UNIVERSITY OF OSLO
Department of Informatics

Simulation of subsea communication network

Master Thesis

Håvard Austad

Spring 2014

Abstract

The underwater acoustical communication channel is a harsh environment, making it tough to transmit data error free. Due to high absorption of electromagnetic energy, neither radio nor optics are suited for long range transmissions under water. Despite high attenuation in the upper frequency range, a great deal of noise in the lower end, limited bandwidth, and high propagation delay, acoustics is the most suited way to carry information underwater.

The problems stated above makes achieving a robust and reliable network hard and tough requirements are set to the protocols. Because of the long and varying propagation delay, the focus of this thesis has been to improve a network protocol with focus to increase packet delivery ratio (PDR). The changes were implemented, and to verify the changes, simulations were done using measurement based Look up Tables supplied by the RACUN (Robust Acoustic Communication in Underwater Networks) project. There were also investigated if the improvements to the protocol can "out of the box" solve other scenarios than the focus scenario, like a Postman-scenario using an AUV (Autonomous Underwater Vehicle) to gather and deliver packets. The protocol's ability to survive topology changes was also simulated. The thesis focuses on simulations of long-range networks (10 - 20 nautical miles from source to sink).

Initial simulations indicated that adding retransmissions to Dflood (the network protocol), PDR increased significantly in some cases, but only slightly increased in other. In the latter cases, the original Dflood protocol performed rather well.

Overall, with the implementation of the improvements to Dflood, the protocol performed better in terms of PDR. But realization of a Postman scenario on the network layer using Dflood, was not a complete success.

Improvements of Dflood protocol implemented in this thesis have been added to the RACUN software framework and will be applied during the RACUN Sea Trial in May 2014.

Simulation of subsea communication network

Håvard Austad

Spring 2014

Contents

List of Figures

List of Tables

Preface

This thesis completes my master degree from the Department of Informatics: Networking and programming, at the University of Oslo. I would like to thank my supervisor Dr. Roald Otnes at Norwegian Defence Research Establishment (FFI) for providing an interesting thesis and excellent support and guidance during the thesis period. A thanks to FFI for providing with a office facility. Thanks are also due to the RACUN project for letting me use their collected data, simulation framework and network protocols developed in RACUN. The GUWMANET protocol is used in some simulations in this thesis and the protocol is provided to RACUN as background information by WTD71/FGW (Germany) [10].

Thanks to my second supervisor Prof. Dr. Josef Noll for support and guidance.

Special thanks to my wife, Marianne, for letting me study to the master degree and supporting me on the way there. And to Amanda, my daughter, for disturbing my work with a smile and a tea party with her teddy bears.

Chapter 1

Introduction

"If you cause your ship to stop, and place the head of a long tube in the water, and place the other extremity to your ear, you will hear ships at a great distance from you". With Leonardo da Vinci's first passive sonar, the area of underwater acoustics was born [2]. Since then, there have been sincere amounts of research in the field of sonar (especially during WWI and WWII) and later underwater transmission of information. In modern time the concept of small wireless sensors forming a network has become rather popular in terrestrial environments. With the sonar technology constantly improving, the idea of making a network of sonar-like sensor nodes for surveillance of a geographic area has been introduced. The ongoing exploration of the oceans calls for sensors to cover large geographic areas. The development of Autonomous underwater vehicles (AUVs) has also been a driver for further development of underwater communication. But the underwater environment is harsh, and communications between the nodes suffer long delays, variable reliability and low bandwidth. One solution may be to improve the network layer to make the data delivery more reliable and robust. In this stage of the development, the goal is not low latency or high bit rate, but to make sure the packets are delivered. This will be the focus area and main evaluation criteria for this thesis.

1.1 Overview

This thesis gives a background overview in Chapter 2. The pros and cons of simulations together with a description of the relevant simulation tools are given in Chapter 3. The underwater acoustic channel is described in Chapter 4. A description of state-of-the art technologies in various underwater networking techniques and protocols is given in Chapter 5, 6 and 7. A description of the RACUN project is given in Chapter 8 before the simulation setup is described in Chapter 9. The results are presented in Chapter 11 and some conclusions are drawn in Chapter 12

Chapter 2

Background

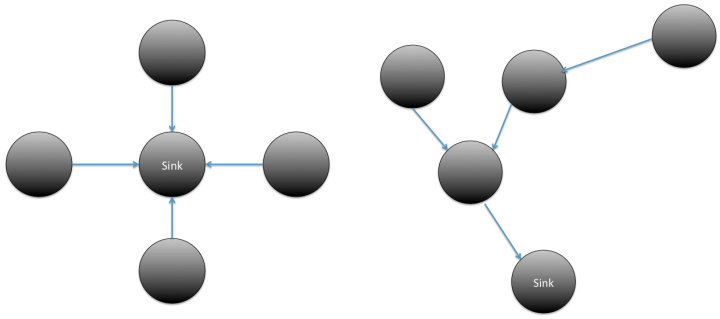
Wireless terrestrial networks have been around for decades, spoiling the human race with wireless internet and high capacity cell phones. When it comes to wireless communication underwater, the situation is different. The underwater channel is among the most harsh and challenging communication channel known to man. Very high absorption of electromagnetic energy making the use of traditional radio difficult over larger distances. Large spreading of light making the use of optical transmissions difficult as well, leaving acoustics. With the ocean filled with ships, whales, fish and other animals making sounds, there are a large amount of noise in the ocean. This noise, combined with the underwater channels ability to absorb high frequencies, results in a small usable frequency band underwater. The long propagation delay is not beneficial either.

2.1 Sensor Network

A sensor network is as the name indicates, a network with sensing capabilities. The network consists of nodes with different sensing capabilities forming a network, either wired (not so common) or wireless. The main idea is that the network should, to some extent, be self-configuring so applying (or removing) nodes should be a treat. The data collected from the sensor nodes are often sent to a ”master node” with gateway functionality, responsible to forward the information to a server or operator who can interpret the data. Data collected can be of various types, but examples are temperature, humidity, movements, vibrations etc.

The size of the network is determined by many factors. Transmission range and battery capacity is often two opposite requirements to a network, since increased transmission range corresponds to higher power usage. But of course if the physical size of the nodes and the battery technology allows it, both these requirements can be met. To increase the range without increasing the transmission power, multi hop networks can be applied. This requires a larger number of nodes, better control mechanisms to handle the increased number of transmissions, and preferably a routing protocol/mechanism for efficient packet delivery. A multi-hop network

can, on the other hand, reach over a greater distance compared to a single hop topology as illustrated in Figure 2.1.



(a) Single hop topology. (b) Multi hop topology

Figure 2.1: Different topologies

Nodes are inserted into a network, often without an overlaying structure, meaning the nodes becomes both the users and the backbone of the network, making them responsible for the application (sensing, analyzing etc) and maintaining and administrating the network structure like e.g. maintaining routing tables and neighbor discovery.

2.2 Scenarios

There is (as with many thing these days) only the imagination that limits to what scenario a sensor network can be applied. There can be personal sensor networks sensing and reporting the status of your body, private sensor networks sensing, reporting and maintaining the home in various ways or larger networks monitoring fields or buildings. Underwater sensor networks have many applications as well. There can be monitoring the ocean itself e.g. monitoring the global warming impact on the ocean or sensing for leaks along oil and gas pipelines. Another scenario is in the aquaculture business where the farming cages can be moved out to sea and lowered below the surface. One major scenario is underwater surveillance. Here the nodes can listen for intruders and report to a command center. There can be surveillance of a harbor, hostile territory or other relevant areas. The nodes in a underwater network are located at various depths, and can be either static or mobile. The static nodes may not be so static as one could wish due to drifting with the ocean current, but they can be e.g. mounted on the sea bottom tightly for resisting the current. The static nodes can also be located in other depths between the bottom and the surface, but they are intended to stay at the same place. The mobile nodes, on the other hand, can often move in both vertical and horizontal directions. The mobile nodes can either be towed by a boat or self-driven often without a driver, an Autonomous underwater vehicle (AUV). With the AUVs, the network range can easily be expanded by letting the AUV have a postman role, delivering packets either inside a network or from one network to another. The AUV can also eliminate the need for some

networks in terms of traveling to the all the nodes and collect the gathered data for later deliver it to the sink. With this technique, the nodes can use less energy for transmissions than a network requires and can focus the power consumption on the sensing part. The data in this kind of scenario cannot be time critical.

This thesis will focus on a surveillance scenario containing static bottom nodes and an AUV. The scenario consists of multiple barriers making a multi hop network. The scenario is further described and illustrated in Section 11.1. How the nodes sense and for what, is outside the scope of this thesis. The sensed data will be treated as general data.

2.3 Underwater transmission

Because of the conductivity in seawater, electromagnetic waves are quickly attenuated. Seawater has conductivity about 4 S/m [18]. Attenuation of electromagnetic waves in water is given in Formula (2.1) from [18].

$$\alpha = 0.0173 \times \sqrt{f * \sigma} \tag{2.1}$$

where α is the attenuation in dB/meter, f is the frequency in Hz and σ is the conductivity. Figure 2.2 shows a plot of the rapid attenuation of electromagnetic waves in seawater with different conductivity, where the second curve from the top indicates average seawater (4 S/m).

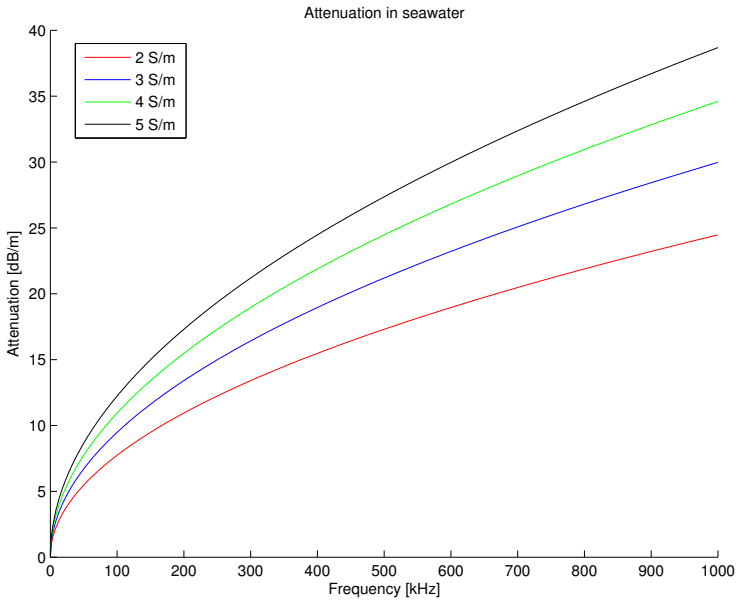


Figure 2.2: Attenuation of electromagnetic waves i seawater

Despite the large attenuation of electromagnetic waves, submarines use very low frequencies (VLF) in the range 3-30kHz for some communication [19]. Che et al. calls for a re-evaluation of electromagnetic waves

underwater in [20]. They present a case study where the VLF frequency band is used (3kHz) for communication between the nodes with a distance of 40m. This gives a path loss of ≈ 2.4 dB with a $\sigma = 4$ S/m. They conclude that electromagnetic waves can be a both feasible and effective in a specific set of applications. But this is short-range. It states in the article that for distances > 1 km the bit rate will be < 1 b/s making electromagnetic an option for short-range transmission, but a poor choice for transmission over a greater distance.

Hanson and Radic suggest optics as a transmission method in [21] and presents a case study where there is achieved a data rate of 1 Gbps. This high data rate is achieved in a laboratory through a 2 meter long water pipe. Further it is referred to experiments where transmitting over a 91 meter water tank, but this is relatively short distances for applications like temperature monitoring the ocean floor (it would require a large number of nodes).

The third option for wireless underwater transmissions is acoustics. Here the carrier waves are acoustical and not electromagnetic. Sound propagates much better in water than air. The nominal speed of sound is about 1500 m/s, while in air the sound speed is about 300 m/s. But the light speed, which electromagnetic waves travel with, is about 200 000 times faster. The acoustic waves attenuate much less than the electromagnetic waves so despite the large propagation delay this is more suitable for transmissions over greater distance under water (more in Section 4.2).

In short distance network, like the postman scenario described in Section 12.2.2, both electromagnetic and optics can be usable options as transmission methods.

2.4 Challenges with acoustic underwater networks

The small useable bandwidth, low transmission frequency and noise (see chapter 4) give the underwater networks poor performance due to low data rates. So the current underwater networks are most suited to transmit small packets, like sensor information. This excludes high resolution photo/video or other large files for further analyzing of the sensed data, so if the sensed data is large files that needs to be analyzed, it may be beneficial to do the processing on the node itself. Time synchronization is also an issue in underwater networks. All clocks have a skew and will over time drift making time synchronizing necessary. But with a very long and varying propagation delay making time synchronization in AUN a very hard task to perform. Other challenges are obvious, due to the nodes physical location like changing batteries and other maintenance operations. But its location also prevents people from tampering with the nodes.

Chapter 3

Simulation

Performing real life sea trials is the best way to verify and test a new network protocol, scenario, features or anything else. But these sea trials require expensive underwater nodes and highly qualified personnel, and ship time (also expensive). A good simulation tool is very handy developing, implementing and testing new network features. Here all the work can be performed on a single (or multiple) PC and simulating a weeklong scenario can be done in hours (depending on the hardware). Simulations are based on statistics and previous experiences. In order to get an accurate measurement of the network performance, a real-life sea trial has to be done eventually.

3.1 Network Simulators

There is a number of network simulator available out there. Some are commercial and some are open-source. Simulators comes in generally two types; continuous and discrete [22]. A discrete model considers only discrete moments in time that correspond to events that impact the simulated network. This is referred to as *discrete event* simulations (DES), and requires the simulation software to maintain a clock so the current simulation time can be monitored. Between the events, nothing happens in the network and the time between the events is not interesting either. *Continuous* simulations consider all points in time to the resolution of the host's hardware limitations (all simulations are discrete due to its running on a digital platform). Discrete methods are most commonly used for network simulations. The simulations can either be local (running on one computer) or distributed on many computer in a computer network (not simulated, but the simulating computers are interconnected). There can also be some simulated nodes (local or distributed) and some real nodes in combinations (emulation)

3.2 Network Simulator 2

Network Simulator 2 (NS-2) [23] is a popular simulation tool. Otnes and Haavik applied the simulator in [24] to test their protocol, and so does Nicolaou et al. in [7]. Goetz and Nissen mentioned to use NS-2 for in [10], also for testing their protocol. NS2 is a discrete event simulator and supports multicast and several routing protocols for wired and wireless (local and satellite) networks. The simulator is written in C/C++ and is interfaced with TCL/OTCL. The distribution between the two languages is that the simulator kernel and the network modules are written in C/C++ and compiled due to performance. Interfacing with the simulator is done with TCL/OTCL where the network will be initiated, the topology built and the different events in the simulation configured. With this distribution of languages the compiled modules performs well, but there is no need for recompiling every time there is a change in the topology.

3.2.1 NS-MIRACLE

NS-MIRACLE (Multi-InteRfAce Cross-Layer Extension) is a extension to NS2 designed to enhance the the functionality of NS2. One of the primary goals of NS-MIRACLE [25] is to facilitate the interconnection of different protocol modules. A important piece of the NS-MIRACLE framework is the *Module* class which contains the *sendDown()* and *sendUp()* functions handling the interconnection between the layers. Some of the modules in standard NS-2 also contains functions called *sendDown()* and *sendUp()* (e.g. the mac module) but NS-MIRACLE introduces in the Module class sendDown() and sendUp() as abstract methods that have to be implemented by the different modules. With the Module class, NS-MIRACLE enables the coexistence of multiple modules within each layer at the protocol stack. NS-MIRACLE framework also contains multiple libraries especially in the PHY (physical) layer including *UnderwaterShannon* and *Underwater BPSK*. Other extensions to NS2 builds uses NS-MIRACLE as an interface and actually extends NS-MIRACLE rather than NS2 directly as extensions described in Section 3.2.2 and Section 3.2.3.

3.2.2 DESERT underwater

DESERT Underwater or DESERT (DEsign, Simulate, Emulate and Realize Test-beds) is a set of C/C++ libraries to support the design and implementation of underwater network protocols [26]. DESERT aims at extending NS-MIRACLE to provide several protocol stacks for underwater networks as well as the support routines required for development of new protocols. There is also in [26] described interfacing with real hardware for emulation and test-bed setup to test newly developed protocols.

3.2.3 WOSS

World Ocean Simulation System (WOSS) is an extension to NS-MIRACLE by Guerra et al.described in [27]. WOSS enables a more specific simulation of underwater propagation models rather than use empirical models. WOSS may use Bellhop ray tracing which require knowledge about the speed sound profile (SSP) (described in Section 4.1), bathymetric profile and the type of bottom sediments. WOSS has a large database of information containing measured SSP from experiments and bathymetric data from General Bathymetric Chart of Oceans. The main advantage of the effort put into gathering this information into a WOSS-database, is the simulation user only have to create an OTCL object where the latitude longitude and network size are set and WOSS handles the rest.

3.2.4 AQUA-sim

Aqua-sim is an extension package to the NS2 core (as described in Section 3.2) as is described in detail in [1]. Aqua-sim builds on the same principles as NS2 (with the two language C++ and OTcl) and implements specific underwater parameters and protocols. This simulation tool implements some of the most common MAC and routing protocols used in underwater networks. To verify their simulations tool, Xie et al. have in [1] ran a topology in a testbed and compared the results with the same topology in Aqua-sim. The results have some variation as illustrated in Figure 3.1, but are still pretty close.

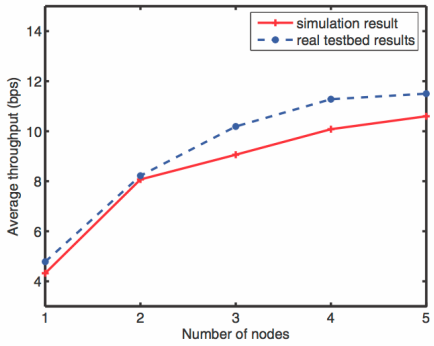


Figure 3.1: Comparing results test bed vs simulations [1]. Throughput with fixed input traffic per node

Chapter 4

The Underwater Acoustic Channel

The underwater acoustic channel is one of the most challenging wireless communication channels. Noise at lower frequencies and absorption of higher frequencies makes the usable frequency band limited to a few hundred hertz to tens of kilohertz. The low propagation speed makes the systems vulnerable to Doppler and multi path.

In acoustics the unit *decibel* (dB) differs from the term used in terrestrial radio, where dB is referenced with *watt* or milliwatt (dBm). In acoustics the reference is pressure measured in μPa, but how many μPa differs again with the medium. In air the reference is 20 μPa, while under water 1μPa [28]. In seawater, 1W of radiated acoustic power creates a sound field of intensity 172 dB *re* μPa 1 meter away from the source [29].

4.1 Speed of sound in seawater

The speed of sound in seawater is a complex function of salinity, temperature and pressure. Salinity is per definition a ratio of mass dissolved salt in water. But today this definition is suppressed by *practical salinity*, defined as a ratio in terms of the conductivity in the salt-water resolution in such a way that its value is almost identical to that of absolute salinity expressed in *parts per thousand* by mass. Compared to the temperature the effect of salinity is rather small. Salinity in the major oceans is normally in the range of 34.5 and 35.0 ppt [2, p.130] with a mean salinity of world ocean of 34.72. In the extremity there are the Baltic Sea with 8 ppt and Red Sea with 40 ppt [30]. The empirical formula of sound speed is given in 4.1 [2, p.140] where S is salinity [ppt], T is temperature in Celsius and z is depth in meters (the depth is measured from the surface so no negative numbers).

$$\begin{aligned} c(S,T,z) = {} & 1448.96 + 4.591T - 0.05304T^2 + 2.374 \times 10^{-}4T^3 \\ & + (1.340 - 0.01025T)(S-35) + 0.01630z \\ & + 1.675 \times 10^{-7}z^2 - 7.139 \times 10^{-13}Tz^3 \end{aligned} \tag{4.1}$$

As the depth varies in (4.1) so does the salinity and the temperature. As a result of this, the sound speed may have large variation relative to the depth. The plot of Equation 4.1 as shown in Figure 4.1 has used data from World Ocean Atlas (1999) (published by the National Oceanographic Data Center) to provide temperature and salinity in different depths. Plots like Figure 4.1 are often referred to as *sound speed profiles*.

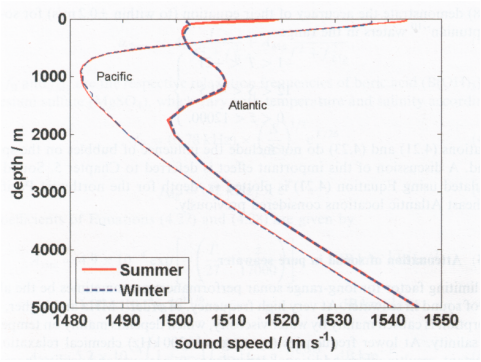


Figure 4.1: Sound speed profiles created using (4.1) and data from WOA. From [2]

4.2 Attenuation and Noise

One limiting factor in the underwater acoustic channel is absorption. For larger frequencies (1 MHz and above) the main absorption is caused by the water viscosity, which depends on temperature and salinity. For lower frequencies (up to about 300 kHz) the chemical composition of the seawater is the driver for absorption. Air has also an influence on the attenuation of sound underwater. The possible presence of bubbles and large amount of fish (with or without gas-bladder) are also important especially in costal areas.

