Contents

Abstract

Underwater Acoustic Networks are an innovative field of study. They represent a completely new scenario for all the researchers and more and more people are interested in it.

In the following work I will introduce a new approach to the routing problem in Underwater Acoustic Networks (UANs). The existing literature covers the the medium access control, and link control layers of the ISO/OSI protocol stack very well, but comparatively less attention has been devoted to the Network Layer. In literature there are several routing techniques that can be transposed to UAN; however they are scenario-dependent, that is, techniques that work well only under particular assumptions. In UAN is very difficult to develop a general routing protocol. As a matter of fact shallow water, deep water, superficial waves, fixed nodes, mobile nodes, networks with or without sinks, nodes with or without location awareness (etc.) are scenarios that require *ad-hoc* protocols, in order to take full advantage of the environmental characteristics. What I am going to do is to define an algorithm that works in every connected network, without any knowledge of the signal propagation patterns or other environment-related features.. Furthermore, the protocol I developed could be used as term of comparison for future works.

The thesis is organized as follows. Section 1 contains a brief introduction about UAN, section 2 introduces some basic concepts underlying this work, such as network stacks and routing protocols, section 3 contains related work, section 4 contains considerations about the statistic of the underwater channel, section 5 describes the new protocol I developed, section 6 contains the results of the simulations, and finally sections 7 and 7 contain proposal for future development of SpreadUW and conclusions.

1 Introduction

1.1 Underwater Acoustic Networks

Underwater networking is a rather unexplored area although underwater communications have been experimented since World War II, when, in 1945, an underwater telephone was developed in the United States to communicate with submarines. Acoustic communications are the typical physical layer technology in underwater networks. In fact, radio waves propagate at long distances through conductive sea water only at extra low frequencies (30-300 Hz), which require large antennae and high transmission power. Optical waves do not suffer from such high attenuation but are affected by scattering. Moreover, transmission of optical signals requires high precision in pointing the narrow laser beams. Thus, links in underwater networks are based on acoustic wireless communications.

The traditional approach for ocean-bottom or ocean column monitoring is to deploy underwater sensors that record data during the monitoring mission, and then recover the instruments. This approach has the following disadvantages: real time monitoring is not possible, no interaction is possible between onshore control systems and the monitoring instruments, if failures or misconfigurations occur, it may not be possible to detect them before the instruments are recovered, the amount of data that can be recorded during the monitoring mission by every sensor is limited by the capacity of the on board storage devices.

Therefore, there is a need to deploy underwater networks that will enable real time monitoring of selected ocean areas, remote configuration and interaction with onshore human operators. This can be obtained by connecting underwater instruments by means of wireless links based on acoustic communications.

Major challenges in the design of underwater acoustic networks are: battery power is limited and usually batteries can not be recharged; the available bandwidth is severely limited; channel characteristics, including long and variable propagation delays, multipath and fading problems; high bit error rates; underwater sensors are prone to failures because of fouling, corrosion.[1]

Accordingly with the definition in [1]:

Underwater acoustic (UWA) networks are generally formed by acoustically connected ocean-bottom sensors, autonomous underwater vehicles, and a surface station, which provides a link to an on-shore control center. While many applications require long-term monitoring of the deployment area, the battery-powered network nodes limit the lifetime of UWA networks. In addition, shallow-water acoustic channel characteristics, such as low available bandwidth, highly varying multipath, and large propagation

[1] Quoted from: `http://www.ece.gatech.edu/research/labs/bwn/UWASN/`

> delays, restrict the efficiency of UWA networks. Within such an environment, designing an UWA network that maximizes throughput and reliability while minimizing the power consumption becomes a very difficult task.

The underwater acoustic channel has something in common with terrestrial radio channels and this is supported by Milica Stojanovic[2] from MIT, one of the expert in underwater acoustic networks. However, underwater propagation may be very complicated, and its study involves different variables. In [2] we can read:

> Path loss of an underwater acoustic communication channel depends not only on the transmission distance, but also on the signal frequency. As a result, the useful bandwidth depends on the transmission distance, a feature that distinguishes an underwater acoustic system from a terrestrial radio one. This fact influences the design of an acoustic network: a greater information throughput is available if messages are relayed over multiple short hops instead of being transmitted directly over one long hop. We asses the bandwidth dependency on the distance using an analytical method that takes into account physical models of acoustic propagation loss and ambient noise.

1.2 Objectives of this work

The goal of this work is to present a very general routing algorithm working in Underwater Acoustic Networks without any assumption on the network. The algorithm proposed works in all conditions, without any requirement or assumption, such as position awareness, depth awareness, fixed nodes, mobile nodes, dense network, sparse network, etc. This leads to an algorithm that can perform better in some scenarios and worse in others, but whatever the scenario is, if an optimal route exist, the algorithm will find it. Furthermore, the algorithm gathers several routes and makes it possible to choose the best route, according some pre-specified objective function, such as: the power consumption, delay, throughput, etc.

[2] Milica Stojanovic personal web page: `http://www.mit.edu/~millitsa/index.html`

2 Basic Notions

This section introduces the basic notions behind the design of the SpreadUW routing algorithm. 2.1 contains a brief introduction about the Underwater Channel, Subsection 2.2 contains a brief description of the Network Stack, Subsection 2.3 contains a concise description of a routing protocol and all its possible models.

2.1 Underwater Channel

The path loss in an Underwater Acoustic Channel depends not only on the distance between the source and the destination, as in terrestrial radio channels, but also on the signal frequency. The frequency-dependent component of the attenuation increases with frequency as well as with distance. This is why a shorter link offers more bandwidth than a longer one. For example, a transmission over 100 km can be performed in one hop, using a bandwidth typically 10 to 100Hz, or by relaying the information over 10 hops, each of which is 10 km long, but offers a bandwidth on the order of the order of some kHz. Hence, in exchange for a more complicated system of relays, a significant increase in information throughput can be achieved[3].

2.1.1 Attenuation

If an underwater acoustic link between two nodes at distance l that transmit at frequency f is established, the attenuation between them is:

$$A(l, f) = l^k a(f)^l \tag{1}$$

where:

- k is the spreading factor;
- $a(f)$ is the absorption coefficient.

The previous formula can be expressed in dB and using Thorp formula which gives $a(f)$ in dB/km for f in kHz;

$$10 \log a(f) = 0.11 \frac{f^2}{1+f^2} + 44 \frac{f^2}{4100+f^2} + 2.75 \cdot 10^{-4} f^2 + 0.003 \tag{2}$$

This formula is valid for frequencies above few hundred Hz, for lower frequencies, it is possible to use:

$$10 \log a(f) = 0.002 + 0.11 \frac{f^2}{1+f^2} + 0.011 f^2 \tag{3}$$

The absorption coefficient $a(f)$ is plotted in Figure 1[4].

[3] Example from [2]
[4] Image from [2]

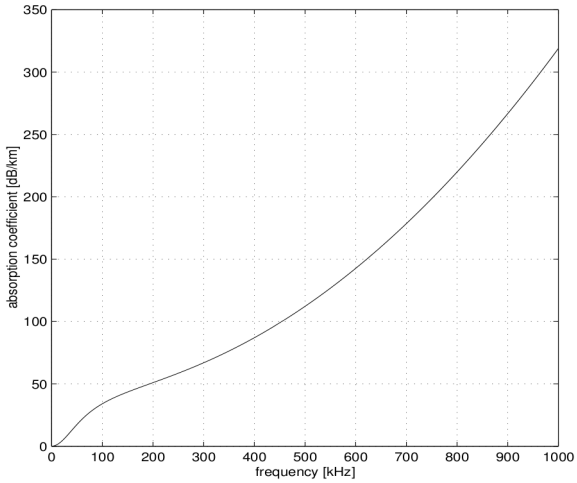

Fig. 1: Absorption coefficient $a(f)$ in [dB/km]

2.1.2 Noise

The noise in oceans is generated by four sources: turbulence, shipping, waves and thermal noise. It is possible to model the power spectral density (p.s.d.) of the four noise components in dB re μ Pa per Hz as a function of frequency in kHz [24]:

$$\begin{aligned} 10\log N_t(f) &= 17 - 30\log f \\ 10\log N_s(f) &= 40 + 20(s-0.5) + 26\log f - 60\log(f+0.03) \\ 10\log N_w(f) &= 50 + 7.5w^{1/2} + 20\log f - 40\log(f+0.4) \\ 10\log N_{th}(f) &= -15 + 20\log f \end{aligned} \tag{4}$$

Turbulence noise affects only very low frequency $f < 10$Hz. Ship noise instead affects the region 10Hz-100Hz and can be modeled with a factor between 0 and 1. Thermal noise is present only for $f > 10$kHz. The most significant contribution to the noise is given by the wind speed that affects the frequency region 100Hz-100kHz which is the operating region of acoustic transmission.

A graphical representation of the Noise in Underwater Channel is presented in Figure 2[5]

2.1.3 SNR

The Signal to Noise Ratio is the power ratio between a signal and the background noise, and often it is expressed using the logarithmic scale:

$$SNR_{dB} = 10\log_{10}\left(\frac{P_{SIGNAL}}{P_{NOISE}}\right) \tag{5}$$

[5] Image from `http://telecom.dei.unipd.it/media/download/302/`

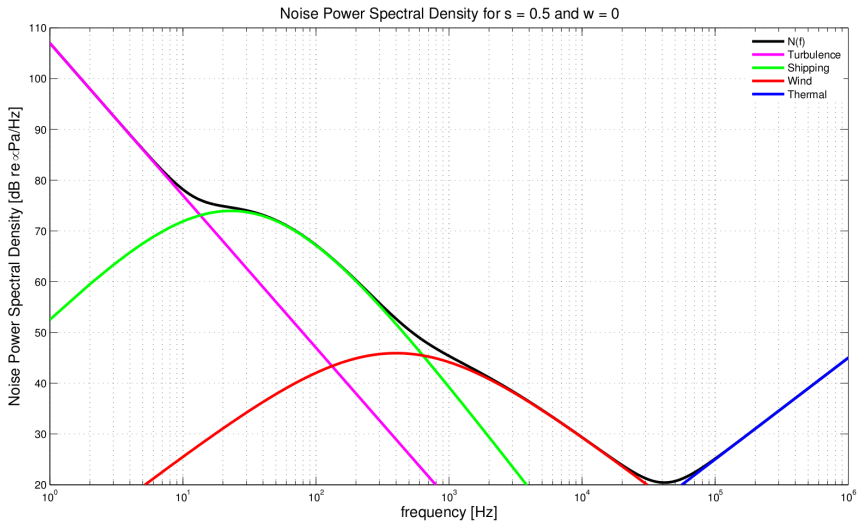

Fig. 2: Noise in Underwater Channel

Using the attenuation $A(l, f)$ and the noise p.s.d. $N(f)$ we can compute the SNR for two nodes at a distance l when the transmitted signal is a tone of frequency f and power P:

$$SNR(l, f) = \frac{P/A(l, f)}{N(f)\triangle f} \tag{6}$$

where $\triangle f$ is a narrow bandwidth arount the transmit frequency f.

$A(l, f)N(f)$ is the component frequency dependent. $1/A(l, f)N(f)$ is represented in Figure 3 for some distances.

2.2 Network Stacks

There are several ways to develop networks protocols. One can develop a protocol that deals with all the tasks, from the modulation of the signal through the physical channel to the management of the application level information. This approach can cope with particular situations, but usually is not recommended because it is very hard to manage and a simple modification/update can upset the entire code.

Instead of developing a unique protocol that performs all the tasks, it is possible to organize the flow of data in different layers. This approach yields several advantages: it is easier to handle errors and exceptions; it is very fast to adapt an entire stack to different conditions (for example to redesign the physical layer); with the same stack is possible to support different applications.

In literature two different stacks mainly exist: OSI from International Organization for Standardization (ISO/OSI) and TCP/IP, the protocol of the Internet.

ISO/OSI standards (ISO 7498[6]) documents are available from the ITU-T as the X.200-series

[6] ISO 7498: `http://www.iso.org/iso/catalogue_detail.htm?csnumber=14256`

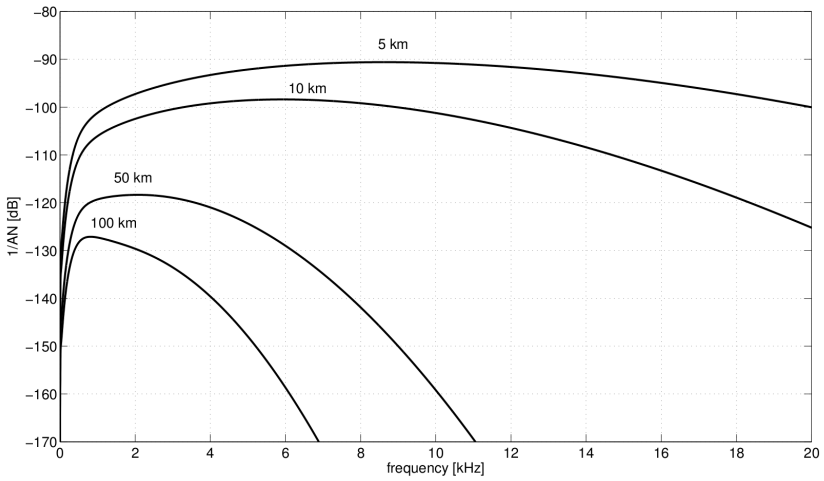

Fig. 3: The quantity $1/A(l, f)N(f)$ for some distance

of recommendations[7]. The standard focuses on a network architecture with seven layers, each one requesting information from the layer just below in the stack and providing information to the layer above. In Table 1 a brief list of these seven layers (a full description of each layer can be found in X.200).

