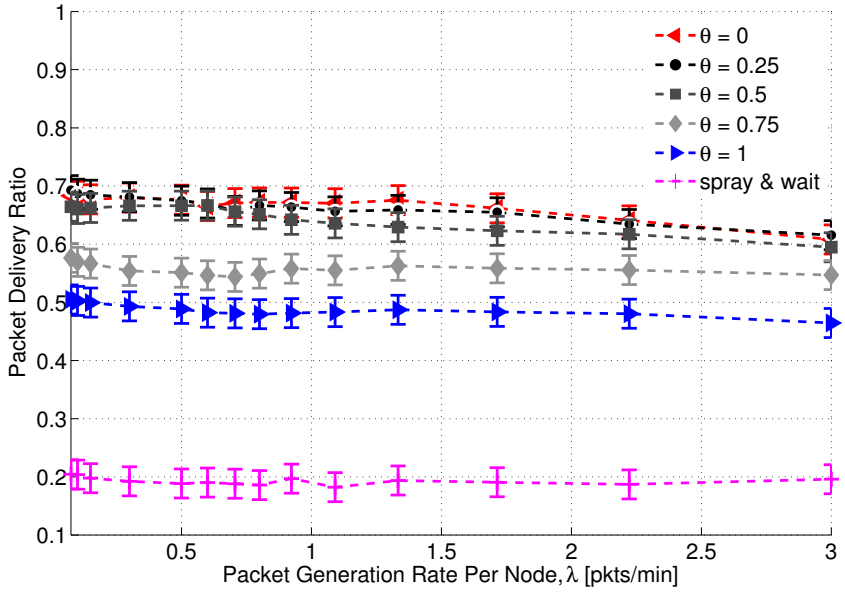


Figure 5.9. *Packet delivery ratio as a function of the data generation rate per node per minute for a network of* 5 *nodes and a sink.*

in the UDTN-Prob is adapted based on the computation of T_s, whereas no adaptation is performed in SAW. Because of the lower performance of SAW, we are excluding it from our further results discussions. From this point onward, we will concentrate on the effects of various parameters of UDTN-Prob.

As we have observed from the previous results discussion that with increasing θ, traffics in the network decreases, which results in the lower PDR and lower overhead. Therefore, it is essential to observe the the impact of θ over the PDR and the overhead. To account for this aspect, we simulate various θ values in both a 5 nodes and a 9 nodes scenario for various λ and the results are illustrated in Fig. 5.11 and Fig. 5.12. As it can be observed from the results, in every case when θ is low, the packet overhead is high since the contacting nodes typically exchange more packets; hence, the PDR is also higher. It can also be observed that when there are a higher number of nodes in the network, the overhead is even higher since more nodes in the network means more contact between the nodes. Therefore, the PDR is also high for such scenario. In most of the scenarios, the overhead approaches to zero when $\theta = 1$. However, when the traffic generation rate is the highest, i.e., $\lambda = 3$, all the nodes have data to send to the destination almost every time they meet with different nodes. Therefore, the overhead is higher even for the higher θ values than the other traffic generation rates.

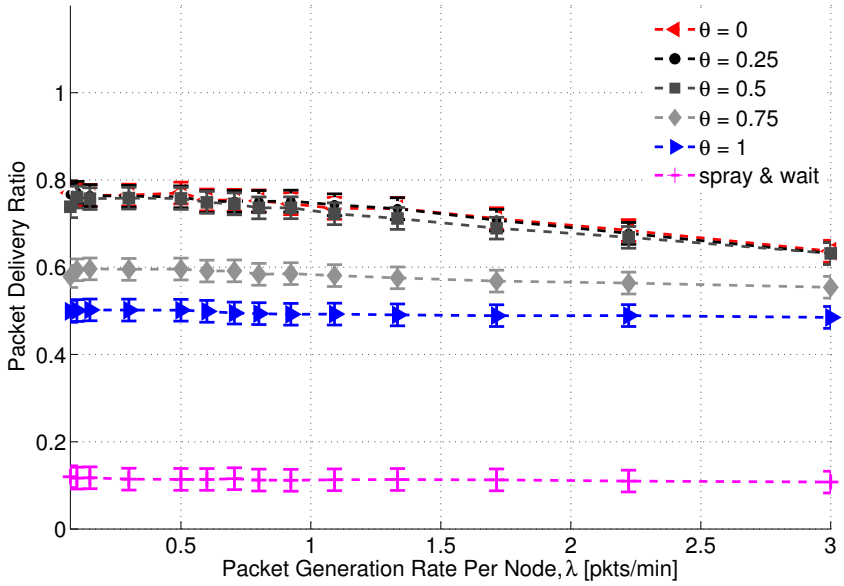


Figure 5.10. *Packet delivery ratio as a function of the data generation rate per node per minute for a network of 9 nodes and a sink.*

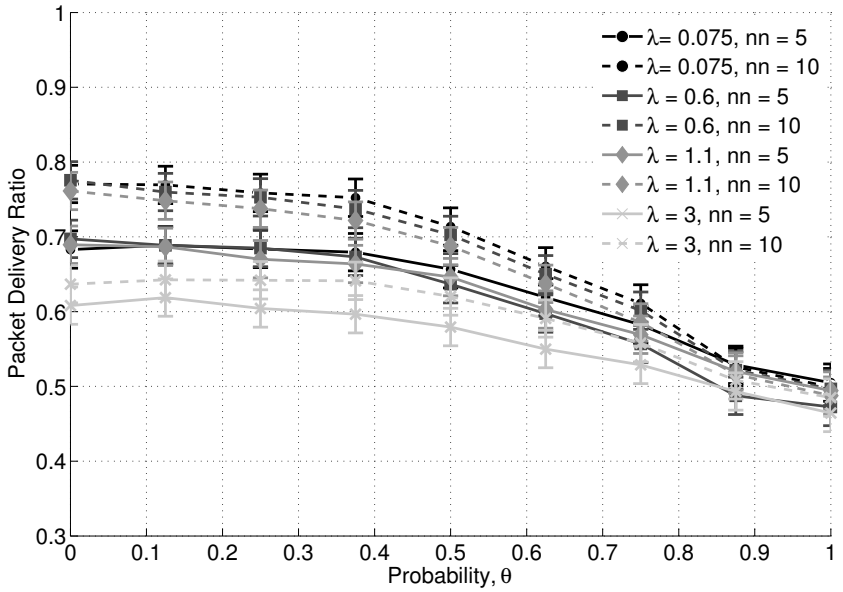


Figure 5.11. *Packet delivery ratio as a function of probability θ for various numbers of nodes in the network and a sink.*

Thus, UDTN-Prob keeps it open for the users to decide what delivery ratio they wish to achieve and what overhead they can compensate.

In Fig. 5.13, we demonstrate the impact of the node velocity on the performance of the

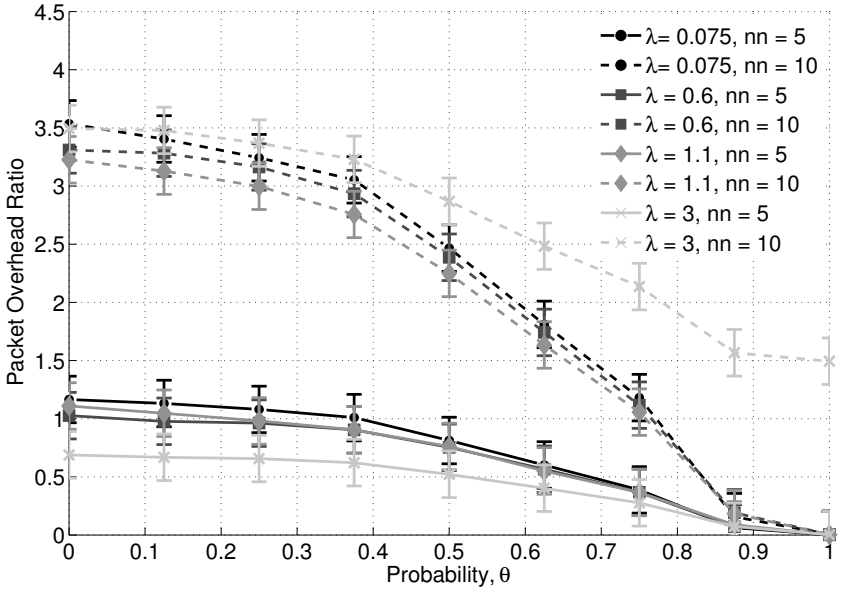


Figure 5.12. *Packet delivery ratio as a function of the data generation rate per node per minute for a network of* 9 *nodes and a sink.*

protocol for the velocity of $2\,\mathrm{m/s}$ and $4\,\mathrm{m/s}$ in a network of 5 nodes. From a general point of view it can be argued that if the velocity of the nodes increases, the contact probabilities also increase if they are moving within a fixed area. This argument also seems true for the velocity $4\,\mathrm{m/s}$. Therefore, in the higher θ values, the PDR of the UDTN-Prob with higher velocity is larger than that of lower velocity.

The impact of the window size over their results performance of the protocol is given in Fig. 5.14. To obtain the impact we fixed T_s to $40\,\mathrm{s}$, so that every node tries to send a higher number of packets within a shorter time epoch. Fixing the contact time epoch also indicates that this protocol can be possible to employ in networks where it is hard to calculate T_s. As it is demonstrated in Subsection 5.3.2 that window size M of the USR is adaptive, but we need to fix M_F in this scenario. Again, if the distance between the nodes are long enough to be $M_F \leq M$; multiple packet transmissions will not affect the reception of the *ACK* of the earlier transmitted packet. On the contrary, when $M_F > M$, the reception of the *ACK* of the previously transmitted packet will be affected by the packet transmission. Therefore, the window size is considered as the $min(M, M_F)$ and called M' for the rest of the discussion. From the figure it can be noticed that when the traffic generation rate λ was low, M' has little impact on the PDR. However, when λ is high, with increasing M' the PDR also increases.