The path loss over a distance l for a signal with frequency f is often approximated as [29]

$$A(l, f) = A_0 l^k a(f)^l \tag{4.2}$$

where A_0 is a unit-normalizing constant, k is a spreading factor and $a(f)$ is the absorption coefficient. In dB the path loss is

$$10 \log A(l, f)/A_0 = k \times 10 \log l + l \times 10 \log a(f) \tag{4.3}$$

The first part describes the spreading loss and the spreading factor, k, described the geometry of propagation. Commonly used values for k are $k = 2$ for spherical spreading, $k = 1$ for cylindrical spreading and $k = 1.5$ for so-called practical spreading [29].

The absorption coefficient is often given by Thorp's empirical formula given in (4.4) for frequencies above a few hundred Hz and (4.5) for lower frequencies [29]. This gives

$$10\log(a(f)) = 0.11\frac{f^2}{1+f^2} + 44\frac{f^2}{4100+f^2} + 2.75 * 10^{-4} f^2 + 0.003 \quad (4.4)$$

$$10\log(a(f)) = 0.002 + 0.11\frac{f^2}{1+f^2} + 0.011 f^2 \quad (4.5)$$

where $a(f)$ in dB/km and f in kHz. A plot of the absorption coefficient is shown in Figure 4.2 and Figure 4.3.

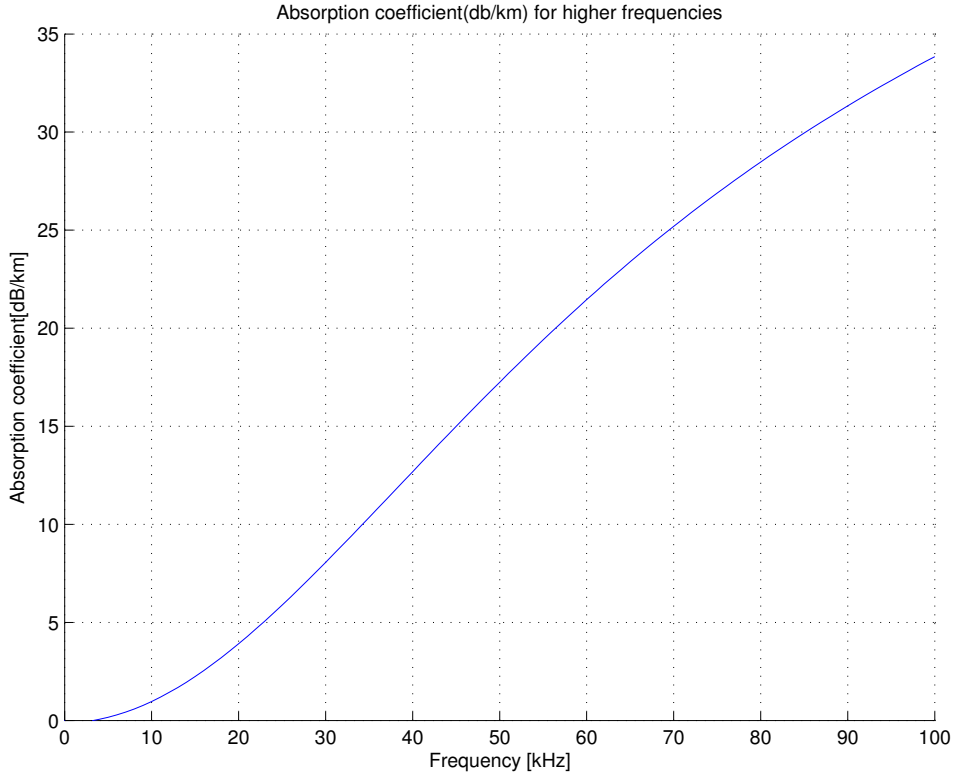


Figure 4.2: Absorption coefficient for higher frequencies from formula 4.4

Ainslie has another approach to finding the absorption coefficient in [2] [1]

$$a_{water} = a_{visc} + a_{chem} \quad (4.6)$$

where a_{chem} takes in consideration the relaxing frequency[2] of boric acid ($B(OH)_3$) and magnesium sulfate ($MgSO_4$) which vary with temperature and salinity. a_{visc} is the absorption due to viscosity of the water. Urick state in [33] that the dominant cause of absorption below 100 kHz is the ionic relaxation of the magnesium sulfate.

[1] Formula (4.6) is based on a simplification from [30] of the formula presented by Francois and Garrison in [31].

[2] Relaxation frequency is the frequency at which the dielectric loss factor reaches a maximum, for a dielectric material that has no static (d.c.) conductivity and that is subjected to an alternating electromagnetic field.[32]

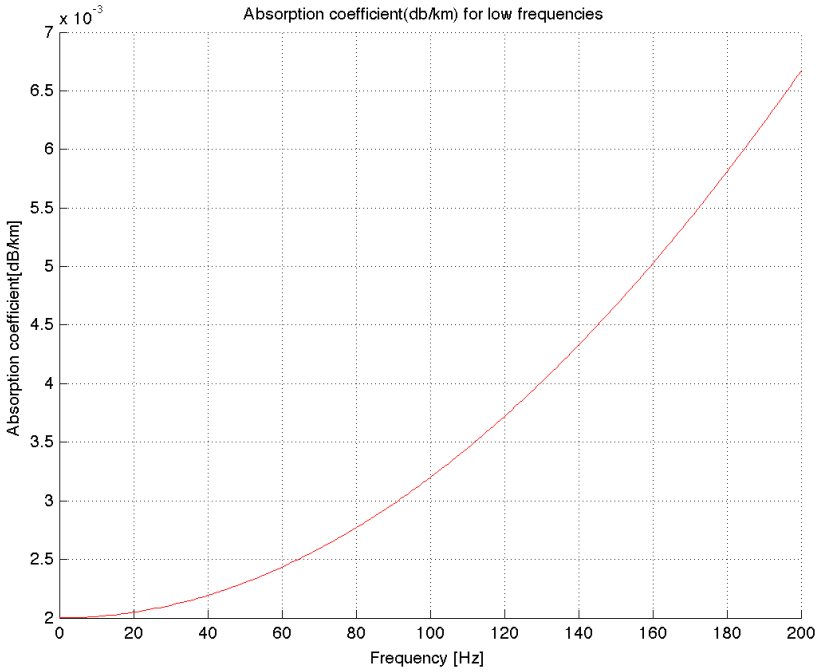


Figure 4.3: Absorption coefficient for lower frequencies from formula 4.5

4.2.1 Noise

There is a lot of ambient noise in the underwater acoustical channel. Ambient noise is always present as a background noise and can be modeled using four sources; turbulence in the water, breaking waves, thermal noise and shipping. These noises can be approximated by using the following empirical formulas (in dB) where f is frequency in kHz [29]:

$$10logN_t(f) = 17 - 30\log f \tag{4.7}$$
$$10logN_s(f) = 40 + 20(s - 0.5) + 26\log f - 60\log(f + 0.03) \tag{4.8}$$
$$10logN_w(f) = 50 + 7.5w^{\frac{1}{2}} + 20\log f - 40\log(f + 0.4) \tag{4.9}$$
$$10logN_{th}(f) = -15 + 20\log f \tag{4.10}$$

Other sources of noise is more varying like e.g. cracking ice in the polar regions or snapping shrimps in warmer waters.

Investigations and measurements done on noise by Wenz in [34] resulted in a Wenz curve as shown in Figure 4.4 [3]. This curve gives a clear overview of what noise is dominant on different frequencies.

[3]Figure 4.4 appears in [34] but this figure is from [35] since it was in color and therefore easier to read

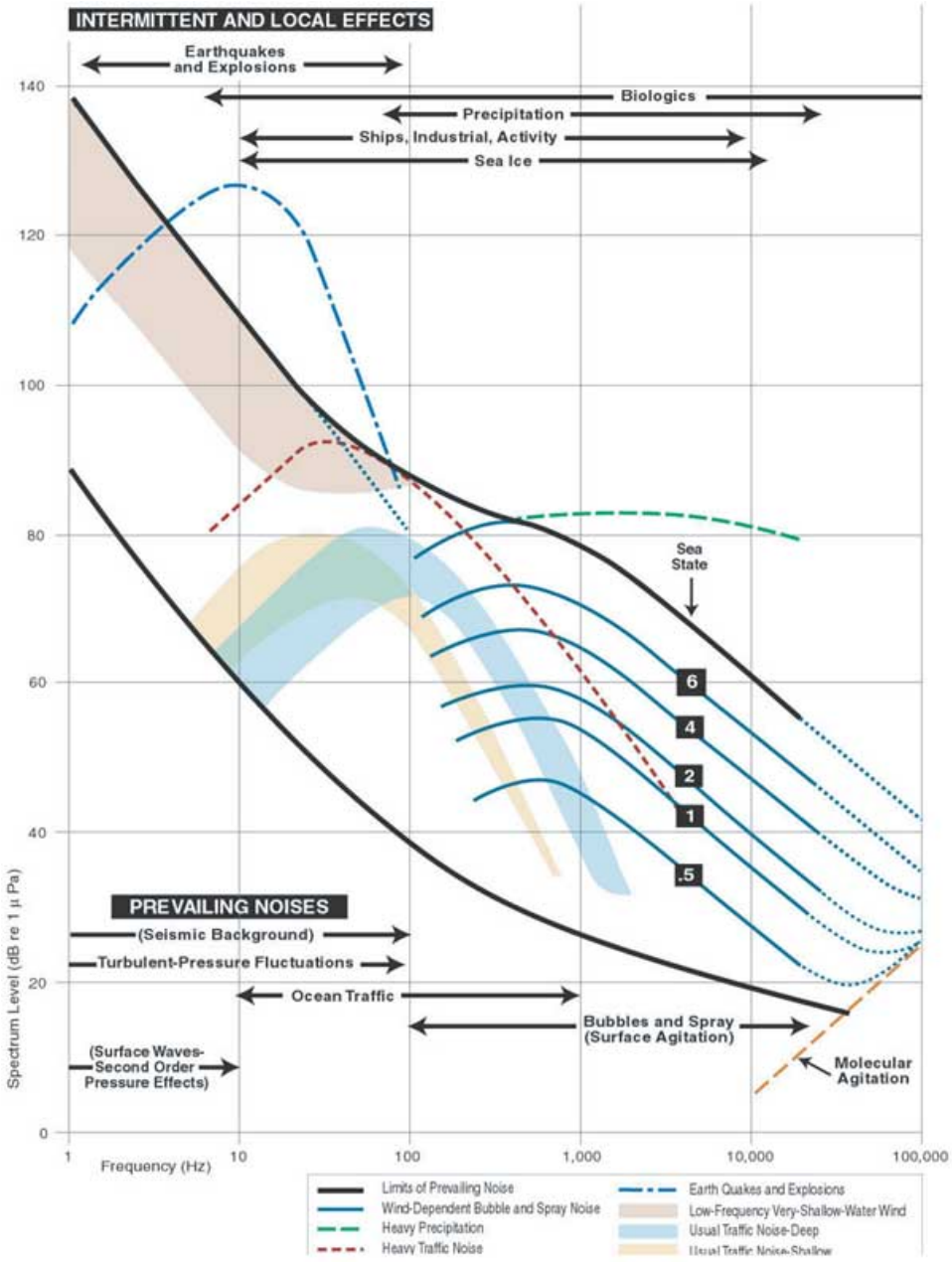
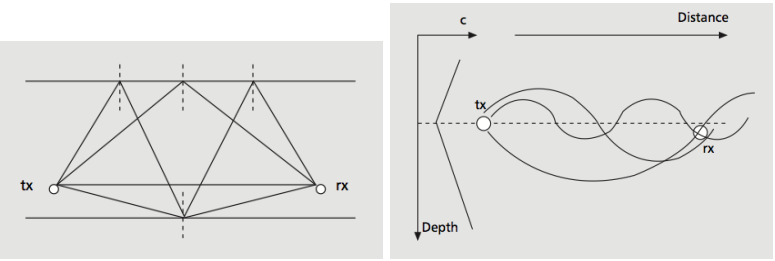


Figure 4.4: Wenz curve showing different sources of noise

4.3 Multipath

As in terrestrial wireless environments, there are multipath effects in the underwater acoustic communication channel as well. There are two main reasons for multipath effect in underwater environments; sound reflections at the surface, bottom or other objects in the water as illustrated in Figure 4.5a, and sound refractions in the water itself [3]. The latter reason is because of the varying sound speed with its variables described in Section 4.1. Sound speed obeys Snell's law bending towards the region

of lower propagation speed, resulting in a multi path effect as illustrated in Figure 4.5b.



(a) Multipath due to reflection on surface and bottom

(b) Multipath due to variating sound speed

Figure 4.5: Underwater multipath, from [3]

4.4 Reflection and scattering

If the sea surface was perfectly smooth, is would form an almost perfect reflector of sound [33]. When the sea is rough (as it is most of the time), the reflection loss is no longer zero. Urick refers to, in [33], a measurement where the sea surface reflection loss is 3dB at 25 kHz in 1-feet (0.3 m) waves. The surface is time varying making it a source of Doppler spread.

The *roughness* or *smoothness* of the surface is given by a *Rayleigh parameter* as $R = kH\sin(\theta)$ where k is the wave number $k = 2\pi/\lambda$, H is wave height, and θ is the grazing angle. When $R \ll 1$ the surface is a reflector, reflecting the signal, but when $R \gg 1$ the surface act as a scatterer sending incoherent energy in all directions [33].

The sea bottom has a more complex structure then the surface because of its multilayer composition. The bottom is also more variable in acoustic properties since it varies in composition from hard rock to soft mud. But it also has similarities to the surface as it acts as a scatterer and reflector.

4.5 Doppler Effect

In every wireless system with mobile nodes, the Doppler effect has to be considered. This causes frequency shift and frequency spreading. The magnitude of the Doppler effect is proportional to the ratio $a = v/c$ [3]. Due to the low speed of sound in water (compared to electromagnetic waves in air) Doppler induced distortion can be large in the underwater acoustic channel. Electromagnetic waves in air have a $c \approx 3 * 10^8 m/s$ which will give a Doppler magnitude of $a = 9.3 * 10^{-8}$ for a station traveling at 100 km/h. This is low enough that Doppler spread can be neglected (it don't have to be explicitly accounted for in symbol synchronization) [3]. In underwater acoustical environment where $c = 1500$ m/s gives $a = 3.4 * 10^{-4}$ for a node moving at 1 knot (0.5 m/s), this effect have a much bigger impact on the system and the Doppler spread have to be considered.

Unlike radio systems, where the time dilation is negligible and Doppler shift appears equal on all subcarriers, in the underwater acoustic channel the subcarriers may suffer from different Doppler shift, making a non-uniform Doppler distortion across the signal bandwidth.

4.6 Bandwidth and frequency

The useable bandwidth in underwater acoustical networks is very limited. In higher frequencies much of the energy is absorbed and the lower frequency bands are affected by noise. So as a rule of thumb, the greater the distance, the larger the absorption and the smaller the bandwidth gets. Compared to terrestrial radio bandwidth is very limited: Akyildiz prints in [17] a table showing the bandwidth versus range in Table 4.1. The frequency band of the modems varies of how long the transmission range its designed to cover, but the Evo Logigs S2CR 7/17 have a frequency band of 7-17 kHz according to the manufacture [36]. Other modems indented for shorter distances have a higher frequency band, but generally below 100 kHz.

Range	**Range [km]**	**Bandwidth [kHz]**
Very Long	1000	< 1
Long	10 - 100	2 -
Medium	1 - 10	$\approx$10
Short	0.1 - 1	20 - 50
Very Short	< 0.1	> 100

Table 4.1: Bandwidth versus range. From [17]

4.7 Summary

The underwater acoustic channel is a challenging channel for transmission of data from one node to another. The variation of the surface, the bottom, temperature and salinity, speed of sound together with ambient and site specific noise, makes it a very hard channel to predict. This combined with a limited useable frequency band makes the total amount of bandwidth relatively small. Despite the large variation in the channel, acoustics is the most suitable way of transmission over larger distances under water. With a though and unpredictable channel the other layers in the OSI stack gets a more important role.

Chapter 5

Physical Layer

As all other layers in the OSI/ISO model, the physical layer has to be simulated as well. This layer is often referred to as Layer 1 (with reference to the OSI model in Figure 6.1). When the packet has travelled from the Application layer, down all the way to the physical layer, the data is no longer a packet or frame, but handled as raw bits. The physical layer may combine bits into bitstreams or symbols upon transmission, and assembles it back at the receiving side. The physical layer also concerns with modulations and coding schemes.

5.1 Physical layer in DESERT

DESERT contains a module to perform simulations of the physical layer. This is done by calculate an attenuation as

$$10\log_{10} A(d,f) = b * 10\log 10(1000d) + d * a(f) \tag{5.1}$$

where b is spreading factor, d is the distance between the receiver and transmitter and $a(f)$ is the Thorp absorption coefficient described in (4.4).

Assuming no interference, DESERT computes a Signal-To-Noise Ratio (SNR) using

$$SNR(d,f) = \frac{P}{A(d,f)N(f)B} \tag{5.2}$$

where P is the transmitter source level $P = 10^{P_{dB}/10}$ (P_{dB} re μPa^2 1 meter away from the source), $N(f)$ is the noise and B is the system bandwidth in Hz. Using the SNR, DESERT calculate a packet error rate (PDR)

$$PER_n(d,f) = 1 - (1 - 0.5erfc(\sqrt{SNR(d,f)}))^L \tag{5.3}$$

where n is noise and L is the packet length. Using $PER_n(d,f)$, DESERT flips a coin to decide if the packet is correct or not. This information is passed upwards the protocol stack, making the higher protocols make decisions in the way they are designed.

In presence of interference, DESERT leverages on the capability to NS-MIRACLE to track the time-varying interference power due to concurrent

transmissions ($I(t)$ in μPa^2), in order to divide a received packet into chunks where the interference is constant. Referring to Figure 5.1, DESERT is finished receiving packet $j-1$ when starting to receive packet j, which last from t_j^s to t_j^e. Assuming interference form 4 other packets, labeled I_1 to I_4. Based on the start and end time of the interfering packets, DESERT divides packet j into c_k chunks where $k = 1, ..., 7$ in Figure 5.1. Each chunk is defined as a time interval where the interference is constant. DESERT then calculates a signal-to-interference-and-noise-radio (SINR) for each chunk

$$SINR_k(d, f) = \frac{P}{A(d, f)N(f)B + I(t_k)} \tag{5.4}$$

which are inserted into (5.3) to yield the probability $PER_k(d, f)$ that chunk k is correctly received or not. The packet is then declared correct if and only if all chunks are correct hence to

$$PER_i(d, f) = 1 - \prod_{k=1}^{c}(1 - PER_k(d, f)) \tag{5.5}$$

where i stands for interference. In order to understand if a packet is corrupted by noise or interference, DESERT first tests whether the packet is corrupted due to noise and if this test passes, it flips a coin to test if any of the chunks are corrupted by interference. If both these test passes, the packet is declared a success and passed upwards the protocol stack.

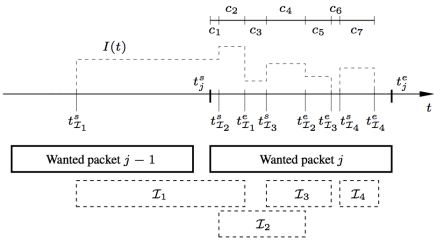


Figure 5.1: Chunk interference model in DESERT. From [4]

5.2 Physical layer in RACUN

Previously in the RACUN project, there were performed a Sea Trial measuring the acoustic underwater channel. Using the results of those measurements, several Look-Up-Tables (LUTs) where made. The modulations used for the Sea Trial is described in Section 9.3, and the LUTs are more described in Section 9.4

Chapter 6

Medium Access Control

The Medium Access Control layer is a sub layer of the Data Link Layer in the OSI/ISO model (Figure 6.1). The MAC layer provides addressing and control mechanism for the shared medium, making it possible for multiple terminals to access the same medium. Today there exist many MAC protocols serving different purposes both for the Radio Frequency (RF) domain and for the underwater acoustical medium. Some of the MAC protocols original designed for the RF domain are also applied to the underwater domain (directly and with modifications). Most of the MAC protocols can be divided into three subgroups; *Time Division Multiple Access*, *Channel Reservation* and *Random Access protocols*.

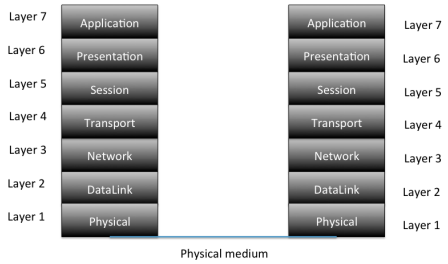


Figure 6.1: The OSI model

6.1 Random Access

In a random access type of protocol the node start to transmit when it has something to transmit. The advantage of this technique is the lack of waiting for the right time to arrive. But this can result in two nodes having something to transmit at the same time, causing a collision at the receiving node. There are several mechanisms to either avoid or detect collisions.

6.1.1 ALOHA

ALOHA is a random access protocol that comes in several flavors. The original ALOHA was introduced by Abramson in 1985 [37]. Here the

node starts to transmit whenever it has something to transmit, as shown in Figure 6.2. This may cause many collisions (depending of number of transmissions in the network), and there is no technique for retransmission or detection of collisions.

