FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

High Definition Wireless Video Streaming using Underwater Data Mules

João Pedro Teixeira Loureiro

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

Supervisor: Rui Lopes Campos

Co-Supervisor: Filipe Borges Teixeira

March 12, 2021

Resumo

As comunicações sem fios em meios aquáticos e subaquáticos revelaram desde sempre uma grande dificuldade de implementação devido às grandes limitações e características adversas dos ambientes marítimos. Nos últimos anos, o interesse pelo desenvolvimento de soluções tecnológicas que permitem realizar comunicações sem fios eficientes e com custo reduzido nestes meios tem aumentado substancialmente, motivado por várias aplicações práticas, tais como estações de monitorização ambiental, atividade da indústria petrolífera ou recolha de dados oceanográficos, incluindo filmagens subaquáticas, utilizando veículos subaquáticos autónomos (AUVs). Em meio subaquático, existem atualmente três tecnologias que são utilizadas para comunicações sem fios: comunicações óticas, ondas acústicas e rádio frequência. Apesar de cada uma delas possuir vantagens distintas, as suas limitações, em termos de alcance ou débito, fazem com que não seja possível construir um sistema fiável e de banda larga, principalmente a distâncias de centenas ou milhares de metros. Não existe, por isso, uma solução para transferência de vídeo em alta definição em tempo real para ambientes subaquáticos nas distâncias praticadas pelos AUVs.

Este trabalho pretende colmatar estas dificuldades de comunicação sem fios abordando, de maneira mais específica, o problema de transmissão de vídeo de alta definição para distâncias elevadas, na ordem dos quilómetros, e com débitos na ordem dos Mbit/s, utilizando o conceito de mulas subaquáticas autónomas, baseando-nos na solução GROW. Estas mulas transportam dados entre duas unidades, uma que recolhe dados de vídeo em profundidade, como um AUV (Survey Unit), e outra que recebe os dados para serem exibidos à superfície (Central Station Unit), tirando partido das características de cada uma das tecnologias de comunicações subaquáticas. A rede utilizada para executar as comunicações e transferência de dados é baseada no conceito de redes tolerantes ao atraso (DTNs), um tipo de redes que é capaz de lidar com elevados atrasos e conectividade intermitente. A contribuição desta dissertação consiste num mecanismo que permite a transferência de vídeo de forma contínua, fiável e adaptativa em ambientes subaquáticos.

Para alcançar este objetivo, foi desenvolvido uma mecanismo de ligação fiável, capaz de retransmitir os pacotes que são perdidos devido a falhas ou atrasos durante a viagem das mulas permitindo, assim, complementar a ligação de *streaming* com características *best-effort* e abrindo a possibilidade ao utilizador de rever, mais tarde, o vídeo de forma completa. Foi também desenhada uma versão melhorada do protocolo de comunicação fora de banda da solução GROW que permite o controlo, escalonamento e sincronismo entre as várias mulas. O sistema desenvolvido foi testado em ambiente subaquático laboratorial nas instalações da FEUP/INESC-TEC, de forma a validar a solução proposta. Os testes realizados consistiram em medir vários parâmetros relativos aos dois tipos ligação propostos (*best-effort* e fiável). Para isso, foram submergidos 3 nós (Survey Unit e duas mulas) num tanque com água doce, tendo para a Central Station Unit sido usado um PC colocado à superfície. Os resultados obtidos demonstraram que é possível obter uma transmissão de vídeo de forma contínua após poucas viagens das mulas assim como obter, com um atraso aceitável, os pacotes que falham para reprodução posterior, oferencendo uma solução com melhor Qualidade de Experiência para o utilizador quando comparada às utilizadas atualmente.

Abstract

Wireless communications in underwater scenarios have always revealed hard to implement due to the harsh characteristics of the maritime environment. Over the past years, the interest in developing technological solutions to perform efficient and cost-effective broadband wireless underwater communications has substantially increased motivated by practical applications such as environmental monitoring, oil and gas industry or ocean data collection, especially video recorded by Autonomous Underwater Vehicles (AUVs). There are three technologies that are currently being used in underwater wireless communications: optical communications, acoustic waves and radio-frequency. Despite the advantages of each technology, their limitations regarding data rate or range of operation make high-definition underwater video transfer unfeasible for distances greater than a few meters. Therefore, there is no solution for real-time high-definition wireless video transfer for underwater scenarios in the order of hundreds to thousands of meters, where AUVs are typically used.

In this dissertation, we aim to develop a system capable of performing adaptive high definition video streaming in a long range (in the order of the kilometers) and with data rates in the order of Mbit/s, taking advantage of the GROW concept of underwater autonomous data mules. These mules are able to transport data between two units, one that records the video data in deep water, such as an AUV (Survey Unit), and other that is at surface receiving and displaying the video (Central Station Unit), taking advantage of the characteristics of each underwater communications technology. The network used to perform the communication and data transfer between the mules and the other units is based on the concept of Delay-Tolerant Network (DTN), a type of network that can handle high delays and intermittent connectivity between network nodes. The contribution of this dissertation consist in a mechanism that allows continuous, reliable and adaptive video streaming in underwater scenarios.

To achieve this goal, we developed a new mechanism for reliable connection, capable of performing the retransmission of lost video chunks due to mule failure, thus complementing the close to real-time video streaming link. Also, it was developed and implemented an enhanced version of the out-of-band protocol presented in GROW solution that allows the control, scheduling and syncronization of different mules. The system was tested in an underwater environment in FEUP/INESC-TEC laboratory, in order to validate our proposed solution. The experimental tests consisted in measuring several parameters concerning both connections (best-effort and reliable). To do this, we have submerged three nodes (Survey Unit and two data mules) in a freshwater tank and used a PC to emulate the Central Station Unit, placing it at surface near the tank. The results obtained demonstrate that it is possible to obtain continuous video streaming after a few data mule travels and that the lost bundles are transmitted with an acceptable delay through the new reliable virtual connection, offering a new solution with better Quality of Experience for the user when compared with currently used systems.

Agradecimentos

Em primeiro lugar, quero agradecer aos meus orientadores, Doutor Rui Lopes Campos e Eng. Filipe Borges Teixeira por me terem aceitado no INESC-TEC e por me terem acolhido como parte da "família" no CTM. Sem o acompanhamento incansável de ambos, nunca teria conseguido terminar esta dissertação.

Teria sido bem mais difícil chegar aqui sem todos aqueles que me acompanharam nas "trincheiras" deste curso. Agradeço em especial àqueles que sei que levo como amigos para o resto da vida: ao Varges, ao qual devo não só uma quantidade de boleias absurda, mas também o empenho desumano nos trabalhos de grupo, ao Mestre Santos, por todas as lições de vida, e à Luísa, ao Hugo, à Paula, ao Tiago, à Sofia e a outros que sempre estiveram lá, obrigado.

Aos meus pais, nunca conseguirei agradecer o suficiente, não só por terem financiado esta aventura que foram os últimos 5 anos, mas por toda a educação, pelo apoio em todas as etapas da minha vida e, principalmente, por terem sempre acreditado em mim. Da mesma forma, agradeço aos meus irmãos, por toda a paciência e cumplicidade (sei bem que não sou nada fácil de aturar).

Agradeço, também, ao meu avô (que continua sem perceber muito bem que curso estou eu a tirar) e à minha família em geral, porém, um profundo agradecimento terá de ir, em especial, para ao Toninho e para à Fatinha, por me terem recebido na sua casa nestes 5 anos de forma extraordinária, onde me senti tratado como um filho.

Da minha casa ao Porto são quase 100 km de viagem e, como tal, é impossível esquecer a primeira cara que vejo sempre que retorno a casa de comboio: obrigado Tia Zeza, por todo o carinho que tem por mim e por todas as sandes de presunto e salpicão.

Durante estes últimos anos, dividi o meu tempo entre a FEUP e a Universidade do Minho, pelo que seria hipócrita não agradecer tanto aos meus colegas de casa emprestados, Sara e Collina, como também a simpatia e disponibilidade do pessoal de ambas as bibliotecas e respetivos bares, onde passava grande parte do meu tempo, a estudar.

Ora, se vivi entre o Porto e Braga, foi porque nesta última cidade era onde se encontrava a minha "boia de salvação". A ti, Filipa, será sempre impossível agradecer por tudo. Pelas noites em branco em que ficavas a fazer companhia enquanto estudava, pela preocupação e carinho que sempre demonstraste, pela boa disposição, mas, sobretudo, por acreditares mais nas minhas capacidades do que eu próprio, nunca me deixando desistir. Partilhamos os nossos percursos, com todos os altos e baixos que foram aparecendo e, no fim, conseguimos alcançar o que pretendíamos. Que estes sejam apenas os primeiros de muitos anos de vivência juntos!

Por último, mas decididamente, não menos importante, um agradecimento especial a duas das pessoas mais importantes da minha vida, uma que perdi no meio deste percurso e outra que, infelizmente, não me viu chegar sequer ao metro e meio de altura: às minhas avós, agradeço por terem feito de mim muito do que sou hoje, com exemplos de humildade, de sinceridade e de respeito.

"And if I have a prophet's power, and have knowledge of all secret things,
and if I have all faith, by which mountains may be moved
from their place, but have not love, I am nothing."

1 Corinthians 13:2

Contents

List of Figures

List of Tables

Abbreviations and acronyms

API	Application Programming Interface
ARQ	Automatic Repeat Request
AUV	Autonomous Underwater Vehicle
AVC	Advanced Video Coding
BP	Bundle Protocol
BSS	Bundle Streaming Service
CTM	Center of Telecommunications and Multimedia
DCT	Discrete Cosine Transform
DT-MDS	Disruption-Tolerant Multimedia Delivery System
DTN	Delay Tolerant Network
DTNRG	Delay Tolerant Networking Research Group
EID	Endpoint Identifier
EM	Electromagnetic
HAS	HTTP Adaptive Streaming
HD	High Definition
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Taskforce
ION	Interplanetary Overlay Network
IPN	Interplanetary Internet
ISO/IEC	International Organisation for Standardisation/International Electrotechnical Commission
IUT-T	International Telecommunication Union
JPL	Jet Propulsion Laboratory
km	Kilometer
LED	Light Emitting Diode
LOS	Line Of Sight
LPD	Licklider Transmission Protocol
MOS	Mean Opinion Score
NAL	Network Abstraction Layer
PD	Propagation Delays
PER	Packet Error Rates
PSNR	Peak signal-to-noise ratio
QoE	Quality of Experience
RF	Radio-Frequency
RTMP	Real-Time Messaging Protocol
RTP	Real-time Transport Protocol
SNR	Signal to Noise Ratio
SORT	System for Adaptive Transmission of Video Over Delay Tolerant Networks
SPICE	Space Internetworking Center

SD	Standard Definition
SVC	Scalable Video Coding
TCP	Transmission Control Protocol
TTL	Time-to-live
UDMP	Underwater Data Muling Protocol
UDMP-S	Underwater Data Muling Protocol for Streaming
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
VoD	Video on Demand
VCL	Video Coding Layer

Chapter 1

Introduction

1.1 Context

The interest in providing reliable solutions to perform underwater communications has increased substantially over the past years, motivated by applications such as environmental monitoring, oil and gas industries or marine information gathered by Autonomous Underwater Vehicles (AUVs), where video capture is frequent. The harshness of the ocean makes it very hard to implement a viable broadband communications system, namely to perform video transfer collected from the AUVs. Although there are three technologies for underwater communications, they can not provide high quality video transfer with real-time characteristics in a range of hundreds to thousands of meters. Optical Communications allow very high bitrates but they require line-of-sight and are affected by water turbidity. In the case of the Radio-Frequency, the bitrates obtained are generally high enough for video transfer, but a strong attenuation limits the effective range. Finally, acoustic communications provide a higher range of operation but the bandwidth is low due to the frequency of operation and the propagation of sound underwater, making them unsuitable to be used in video transfer, even at low resolutions. Among all the possible scenarios for using these type of communications, streaming video is a relevant one that requires high bitrates and involves the transfer of large amounts of data, making the use of existing approaches unfeasible. The GROW Data Muling approach [2], where data mules are used to carry data from a survey AUV and a station at the surface, offers a better solution to underwater scenarios, enabling broadband communications at hundreds to thousands of meters.

1.2 Motivation

The limits to existing solutions make imperative to find a new reliable strategy to provide underwater communications, especially for high-definition video streaming services. The GROW solution offers a new approach to solve this problem. The system is based in a data muling approach for broadband and wireless underwater communications. The data mules, which in this case are a type of AUVs, collect data from the underwater source AUV (Survey Unit) and travel to the surface to

deliver it to the Central Station Unit that made the request. The network is based on a Delay Tolerant Network (DTN) [3], a type of network designed for environments characterized by intermittent connectivity, allowing to cope with high delays in the response of the network nodes. A great deal of progress has already been achieved in [4], where the authors proved that the use of data mules to transfer large files is much more efficient than the traditional acoustic based solutions. In Figure 1.1 we can see an illustration for the GROW proposed solution.

Despite the advances to the state of the art, the previous work is not focused on streaming data, where a constant flow of data should be received to avoid buffer underruns and a bad Quality of Experience. The need for video streaming in this environments covers several practical applications such as environmental monitoring and object recovery in shipwrecks. If we take as example the recovery of a flight recorder (commonly known as black box of an airplane), the time we have to wait to get the first video images collected from the survey AUV should be the shortest possible. In the same way, the results of environmental monitoring have to be gathered in a short period of time to maintain an updated environmental characterisation. Considering these aspects, the solution proposed in this dissertation aims to offer a better performance for video streaming in these scenarios.