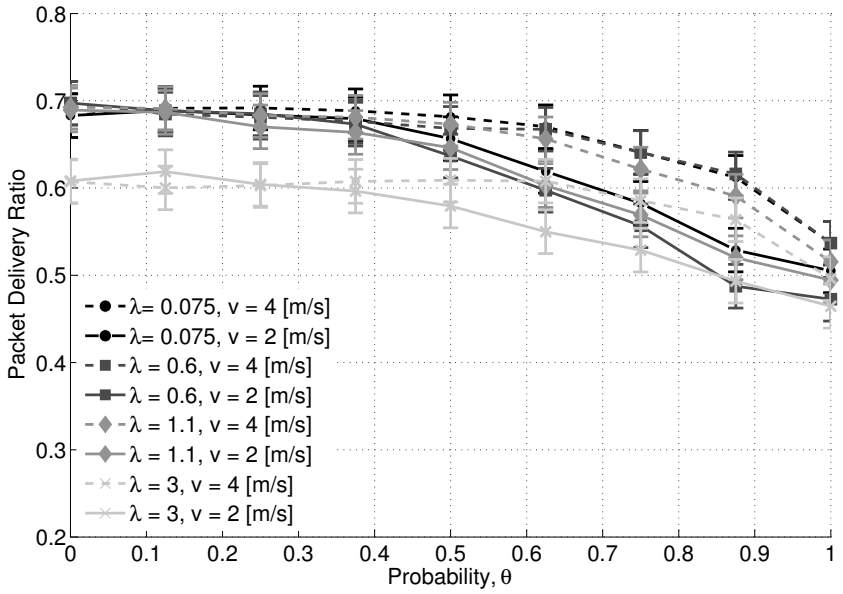


Figure 5.13. *Packet delivery ratio as a function of node velocity for a network of 5 nodes and a sink.*

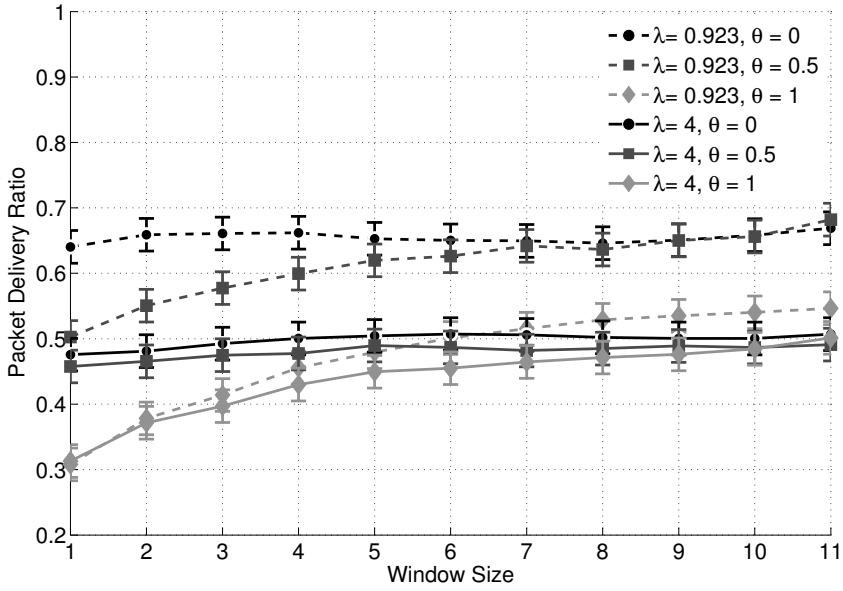


Figure 5.14. *Packet delivery ratio as a function of window size M for a network of 5 nodes and a sink.*

Because, when M' is higher, a node can transmit more packets in a single RTT and hence, achieves a higher delivery ratio.

All the results we have discussed so far are obtained employing a *1-hop forwarding* technique, which is relatively easy and simple to implement. However, in Subsection 5.4.2, another technique of selecting minimum deadline is also discussed, called *2-hop forwarding*

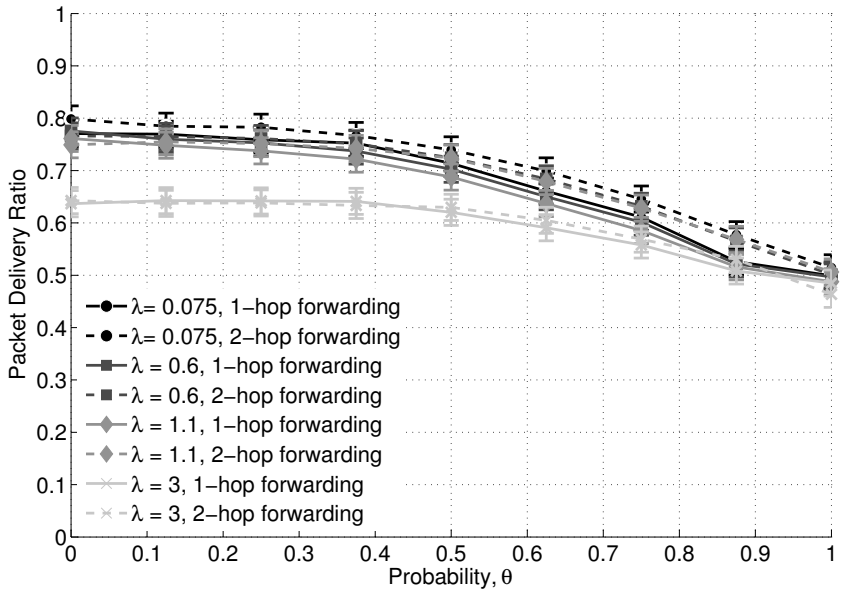


Figure 5.15. *Packet Delivery Ratio as a function of probability θ for a network of 9 nodes and a sink.*

technique. In Fig. 5.15 and Fig. 5.16, the PDR and the overhead of these two forwarding techniques are shown for a network of 9 nodes. From the figures, it can be observed that when θ is low, both the techniques assist a node to transmit an almost similar number of packets to the other contacting nodes, hence, their PDR and overhead are comparable. However, for higher values of θ, the PDR of *2-hop forwarding* is higher; because, it assists a node to discover a lower minimum deadline than that of *1-hop forwarding*. In *2-hop forwarding* technique, a node not only considers its own meeting time with the destination, unlike in *1-hop forwarding*, but also consider whether or not the met node can deliver a packet earlier through another node. Therefore, also in higher probability, every node exchanges relatively higher number of packets which is again reflected in the overhead showed in Fig. 5.16. This is up to the users of the protocol whether they want further complexity in the protocol to achieve a slight improvement over the delivery ratio.

5.5 Conclusions

In this chapter, we described two *replication-based* routing protocols, named Underwater DTN (UDTN) and Underwater DTN routing protocol with probabilistic spray technique (UDTN-Prob). Both the protocols employ an efficient data exchange technique to make the

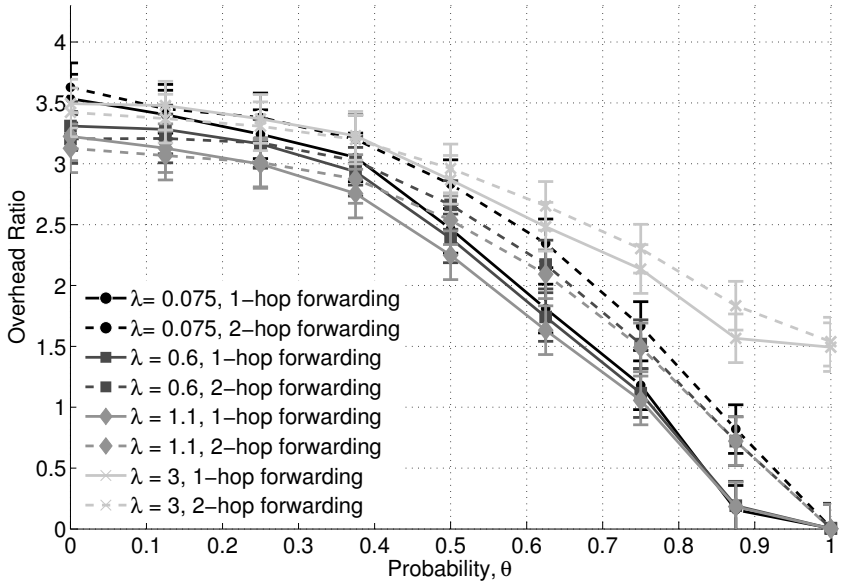


Figure 5.16. *Overhead as a function of probability θ for a network of 9 nodes and a sink.*

best use of the infrequent contacts among the nodes. The protocols estimate the contact duration among the two nodes that understand to be in contact with each other by exchanging information through control packets, so that both nodes in contact can have a fair share of the contact time to transmit their own data. They also incorporate a modified version of the USR ARQ technique in order to handle the erroneous characteristics of the underwater channel.

Since UDTN is designed specifically for coastal patrol and surveillance networks, the nodes exchange a summary of the information in their buffer so that they can feed the shore-based control center with the newest information. UDTN does not employ any technique to restrict replication of the packets other the expiration of packet lifetime. However, the UDTN-Prob is designed for other DTN architectures. Therefore, it does not include any buffer information in the header. It employs a probabilistic spray technique and a binary spray technique for efficient forwarding of the packets and for restricting the number of packets injected into the network. With other information, every node also includes the minimum deadline information in the control packets which instructs the other node to send only those packets which have higher lifetime than the provided minimum deadline. Thus, the UDTN-Prob lower the number of packets injected into the network by allowing every node to exchange only those packets which other node may be able to deliver to

the destination by itself or through another node. The minimum deadline is computed with the help of a probability distribution table where the CDF of meeting with different nodes are stored for various inter-contact duration. Furthermore, a binary spray technique is employed in the protocol which reduces the replication of the packets even more.