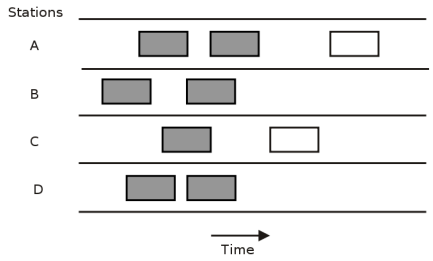


Figure 6.2: The original ALOHA. The nodes start to transmit whenever they have something to transmit. The grey frames are collisions

An improvement to the original ALOHA, is ALOHA with carrier sense (ALOHA-CS). Here the node listens to the medium to make sure that no one is transmitting, before it starts to transmit. The drawback is that if two (or more) nodes listen simultaneously at the silent medium, the transmissions could still start simultaneously and a collision may occur. This is a bigger issue in underwater network than terrestrial due to the long propagation delay.

6.1.2 Slotted ALOHA

Slotted ALOHA (S-ALOHA) is a random access protocol, but instead of the nodes transmitting when they have something to transmit, they wait for the beginning of the next time slot. Unlike TDMA the time slots are not reserved to any nodes. This concept is illustrated in Figure 6.3.

Due to the long propagation delay the time slots have to rather big, depending of the geographical size of the network and the packet size.

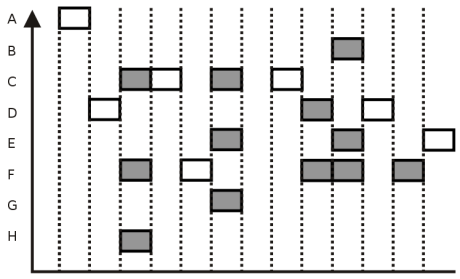


Figure 6.3: SLOTTED ALOHA. The nodes can only transmit at the beginning of a time slot

6.1.3 Carrier Sense Multiple Access

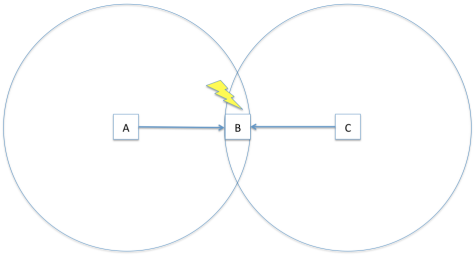
Like ALOHA-CS, carrier sense multiple access (CSMA) senses the shared medium before the node starts its transmission. There are several *persistent* modes of CSMA. *1-persistent* is equivalent to ALOHA-CS; when the node has something to send it senses the medium and if it detects that the medium is free, it transmits. Here it can occur collisions due to two or more node starts listening and transmitting at the same time (after sensing the same amount of time). Another mode is the *p-persistent*. With $0 < p \geq 1$ the probability that the node will transmits after the medium is detected free, is p. Then again, it is 1-p probability that the node will not send but instead wait a pre defined back-off time before sensing the medium again. If the medium is free, the node will transmit with a probability of p. This process is repeated until the frame is transmitted. In *non-persistent* mode the node senses the medium before it starts transmitting. If the medium is busy it waits for a random back-off time (a multiple of the maximum propagation delay) before it starts to sense again. Since the back-offs are multiple of propagation delay, this will result in long waiting times in underwater water acoustical networks.

6.2 Channel Reservation

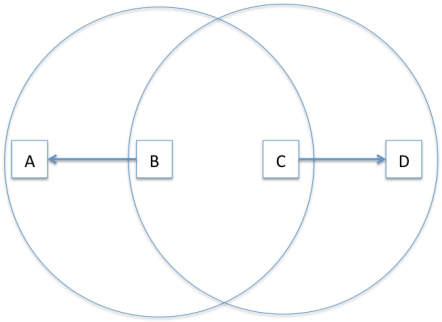
With the channel reservation technique, the node has to reserve the channel before it starts to transmit. This results in fewer collisions, but require a lot more overhead, more control messages and more waiting time. The node performs a handshake before the transmission starts, fighting the *hidden node* and *exposed node* problems. The *hidden node problem* is illustrated in Figure 6.4a; if node C transmits to node B, and node A listens to the medium, then it will not hear the transmission of node C and senses the medium as free and starts the transmission resulting in collisions at node B. In the *exposed node problem* illustrated in Figure 6.4b; node B and C can hear each other but want to transmit to respectively node A and D and this would not be a problem, since the collision occurs at the receiver side. But as one of them senses the medium as busy while the other transmits, the first node has to wait unnecessary.

6.2.1 Multiple Access Collision Avoidance

Multiple Access Collision Avoidance (MACA) was proposed by Karn [38] in 1990 as an attempt to *"finally make single-frequency amateur packet radio networks practical"*. MACA starts the transmitting process by listening to the medium. When the medium is available the node does not start to transmit, but sends a short Ready To Send (RTS) message to the receiver who replies with a Clear To Send (CTS) message. When the sender (also called the initiator since both parties send and receive) receives the CTS from the responder, it starts to transmit its data frame. This handshake method has been implemented by other random access protocols as well



(a) Hidden node problem



(b) Exposed node problem

Figure 6.4: Problems channel reservation solves

and defines the channel reservation category. Doing this handshake the *hidden node problem* (Figure 6.4a) is solved.

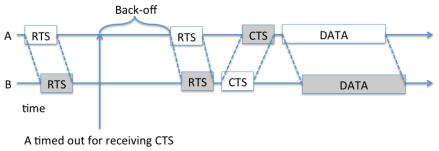


Figure 6.5: the CTS/RTS concept of MACA with back-off period

6.2.2 MACA for Underwater

MACA for Underwater (MACA-U) was proposed by Ng et al. in [39] in 2008 as an adaption of MACA for underwater system. MACA-U has five states where it can be; *IDLE, CONTEND, WFCTS, WFDATA and QUIET*. From *IDLE* the protocol goes into *CONTEND* when it have something to transmit. Here it stays for a defined amount of time before the node transmits a RTS and goes into *wait for CTS* state. The node waits for CTS in $T_{wait} = 2\tau_{max} + T_{CTS}$ where τ_{max} is the maximum propagation delay. When the receiving node sends the CTS it goes into *Wait for DATA* state for a duration of $T_{wait} = 2\tau_{max} + T_{DATA}$. To avoid collisions all the neighboring nodes are in *QUIET* state when overhearing CTS/RTS from other nodes. Another technique done by the MACA-U is in the packet forwarding strategy. To provide end-to-end throughput each node maintains two FIFO

(First in, First out) queues; one for data from the node itself and one for relay. The relay queue has priority. Ng et al. simulate their protocol in [39] and compare it to MACA and ALOHA. They also simulate a carrier sense version of MACA-U (CS-MACA-U).

6.3 Time Division Multiple Access

In Time division multiple access (TDMA) the medium is divided into time slots. Each node in the network gets a certain amount of time slots to send data and another amount of time slots to receive. During this time, the node occupying the medium gets to use the entire frequency band as shown in Figure 6.6. If the node has something to transmit it has to wait to its time slot regardless if other is transmitting.

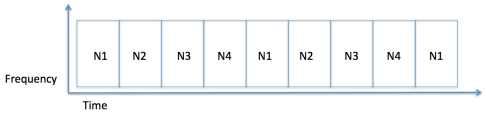


Figure 6.6: TDMA. The different nodes (N) get their time slot to transmit or receive

The length of the time slot T_s has to be transmission time of a data packet T_d plus the propagation delay τ.

$$T_s = T_d + \tau \tag{6.1}$$

According to [40] this gives a channel utilization of

$$T_d = T_d/(T_d + \tau) \tag{6.2}$$

In underwater environment the propagation delay is very high giving $T_d << \tau$ resulting in low channel utilization.

Zhong et al. tries to take advantage of the long propagation delay to improve the traditional TDMA in [40]. They introduce I-TDMA where the nodes interleave the transmission as illustrated in Figure 6.7.

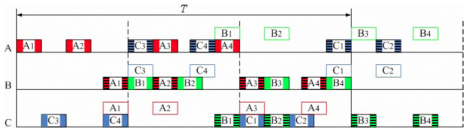


Figure 6.7: I-TDMA. Illustrates how three nodes interleaves

Simulations done in [40] indicates that I-TDMA performs better than TDMA, but this is only when the nodes are at an equal distance to each other and for a maximum of three nodes.

6.4 Summary

Because of the varying sound speed, clock skew and the long propagation delay, a clean TDMA scheme is not very suitable to apply in UANs. A channel reservation scheme with RTS/CTS will be possible to implement, but will result in larger waiting times. If the packets are large enough, then a RTS/CTS scheme may be beneficial. For shorter packets, a random access scheme seems to be the better option for MAC in UAN.

Chapter 7

Network Layer

The network layer is the layer 3 in OSI model (Figure 6.1) and describes the way data is delivered from one node to another, either inside the same network or in another network (inter domain network routing). There are two main methods to do this, either routing or different flooding techniques. Routing defines which way a packet shall take in the network to get to destination using the best path based on *metrics*. One kind of metric can be hop-count and the best path will be the path with fewest hops, while other metrics can be bandwidth, delay, link quality or different combinations. One usually divides routing protocols into two main categories; distance vector (DV) and link state (LS). In a DV protocols the node only knows how far away the target is using different next hop addresses, while in LS the node have information about the entire or partial network (link information), and calculates the best path locally. Because of the long propagation delay underwater the terrestrial routing protocols cannot be applied directly. In terrestrial routing we can assume than the transmission length is bigger than the propagation delay, opposite to the underwater environment.

7.1 Flooding

The simplest form of forwarding data is *flooding*. Here the source node broadcast its packets to all its neighbors who again broadcast to its neighbors and so on. The packets may have a certain Time-To-Live (TTL) and are discarded if this limit is exceeded. This prevents the packets from living eternally in the network. In [41] Rustad proposes a flooding technique for network discovery (NC); a master node initial a flood and as the nodes receives the packet they flood it further, delayed by a random timer. The NC packets contains an accumulated routing cost enabling the nodes to figure out which neighbor has the best path back to the master node. The next step is reporting back to the master node using the ”best-neighbor-path”.

7.2 Proactive Routing

Proactive routing protocols periodically establish and maintain routing tables stored locally on the nodes in the network. This results in not only large amounts of overhead and information packets, but also require some processing cycles on the nodes calculating routing tables. Proactive routing is best suited for static networks not having to update routing tables too often. Examples of proactive routing protocol that exists in WSN today are Optimized Link State Routing (OLSR) [42].

7.3 Reactive routing

Reactive routing is a technique where a node does not store routing information before they need it, but request a route whenever it has data to send. If a node (Q) has data to send to another node (Z) trough a network, it starts by flooding a route request (RREQ) into the network. As the RREQ travels through the network, it stores the path it is sent. When the destined node receives the RREQ it waits for some time to see if more RREQ arrives with a different or better route. When the waiting timer expires, the node responds to the RREQ by sending a route reply (RREP). The RREP packet contains the best path from $Q -> Z$ as illustrated in Figure 7.1.

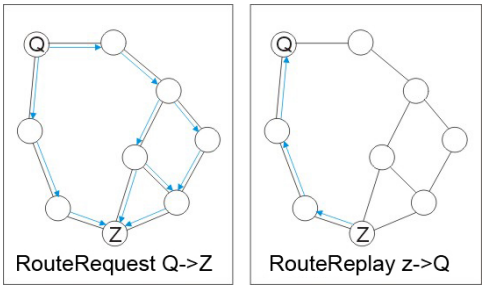


Figure 7.1: Principles of Route request and Route reply

Examples of reactive routing protocols are Ad-hoc Distance Vector (AODV) [43] and Dynamic Source Routing (DSR).

7.4 Geographic Routing

Geographic routing is based on calculating the best route using the shortest geographic distance. One example used in terrestrial networks is the protocol Greedy Perimeter Stateless Routing (GPSR) [44] that uses a greedy algorithm to find the shortest path to the destination. Here the nodes forward the packet to its one-hop neighbor who is closest to the destination, as illustrated in Figure 7.2. The protocol suggested in [45], RGRP, calculates the shortest path based on the total distance from the source to destination. In the geographic routing protocols, the nodes have to know about their positions, and the most common way is to use Global Positioning System (GPS).

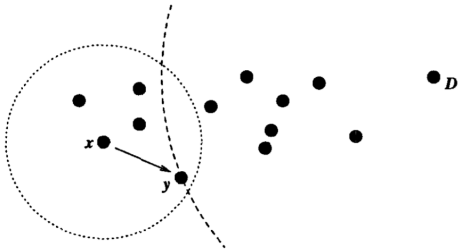


Figure 7.2: A greedy protocol: y is x's closest neighbor to D

In the underwater environment the GPS signal is quickly absorbed and is not suitable. But there are alternatives to GPS. Commercial underwater positioning systems based on acoustical transducers are available. Another way to enabling geographical routing is the use of nodes mounted on the bottom, pre-programmed with their position while the AUVs have some sort of navigation system for multiple purposes. In [46] the authors explore the possibility for making a positioning system under water using the Dilution of Precision (DOP) technique.

7.5 Routing in underwater networks

When it comes to routing in underwater networks there is no typical way of performing this. There are suggested multiple protocols to solve this problem, each focusing on solving their own issue(s) in a certain scenario. In this section some of the suggested protocols are described and some are compared in simulations done by the authors. One thing the most scenarios have in common is that most of the traffic is going *from* sensor nodes towards a sink node (either a relay node with a radio interface or a AUV) and in some cases, some control packets towards the nodes or AUV. The nodes in the different scenarios also vary; some of them are static (attached to the bottom) while others are drifting in the sea current. Some are stationed in the same depth while others take into account different depths.

7.5.1 GUWMANET

Gossiping in Underwater Acoustic Mobile Ad-Hoc Network (GUWMANET) is a routing/forwarding protocol designed for AUN and proposed by Goetz and Nissen in [10]. It is tailored to be used with a specific application language called *Generic Underwater Application Language* (GUWAL), which the authors of [10] also have developed.

GUWMANET uses the GUWAL addressing scheme for forwarding and sending packets. A GUWAL address consists of 6 bits where the first 2 bits are *Group id* and the last 4 bits are defined as *node id*. In addition, GUWMANET introduces a 5 bit local address referred as a nickname. This nickname is unique in the two-hop neighborhood. This last address is

configured by the node itself to assure ad-hoc functionality. When the node is added to network it listens for T_L amount of time, trying to overhear the neighbor's nicknames to avoid conflicts. If no transmissions are overheard the node selects its nickname and generates a nickname notification (NN) packet. If any of the nodes have an objection they reply with a Nickname Collision Notification (NCN).

GUWMANET uses the information in the GUWAL header actively, which makes this protocol only usable with GUWAL parcels. By implementing the application layer with the network layer so close may be give a better performance, but is not very flexible.

GUWMANET is a gossiping protocol using the information it has overheard to make a decision if a received packet is interesting for its neighbors.

7.5.2 Focused Beam Routing

Focused Beam Routing (FBR) is proposed by Jornet et al. in [5] where the nodes try to reach the destination using as low power as possible, concentrating the transmission in one direction. This technique requires that the routing protocol can control or have an influence on the output power and direction. The method is illustrated in Figure 7.3. Here node A is trying to send to node B. Node A starts by sending a RTS to its neighbors using the lowest power setting, P_1 within a cone defined by $\pm\theta/2$. The RTS contains the location of the source node (A) and the destination node (B) and its sent by multicast. In this example there are no nodes within the range so after an expected round-trip-time (RTT) without any answers, node A transmits again using the next power setting P_2. To every power setting P_1 trough P_N there is a corresponding transmission radius d_n. The RTT is calculated using the d_n in (7.1) where c=1500 m/s is the nominal sound speed in water. When a node receives a packet (in the example node C and D), they calculate their location relative to the AB line marked in Figure 7.3 to determine if they are candidates for relaying. Candidate nodes are those that hit the AB line using a cone with an angle of $\pm\theta/2$. If a node determines that it is a relay node it will reply to the RTS with a CTS.

$$RTT_{P_1} = \frac{2 \times d_1}{c} \tag{7.1}$$

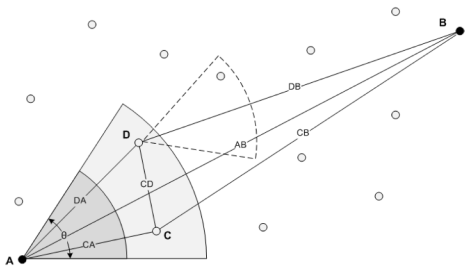


Figure 7.3: Focused Beam Routing [5]

7.5.3 Vector Based Forwarding

Vector Based Forwarding (VBF) is proposed by Xie et al. in [6] is essentially a location based routing protocol. In [6] it is assumed that the nodes have a way to detect the Angle of Arrival (AOA) on the incoming frames. Each packet carries position information about the sender (SP), target (TP) and forwarders (FP). The packet also contains a RANGE field and when arriving at the TP defined the packet is flooded in the area controlled by the RANGE field. The "routing pipe" is defined as a vector from SP to TP with a RADIUS defined. To reduce the number of forwarded packets and for saving energy, the authors introduce the *Desirableness Factor* (α), which favors the nodes closest to the routing vector.

When a node receives a packet it first computes its position and decides if it is in the routing pipe or not. If yes, the node holds the packet for a time $T_{adaption}$ before forwarding.

$$T_{adaption} = \sqrt{\alpha} \times T_{delay} + \frac{R - d}{v_0} \tag{7.2}$$

where T_{delay} is a predefined maximum delay, v_0 is the nominal propagation delay (1500 m/s), R is the transmission range, d is the distance from the forwarding node to the source node of the packet and α is the Desirableness Factor.

If a node receives a duplicate of the same packet during $T_{adaption}$ from other nodes, the forwarding node has to calculate the Desirableness Factor relatively to the original source and the source of the duplicates. If itself has the lowest Desirableness Factor then it forwards the packet.

Figure 7.4 gives an impression of how the routing pipes are computed in a simple network topology where node A, B and C are sending data towards a sink.

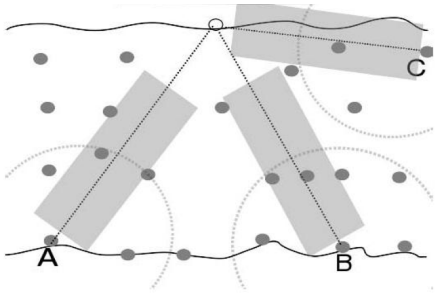


Figure 7.4: Showing the routing pipes selected by VBF [6]

Hop-by-Hop Vector Based Forwarding

Hop-by-Hop Vector Based Forwarding (HH-VBF) is an extension of VBF described in Section 7.5.3 and was proposed by Nicolaou et al. in [7]. Instead of using one virtual pipe (in VBF) from the source to the destination, HH-VBF defines a virtual pipe around per-hop vectors. In this way each node can adaptively make packet-forwarding decisions based

on its current location. Since each node has its own routing pipe then the radius of the pipe will be the nodes transmission range. This will solve the radius sensitivity problem in VBF. Since routing pipes are created per-hop the number of possible paths is bigger, resulting in (according to [7]) increasing packet delivery ratio in sparse networks and decreased amount of energy used.

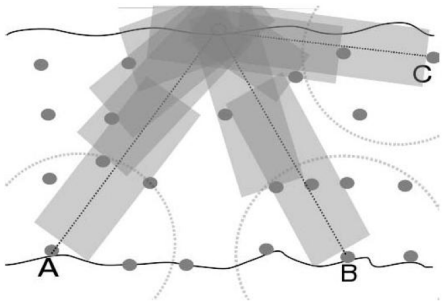


Figure 7.5: The same network as referring to in Figure 7.4 but with per-hop routing pipes [7]

7.5.4 Low Overhead Routing Protocol for Underwater Acoustic Sensor Network

Low Overhead Routing Protocol for Underwater Acoustic Sensor Networks (LOARP) is a routing protocol suggested by Rahman et.al in [8]. LOARP is a on-demand routing protocol containing two protocol operations: *Route Discovery* and *Route Maintenance*. The Route Discovery almost similar to AODV [43] and are not explained. The LOARP header (as shown in Figure 7.6) has a fixed length of 11 bytes as is used for any operation by the type of packet is defined in the *type*-field. The protocol has three types of messages: Route Request (RREQ, type 1), Route Reply (RREP, type 2) and Route Alive (RAVL, type 3). The first two types are used for Route discovery as in [43]. The last type is an optional type serving two purposes: 1) Its used to check where a route is still alive and 2) Helps the Route Recovery process. The nodes also maintain a routing table with entries as shown in Fig 7.7.

Type (1 byte)
Req_ID/Reserved (2 bytes)
Destination IP Address (4 bytes)
Originator IP Address (4 bytes)

Figure 7.6: The header structure of LOARP [8]

When a node S has data to send to node D, it first traverses the routing table looking for an entry. If this is not present, the source node S generates

a RREQ messages with a new Req_ID value and broadcasts it. When node I ($I \neq D$) receives the RREQ (it may receives multiple RREQ from several nodes, but will only process the first received) it will process it, checking if it is its destination. If not, it will re-broadcast it with the values unchanged. When node I receives a RREQ from node P it will consider P the next hop towards S and create a routing table entry. When D receives the RREQ it will generate a RREP and unicast it to node Q where node Q is the node provided D with the first RREQ. When node I ($I \neq S$) receives a RREP from X it will create a routing entry towards D with next hop X and forwards the RREP towards S. When S receives the RREP a bi-directional route from S -¿ D exist as shown in Fig 7.8.