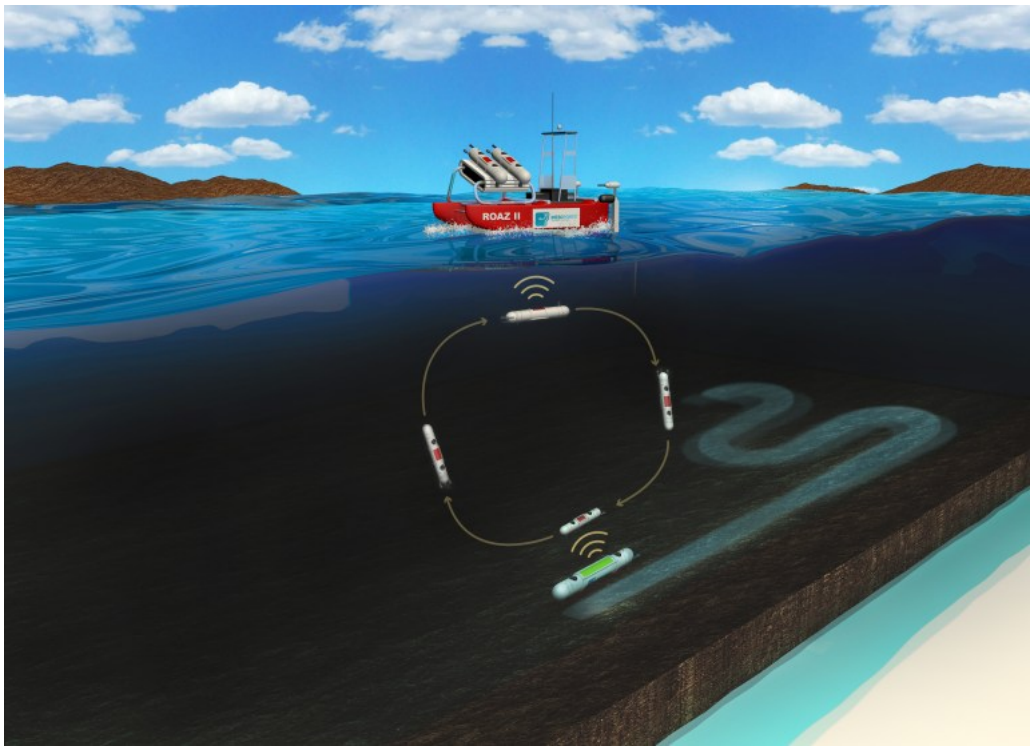


Figure 1.1: GROW solution for broadband underwater wireless communications [5]

1.3 Goals

Based on previous work developed at INESC TEC, the main goal of this dissertation is to provide a video streaming service using a data muling approach that enables a constant flow of video and avoid buffer underruns and low Quality of Experience. For this purpose, since the work developed so far is only designed to deal with regular files and not to stream video data, we intend to extend it to schedule the data mules properly for video streaming and to prepare the entire system for this objective. To achieve this goal, the following main objectives will be presented:

- Design of a retransmission mechanism for reliable data exchange;
- Develop an adaptive bitrate video streaming mechanism to cope with the variable capacity provided by the data mule network;
- Design and implement an enhanced version of the Underwater Data Muling Protocol (UDMP) presented in [2] to enable control and execution of a scheduling mechanism of underwater data mules in a DTN;
- Perform tests in a lab testbed composed of one Survey Unit, one Central Station Unit and two Data Mule Units.

1.4 Contributions

The main contribution of this MSc dissertation is the design and implementation of a mechanism that allows high-definition, continuous, reliable and adaptive videostreaming in underwater scenarios over a DTN network. The solution intends to offer a step forward in maritime environment missions providing a faster and more reliable way to stream video data in underwater environment.

1.5 Document Structure

This document is structured in six chapters. Chapter 2 presents the current state of the art concerning underwater wireless communications, Delay-Tolerant Networks and adaptive video streaming techniques. In Chapter 3, the concept and details of the proposed solution are described and analysed. Theoretical assumptions are also presented in order to serve as a comparison point with the experimental results. The testbed design, the implementation specifications and the experimental planning are depicted in Chapter 4. Chapter 5 contains the experimental results and discussion. Finally, in Chapter 6, we draw the conclusions and present the future work.

Chapter 2

State of the Art

In this chapter, we present the state of the art related to the underwater wireless communications technologies, the usage of Delay-Tolerant Networks for transferring high amounts of data, and the principles of adaptive video streaming. We also provide an overview of existing solutions for streaming using DTNs.

2.1 Principles of Underwater Wireless Communications

For several years, underwater wireless communications have been limited to ranges of just a few meters or to applications that use low data rates. This is due to the communication challenges offered by maritime environments.In [6], several limitations in underwater and maritime communications are presented in order to establish a foundation to motivate a new research interest in this area. The authors show that, although developing of maritime networks is more needed than ever, there are only a few solutions that are truly practical but more research in this field is needed.

There are, essentially, three main technologies that are used to implement underwater wireless communication systems: Radio-Frequency (RF), acoustic waves and optical communications. Every approach has advantages and limitations. In [7] a list of the characteristics of this 3 approaches is presented and we have summarised them in form of a table that ca be seen Table in 2.1.

Acoustic communications is a proven technology for underwater communications having transmission ranges up to 20 km. It is not a cost-effective solution due to the cost of the transceivers

Table 2.1: Main characteristics of each underwater wireless communications technology

Technology	Acoustic	Optical	Radio
Main Advantage	Acceptable Range	Very high data rates	High data rates
Main Disadvantage	Low bandwidth and efficiency	Easly afected by external factors and Line-of-Sight	Low range due to electromagnetic properties

and the high power consumption. Also, they provide a limited bandwidth and they can be easily affected by maritime environment characteristics such as turbidity, ambient noise, salinity and pressure gradients. In addition, acoustic waves can have an adverse impact on marine life, which should not be neglected. In the case of the RF solutions, a higher bandwidth can be achieved and aspects like turbidity, ambient noise, salinity or pressure gradients do not affect the communications performance. Nonetheless, RF communications are susceptible to electromagnetic interference and have a limited range through water due to strong attenuation. Finally, optical communications have the great advantage of ultra-high bandwidth: up to gigabits per second. Despite this benefit, optical systems systems are still applicable to short range scenarios, do not cross water/air boundary easily and require line-of-sight.

In the next 3 subsections we analyze in more detail each of these types of underwater wireless communications, presenting simple characterizations and models for comparison.

2.1.1 Acoustic Communications

As stated previously, acoustic underwater communications have an acceptable range of operation, but they also suffer from low bitrates and high delays. To evaluate the performance of this type of communications we should look into a characterization model. Khan *et al.* [8], show that the speed of sound in underwater mainly depends on a few aspects such as the pressure, salinity and temperature. The variation of sound speed due to temperature and pressure is high in shallow and deep water respectively. A key parameter to evaluate this type of technology is explained in [8] and refers the Propagation Delay (T_p) that represents the time taken to reach the destination from the sender. T_p can be calculated through the quotient between the distance d, (in meters) from the destination from the sender and the speed of sound underwater c (in meters/second) as we can see in Eq. 2.1.

$$T_p = \frac{d}{c} \tag{2.1}$$

The speed of sound c is obtained through a formula depicted in Eq. 2.2 that depends on Temperature T in degrees Celsius, Salinity S in parts per thousand and Depth of the water D in meters. This formula was proposed by Medwin in [9].

$$\begin{aligned} c = 1449 + 4.6T + 0.0055T^2 \times 10^{-2}T^2 + 2.374 \times 10^{-4}T^3 + 1.340(S-35) + 1.630 \times 10^{-2}D \\ + 1.675 \times 10^{-7}D^2 - 1.025 \times 10^{-2}T(S-35) - 7.139 \times 10^{-13}TD^3 \end{aligned} \tag{2.2}$$

In [10], other two parameters are analyzed concerning the performance of acoustic underwater communications: Transmission Loss (T_L), which represents the decrease in sound intensity from sender to receiver and Spreading Loss (P_L) that can be considered a type of transmission loss, which occurs at the time when sound travels away from the source to destination . The T_L can be obtained using Eq. 2.3.

$$T_L = SS + \alpha \times 10^{-3} \tag{2.3}$$

SS is the spreading factor and can be calculated through Eq. 2.4 and α is the Attenuation factor that is given by Eq. 2.5, where r is the range in meters and f the frequency in kHz.

$$\alpha = \frac{0.11f^2}{1+f^2} + \frac{44f^2}{4100+f^2} \quad [dB/km] \tag{2.4}$$

$$SS = 10log(r); \tag{2.5}$$

The Spreading Loss P_L is given by Eq. 2.6. It can be of two types: spherical spreading ($k = 2$) and cylindrical spreading ($k = 1$). Once again, r is the range in meters.

$$P_L = k \times \log(r); \tag{2.6}$$

There are a few other aspects that can be found for modeling acoustic communications that are presented in ([10]), such as *Signal to Noise Ratio* (SNR) or Absorption Loss but are out of the scope of this work.

A list of some acoustic modems available on the market is depicted in Table 2.2 [11, 12], showing the characteristics of each modem. It is possible to see that the maximum distance is 10 km at a maximum rate of 5 kbps. On the other hand, the maximum rate obtained is 31.2 kbps at a range of 1000 m.

Table 2.2: Available acoustic modems and their characteristics

Model	Distance (m)	Rate (kbps)	Operating Frequency(kHz)	Power (Watts)	Depth (m)
LinkQuest UWM1000	350	9.6 to 19.2	26.77 to 44.62	2	Up to 200
LinkQuest UWM2000	1500	9.6 to 19.2	26.77 to 44.62	8	Up to 4000
LinkQuest UWM3000	5000	2.5 to 5	7.5 to 12.5	12	Up to 7000
LinkQuest UWM4000	4000	4.8 to 9.6	12.75 to 21.25	7	Up to 7000
LinkQuest UWM10000	10000	2.5 to 5	7.5 to 12.5	40	Up to 7000
EvoLogics S2CR 48/78	1000	31.2	48 to 78	60	Up to 2000
EvoLogics S2CR 42/65	1000	31.2	42 to 65	60	Up to 2000
EvoLogics S2CR 18/34	3500	13.9	18 to 34	80	Up to 2000
EvoLogics S2CR 7/17	8000	6.9	7 to 17	80	Up to 6000

These characteristics let us conclude that acoustic communications are not suitable to transfer large amounts of data such as videos, pictures or bathymetric information in long-range scenarios. Despite all these limitations, acoustics allow the establishment of a out-of-band channel for control purposes.

2.1.2 RF Communications

Since the beginning of wireless connections, Radio-Frequency (RF) communications [13], have been one of the most used technologies to establish links to exchanged data: Wi-Fi, analog radio or television broadcasting are some application examples of this technology.

Although this kind of communications is accepted as the main solution for wireless connections on terrestrial scenarios, in underwater environments it has been considered not very effective because electromagnetic radiation has limited range underwater [7].

To evaluate underwater RF performance we can pick a few parameters that affect the propagation of EM waves such as conductivity σ and wavelength λ. Pure Water is a pure insulator [14]. However, this is not the case of water found in its natural state because it contains dissolved salts and other particles that makes its conductivity not null. In [14], the authors state that the higher water's conductivity, the greater the attenuation of the radio signals that pass through it. The attenuation of radio waves in underwater environments, α (in dB/m) - Eq. 2.7, increases with both frequency of the wave/signal f (in Hz) and conductivity σ (in S/m):

$$\alpha = 0.0173\sqrt{f\sigma} \tag{2.7}$$

We can also consider the wavelength λ referred above. To obtain this variable (in meters) we can use Eq. 2.8

$$\lambda = 1000\sqrt{\frac{10}{f\sigma}} \tag{2.8}$$

As we can see, the wavelength, unlike the attenuation, reduces its value with both frequency and conductivity. These considerations lead to the conclusion that the values of wavelength in terrestrial and underwater environments will differ much from each other making the design of antennas for these two scenarios very different. In a project developed also at INESC TEC [15], the tests and simulations performed in a dipole antenna designed for underwater use concluded that the input impedance and resonance frequency of the antenna increased when the conductivity of the water also increases, meaning that lower frequencies can be used in seawater for the same antenna size.

In summary, RF underwater communications are useful for broadband short-range communications provided that sub-GHz values are used, to cope with the increase of wave/signal attenuation. This reasoning lead to consider its future use in this work for the short-range link between the mule unit and the survey/station units as a viable solution.

2.1.3 Optical Wireless Communications

Optical communications take advantage of the properties of light (optics) to transmit information. Similar to RF solutions, optical communications also allow high throughput in the order of several Mbit/s and even Gbit/s at short-range links. In [16], it was proved that the use of Light Emitting Diodes (LEDs) can be used for high-speed underwater visible light communications. This approach has also the benefits of low latency thanks to the high propagation speed of light in the underwater and a possible lower cost when compared to acoustic based systems [17]. Despite these advantages, optical communications suffer from several limitations such as the need of line-of-sight (LOS) and the attenuation that, although it is generally low in optical signals, it increases considerably with distance. Attenuation, measured by the Attenuation Coefficient, can fluctuate

with different factors, such as the type of water in the deploymed scenario. In [18], the Propagation Loss Factor in optical communications was measured for different types of water at different distances as depicted in Figure 2.1.