	Layer	Function
7	Application	Network process to application
6	Presentation	Data representation, encryption and decryption, convert machine dependent data to machine independent data
5	Session	Interhost communication
4	Transport	End-to-end connections, reliability and flow control
3	Network	Path determination and logical addressing
2	Data Link	Physical addressing
1	Physical	Media, signal and binary transmission

Tab. 1: OSI Model

ISO/OSI is a very complete model, but in reality offers more than what is needed. An alternative and worldwide known stack is the TCP/IP one, on which is based the Internet.

In the TCP/IP model, protocols are deliberately not as rigidly designed into strict layers as in the OSI model. RFC 3439[8] contains a section entitled "Layering considered harmful". TCP/IP does recognize four broad layers of functionality which are derived from the operating scope of their contained protocols, namely the scope of the software application, the end-to-end transport connection, the internetworking range, and lastly the scope of the direct links to other nodes on the local network. Even though the concept is different from the OSI model, these layers are

[7] X.200 Recommendation: `http://www.itu.int/rec/T-REC-X.200-199407-I/en`
[8] RFC 3439: `http://www.ietf.org/rfc/rfc3439.txt`

often compared to the OSI layering scheme in the following way: the Internet Application Layer includes the OSI Application Layer, Presentation Layer, and most of the Session Layer. Its end-to-end Transport Layer includes the graceful close function of the OSI Session Layer as well as the OSI Transport Layer. The internetworking layer (Internet Layer) is a subset of the OSI Network Layer, while the Link Layer includes the OSI Data Link and Physical Layers, as well as parts of OSI's Network Layer[9].

In Table 2 a graphical comparison between the two stack models is presented.

TCP/IP	ISO/OSI
Application Layer	Application Layer
	Presentation Layer
	Session Layer
Transport Layer	Transport Layer
Internet Layer	Network Layer
Network Access Layer	Data Link Layer
	Physical Layer

Tab. 2: TCP/IP vs ISO/OSI

This thesis focuses on concerns a routing protocol developed in a stack based structure. This leads to a pure routing protocol that requires information from the layers below and provides a set of services to the upper layers. One of the best advantages is that it is possible to use this protocol with different layers, such as wireless networks or underwater networks, only by changing the underlying layers. Otherwise, it may be used with different applications without any significant changes.

2.3 Routing Protocols

Routing is the process of selecting paths in a network along which to send network traffic. Routing is performed for many kinds of networks, including the telephone network (Circuit switching), electronic data networks (such as the Internet), and transportation networks. In packet switching networks, routing directs packet forwarding, the transit of logically addressed packets from their source toward their final destination through intermediate nodes. The routing process usually administers the forwarding process based on routing tables which maintain a record of the routes to various network destinations. Thus, constructing routing tables, which are held in memory, is very important for efficient routing. Most routing algorithms use only one network path at a time, but multipath routing techniques enable the use of multiple alternative paths. Routing, in a more narrow sense of the term, is often contrasted with bridging in its assumption that network addresses are structured and that similar addresses imply proximity within the network. Because structured

[9] From: William Stallings, Data and Computer Communications, Prentice Hall 2006, and RFC1958: Architectural Principles of the Internet `http://tools.ietf.org/html/rfc1958`

addresses allow a single routing table entry to represent the route to a group of devices, structured addressing (routing, in the narrow sense) outperforms unstructured addressing (bridging) in large networks, and has become the dominant form of addressing on the Internet, though bridging is still widely used within localized environments[10].

In literature exists different routing aspects according to the following aspects[11]:

- Who decides the route:
  - *Source routing* is a way of moving a packet through a network in which the path is predetermined by the source or some device that tells the source about the path. The path information is placed in the packet. When the packet arrives at a switching device, no forwarding decision is necessary. The device looks at the path information in the packet to determine the port on which it should forward the packet[12]. This is the opposite of hop-by-hop IP routing, where packets contain only the destination address and routers at each junction in the network determine how best to forward the packet.
  - in contrast, in *hop-by-hop routing* protocols, where packets contain only the destination address and routers at each junction in the network determine how best to forward the packet.
- Static vs Adaptive Routing:
  - *static routing* is the type of routing characterized by the absence of communication between routers regarding the current topology of the network. In these systems, routes through a data network are described by fixed paths (statically). These routes are usually entered into the router by the system administrator. An entire network can be configured using static routes, but this type of configuration is not fault tolerant. When there is a change in the network or a failure occurs between two statically defined nodes, traffic will not be rerouted. This means that anything that wishes to take an affected path will either have to wait for the failure to be repaired or the static route to be updated by the administrator before restarting its journey[13];
  - *adaptive routing* describes the capability of a system, through which routes are characterized by their destination, to alter the path that the route takes through the system in response to a change in conditions. The adaptation is intended to allow as many routes as possible to remain valid (that is, have destinations that can be reached) in response to the change. People using a transport system can display adaptive routing. For example, if a local railway station is closed, people can alight from a train at a different

[10] From: Kurose, James E. and Ross, Keith W. (2004). Computer Networking, Third Ed. Benjamin/Cummings, and Doyle, Jeff and Carroll, Jennifer (2005). Routing TCP/IP, Volume I, Second Ed.. Cisco Press.
[11] From: Huitema, Christian (2000). Routing in the Internet, Second Ed.. Prentice-Hall.
[12] Quoted from: `http://www.linktionary.com/s/source_routing.html`
[13] From: TCP/IP Tutorial and Technical Overview (IBM RedBooks Series) `http://www.redbooks.ibm.com/redbooks/pdfs/gg243376.pdf`

station and use another method, such as a bus, to reach their destination. Another example of adaptive routing can be seen within financial markets. For example, ASOR or Adaptive Smart Order Router (developed by Quod Financial), takes routing decisions dynamically and based on real-time market events. The term is commonly used in data networking to describe the capability of a network to 'route around' damage, such as loss of a node or a connection between nodes, so long as other path choices are available. There are several protocols used to achieve this for example RIP[14] and OSPF[15]. Systems that do not implement adaptive routing are described as using static routing, where routes through a network are described by fixed paths (statically). A change, such as the loss of a node, or loss of a connection between nodes, is not compensated for. This means that anything that wishes to take an affected path will either have to wait for the failure to be repaired before restarting its journey, or will have to fail to reach its destination and give up the journey. For more info [3].

- Distance vector- vs Link-state algorithms:

  - *distance vector algorithms* use the Bellman-Ford algorithm[16]. This approach assigns a number, the cost, to each of the links between each node in the network. Nodes will send information from point A to point B via the path that results in the lowest total cost. The algorithm operates in a very simple manner. When a node first starts, it only knows of its immediate neighbors, and the direct cost involved in reaching them. Each node, on a regular basis, sends to each neighbor its own current idea of the total cost to get to all the destinations it knows of. The neighboring node(s) examine this information, and compare it to what they already 'know'; anything which represents an improvement on what they already have, they insert in their own routing table(s). Over time, all the nodes in the network will discover the best next hop for all destinations, and the best total cost. When one of the nodes involved fails, those nodes which used it as their next hop for certain destinations discard those entries, and create new routing-table information. They then pass this information to all adjacent nodes, which then repeat the process. Eventually all the nodes in the network receive the updated information, and will then discover new paths to all the destinations which they can still "reach". An example of distance vector algorithm is Routing Information Protocol (RIP)[17].
  - in *link-state algorithms*, each node uses as its fundamental data a map of the network in the form of a graph. To produce this, each node floods the entire network with information about what other nodes it can connect to, and each node then independently assembles this information into a map. Using this map, each router then independently determines the least-cost path from itself to every other node using a standard shortest paths algorithm such as Dijkstra's algorithm[18]. The result is a tree rooted at the current

[14] RIPv2 RFC 2453: `http://tools.ietf.org/html/rfc2453`
[15] OSPFv2 RFC: `http://www.ietf.org/rfc/rfc2328.txt`
[16] Brief description of Bellman-Ford algorithm: `http://www.csanimated.com/animation.php?t=Bellman-Ford_algorithm`
[17] RIPv2 is defined in RFC2453: `http://tools.ietf.org/html/rfc2453`
[18] Dijkstra's algorithm description: `http://www.cs.auckland.ac.nz/~jmor159/PLDS210/dijkstra.html`

node such that the path through the tree from the root to any other node is the least-cost path to that node. This tree then serves to construct the routing table, which specifies the best next hop to get from the current node to any other node. An example of distance vector algorithm is Open-Shortest-Path-First protocol (OSPF)[19]. A very good comparison between the two algorithm is presented in [4].

[19] OSPFv2 is defined in RFC2328: `http://tools.ietf.org/html/rfc2328`, the update for IPv6, OSPFv3, is defined in RFC5340: `http://tools.ietf.org/html/rfc5340`

3 Related Work

Considerable work has been done at the channel layer [3, 8] and at the physical layer [5, 6, 7]. A crucial aspect is the energy-efficient reliability [9, 10, 11] and very interesting features of the acoustic channel to keep into account are the bandwidth-distance and power-distance relationships in underwater networks [12].

In [12] there is a very complete comparison between different routing approaches, especially there are some very interesting results: "a routing algorithm should approach the optimal distance from below (*i.e.* using distances that are shorter than optimal)" and "a reasonable heuristic to find energy-efficient paths is to look for relays that [...] try to cover a hop distance close to the one that characterizes the optimal path".

The results highlight that there are two approaches very close to optimum for choosing the next hop: *bounded distance from above* and *bounded distance from below*, where the first one works fine only in a sufficiently dense network. In *bounded distance from above* each node forwards the packets to the closest node it can reach along the direction of the destination within a search area of given radius; in *bounded distance from below* each node forwards the packets to the farthest node that it can reach within the same area .

There exist several algorithms developed for wireless sensor networks like "Ad-hoc on-demand distance vector routing" (AODV) [13] that could be adapted for UAN. This adaptation is needed considering the different environment (delay, propagation time, bandwidth, ...) and some attempt was already done like the one in [14] in which the authors tried to modify AODV creating a protocol called AODV-B1. AODV-B1 is a reactive routing protocol with low overhead. In adapting AODV the following modifications are proposed. Firstly the route created are unidirectional, in order to reduce the overhead. Secondly the lifetime of a valid route changes: in AODV routes expire after a period of inactivity but this in underwater scenario could lead to a high overhead. Consequently in AODV-B1 a route is removed upon the receipt of error notification packets.

As the previous work confirms, direct application of routing algorithms for static networks to UAN with dynamic topologies requires routing information updates. This would lead to significant communication overhead, low energy efficiency and high end to end delays.

Another algorithm that can be modified in order to adapt it to UAN is SPIN [15]. In SPIN routing is initialized by sensor nodes. When a node want to send data, it first broadcast a description of the data. A neighboring node decides whether to request the data based on its local resources. Clearly also this protocols requires adaptation to the underwater environment, but it can be a good start point for future works.

Also Geographic Random Forwarding Geographic random forwarding (GeRaF) for ad hoc and sensor networks [16] offers good starting points to develop new algorithm for UAN. In GeRaF the authors propose a novel forwarding technique based on geographical location of the nodes involved and random selection of the relaying node via contention among receivers.

VBF [17] is a vector based forwarding protocol develop for underwater networks. In VBF the packet delivery is guided by the vector from a the source to the destination. Only those nodes the are within a range R of the vector will forward packets. These algorithm works well for underwater sensor networks but it requires location information of each sensor node.

DBR: Depth-Based Routing for Underwater Sensor Networks [18] is a protocol inspired by VBF

but that does not require full-dimensional location information of sensor nodes. It needs only local depth information. This protocols is developed for a scenario in which different nodes have to send data to the surface where sinks are located. The results show that DBR can achieve very high packet delivery ratios for dense networks. It is a quite general protocol because it requires just that the nodes know their depth but it not a general algorithm regarding the flow of data: from the bottom to the surface and in SpreadUW the sinks can be everywhere in the network.

Dynamic Source Routing (DSR) described in [19] is a very famous approach in routing for mobile ad hoc networks. DSR, a protocol specifically designed for use in mobile, ad hoc, wireless networks, suffers in the high latency of the underwater acoustic environment. The topology rate of change is too great compared with the acoustic communication latency: the topology changes more quickly than DSR can adapt. These are the reason that led the author of [20] to develop a new protocol called Location-Aware Source Routing (LASR) that uses two techniques to compensate for the high latency of acoustic links. The first is location awareness and the second is the use of a link quality metric to replace the simple hop-count metric used by DSR. The results of the article are very promising, LASR almost always outperforms blind flooding by a significant margin and outperforms DSR when the network topology is changing rapidly.

SpreadUW takes into account all the results from these works in order to create a very general routing algorithm without any assumption on the network.

4 Considerations About Channel Simulation

The network used in this thesis is static, this means that all the nodes have a fixed position in space. This to concentrate all the effort to the protocol performance, and leave the implementation of a mobility model for a future upgrade.