Destination IP Address (4 bytes)
Next Hop IP Address (4 bytes)

Figure 7.7: A routing table entry in LOARP [8]

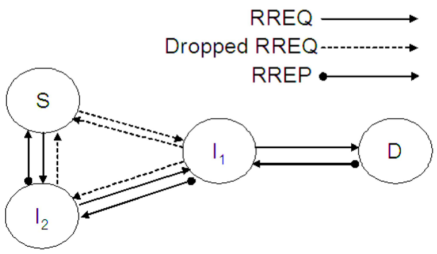


Figure 7.8: A bi-directional route from S to D is established

Rahman et.al have in [8] done some simulations of their proposed protocol, comparing it with other Ad-hoc routing protocols. And as it seems the LOAPR, performs better than the other with respect to throughput, packet delivery ratio and control overhead. Its worth mentioning that the compared protocols are designed for terrestrial networks. All the protocols are given the same parameters (e.g. propagation delay) but for some the protocols, very ”unnatrual” values are used, as they are created for terrestrial environments.

7.5.5 Mobicast Routing

The feature of Mobicast (or ”spatiotemporal multicast”) is as described in [47], the delivery of information to all the nodes in a given area in a certain time. This technique can be used as a basis for a routing protocol as suggested by Chen and Lin in [47]. Here the authors describes a scenario where a number of nodes are deployed in a 3D environment, floating with the current, and an AUV circulating around the nodes collecting data as nodes are covered. All the nodes in the 3D geographic zone reachable

$$Z_t(N_i) = (X_i - X_A)^2 + (Y_i - Y_A)^2 + (Z_i - Z_A)^2 - R^2 = 0$$

Figure 7.9: N_i is a single sensor node (X_i, Y_i, Z_i) and the center of the ZOR (AUV) is X_A, Y_A, Z_A

by the AUV is referred to a 3D Zone of Reference (3D-ZOR). As shown in Figure 7.10 ZOR_t^3 is covered at the time t and ZOR_{t+1}^3 is covered at time t+1. The AUV travels a pre-defined route collecting data from the nodes in the different ZOR's. In the routing process, the first step is that the AUV defines the ZOR_t^3 and broadcasts a control packet to the nodes covered in ZOR_t^3. This packet wakes the nodes from sleep mode. The AUV calculates the ZOR using the formula shown in Figure 7.9 where the radius ($R = hop\_distance \times h$) is defined by the communication range (Hop distance) and h is an integer that is defined by a user to indicate an expected range for data collection.

When a sensor node receives a control packet from the AUV, and if it calculates that it exist inside the defined ZOR_t^3, it will start to transmit it sensed data to the AUV. Then ZOR_{t+1}^3 is defined and the nodes go back to sleep to save power. At some point along the path of the AUV, it will get in range with the control station and will deliver all the gathered sensor data. Here the AUV operate as a "postman", gathering and delivering packets as it travels around.

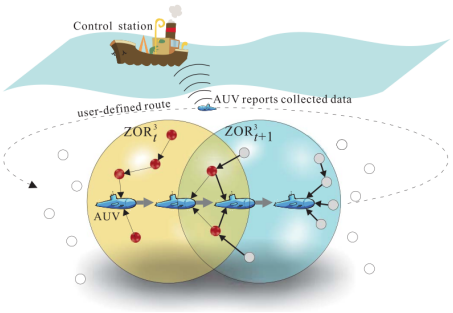


Figure 7.10: Mobicast

7.5.6 Link-state based Adaptive Feedback Routing

Link-state based Adaptive Feedback Routing (LAFR) proposed by Zhang et al. in [48] addresses the problems due to beam width of the sensor nodes and the 3D problem (that underwater nodes are in a 3D area). The authors claim that this results in asymmetric links. In the LAFR protocol the nodes define itself as a *reference* node and keeps a table of *upstream nodes* and *downstream nodes* as illustrated in Figure 7.11. The nodes keeps an upstream table and a downstream table, maintained with *Link detection* messages sent periodically. The message is illustrated in Figure 7.12. When a node receives a link detection message it puts the sender ID into the upstream table and checks if senders upstream table contains the receiving nodes

ID. If this is the case, then the receiving node puts the senders ID into the downstream node table.

If the node needs a route to a destination that is not in the nodes local routing table, the node has to send a routing request packet (as illustrated in Figure 7.13). The fields in the routing information are described below

Relay node ID
The ID of the relay node

Link State Information
Determents if the link is symmetrical or asymmetrical

Receiver-side SNR
The signal to noise ratio of the corresponding relay node side

Relay node surplus energy
The residual energy of the corresponding relay node

During the routing inquiry process the nodes uses a priority forward mechanism and using the Sender ID and sequence number, only forwards the first routing request. This is for energy saving purposes.

When a sink receives a route request packet it will wait for a period to accept other incoming route request packets (from other routes) and use this information to assemble a route back. If the route (all the links) is symmetric, the sink node can simply replies back using the collected route. If the route is asymmetric then the sink node has to send a route request back to the source. When it comes to route optimization and selection of routes, the LAFR makes a score of each route $k = f(e, d, snr)$ where ”e” is the residual energy, ”d” is the total length of the route (hop-count) and ”snr” is the signal-to-noise ratio (the sum of SNR from source to destination).

Due to performance, the authors mainly compare their protocol with the one described in section 7.5.7. In those simulations the LAFR protocol outperforms the DBR due to packet delivery ratio and energy consumption, but increases the end-to-end delay.

The LAFR uses as mentioned SNR and energy levels to calculate its best routes. How this information is achieved is not described in the article

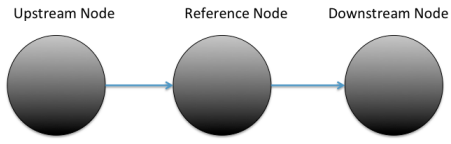


Figure 7.11: Nodes definitions in LAFR

7.5.7 Depth Based Routing

Depth Based Routing (DBR) is suggested by Zhang et al. in [49] and uses a greedy routing algorithm trying to deliver packets from source nodes

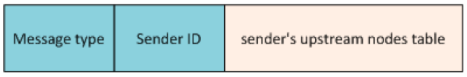


Figure 7.12: Link detection packet in LAFR

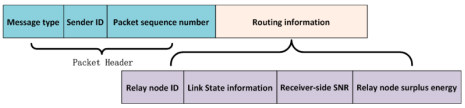


Figure 7.13: The routing request packet in LAFR

to sinks. The protocol assumes that the sink is located at the surface and therefore only forwards packet coming from a node with a lower depth than itself.

When a node N is receiving a packet is looks at the *depth* d_r field in the header (illustrated in Figure 7.14) and compares it with its own depth d_n. If $d_r > d_n$ then the node N is closer to the surface and consider itself as qualified to forward the data. If not, the packet is dropped because the node that sent forwarded the packet is a better node, e.g. closer to the surface.

DBR maintains a *priority queue*, Q1, and a *packet history buffer*, Q2. When a node receives a packet it holds it for a holding time. This holding time indicates the priority in Q1. When a packet is sent successfully the packet sequence number is put into Q2 to reduce redundant packet transmissions.

The authors have done some simulations simulating different parameters and compares DBR with one and multiple sinks with the VBF (described in Section 7.5.3). The simulations shows that VBF performs well in dense networks but not so well in spare networks. According to simulations done only the DBR with multiple sinks performs better then VBF (du to packet delivery ratio and end-to-end delay). But in energy consumption and in multiple sink mode, the DBR outperformance VBF.

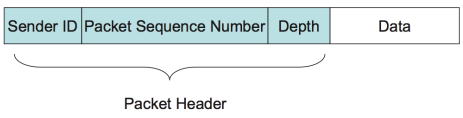


Figure 7.14: DBR header

7.5.8 Reliable Energy-efficient Routing Protocol based on physical distance and residual energy

Reliable Energy-efficient Routing Protocol based on physical distance and residual energy ($R - ERP^2R$) is a routing protocol suggested by Wahid et al. in [50]. $R - ERP^2R$s main idea is to use physical distance as metric and balance energy consumption among the sensors. $R - ERP^2R$ is not a geographically based routing protocol, so it has to measure the physical distance. The ETX is a well-known measurement used in both underwater

networks and terrestrial sensor networks. ETX is defined as

$$ETX = \frac{1}{d_f \times d_r} \tag{7.3}$$

Where d_f is the delivery ratio from node A to node B and d_r is vice versa. To compute ETX the nodes have to exchange HELO packets with their neighbors after a specified interval for a predefined duration, CHECK_TIME. CHECK_TIME is the time period defined for exchanging HELO packets. All the nodes know the value of the HELO packets and CHECK_TIME. After CHECK_TIME the nodes calculates the delivery ratio and the ETX. To compute the physical distance the sink node broadcast a HELO packet. The receiving nodes uses the Time of Arrival (ToA)/Time Difference of Arrival (TDoA) to calculate the distance. Then the nodes re-broadcast the packet, now including the calculated distance. The next node calculates the distance and adds the last distance:

$$D_{N,S} = D_{N,N-1} + D_{N-1,S} \tag{7.4}$$

where $D_{N,S}$ is the distance to the sink and $D_{N,N-1}$ is the distance to the node forwarding the packet.

During the forwarding data phase each sender selects its forwarding node that has to be closer to the sink node. The sender calculates the cost to all its neighbors and selects the neighbor with the lowest cost. When a node receives a packet it takes a look at the ID in header and matches it against its own ID. Only if the IDs are matching the node will process the packet. Due to movement of the nodes (sea current) the distance may change over time. The solution to this is to periodically update the cost in the same way as in the initiation phase.

The authors have simulated their protocol comparing it with other known protocol e.g. DBR as described in Section 7.5.7. When it comes to network lifetime (time before the batteries in the nodes are exhausted) $R - ERP^2R$ outperforms DBR in all the different topologies. The same is said when it comes to end-to-end delay. In a grid formed (versus a random distribution) network the DBR performs better than $R - ERP^2R$ for a number of nodes up to about 100. Other than that $R - ERP^2R$ performs a higher packet delivery rate then DBR.

This protocol relays on time measurement to calculate the distance to a neighbor node, which again requires synchronized clocks. Synchronizing clocks in underwater networks are a difficult task to perform due to the long and varying propagation delay.

7.6 Flooding techniques in underwater networks

Flooding can be applied to an underwater network, as well as a terrestrial. The flooding technique is described in Section 7.1. This section describes some protocols proposed, based on the flooding technique.

7.6.1 Directional Flooding

Directional Flooding-based Routing (DFR) is a proposed protocol by Hwang and Kim in [9] based on limited flooding. When a node has a packet to send it floods it towards the sink but limits the flood to a *flooding zone*. The authors assume that all the nodes know their location and the location of the nodes in the two-hop neighborhood as well as the location of the sink. Another assumption made, is that the nodes can measure the link quality among neighbors.

When a node has a packet to send, it broadcasts the packet into the network with a BASE_ANGLE and its current location. When a node within the transmission range receives such packet, it looks at the packet, comparing its BASE_ANGLE with the node's CURRENT_ANGLE (the angle between the source and itself). If the forwarding node's CURRENT_ANGLE is smaller than BASE_ANGLE then the packet is discarded because the forwarding node sees itself outside the flooding scope.

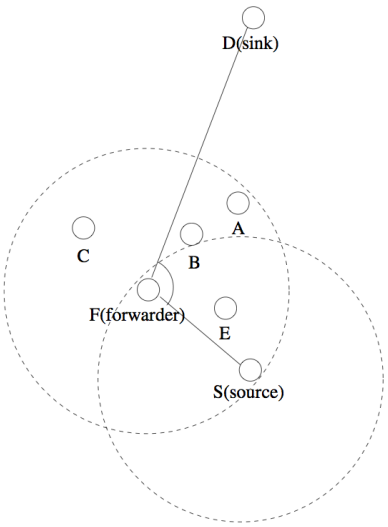


Figure 7.15: Directional flooding [9]

In the example illustrated in Figure 7.15, node S broadcasts its packet with a BASE_ANGLE set to A_MIN (a predefined minimum angle) and its location. Node F then receives the packet and if F's CURRENT_ANGLE (the angle between $\overrightarrow{FS}$ and $\overrightarrow{FD}$) is larger than the BASE_ANGLE included in the packet, then node F adjusts the BASE_ANGLE according to the average link quality with A and B. Then node F floods the packet with S's location and the new BASE_ANGLE.

7.6.2 Reduced duplication flooding

Otnes and Haavik propose in [24] a protocol that aims to reduce the number of duplicates transmitted in a network using flooding as a forward mechanism. Here the packets have a sequence number and this combined

with source and destination address forms a unique identifier used to recognize duplicate packets. This protocol is named Dflood and will be subject to improvements later in the thesis, so this protocol is described more in detail in Chapter 10.

7.7 Security in underwater networks

As in every network, and especially wireless network, network security is an issue that has to be handled somehow. It can be implemented on one or more layers in the protocol stack, making the system more or less secure. In acoustic underwater networks, the focus has been to improve the capacity, reliability and robustness, rather than securing the network. This is a logical approach, since it is rather pointless to secure a network with already poor performance (as security cost resources, either in terms of processing power of bandwidth requirements). Most underwater networks are sensor networks, sensing and reporting an environment, either it is ocean current, temperature or intruding hostile vessels. Some of this information might not be critical if tampered with, but other information, like e.g. real-time intelligence sensing, might be highly delicate and worth protection.

There are many ways to attack a network, but for underwater acoustic networks we can divide them into two interesting groups; Denial of Service (DoS) and Data tampering/Man-in-the-middle. A DoS attack has the purpose of making the network unavailable for everyone. This type of attack can consist of one or multiple attackers, knows as Distributed Denial-of-Service (DDoS). The attacker(s) floods the network with traffic, either empty/uninteresting data packets or control packets, overloading the nodes with work to process the bogus packets making no room for them to process the real data packets. The bogus packets can seem very real to the nodes, not realizing they are under attack. In sensor networks these attacks may also have the intention to drain battery.

In data tampering (DT) or Man-In-The-middle (MITM) attack, one or more hostile node(s) are inserted into the network. The hostile node captures packets transmitted in the network, tampering with the data, before forwarding the packet towards its original destination (or another destination). In these attacks, the hostile node often pretends to be a central or important node, like e.g. a gateway. One way to obtain this central role is to offer good parameters to the network, like e.g. low delay towards the gateway, making the other nodes choosing the bogus node as its preferred routing/forwarding. The offered parameters can be fake or real, like the low delay offered in a wormhole attack.

A wormhole is a physics definition and is according to [51] ”A wormhole, also known as an Einstein–Rosen bridge, is a hypothetical topological feature of space-time that would be, fundamentally, a ”shortcut” through space-time”. In an underwater wormhole attack you make a bridge bypassing the long propagation delay with e.g. two nodes connected with wire or RF above the surface as described in [52]. Here the network protocol will

most likely (sometimes after a while) select the route via the wormhole since it is faster, making the attacker, who controls the nodes forming the wormhole, have access to all the data. Another form of man-in-the-middle attack underwater is jam-and-replay. Here the hostile node receives the packet and jams the transmission so other nodes don't hear is. Then the hostile node does whatever it wants with the packet before retransmitting it.

7.7.1 SeFLOOD

In [53] Dini and Duca proposes a network discovery protocol called Secure Flood (SeFLOOD). SeFlood is based on the FLOOD protocol suggested in [41] and focuses on securing it from spoofing- based DoS attacks and spoofing-based attacks against integrity. The way that security is achieved in SeFLOOD is that each pair of nodes in the network (i,j, $i \neq j$) share a *Link Key* K_{ij} which is used to protect unicast messages between node i and j. The nodes are spilt into one or several clusters that are in the same broadcast domain. Then each node generates a *cluster key* K_i and distributes this to all the other nodes in the cluster. Each node maintains a *Cluster Key Table (CKT)* containing all the other nodes keys. Then the node i uses its K_i to encrypt its broadcasts messages and uses K_j to authenticate messages from node j.

7.8 Flat and hierarchal routing

In a flat routing scenario all the routers/nodes are peered to each other, while in hierarchical routing one or more router(s) form a backbone. In hierarchical routing scheme, a packet from a non-backbone router has to travel first to a backbone router before it is forwarded. In larger networks it is not uncommon to define logical communities/areas/domains and in the hierarchical model only some routers can communicate between domains (but inside the domain it can be a flat routing structure) [54].

7.9 Cross layer network

There are two ways to approach a good networking strategy. There is the layered approached, using the OSI/ISO model as shown if Figure 6.1, making the network layer independent of the other layers. The other method is a cross-layer approach. This method is more dependent on the other layers, either upwards in the OSI model, or downwards, making it a less modular system. Making one layer at the protocol stack dependent of others makes the protocol less flexible. If. e.g. the network protocol builds a routing table based on the best SNR from the neighbors it may collect this information from the data link - or the physical layer. This information can be used to optimize the routing, but if e.g. the MAC protocol is switched to one that does not provide the information required by the routing protocol, the routing protocol will be

useless. GUWMANET (described in Section 7.5.1) uses the parcels from the application language GUWAL directly making the connection between those very tight. In fact, GUWMANET can only be used together with GUWAL. This dependency reduces the GUWMANET network overhead and gain network performance, but means that it cannot be used with other application layer protocols.

Chapter 8

RACUN

Robust Acoustic Communications in Underwater Networks (RACUN) is a project under the umbrella of the European Defense Agency (EDA) and sponsored by national Departments of Defense [55]. The project has partners from Germany, Netherlands, Norway, Italy and Sweden and was started in 2010 and is ending in 2014.

8.1 Work of RACUN

The RACUN working group has done a large amount of work in their pursuit for a robust acoustic underwater network. Some of the work includes a careful study of the underwater channel and measuring of this in multiple environment and settings. These measurements have been used to create look up tables (LUT) for simulation of the physical layer. The measured signal is inserted into a channel *estimator* and further into a channel *simulator*. Noise is added and the original message is compared to the one measured for comparison and calculations of bit error rate (BER) and packet error rate (PER). A more detailed overview of the process is illustrated in Figure 8.1.

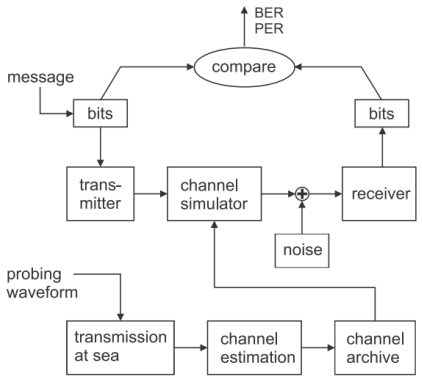


Figure 8.1: How the LUTs are created based on measured channels. From [4].

8.2 Scenarios

The objective and scenarios to be reviewed in the project are rooted in user requirements from the different navies. The operational needs from the navies have been expressed in terms of four reference scenarios [55]:

- General Purpose communication network between static and mobile nodes
- Establishing and maintain a safe operating are
- Forward ISR (Intelligence, Surveillance, Reconnaissance)
- Mine reconnaissance using several AUV's

For all these scenarios the focus is set on littoral waters and is specified in [15].

Most of these scenarios serve the same point for the networking part; the nodes need to handle multi-hop networking and they need to make decisions on how to handle received packets (where and when to send, or should they be discarded).

From the network layer point of view, most of the scenarios serve the same points. Most of the nodes need to handle multi-hop networking (a challenge for the MAC-layer) and the nodes need to make decisions on how to handle received packets; where to send it, when to send or discard it.

Chapter 9

Simulation Framework

The RACUN project had at the start of this thesis work implemented a complete simulation framework based on NS2 (described in Section 3.2), and this was used for the simulations in this thesis. Some modifications were made to some of the protocols and simulations scripts.

9.1 Protocol Stack

The protocol stack used in the simulations is a set of modules from the DESERT library and some protocols written for the RACUN project by its participants. A brief overview of the protocol stack is illustrated in Figure 9.1.

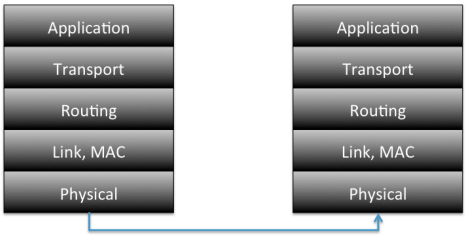


Figure 9.1: An overview of the protocol stack used in the simulations

In detail the different layers contains the modules described below [4]

- Application layer
    1. GUWAL application layer
    2. GUWAL wrapper
    3. Application-DESERT interface
- Transport layer
    4. UPD
- Routing layer

5. Multicast filter
6. Routing/forwarding module (in most cases Dflood)

- Link, MAC and physical layer

7. Convergence layer
8. Adaption layer
9. DESERT-Physical interface
10. Modem module (only simulation)
11. Physical layer module (only simulation)

The Generic UnderWater Application Language (GUWAL) forms the messages (described in Section 9.2) from the application layer. The *GUWAL wrapper* and *Application-DESERT interface* converts the GUWAL messages into a format that suits DESERT.

UDP is a DESERT module that provides basic transport layer functionality, like port multiplexing and filtering of duplicate packets from lower layers towards the application layer

The *Routing/forwarding module* manages all actions regarding forwarding packets towards their destination. Here there are multiple options, but in the simulations in this thesis, the *Dflood* and *GUWMANET* protocols are investigated. Depending on scenario and application requirements, a node may need to send multicast messages. The *multicast filter* interacts with the *Routing/forwarding module* by stopping packet replication once a packet reaches the prescribed multicast group in scenarios where multicast groups are defined [4].