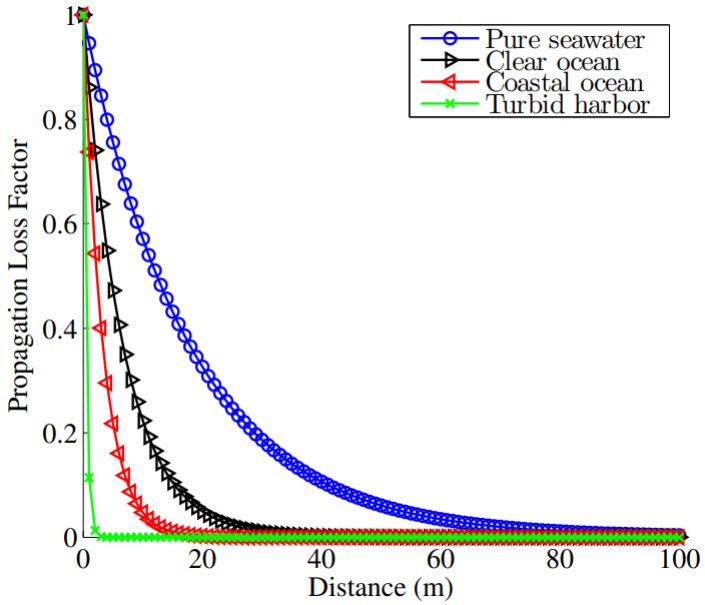


Figure 2.1: Attenuation of the optical signal considering water types and distance [18]

As we can observe, the type of water can be a factor that causes a strong influence in the performance of optical communications in underwater scenarios due to phenomenons called absorption and scattering, that depend both on the biological and chemical structures of the water [17].

Concerning the wavelength used for transmission, we may see in Figure 2.2 the variation of the absorption coefficient of electromagnetic radiation at different wavelength values. The conclusion obtained by the inspection of this graphic is that visible light frequencies are the ones that suffer less attenuation, making them the best choice to use in this type of communications.

In summary, optical wireless communications can provide data rates in the order of Mbit/s and generally systems based on this approach can be designed with a lower cost than acoustic communications. Despite these benefits, optical technologies also suffer from big limitations being the need of LOS the main one, making this solution only suitable for a limited set of applications.

2.2 Delay-Tolerant Networks

Nowadays, almost every electronic device has some kind of Internet connection as one of its main features. The notion that these connections have to be available even in the most remote scenarios lead to the concept of Delay-Tolerant Network (DTN), defined by the Internet Engineering Taskforce (IETF) in [3]. As we will see later in this section, a DTN is a network of smaller networks

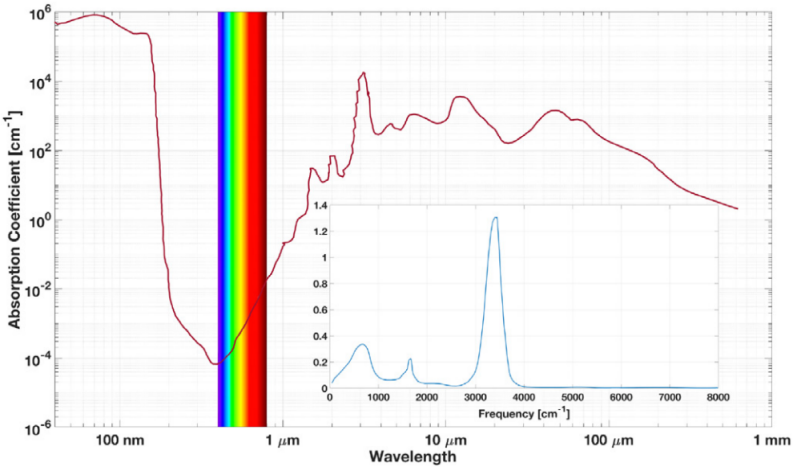


Figure 2.2: Absorption coefficient of optical waves in the aquatic medium [17]

that support connections between network nodes with long delays and/or disruption, making it an important principle for the solution presented in this dissertation.

2.2.1 Principles

Originally, DTNs were developed for interplanetary use where even the speed of light seems slow [19]. However, currently DTNs are used for a large set of applications here on Earth where disruption and delay-tolerance are most needed. This is the case of underwater wireless communications due to the limitations we have shown in Sec. 2.1. The Internet expects short round-trips, continuous and bidirectional end-to-end path, low and symmetric bit rates. This is not the case in underwater wireless networks, so the protocols usually used for communications as, for example, TCP/IP are not designed for these scenarios.

2.2.2 Routing in DTNs

DTNs offer an solution to deal with problems like intermittent connectivity and temporarily broken links as well as momentaneous high error rates. For this purpose, DTNs use a method known as *store-and forward message switching* [20]. This method started to be used decades before Internet at postal systems and consists in moving entire data packets (or fragments) from one storage place in one node to other storage place in another node. Based in this method, a DTN router needs to have persistent storage to hold messages for a considerable amount of time for several reasons, such as [19]:

- A link to the next hop may not be available for a long time;
- One node may send or receive data much faster or more reliably than other node;
- A message may need to be retransmitted if an error occurs or if a receiving node declines acceptance of a forwarded message.

2.2.3 Intermittent Connectivity

Today, communications devices such as smartphones are constantly in motion and operate on limited power. This can cause connection losses due to the obstructions [19]. When this occurs over the Internet, the data is lost and later, when the connection is re-established, TCP/IP mechanisms perform the retransmission of the lost packets. However, this does not happen when we are using DTNs. These networks support communications between intermittently connected nodes by isolating delay and disruptions with the store-and-forward technique as referred above. We can consider two types of intermittent connectivity:

- **Opportunistic Contacts**: in this case, two nodes make a contact that was not scheduled i.e. two nodes that happen "by chance" to be in line-of-sight exchange information.
- **Scheduled Contacts**: these contacts take place when we know that two nodes can be unreachable for a long time but with predictable future positions in space, allowing to schedule information exchange taking into account the delay expected. This type of contacts require time-synchronization throughout the DTN.

2.2.4 Bundle Protocol

Now that we have presented the main characteristics of a DTN, we must take a look on some details of its protocol architecture. To implement the store-and-forward message switching technique, DTNs use a new protocol - the *bundle protocol* (BP) [21]. This new protocol stays between the application layer and the other lower layers in the protocol stack as depicted in Figure 2.3. The interface defined between the BP and the internetwork protocol suite is called *convergence layer adapter*.

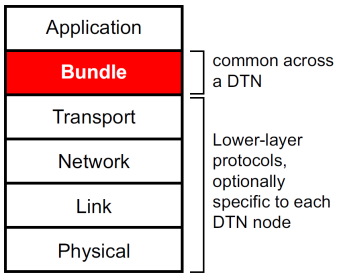


Figure 2.3: DTN Protocol Stack [19]

The BP data unit is called a *bundle* that includes: 1) a header consisting of one or more DTN blocks inserted by the BP agent, 2) a source-application's user data, including control information sent from source node to destination node defining how to process, store, dispose of and otherwise handle the user data, and 3) an optional bundle trailer, composed of zero or more DTN blocks, inserted by the bundle protocol agent [19]. The size of a bundle is arbitrary. Since in intermittently connected links long delays are expected, DTN nodes communicate with each other using minimal

or even without roundtrip. Any acknowledgement from the receiver is optional, depending on the service selected.

Concerning DTN nodes, they are an entity with a BP agent overlaid on lower-layer communication protocols and their identification is done using Endpoint Identifiers (EIDs), that can be treated as Uniform Resource Identifiers (URIs) [22]. They can act as source, destination or forwarding node. In the case of being a forwarding node, DTN nodes can be of 2 types:

- **Routing-equivalent Forwarding**: the node forwards bundles between two or more nodes that implement the same lower protocols;
- **Gateway-equivalent Forwarding**: the node forwards bundles between two or more nodes that implement different lower protocols.

DTNs support node-to-node retransmission of lost data at both the transport and the bundle protocols. This process is done using a technique know as *Custody Transfer*. Using this technique, when a node forwards a bundle to another it also sends a custody transfer request. If the next node accepts the request, the custody is now of this second node and the data should be stored in persistent memory until the custody is transferred to another node.

The Bundle Protocol and DTNs were the basis for several works concerning these networks that face challenges because of several natural and non-natural limitations (called challenged networks). One example is the system developed in [23] that consisted in bringing the Internet connection from a city to an isolated village, relying on the data transmission from the vehicle traveling between the two places. Another application that was tested in simulation was a "car-to-car" communication system in [24].

2.2.5 Bundle Protocol Implementations

Since the BP was defined in [21], several implementations were proposed, mainly open-source and developed for Linux/UNIX based systems. The three main open-source implementations that were studied and tested in [25] are:

- DTN2 [26]: developed by the Delay Tolerant Networking Research Group (DTNRG), is the reference implementation of the Bundle Protocol. It provides a flexible framework for DTN experiments and extensions for routing, storage and convergence layers can be easily installed through XML interfaces;
- ION (Interplanetary Overlay Network) [27]: a BP implementation designed to run in robotic spacecrafts on a Real-time operating systems by Jet Propulsion Laboratory (JPL). The system is based on shared memory concept and it is much like a database;
- IBR-DTN [28]: a *lightweight, modular and highly portable Bundle Protocol implementation* designed for embedded systems running OpenWRT [29].

In [25] the authors concluded that, in a general overview, IBR-DTN outcomes the other two implementations, specially in terms of throughput. For this reason and because IBR-DTN already have a streaming application that can be explored, it was the implementation chosen for this work. In Figure 2.4 we can observe the IBR-DTN architecture overview.

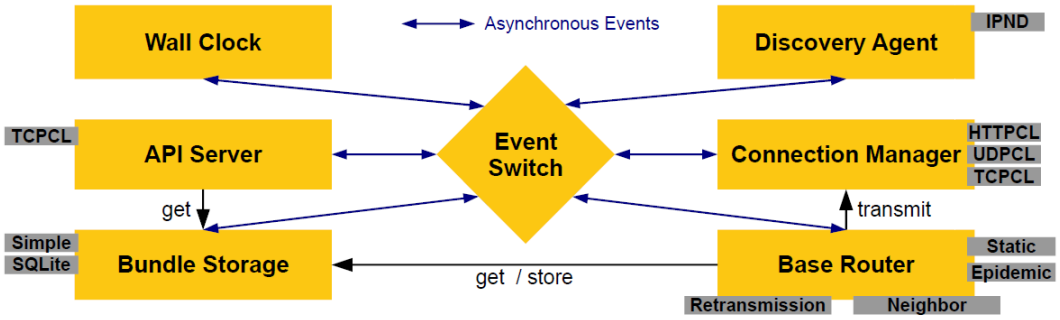


Figure 2.4: IBR-DTN architecture overview [28]

2.2.6 DTNs in underwater scenarios

In previous subsections, we have seen that the concept of DTN can be used in environments where we can not maintain a constant and permanent end-to-end path between the communications nodes. The conditions of an underwater wireless communications system, like the one proposed in this dissertation, is identical of the ones where DTNs were used previously.

In [30] it was demonstrated that data muling using AUVs in underwater scenarios is a very useful approach for long-term environmental monitoring or surveillance. In [31] and [32], the authors present experimental results related to underwater communications systems using two different bundle protocol implementations. Both use acoustic modems to perform the communications between the underwater nodes. The use of the DTN concept in these environments was proven to improve the communications performance with either of the two tested implementations. It was also proven, specially in [31] and [33], that the routing algorithms are also important in the performance of an underwater DTN. In [34], the author evaluated and studied the use of data mules to transfer data in an underwater scenario using the concept of DTN. This work showed that this solution to exchange data between underwater nodes has a better performance, i.e. better equivalent throughput and range when compared with existing solutions.

Following the reasoning of the previous referred work, the mechanism and the implementation were revisited in [4] where, as mentioned previously in 2.2.1, a new protocol was developed. The Underwater data Muling Protocol (UDMP) consists in a communications protocol that enables the control an execution of the scheduling of the data mules. The results have shown that, even using just one data mule, this solution is better than a traditional acoustic link and that the transfer of large amounts of data with this approach is preferable because most time is spent in the data mules travel between sender node and receiver node. As explained before, it is in this context that this

dissertation appears, in order to go beyond of the successful transport of regular files and perform high definition video streaming using an identical system based in data muling in an underwater DTN.

2.3 Adaptive Video Streaming

Video streaming applications and services are truly sensitive to changes in the link quality between the sender and the receiver. Since nowadays most traffic in the Internet consists of video streaming and VoD (Video on Demand) services, a new approach was developed in order to contour these unpredictable changes on the video transfer: adaptive streaming.

Adaptive Streaming techniques aim to improve user's QoE, analyzing both network properties, such as bandwidth, and the user's device characteristics, such as CPU capabilities in order to adapt the video streaming properties as, for example, bitrate or frame rate.

Although there are several video characteristics that can be changed in adaptive streaming technologies, hereafter we will focus only on the adaptive bitrate ones, which are the most commonly used. Before we show existing solutions for this kind of applications, we start by presenting the basic concepts of video coding.

2.3.1 H.264/AVC (Advanced Video Coding)

Created by the International Organisation for Standardisation/International Electrotechnical Commission (ISO/IEC) and International Telecommunication Union (ITU-T), H.264 or Advanced Video Coding (AVC) defines a format for video compression and decompression that is currently used across the world [35]. Published in 2003, it was based in MPEG-2 and MPEG-4 standards and the main objective was to provide a more reliable and efficient solution to compress video.

H.264 syntax definition is based on two concepts: profiles and levels [36]. Profiles are intended to define which techniques or algorithms are going to be used to generate the bitstream, making a convenient meeting point for device manufactures or video producers. On the other hand, levels are meant to specify the maximum data rate and video resolution that a device can play back.

Now, considering the compression process itself, we have to consider two types of compression techniques. The first is prediction, which computes only the frame values that have changed since the last frame. The other one is transformation that essentially performs a frequency analysis, using Discrete Cosine Transform (DCT) for example, in order to reduce sample repeatability. AVC enables to use a combination of multiple encoding techniques and algorithms to compress a video file.

H.264 format divides the video compression in two different layers [37]: Video Coding Layer (VCL), which is the one responsible for truly preforming the coding process, and the Network Abstraction Layer (NAL), which works with information related to the video data. VCL units are encapsulated into NAT units, in order enable the existence of inter-prediction encoding, i.e. analyze the previous units in order to have a better and efficient compression.