From the results discussed in the previous section, we can conclude that both the proposed protocols perform better in their respective scenarios than other existing ARQ based DTN routing protocols, i.e., SAW.

Acknowledgment

This work has been supported in part by the Italian Institute of Technology within the Project SEED framework (NAUTILUS project), and by Whitehead Alenia Sistemi Subacquei, Livorno, Italy, and the European Defense Agency, under the RACUN project.

The authors would like to thank Prof. Simin Nadjm-Tehrani for sharing an ns2 version of the Spray-and-Wait protocol.

References

[1] S. Azad, P. Casari, and M. Zorzi, "Coastal patrol and surveillance networks using AUVs and delay-tolerant networking," in *Proc. of MTS/IEEE OCEANS*, Yeosu, South Africa, May 2012.

[2] C. E. Perkins and E. M. Royer, "Ad Hoc on-demand distance vector routing," in *IEEE WMCSA*, New Orleans, LA, US, February 1999.

[3] P. Xie, J.-H. Cui, and L. Lao, "VBF: vector-based forwarding protocol for underwater sensor networks," UCONN CSE, Tech. Rep. UbiNet-TR05-03 (BECAT/CSE-TR-05-6), Feb. 2005.

[4] M. Goetz, S. Azad, P. Casari, I. Nissen, and M. Zorzi, "Jamming-resistant multi-path routing for reliable intruder detection in underwater networks," in *Proc. of ACM WUWNet*, Seattle, Washington, USA, Dec. 2011.

[5] G. Toso, R. Masiero, P. Casari, O. Kebkal, M. Komar, and M. Zorzi, "Field experiments for dynamic source routing: S2c evologics modems run the sun protocol using the desert underwater libraries," in *Proc. of MTS/IEEE OCEANS*, Virginia, USA, Oct. 2012.

[6] S. Azad, P. Casari, F. Guerra, and Michele Zorzi, "On arq strategies over random access protocols in underwater acoustic networks," in *Proc. of IEEE/OES OCEANS*, Santander, Spain, Jun. 2011.

[7] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Department of Computer Science, Duke University, Tech. Rep. CS-2000-06, Apr. 2000.

[8] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and Wait: an efficient routing scheme for intermittently connected mobile networks," in *Proc. of ACM WDTN*, Philadelphia, USA, Aug. 2005, pp. 252–259.

[9] M. S. Rahim, P. Casari, F. Guerra, and M. Zorzi, "On the performance of delay-tolerant routing protocols in underwater networks," in *Proc. of IEEE/OES OCEANS*, Santander, Spain, Jun. 2011.

[10] A. Lindgren, A. Doria, and O. Scheln, "Probabilistic routing in intermittently connected networks," in *Proc. of ACM MobiHoc*, 2003.

[11] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "DTN routing as a resource allocation problem," in *Proc. of ACM SIGCOMM*, Kyoto, Japan, Aug. 2007, pp. 373–384.

[12] "Aquacomm: Underwater wireless modem." [Online]. Available: http://www.dspcomm.com/products_aquacomm.html

[13] "Underwater acoustic modem models." [Online]. Available: http://www.link-quest.com/html/models1.html

[14] "S2C R 48/78 underwater acoustic modem." [Online]. Available: http://www.evologics.de/en/products/acoustics/s2cr_48_78.html

[15] Z. Guo, G. Colombit, B. Wang, J.-H. Cui, D. Maggiorinit, and G. P. Rossi, "Adaptive routing underwater delay/disruption tolerant sensor networks," in *Proc. of Conference on Wireless on Demand Network Systems and Services*, Garmisch-Partenkirchen, 2008.

[16] F. J. L. Ribeiro, A. C. P. Pedroza, and L. H. M. K. Costa, "Deepwater monitoring system in delay/disruption tolerant network," vol. 10, no. 1, pp. 1324–1331, 2012.

[17] M. Zorzi, P. Casari, N. Baldo, and A. F. Harris III, "Energy-efficient routing schemes for underwater acoustic networks," *IEEE J. Select. Areas Commun.*, vol. 26, no. 9, pp. 1754–1766, Dec. 2008.

[18] R. Coates, *Underwater Acoustic Systems*. Wiley, 1989.

[19] M. Stojanovic, "On the relationship between capacity and distance in an underwater acoustic communication channel," in *WUWnet*, 2006.

[20] N. Baldo, F. Maguolo, M. Miozzo, M. Rossi, and M. Zorzi, "NS2-MIRACLE: a modular framework for multi-technology and cross-layer support in network simulator 2," in *Proc. of ACM NSTools*, Nantes, France, Oct. 2007.

[21] F. Guerra, P. Casari, and M. Zorzi, "World Ocean Simulation System (WOSS): a simulation tool for underwater networks with realistic propagation modeling," in *Proc. of ACM WUWNet 2009*, Berkeley, CA, Nov. 2009.

[22] M. Porter *et al.*, "Bellhop code." [Online]. Available: http://oalib.hlsresearch.com/Rays/index.html

[23] B. Liang and Z. J. Haas, "Predictive distance-based mobility management for PCS networks," in *IEEE INFOCOM*, New York, NY, US, March 1999, pp. 1377–1384.

[24] M. Rossi, L. Badia, N. Bui, and M. Zorzi, "On Group Mobility Patterns and Their Explotation to Logically Aggregate Terminals in Wireless Networks," in *IEEE VTC Fall*, Dallas, TX, US, September 2005.

[25] R. Masiero, S. Azad, F. Favaro, M. Petrani, G. Toso, F. Guerra, P. Casari, and M. Zorzi, "Desert underwater: an ns–miracle-based framework to design, simulate, emulate and realize test-beds for underwater network protocols," in *Proc. of MTS/IEEE OCEANS*, Yeosu, South Africa, May 2012.

Chapter 6

Conclusions

In this thesis, we first presented the impact of the environment on the performance of the MAC and routing protocol in a shallow water scenario. From our investigation, we observed that the performance of the protocols may vary significantly because of the temperature changes within a day. Then we discussed several protocols which are implemented specifically for underwater acoustic networks to serve different purposes and for different network architectures.

The first protocol we discussed is Underwater Selective Repeat (USR) which is an error control protocol designed to assure reliable data transmission at the MAC layer. USR is designed in such a way that it utilizes the long propagation delay by transmitting multiple packets in a single RTT using an interlacing technique.

After USR, we presented two routing protocols for the surveillance networks, i.e., Multi-Sink Routing Protocol (MSRP) and Multi-path Routing with Limited Cross-Path Interference (L-CROP). Among them, MSRP is a source routing protocol which selects multiple alternative paths employing the graph based technique. However, the problem of the MSRP is that it suffers from large overhead (every packet includes the full path description) with respect to other routing techniques, and also suffers from the unidirectional link problem. To overcome these limitations, we proposed L-CORP protocol which employs a neighbor-aware multi-path discovery algorithm to support low interference multiple paths between each source-destination pair.

Following that, we discussed two routing protocols specifically designed for underwa-

ter delay-tolerant networks, i.e., Underwater Delay-Tolerant Network (UDTN) and UDTN Probabilistic (UDTN-Prob) routing. UDTN is designed for next generation coastal patrol and surveillance networks, whereas, UDTN-Prob is for the other types of DTN networks. Both protocols calculate and divide their contact duration equally so that every node gets a fair share of the contact duration to exchange data and a modified version of USR is employed for error correction. Moreover, UDTN-Prob employs a probabilistic spray technique to restrict the number of packet transmissions.

In the appendix, we presented a simulation framework, named DESERT Underwater (short for DEsign, Simulate, Emulate and Realize Test-beds for Underwater network protocols) which was designed to realize underwater communication through simulation. It is an underwater extension of the NS-Miracle simulator to support the design and implementation of underwater network protocols, which assists the researchers in utilizing the same code designed for the simulator in actual hardware devices.

Appendix A

DESERT Underwater Simulator

Contents

A.1 Overview

The ocean sensing and the monitoring via underwater acoustic networks is fostering a lot of interest in the research community, as it can provide key information about the mechanisms that regulate our planet, as well as the ability to effectively survey water, sea floors, and the coasts on a large scale, in support to various kinds of missions. Recent advances in

robotics, acoustic modems, and advanced control, as well as the innovations expected in the near future, provide most of the ingredients required for the realization of such tasks. One of the missing key enablers for any practical application is, however, a communication and networking architecture that allows heterogeneous nodes to communicate effectively and reliably in the harsh underwater environment. When pursuing the latter goal, researchers need to easily simulate and prototype their protocol solutions, as well as to share the obtained results and allow others to easily repeat the same experiments. A flexible, reliable and publicly accessible tool for performance evaluation is of fundamental importance to test and improve the design of network protocols. Based on the well known and widespread network simulator ns2 [1], DESERT Underwater aims at becoming a useful tool to DEvelop, Simulate, Emulate and Realize Test-beds for Underwater network protocols for the research community interested in the applications of underwater acoustic communications.

The main objective of our work is the realization of a complete set of public C/C++ libraries [2] for supporting the design and implementation of underwater network protocols. In this perspective, DESERT Underwater will extend the NS-Miracle [3] simulation software library, developed at the University of Padova, in order to provide several protocol stacks for underwater networks, as well as the support routines required for the development of new protocols.

NS-Miracle enhances the network simulator ns2 with an engine for handling cross-layer messages and, at the same time, for enabling the co-existence of multiple modules within each layer of the protocol stack. In fact, NS-Miracle shows a high modularity and has been designed to simulate nodes whose logical architecture is as close as possible to what would be found on actual devices.