In order to simulate the channel two option were considered:

- to use a ray tracer like Bellhop[20];
- or to use empirical formulas like the ones in [8].

The ray tracer was used to obtain statistics of the channel, and once noted that these are stable, was designed a protocol which concept derives from these observations. The network was simulated using the empirical model described in 2.1. This model is deterministic, but the stability of the link measured by the ray tracer, makes the deterministic approach a good approximation of the more realistic case.

An important consideration about the channel is the statistics of the links. We tested a scenario with one fixed source and one fixed receiver with a non static environment that is of interest for simulations, and we observed that the channel has a statistical stable over time. In order to proof this statement, a program for Matlab was developed. This program uses bathymetry files, sound speed profile files and sea bed composition and parse them to create a configuration file for Bellhop. The retrieval of the bathymetry files is made by using the General Bathymetric Chart of the Oceans (GEBCO) database[21]. The sound speed profile comes from the World Ocean Atlas 2009 (WOA09)[22]. The seafloor composition are obtained from the NGDC "Deck41"[23].

The are considered was a region between the Norwegian Sea and the Batents Sea in the North Norway. The exact coordinates are reported in Table 3.

	First Coordinate	Second Coordinate
Longitude	10.34° E	10.28° E
Latitude	66.44° N	66.35° N

Tab. 3: Coordinates for the simulations

The source was positioned 100 meters deep at 10.34° E and 66.44° N. The destination was positioned in the direction 10.28° E, 66.35° N, 2 kilometers away from the source at the same depth. 2 kilometers is the typical distance from nodes in the network simulations of SpreadUW.

The sound speed profile point coordinates are in Table 4, a graphical representation of it is in Figure 4.

[20] BellHop website: `http://oalib.hlsresearch.com/Rays/index.html`
[21] GEBCO homepage: `http://www.gebco.net/`
[22] WOA09 homepage: `http://www.nodc.noaa.gov/OC5/WOA09/pr_woa09.html`
[23] Deck41 homepage: `http://www.ngdc.noaa.gov/mgg/geology/deck41.html`

	Sound Speed Profile Coordinates
Longitude	10.5° E
Latitude	66.5° N

Tab. 4: Sound Speed Profile Coordinates

In the area of the simulation, as reported in Deck41, the sea bottom is composed by clay, and the clay parameters for bellhop are reported in Table 5.

	cp_bottom	cs_bottom	density_bottom	alpha_bottom
clay	1510	95	1.51	[0.8 2.5]

Tab. 5: Sea Bottom Parameters

The goal of this simulation was to prove that with fixed position for the source and destination, and variations in the environment, the statistic of the link is stationary. In order to introduce perturbations in the environment, the sound speed profile has been progressively modified in each iteration. From Bellhop was extracted the Transmission Loss (TL) parameter. Two examples of the output are available in Figure 5 and 6.

Using the TL from BellHop and the equations in [21] for the noise, the SNR between the source and the destination was computed. Some results are shown in Table 6.

Iteration	SNR (in dB)	Variance (in dB^2)
1	45.3881	1.2205
2	45.4395	0.5075
3	45.3638	1.8438
4	45.4470	0.6411
5	45.4145	0.8241

Tab. 6: SNR measured

The results are very interesting because all the values are very close.

Were carried out other simulations in different scenario:

- an area close to the Coral Reef in Australia (decreasing bathymetric profile);
- and one in the Caribbean Sea (increasing bathymetric profile).

With different thermal conditions (summer and winter) and different depths for the source and the destination. The results are similar in all the scenarios. This justifies, with a good approximation, the initial hypothesis: it can be appropriate to avoid refreshing the routes too frequently, because the quality of links is almost static.

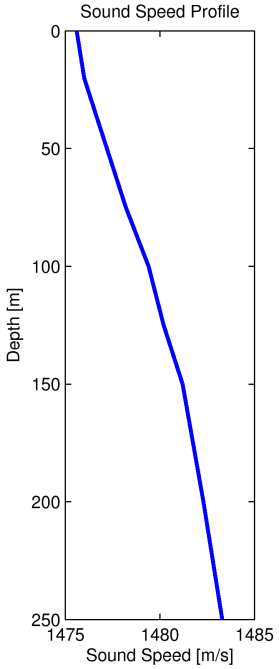

Fig. 4: Sound Speed Profile

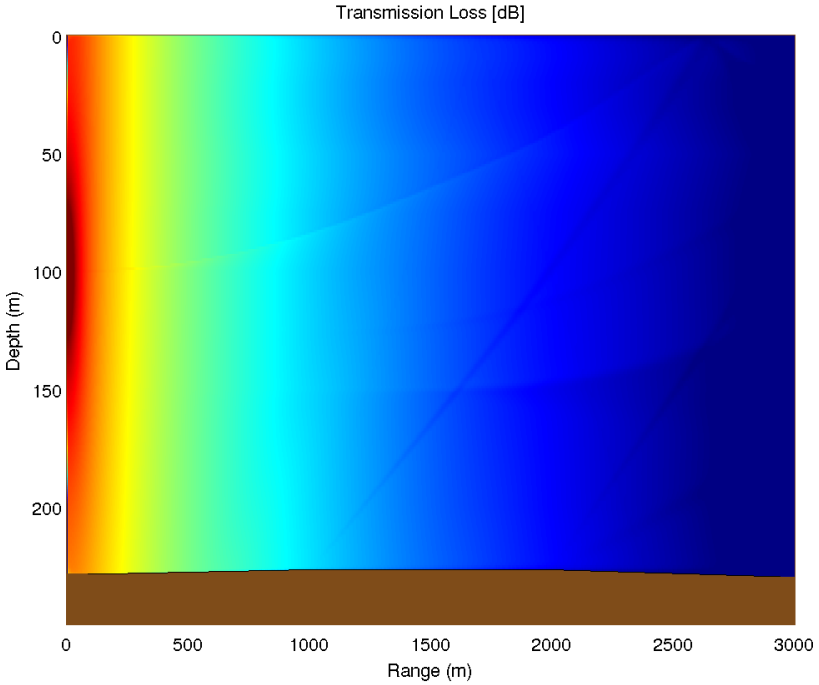

Fig. 5: Bellhop output, first run

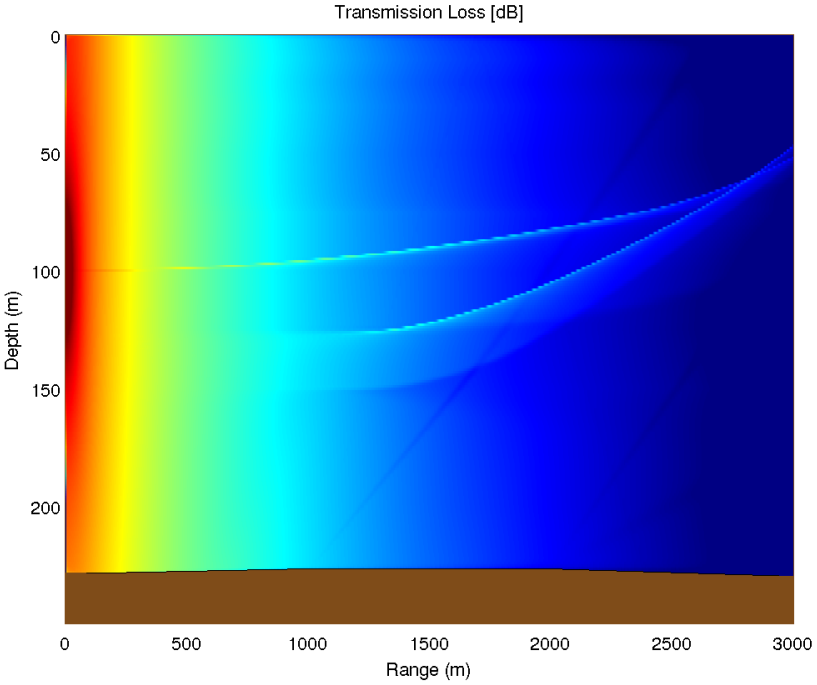

Fig. 6: Bellhop output, second run

5 SpreadUW Routing Protocol

In this section I am going to present SpreadUW protocol in detail.

5.1 Network Architecture

There are no assumptions on the nodes and sinks, which can be both fixed and mobile, or have different speed. There is no difference between shallow water and deep water, because, for the channel simulation, SpreadUW uses empirical formulas developed in [21]. Planned for the future is a version of SpreadUW that uses for the physical layer and the channel a framework such as WOSS[24] that uses a ray tracer model like BellHop[25].

SpreadUW is a pure routing protocol which does not take care of the layers below. As MAC protocol was used a modified version of ALOHA, called CSMA/ALOHA which permits to the node to sense the carrier before the transmission. The data modulation in the channel is assigned to the BPSK scheme. More information about physical an MAC layers in NS-Miracle is available in [22].

As already mentioned, the nodes have no knowledge of the network properties and are completely blind about their position. Even though the protocol was tested in configurations with multiple sinks, the results presented here concern only the case with one sink.

5.2 Protocol Overview

This subsection starts from an high level description of SpreadUW, describing its basic principles, and continues by introducing some specific characteristics.

SpreadUW is mainly a reactive and source routing protocol which has been slightly modified. The reasons why a reactive and source routing protocol is the best choice for a network with mobile nodes have already been described in 3. The protocol is reactive since it can build a special packet called *path request*, if a node has to transmit data. This special packet will be propagated through the network avoiding flooding. At the end of the process, if the node will receive at least one answer it will start to send packets to the closest sink following the best path. The reactive paradigm was modified as follows: when the other nodes send back the *path answer* they will keep some information about the path for a while, and if they have to transmit some data just after this information, they can use it instead of requesting a new path. The protocol is a source routing protocol because the node that has to transmit will embed all the paths to the destination in the header of the packet.

SpreadUW considers two entities: sinks and nodes.

Sinks are passive entities because their only task is to:

- send periodically probe messages in broadcast, in order to advise the nodes in their range about their presence;
- receive data packets.

No ack system is implemented in sinks because this is left to the upper layers.

[24] WOSS project homepage: `http://telecom.dei.unipd.it/ns/woss/`
[25] BellHop website: `http://oalib.hlsresearch.com/Rays/index.html`

Nodes are the main characters of SpreadUW. They can:

- send data;
- ask for paths;
- answer for path requests;
- act as relay;
- notify route problems such as the lost of a sink or a next hop.

Two new packet types were also defined with their respective header:

- one packet called *path establishment* concerning all the control information like paths, probes, messages;
- and one called data that contains the *data* to transmit.

The path search mechanism in SpreadUW is very simple. Every sink periodically sends a *probe advertisement* in broadcast in the channel; if a node can listen the message, that means that it can also send messages directly to the sink and it does not require further hops. This special node is an end node, a node with hop count 1. This node in the future will take care to retransmit packets directed to the sink. A node will be an end node only for a pre-defined period; if in this period it does not receive any *probe advertisement* this means that the sink moved out its range.

In SpreadUW when a node wants to transmit, it checks it if has a valid route to the sink; if so it creates a *data packet* adding to the header the hop list necessary to reach the sink. It is has not a valid route, it creates a *path establishment* packet setting up the option field as *path request*, and then sends in broadcast the message. When a node receives a *path establishment packet* with the option field set as *path request*, it checks if it has already processed the packet; if so it drops the packet in order to avoid flooding, if not it adds it is ID to the header. In this way, a list of hops to reach the destination is automatically created. This process continues until the packet reach a node with hop count 1. The node with hop count 1 knows that it can reach directly the sink. It changes the option to *answer* and it sends back the packet, using the hop list contained in the header. The packet will cross the entire network reaching the node that requested the path. An additional task that a relay node can carry out is the following: when a node reads an *answer* it can add the information contained in the header to its routing table. Doing this it can obtain a path to the sink without any effort.

If a path to the sink exists, the source at some point will receive the first path *answer*. This probably is the fastest path to the sink, and the node can use it as path to send data. Usually the first path received is the one with less hops, but that does not mean that it is the best one. Nevertheless, the node continues to listen to the channel for other answers. If it considers a new

path as the best one, it will change its routing table, and all the new *data* packets will be sent with the new route.

When a node with hop count 1 receives a *data* packet, it forwards it directly to the sink. If a node detects that the next hop is no more available, it will acknowledge it to the previous nodes, so that the acknowledgment will reach the source. In that case the source can decide between sending a new *path search* or using another path. In fact when the source receives multiple *path answer* packets, it can keep a queue of paths, indexed from the best to the worst path.

5.3 Protocol Design

As described in 6 SpreadUW was implemented in a complete network simulator called NS-Miracle. The usage of a simulator lead all the efforts to develop the SpreadUW algorithm and not the other protocols used by it. SpreadUW was developed in C++ in order to obtain the best performance and it was created to have as little overhead as possible. The only requirement in NS-Miracle is that the algorithm needs an IP module below, and therefore is encapsulated in an IP header.

SpreadUW uses three packet types, one for *probe* messages illustrated in Table: 7, one for *path request* illustrated in Table: 8 and one for *data packets* Table: 9.