A coded video sequence in H.264/AVC consists in a group of coded pictures. Every picture is partitioned into fixed size macroblocks that are basic building blocks for the decoding process is specified. These macroblocks are organized into slices, which represent regions of a given image that ca be decoded independently. A picture can have one or more slices and they do not need to be ordered. A schematic structure for the coding process using H.264 can be seen bellow in Figure 2.5.

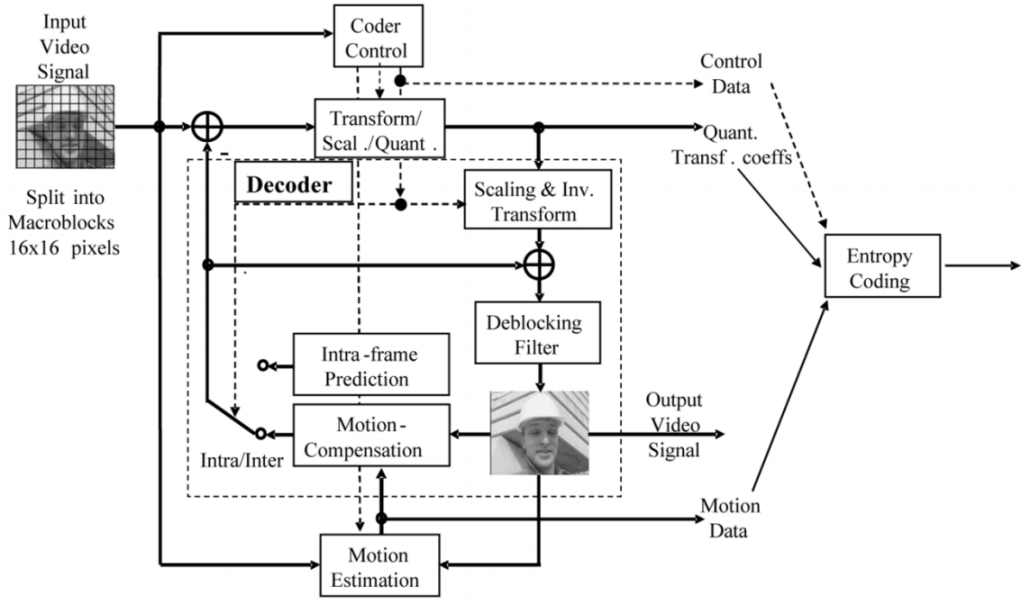


Figure 2.5: Basic coding structure for H.264/AVC for a macroblock [36]

In short, as we can see in [36], although H.264 is similar to previous video coding standards, its performance is superior and presents several improvements, such as the use of spatial transform coding for both inter-prediction (between frames/units) and intra-prediction (between samples of the same frame/unit).

2.3.2 Adaptive Bitrate Streaming Solutions

As stated previously, live streaming is a delay sensitive service and its time critical characteristics make the implementation of such services way more complex that similar ones like VoD systems [38]. The difficulties found to guarantee user's QoE lead to the adaptive streaming approaches, as we have seen before in the beginning of this section.

One of the concepts that have emerged for adaptive bitrate solutions is the Scalable Video Coding (SVC). It was defined in the Annex G of the H.264 standard ([35]) and it is a layered coding technique where the video is distributed across different layers. Each of these layers or subsets can be recombined to obtain different bitrates. The lower layer is the first to be transmitted and consists in the lowest bitrate video data in a initial phase called Initial Quality Adaptation (IQA). After verifying the quality of the transmission using different parameters, more layers can be added, increasing the bitrate, and consequently improving the quality of the video. This

latter phase is denominated Progressive Quality Adaptation (PQA). If the transmission resources decrease, layers are removed, in order to establish a effective video streaming again.

In [39] the most recent implementations and solutions for this kind of systems are presented. Since it is the most common protocol used in the Internet nowadays, all these solutions relay on Hypertext Transfer Protocol (HTTP), becoming known as HTTP Adaptive Streaming (HAS) services. Before HAS approaches became standard use, protocols such as Real-Time Transport Protocol (RTP) or Real-Time Messaging Protocol (RTMP) were the choices for most of these systems. The use of these two latter protocols have decayed in adaptive streaming services due to the fact of being connection-oriented protocols.

HAS implementations follow a typical session pattern. First, after the session is started, the client receives what is called the manifest where the metadata of the video, audio, subtitles, and other features are contained. Then, the service constantly keeps track of certain parameters such as available network bandwidth, buffer status, device battery, CPU levels and others. According to the measurements results, the HAS client fetches, several times, the most suitable of the video representations among the one available at the server.

HAS-based solutions can be splitted in four categories considering the place where they are implemented, i.e. where the adaptation decisions are made: client-based, server-based, network-based or hybrid-based. On the Internet, client-based solutions are the favourite choice [39]. The most used implementations are HTTP Live Streaming (HLS) [40] and Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [41]. The first one is property of Apple, Inc. and the second is open source. MPEG-DASH have conquered a lot of trust lately due to the simplicity and to the fact that no license is required for its use.

Several problems concerning HAS systems, such as the frequent bitrate switching or the competition for the available network bandwidth, can decrease the user's QoE. Since there is a lack of models to evaluate QoE in adaptive streaming services, a study to obtain some conclusions was performed in [42]. In this paper, the authors have obtained a list of the best parameters and some conclusions to evaluate the performance of adaptive streaming systems and to define QoE. To obtain such data, objective parameters like Peak signal-to-noise ratio (PSNR), and subjective ones like Mean Opinion Score (MOS) were used. According to the results in [42], the characteristics that can better suit as QoE parameters in this type of services are: bitrate distribution, average bitrate, number of bitrate changes, stall time, etc. The authors have also reached some conclusions, namely that the fluency of the video service influences much more the QoE than the video definition/quality; the client prefers to have a good startup bitrate even it decreases later; frequent switching bitrates severally influences QoE.

To close this section and analyzing every peace of bibliography read, the conclusion is that the adaptive bitrate video streaming results are better than for fixed bitrate. For this reason, it is our objective to develop a mechanism for adaptive video streaming for underwater data muling systems.

2.4 Streaming using Delay-Tolerant Networks

Data streaming services, especially video streaming services, are everywhere these days. When we think about this type of services, we tend to imagine them with real-time characteristics, i.e. live video streaming. Certainly this is not the case when we talk about streaming in underwater scenarios. In last sections we have seen how difficult it can be to maintain simple communications between underwater nodes and, because of that, a streaming service is still a challenge in these environments. In this section, we provide some examples of streaming solutions over DTNs found in the literature, including adaptive streaming solutions for these networks.

2.4.1 Viability of a streaming system using DTNs

Several works support the use of DTNs to implement streaming services where the communication suffers from great limitations. In [1] a testbed was designed to demonstrate that streaming a large amount of data can be feasible in a DTN. In this system, a video was captured by a camera and streamed over four DTN nodes: a sender connected to the camera, two other nodes connected to it and a receiver also connected with the previous two. Two network links are provided to guarantee that the video is only interrupted if both nodes are unavailable. A picture of this system can be seen in Figure 2.6. At the destination node, the stream is received and displayed with a delay that depends on the local video buffer size. The camera provides a constant real-time transport protocol (RTP) and both the receiver and the sender are regular PCs running Linux while the other nodes are are embedded devices running OpenWRT [29].

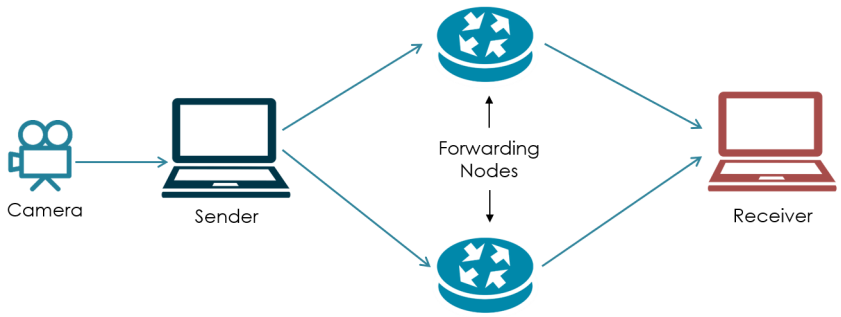


Figure 2.6: Proposed Streaming System Architecture [1]

The implementation of the bundle protocol used was the IBR-DTN and the application implemented on this paper is now part of the IBR-DTN Tools [43]. The operation of the application can be resumed in a few steps:

- In the sender, the data is gathered for a short time, encapsulated into bundles and then forwarded;
- The receiver re-orders the bundles using the sequence number of the stream-block;
- Optionally, a timeout can be defined to the lost bundles. Since there is no retransmission of lost bundles, this is the only way to deal with this issue.

The authors conclude that the implementation of streaming systems in DTNs is feasible without any losses and the reliability is also increased.

2.4.2 Solutions and Guidelines for Streaming in DTNs

Although the idea of streaming data in challenged networks implemented by DTNs is a concept that was not widely explored, several authors have already proposed some concepts and guidelines to design a system with this characteristics.

a) Robust Streaming in DTNs

If we take a look to [44], a on-the-fly coding mechanism called *Tetrys* was proposed and tested with the objective of providing fast and full reliability without retransmission in-order bundle delivery in comparison to classical erasure coding schemes. In this paper it was also studied how the DTN characteristics impact on the overall performance of streaming applications. As results, despite of a relatively stable average transmission delay, the standard deviation of the delay of each bundle in a given flow can be very large. The authors also argue that the way to design robust and reliable streaming services in DTNs is to prevent Automatic Repeat Request (ARQ) mechanisms, proposing *Tetrys* as possible solution.

b) Bundle Streaming Service

Another important existing work in this context was presented in both [45] and [46] and consists in a new framework implementation of the BP, specially designed for streaming purposes called the Bundle Streaming Service (BSS). Designed for Interplanetary Internet (IPN), BSS enables streaming data to be conveyed via DTN bundles supporting 2 parallel approaches:

- In-order Stream Processing with minimal latency (“real-time”) – best-effort option;
- Reliable delivery of all data (to replay later) – reliable transport option.

All network functionality of BSS is confined within the bundle layer except for the re-ordering of the packets that happens at the application layer. The bundles creation time is stored and every bundle sent by a BSS-enabled node must be custodially acknowledged. We can consider two basic components in BSS: a library for building streaming-oriented applications and a forwarder daemon. A test scenario was developed and the results presented in [45]. The scenario consisted of a network stack emulated on the DTN testbed of Space Internetworking Center (SPICE) and a streaming process that was simulated by developing: bssStreamingApp (an emitter application) and bssRecV (a receiver application). The results consisted in the evaluation of the values of two parameters - Propagation Delays (PD) and Packet Error Rates (PER) - in both the reliable connection that used TCP and the best-effort connection that used Licklider Transmission Protocol (LPD), which is equivalent to UDP but for space networks. These results can be seen in Figure 2.7.

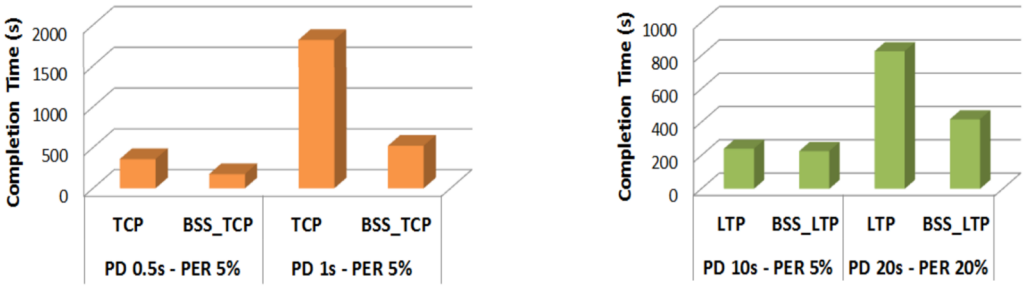


Figure 2.7: Results for PD and PER concerning the use of BSS [45]

It was then proved that BSS achieves better results only in cases where the error rate of the channel is above 10%, not being very effective when the error rate is lower.

This framework is a good approach if we want to maintain both a reliable and best-effort option in our streaming service but it is also not very explored when compared to regular BP implementations like DTN2 or IBR-DTN.

c) IBR-DTN: *dtnstream()* tool

To close this section about the related work concerning streaming in DTNs, a quick presentation of one of the IBR-DTN framework tools is given: the *dtnstream()* tool.

As described in its specification [47], this application allows to receive an internet radio stream on one DTN node and receive it with another node. A quick tutorial proposed to use it can be depicted in few steps:

1. Set up the receiver, pipe the incoming stream into a suitable player like *mp123* or *VLC*:

```
user@host2: dtnstream -s streamReceiver | mpg123 -
```

2. On the sending node, we use *wget* to get the stream and pipe it into dtnstream:

```
user@host1: wget -O - http://stream.laut.fm:80/jazzloft |
dtnstream  -d dtn://host2/streamReceiver -s streamSource
```

3. The stream is then received on host2 via DTN.

Despite the lack of documentation concerning this tool, it is a application provided by one of the most used frameworks that implement the bundle protocol i.e. IBR-DTN.

2.4.3 Adaptive Streaming in DTNs

In the previous section regarding current adaptive streaming solutions, we have referred our intention to implement this type of mechanism in this dissertation. Following this reasoning we have looked for state-of-art works concerning the implementing of adaptive streaming techniques in

DTN scenarios. Despite the lack of work in this field, we have found two related works that will be described in more detail.

In [48], since both HAS-based services and DTN nodes split multimedia data in segments, the authors believe that HAS-based multimedia delivery approaches can be suitable to use in DTNs. Also, they considered that although video streaming is time critical, in emergency situations the responders can not immediately watch the videos and have few time to lose. These considerations lead to the assumption that the videos should be short-duration and that it is better to wait a little longer for the first packet and having a better receiving quality. The main objective of this paper is to offer a concept-implementation for a HAS-based system in this scenarios called Disruption-Tolerant Multimedia Delivery System (DT-MDS). Basically, additional headers are considered in the messages, defining priorities for the representations of the video (lower bitrate representations have higher priority) and other properties for the video streaming.