Adaption layer (AL) is a module from DESERT, making the core of the serialization process, converting NS-MIRACLE structures to bit-streams and vice-versa. The module performs three functions;

- Bit stream creation based on rules defined by each module included in the node.
- Packet fragmentation to match the PSDU size allowed by the acoustical modem
- Re-assembling of received fragments into packet that can be re-converted to internal NS-MIRACLE data structures.

Together with the AL the *convergence layer* works to avoid the serialization of data can be inferred from other fields. The packets (or fragments) ready to be transmitted are managed by the *DESERT-Physical module* that implements basic MAC functionalities (ALOHA [4]) and communicates with the modem.

The two last modules in the protocol stack are used in the simulations, but are replaced with real acoustical modem and the real-life underwater acoustical channel in sea trial. The *modem module* emulates the modem behavior and interactions with *DESERT-Physical interface*. The *Physical layer* is simulated using either a module from DESERT or the RACUN-specific Look Up Tables (see Section 9.4)

9.2 GUWAL

The application layer used in the simulation is the *Generic UnderWater Application Language* (GUWAL) [10]. GUWAL supports four types of parcels:

- Data request
- Data (sensor data, status, position, ...)
- Command and Control (Sleep, move, change mode, ...)
- Text Message

The parcel is 128 bits long, but can have a variable length if required. The format of the GUWAL parcel is illustrated in 9.2. GUWAL uses a 6-bit operational address where the two first bits indicates type of node; *Gateway Node*, *Bottom Node*, *Mobile node* or *Surface or Air node*. This leaves the last four bits to addressing the nodes (all zeros are broadcast to nodes of the same type. All 6 bits set to 1 is broadcast to every node in the network). This addressing scheme results in 15 usable addresses in group 1-3 and 14 in group 4, leaving the total number of unique addresses to 59. But using this scheme also resulting in having 15 bottom node addresses as well as 15 gateway node addresses is not very efficient use of either addresses or gateway nodes. So in larger topologies not containing 15 gateway nodes, the addressing may need to be done in a different way, e.g. borrowing gateway addresses for bottom nodes. In combination with GUWMANET described in 7.5.1 the authors proposes that is it possible for nodes to share addresses, using the addresses as multicast groups.

Position	Length	Field
1-2	2	Parcel Type
3	1	End-to-End Acknowledgment
4	1	Priority Flag
5-10	6	Operational Source Address
11-16	6	Operational Destination Address
17-112	96	*Payload*
113-128	16	Checksum

Figure 9.2: Format of GUWAL parcel from [10]

9.3 Modulations

In the simulations there are two different physical layer models. One of them is the model available in DESERT (described in Section 5.1) and uses a BPSK model. The others are based on previous work and sea-trials performed by the RACUN-partners. The other modulations are described below and the simulation are based on Look-Up-Tables made from measurements from a sea-trial.

9.3.1 BPSK

Binary phase shift keying (BPSK) is the simplest PSK modulation. The phase is modulated to represent different binary values, and in BPSK the phases are separated by 180° making the symbol length 1-bit as illustrated in Figure 9.3.

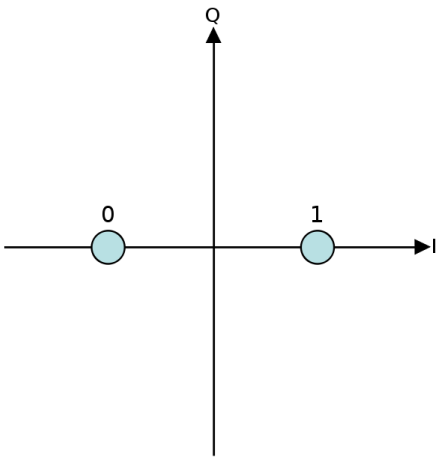


Figure 9.3: BPSK. The phase decides witch symbol that arrives. From [11]

9.3.2 OFDM

Orthogonally frequency division multiplexing (OFDM) is a specialized Frequency division multiplexing (FDM) where all the carrier signals are orthogonal to each other. OFDM divides the available bandwidth into several small sub-bands as illustrated in Figure 9.4. The OFDM transmits one symbol per sub-band, resulting in sending many slow symbols simultaneously instead of one fast symbol at the time. The symbol duration is longer than the channel's multipath delay, making OFDM robust to inter-symbol interference (ISI) caused by multi-path.

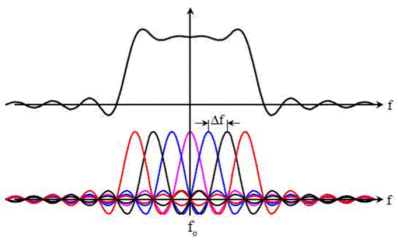


Figure 9.4: OFDM: The available bandwidth is divided into several sub-bands. From [12]

9.3.3 FMT

Filtered Multi tone (FMT) is a hybrid between singe-carrier transmissions and OFDM [13]. FMT divide the available bandwidth, B, into M separate frequency bands with bandwidth $\frac{B}{M}$ onto which single carrier signals are modulate to create a multi band waveform. Raised cosine pulses are used to achieve the best spectral efficiency, this in contrast to OFDM, which have rather large overlaps in the spectrum. The symbol to be transmitted are oversampled and then filtered by a band pass filter to fit to the bandwidth. The difference between OFDM and FMT is illustrated in Figure 9.5

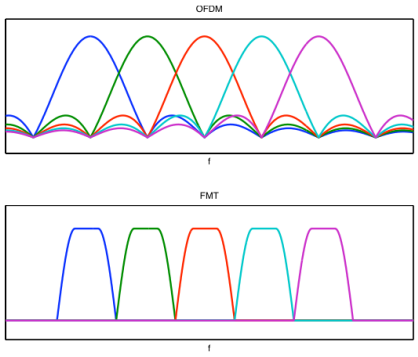
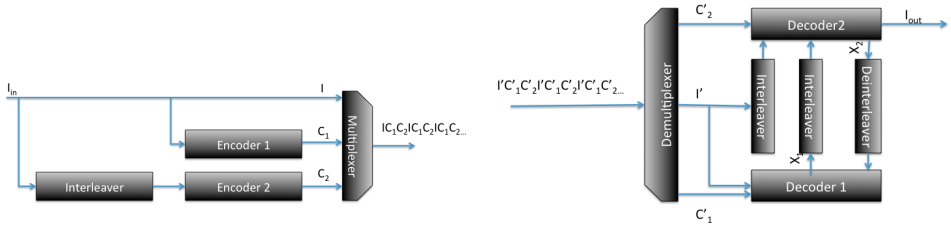


Figure 9.5: The difference between OFDM and FMT. From [13]

9.3.4 SCTE

In Single Carrier Turbo-Equation (SCTE) the information is Turbo encoded and transmitted on a single carrier. Turbo coding is a coding scheme that preform well due to bit error probability and is very close to the Shannon limit. Most turbo codes are based on convolutional codes [14]. One example of a Turbo encoder is illustrated in Figure 9.6a. Here the decoder produces a check bit, C_1 , for each input bit, I, and another check bit, C_2, which is interleaved. This gives a code rate of $1/3$.



(a) Encoder (b) Decoder

Figure 9.6: An example of turbo encoding and decoding from [14]

9.4 Look Up Tables

In the first sea trial of the RACUN project, measurements of the underwater acoustical channel were done. The results were used to create LUTs that can be used by the simulation framework (see Section 8.1). The LUTs have been anonymized because the information is internal to RACUN. One of the parameter settings that are varied in the LUT creations is the frame size, resulting in that two of the LUTs have to fragment the packet/frame and transmit it in two burst (do not apply to GUWMANET). The frame size the modem handles, may vary from modem to modem, but the size used in this simulation are anonymized. This results in some of the LUTs referred to will fragment the packets in some of the protocols. An overview is given in Table 9.1

	DFLOOD	**GUWMANET**
LUT 1	Fragmentation	No fragmentation
LUT 2	Fragmentation	No fragmentation
LUT 3	No Fragmentation	No fragmentation
LUT 4	No Fragmentation	No fragmentation

Table 9.1: Overview of LUT and fragmentation

In Table 9.1 fragmentation means that the packet have to be transmitted in *two* fragments and also have to be reconstructed at the receiver side, because the PSDU (Physical Layer Service Data Unit; the size of the frame in the physical layer) is to small to carry both protocol header and data payload. This can have some influence on the performance since the first fragment can be error free, but if the second fragment contains an error, then the packet is corrupt and dropped.

Figure 9.7 shows an illustration of the difference in packet error rate (PER) with the channel estimation found in DESERT and a LUT

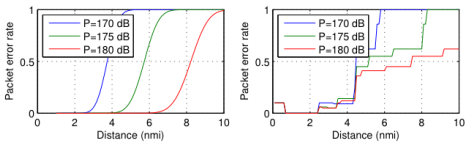


Figure 9.7: Left: PER using BPSK based model from DESERT. Right: PER using a LUT. From [4]

9.5 Simulation scripts

The simulation setup contains multiple TCL/OTCL script, each script handles on part of the simulations, nested together in one ”main” script. The main script handles the input parameters that are interesting to vary. This setup is based on the simulation setup used in RACUN. For varying the input parameters the main simulation script is again controlled by

a bash script, looping through a set of parameters, and running the simulations, one at the time. The interesting output is stored in files that are later on analyzed and evaluated.

The main idea of dividing the simulations into multiple scripts is to reuse code, and to make the results more comparable (e.g. the same script is used to create the nodes in all scenarios, so a change in this will affect all the scenarios equally)

Run scenario script is a bash script for running one or multiple scenarios with different input parameters. Examples of input parameters will be transmit power or retransmission parameters to Dflood (as described in Section 10.2)

Scenario script is the ”main” script that runs the scenario specific simulation.

Position script is a scenario specific script that contains information about node position given in WGS84 coordinates. Its not necessary to use real-life position, like WGS84, as long as the nodes have the same position in the coordinate system (the WGS84 coordinates have in other words no relevance to the simulations. It is the distance and position relative to the other nodes that is relevant.).

Node Setup script handles creating the nodes. Here the ns2-nodes are created and attached the relevant modules.

Address script that handles address creating for the different modules

9.6 Evaluation Criteria

The main evaluation criteria for a robust communication (in the network layer) is the Packet Delivery Ratio (PDR) and Packet Error Rate (PER) where $PER = 1 - PDR$. This will be the main evaluation criteria. Delay and energy usage is also considered.

Another criteria to measure robustness will be the ability to handle ad-hoc situations with varying number of nodes. If a node disappear from the network (either death-by-battery, hardware failure, maintenance or in military applications the nodes can be jammed, destroyed or removed by an enemy) or more nodes are added (to achieve better performance, increase sensing capabilities or other purposes) the network should be able to handle this changes to the best. The nodes have their limitations (e.g. transmission range) so if a network has a single point of failure, and this is removed, it is not expected from the network to achieve full functionality.

PDR varied over time (as nodes appears/disappears) is used as numerical evaluation criteria for measuring ad-hoc capabilities. This will give an indication of how fast the new networks converges. A weakness with this evaluation criteria is the variation of traffic in time. This is solved using a constant bit stream (CBR) where traffic is generated an transmitted from the node with a constant interval for a given time.

9.7 Environment

Since the simulation framework is developed for Linux/Unix it was natural to run on a Linux platform. Multiple options were considered, including running the framework on a MAC/Unix native environment and Windows environment using Cygwin (a Linux emulator). The third option was using a clean Linux install, which was the better and easier choice.

The simulations were run on Linux Ubuntu 12.04.4 LTS with kernel 3.5.0-46-generic. This was a virtual appliance ran in VMware Fusion version 4.1.0. Some of the larger simulations were run on a dedicated server using Ubuntu 12.04.3 LTS with kernel 3.8.0-29-generic.

The NS-2 version 2.34 was used and obtained together with the rest of the framework from RACUN subversion server, revision 4429 (checked out 11.12.2013).

Chapter 10

DFLOOD

One of the main focuses of this thesis was to look at how to improve Dflood without changing the reduced flooding concept of the protocol.

10.1 Dflood

As mentioned in Section 7.6.2, the goal of Dflood is to be a flooding protocol with a technique to reduce duplicate transmissions. According to [24] this is done using some rules.

All packets have a unique sequence number that is used to identify duplicates. When a node has data to transmit (it gets data from a layer above the network layer) is sends the data down to the MAC layer immediately, setting the hop counter to 1. When a node is receiving a packet from another node, the forwarding is delayed with a time drawn uniformly from $[T_{min}, T_{max}]$. Packets with the same sequence number are treated as duplicates and discarded if the packet has already been forwarded. Duplicates received with a *higher* hop-count will be counted as n_d, but if a duplicate with a *lower* hop-count is received, the packet is queued to be forwarded is updated with the new hop-count value. Each time a duplicate with a higher hop-count is received, the protocol checks will draw a random number $r \in (0, 1]$. If the received number of duplicates $n_d > N_{Dupl} - r$ the forwarding is discarded (N_{Dupl} is a pre defined maximum of duplicates, and can be non-integer). When a node receives a packet destined for itself, it immediately broadcast a 1-hop "Received Notification", and nodes receiving this will discard the forwarding of this packet. Whenever a packet is forwarded from a node, the hop-counter increases by 1.

Referring to simulations done in [24], the authors achieved their goal by reducing the number of duplicates.

10.2 Retransmissions in Dflood

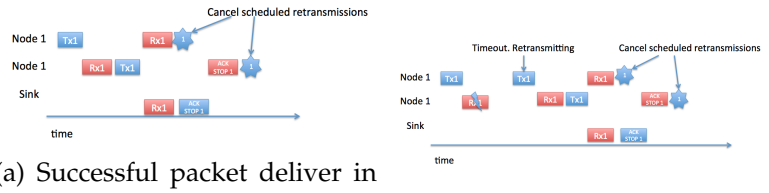
Simulations done in RACUN, indicated that GUWMANET performed better with repetitions of transmissions than without. The concept is

that for every data packet that is sent, the node reschedules a copy of the packet to retransmit at time $t = t + \tau$ where τ is the retransmission delay. The retransmission technique was adapted into Dflood with a twist. GUWMANET rescheduled its packet with a static time interval in the version that was available during the simulations (30, 60 and 120 seconds) which was adopted for Dflood.

One of the issues addressed in one of the scenarios in the RACUN was that two nodes accidentally transmitted packets at exactly the same time. The two nodes detected the same passing vessel at the exact same time and tried to report this. This resulted in a collision making none of the packets coming through. This issue is solvable by the MAC-layer sensing the medium and adding a back-off time before transmitting. But by making the MAC-layer handling the issue by first sensing the medium and then adding a back-off timer, would not necessarily resole this issue because of the long propagation delays in the underwater channel. Addressing this issue gives light to the issue if a node starts to transmit while the medium is busy, resulting in a collision. Both this issues are resolvable by retransmissions.

Retransmissions in a flooding inspired protocol designed to reduce the number of transmissions may sound like a oxymoron. But the idea was by doing this the PDR would increase, making the network more robust. The duplicate reduction scheme from [24] also applies to retransmissions:

The original techniques in Dflood were designed to mark a packet for transmitted for a certain time so if the same packet was received after it was transmitted, the node would not re-transmit the packet. With this in mind, conditions for implementing the retransmission scheme were almost ideal.



(a) Successful packet deliver in Dflood, no retransmissions required

(b) A collision occurs, retransmission is needed

Figure 10.1: Retransmission scheme in Dflood

Before the node sends the packet down from the network layer towards the MAC layer (using the sendDown() function in MIRACLE), the node makes a copy of the packet, sending one copy down to the MAC layer and reschedule one copy with the *delay* parameter. The nodes also keeps track of how many times the packets are transmitted so its not exceeds the *number of retransmissions* parameter, which defines the maximum number a packet should be retransmitted (0 means the original Dflood protocol). As the original Dflood protocol, if a packet that is marked as transmitted is received, it is canceled, regardless if it still has to be retransmitted.

The first idea was to re-use the retransmission scheme from GUWMANET as it was at that time, making the retransmission interval static

(e.g. the nodes retransmitted the packets ever 30 second for 5 times). But simulations quickly indicated that a delay parameter set equal to the the original forwarding delay (drawn uniformly from $[T_{min}, T_{max}]$) was performing better than the static delay. With this in mind a randomized delay parameter came in place.

Instead of making the input parameter the actual delay, the delay is now drawn randomly based on the delay parameter to the protocol

$$\tau = \frac{t_{\text{delay}}}{2} + \text{rand}(\frac{t_{\text{delay}}}{2}) \tag{10.1}$$

where t_{delay} is the input delay parameter and τ is the actual retransmission delay.

With the latter delay technique the issue with the two node transmitting at the exact same time was resolved.

Since DFLOOD is located at the network layer and retransmissions is applied here, this will not improve nor worsen due to fragmentation on lower layers (as discussed in Section 9.4)

Randomized retransmission interval has now been introduced in GUWMANET as well.

Other enhancements of DFLOOD

While this work was done, another enhancement was introduced in Dflood by one of the RACUN-partners. These improvements are outside the scope of this thesis and are not used or considered in the simulations or discussions.

Dflood with retransmissions in RACUN

After extensive testing by the RACUN partners, RACUN decided to use the implemented enhancement of Dflood done in this thesis in their Sea Trials in May 2014 (after delivery of this thesis).

Chapter 11

Results

In this chapter, different simulations are done to support theories and implementations done in this theses. The simulations are explained and results are presented and discussed.

11.1 Main Scenario

The simulations focus on one scenario that is used. The scenario is a general scenario consisting of three barriers of static nodes and one AUV traveling back and forward. The scenario is illustrated in Figure 11.1 where the barriers are bottom nodes and the ”single” node is the AUV with the arrow marking its route. This scenario is inspired by work in RACUN. The scenario serves several purposes; it needs multi-hop properties to reach the destination node. But instead of a simple multi-hop scenario with nodes in a row, the barrier nodes overhears packets not destined to themselves and have to make a decision of how to handle those. With these main criteria, this scenario is suited to simulate the improvements in Dflood compared with other protocols.

One of the purposes of the barriers is redundancy. In one of the simulations below, ad-hoc ness in the protocols are tested by removing several redundant nodes during the simulation run. Another purpose for the barriers is the application aspect, where a larger number of nodes cover larger sensing area.

11.2 Postman scenario

One possible way to run a non-time-critical or delay-tolerant network (DTN) over some distance is to use one or several AUVs as ”postman”. In this scenario the bottom nodes do their sensing as in the previous scenario, but instead of forwarding the packets to other bottom nodes, they send the packets to the AUV, when reachable. Then as the AUV passes the barriers it tries to deliver the packets to the sink. This kind of scenario addresses several problems and some opportunities. The latter will be that the network can be larger in geographic area with nodes spread outside

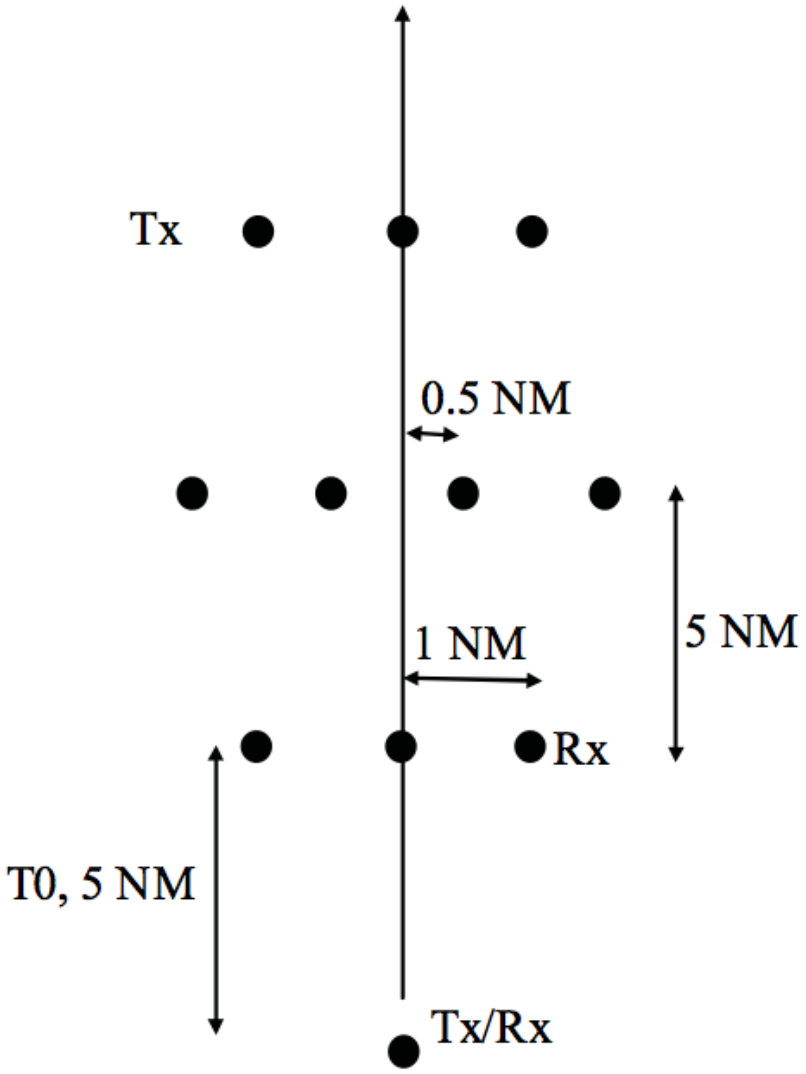


Figure 11.1: Topology used in simulations. From [15]

its transmission range (or with reduced transmission power making the node batteries last longer). This will consume more battery from the AUV, but as the AUV is mobile it can surface and be charged. A drawback with this kind of scenario is that the redundancy of the network is equal to the number of ”postmen” (AUVs), where AUVs are more expensive than bottom nodes. One of the problems this scenario addresses is how the bottom nodes know when the AUV is within transmission range, and likewise when does the AUV know when the receiving nodes are in transmission range. This can either be solved at the application layer where the AUV know where the bottom nodes are placed, or at the network layer. In the latter solution there are several ways to deal with this

problem; the AUV transmits constantly a ”HELO” type of packet, making the nodes aware of its present and making the nodes to reply and transmit its collected data. The other possible solution is the other way around; the bottom nodes constantly transmit ”HELO” like packets and waiting for the AUV to reply.