The *Probe Header* is very minimal, it contains just the header length and the IP of the sink. The IP field is redundant because it is contained also in the IP header. It was placed in the header just to facilitate procedures without using the piece of information in the IP level.

Header Legth	IP of sink associated

Fig. 7: Probe Header

Path Request header contains a Packet Type option that can be *search* or *answer* (used to define the "direction" of the message), the header length, the Metric option used by the metric algorithm (described in Section 5.4), the IP of sink associated and the list of hops. In this header, the sink IP is necessary because if a node is in the range of two or more sinks, in this field it has to write which of the two is the designated one. List of hops is simply a list with the IPs necessary to reach the last node.

Packet Type	Header Legth	Metric	IP of sink associated	List of Hops:	IP hop 1	IP hop ...	IP last hop

Fig. 8: Path Request Header

Data packet is very similar to the *Path Request* header but in this case it does not contain an option field.

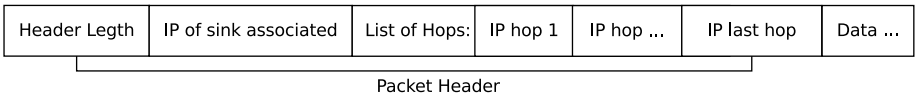

Fig. 9: Data packet Header

The choice to use ad IP to address the hosts maybe is not the best choice in UAN. An IPv4 is 32 bit long, and 32 bit of overhead are a not insignificant. As a matter of fact the number of nodes is usually less than 100, and to identify 100 nodes only 6 bit are needed, which are five time less than 32. The choice of IP depends on simulation issues: a solution with custom ID would required a complete new Network Layer and also if the IP module is removed, various application level protocols in NS-Miracle based on IP will not operate. A future work should include this modification in order to reduce the overhead of SpreadUW.

SpreadUW routing protocol for a node is based mainly on four procedures:

- the first one that processes a packet when it is received by the node (Algorithm 1);
- the second one is used when the node has to process a *path search* packet (Algorithm 2);
- the third one is similar to the second but is used for *path answer* packets (Algorithm 3);
- the last one is used by the nodes to forward data packets (Algorithm 4).

Algorithm 1 Receive Packet

```
1: if packet = probe then
2:    hopcount ← 1
3: else if packet = pathrequest then
4:    if option = search then
5:       call process path search
6:    else if option = answer then
7:       call process path answer
8:    end if
9: else if packet = data then
10:   if hopcount = 1 then
11:      send data to sink
12:   else
13:      call forward data packet
14:   end if
15: end if
```

Algorithm 2 Process Path Search

```
1: if packet already processed = true then
2:    drop packet
3: else
4:    if hopcount = 1 then
5:       set option to answer and send back the packet
6:    else
7:       add my IP in the header
8:       send in Broadcast the packet
9:    end if
10: end if
```

Algorithm 3 Path Answer

```
1: if destination = my IP then
2:    evaluate path
3: else
4:    if my IP in the header then
5:       forward back the packet
6:    else
7:       drop the packet
8:    end if
9: end if
```

Algorithm 4 Forward Data Packet

```
1: if I an the right next hop then
2:    if hopcount = 1 then
3:       send data to the sink
4:    else
5:       forward to the next hop
6:    end if
7: else
8:    drop the packet
9: end if
```

5.4 Discussion on Routing Metrics

Metrics are used to decide the best path to reach the destination. A metric can be considered the core of the routing algorithm. Once a node receives a list of path answers, it has to decide which is the path to use in order to transmit data . This decisio is made using routing metrics.

It is almost impossible to decide which is the best metric a priori. In other words, there is no generally "best" metric, as the concept of best depends on several constraints, usually dictated by the scenario, network topology, application requirements and hardware limitations.

SpreadUW is a generic routing algorithm in all its aspects, including the metrics. SpreadUW offers by default three metrics: Lowest Hop Count, Best Average SNR and Lowest Average Load, which performs quite well in all environments. In addition to these three metrics, SpreadUW has a system that allows the user to develop very easily a new metric.

5.4.1 Lowest Hop Count

As the name suggests, the Lowest Hop Count is the simplest metric that can be used. When a node receives a list of paths, it will elect as the best the one with the less number of hops. If two or more paths have the same minimum number of hops, the node will keep as the best path the one processed earlier.

Pros: the algorithm is very simple, and the nodes do not need additional information. In the packet header, when a node processes a path request, it simply increases the metric field by one.

Cons: the algorithm does not perform very well because it assigns the same quality to each link, and obviously this is not a good assumption. Trying to minimize the Hop Count, this algorithm prefers the longest paths, which are generally the ones with lower SNR. Further information can be found in [23].

5.4.2 Best Average SNR

The Best Average SNR is a simple but very effective metric.

When a node receives a path request to process, it knows the power level at which it received the packet, called P_{rx}, and it also knows the power of the noise of the underwater channel, called P_{noise}. Given these two values, it is straightforward to calculate the $SNR_{n,m}$ of the link (n, m) where n is the previous node and m is the current node.

$$SNR_{n,m} = \frac{P_{rx}(n,m)}{P_{noise,m}}$$

A node that receives a path request adds to the quality field in the header its own SNR. When the packet will reach the destination, the quality field will contain the cumulative SNR of the path, indicated as $\sum_i SNR_i$. $\sum_i SNR_i$ is equal to the sum of all the single hops SNRs. The source can compute the mean SNR of a path α (SNR_α) by means of the following formula:

$$SNR_\alpha = \frac{\sum_i SNR_i}{numer\,of\,path\,links} \tag{7}$$

5.4.3 Lowest Average Load

In the previous section Best Average SNR was presented. The Best Average SNR is a very

effective technique to choose the path: it selects the path with the best signal to noise ratio, and this choice ensures that once a packet is sent, it follows a path with the higher mean quality. However there is an aspect that this metric does not take into consideration: the load of a path. The load of a path can be defined as the sum of the loads of the single nodes. The load of a single node is a metric that tries to evaluate whether or not a node can to send packets correctly to the following hop. It is important to notice that, while Best Average SNR depends only on the performance of the link towards the following relay, the Lowest Average Load depends on the load of the next relay, and thus captures a broader view of the relay performance.

In particular SpreadUW implements a mechanism that depends on: the number of packets processed, the number of packets correctly sent and the number of status packets received, all of them considered in a fixed period of time, usually one minute. As already mentioned, this metric needs status packets, and consequently it would not be useful in a network without them.

Each node contains three counters:

- a counter that updates the number of packets processed in the last 60 seconds. The processed packets include all the packets generated by the node itself and the packets that it received and has to forward;
- a counter that updates the number of packets transmitted in the channel. This number is always less than the first because sometimes a node can receive packets with errors (discarding them), or alternatively the underwater channel is not free and the internal buffer is full, so that the node has to discard packets, etc.;
- a counter that updates the number of status packets received in the last minute.

To evaluate the load, a node i uses:

$$\begin{aligned} load_i &= \alpha(pkts\,processed) + (1-\alpha)(pkts\,errors) \\ &= \alpha(pkts\,processed) + (1-\alpha)(pkts_{tx} - acks) \end{aligned} \tag{8}$$

The load of a node i is the number of packets processed in the last minute multiplied by a factor α plus the number of errors in transmission. The number of errors in transmission is the number of packets transmitted minus the status packets received, all multiplied by $(1-\alpha)$.

α is a variable that can be adapted in order to penalize more or less the fact that a node does not receive status packets. In SpreadUW, by default, the value of α is less than 1/3. With values smaller than 1/2, the algorithm penalizes the nodes that do not receive status packets. This is a desirable behavior because the fact that a node has to process many packets can be a symptom of load, but if it can send all the packets correctly to the next hop this is not a problem. The problem arises when a node sends packets to the next hop and it does not receive status packets.

When a node processes a path request with a less load metric, it simply adds its load value to the quality field in the header.

A source node that receives a path answer has to calculate the average load of the nodes in a path as follows:

$$average\,load = \frac{\sum_i load_i}{numer\,of\,nodes\,in\,the\,path} \tag{9}$$

Section 6.3.4 contains a comparison between the three metrics implemented in SpreadUW.

6 Performance Evaluation

The chapter presents the simulator used to implement the protocol, the scenario, the parameters of the configurations and the results of the tests.

6.1 Simulators: NS2 and NS-Miracle

In order to test the protocol, a network simulator with underwater channel simulation capability is needed. The final choice was NS-Miracle, a Multi-InteRfAce Cross-Layer Extension library for the Network Simulator 2.

6.1.1 Network Simulator 2 (NS2)

Ns is a discrete event simulator targeted at networking research. Ns provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. Ns began as a variant of the REAL network simulator in 1989 and has evolved substantially over the past few years. In 1995 ns development was supported by DARPA through the VINT project at LBL, Xerox PARC, UCB, and USC/ISI. Currently ns development is supported through DARPA with SAMAN and through NSF with CONSER, both in collaboration with other researchers including ACIRI. Ns has always included substantial contributions from other researchers, including wireless code from the UCB Daedelus and CMU Monarch projects and Sun Microsystems[26].

6.1.2 NS-Miracle

NS-Miracle is a set of libraries designed to enhance the functionalities provided by the Network Simulator ns2. It provides an efficient and embedded engine for handling cross-layer messages and, at the same time, enables the coexistence of multiple modules within each layer of the protocol stack. For instance, multiple IP, link layers, MACs or physical layers can be specified and used within the same node. The NS-Miracle framework facilitates the implementation and the simulation of modern communication systems in ns2; moreover, due to its modularity, the code will be portable, re-usable and extensible[27].

The official web page for NS-Miracle:

`http://dei.unipd.it/wdyn/?IDsezione=3966`

Ns-Miracle library can be found here:

`http://telecom.dei.unipd.it/ns/miracle/doxygen/`

and an installation guide here:

`http://telecom.dei.unipd.it/ns/miracle/ns2-33-miracle-mini-HowTo.html`.

6.2 Scenarios and Parameters

Different and highly variable configurations were used. For each test in section 6.3 the parameters will be mentioned, but in generally the configuration of the network are the following:

[26] From: `http://nsnam.isi.edu/nsnam/index.php/User_Information`
[27] From: `http://www.dei.unipd.it/wdyn/?IDsezione=3966`

Environment Area	10000 meters wide 10000 meters long 200 meters depth
Number of nodes	32
Number of sinks	1
Simulation Time	100000 seconds
Iteration for each test	20
CBR packet size	1250 kbyte

Tab. 7: Simulations general settings

The parameters of the modem have been set accordingly to the specifications of the real modem WHOI Micro-Modem, a small-footprint, low-power acoustic modem based on the Texas Instruments TMS320C5416 DSP[28]:

Carrier Frequency	25kHz
Bandwidth	5kHz
Transmission Power	150-160 dB
Bit rate	4800 bps
Modulation scheme	BPSK

Tab. 8: Micromodem parameters

In order to realize a real network stack, other layers are needed. In the case of this paper the following modules from NS-Miracle were used:

Layer Level	Name of Module	Brief description
Application	CBR	Module used to transmit packets with constant bit rate.
Transport	Port	Module used to multiplex several streams coming from the layers below.
Network Routing	SpreadUW	General Routing Algorithm for UAN.
Network	IP*	Module that reflects the IP behavior.
Data Link	MLL	Module for ARP-resolve.
	Aloha-CSMA	Module that replicates Aloha and CSMA protocols behavior.
Physical	BPSK	Module for physical modulation.
Channel	Underwater Channel	Module the represents the behavior of an Underwater Channel.

Tab. 9: NS-Miracle Stack

[28] More information about micromodem WHOI at `http://acomms.whoi.edu/umodem/`

*The IP Module contained in NS-Miracle was extended in order to implement some extra functions that were not implemented before (for example now the IP Module handles the TTL).

6.3 Results

This section will present the results of the simulations. All the simulations consider environments with the settings in Tables 7 and 8. The packets size considered are 125, 500 and 1250 byte long. One sink, with a modulation bit rate of 4800 bps and with these packet's lengths, is able to support the following periods of transmission:

$$period_1 \geq \frac{pktsize * number\,of\,nodes}{modulation\,bit\,rate * number\,of\,sinks} = \frac{125 * 8 * 32}{4800 * 1} = 6,667s \tag{10}$$

$$period_2 \geq \frac{500 * 8 * 32}{4800 * 1} = 26,667s \tag{11}$$

$$period_3 \geq \frac{1250 * 8 * 32}{4800 * 1} = 66,667s \tag{12}$$

Where each period corresponds to different packets size. The value 8 is needed because the modulation bit rate is in bit and the packet size in byte. The values 7, 27 and 67 seconds are the minimum periods that will assure that every packet can hypothetically reach the destination. The word hypothetically is mandatory because the channel is not free of error and the efficiency of the Aloha protocol is less than 100%.

Hearafter λ will correspond to the traffic generation frequency measured in packets sent in a minute by each node, and it corresponds to $\frac{1}{CBR_{period}}$, so:

$$\lambda_1 = \frac{1}{period_1} = \frac{1}{6.667}$$

$$\lambda_2 = \frac{1}{period_2} = \frac{1}{26.667}$$

$$\lambda_3 = \frac{1}{period_3} = \frac{1}{66.667}$$

When specific parameters will be used they will be mentioned.