Another work that explores the implementation of adaptive streaming in DTNs is shown in [49]. The proposed system is called SORT (System for Adaptive Transmission of Video Over Delay Tolerant Networks) and provides an end-to-end adaptation that only involves the application layer at the source and destination nodes. It uses the SVC concept and a notion called cumulative acknowledgment i.e. the acknowledgments have the delivery delay for the last few bundles, decreasing the impact when some bundles are lost. It is also proposed a Congestion and Delay Estimation algorithm (only between source and destination, not on the overall DTN network). Giving the results of the algorithm the system takes some measures. For instance, in the presence of severe congestion, the source does not transmit one or more of the higher layers. The TTL (Time-to-live) of higher layers can be reduced to ensure that they spend less time in the buffers in order to give priority to lower layers that are more urgent. At the end of the paper, it was shown that this system can provide significant performance gains when implemented in DTN video streaming scenarios.

2.5 Summary

Through the review of the state of the art, we can conclude that streaming high definition video underwater without cables for long distances is not possible with current solutions due either bandwith or range limitations. The use of DTNs has been successfully tested previously, and different solutions and concepts have already been evaluated, namely through IBR-DTN or BSS frameworks. The reliability of the connection over a DTN can be achieved through the use of two connections - one for best-effort and other for reliability, like in BSS. However, a proper retransmission mechanism is required. The data mule approach in underwater scenarios has been previously explored in several works, but there is still a lack of solutions regarding underwater communications that demand close to real-time operation. We can also conclude that an adaptive video streaming mechanism can improve the performance of such system, although this kind of approach has not been widely investigated in these scenarios.

Chapter 3

Proposed Solution

In this chapter we present the proposed solution for an adaptive and reliable broadband video stream for underwater networks based on data mules. All parts and details of the system architecture will be explained, including the Underwater Data Muling Protocol for Streaming (UDMP-S), the reliable mechanism and the adaptive bitrate mechanism.

3.1 Overview

Nowadays, there is a lack of solutions for reliable and long-range broadband wireless video transfer with real-time characteristics. The main reason for this gap are the limitations of current technologies for underwater communications, which can either provide long-range or broadband communications, and not both simultaneously.

An innovative solution for long-range broadband underwater wireless communications was designed in GROW [5]. This solution takes advantage of concepts such as DTN and AUVs to implement a system in which a Survey Unit (stationary or mobile) gathers data in the underwater, a Central Station Unit at surface works as a final receiver, and one or more Data Mules, which consist of fast and agile AUVs that travel between the previous two nodes, creating a virtual connection between . High bitrate short-range wireless communications are used to transfer the data between the data mules and the other nodes, and there are long-range acoustic out-of-band control link connects all nodes for control and scheduling purposes.

In this work we aim to build a solution for adaptive high definition video streaming in underwater scenarios, improving the system that have already been used for regular file transfer in [2]. Before we present a more detailed explanation for our proposed solution and the design of the testbed that was built to perform the tests, we start by providing an overview of how the system will work and how we can measure some parameters of its operation. Figure 3.1 depicts a two-dimensional representation of the general scenario where the GROW project is applied, and in concrete, the solution being presented in this work. In the model represented we can see that in GROW, the survey can be a mobile or a stationary node. This work will tackle the fixed position survey unit problem.

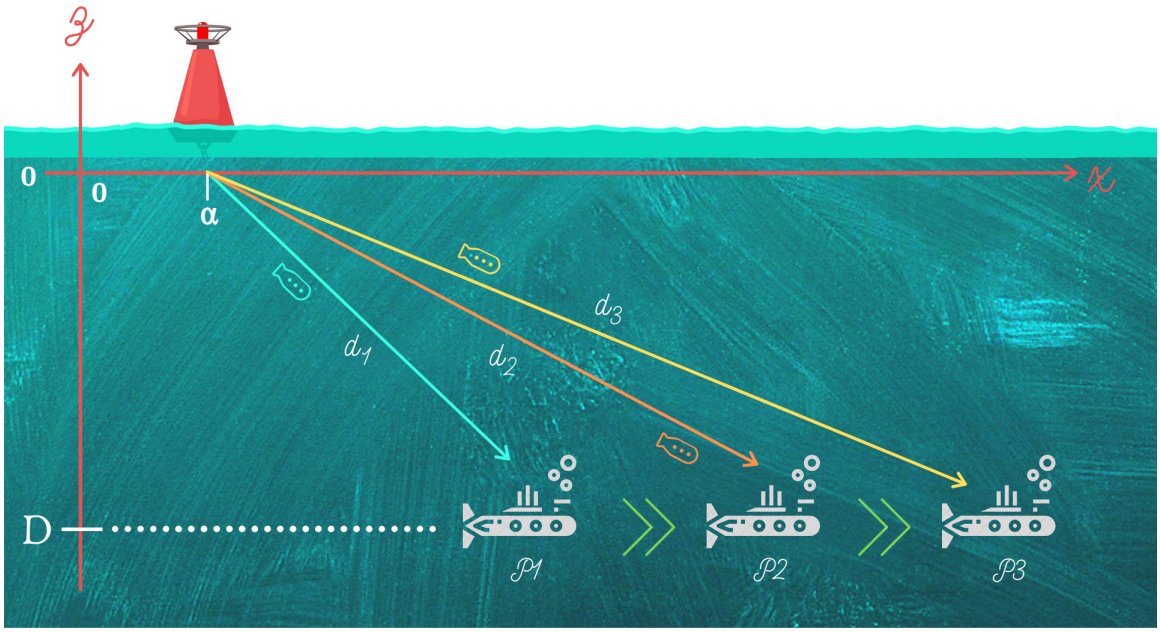


Figure 3.1: Two-dimensional model for a typical application scenario in GROW

Data muling has been successfully used for transferring large amounts of data in underwater communications. The solution for the adaptive and reliable streaming takes advantage of the GROW solution, which considers the usage of a DTN and AUVs.

In the same way as in GROW, the system considers three types of communication nodes: a Survey Unit that collects video data in, Data Mule Units that transport the video from the Survey Unit to the surface and a Central Station Unit that receives and displays the video stream.

The video data is transported through a DTN implementation using a RF based link for short-range communications and an acoustic based solution for long range control messages. We consider two virtual connections: one for adaptive best-effort purposes (video streaming) and other with reliable characteristics, i.e. to be replayed later, similarly to a VoD service and capable of handling a mule loss or excessive delay. The control of the data mules was done through an enhanced version of the Underwater Data Muling Protocol (UDMP) [4] tailored for the high definition streaming requirements. Since the data mules have to be scheduled correctly in order to maintain both virtual connections, this new version of the protocol is essential for the proposed system operation. Also, an adaptive bitrate video streaming mechanism is specified and takes into consideration the distance and the short-range speed to compute the recommended video bitrate.

3.2 Reliable Virtual Connection

Understanding the best-effort connection that supports the basic video streaming is easy, but the fact we have also implemented a second and reliable connection can be confusing to the reader, so in this section we will better explain why and how we have designed such approach.

This second connection, differs from the first since it is not time sensitive. The best-effort connection intends to offer the video data as quick as possible to the user, for example, for an emergency situation where we can not lose time or even for climacteric purposes, where the data has to be always updated. The reliable connection works very differently, like a VoD service, i.e. the video data is meant to be replayed and processed later, for applications where the user can wait a little longer in order to see more accurate video images such as in an ocean monitoring mission.

In our solution this is achieved as follows: the video streaming being transported by the data mules using the best-effort connection is temporarily stored as individual files (one for each IBR-DTN bundle) at the Station unit. Every time we detect a certain number of bundles was lost due to a mule failure, for instance, another data mule is sent to the Survey unit to retransmit the chunks of the video that were lost.

In the end, after receiving all bundles that were generated until the user closes the session, the individual bundle files are assembled into one unique file and it can be reproduced as a video content.

3.3 Adaptive Video Streaming Mechanism

The adaptive video streaming mechanism was conceptually designed, in the best-effort connection, as follows: first, the video would be coded in different bitrates at the Survey Unit. Then, we use the Control Channel (acoustic link) to send the Mule-Station connection parameters to the Survey Unit. The crucial estimator we wanted to use was the Effective Throughput - $R_{b,eq}$ - of this connection between the Mule Units and the Station Unit. Considering this Station message, the Survey Unit runs an algorithm that computes the best suitable bitrate to send to the Data Mule. At every Data Mule Unit arrival at the Station Unit, one of these messages containing the connection parameters ($R_{b,eq}$, etc.) should be sent.

3.4 Underwater Data Muling Protocol for Streaming

As stated previously, we have designed an enhanced version of the Underwater Data Muling Protocol, initially defined in in [4]. The Underwater Data Muling Protocol for Streaming (UDMP-S) have the same properties and structure of UDMP but we added messages concerning the control of the new reliable connection. Also, we adapted some of the messages that were used to retrieve files in order to accommodate them for video streaming purposes.

3.4.1 UDMP-S Specifications

The Underwater Data Muling Protocol (UDMP) which was previously designed in [4] is a communications protocol for data muling scenarios that enables the control and execution of the scheduling of the Data Mule Units. The API and messages of such protocol are prepared for scenarios where we want to exchange static files between nodes. In this dissertation, we decided to make some few modifications in this specifications in order to obtain the correct operation for video streaming, including new control messages for the best-effort link, the reliable connection and for the adaptive streaming mechanism. For simplicity we have called this new version of the protocol Underwater Data Muling Protocol for Streaming (UDMP-S).

Similar to the original UDMP, the UDMP-S protocol will be running in every of the 3 types of nodes in the DTN and runs in two wireless interfaces: 1) short-range high speed DTN implemented using the state-of-the-art framework IBR-DTN [28]; 2) a permanent long-range out-of-band acoustic link that connects all nodes to send and receive control commands.

UDMP already had ten message types defined for almost every basic operation needed in these scenarios (docking requests, mule requests, etc.). We decided to add 4 new types of messages for streaming purposes:

- **Type 11**: req_new_video_stream() - Used to request a new video streaming session by the user.
- **Type 12**: resp_video_stream_available() - Used to inform the Central Station that the Survey is available to start a new video stream.
- **Type 13**: req_mule_recover() - Used to request the lost Data Mule to return to the Survey Unit.
- **Type 14**: req_lost_bundle(seq_nr) - Used to request to the Survey a lost bundle with the sequence number specified.
- **Type 15**: send_bandwidth_estimator() - Used to send an estimation parameter of the bandwidth connection between Central Station Unit and the Data Mule to the Survey.

3.4.2 UDMP-S Diagrams

The control messages that were specified in the last subsection are sent by an out-of-band acoustic link in our solution implementation. In order to better understand the operation of the new protocol adaptation we will go through a series of diagrams, showing how every type of node behaves when running it. First, we will take a look to the sequence diagram displayed in Figure 3.2.

We have considered a situation where a data mule failure occurs in one of the Data Mule Units and a set of video chunks are lost. If this situation occurs, the Central Station waits a specified time interval - $T_{timeout}$ - and if the Data Mule does not arrive after that period time elapsed, it sends a request to recover that mule or to send a new one to the survey.