The set of libraries called World Ocean Simulation System (WOSS) [4] endows the network simulator with the chance to simulate the desired protocols over realistic underwater acoustic channel realizations, and has also been developed based on the NS-Miracle. The design of protocol solutions for underwater networks in NS-Miracle yields two main advantages: i) the developers can reuse a lot of code already written for ns2 with minor modifications, and exploit the modularity of NS-Miracle to better organize the design of their solutions; ii) it is possible to evaluate the performance of the designed protocol stack via simulations that employ accurate channel model tools such as WOSS (introduced above).

Moreover, DESERT Underwater will make it possible to evolve from pure simulation towards the realization of actual prototypes by framing the hardware of real acoustic modems into NS-Miracle itself. In line with recent papers such as [5,6], the idea is to wrap all the commands required to communicate with the modem hardware within an NS-Miracle module. In this perspective, the developer can rely on two supported experimental settings: i) a (small-scale) EMULATION setting, where multiple acoustic modems are connected with a single device (e.g., PC, laptop) and controlled by a single NS-Miracle process as illustrated in Fig. A.1; ii) a TEST-BED setting, where each acoustic modem is controlled by its corresponding unique device (or by a unique NS-Miracle instance, independent of other instances), as depicted in Fig. A.2.

The rest of this chapter is organized as follows. In Section A.2, my contributions in the DESERT simulator are discussed. The different DESERT Underwater libraries are presented in Section A.3; in Section A.4, we discuss preliminary tests that confirm the feasibility of a network prototype exploiting the code of DESERT; Section A.5 concludes the chapter.

A.2 Contributions

The DESERT came to the light because of the contributions of the several people. In this framework, I developed a couple of protocols specifically designed for underwater communication; they are: UnderWater Transport Layer (UWTP), UnderWater ALOHA (UW-ALOHA), UnderWater Medium access Link Layer (UWLL), UnderWater Selective Repeat (UWSR), WOSS Gauss-Markov Mobility in 3D space (WOSSGMMOB3D) and WOSS Group Mobility in 3D space (WOSSGROUPMOB3D). I also developed a couple of other protocols which are going to be included in the next version of the DESERT; they are: Multipath Interference-Limiting Underwater Routing (MILUR) protocol, Probabilistic Spray (Probspray) DTN protocol, Undetwater DTN (UDTN) protocol, Underwater DTN with Probabilistic Spray (UDTN-Spray) and Underwater FiXed Way Point (UFXWP) mobility model.

Nevertheless, for better understanding of this chapter and for realizing the full essence of the DESERT simulator, I include all the descriptions of the modules currently present in it and the experiments which are performed using it.

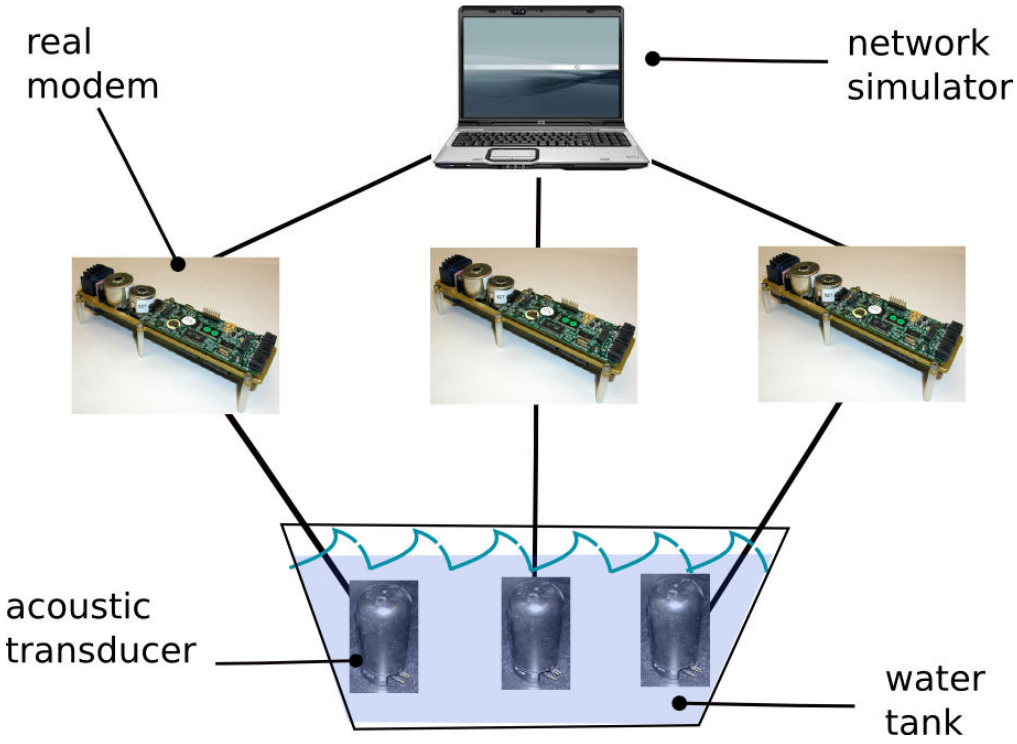


Figure A.1. *Illustration of the EMULATION setting: a single host (or a single NS instance) controls multiple modems.*

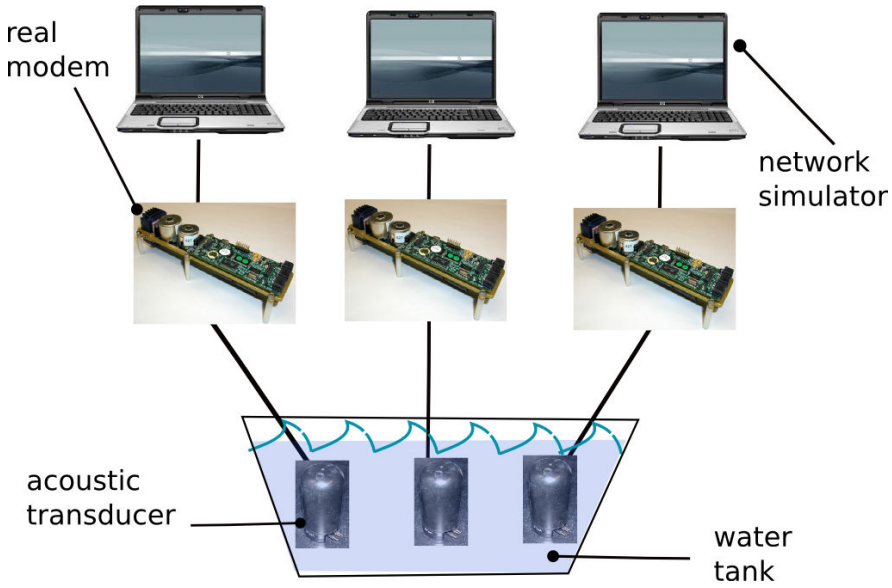


Figure A.2. *Illustration of the TEST-BED setting: each modem is controlled by a single host (or a single NS instance).*

A.3 The DESERT Underwater libraries

In this section we summarize and briefly describe all the NS-Miracle modules that compose the first release of the DESERT Underwater libraries [2]. The objective is to provide a clear picture of the currently available DESERT protocols, in the form of an accessible list of modules grouped according to the stack layers defined by the TCP/IP standard. The presentation follows a top-down approach, i.e., it starts from the upper layers (Application and Transport layers), and proceeds through the Network and Data Link layers. Finally, the modules implemented for the Physical layer are illustrated: these include the interface between the network simulator and the actual modem hardware. At the end of the section, we also illustrate four additional modules implemented to simulate node mobility for underwater network scenarios.

To distinguish NS-Miracle modules that belong to DESERT from those coming from different libraries (e.g., the original NS-Miracle libraries or the libraries of WOSS), all the modules in DESERT are named with a prefix `UW-`. This prefix, however, does not mean that the protocol solution implemented by a given module is optimized for underwater networking (as a matter of fact, some modules are the same as their terrestrial radio counterparts, as is the case for the UDP module). Since ns2, and therefore NS-Miracle, is a simulator based on two different languages (C/C++ for module development and Tcl/OTcl for parameter settings[1]), generally we can refer to a given module by means of three names: i) the name of the module (which corresponds to the name of the folder that contains the C/C++ source files); ii) the name of the corresponding dynamic library that must be loaded before using the module itself, and iii) the name[2] of the corresponding OTcl object (that we must use in the parameter configuration file to create the module itself). Since we need to know all three names in order to use a given NS-Miracle module, in the following we report them for all the presented modules. If not otherwise specified, all the DESERT modules implemented for communication protocols can be used for simulation as well as for both emulation and test-bed purposes.

[1] Tcl is a scripting language that allows simple access to a set of library functions, whilst OTcl is an extension of Tcl to provide object-oriented functionality.

[2] Technically speaking, this is the name of the "shadow" OTcl object associated with the C++ object implemented in the source files. For more details, we refer the reader to the ns2 manual [1].

A.3.1 Modules for the Application Layer

Currently, in DESERT there are two modules for the application layer: the `uwcbr` and the `uwvbr` modules, detailed in the following. Both modules send dummy packets (i.e., with a random payload) and serve to generate network traffic according to two different mechanisms. To work correctly, all the modules of the application layer must be connected to a module of the transport layer (for the technical details on how to connect two or more NS-Miracle modules, the reader may refer to the NS-Miracle documentation [3]).

Name: `uwcbr`

Description: This module implements a Constant Bit Rate (CBR) packet traffic between a sender and a receiver. The data traffic can be generated either by injecting packets in the network with a constant time period or according to a Poisson process with given mean. A single `uwcbr` module represents a data flow between a pair of nodes: if there are two or more nodes transmitting to the same destination, the latter should have an equal number of `uwcbr` modules, one for each flow.

Library name: `libuwcbr.so`

Tcl name: `Module/UW/CBR`

Name: `uwvbr`

Description: This module implements a Variable Bit Rate (VBR) packet traffic between a sender and a receiver. The data packet generation process takes place by switching between two different CBR processes, e.g., having different average packet inter-arrival times. The switch between the processes can be configured by the user by providing the switching epochs. Otherwise, the simulator can be instructed to switch at constant or exponentially distributed intervals. A single `uwvbr` module represents a data flow between a pair of nodes: if there are two or more nodes transmitting to the same node, the latter should have an equal number of `uwvbr` modules, one for each flow.