There are mainly four blocks of simulations:

- Section 6.3.1 will present the performances of the network using a fixed metric (in this case Best Average SNR) and without the status packets mechanism;
- Section 6.3.2 will present the performances of the network using a fixed metric (in this case Best Average SNR) and with the status packets mechanism;

- Section 6.3.4 will present a comparison between the three different metrics implemented in SpreadUW to choose the best path: Lowest Hop Count, Best Average SNR and Less Load;

6.3.1 Simulation Without Atatus Packets

As the title of this section suggests, the nodes do not use any status packet system. This means that there is less network traffic, but also that the only way that the nodes have to discover "broken paths" is by using a timeout mechanism. In all the following data, Best Average SNR (Section 5.4.2) is the metric chosen to decide the best path.

Figure 10 contains the throughput per node (in bits/s) as a function of λ (packets/minute per node) for different values of packet size. The charts clearly show that for each packet size there is a value of λ that returns a maximum throughput. With low packet sizes high throughput are achievable for high values of λ, and with an increase of the packet size, the value of λ necessary to obtain high throughput decreases. All the curves have a similar shape: a first section where they increase, a couple of peaks with high values of throughput and a section where they decrease. As can be expected, higher values of throughput can be obtained with a high packet size and low values of λ. For example, the best throughput for a packet 125 bytes long is obtained with $\lambda = \frac{4}{3}$ and is 7.05 bps, instead, with a packet size of 1250 bytes, the best throughput is obtained with $\lambda = \frac{4}{3}$ and it is ~ 14 bps.

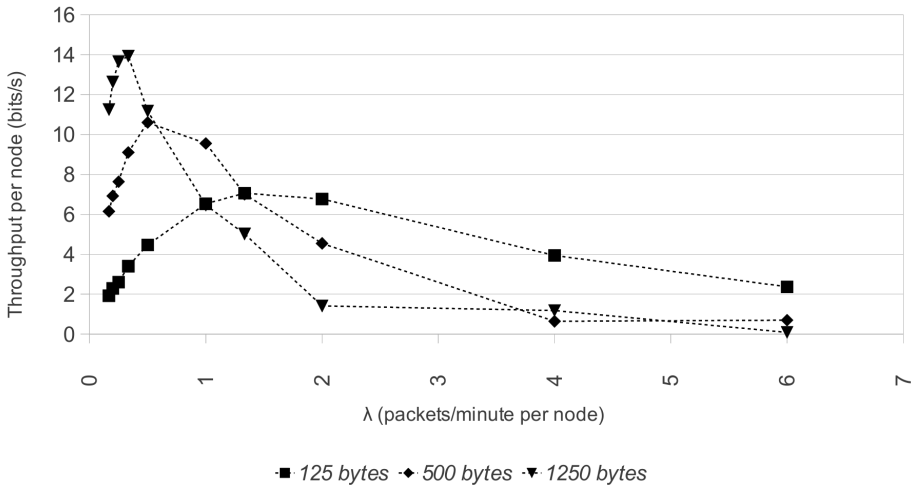

Fig. 10: Mean throughput per node (in bits/s) as a function of λ (packets/minute per node) for different values of packet size. Configuration w/o status packets and with SNR as metric.

Figure 11 contains the packet probability of successful transmission (P_{succ}) as a function of λ (packets/minute per node) for different values of packet size. Also in this graph the results are expected: lower values of packet size mean higher P_{succ} considering the same λ. This is obvious

because shorter packets require less time to be transmitted and this means that the channel is free for more time. The shapes of the curves are very similar: with low λ the P_{succ} is high, with an increasing value of λ the P_{succ} decreases very fast. P_{succ} under 0.1 corresponds to the following values:

Packet Size	λ for a P_{succ} lower than 0.1
125	1
500	4/3
1250	4

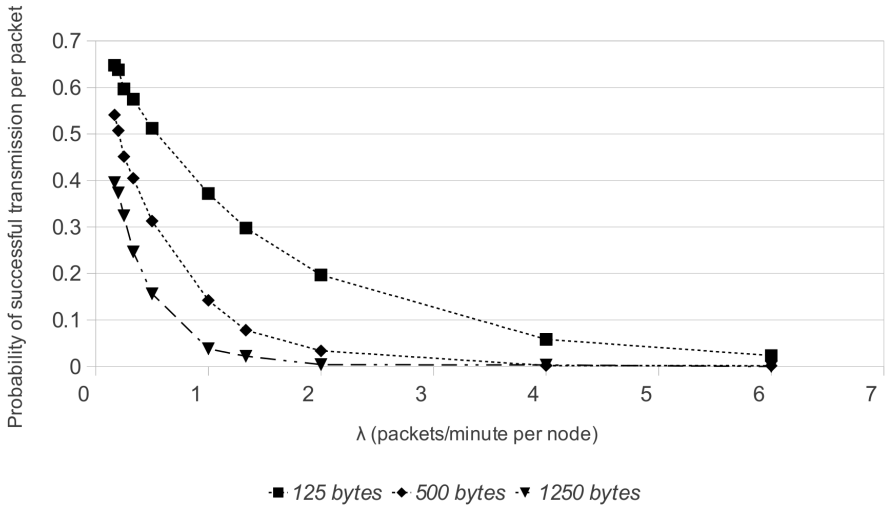

Fig. 11: Packet probability of successful transmission as a function of λ (packets/minute per node) for different values of packet size. Configuration w/o status packets and with SNR as metric.

In an environment with a sink positioned in the center of an area 10 km wide, 10 km long and 200 meters deep, the maximum distance from a node to the sink is $\sqrt{(10000/2)^2 + (10000/2)^2 + (200/2)^2} \simeq$ 7071.77 meters; with a $P_{tx} = 150dB$ a minimum number of hops equals to 3 is needed to reach the destination (this happens if the less hop metric is used). In this kind of network there will be nodes with hop count 1 (directly connected with the sink), nodes with hop count 2 and nodes with hop count 3 or more. Is the throughput the same in each case? As one can expect this is not true: a packet sent by a node with hop count 1 is sent directly to the sink. On the contrary, a packet sent by node with hop count higher that 1 has to send it to multiple repeaters, each one adding a probability of collision.

So far I have talked about mean throughput, which does not necessarily imply that it is distributed uniformly to all the nodes. Usually is the opposite: a very high mean throughput implies that the nodes with low hop count cannibalize the channel, completely preventing other nodes the possibility to transmit.

Which is the best value for λ that assures a compromise between throughput and uniform distribution of it? The answer is contained in the following results. Figures 12, 13 and 14 contain the mean throughput per node (in bits/s) as a function of λ (packets/minute per node) for different values of Hop Count and Packet Size.

For example take into account packets 125 bytes long. From Figure 10 we know that the value for λ that returns the best mean throughput per node is $\frac{4}{3}$, but as Figure 12 clearly shows, the nodes with Hop Count 1 have a throughput $\sim$ 3 times higher than the nodes with hop count 2, and $\sim$ 16 times higher than the nodes with hop count 3. In a well balanced network this is not a desirable behavior, because generally the throughput between the nodes should be balanced without decreasing too much the overall throughput. If we consider $\lambda = \{1, \frac{1}{2}\}$ we can obtain a network much more fair. In the case of $\lambda = 1$ the overall mean throughput is 93% the maximum mean throughput, but the ratio of the throughput between nodes with hop count 1 and 2 is 1.27 and instead of 3, and the one between nodes with hop count 1 and 3 is 1.91 instead of 16. In the case of $\lambda = \frac{1}{2}$ we obtain a network more balanced but with a lower throughput. The overall mean throughput is 68% the throughput for $\lambda = \frac{4}{3}$ but the ratio between the nodes with hop count 1 and 2 is 1.01, and the ratio between nodes with hop count 1 and 3 is 1.14 instead of 16.

As it is possible to understand in Figures 12, 13 and 14 there are particular values for λ that give a well balanced network, but in some cases this is not always achievable. For example in Figure 13 for a packet size of 500 byte it is very hard to find a value of λ that balances the throughput of the nodes with different hop counts. In that case a value of $\lambda = \frac{1}{2}$ is the value that best balances the throughput, but the ratio between the nodes with different hop count is still high.

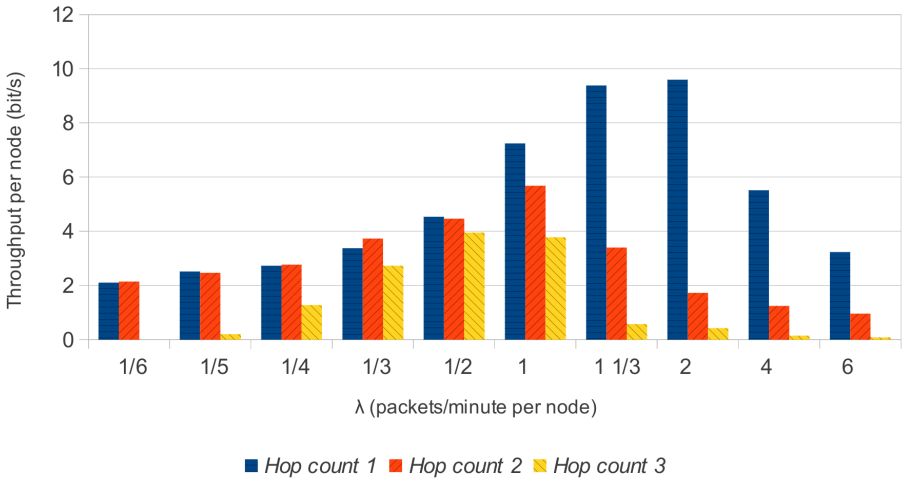

Fig. 12: Mean throughput per node (in bits/s) as a function of λ (packets/minute per node) for different values of Hop Count. Configuration with packets 125 bytes long, w/o status packets and with SNR as metric.

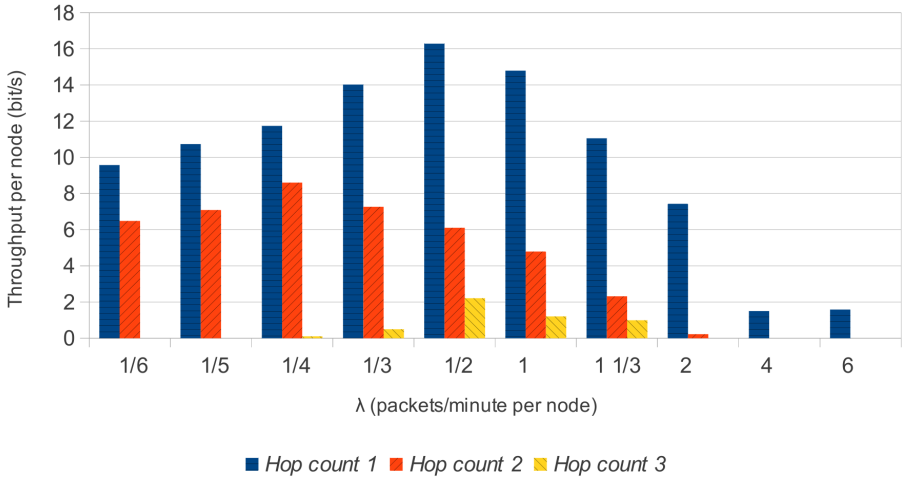

Fig. 13: Mean throughput per node (in bits/s) as a function of λ (packets/minute per node) for different values of Hop Count. Configuration with packets 500 bytes long, w/o status packets and with SNR as metric.

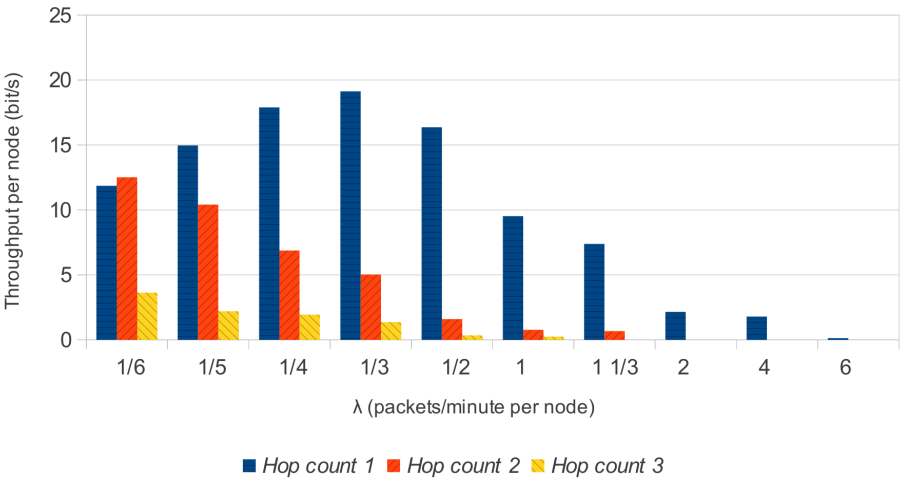

Fig. 14: Mean throughput per node (in bits/s) as a function of λ (packets/minute per node) for different values of Hop Count. Configuration with packets 1250 bytes long, w/o status packets and with SNR as metric.

Figures 15, 16 and 17 contain the mean throughput per node (in bits/s) as a function of λ (packets/minute per node) for different values of Packet Size with a fixed value of Hop Count. These graphs show how and how much the mean throughput per nodes with the same hop count changes with different values of packet size.