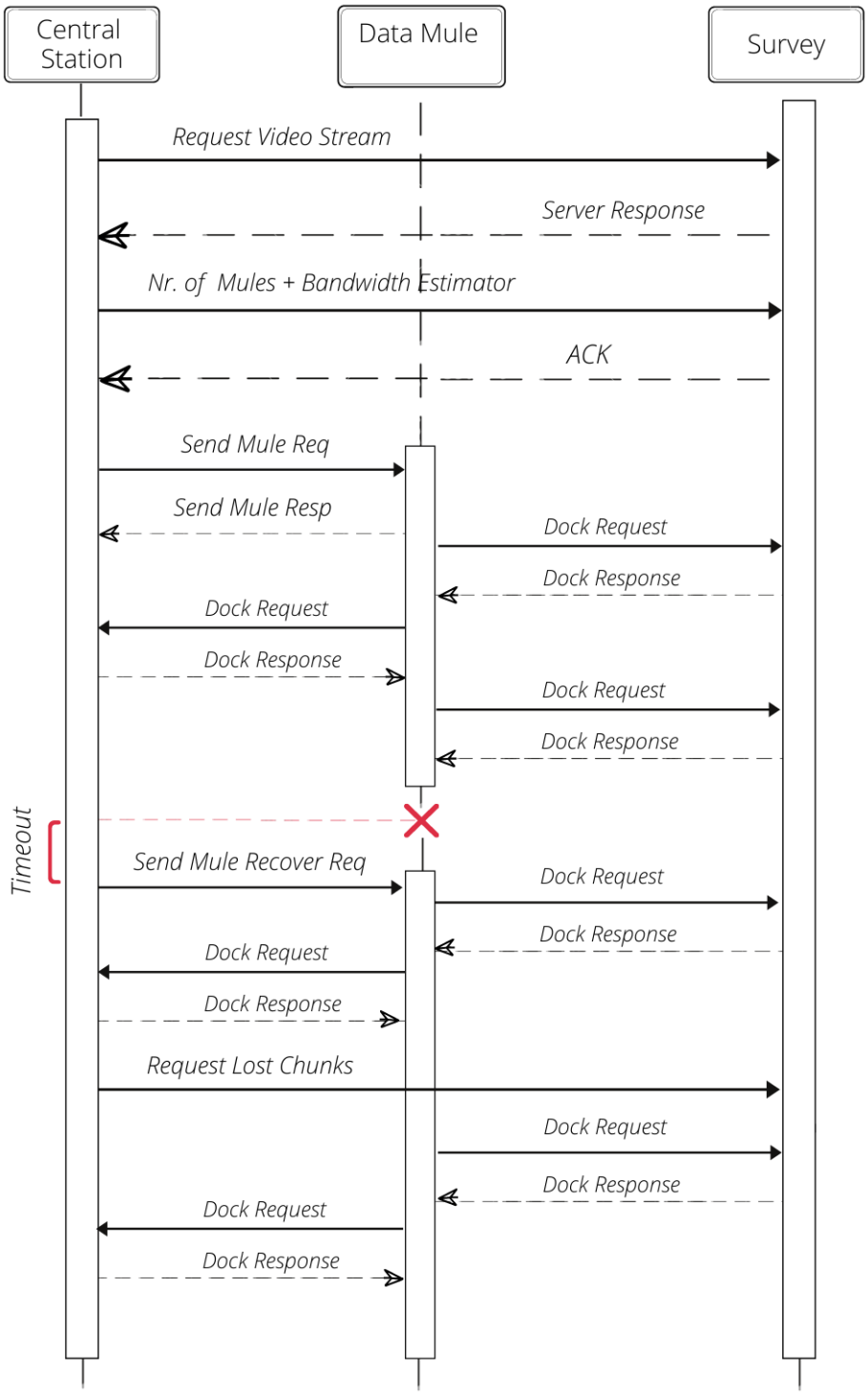


Figure 3.2: UDMP-S Sequence Diagram

The video streaming session starts when the user at the Central Station Unit requests a new video stream. The server answers affirmatively if it is available to perform video streaming or negatively if it is busy. After the affirmative response, the Central Station sends the number of data mules, as it was part of original UDMP, and together with that information it also sends a new message containing the bandwidth estimator to regulate the adaptive streaming mechanism we designed at the Survey Unit. The mule travels between the station and the survey transporting the video chunks, being Docking and Undocking requests exchanged between the mule and the other two nodes. Considering that at a certain point, the Data Mule failed to respond in a predefined time interval ($T_{timeout}$) the system assumes that it got lost somewhere in the underwater. After that, a request is sent to the mule in order to make it return to the Survey Unit. After completing the new travel to the Central Station, the receiver notices that some bundles were lost and sends a message request to the Survey Unit, asking for those chunks. In the next travel, the Survey reliable connection script sends the lost bundles to the mule, that delivers them to the Central Station.

We decided to give an example where it is the same Data Mule Unit that will gather the data that was lost, but it could have been sent a second Data Mule after the detection of losing the first. In that case, there was no need to sent a request for lost bundles because when the second mule reaches the survey, it will bring every chunk that was generated until there, making those bundles arrive in time to be displayed in the best-effort streaming connection.

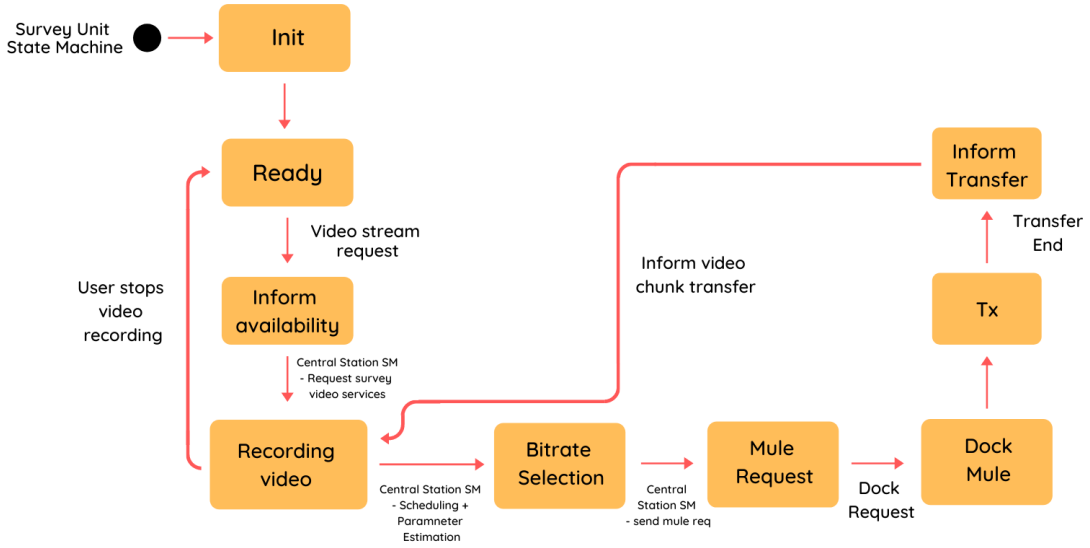


Figure 3.3: Survey Unit State Machine

We will now move on to the State Machine diagrams. Similarly to regular UDMP, the UDMP-S version has three different state machines, one for every type of node considered in our network.

The state machine related to the Survey Unit is in Figure 3.3. After initializing the several processes needed to perform a new video streaming session, it begins a waiting phase until a new request from the Central Station Unit arrives. After that, it informs the Central Station of its availability status and goes to the recording or processing of local video data. After receiving

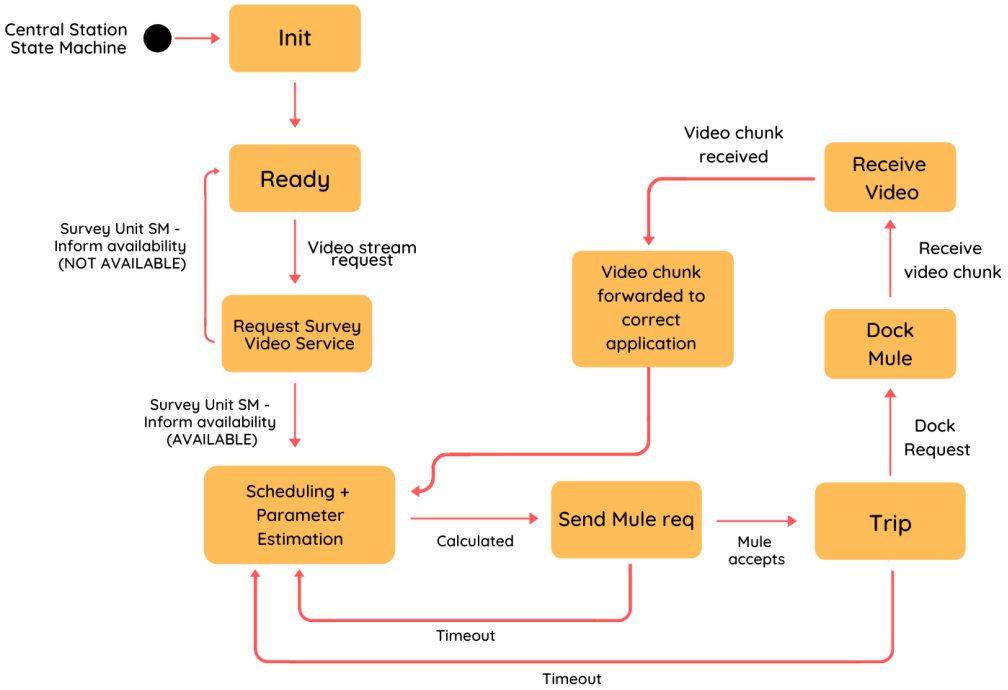


Figure 3.4: Central Station Unit State Machine

the scheduling parameters and the bandwidth estimator from the Central Station Unit, it selects the most suitable bitrate stream to send and waits for a new mule dock request. Following the mule docking, video transfer and undocking, the survey state machine returns the state where it is recording or processing video data to send, until a new value for the estimator arrives.

In Figure 3.4 we can see the state machine for the Central Station Unit. It starts by waiting for a new user request followed by a request for a new video streaming session to the Survey Unit. After the affirmative response of the survey, it goes to a state where the bandwidth is estimated and the mules scheduled, sending these parameters to the Survey Unit. After sending the Data Mule to the survey, it waits for its arrival with new video data. When the mule arrives, it handles the docking request and then forwards the received bundles to the correct application of IBR-DTN framework (some go to the close to real-time video streaming, the others go to the reliable connection to replay later). Next, it returns to the state where the parameters are estimated to repeat the process described.

Finally, we have the state machine of the Data Mule Units represented in Figure 3.5. The reasoning behind this state machine is very straightforward. First, the mule initializes the processes needed to perform the communications and process the DTN bundles forwarding. Then, it keeps waiting until the Central Station Unit sends a request asking to travel to the Survey Unit to gather the video stream chunks. The Data Mule goes then to the survey, gets the video data after the docking and undocking process, and returns to the Station Unit to deliver the video stream bundles after a new docking/undocking procedure. After this, the Data Mule returns to the state where it waits for new instructions from the central station, repeating the process.

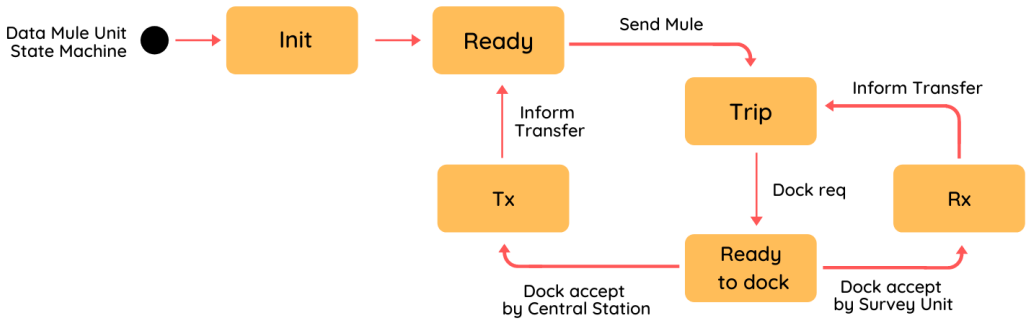
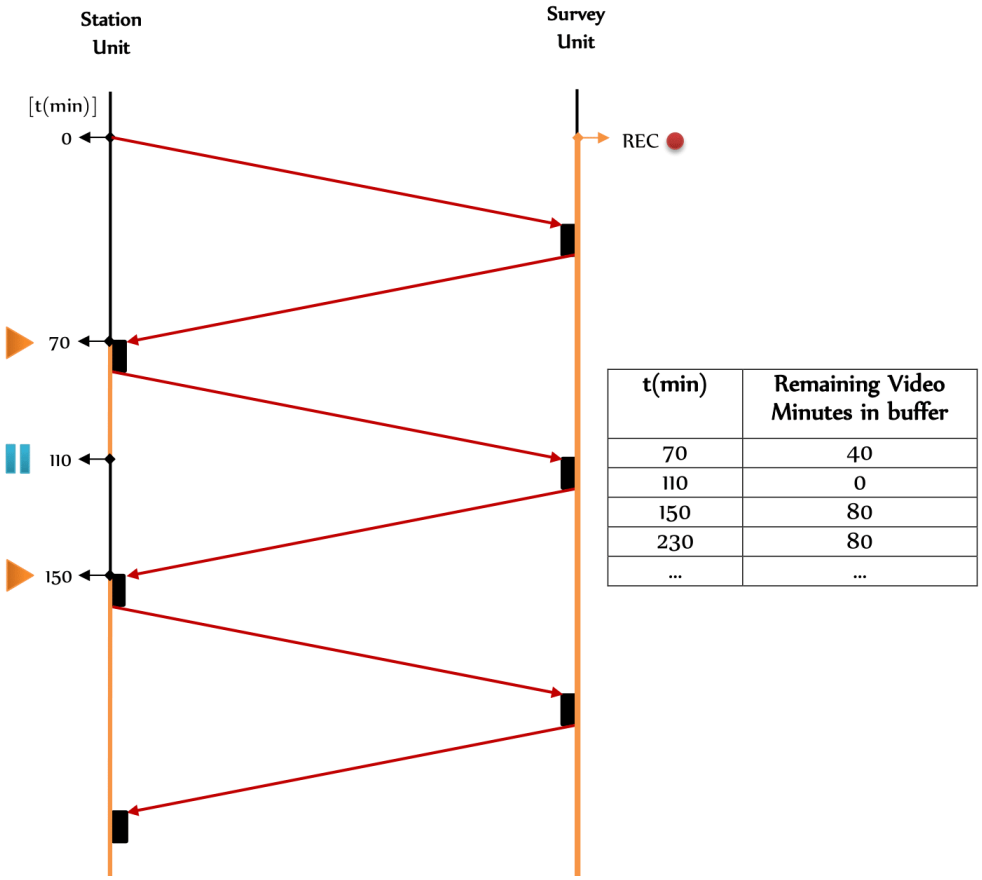