Library name: `libuwvbr.so`

Tcl name: `Module/UW/VBR`

A.3.2 Modules for the Transport Layer

In DESERT Underwater, two modules are provided for the transport layer: a simple module called `uwudp` and a more sophisticated one named `uwtp`, short for underwater transport protocol. The main features of these two modules are detailed below.

Name: `uwudp`

Description: This module implements the flow multiplexing and demultiplexing from and to the upper layers, respectively. It does not support link reliability, error detection or flow control.

Library name: `libuwudp.so`

Tcl name: `Module/UW/UDP`

Name: `uwtp`

Description: As the previous `uwudp`, this transport layer module handles the multiplexing and demultiplexing of data flows, but it also supports an error control technique, as well as in-order data delivery to the upper layers. In more detail, `uwtp` includes an Automatic Repeat reQuest (ARQ) error control technique. This module can handle both ACK and NACK messages, as well as cumulative ACKs. To work correctly, `uwtp` has to know the destination port number of each application flow it handles: this information must be provided during the setting of the parameters for the simulations or tests that have to be conducted.

Library name: `libuwtp.so`

Tcl name: `Module/UW/TP`

A.3.3 Modules for the Network Layer

The Network Layer is in charge of providing tools for the network interfaces (e.g., addresses) and mechanisms for data routing. In DESERT, we have developed three algorithms for routing and a simple module to manage network addresses whose format are compliant with the IP standard. The details of these modules follow.

Name: `uwstaticrouting`

Description: This module makes it possible to simulate and test data traffic which has to follow predetermined routes. For each network node, there is an option to choose a default

gateway and/or fill a static routing table (whose maximum size is hard-coded and fixed to 100 entries). This information is then exploited locally at each node to forward the network packets, hop by hop, throughout the predetermined paths.

Library name: `libuwstaticrouting.so`

Tcl name: `Module/UW/StaticRouting`

Name: `uwsun`

Description: This module implements a dynamic, reactive source routing protocol. The generation of routing paths can be made based on different criteria, such as the minimization of the hop-count or the maximization of the minimum Signal to Noise Ratio (SNR) along the links of the path. `uwsun` is also designed to collect and process different statistics of interest for the routing level. Currently, this module supports all application modules provided in the DESERT (i.e., `uwcbr` and `uwvbr`), and it can be easily extended.

Library name: `libuwsun.so`

Tcl name: `Module/UW/SUNNode` for the nodes; `Module/UW/SUNSink` for the sinks.

Name: `uwicrp`

Description: This module, which requires very few configuration parameters, implements a simple flooding-based routing mechanism called Information-Carrying Based Routing protocol, see [7].

Library name: `libuwicrp.so`

Tcl name: `Module/UW/ICRPNode` for the nodes; `Moudle/UW/ICRPSink` for the sinks.

Name: `uwip`

Description: This module is used to assign an address to the nodes in a given network according to the standard IPv4 addresses; it provides the Time-To-Live (TTL) functionality and does not implement any routing mechanism. It can be configured to provide all the functional and procedural means intended for an Internet Protocol module (e.g., fragmentation, data reassembly and notification of delivery errors).

Library name: `libuwip.so`

Tcl name: `Module/UW/IP`

A.3.4 Modules for the Data Link Layer

The core of the data link layer is the Medium Access Control (MAC), that administrates the access to the acoustic communication channel. The DESERT Underwater libraries provide six modules which implement as many MAC techniques: `uwaloha`, `uwsr`, `uw-csma-aloha`, `uwdacap`, `uwpolling` and `uw-t-lohi`, all explained in the following. Additionally, DESERT provides `uwmll`, a module to map IP addresses to their corresponding MAC addresses.

Name: `uwmll`

Description: Since node-to-node communications at the link layer are performed using MAC addresses whereas the communications at the upper layers employ IP addresses, a method to associate the latter to the former is required. With `uwmll`, it is possible to set the correspondence between IP and MAC addresses a priori, by filling an ARP table for each network node. Alternatively, ARP tables can be automatically filled using the Address Resolution Protocol (ARP). The `uwmll` module must be placed between one (or more) IP module(s) and one MAC module.

Library name: `libuwmll.so`

Tcl name: `Module/UW/MLL`

Name: `uwaloha`

Description: ALOHA is a random access scheme, i.e., a protocol that allows nodes to send data packets directly without any preliminary channel reservation process. In its original version [8], neither channel sensing nor retransmission is implemented and each node can transmit whenever it has data packets to send. As a consequence, packet losses can occur. In later adaptations [9], ALOHA has been enhanced with acknowledgment packets (ALOHA-ACK). The `uwaloha` module implements the functionality of the basic ALOHA protocol as well as its enhanced version using ARQ for error control. It is possible to freely switch between basic ALOHA and ALOHA-ACK.

Library name: `libuwaloha.so`

Tcl name: `Module/UW/ALOHA`

Name: `uw-csma-aloha`

Description: This module implements ALOHA-CS [9], an enhanced version of the ALOHA

protocol introduced above. ALOHA-CS adds a carrier sensing mechanism to basic ALOHA, in order to help reduce the occurrence of collisions.

Library name: `libuwcsmaaloha.so`

Tcl name: `Module/UW/CSMA_ALOHA`

Name: `uwdacap`

Description: This module implements DACAP (Distance Aware Collision Avoidance Protocol) [10], which provides a collision avoidance mechanism via a handshake phase prior to packet transmission. This phase involves the exchange of signaling packets such as Request-To-Send (RTS) and Clear-To-Send (CTS). This protocol introduces also a very short warning packet in the RTS-CTS mechanism to further prevent collisions among nodes. DACAP is designed for underwater networks with long propagation delays and can be implemented with and without acknowledgements; `uwdacap` implements both solutions.

Library name: `libuwdacap.so`

Tcl name: `Module/UW/DACAP`

Name: `uwpolling`

Description: This module implements a MAC protocol which is based on a centralized polling scheme. To fix ideas, focus on an AUV patrolling an area covered by an underwater sensor field; the AUV coordinates the data gathering from the sensors in a centralized fashion using a polling mechanism. This mechanism is based on the exchange of three types of messages: a broadcast TRIGGER message, that the AUV sends to notify the sensor nodes of its presence; a PROBE message, that the sensors use to answer the initial TRIGGER message; and a POLL message, sent again by the AUV and containing the order in which the underwater nodes can access the channel to communicate their data. This order of polling is determined by the AUV, given the information collected from the PROBE messages which may include, among others, such metrics as the residual energy of the nodes, the timestamp of the data to be transmitted or a measure of their priority. Because of its nature, the algorithm implemented by `uwpolling` does not require any routing mechanism on top of it.

Library name: `libuwpolling.so`

Tcl name: `Module/UW/POLLING_AUV` for the AUV; `Module/UW/POLLING_NODE` for the sensor nodes.

Name: `uw-t-lohi`

Description: Tone-Lohi (T-Lohi) [11] is a MAC protocol that uses tones during contention rounds to reserve the channel. Other nodes competing for channel access are detected during a contention round, by listening to the channel after sending the reservation tone. T-Lohi takes advantage of the long propagation delays present in underwater networks to count the number of contenders reliably, and act accordingly during the contention round. Tone-Lohi can be: 1) synchronized (ST-Lohi); 2) conservative unsynchronized (cUT-Lohi), which enables the counting of all contenders by extending the duration of the contention round to twice the maximum expected propagation delay, and 3) aggressive unsynchronized (aUT-Lohi), to reduce idle times (while increasing the probability of collisions). The `uw-t-lohi` module implements all of three versions of T-Lohi above; the user can freely choose which one to enable. However, since the possibility of transmitting actual tones depends on the available hardware, this module has not been considered for emulation and test-bed; differently, it can be exploited for simulation purposes using WOSS.

Library name: `libuwtlohi.so`

Tcl name: `Module/UW/TLOHI`

Name: `uwsr`

Description: Automatic Repeat reQuest (ARQ) is the basic mechanism to ensure that no erroneous packets are delivered to layers higher than the data-link. When a packet is received with errors, the receiver may ask for retransmissions by sending a small packet called NACK (Negative ACKnowledgment). Similarly, a successful transmission can be notified to the transmitter via an ACK packet. Selective Repeat ARQ allows the transmitter to send up to N consecutive packets before waiting for ACKs or NACKs. The packets sent and not yet acknowledged must be buffered by the transmitter; the receiver can also buffer packets and, in case of errors, only the erroneous packets are sent again. `uwsr` implements a Selective Repeat ARQ mechanism in combination with an Additive Increase and Multiplicative Decrease (AIMD) congestion control technique, similar to TCP's congestion window size adaptation. This protocol has been shown to be effective because the underwater channel

propagation delay is sufficiently large to accommodate more than one packet transmission within one round-trip time (RTT) [12].

Library name: `libuwsr.so`

Tcl name: `Module/UW/USR`

A.3.5 Modules for the Physical Layer: interfaces for real acoustic modem hardware

Currently, DESERT is supporting three different hardware platforms for emulation and test-bed realization: the S2C hydroacoustic modem, a system developed by Evologics [13] which exploits the Sweep-Spread Carrier (S2C) technology for underwater data telemetry and communications; the FSK and PSK WHOI micro-modems, two small-footprint, low-power acoustic modems based on a Texas Instruments DSP and developed by the Woods Hole Oceanographic Institution (WHOI) [14]. However, the `uwmphy_modem` module, which implements the general interface between the acoustic modem hardware and NS-Miracle, can be easily extended to support other different hardware. In the following, we describe `uwmphy_modem`, along with the DESERT modules specialized for the above hardware (`mfsk_whoi_mm`, `mpsk_whoi_mm` and `ms2c_evologics`), as well as an additional module called `uwmphypatch` that is intended for preparing the OTcl scripts needed to set the parameters of the networking prototypes to test.