In the case of Figure 15 is it possible to notice that the maximum mean throughput per node with hop count 1 is reachable with $\lambda = \frac{1}{3}$ for packets 1250 bytes long, $\lambda = \frac{1}{2}$ for packets 500 bytes

long and $\lambda = 2$ for packets 125 bytes long. Very different values of λ can be obtained also in the case of Figure 16 and 17 but the shape is always the same: increasing values, a peak and and a rapid decline of the mean throughput.

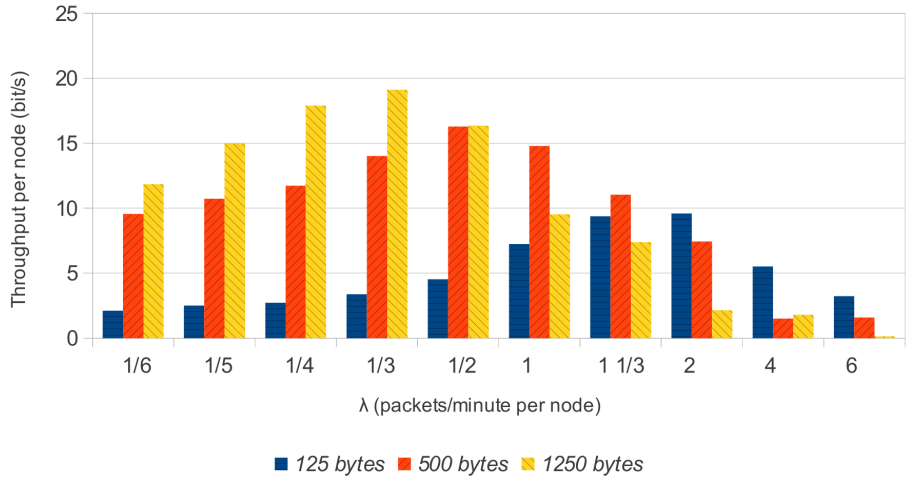

Fig. 15: Mean throughput per node (in bits/s) as a function of λ (packets/minute per node) for different values of Packet Size. Configuration with nodes with Hop Count 1, w/o status packets and with SNR as metric.

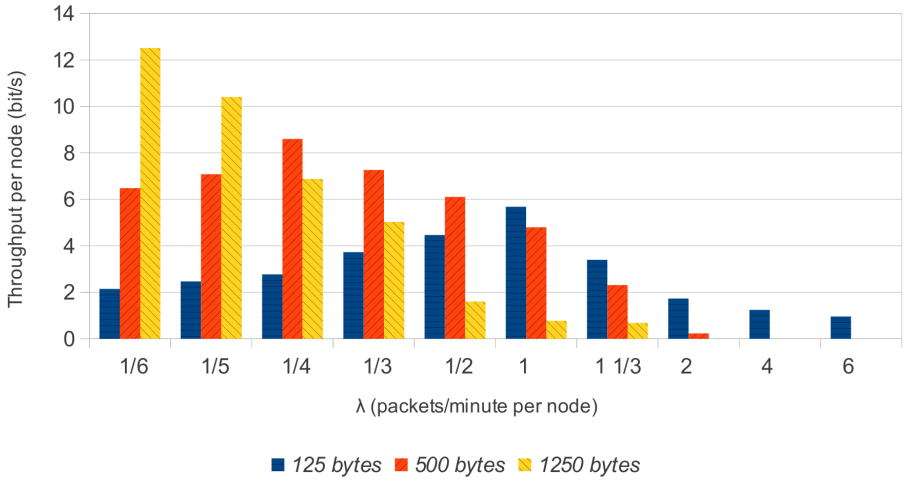

Fig. 16: Mean throughput per node (in bits/s) as a function of λ (packets/minute per node) for different values of Packet Size. Configuration with nodes with Hop Count 2, w/o status packets and with SNR as metric.

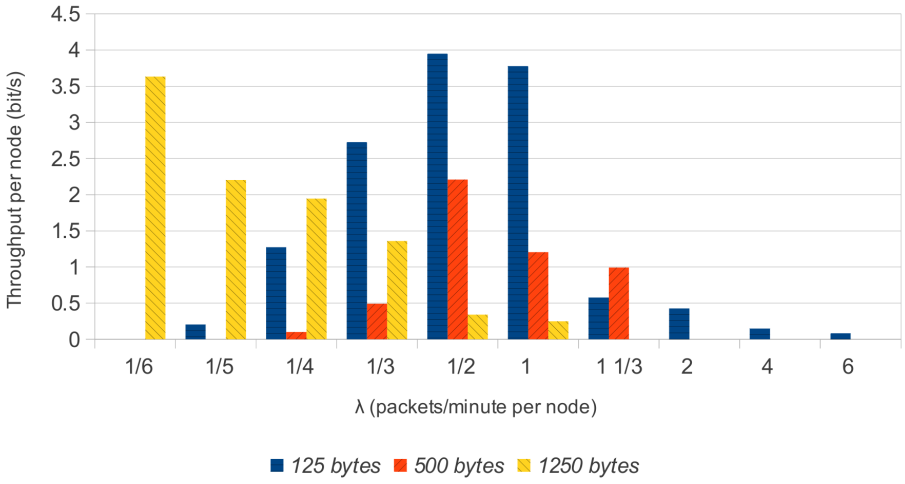

Fig. 17: Mean throughput per node (in bits/s) as a function of λ (packets/minute per node) for different values of Packet Size. Configuration with nodes with Hop Count 3, w/o status packets and with SNR as metric.

6.3.2 Simulation With Status Packets

This section considers the same scenarios simulated in Section 6.3.1 but with the status packets system. In SpreadUW the status packets system is not used for packets retransmission: SpreadUW is a routing protocol, whereas retransmissions are the main concern of the link control and transport layers. Status packets in SpreadUW are used to obtain information about the state of the links. When a packet reaches a node, the node sends back a message to acknowledge that the link is up and that its quality is high enough to receive packets. If a node does not receive a given number a of status packets it declares the next hop as unreachable. If the node that discovered a broken link receives other packets with an unreachable hop as the next destination, it will create an error message and send it back to the source in order to notify the problem. The source node can use another path or starts a new path search.

Figures 18, 19, 20, 21, 22, 23, 24 and 25 contain the results.

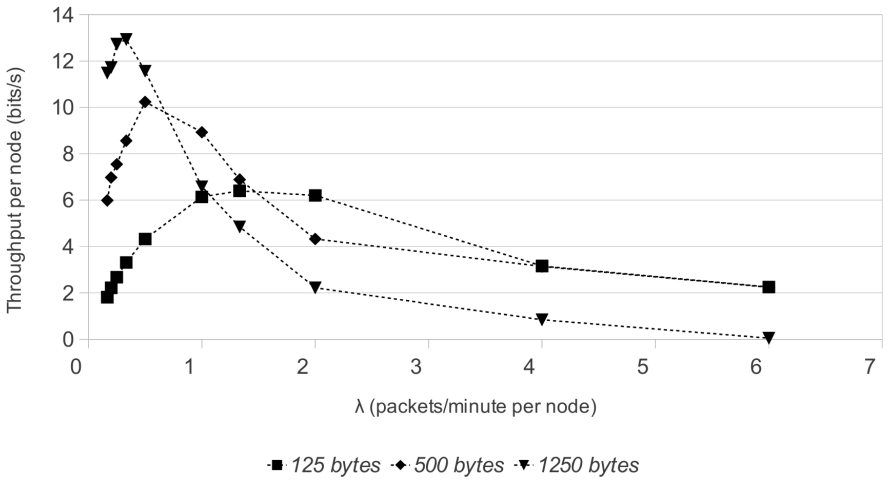

Fig. 18: Mean throughput per node (in bits/s) as a function of λ (packets/minute per node) for different values of packet size. Configuration with status packets and with SNR as metric.

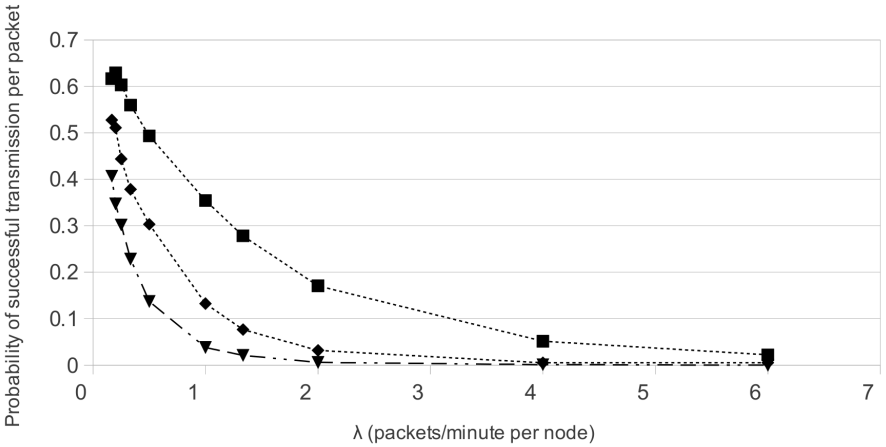

Fig. 19: Packet probability of successful transmission as a function of λ (packets/minute per node) for different values of packet size. Configuration with status packets and with SNR as metric.

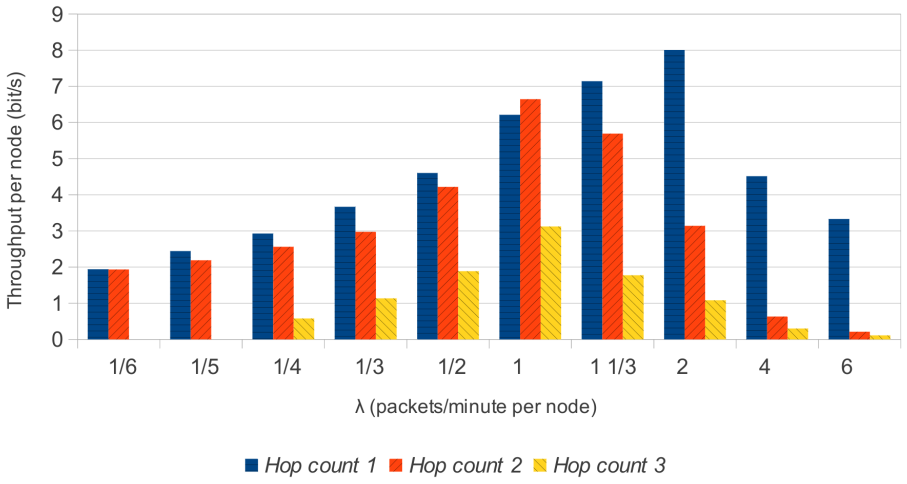

Fig. 20: Mean throughput per node (in bits/s) as a function of λ (packets/minute per node) for different values of Hop Count. Configuration with packets 125 bytes long, with status packets and with SNR as metric.

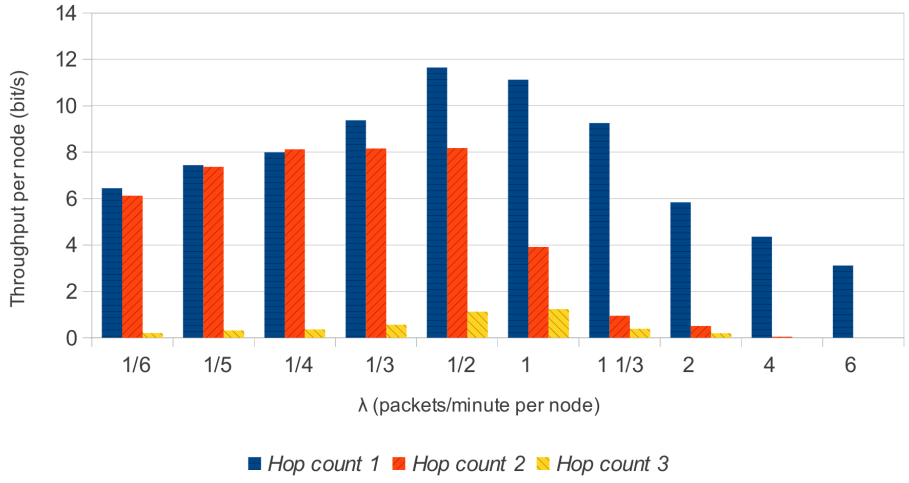

Fig. 21: Mean throughput per node (in bits/s) as a function of λ (packets/minute per node) for different values of Hop Count. Configuration with packets 500 bytes long, with status packets and with SNR as metric.