Figure 3.5: Data Mule Unit State Machine

3.5 Delay and Time Diagrams

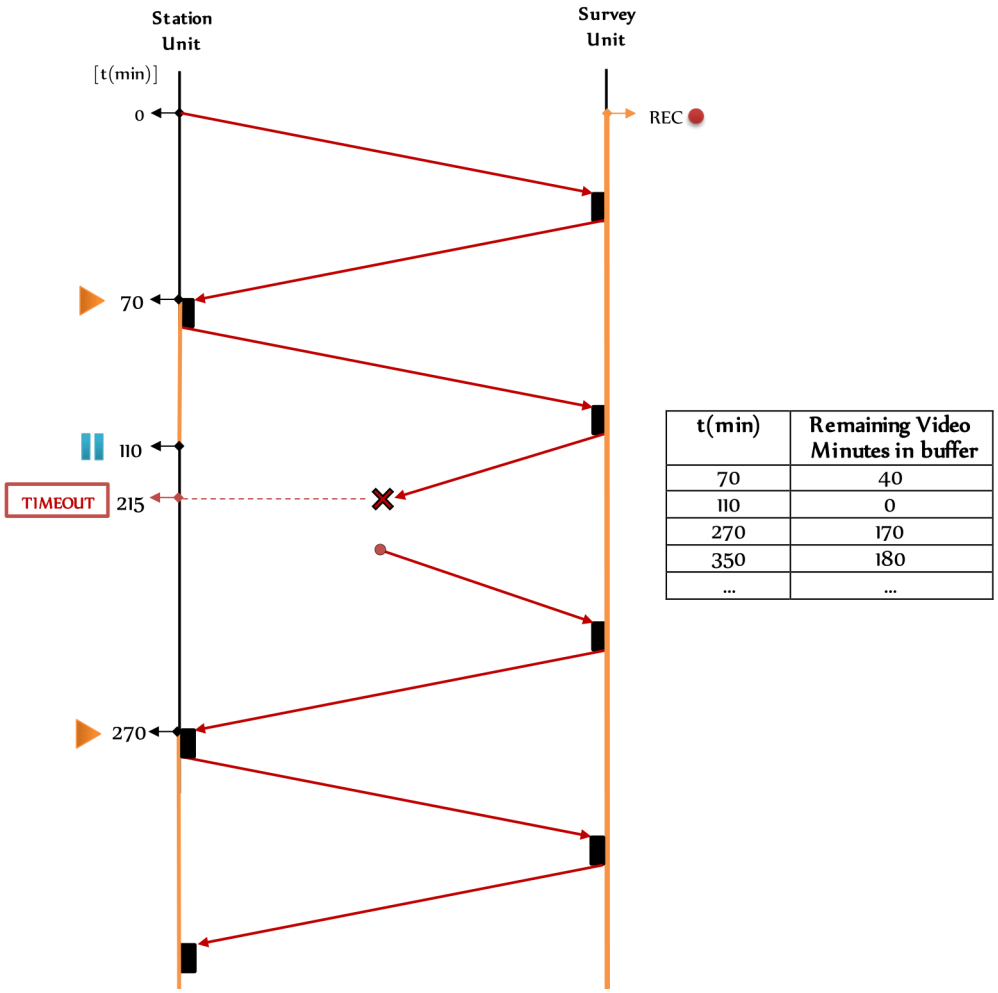
A good way to understand how our proposed solution works is through sequence diagrams. These diagrams display the data mule units traveling in both directions between the Central Station and the Survey, allowing to know the size of every video chunk transported and to calculate maximum stall time or delays, for example. The theoretical assumptions being presented here serve as a start and a comparison point to evaluate the experimental results, later in this document. In Figure 3.6 we can see a Time Diagram for a very simple scenario where we are using just one data mule, remembering that we are considering the Survey Unit to be in a fixed position. We have also defined the data mule travel time to be fixed (30 minutes) and the transfer time to be also constant (10 minutes). With this diagram, we can conclude that, in this case, the user has to wait 70 minutes after he first made the request to see the first images, which is the time spent by the data mule to travel from Station Unit to the Survey Unit and then back to the station, and that the video streaming will be stalled for 40 minutes before the mule reaches the Central Station unit for a second time. The only way to reduce the first byte time is to start the Data Mule Unit travel prior to the start of the recording. Nonetheless, after the mule arrives for the second time at the Station Unit, the streaming will not experience any more problems and the video will continue to be displayed continuously at the Station unit.

Now, considering again the same simple scenario, where the survey node still motionless in a steady position, and using the same constant values for data mule travel time and transfer time, we can study a case where the data mule gets lost, similar to the one presented in the message sequence diagram of the Figure 3.2. Considering that when the message sent to the data mule is received, it is halfway from the Survey Unit, the next video chunk will be received at t = 270 min. Similar to the last studied case after the mule arrives for the second time at the Station Unit, the streaming will not experience any more problems and the video will continue to be displayed continuously at the Station unit, although it has taken almost 3 complete travels in contrast with the 2 of the previous situation.



t(min)	Remaining Video Minutes in buffer
70	40
110	0
150	80
230	80
...	...

Figure 3.6: Sequence diagram using one Data Mule and a stationary Survey Unit



t(min)	Remaining Video Minutes in buffer
70	40
110	0
270	170
350	180
...	...

Figure 3.7: Sequence diagram using one Data Mule and considering a Data Mule failure

Chapter 4

Implementation and Experimental Planning

In this chapter we will undertake the details regarding the implementation of the video streaming solution. The focus is on the presentation of the aspects and design of the testbed, followed by the step-by-step implementation in hardware and software of our proposed system.

4.1 Testbed Design

In order to obtain an experimental validation of our solution, a testbed was designed and built. A representation diagram of the testbed is depicted in Figure 4.1.

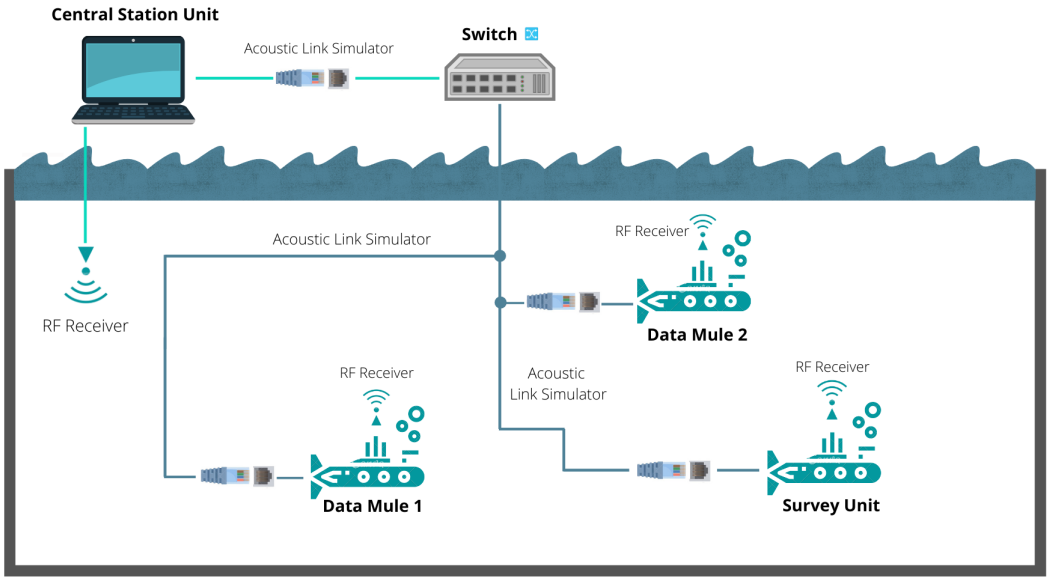


Figure 4.1: Proposed Testbed

According to the testbed diagram, we have emulated four underwater (DTN) nodes: one representing the Central Station Unit, other representing the Survey Unit and two representing Data Mule Units that transport the video data. The first of the Data Mule Units guarantees the best-effort connection and the second is reserved for the reliable mechanism, i.e. for retransmission of the lost data due to excessive data mule delay, or data mule malfunction. The short-range communications for video streaming purposes were based on RF solutions and the control messages were exchanged with a simulated out-of-band acoustic link, in this case using an Ethernet connection.

The four nodes were equipped with Wi-Fi cards to implement the short-range RF communications and were connected to a network packet switch by Ethernet cable to allow the emulation of the out-of-band acoustic link. Every node ran a modified version of IBR-DTN framework that we developed in order to meet the needs of our solution. Such modifications will be explained and justified later in this chapter. A picture of the testbed deployed at the FEUP lab is shown in Figure 4.2.

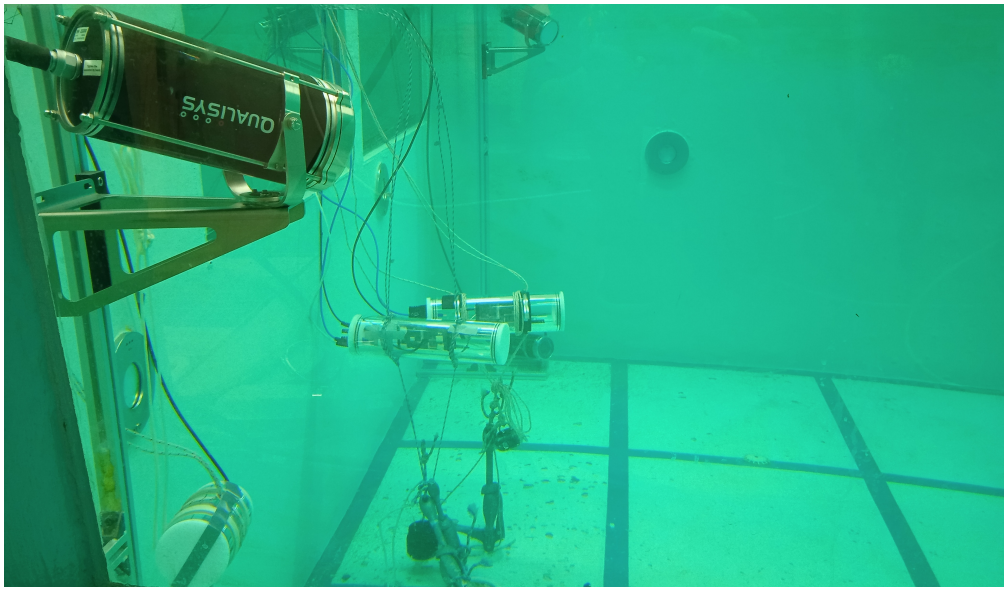


Figure 4.2: Testbed implemented at FEUP freshwater tank.

4.2 Hardware and Software Specifications

With the testbed schematic illustrated in Figure 4.1, we must now identify the hardware and software specifications for each element represented.

The Central Station Unit, i.e. our receiver node, was simulated using a PC running one of the regular Linux distributions - Ubuntu - and placed near the water tank. The freshwater tank has the following dimensions: length of 460 cm, width of 440 cm, and depth of 170 cm. We also used an external IEEE 802.11n 2.4 GHz Wi-Fi network interface card to perform the RF communications. This machine had a custom version of IBR-DTN installed in order to communicate with the other DTN nodes, as well as the VLC Player for the display of the received video streaming.

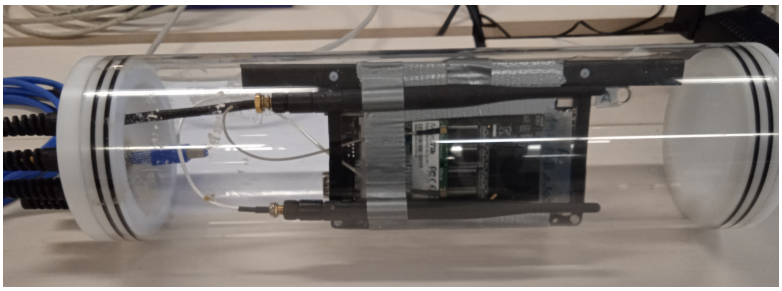


Figure 4.3: One of the nodes involved in an airtight PVC cylinder

Furthermore, we used three ALIX system boards [50] with the same IBR-DTN implementation as the Station installed, running over Debian 8 and using a standard 2.4 GHz Wi-Fi module. Two of these devices served as Data Mule units and the third as the Survey unit. All these hardware was involved in airtight PVC cylinders as we can see in Figure 4.3. We have defined DTN routes in the IBR-DTN daemon configuration file to make the Data Mules as intermediary nodes between the previous the Central Station Unit and the Survey Unit.

As mentioned before, the acoustic link for control messages exchange purposes was emulated by an Ethernet cable connection. All nodes were connected to an ethernet switch at the surface, connecting them to the same local network. To implement both the best-effort virtual connection for regular video streaming, and the reliable virtual connection for lost bundles recovery, we had to develop several changes in IBR-DTN source code. A summary of our system software architecture is depicted in Figure 4.4.

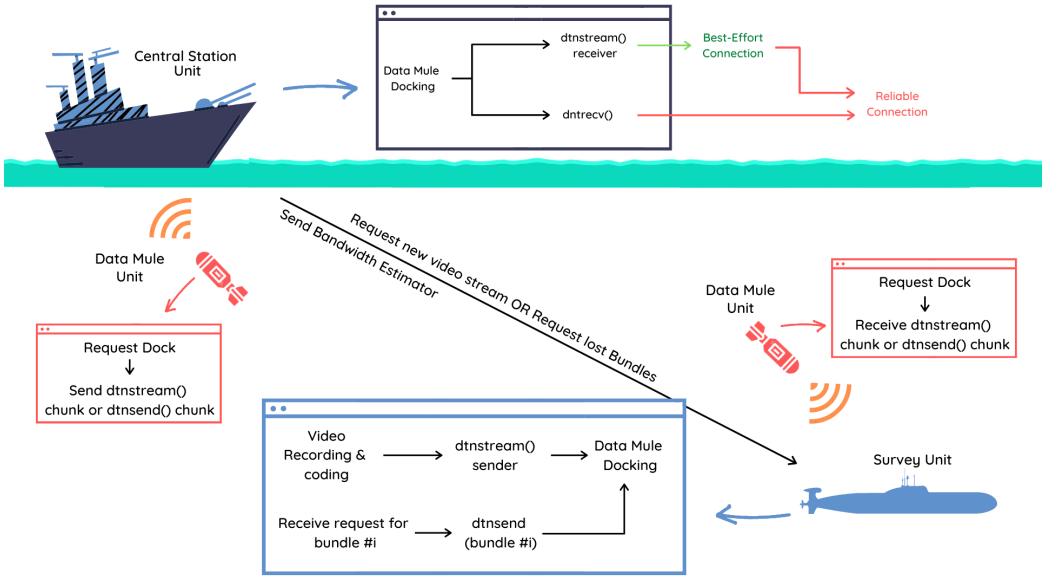


Figure 4.4: Proposed System architecture resume

For the regular video streaming implementation, we decided to use the *dtnstream* tool in both the Central Station Unit acting as the receiver, and in the Survey Unit acting as the emitter in a best-effort approach where the central Station receives and reproduces the video data that is

brought by the Data Mule Units into the VLC Player. To order the bundles when they arrive at their final destination and identify when a bundle was lost, the *dtnstream* tool checks a parameter called Streaming Sequence Number. The expected operation was that the *dtnstream* receiver would order the bundles with increasing Streaming Sequence Number and ignore a certain bundle or set of bundles if new bundle arrives with higher sequence number than expected or if a prespecified timeout elapses. Although this was the expected behaviour in IBR-DTN documentation for *dtnstream* tool, we have realized this was not the case and the application always kept waiting for the next consecutive sequence number, even when bundles with higher sequence numbers arrived. After several tests and some analysis of the source code, we have detected a small logic error in one of the methods concerning the handling of the bundles that arrive out of order at the receiver. We have corrected this problem and reported it to one of the IBR-DTN developers, being this documented in Issue #283 in the IBR-DTN repository main page [51].