A third method is also possible and investigated in simulations done in this thesis. This method uses the retransmission scheme in Dflood to retransmit the packet until it is heard by the AUV. When the AUV receives the packet, Dflood will make sure the AUV tries to retransmit it further making the bottom node revokes a copy and cancel the rest of the retransmissions. The main problem with this technique is when the AUV retransmits a packet to a node that is not its destination, the node will retransmit it making the AUV receiving it and cancel the packet for further retransmissions assuming the bottom node can handle the retransmission further. To solve this problem, the AUV need to only cancel packets of ACK-STOP types.

The practical implementation of this scenario for simulations uses the existing scenario described in Section 11.1 and reduces the transmission power to a level where the barriers cannot reach each other. To achieve a good PDR in this scenario there have to be fewer packets transmitting as the AUV have to receive and store all the packets before it can delivers to the receivers.

The extensions done to Dflood in terms of the retransmission scheme are not designed to solve the challenge with these kinds of networks. But simulations are performed to investigate if the retransmissions scheme could yet solve a scenario like this.

11.3 Simulations of retransmissions in DFLOOD

One of the improvements in Dflood was to implement retransmissions of the data packets as described in Section 10.2. Simulations were performed to analyze the modifications and determine if this was an improvement or if the extra number of packets transmitted would only result in a larger number of collations and then again poorer network performance. To determine this, a set of static parameters were used only variating the focus parameters; numbers of retransmissions and retransmission delay. These parameters were varied in a nested for loop, varying all possible combinations of these.

There were done several simulations of the retransmission scheme. The first was an initial simulation with large variations of retransmissions and retransmission delay, with the goal to make an overview and locate a focus area to make more specific simulations, and discover the best-performing parameters to use in other simulations. This simulation varied over 0 - 10 retransmission and over a delay parameter of 0 - 200 seconds with 10 seconds intervals. The results are presented in Section 11.4.

Other parameters used are presented in Table 11.1. All simulations were run 10 times with different seed to the random generator in NS2 (the seeds

were 1 - 10) and an average over the ten simulation runs were made and represents the results presented.

Parameter	Value
Number of retransmissions	0 - 10
Delay parameter	0 - 200
Transmission power	170 dB (re μPa^2m^2)
CBR period	300 s
No. Packets	300
Dflood T_{min}	5
Dflood T_{max}	60
Dflood T_{Dupl}	20
Dflood N_{Dupl}	2.5

Table 11.1: Overview of parameters used in simulations

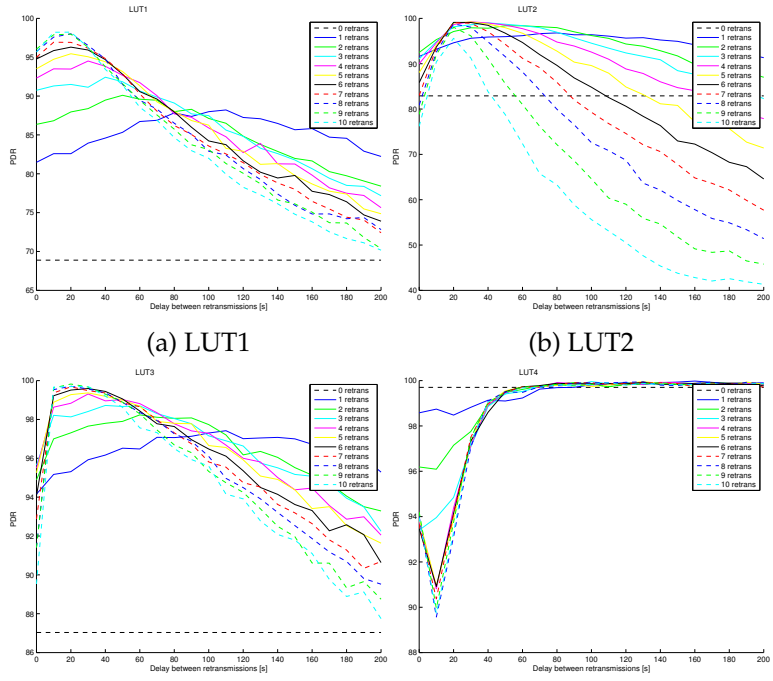
11.4 Results of retransmissions in DFLOOD

This section describes the results of simulations done to verify and find the best parameters of the retransmission extension to Dflood. The results are divided into three evaluation parameters; PDR, delay and energy usage, where the main focus will be PDR.

The results show that retransmissions in Dflood give a better network performance in terms of PDR. This is illustrated in Figure 11.2 where the four LUTs are shown for Country 3. The horizontal dashed line show performance without retransmissions (the original Dflood)

To better understand both the results presented in Figure 11.2 and the LUTs, Figure 11.3 shows how the links are connected in the different LUTs, called *connectivity maps*. The thicker and blacker line, means better packet error rate (PER) while a vanished white line corresponds to PER = 1. Here we can clearly see that there is a relation between the PER on the links and the overall PDR and how retransmissions increases overall PDR in LUTs with worse link to link PER. The special behavior in LUT4, compared to the other in Figure 11.2 is a direct result of the modulation performing better on link to link basis, so retransmissions in this case do not increase the PDR significant, but rather decreases PDR in the case of large number of retransmissions with a short delay parameter.

Some interesting sets of parameters are the ones using multiple retransmissions with zero as delay parameter. This will result in a ”burst” or retransmissions after the original forwarding is done. When the number of retransmissions is relatively low, zero as a delay parameter may actually perform better than spreading the retransmissions in short amounts of time. The burst are only delayed by the time the modem uses to transmit a packet and if the number of retransmission parameter is low, then all the packets have time to reach the receiving node before it has time to forward the packet (and cancel the retransmissions). But if the retransmissions



(a) LUT1 (b) LUT2

(c) LUT3 (d) LUT4

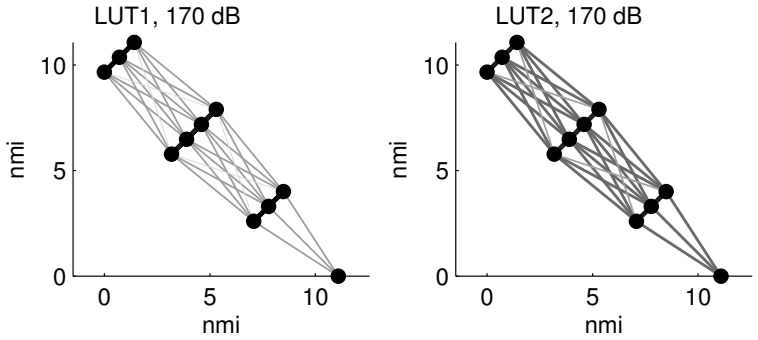
Figure 11.2: Showing performance of retransmissions in Dflood. The horizontal dashed line shows zero retransmissions (the original Dflood)

are many and spread in time, then collisions occur and the overall PDR decreases (this is what happens in Figure 11.2d). When the retransmissions are delayed even more, the packets have time to reach the node and be forwarded and then cancels scheduled retransmissions.

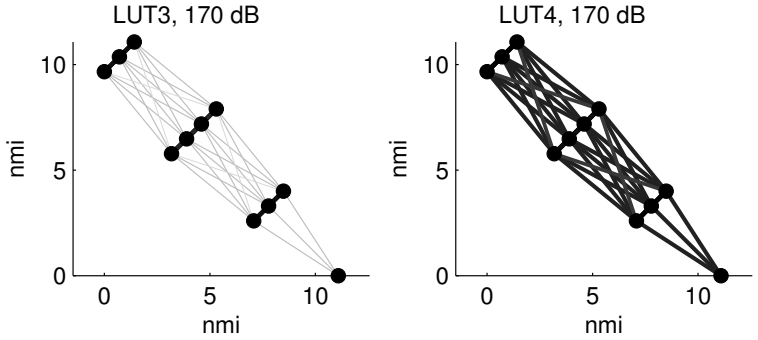
11.4.1 Packet delivery ratio

The initial simulation showed that there was an improvement in the packet delivery ratio (PDR) in almost all LUTs, when applying retransmissions to Dflood. But the variations of the improvements vary from almost nothing (see Table 11.2, Country 3, LUT 4) to very large improvements. As a sum of Table 11.2, also indicated from the graphs in Figure 11.2, an increasing number of retransmissions with a relatively short interval give the best performance (one exception is Country3 LUT4 which only gave a slight PDR increment with 1 retransmission with a delay parameter of 160 seconds). These results gave interest to try finding the inflection point due to number of retransmissions.

The first simulations clearly showed that retransmissions were beneficial to increase the PDR. To find the turning point where the number of retransmissions started to be a disadvantage to the network PDR, a simulation with up to 100 retransmissions where done. The retransmission delay was set to 10 seconds, as this value stood out as a good delay parameter in Figure 11.2. This simulation was done only one time per setting, so there may be some deviation when comparing the results with Table 11.2. The



(a) Connectivity map LUT1 (b) Connectivity map LUT2



(c) Connectivity map LUT3 (d) Connectivity map LUT4

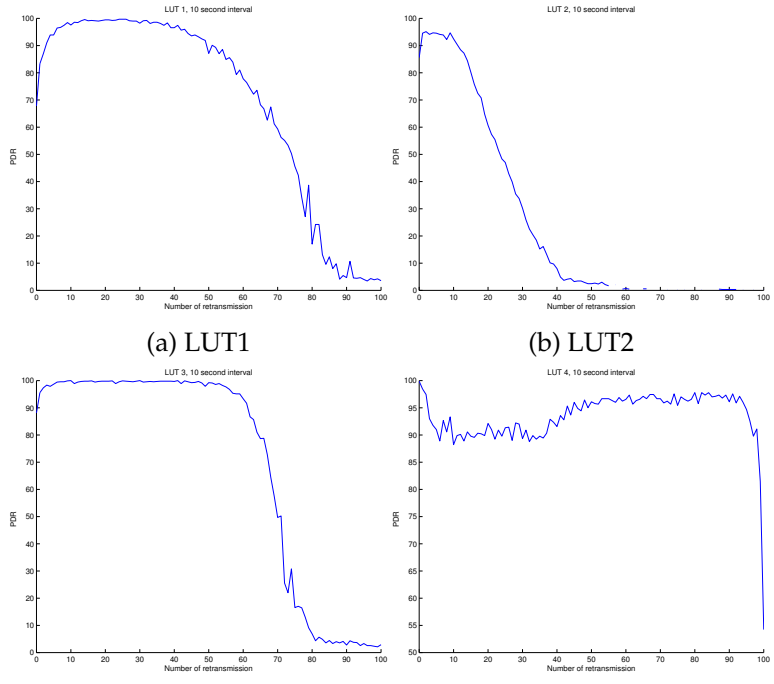
Figure 11.3: Connectivity maps showing how the links are connected. Thicker, blacker line means worse PER. From [16]

results, presented in Figure 11.4, shows that it takes a certain number of retransmissions before the PDR starts to decrease. It is worth pointing out that the PDR is climbing as the number of retransmissions increases, but relatively fast flattens, and for some of the LUTs it stays that way.

Another aspect to Figure 11.4 and Figure 11.2, is that this is a network with rather low traffic intensity. There are 300 seconds between each transmission (CBR period). Another simulation were done where the traffic intensity were increased. The time between packet creations was 30 seconds, one tenth of the original. It is worth mentioning that reducing the CBR period, will also reduce the simulation time (the stimulation ends $t_{stop} = No_{packets} * CBR_{period}$). In the first simulation lasted for $300s * 300s = 90000s$ while the latter simulation run lasted for only 9000 seconds (simulated time, not time to run simulation).

The results, illustrated in Figure 11.5, confirm that increased traffic intensity will reduce the PDR drastically, and decrease the maximum number of retransmission before the network is congested.

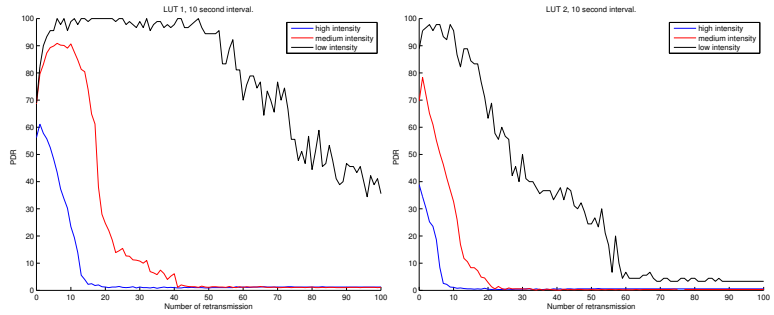
A middle intensity was also introduced with a CBR period of 100 seconds. This further supports the theory that with higher network intensity, the maximum number of retransmissions should be reduced.



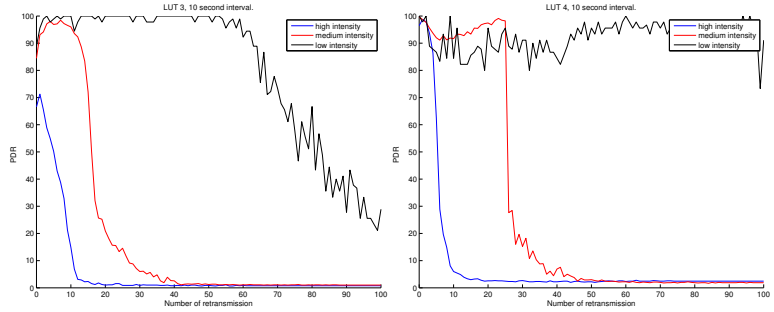
(a) LUT1 (b) LUT2

(c) LUT3 (d) LUT4

Figure 11.4: Simulation of 100 retransmissions



(a) High and low intensity LUT1(b) High and low intensity LUT2



(c) High and low intensity LUT3(d) High and low intensity LUT4

Figure 11.5: Comparing high and low network intensity in Dflood

11.4.2 Energy use

With increasing number of retransmissions, the energy use increases as well. This increased energy use is a price to pay for increased PDR. So one interesting aspect is to review the amount of energy needed to deliver one bit successfully. An overview is showed in Figure 11.6. Here there is worth pointing put that for the first five retransmissions the energy usage is rather constant varying over the retransmission delay parameter. But as the number of retransmissions exceeds five, the variation is larger and the trend is that for higher number of retransmissions, the energy usage per bit increases with the retransmission delay parameter. This is because there exist more packets in the network making more collisions appear, but with a high retransmission number, the nodes trying and trying to get its packet trough, resulting in the large energy usage.

Comparing Figure 11.6 and Figure 11.2b shows that there is a peak in PDR in Figure 11.2b at the same delay parameter, about 20 second, where there is a dip in energy usage in Figure 11.6. Also comparing the fewer retransmissions, there is a direct relation between the PDR and energy usage (per bit). This verifies the cancel technique in Dflood that cancels the transmission(s) of copy(s) waiting; if a packet is received by the neighbor and overheard by a node, then the remaining retransmissions will be canceled, making the energy usage less per bit. But as both Figure 11.6 and Figure 11.2b shows, many retransmissions with high delay parameter will cause a lot of packets living in the network and again causing collisions, making the nodes try to retransmit again and again, using a lot more energy per bit.

11.4.3 End-to-end delay

With the retransmission scheme, there was also introduced a delay parameter, to determine when the copy is to be retransmitted. This obviously introduces a increased end to end delay (the time it takes for a packet to travel from source to sink). With increased number of retransmissions (to a certain point), the number of successfully received packets increases and hereby increases the average end-to-end delay.

It is only the successfully received packets that can be measured end to end delay. So if one packet is received quickly after transmissions, but 100 is dropped, then the average delay is still pretty good CONFRIM THIS!!!!!

Comparing Figure 11.2b with Figure 11.7, shows that for

As the PDR goes down (packets are dropped), the average end-to-end delay flattens (especially in the larger numbers of retransmission). This is illustrated in Figure 11.7

11.4.4 Interframe period

Some of the LUTs (LUT 1 and LUT 2) fragment the Dflood packets into two frames. This means that there could be an interval between the transmissions of the frames. This is called interframe period. This section

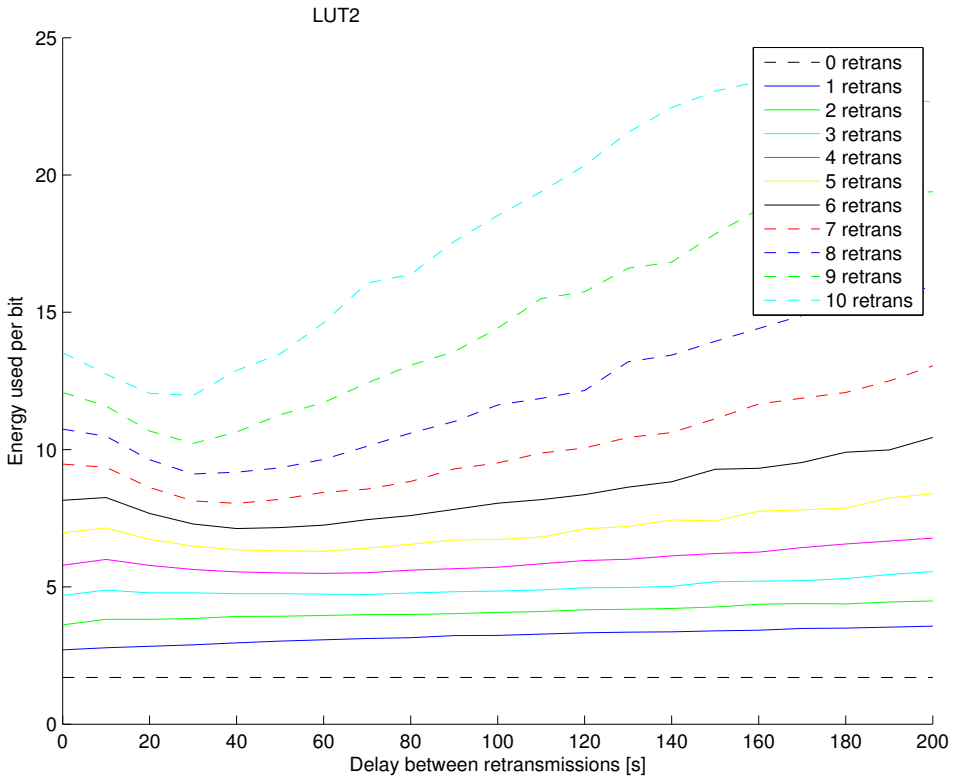


Figure 11.6: Energy usage

will investigate simulations done changing the interframe period to see how this will affect the PDR.

The interframe period that has been investigated spanned from 1 to 5 seconds with steps of 1 second. The interframe interval only applies to LUT1 and LUT2 as LUT3 and LUT4 do not require fragmentation. To save some time the simulations were done only once with one random seed. Therefore, the results may vary when comparing the same parameters to the earlier simulations.

The results indicate that changing the interframe period does not have a big impact on the network performance. If the retransmissions are done with a relatively long delay, then the shorter interframe interval are beneficial.