Name: `uwmphypatch`

Description: `uwmphypatch` is a dumb module to patch the absence of a physical layer when a MAC module is used. It just receives and forwards a packet handling the cross-layer messages required by all the MAC layers of DESERT. The main aim of this module is to observe the behavior of a given network protocol over an ideal channel, before interfacing the network simulator engine with real hardware. It should be used in conjunction with the underwater channel or the dumb wireless channel provided by the NS-Miracle libraries, and mainly to gather insight about the mechanisms of the investigated network protocol (independently of the errors that can be introduced by the channel). Any usage related to performance evaluation is not recommended.

Library name: `libuwmphypatch.so`

Tcl name: `Module/UW/MPhypatch`

Name: `uwmphy_modem`

Description: This module defines and implements the general interface between ns2/NS-Miracle and real acoustic modems. `uwmphy_modem` manages all the messages needed by NS-Miracle (e.g., cross layer messages between MAC and PHY layers) and contains all the simulation parameters that can be set by the user, along with the methods to change them. This module is an abstract class that must be used as base class for any derived class that interfaces NS-Miracle with a given hardware. Therefore, neither an actual object for this module nor its corresponding shadowed Tcl object can be created, hence there is no "Tcl name" associated to this module.

Library name: `libuwmphy_modem.so`

Tcl name: —

Name: `mfsk_whoi_mm`

Description: Module derived from `uwmphy_modem` to implement the interface between ns2/NS-Miracle and the FSK WHOI micro-modem.

Library name: `libmfsk_whoi_mm.so`

Tcl name: `Module/UW/MPhy_modem/FSK_WHOI_MM`

Name: `mpsk_whoi_mm`

Description: Module derived from `uwmphy_modem` to implement the interface between ns2/NS-Miracle and the PSK WHOI micro-modem.

Library name: `libmpsk_whoi_mm.so`

Tcl name: `Module/UW/MPhy_modem/PSK_WHOI_MM`

Name: `mgoby_whoi_mm`

Description: Module derived from `uwmphy_modem` to implement the interface between ns2/NS-Miracle and the WHOI micro-modems (both the FSK and PSK version) using the Goby software [15] to handle the connection with the modems.

Library name: `libmgoby_whoi_mm.so`

Tcl name: `Module/UW/MPhy_modem/GOBY_WHOI_MM`

Name: `MS2C_EvoLogics`

Description: Module derived from `uwmphy_modem` to implement the interface between ns2/NS-Miracle and the S2C EvoLogics modem.

Library name: `libmstwoc_evologics.so`

Tcl name: `Module/UW/MPhy_modem/S2C`

A.3.6 Additional modules to simulate mobility

When underwater networks are simulated, the tested solution should be investigated using accurate models that encompass, among other things, realistic mobility patterns. The DESERT libraries also include four modules to simulate node mobility in 2D as well as 3D scenarios (since in underwater environments nodes can move along any direction in space): `uwdriftposition` and `uwgmposition` are stand-alone mobility modules, whereas both `wossgmmob3D` and `wossgroupmob3D` require the installation of WOSS to work. All of them are detailed in the follows.

Name: `uwdriftposition`

Description: This module implements a mobility model that mimics the drift of a node caused by ocean currents. Given the mean speed and direction of the waves, the initial node's position and its velocity, `uwdriftposition` continuously update the node's location in order to follow the direction of the current. At each update, the new node position is also affected by a random noise that aims at reproducing the waving movement typical of objects floating in the water.

Library name: `libuwdriftposition.so`

Tcl name: `Position/UW/DRIFT`

Name: `uwgmposition`

Description: This module implements the Gauss-Markov Mobility Model [16] (both in 2D and 3D), a solution designed to produce smooth and realistic traces by appropriately tuning a correlation parameter α. When required, `uwgmposition` updates node speed and direction according to a finite state Markov process. Once the desired means speed v_{mean} is fixed, α controls the correlation between the speed vector and direction at state k and that at $k-1$.

Library name: `libuwgmposition.so`

Tcl name: `Position/UW/GM`

Name: `wossgmmob3D`

Description: This module also implements the Gauss Markov Mobility Model, but with some changes that make it directly usable with WOSS. For example, WOSS employs the geographic coordinate system (i.e., latitude, longitude and altitude/depth) for the positions of the nodes; therefore, `wossgmmob3D` also adopts the geographical coordinate system to describe the node movements.

Library name: `libwossgmmobility.so`

Tcl name: `WOSS/GMMob3D`

Name: `wossgroupmob3D`

Description: This module implements a leader-follower paradigm (also known as "group mobility model"). According to this model, we have: i) a leader node, that moves either randomly (i.e., according to a Gauss-Markov Mobility Model) or by following a pre-determined path, and ii) one or more followers that tune their movements so as to mimic the route of the leader. The movement of the followers is generated as the sum of two components: a movement that attracts the follower towards the leader and a random movement. The first one is obtained according to the mathematical model that describes the attraction between two electrical charges [17], whereas the second one is still based on a Gauss-Markov Mobility Model. Three parameters regulate the attractive component of the overall movement of a follower: the "charge of the leader", the "charge of the follower", and a third parameter α which is used to determine the intensity of the "attraction field"; in particular, a negative value of α attracts the follower towards the leader, whereas a positive value pushes the follower away. Like `wossgmmob3D`, also `wossgroupmob3D` supports 3D mobility and, currently, can be used only in conjunction with WOSS.

Library name: `libwossgroupmobility.so`

Tcl name: `WOSS/GroupMob3D`

A.4 Emulation and Testbed settings: A first feasibility test

As illustrated in the previous section, the DESERT Underwater libraries currently provide interfaces between the NS-Miracle network simulator and two commercial modems: the WHOI Micro-Modem, developed by the Woods Hole Oceanographic Institution, and

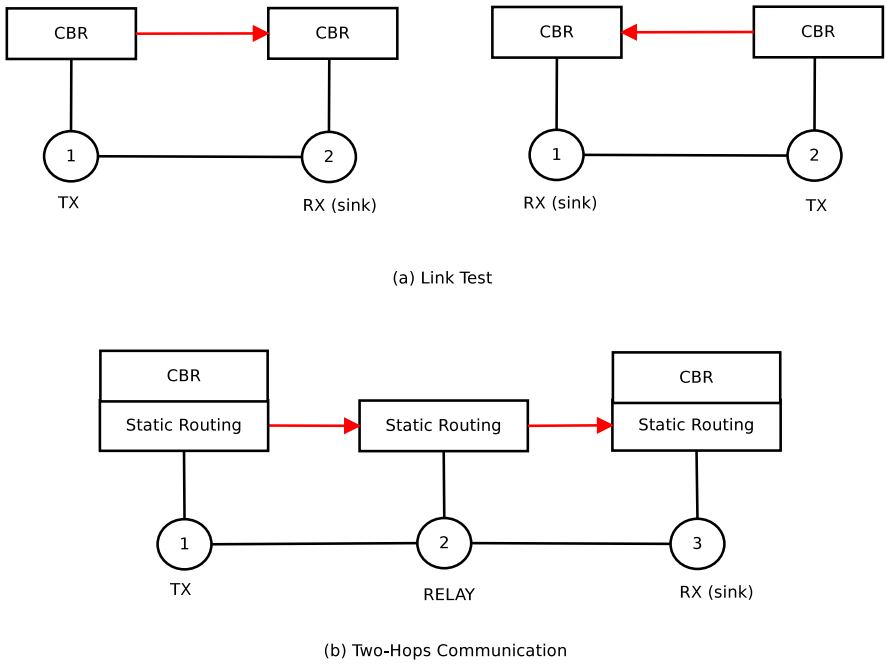


Figure A.3. *Sketches of the performed feasibility tests: (a) bidirectional Link Test (from node 1 to node 2 and vice-versa); (b) Two-Hop Communication.*

the S2C acoustic modem, manufactured by EvoLogics. To thoroughly test and improve the designed interfaces, we are planning to realize extensive experimental campaigns in collaboration with both partners. However, some preliminary tests have already been performed to show the feasibility of the adopted solution.

Part of these tests have been conducted at the Department of Information Engineering of the University of Padova, using three FSK WHOI Micro-Modems. These modems work in the 3-30 kHz frequency range, have a data rate of 80 bit/s and, when powered at 36 V, can reach a working range of up to 2 km horizontally and up to 9 km vertically. They can be controlled using the NMEA [18] communications standard to handle both host-to-modem and modem-to-modem communications. The communication software also provides the use of "minipackets", that have been exploited during our experiments, as they do not require the preliminary transmission of control packets (e.g., initialization messages compliant with the NMEA standard).