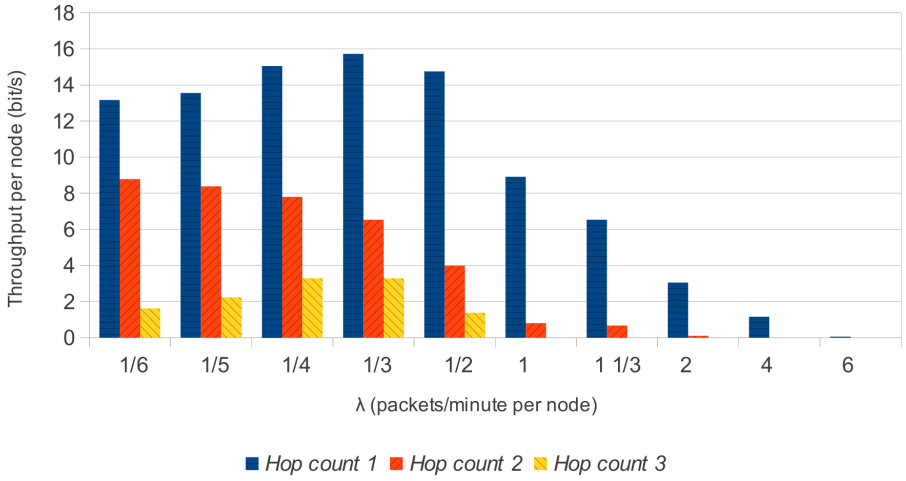

Fig. 22: Mean throughput per node (in bits/s) as a function of λ (packets/minute per node) for different values of Hop Count. Configuration with packets 1250 bytes long, with status packets and with SNR as metric.

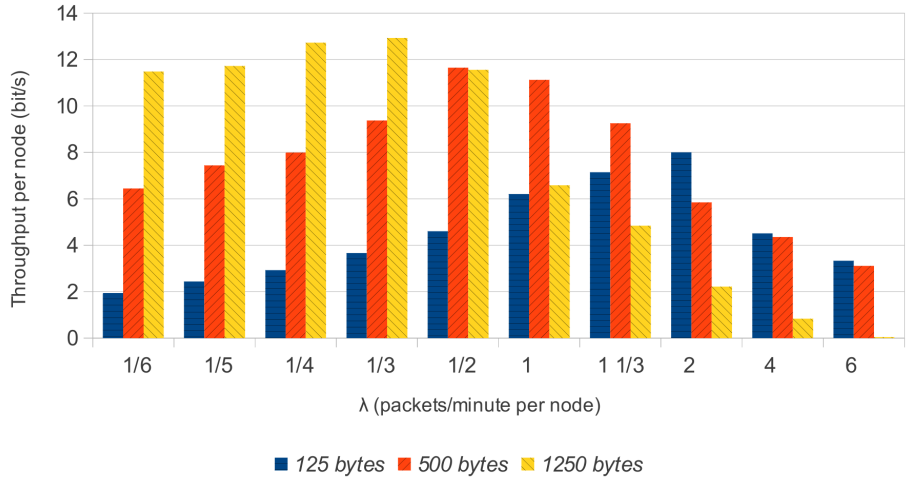

Fig. 23: Mean throughput per node (in bits/s) as a function of λ (packets/minute per node) for different values of Packet Size. Configuration with nodes with Hop Count 1, with status packets and with SNR as metric.

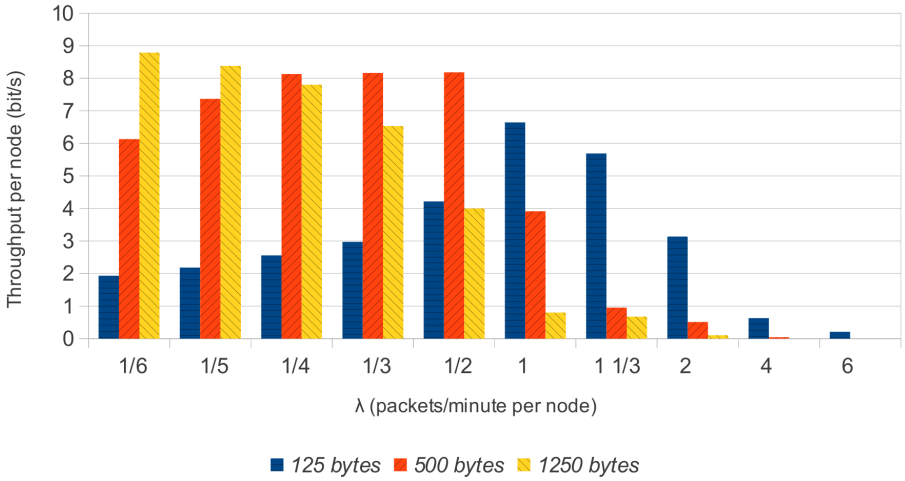

Fig. 24: Mean throughput per node (in bits/s) as a function of λ (packets/minute per node) for different values of Packet Size. Configuration with nodes with Hop Count 2, with status packets and with SNR as metric.

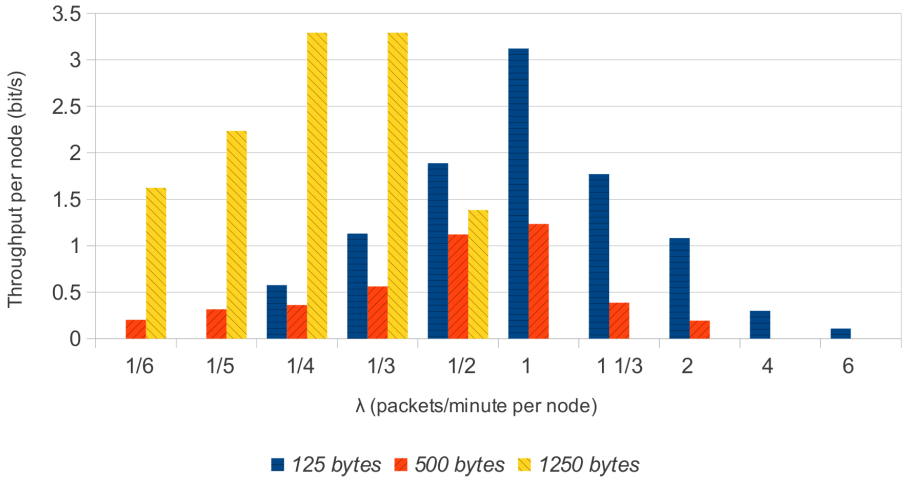

Fig. 25: Mean throughput per node (in bits/s) as a function of λ (packets/minute per node) for different values of Packet Size. Configuration with nodes with Hop Count 3, with status packets and with SNR as metric.

6.3.3 With or Without Status Packets

As shown in Sections 6.3.1 and 6.3.2 the throughput of SpreadUW without the status packets system is higher than that with the status packets system. This happens because there are less packets in the channel and therefore less collisions.

Figure 26 illustrates a comparison of mean throughput per node between the version of the protocol with and without status packets. The version without akcs generally performs better and the mean gain is $\simeq 1.48\%$.

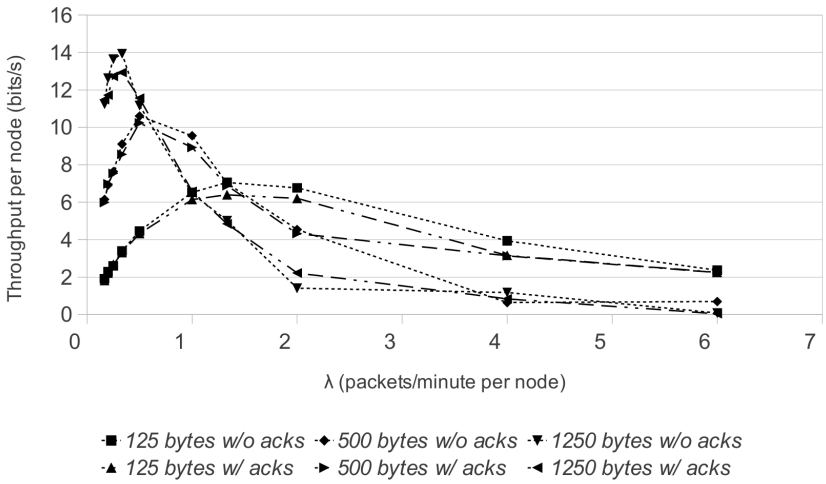

Fig. 26: Comparison of w/ status packets and w/o status packets. Mean throughput per node (in bits/s) as a function of λ (packets/minute per node) for different values of Packet Size. Configuration with SNR as metric

Figure 27 presents a comparison of packet probability of successful transmission between the versions of the protocol with and without status packets. For the same reasons that affect the throughput, also in this case the version without akcs generally performs better and in this case the difference is $\simeq 3.4\%$.

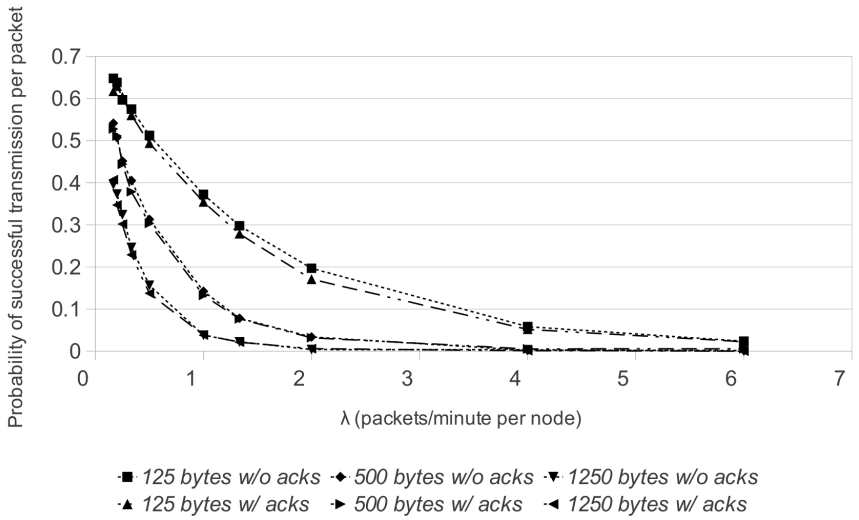

Fig. 27: Comparison of w/ status packets and w/o status packets. Packet probability of successful transmission as a function of λ (packets/minute per node) for different values of packet size. Configuration with SNR as metric

Considering the total amount of status packets sent by the protocol, the algorithm performs very well because they do not affect significantly the performances. For example, in a network with: n_1 nodes with hop count one, n_2 nodes with hop count two and n_3 nodes with hop count 3, where each node sends k packets without errors, the total number of status packets sent is:

$$number\,of\,status\,packets = (n_1 + n_2 + n_3) \cdot k$$

As will be reported in Section 6.3.4, if the mean hop count in a network that uses SNR as metric is $\simeq 1.80$, the

$$number\,of\,status\,packets = 1.80 \cdot (n_1 + n_2 + n_3) \cdot k$$

So the number of status packets is approximately twice the number of packets sent correctly.

An interesting result is that the status packets mechanism in some cases normalizes the throughput of the nodes with different hop count values. This is valid especially for intermediate values of λ.

Figure 28 illustrates the ratios between the mean throughput of nodes with hop count 2 and hop count 1. When the ratio is close to one, that means that the nodes with different hop counts have a similar throughput. This example clearly shows that with packets 500 bytes long the status packets version performs better than the one without status packets. There are two main reasons behind this result:

- the mean throughput of the nodes in the case with status packets is lower, and it the ratio between the throughput of nodes with hop count 2 and 1 is lower also;

- when the nodes use the status packets mechanism, they discover an unreliable path earlier than without status packets. This happens because without status packets the only way to discover them is by an expiration of a timer; on the contrary, thanks to the status packets mechanism, a node finds out the problem earlier.

These are the reasons why the ratio is much more close to 1 in the case of the status packets version.

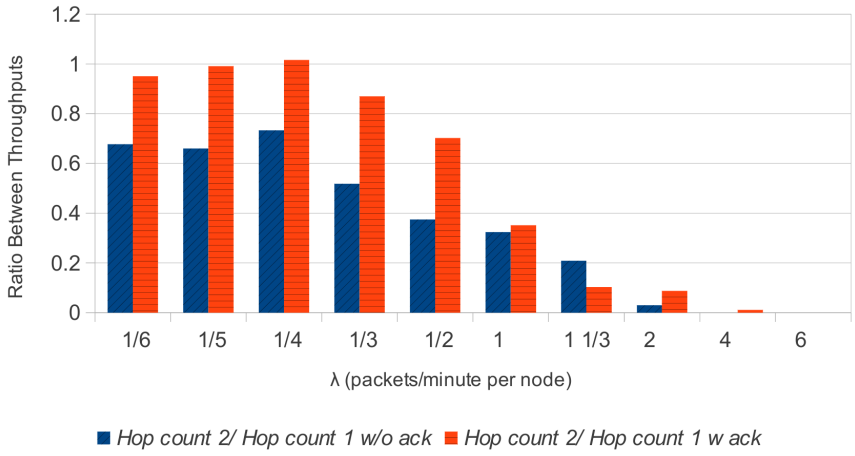

Fig. 28: Comparison of w/ status packets and w/o status packets. Ratio between mean throughput of nodes with hop count two and hop count one. Configuration with SNR as metric and packets 500 bytes long.

Figure 29 is another example, in that case the previous consideration are valid only for $\lambda \in [\frac{1}{4}, 2]$.