11.4.5 Numerical results

The best performing parameter set in Dflood retransmission simulations are presented in Table 11.2. Due to an error that was discovered in one of the simulation scripts, not all of the LUTs were simulated the first time. There was not enough time to simulate the remaining LUTs using the same parameters. LUTs marked * were simulated varying retransmissions from 0-9 and delay parameter 0-130 seconds. The LUT marked ** were no time to simulate. This table shows that multiple retransmissions with a shorter delay interval performed better. 10 retransmissions with a delay parameter

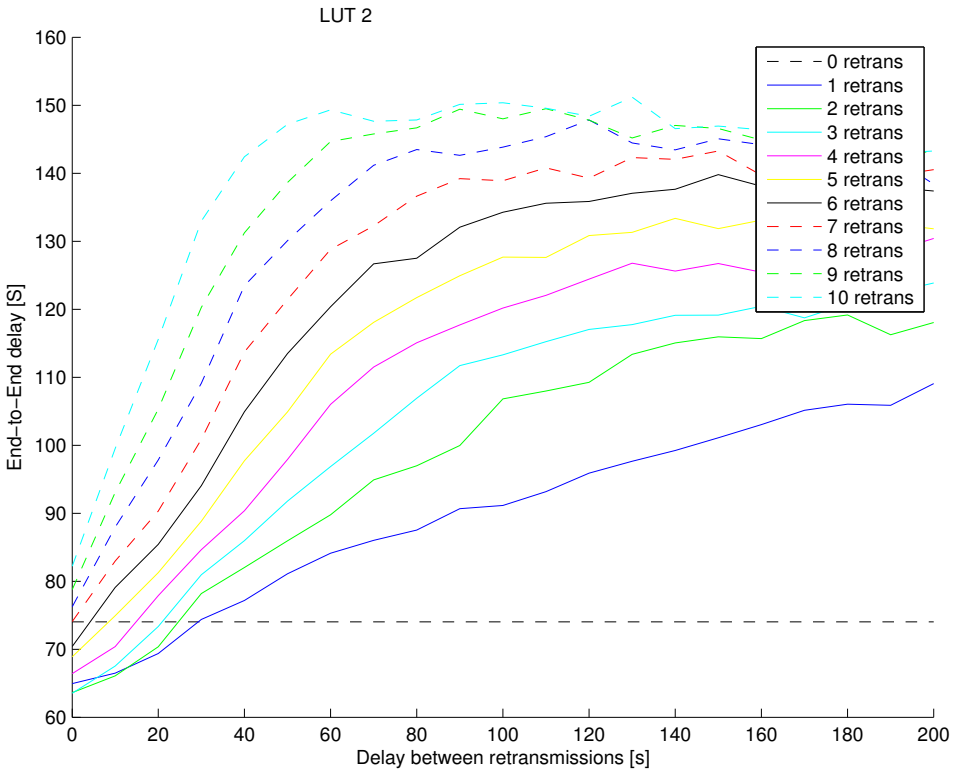


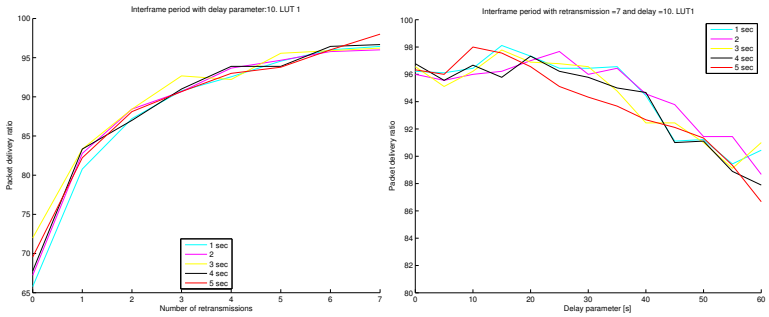
Figure 11.7: End to end delay (the horizontal dashed line is original Dflood)

of 10 seconds, was pointed out to perform best in the most scenarios. But the increased PDR has a great cost in terms of a big increase of energy used to successfully transmit each bit from end to end. It is worth pointing out that the optimal parameter will be different for another traffic load, see e.g. Figure 11.5

11.5 Dflood as postman forwarding

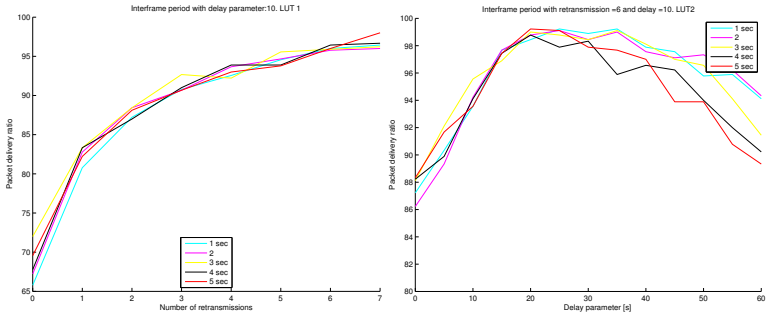
With a slightly modified version of Dflood with retransmissions, the postman scenario described in Section 12.2.2. The modification is that the node is aware if it should act as postman or not. In the scenario simulated, this is valid for the AUV(s). When a node acts as postman, it only cancel packets from ACK-STOP and not if it overhears a packet that is stored in its buffer. The traffic goes from node 1 to node 8 according to the network in Figure 11.12 using a constant bit stream (CBR) model. Traffic is only carried by the AUV. Since the only network carrier is the AUV, the number of packets is few and generated with a relatively long interval (300 seconds).

The results of this simulation are very varying. The number of retransmissions and the delay are more connected to the AUV then in the other scenarios. The AUV speed is critical. One set of parameter can give a high PDR with one speed, but can give another value when changing the AUV speed. The distance between the start point and the returning point



(a) LUT1 comparing retransmissions and interframe period

(b) LUT1 comparing delay and interframe period



(c) LUT2 comparing retransmissions and interframe period

(d) LUT2 comparing delay and interframe period

Figure 11.8: Interframe period results

of the AUV is 20 nautical miles or 37 km. Table 11.3 gives an overview over how long time the AUV uses from the start point to the returning point (covering all barriers).

Distance	**Speed [knots]**	**Speed [m/s]**	**Time [min]**
20 nm	4	2.05778	308
20 nm	10	5.144	120
20 nm	20	10.288	60

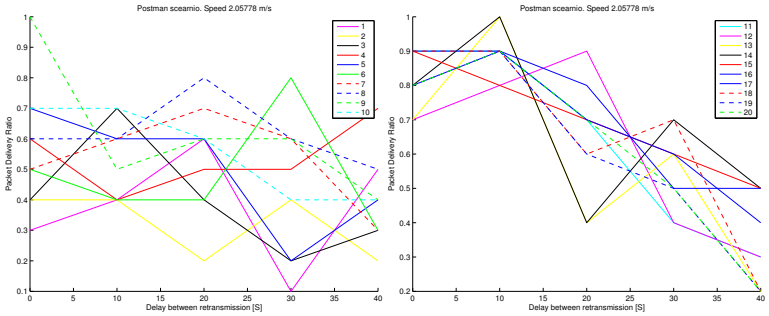
Table 11.3: An overview over time versus speed used in the simulation of Postman scenario

The results show a varying PDR (shown in Figure 11.9). This also illustrates that AUV speed influence the results. But in both the speeds simulated, there were parameters that gave 100% PDR, but it is worth pointing out, that this simulation was done with only one run per parameter setting.

The focus speed in further investigations will be 4 knots or 2,0778 m/s since this is the speed used in previous simulations. The above results were obtained running each simulation one time only. Table 11.4 shows the best parameter settings simulated 30 times with different seed to the random generator. The results investigated were the parameter settings that obtained a PDR of 100% (13 and 14 retransmissions with a delay

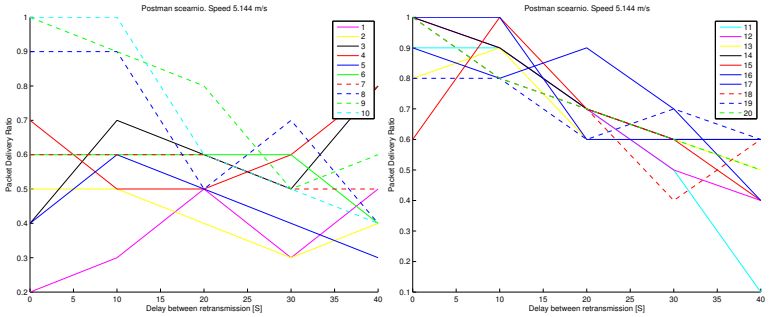
parameter of 10 seconds). Also 10 retransmissions were included in the verification test to see how this setting performed (since this performed the best in terms of PDR in previous simulations). The best performing parameter setting seems to be 14 retransmissions with a delay parameter of 10 seconds. But still this performs below 90%.

With the numbers from Table 11.3, an optimal delay parameter can be calculated. As the AUV has to return to the transmitting node to pick up a new packet, the round-trip-time (RTT) is needed. With a speed of 4 knt (2.05778 m/s) this is 36000 seconds or 600 minutes, meaning that it have to bee about 36000 seconds between the retransmissions. If the delay parameter is set to this value, there will be two issues; One, the actual retransmission interval will be from 18000 seconds to 36000 seconds and two, there will be only one transmissions per time the AUV is in range (and the transmission will be longer and longer out of reach for the AUV). To cope with this issue, the better solution is to adjust the CBR period (how often packets are generated). With the CBR period set to 3600 with a retransmission parameter of 10 and delay parameter to 10 seconds, this issue is solved. Averaging over 30 runs with variation of the random seed to NS-2, the overall PDR became about 84%.



(a) Speed 4 knt 1-10 retransmissions

(b) Speed 4 knt. 11-20 retransmissions



(c) Speed 10 knt. 1-11 retransmissions

(d) Speed 10 knt. 11-20 retransmissions

Figure 11.9: Comparing speed in postman scenario

Retransmissions	Delay parameter	Average PDR	Energy used	End to end delay
10	10	0,78	12,21	147,21
13	10	0,79	14,42	147,71
14	10	0,88	13,95	153,51

Table 11.4: Average of best performing parameters

11.5.1 Energy usage in postman scenario

The results from Section 11.5 showed that the retransmission interval that could reach a good PDR was at 10 seconds, with relatively high amount of retransmissions. Compared to the results presented in Table 11.2 the numbers in Figure 11.10 is very high. This is because of all the retransmissions actually taking place, instead than be a maximum value. Still, the best performing parameter set (14 retransmission 10 second delay), marked as red, is not the highest value of energy usage per bit. This indicates that the better performing parameter is the same that uses least energy per bit. This makes sense, since the retransmission mechanism is canceled when receiving a copy of the same packet, in other words, a packet is delivered to its neighbor.

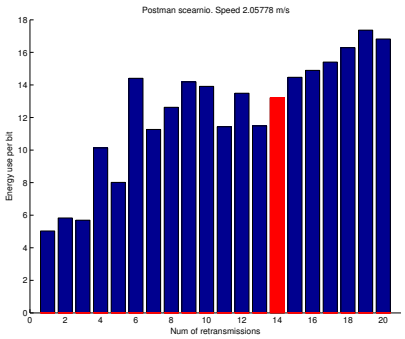
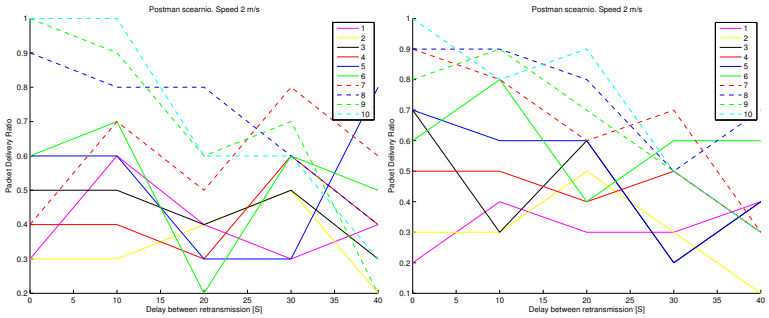


Figure 11.10: Energy usage in the postman scenario.

The ”Ideal run” (described in Section 11.5) gave an energy usage of 10.88 J/bit, which is lower than the red value in Figure 11.10.

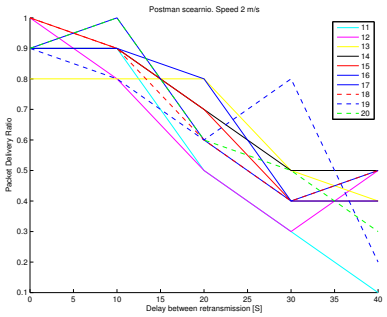
11.5.2 Multiple AUVs

In an attempt to improve the robustness and increase the PDR with a lower energy usage, an additional AUV was introduced. The second AUV travels in the opposite direction of the first, using the same path. Introducing another AUV in a real life network is expensive and leads to more administration. By comparing the graphs in Figure 11.11 shows difference in PDR when adding a second AUV to the network. The results are almost identical, proving that in this scenario, adding a second AUV is not so effective. This is because all the traffic goes from only one node, that adding a second AUV going the opposite direction, have almost no influence on PDR.

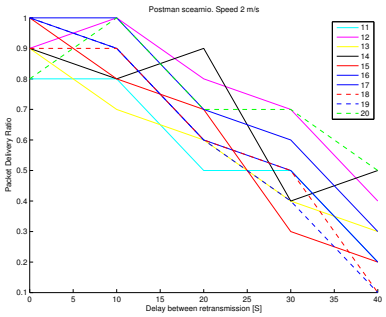


(a) One AUVs Retransmission 1-10

(b) Two AUVs Retransmission 1-10



(c) One AUVs Retransmission 11-20



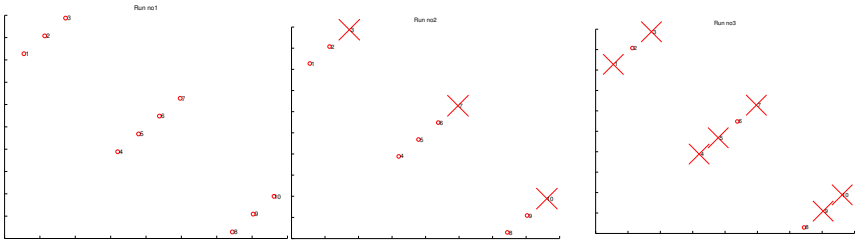
(d) Two AUVs Retransmission 11-20

Figure 11.11: Comparing different parameters in postman scenario

The ”Ideal run” (described in Section 11.5) was also simulated with two AUVs, using the same path as the other simulations done with two AUVs. Since the CBR period was tailored for one AUV, the results do not differ much using one or two AUVs. Using dual AUVs traveling back and forward, the average PDR increased some from 84% to 87%. This is probably because on a certain time in the simulation, the second AUV would act as a bridge between the barriers.

11.6 Ac-hoc ness

In any sensor network, it is not unlikely that one ore more nodes fails eventually, as mentioned in Section 9.6. To simulate the Ad-hoc robustness, some of the nodes are removed during the simulation run. The nodes are removed some time into the simulation, giving the network time to establish itself and potentially making routes (if a routing protocol is applied). Network performance is expected to reduce after the nodes are removed from the network. The evaluation criteria to measure the ad-hoc robustness used here, are PDR over time. This means that the total PDR for the network are sampled with an interval of 500 seconds. As a result of this, the PDR may vary a lot, in terms that in one sample, a packet has not yet arrived, making the PDR poorer, while in next sample the delayed packet has arrived, increasing the PDR again. This phenomena will only affect



(a) Run 0 (b) Run 1 (c) Run 2

Figure 11.12: The different runs with failing nodes

the PDR in short terms (and have largest impact in the beginning of the simulations when fewer packets have been transmitted) and are ignored in the analysis.

The measurement is done during the simulation and results are plotted in Figure 11.13. The simulated scenario is described in Section 11.1, and is done in four runs (four different simulations):

- The first run contains all nodes in the scenario. This is used as reference/control run and is referred to as Run0. This is illustrated in Figure 11.12a.
- The second run reduces all the barriers with one node (removing one node per barrier), and is referred to as Run1. This is illustrated in Figure 11.12b.
- The third run leaves only one node in each barrier, and is referred to as Run2. This is illustrated in Figure 11.12c.
- The fourth run removes the AUV in the scenario, and is referred to as Run3. This gives the same statics nodes as in Run 0.

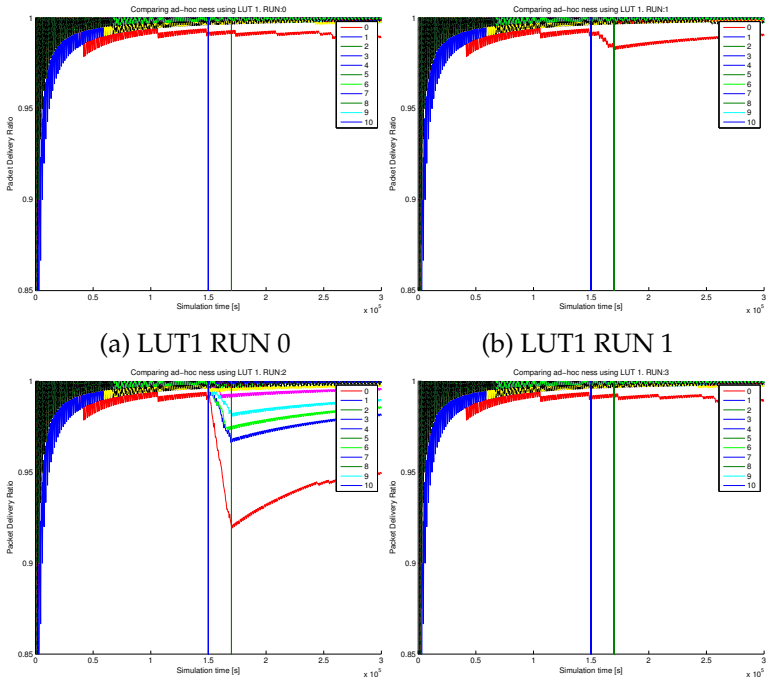
All the runs are illustrated in Figure 11.12, showing the bottom nodes. The nodes represented by 'x' are the ones failing during the simulation. After 150 000 seconds (about halfway) into the simulation, the intended nodes are removed, and 20 000 seconds after the removed nodes are reinstated at their original positions. This will give the network some time to establish new paths. The reinstatement is done to see how the network reacts to increasing number of nodes. In the figures, the vertical line to the left indicates when the nodes are dropping, and the vertical line to the right, when the nodes are reinstated into the network.

The traffic in these simulations goes from node 2 to node 8 with a CBR model, i.e. either the source nor the sink is removed from the network.

DFLOOD ad-hoc

Since DFLOOD is a flooding protocol it is expected that this protocol should perform well when the network is loosing nodes. But of course, if there are not enough nodes to establish communications, then the

communication is broken and packets are lost. This event does not seem to happen.



(a) LUT1 RUN 0 (b) LUT1 RUN 1

(c) LUT1 RUN 2 (d) LUT1 RUN3

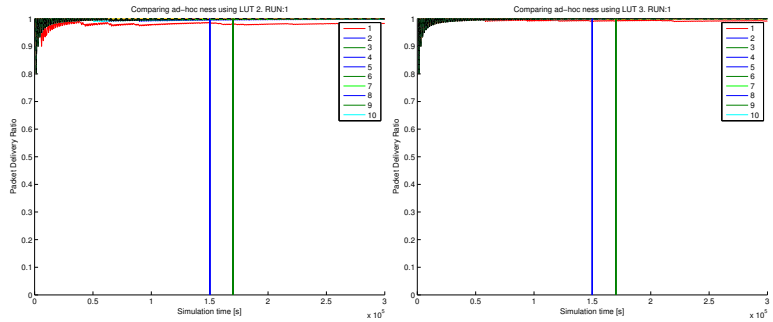
Figure 11.13: Comparing PDR in the different RUNs using Dflood

RUN0 (the control run) and RUN3 (the run where the AUV is removed) contains the same bottom nodes (as Figure 11.13a and Figure 11.13d illustrates). The Dflood with retransmissions turns out to be a more robust protocol when it comes to a network with varying number of nodes than the original protocol. As Figure 11.13b indicates, 0 retransmissions are the only parameter setting is dropping packets. Figure 11.13c supports this, but here there are other settings that also drop packets. If the number of retransmissions is 5 or more, then the number of dropped packets is negligible.

The small variations in all the graphs presented in Figure 11.13 are caused by delay and maybe a to small sampling rate (in sample period *n* there are transmitted *x* packets but only received *x-1*. In period *n+1* then the last packet has arrived causing variations the PDR over time).

Figure 11.14 illustrates that Dflood is a rather robust protocol when nodes are failing and there is still a path. This is an expected behavior as it is a flooding type of network protocol. The number of retransmissions does not have a large impact, in terms that several retransmissions do not have a different impact of the PDR when nodes are failing, than few retransmissions.

When it comes to retransmissions in Dflood, this has almost zero impact on the ability to rebuild paths. It has impact on the overall PDR, but as Figure 11.14b shows that the different lines are either overlapping (due to



(a) Dflood PDR when nodes are failing. RUN 1 LUT 2

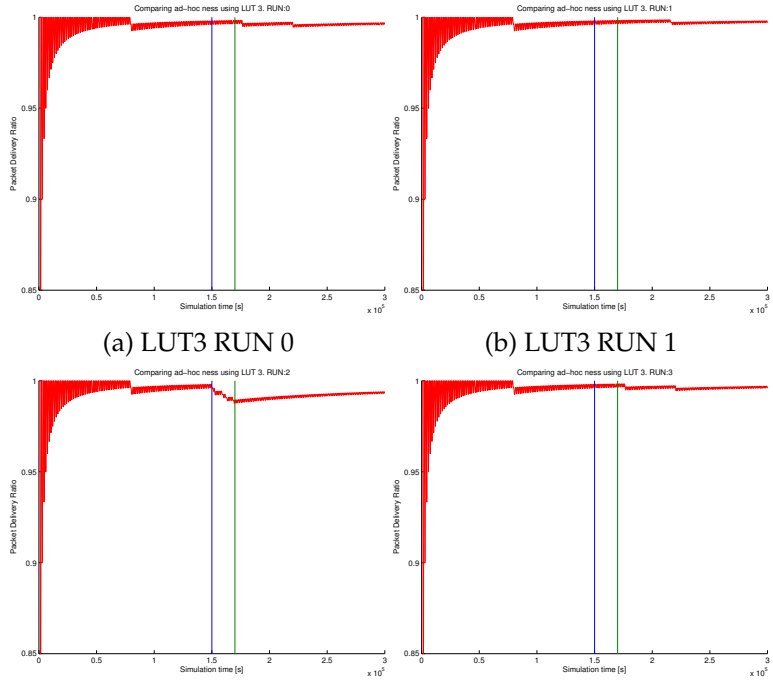
(b) Dflood PDR when nodes are failing. RUN 1 LUT 3

Figure 11.14: Ad-hoc ness i Dflood. Run 1

large scale) or parallel.