A schematic representation of our first feasibility tests is reported in Fig. A.3: (a) first, we have successfully realized a point to point communication, both in the EMULATION

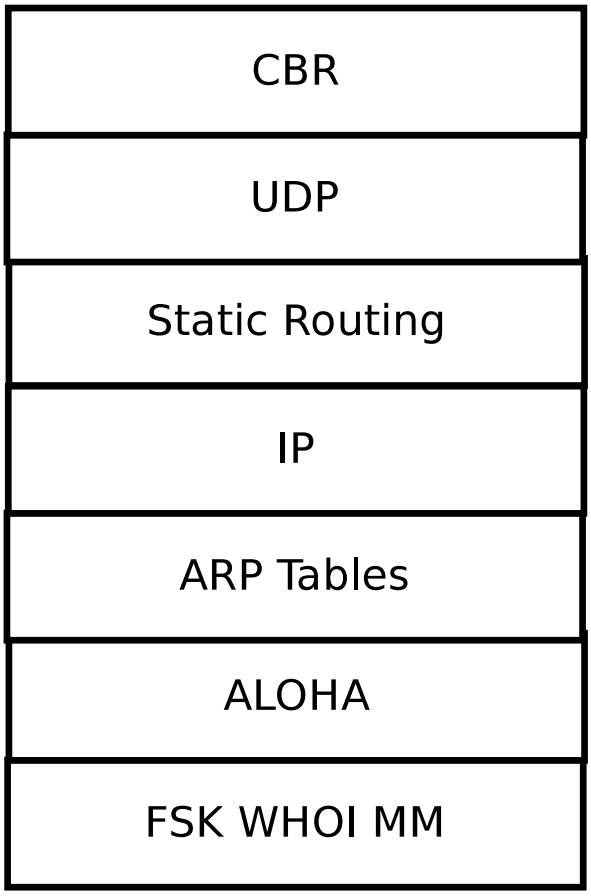


Figure A.4. *Illustration of the protocol stacks used during the feasibility tests.*

and TEST-BED setting; (b) then, in the TEST-BED setting, we have realized a two-hop communication between nodes 1 and 3, using node 2 as a relay. In detail, using the engine of NS-Miracle, each node has been created according to a simple but complete protocol stack using the following modules: `uwcbr`, `uwudp`, `uwstaticrouting`, `uwip`, `uwmll`, `uwaloha` and `mfsk_whoi_mm` (see Fig. A.4). At the transmitter, each data packet is generated by the CBR application; the packet is then sent down through several layers of the chosen protocol stack according to the engine of the network simulator; the information contained in each packet and in its header is then compressed to fit in the Micro Modem minipacket payload using the `mfsk_whoi_mm` module; finally, the minipacket is transmitted over the acoustic channel. At the destination, each received packet follows the reverse path: it is re-generated starting from the payload of the minipacket and mapped into the corresponding data structure of the network simulator; finally, it is sent through the whole protocol stack, up to the

application layer of the destination node.

While there is indeed a functional difference between the EMULATION and the TEST-BED setting, the only technical difference between the two approaches is the fitting of NS-Miracle packets into modem payloads and the reverse operation. In the EMULATION setting, in fact, multiple acoustic modems are connected with a single device and therefore packets can be exchanged among the nodes using the same mechanism as in the network simulator (in our implementation, this is accomplished by transmitting a pointer to the data structure containing all the needed packet fields). In the TEST-BED setting, instead, we need to send all the information necessary to rebuild, at the receiver side, the original NS-Miracle packet as it had been created and modified by the network simulator running on the transmitter host.

In Tables A.1–A.4, we report portions of a typical trace-files obtained for the feasibility tests illustrated in Fig. A.3, when data packets have been sent from the source to the destination with a packet inter-arrival period of 7 s. These files trace the packet transition from one layer of the protocol stack (identified by the corresponding NS-Miracle module) to the subsequent one; they are organized in six columns, with the following meaning: 1) packet origin (sent packet, "s" or received packet, "r"); 2) simulation time at which the packet transition between layers has occurred; 3) node ID; 4) source DESERT module; 5) destination DESERT module; 6) sequence number of the data packet. Table A.1 and Tables A.2–A.3 refer to the feasibility test (a), for the EMULATION and the TEST-BED setting, respectively. In the case of EMULATION, we collected all the traces in a single file (as commonly done when we run simulations using NS-Miracle); in the case of TEST-BED, instead, the traces have been created independently for each modem by the corresponding hosts (or ns2 process). In both cases, we can observe that all the considered protocol stacks have been correctly traversed by the exchanged packets, at the transmitter as well as at the receiver (the first packet is rendered using a red boldface font, both at the transmitter and at the receiver). Note also that the simulation time elapsed between the transmission of a given packet and its reception in the TEST-BED case increases with respect to the EMULATION case (from about 3 to about 5 s); this is due to two issues related to the TEST-BED case: the asynchronous simulation timers of the two nodes (during these TEST-BED tests, simulations have been launched manually and always starting from the receiver), and the increased complexity for mapping

Table A.1. *Trace file logged during the feasibility test (a) in Figure A.3 (EMULATION setting). Node 1 is the transmitter and Node 2 is the receiver.*

s 7.025577068 1 CBR UDP SN=1

s 7.025577068 1 UDP Static_Routing SN=1

s 7.025577068 1 Static_Routing IP SN=1

s 7.025577068 1 IP ARP_Tables SN=1

s 7.025577068 1 ARP_Tables ALOHA SN=1

s 7.025723219 1 ALOHA FSK_WHOI_MM SN=1

r 10.015380144 2 FSK_WHOI_MM ALOHA SN=1

r 10.015380144 2 ALOHA ARP_Tables SN=1

r 10.015380144 2 ARP_Tables IP SN=1

r 10.015380144 2 IP Static_Routing SN=1

r 10.015380144 2 Static_Routing UDP SN=1

r 10.015380144 2 UDP CBR SN=1

```
s 14.025732994 1 CBR UDP SN=2
s 14.025732994 1 UDP Static_Routing SN=2
s 14.025732994 1 Static_Routing IP SN=2
s 14.025732994 1 IP ARP_Tables SN=2
s 14.025732994 1 ARP_Tables ALOHA SN=2
s 14.025867224 1 ALOHA FSK_WHOI_MM SN=2
r 17.018894196 2 FSK_WHOI_MM ALOHA SN=2
r 17.018894196 2 ALOHA ARP_Tables SN=2
r 17.018894196 2 ARP_Tables IP SN=2
r 17.018894196 2 IP Static_Routing SN=2
r 17.018894196 2 Static_Routing UDP SN=2
r 17.018894196 2 UDP CBR SN=2
...
```

Table A.2. *Trace file for the feasibility test (a) in figure A.3 (TEST-BED setting) logged at node 1 (transmitter).*

s 7.012847900 1 CBR UDP SN=1

s 7.012847900 1 UDP Static_Routing SN=1

s 7.012847900 1 Static_Routing IP SN=1

s 7.012847900 1 IP ARP_Tables SN=1

s 7.012847900 1 ARP_Tables ALOHA SN=1

s 7.013003111 1 ALOHA FSK_WHOI_MM SN=1

```
s 14.013986111 1 CBR UDP SN=2
s 14.013986111 1 UDP Static_Routing SN=2
s 14.013986111 1 Static_Routing IP SN=2
s 14.013986111 1 IP ARP_Tables SN=2
s 14.013986111 1 ARP_Tables ALOHA SN=2
s 14.014182091 1 ALOHA FSK_WHOI_MM SN=2
...
```

Table A.3. *Trace file for the feasibility test (a) in figure A.3 (TEST-BED setting) logged at node 2 (receiver).*

r 12.013406992 2 FSK_WHOI_MM ALOHA SN=1

r 12.013406992 2 ALOHA ARP_Tables SN=1

r 12.013406992 2 ARP_Tables IP SN=1

r 12.013406992 2 IP Static_Routing SN=1

r 12.013406992 2 Static_Routing UDP SN=1

r 12.013406992 2 UDP CBR SN=1

```
r 19.020930052 2 FSK_WHOI_MM ALOHA SN=2
r 19.020930052 2 ALOHA ARP_Tables SN=2
r 19.020930052 2 ARP_Tables IP SN=2
r 19.020930052 2 IP Static_Routing SN=2
r 19.020930052 2 Static_Routing UDP SN=2
r 19.020930052 2 UDP CBR SN=2
...
```

Table A.4. *Trace file obtained for the feasibility test (b) in figure A.3 (TEST-BED setting) logged at node 2 (relay node).*

```
r 11.011646986 2 FSK_WHOI_MM ALOHA SN=1
r 11.011646986 2 ALOHA ARP_Tables SN=1
r 11.011646986 2 ARP_Tables IP SN=1
r 11.011646986 2 IP Static_Routing SN=1
s 11.011646986 2 Static_Routing IP SN=1
s 11.011646986 2 IP ARP_Tables SN=1
s 11.011646986 2 ARP_Tables ALOHA SN=1
s 11.011960030 2 ALOHA FSK_WHOI_MM SN=1
...
```

and de-mapping NS-Miracle packets into modem payloads (even though the delay introduced by this second issue is negligible with respect to the first one). Table A.4, instead, shows a portion of the trace file generated for the relay node 2 during the feasibility test (b). In this case, we can observe the behavior that we imposed for the routing protocol at the network layer: since node 2 must act as a relay, when it receives a packet for node 3 from node 1, this packet has been propagated only up to the network layer and then immediately sent down to be forwarded to its intended destination.

To move beyond just feasibility tests done in the laboratory, we also conducted a first field experiment in the Piovego channel which flows near our department. We put the transducers at a distance of 2 m from the bank, where the channel is around 80 to 90 cm depth, with a muddy bottom. We also decreased the supply voltage of the micro-modems to 12 V, in order to decrease their transmission ranges. Using the same script prepared for test (a) in Fig. A.3, we have performed several point-to-point transmissions (in the TEST-BED setting) varying the distance between nodes: i.e., we kept node 1 fixed and moved node 2 away, in steps of 3 m. For each tested link, we sent 50 packets, separated by a time interval of 4 seconds in both directions (i.e., from node 1 to node 2 and vice-versa).