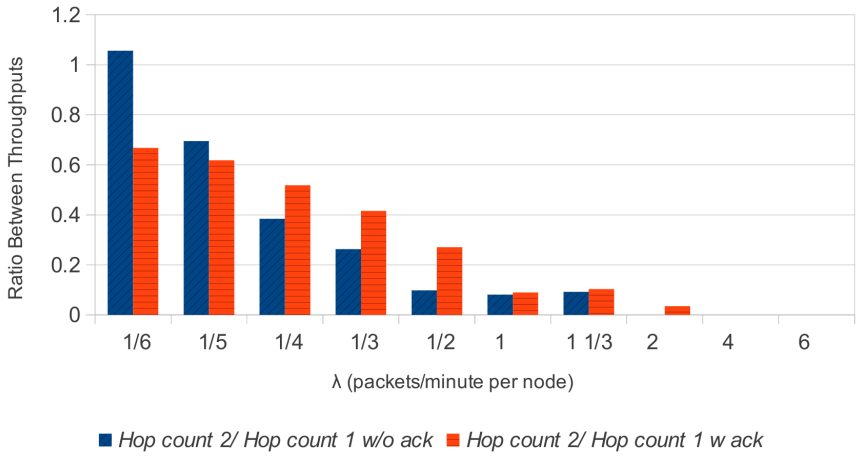

Fig. 29: Comparison of w/ status packets and w/o status packets. Ratio between mean throughput of nodes with hop count two and hop count one. Configuration with SNR as metric and packets 1250 bytes long.

Nodes with hop count 3 were excluded by this analysis because their number is to low to obtain meaningful results.

6.3.4 Comparison Between Metrics

This section contains a comparison of the performances of the three metrics implemented in SpreadUW.

All the simulations are based on a scenario with packets 500 bytes long, and a value of $\lambda = 1/2$.

Figure 30 contains a comparison of the throughput of the three metrics. Lowest Hop Count metric performs very similarly to the Lowest Average Load metric for values of hop count equals to 1 and 2; the behavior changes when the hop count is 3 or 4. In the last two cases the Lowest Average Load metric is able to connect successfully the nodes and the sink, while Lowest Hop Count metric is not able to establish a valid link.

The metric that performs more equally between all the nodes is the Best Average SNR: the throughput for nodes with hop count 1 is lower if compared with the other two metrics, but in the case of hop count 2, 3 and 4 the throughput of a node using Best Average SNR is always higher than the other two.

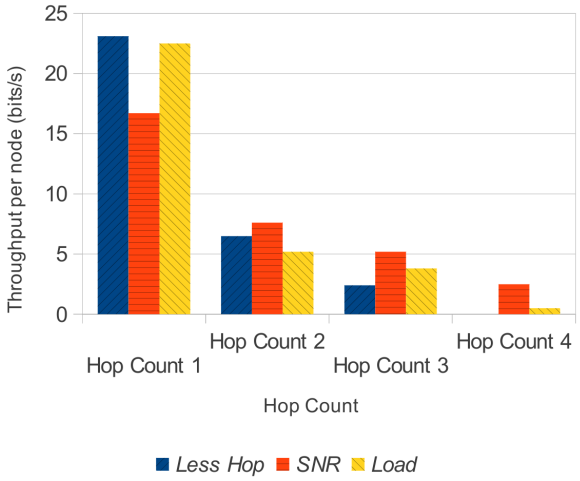

Fig. 30: Mean throughput per node (in bits/s) as a function of the hop count for different metrics. Configuration with status packets, packets 500 bytes long and $\lambda = 1/2$.

In Figures 31 and 32 there is a comparison of the mean throughput and the packets delivery ratio of the three metrics.

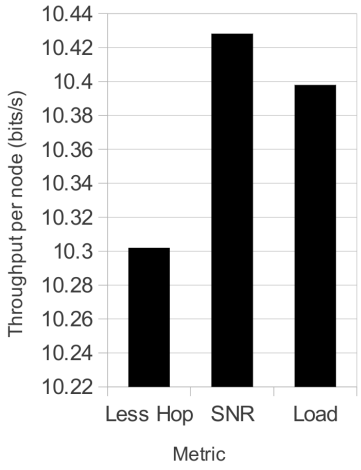

Fig. 31: Mean throughput per node (in bits/s) as a function of metric. Configuration with status packets, packets 500 bytes long and $\lambda = 1/2$.

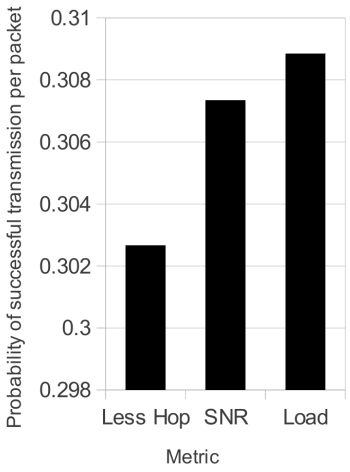

Fig. 32: Packet probability of successful transmission as a function of different metrics. Configuration with status packets, packets 500 bytes long and $\lambda = 1/2$.

From the results it is possible to conclude that the overall performance of the three algorithms is very similar. As regards the throughput they differ only by a 2%, and the same consideration is valid for the packet probability of successful transmission. This happens because the network is static during the entire simulation. The most important fact is that, even if the throughput of Best Average SNR is very close to the throughput of Lowest Hop Count metric, the throughput is distributed in a very different way across the nodes.

7 Conclusions

Even though Underwater Acoustic Networks are a field of studies completely new, the high number of quality publications demostrate the interest of the scientific community in this subject . There are several pubblications for what concerns the Physical and Data Link Layer, but as for the upper layers, like the Networking one, pubblications regard mainly the application of knowledge for terrestrial wireless networks into underwater acoustic networks.

To complicate the scenario, most of literature refers to a specific scenario, limiting the generality of results. This contributed to the development of a generic protocol which can adapt to different scenarios and conditions. SpreadUW is a routing algorithm based on two principles: modularity and generality. It is modular because the decision algorithms of the protocol, as the decision of the best paths, are easily modified/modifiable, and it is general because SpreadUW does not require any information about the network and is able to operate in any scenario.

This paper describes the basic principles, ideas and the implementation decisions of SpreadUW. It procedees/continues with the development and integration of the algorithm in a network simulator, in order to test the real performance with a complete network stack of protocols.

The algorithm is proposed in three metrics for the best path decision problem: Lowest Hop Count, Best Average SNR and Lowest Average Load. The final user has the possibility to create new policies for the best path selection and to decide whether to use the status packets system. The same is valid for the possiblity of setting up the various parameters that concern the routes validity.

The performance evaluation concerns mainly an analysis of the throughput of the entire network, the packet delivery ratio and the throughput for every single node with different hop counts. The latter parameter is the one that required more attention: in a communication network of nodes it is important that every entity is able to transmit data to the destination. The nodes must be able to cooperate in an autonomous way to give everyone the opportunity to transmit data.

The simulation results shown that, changing the hop count metric, the overall performance of the network are uniform. The most influenced result is the distribution of throughput according to the hop count value of a node. In the simulations, the value of hop count started from 1 up to 4. Although the algorithm Less Hop Count offers high thorughput, it is not able to fairly distribute the traffic between the nodes; Best Mean Mean SNR and Less Load behave much better/are much more efficient.

The presence of a status packets system does not affect network performance significantly. Although the additional traffic generated by that mechanism does not visibly degrade network performance, it is unable to give a performance boost. On the contrary, in a future version with mobile nodes (that will be different, because) the status packets system will be essential to identify problems and make it possible for nodes to find out first a “broken” link.

SpreadUw aims to be a general routing algorithm, stable, able to operate in any simulated environment and customizable by the end-user. It is also proposed as an algorithm to be used as a yardstick for future benchmarks.

Future work on SpreadUW will concern:

- the implementation and the integration of a mobility model. Currently there are no comprehensive studies about mobility in an underwater environment, and also there are no underwater-specific mobility models implemented for simulation;
- the implementation of new metrics for path selection and also the possibility to rank paths from the best to the worst. Thanks to this, the nodes will be able to pick different paths in real time if some problem arise, without waiting the answer for a new path establishment:
- as already mentioned, planned for the future are simulation with of SpreadUW that uses for the physical layer and the channel a framework such as WOSS[29] that uses a ray tracer model like BellHop[30].
- a possible evolution of SpreadUW could involve a porting in the most recent and innovative simulators like NS3;
- a comparison between SpreadUW and other generic routing protocols.

[29] WOSS project homepage: `http://telecom.dei.unipd.it/ns/woss/`
[30] BellHop website: `http://oalib.hlsresearch.com/Rays/index.html`

List of Tables

List of Algorithms

List of Figures

References

[1] Ethem M. Sozer, Milica Stojanovic, and John G. Proakis, “Underwater Acoustic Networks”, *IEEE Journal of Oceanic Engineering*, vol. 25, no. 1, January 2000.

[2] Milica Stojanovic, “On the Relationship Between Capacity and Distance in an Underwater Acoustic Communication Channel”, OCEANS 2008 - MTS/IEEE Kobe Techno-Ocean, May 2008.

[3] Haiyong Xie, Lili Qiu, Yang Richard Yang, and Yin Zhang, “On Self Adaptive Routing in Dynamic Environments“ , Proceedings of the 12th IEEE International Conference on Network Protocols, 2004. ICNP 2004.

[4] Kurose, James E. and Ross, Keith W. “Computer Networking, Third Ed”, Benjamin/Cummings, 2004, ISBN 0321227352.

[5] H. Song, W. Hodgkiss, and W. Kuperman, "MIMO Time Reversal Communications," in Proc. International Workshop on UnderWater Networks (WUWNet), Montreal, Canada, Sep 2007.

[6] S. Hwang and P. Schniter, "Efficient multicarrier communication for highly spread underwater acoustic channels," IEEE J. Select. Areas Commun., this issue, 2008.

[7] D. Lucani, M. Medard, and M. Stojanovic, "Network coding schemes for underwater networks: The benefits of implicit acknowledgement," in Proc. International Workshop on UnderWater Networks (WUWNet), Montreal, Canada, Sep 2007.

[8] R. Urick, Principles of Underwater Sound. McGraw-Hill, 1983.

[9] E. Sozer, M. Stojanovic, and J. Proakis, "Initialization and routing optimization for ad hoc underwater acoustic networks," in Proc. OP-NETWORK, Washington, DC, Sep. 2000.

[10] R. Nitzel, C. Benton, S. G. Chappell, and D. R. Blidberg, "Exploiting dynamic source routing to enable undersea networking over an ad-hoc topology," in Proc. International Symposium on Underwater Technology, Tokyo, Japan, Apr. 2002.

[11] D. Pompili, T. Melodia, and I. Akyildiz, "Routing algorithms for delayinsensitive and delay-sensitive applications in underwater sensor networks," in Proc. Annual International Conference on Mobile Computing and Networking (MobiCom), Los Angeles, CA, USA, Sep 2006.

[12] Michele Zorzi, Paolo Casari, Nicola Baldo, and Albert F. Harris III, “Energy-Efficient Routing Schemes for Underwater Acoustic Networks”, IEEE Journal, vol. 26, no. 9, December 2008.

[13] C.E.Perkins and E.M.Royer, “Ad-hoc on-demand distance vector routing,” Proceedings of the Second lEEE Woihhop on Mobile Computing Systems and Applications, 1999, WMCSA ‘99., pp. 9&100, 25-26 Feb. 1999.

[14] K. Y.Foo, P. R. Atkins, T. Collins, C . Morley and J. Davies, “A Routing and Channel-Access Approach for an Ad Hoc Underwater Acoustic Network”, OCEANS ’04. MTTS/IEEE TECHNO-OCEAN ’04, On page(s): 789 - 795 Vol.2, 2004.

[15] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, “Adaptive protocols for information dissemination in wireless sensor networks” Mobicom99, August 1999.

[16] M. Zorzi, R.R. Rao, “Geographic random forwarding (GeRaF) for ad hoc and sensor networks: multihop performance”, IEEE Transactions on Mobile Computing, vol.2, no.4, pp. 337-348, Oct.-Dec. 2003.

[17] P. Xie, J.-H. Cui, and L. Lao. Vbf: Vector-based forwarding protocol for underwater sensor networks. Proceedings of IFIP Networking, May 2006.

[18] Hai Yan, Zhijie Jerry Shi, and Jun-Hong Cui, “DBR: Depth-Based Routing for Underwater Sensor Networks”, Department of Computer Science anpossibile d Engineering University of Connecticut.

[19] D. B. Johnson, D. A. Maltz, and Y.-C. Hu, “The dynamic source routing protocol for mobile ad hoc networks (DSR)”, INTERNET-DRAFT, July 2004. IETF MANET Working Group.

[20] Edward A. Carlson, Pierre-Philippe Beaujean and Edgar An, "Location-Aware Routing Protocol for Underwater Acoustic Networks", OCEANS 2006, page(s): 1 - 6, 2006.

[21] Milica Stojanovic, ”On the Relationship Between Capacity and Distance in an Underwater Acoustic Communication Channel”, Massachusetts Institute of Technology.

[22] N. Baldo, F. Maguolo, and M. Miozzo, “A new approach to simulating PHY, MAC and Routing,” in Proc. ACM International Workshop on NS-2, Athens, Greece, Oct 2008.

[23] D. S. J. De Couto, D. Aguayo, B. A. Chambers, and R. Morris, “Per- formance of multihop wireless networks: shortest path is not enough,” SIGCOMM Comput. Commun. Rev., vol. 33, no. 1, pp. 83-88, 2003.

[24] R.Coates, Underwater Acoustic Systems, New York: Wiley, 1989.