GUWMANET ad-hoc

GUWMANET is as described in Section 7.5.1. In the simulations below, GUWMANET have gone trough the simulations described in Section 11.6 to analyze its ability to maintain a high PDR when nodes are failing in a network.



(a) LUT3 RUN 0

(b) LUT3 RUN 1

(c) LUT3 RUN 2

(d) LUT3 RUN3

Figure 11.15: Comparing PDR in the different RUNs using GUWMANET

As Figure 11.15 indicates, the PDR over time is very stable showing a low packet drop count. I RUN2 there is some packet loss during the period

where the nodes are gone, but the PDR is established after the nodes are reinstated, showing a stable climb. This indicates that GUWMANET is a rather robust protocol in this test, but not so robust as Dflood with multiple retransmissions.

In this simulation, GUWMANET used a static retransmission scheme with 3 retransmissions with a 60 second delay.

In all simulations done to investigate the ad-hoc robustness, the most interesting run has been RUN2 where there are only three bottom nodes plus one AUV are left in the network. Figure 11.16 shows a comparison of GUWMANET and different Dflood retransmission parameter. The figure clearly shows that using this LUT, Dflood with multiple retransmission is more robust than GUWMANET due to PDR over time.

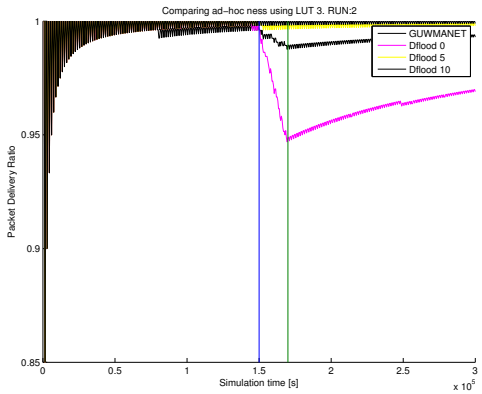


Figure 11.16: Comparing Dflood and GUWMANET ad-hoc robustness in RUN2

11.7 Simulation of modulations

As described in Section 9.3 the RACUN simulation framework are provided with several LUTs based on sea trials done in different countries. Some of the locations have used the same modulations scheme and parameter setup, where the hardware allowed is, making the environment the varying variable. To illustrate just how large role the environment plays in underwater communications, there were performed some simulations. The simulations are based on the same scenario as before and the varying variables are the modulation (in terms of the LUTs) and the total amount of packets sent during the simulation time.

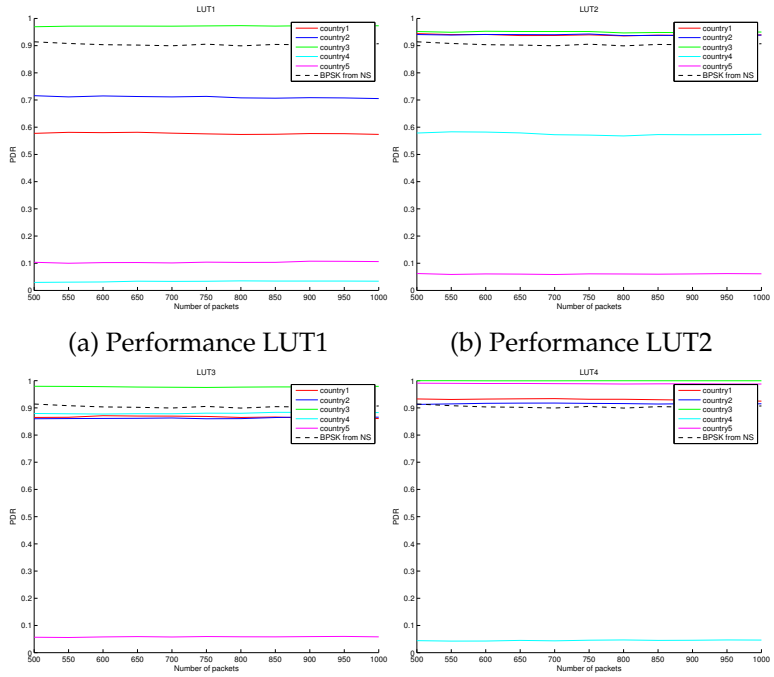
11.7.1 Results of modulation simulation

The results of modulation simulations are showing a rather large variation in the different locations performance. This is a direct result of the variation in the measured underwater acoustical communications channel.

Figure 11.17 shows a rather large variation of the different locations using the same modulations scheme. But is worth mentioning that the

measurements are done *in situ* and real-time so the results may be different if there are to be done a new measurement.

The results of these simulations are intended as an indication of the large impact the underwater environment has to the underwater network, and the performance may not be comparable to other results presented in this thesis (due to difference in parameter setting).



(a) Performance LUT1 (b) Performance LUT2

(c) Performance LUT3 (d) Performance LUT4

Figure 11.17: Comparing different LUTs.

For comparison, a BPSK model used in NS-2 is also included in the simulations.

LUT	No Retrans.	Retrans. Delay	PDR	Original PDR	Delay	Energy pr bit
COUNTRY 1						
LUT 1	10	10	87.178	11.923	124.288	5.860
LUT 2**	N/A	-	-	-	-	-
LUT 3	9	10	96.754	38.523	86.786	6.964
LUT 4	10	10	97.768	49.733	79.494	5.746
COUNTRY 2						
LUT 1	10	10	40.243	8.843	95.045	8.768
LUT 2	4	20	84.734	43.499	115.090	7.505
LUT 3	10	10	45.078	14.848	79.906	11.475
LUT 4	10	10	96.835	31.166	87.918	6.771
COUNTRY 3						
LUT 1	10	20	98.223	68.888	87.613	3.064
LUT 2	4	30	99.243	74.038	84.636	5.633
LUT 3	9	20	99.823	87.034	65.801	4.210
LUT 4	1	160	99.798	99.702	34.078	1.049
COUNTRY 4						
LUT 1*	9	20	6.511	2.176	56.751	23.860
LUT 2	5	10	48.322	20.36	91.518	11.214
LUT 3	10	10	79.857	38.856	88.822	6.489
LUT 4*	9	20	10.444	3.800	63.378	17.788
COUNTRY 5						
LUT 1*	9	20	11.857	6.034	47.816	12.292
LUT 2*	9	30	9.934	5.210	61.811	39.185
LUT 3*	9	10	10.224	5.545	29.324	22.532
LUT 4	10	10	97.277	77.578	65.644	4.889

Table 11.2: Numeric results of the combinations that performed the best PDR.

Chapter 12

Conclusions

In this chapter the main outcome of the simulations are presented and conclusions are drawn.

12.1 Simulations

Simulations are an approximation to the real world. Even though the LUTs are made from real-life measurements of the physical channel in different places, the results in Section 11.7 represents that environment factors has a great influence on network performance. This makes the simulations more a tool to compare and develop networks protocols rather than predict network performance in given circumstances. But for this purposes, simulations are a brilliant way cut cost and time developing and debugging protocols.

The simulation framework was selected and developed by the RACUN project, which has on the roadmap to perform a sea-trial using the protocol stack and some of the software used in the simulation. Comparing results from this sea-trial with the simulation framework would give a good indication of how close to real-life the simulations were. Simulating improvement to a protocol without changing the framework, will give a good glance of how well the improvements work.

12.2 Improvements to Dflood

The implementation of retransmissions gave a overall higher PDR to the network. Shorter interval between the retransmissions presented itself to be a better option than longer intervals in the simulated cases with relatively low traffic intensity. Initial simulations presented *10-10* (10 retransmissions with a 10 second delay parameter) as the best option in most cases/LUTs. With these parameters, the PDR improved in some cases a great deal. Table 11.2 sums up a lot of the results, showing the parameter set that performed the best (in PDR) during that simulation. Here the number of 10 retransmissions stood out as the parameter giving the best result (PDR). But this was also the maximum number of retransmissions

done in that simulations, so this indicated "more retransmissions is better". A new simulation was performed varying of 0-100 retransmissions showing that a range of retransmissions gave a good PDR. In Figure 11.4c as an examples, shows that in the range between 8 and 50 retransmissions, the PDR performs good (almost 100%). The reason there is such interval, is because the number of retransmissions parameter defines the *maximum* number of retransmissions. So this indicates that the actual number of retransmissions is about 8.

With traffic intensity increased from 300 seconds between packet creation (CBR period) to 100 seconds or 30 seconds (ten times as often), the overall PDR for the network sunk drastically with number of retransmissions. This is because the network is overcrowded with traffic.

As indicated in Chapter 11, increased number of retransmissions means increased end-to-end delay as well as increased energy usage per bit. This means that there have to be a trade off between power, delay and PDR.

It is worth mentioning that in the fragmenting LUTs, the two fragments have their own sequence number, so if a node receives fragment 1 of one Dflood packet and fragment 2 from a copy of the same Dflood packet, it cannot recreate the Dflood packet and are counted as lost.

The improvements done to Dflood in this thesis have been adopted into the RACUN framework and will be used during Sea Trial in May 2014.

12.2.1 Ad-hoc ness

Dflood with different number of retransmissions were tested to see how this affected the presence of failing (and reinstating) nodes. Figure 11.13c shows that the increasing number of retransmissions gives a more robust version of Dflood, dropping fewer packets than the original protocol. The simulations also compare the robustness with GUWMANET, a routing protocol. Figure 11.16 shows that GUWMANET is a rather robust protocol (in this scenario), but with 5 or more retransmissions, Dflood proves itself as a robust protocol before, during and after the nodes were failing.

12.2.2 Postman scenario

Dflood with retransmissions are capable to perform this role, if the acceptable PDR are in range 80% - 90%. This is not a very high PDR and calls for improvement. One thing to look into is a REQUEST_ACK structure where the AUV sends a DATA_REQUEST packet and a ACK_DATA packet when receiving the packet. This calls for additional bits in the header in Dflood, or it can be realized on a higher level, e.g. using Transmission Control Protocol (TCP). Dflood can perform the role in simulated scenario, but with a rather high price to pay in terms of energy use and relatively low PDR. It is possible that realizing the scenario at a higher level will be preferred.

12.3 Further work

All of the simulations were done in a friendly environment where the node could rely on each other that none had hostile motives. All the simulations are done with naive nodes thinking the best of the nodes around it, without asking forwarding data to all nodes in the transmission radius. The RACUN projects main goal is to establish a robust underwater network, but in the next phase may be the focus will be turning the robust network into a secure robust network.

One idea is to design a MAC layer which takes in considerations the Dflood retransmission scheme and knows about the Dflood sequence number, so in the case of fragmentations, the receiver could use one the first fragment from one copy and a second from another copy. Another way to improve Dflood, is the other way around; Dflood is aware of the limitations in the modem due to fragmentations or not, and can perform the fragmentations on the network layer. With the latter example, Dflood is less depended on the other layers and can save fragments for an amount of time, waiting for the other half.

Another idea to improve Dflood in high intensity network is to make the retransmission parameter adaptive. If the node experiences great traffic intensity, it will lower the maximum retransmissions to a more suited level.

Bibliography

[1] Peng Xie, Zhong Zhou, Zheng Peng, Hai Yan, Tiansi Hu, Jun-Hong Cui, Zhijie Shi, Yunsi Fei, and Shengli Zhou. Aqua-sim: an ns-2 based simulator for underwater sensor networks. In *OCEANS 2009, MTS/IEEE Biloxi-Marine Technology for Our Future: Global and Local Challenges*, pages 1–7. IEEE, 2009.

[2] Michael A Ainslie. *Principles of sonar performance modelling*. Springer, 2010.

[3] Milica Stojanovic and James Preisig. Underwater acoustic communication channels: Propagation models and statistical characterization. *Communications Magazine, IEEE*, 47(1):84–89, 2009.

[4] Cristiano Tapparello, Paolo Casari, Giovanni Toso, Ivano Calabrese, Roald Otnes, Paul van Walree, Michael Goetz, Ivor Nissen, and Michele Zorzi. Performance evaluation of forwarding protocols for the racun network. In *Proceedings of the Eighth ACM International Conference on Underwater Networks and Systems*, page 36. ACM, 2013.

[5] Josep Miquel Jornet, Milica Stojanovic, and Michele Zorzi. Focused beam routing protocol for underwater acoustic networks. In *Proceedings of the third ACM international workshop on Underwater Networks*, pages 75–82. ACM, 2008.

[6] Peng Xie, Jun-Hong Cui, and Li Lao. Vbf: vector-based forwarding protocol for underwater sensor networks. In *NETWORKING 2006. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems*, pages 1216–1221. Springer, 2006.

[7] Nicolas Nicolaou, Andrew See, Peng Xie, Jun-Hong Cui, and Dario Maggiorini. Improving the robustness of location-based routing for underwater sensor networks. In *OCEANS 2007-Europe*, pages 1–6. IEEE, 2007.

[8] Rony Hasinur Rahman, Craig Benson, Frank Jiang, and Michael Frater. Loarp: A low overhead routing protocol for underwater acoustic sensor networks. *Journal of Networks*, 8(2), 2013.

[9] Daeyoup Hwang and Dongkyun Kim. Dfr: Directional flooding-based routing protocol for underwater sensor networks. In *OCEANS 2008*, pages 1–7. IEEE, 2008.

[10] Michael Goetz and Ivor Nissen. Guwmanet—multicast routing in underwater acoustic networks. In *Communications and Information Systems Conference (MCC), 2012 Military*, pages 1–8. IEEE, 2012.

[11] Wikipedia. Phase-shift keying — wikipedia, the free encyclopedia, 2014. [Online; accessed 14-January-2014].

[12] Torleiv Masseng. *Communication and Information theory*. 2013.

[13] Joao Gomes and Milica Stojanovic. Performance analysis of filtered multitone modulation systems for underwater communication. In *OCEANS 2009, MTS/IEEE Biloxi-Marine Technology for Our Future: Global and Local Challenges*, pages 1–9. IEEE, 2009.

[14] William Stallings. *Wireless communications & networks*. Pearson Education India, 2009.

[15] Henry Dol, Michael Goetz, Thor Husøy, Lianke te Raa, Jan Nilsson, Ivor Nissen, and Roald Otnes. Requirements and objectives for underwater networks, definition of performance metrics racun deliverable d3.2. 2011.

[16] Ivano Calabrese, Paolo Casari, Michael Goetz, Ivor Isse, Arwid Komulainen, Roald Otnes, Cristiano Tapparello, Giovanni Toso, and Paul van Walree. Performance evaluation (simulated) of network protocols. deliverable 4.3.2, 2013.

[17] Ian F Akyildiz, Dario Pompili, and Tommaso Melodia. Challenges for efficient communication in underwater acoustic sensor networks. *ACM Sigbed Review*, 1(2):3–8, 2004.

[18] Lloyd Butler VK5BR. Underwater radio communication. *Originally published in Amateur Radio*, 1987.

[19] Wikipedia. Very low frequency — wikipedia, the free encyclopedia, 2014. [Online; accessed 21-April-2014].

[20] Xianhui Che, Ian Wells, Gordon Dickers, Paul Kear, and Xiaochun Gong. Re-evaluation of rf electromagnetic communication in underwater sensor networks. *Communications Magazine, IEEE*, 48(12):143–151, 2010.

[21] Frank Hanson and Stojan Radic. High bandwidth underwater optical communication. *Applied Optics*, 47(2):277–283, 2008.

[22] Jack L Burbank, William Kasch, and Jon Ward. *An Introduction to Network Modeling and Simulation for the Practicing Engineer*, volume 5. John Wiley & Sons, 2011.

[23] Steve McCanne et al. Ns2 documentation. [Online; accessed 3-November-2013].

[24] Roald Otnes and Svein Haavik. Duplicate reduction with adaptive backoff for a flooding-based underwater network protocol. In *OCEANS-Bergen, 2013 MTS/IEEE*, pages 1–6. IEEE, 2013.

[25] Telecommunications Group Department of Information Engineering. Ns-miracle. [Online; accessed september 2013].

[26] Riccardo Masiero, Saiful Azad, Federico Favaro, Matteo Petrani, Giovanni Toso, Federico Guerra, Paolo Casari, and Michele Zorzi. Desert underwater: an ns-miracle-based framework to design, simulate, emulate and realize test-beds for underwater network protocols. In *OCEANS, 2012-Yeosu*, pages 1–10. IEEE, 2012.

[27] Federico Guerra, Paolo Casari, and Michele Zorzi. World ocean simulation system (woss): a simulation tool for underwater networks with realistic propagation modeling. In *Proceedings of the Fourth ACM International Workshop on UnderWater Networks*, page 4. ACM, 2009.

[28] Wikipedia. Decibel — wikipedia, the free encyclopedia, 2014. [Online; accessed 24-February-2014].

[29] Milica Stojanovic. On the relationship between capacity and distance in an underwater acoustic communication channel. *ACM SIGMOBILE Mobile Computing and Communications Review*, 11(4):34–43, 2007.

[30] Michael A Ainslie and James G McColm. A simplified formula for viscous and chemical absorption in sea water. *The Journal of the Acoustical Society of America*, 103(3):1671–1672, 1998.

[31] RE Francois and GR Garrison. Sound absorption based on ocean measurements. part ii: Boric acid contribution and equation for total absorption. *The Journal of the Acoustical Society of America*, 72(6):1879–1890, 1982.

[32] encyclopedia.com. encyclopedia.com, 2014. [Online; accessed 3-February-2014].

[33] Robert J Urick. Principles of underwater sound. 1983. *McGraw-HiII, New York, London*, 1983.

[34] Gordon M Wenz. Acoustic ambient noise in the ocean: spectra and sources. *The Journal of the Acoustical Society of America*, 34(12):1936–1956, 2005.

[35] National Marine Sanctuary Stellwagen Bank. Passive acoustic monitoring. [Online; accessed 20-April-2014].

[36] Evologics. S2cr 7/17 product information. [Online; accessed 14-February-2014].

[37] Norman Abramson. Development of the alohanet. *Information Theory, IEEE Transactions on*, 31(2):119–123, 1985.

[38] Phil Karn. Maca-a new channel access method for packet radio. In *ARRL/CRRL Amateur radio 9th computer networking conference*, volume 140, pages 134–140, 1990.

[39] Hai-Heng Ng, Wee-Seng Soh, and Mehul Motani. Maca-u: A media access protocol for underwater acoustic networks. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1–5. IEEE, 2008.

[40] Yongxin Zhong, Jianguo Huang, and Jing Han. A tdma mac protocol for underwater acoustic sensor networks. In *Information, Computing and Telecommunication, 2009. YC-ICT'09. IEEE Youth Conference on*, pages 534–537. IEEE, 2009.

[41] Helge Rustad. A lightweight protocol suite for underwater communication. In *Advanced Information Networking and Applications Workshops, 2009. WAINA'09. International Conference on*, pages 1172–1177. IEEE, 2009.

[42] Optimized link state routing protocol (olsr), 2003.

[43] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (aodv) routing, 2003.

[44] Brad Karp and Hsiang-Tsung Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254. ACM, 2000.

[45] Rong Ding and Lei Yang. A reactive geographic routing protocol for wireless sensor networks. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2010 Sixth International Conference on*, pages 31–36. IEEE, 2010.

[46] Gunnar Taraldsen, Tor Arne Reinen, and Tone Berg. The underwater gps problem. In *OCEANS, 2011 IEEE-Spain*, pages 1–8. IEEE, 2011.

[47] Yuh-Shyan Chen and Yun-Wei Lin. Mobicast routing protocol for underwater sensor networks. *Sensors Journal, IEEE*, 13(2):737–749, 2013.

[48] Z Song, Deshi Li, and Jian Chen. A link-state based adaptive feedback routing for underwater acoustic sensor networks. 2013.

[49] Hai Yan, Zhijie Jerry Shi, and Jun-Hong Cui. Dbr: depth-based routing for underwater sensor networks. In *NETWORKING 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, pages 72–86. Springer, 2008.

[50] Abdul Wahid, Sungwon Lee, and Dongkyun Kim. A reliable and energy-efficient routing protocol for underwater wireless sensor networks. *International Journal of Communication Systems*, 2012.

[51] Wikipedia. Wormhole — wikipedia, the free encyclopedia, 2014. [Online; accessed 26-February-2014].

[52] Jiejun Kong, Zhengrong Ji, Weichao Wang, Mario Gerla, Rajive Bagrodia, and Bharat Bhargava. Low-cost attacks against packet delivery, localization and time synchronization services in underwater sensor networks. In *Proceedings of the 4th ACM workshop on Wireless security*, pages 87–96. ACM, 2005.

[53] Gianluca Dini and Angelica Lo Duca. Seflood: A secure network discovery protocol for underwater acoustic networks. In *Computers and Communications (ISCC), 2011 IEEE Symposium on*, pages 636–638. IEEE, 2011.

[54] Cisco Inc. Flat versus hierarchical routing. [Online; accessed 13-October-2013].

[55] Jörg Kalwa. The racun-project: Robust acoustic communications in underwater networks—an overview. In *OCEANS, 2011 IEEE-Spain*, pages 1–6. IEEE, 2011.