Fig. A.5 shows the observed packet error rate (PER) as a function of the distance between the sender and the receiver, in both directions (note that, as expected, the observed

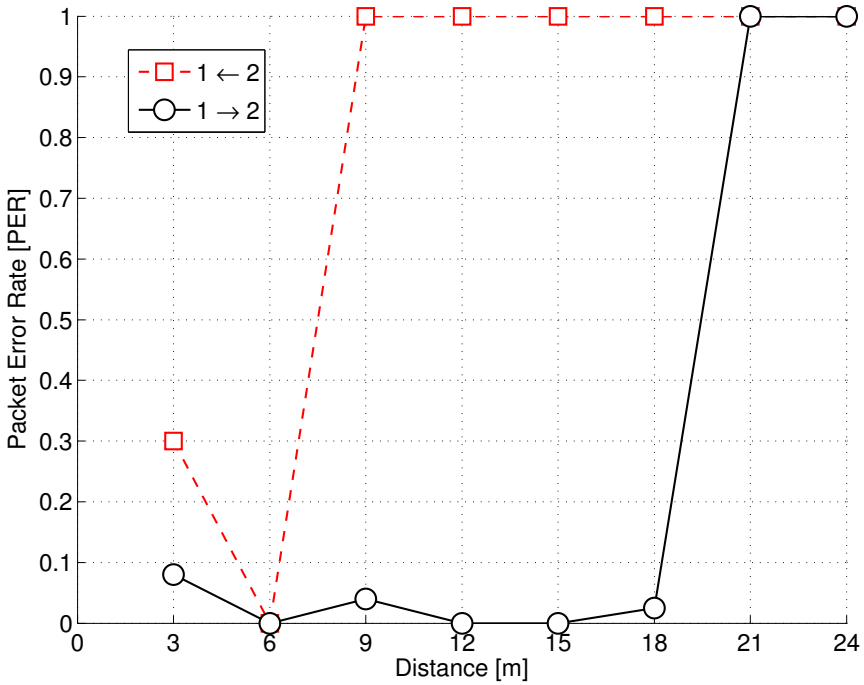


Figure A.5. *Packet Error Rate observed during the field experiment in the Piovego channel.*

underwater channel is not symmetric). Over the acoustic channel we sent FSK packets with an overall length of 32 bits, at a bit rate of 80 bit/s. Despite the adverse conditions (very shallow water, wind-generated surface ripples and noise, proximity to the bank, water turbidity), this test allowed us to verify the feasibility of the DESERT Underwater libraries, when used to drive real modem hardware.

A.5 Conclusions

Focusing on underwater applications, in this chapter we discussed the chance to evolve from the simulations to the realization of underwater network prototypes by interfacing real hardware devices with software network simulators. After having recognized this activity as a key effort to profitably develop and test real world applications, we presented the DESERT Underwater framework. This is a set of public C/C++ libraries based on the NS-Miracle [3] simulation software, developed at the University of Padova. The DESERT Underwater libraries are meant as a flexible and reliable tool to support the design and implementation of underwater network protocols.

We have listed and briefly described all the developed libraries currently available. These libraries are intended to be used jointly, possibly in many different combinations, thus realizing several protocol stacks for underwater networks. We also introduced the general interface designed to integrate the NS-Miracle network simulator with real hardware, as well as its specialized versions for two existing commercial modems.

Finally, we performed some preliminary tests to assess the feasibility of the DESERT Underwater libraries for network prototyping. These tests allow us to successfully perform single-hop as well as two-hop transmissions using the same code implemented in NS-Miracle for simulation purposes. We believe that our work, the public release of the DESERT Underwater libraries [2], and their future development, represent a fundamental step for the study of effective underwater network protocols, moving from the simulations to the real world.

Acknowledgments

This work has been supported by the Italian Institute of Technology within the Project SEED framework (NAUTILUS project). The authors gratefully thank Federico Beccaro, Achille Forzan, Moreno Zorzetto and Ivano Calabrese for their precious help in the organization and realization of the field experiments.

References

[1] "The Network Simulator - *ns-2*," Last time accessed: March 2012. [Online]. Available: http://nsnam.isi.edu/nsnam/index.php/User_Information

[2] "The DESERT Underwater libraries - *DESERT*," Last time accessed: March 2012. [Online]. Available: http://nautilus.dei.unipd.it/desert-underwater

[3] "The Network Simulator - *NS-Miracle*," Last time accessed: March 2012. [Online]. Available: http://dgt.dei.unipd.it/download

[4] "The World Ocean Simulation System - *WOSS*," Last time accessed: March 2012. [Online]. Available: http://telecom.dei.unipd.it/ns/woss/

[5] C. Petrioli, R. Petroccia, J. Shusta, and L. Freitag, "From underwater simulation to at-sea testing using the ns-2 network simulator," in *OES/IEEE Oceans*, Santander, Spain, 2011.

[6] C. Petrioli, R. Petroccia, and J. Potter, "Performance evaluation of underwater MAC protocols: From simulation to at-sea testing," in *OES/IEEE Oceans*, Santander, Spain, 2011.

[7] W. Liang, H. Yu, L. Liu, B. Li, and C. Che, "Information-carrying based routing protocol for underwater acoustic sensor network," in *Proc. of ICMA*, Takamatsu, Kagawa, Japan, Aug. 2007.

[8] N. Abramson, "Development of the ALOHANET," *IEEE Transactions on Information Theory*, vol. 31, no. 2, pp. 119–123, 1985.

[9] X. Guo, M. Frater, and M. Ryan, "A propagation-delay-tolerant collision avoidance protocol for underwater acoustic sensor networks," in *OES/IEEE Oceans*, Singapore, 2006.

[10] B. Peleato and M. Stojanovic, "Distance aware collision avoidance protocol for ad-hoc underwater acoustic sensor networks," *IEEE Communications Letters*, vol. 11, no. 12, pp. 1025–1027, 2007.

[11] A. A. Syed, W. Ye, and J. Heidemann, "T-Lohi: A new class of MAC protocols for underwater acoustic sensor networks," in *IEEE INFOCOM*, Phoenix, AZ, US, Apr. 2008.

[12] S. Azad, P. Casari, F. Guerra, and M. Zorzi, "On ARQ Strategies over Random Access Protocols in Underwater Acoustic Networks," in *OES/IEEE Oceans*, Santander, Spain, 2011.

[13] "S2C R 48/78 underwater acoustic modem." [Online]. Available: http://www.evologics.de/en/products/acoustics/s2cr_48_78.html

[14] "Woods Hole Oceanographic Institution," Last time accessed: March 2012. [Online]. Available: http://www.whoi.edu/

[15] "The Goby Underwater Autonomy Project - *Goby*," Last time accessed: March 2012. [Online]. Available: http://gobysoft.com/

[16] B. Liang and Z. Haas, "Predictive distance-based mobility management for PCS networks." in *IEEE INFOCOM*, New York, NY, US, Mar. 1999.

[17] L. Badia and N. Bui, "A Group Mobility Model Based on Nodes' Attraction for Next Generation Wireless Networks," in *IEE Mobility'06*, Bangkok, Thailand, Oct. 2006.

[18] "The National Marine Electronics Association - *NMEA*," Last time accessed: March 2012. [Online]. Available: http://www.nmea.org/

List of Publications

Journals

1. Saiful Azad, Paolo Casari, and Michele Zorzi, "The Underwater Selective Repeat Error Control Protocol for Multiuser Acoustic Networks: Design and Parameter Optimization," IEEE Transactions on Wireless Communications (submitted on August, 2012)

2. Saiful Azad, Paolo Casari, Marco Zanforlin and Michele Zorzi, "Underwater Delay-Tolerant Network (UDTN) Routing Protocol with Probabilistic Spray Technique," IEEE Transactions on Wireless Communications (in preparation)

3. Saiful Azad, Paolo Casari, M. A. Mottalib and Michele Zorzi, "Multi-path Minimum Interference Distance Vector Routing Protocol for Underwater Networks," IEEE Wireless Communications Letters (in preparation)

Conferences

1. El Hadi Cherkaoui, Saiful Azad, Paolo Casari, Laura Toni, Nazim Agoulmine, and Michele Zorzi, Packet Error Recovery in Multipath Underwater Networks using Reed-Solomon Codes, in Proc. MTS/IEEE OCEANS, Virginia, USA, Oct. 2012

2. Saiful Azad, Paolo Casari, Michele Zorzi, "Coastal Patrol and Surveillance Networks using AUVs and Delay-Tolerant Networking, in Proc. of MTS/IEEE OCEANS, Yeosu, South Korea, May 2012

3. Riccardo Masiero, Saiful Azad, Federico Favaro, Matteo Petrani, Giovanni Toso, Federico Guerra, Paolo Casari, Michele Zorzi, DESERT Underwater: an NS-Miracle-based framework to DEsign, Simulate, Emulate and Realize Test-beds for Underwater network protocols, in Proc. of MTS/IEEE OCEANS, Yeosu, South Korea, May 2012

4. Michael Goetz, Saiful Azad, Paolo Casari, Ivor Nissen, Michele Zorzi, Jamming-Resistant Multi-path Routing for Reliable Intruder Detection in Underwater Networks, in Proc. of ACM WUWNet, Seattle, Washington, USA, Dec. 2011

5. Saiful Azad, Paolo Casari, Federico Guerra, Michele Zorzi, On ARQ Strategies over Random Access Protocols in Underwater Acoustic Networks, in Proc. of IEEE/OES OCEANS, Santander, Spain, Jun. 2011

6. Saiful Azad, Paolo Casari, Chiara Petrioli, Roberto Petroccia, Michele Zorzi, On the Impact of the Environment on MAC and Routing in Shallow Water Scenarios, in Proc. of IEEE/OES OCEANS, Santander, Spain, Jun. 2011

7. Federico Favaro, Saiful Azad, Paolo Casari, Michele Zorzi, On the Performance of Unsynchronized Distributed MAC Protocols in Deep Water Acoustic Networks, WUWNet, Seattle, Washington, USA, Dec. 2011 (extended abstract) SDC agg.to novembre 2008

8. Riccardo Masiero, Saiful Azad, Federico Favaro, Matteo Petrani, Giovanni Toso, Federico Guerra, Paolo Casari, Michele Zorzi, DESERT Underwater: an NS-Miracle based framework to DEsign, Simulate, Emulate and Realize Test-beds for Underwater network protocols, 9th Italian Networking Workshop, Courmayeur, Italy, Jan. 11 13, 2012 (extended abstract)