Furthermore, after handling with these problems respecting the best-effort streaming connection, we moved to the implementation of the reliable retransmission mechanism. As mentioned in previous sections, the objective of this new reliable virtual connection was to give the user the option to locally replay the complete video content later, containing not only the bundles received by the best-effort communication link, but also the ones that were lost. To implement such approach, we started by changing, once again, the source code of *dtnstream* tool to store the video data that arrives in a local directory instead of flushing it after reproduction and also to write the sequence numbers of the bundles in three different files, looking upon their classification in arriving time terms:

- The bundles that arrive on time to be reproduced have their Streaming Sequence Numbers written in a file called *success.txt*;
- The bundles that arrive too late to be reproduced but are received by the best-effort connection (and so, they are stored locally) have their Streaming Sequence Numbers written in a file called *tooLate.txt*;
- The bundles that never arrive in the best-effort connection have their Streaming Sequence Numbers written in a file called *failed.txt*.

We have forced every bundle to be saved as an individual file with the respective Streaming Sequence Number as the file name. After that, we needed a way to recover the bundles that never arrived in the best-effort connection. To do this, we have implemented the reliable mechanism we have specified in section 3.2. To achieve that, we have developed two scripts in Python, one to be run at the Central Station Unit and the other to be run at the Survey Unit.

Additionally, we modified the source code of IBR-DTN *dtnrecv* tool in order to make it to write the bundles that are received into individual files, instead of concatenate them into a single file. The script at the Central Station Unit reads the file that contains the sequence numbers of the bundles that have been lost (*failed.txt*) and using the out-of-band acoustic link it sends an UDMP-S message request to the Survey Unit (defined in Section 3.4.1) requesting the bundles

with the sequence numbers specified. The lost bundles are then gathered by the next Data Mule that approaches the Survey Unit, being the new information for the best-effort part also transported by this mule. At the Central Station, a session of IBR-DTN *dtnrecv* is kept always on, in order to receive the bundles that will be retransmitted and transported by the Data Mules. On the other hand, the script on the Survey Unit side waits for the request messages and sends the bundle files that have the sequence numbers requested through the IBR-DTN application *dtnsend*. Since the *dtnsend+dtnrecv* use does not consider the sending of the Streaming Sequence Number within the bundle as it is in *dtnstream*, we had to find a way to know the corresponding sequence number of every bundle received at the Central Station through the reliable connection. To do this, the Python 3 script that runs at the Survey Unit generates a md5sum hash for every bundle generated by *dtnstream* and stores it together with the correspondent sequence number in a shared file with the Station Unit, using the out-of-band acoustic link (here emulated by Ethernet cable). After that, the Central Station Python script also generates a md5sum hash for every bundle received by the retransmission mechanism and reads this shared file created by the Survey, obtaining the correspondent Streaming Sequence Numbers for all bundles, changing their names to the sequence numbers, as *dtnstream* does for the bundles that it receives. After all bundles that were generated by the sender since the request for new video stream session until the user closes the session are received, the individual bundle files are concatenated into a single file which can be reproduced as a video content.

4.3 Network Architecture and Configuration

As stated before, the implementation of the connection between nodes was implemented using RF communications for the short-range link and an Ethernet cable connection for simulation of the out-of-band acoustic link. The RF connection consisted in a IEEE 802.11n Network, using the IP addressing scheme 10.0.0.X with a netmask of 255.255.255.0. The Ethernet Network was obtained connecting every node to a switch at the surface, using the IP addressing scheme 192.168.0.1XX with a netmask of 255.255.255.0. The IP addresses for every machine in both networks are depicted in Figure 4.5.

To control the allowed input addresses at every node allowing a better management of the networks being used we implemented several scripts containing *iptables* commands. This tool is a command line interface application for Linux systems that allow to open or block network connections regarding a determined interface or a set of addresses.

Every node represented is considered a DTN node, since we used a DTN framework to perform the communications (IBR-DTN) and because of that, every one of them has a DTN node identification in the form of *dtn://"hostname"*. The correspondent DTN addresses for every node are shown in Table 4.1. The address routes to guarantee that all DTN bundles that are exchanged between the Central Station and the Survey are forwarded through the Data Mules are defined at DTN level in the IBR-DTN daemon configuration file.

Table 4.1: DTN nodes Identification

Node	DTN Identification
Central Station Unit	dtn://stationUnit
Survey Unit	dtn://surveyUnit
Data Mule #1	dtn://dataMule1
Data Mule #2	dtn://dataMule2

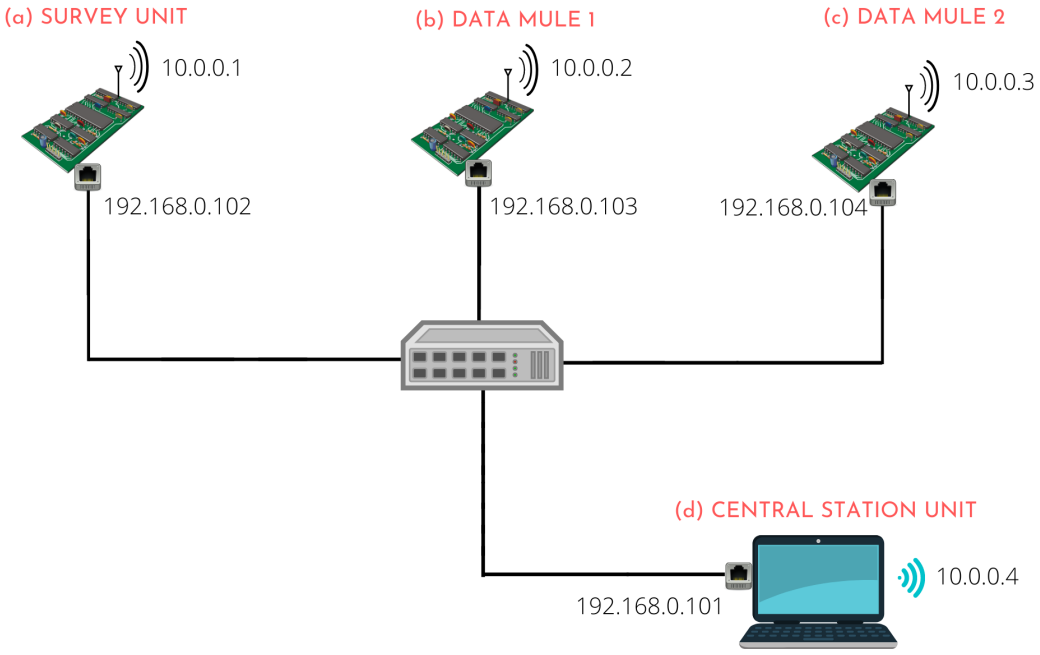


Figure 4.5: Network structure designed for the experimental tests

Due to the limited amount of time to perform the lab tests due to COVID-19 pandemic, the adaptive video streaming mechanism was not tested in lab. However, it is implemented through the usage of UDMP-S messages. In short, this mechanism was made to be executed in two nodes, the Central Station and the Survey Unit, and it has the following implementation: first, the video stream is coded in two different bitrates, one with Standard Definition (SD) and other with High Definition (HD), at the Survey Unit. Then, there is a script, also at the survey, that pipes one of these streams into *dtnstream* sender. To choose which stream is being piped to the sender application, receiver i.e. the Central Station Unit has to send a message to the survey indicating a bandwidth estimator for the short-range link between the Data Mule and the Central Station using *iperf* capabilities (this message was defined in UDMP-S, in the previous chapter). Finally, the survey receives this parameter estimator and compares it to a pre-defined value and chooses the suitable stream to pipe to *dtnstream* the sender application.

Chapter 5

Experimental Evaluation Results

In this chapter, we present and evaluate the experimental results that were obtained during the laboratory tests. We explain how each test was performed, how the results were measured and discuss the major outcomes for every parameter.

5.1 Evaluation Metrics

To evaluate the performance and robustness of our underwater video streaming system, we have considered different evaluation metrics:

- Average short-range throughput;
- Performance of the new UDMP-S;
- Initial delay to receive the first video chunk;
- Average recovery time (average time elapsed between the moment we lost a data mule and the moment where the normal operation of the system is recovered).

Also, as our solution is an adaptive bitrate video streaming system with a "replay later" option, we have studied some QoS and QoE evaluation metrics that are associated with these kind of services, including:

- Percentage of total time waiting for new video chunks;
- Number of Data Mule travels needed to prevent buffer underrun;
- Maximum stall time, i.e., the maximum amount of time in which the video image stays "frozen";
- Average time to recover all lost bundles in the reliable connection.

The values collected concerning these metrics during the laboratory experiments are evaluated in the next sections. Also, we explain the procedures and how we achieved such practical results.

5.2 Average Short-Range Link Throughput and QoE/QoS Considerations

During the experimental tests execution, a set of parameters were considered fixed, such as the Data Mule *Docking Time* and *Undocking time*. Concerning the Travel Time, i.e., the time it takes for the mules to travel between the Survey Unit and the Central Station Unit and vice-versa, we have considered an approximated value driven from the average of the travel times of our experiments. If we consider our Data Mule Units to have a speed of 1m/s, they would have traveled approximately 160m between nodes, in our targeted maritime scenario. Of course, we could have simulated a bigger travel time that would consequently correspond to a bigger distance, but for validation purposes, the values we considered are sufficient. In the case of the *Average Short-Range Link Throughput*, a practical speed test was performed, using the *iperf* tool with actual measurements from our testbed. The resume for these values can be seen in Table 5.1.

Table 5.1: Testbed Parameters.

Parameter	**Value**
Docking Time	5 s
Undocking Time	5 s
Average Travel Time	2min40s
Average Short-Range Link Throughput	16,55 Mbit/s

Table 5.2: RF Short Link *iperf* results.

Test #	**Average Throughput between underwater nodes (Mbit/s)**
1	16,5
2	16,4
3	16,0
4	16,2
5	16,5
6	16,8
7	16,7
8	17,0
9	16,6
10	16,7
Average	16,55
σ	0,29

The value for the *Average Short-Range Link Throughput* was calculated using measurements of *iperf* [52] which is a network measurement tool that allows to perform speed tests between nodes using TCP/UDP injection packets. The measurements consisted in a set of 10 transfer speed test repetitions between two of the underwater nodes, being the values depicted in Table 5.2. After this benchmark test, the average value and the standard deviation of these results were calculated.

We have already mentioned previously that the consideration of QoE and QoS metrics are very important when we evaluate the performance of video content based systems. In order to have a starting point relative to an existing solution on the market to compare with the results that will be presented in the next sections, a short description of a hypothetical acoustic communications solution will be provided subsequently.

Let's then consider that this hypothetical acoustic solution meets the average acoustic modem transfer speed, which as we have seen in Chapter 2 rounds the 30Kbit/s, and that the communication is directly maintained between Central Station Unit and Survey Unit. Considering that the video chunks being transferred have 500 kbits corresponding to approximately 3 seconds of video in the stream we used in our solution tests, and that the video is only displayed when the complete chunk is received, we can evaluate the behaviour in questions of delay using the Figure 5.1.

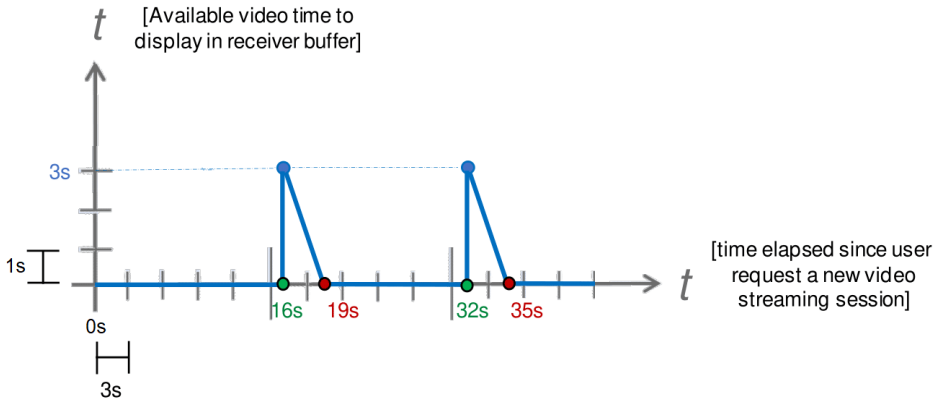


Figure 5.1: Buffer time available in function of time
(Hypothetical Acoustic Solution)

Using these considerations we will discuss the improvements that our solution offers when compared to the current systems like this defined acoustic communications.

5.3 One Data Mule Unit Operation without Failure

This experimental test consisted in using a PC as Central Station Unit at surface and submerging two nodes involved in airtight PVC cylinders in the freshwater tank, one to work as Survey Unit and other as a Data Mule Unit, simulating several successful mule travels between the sender and receiver nodes. The objective was to test the viability of our solution in a real underwater

scenario and to measure several parameters such as delays and maximum video stall time in order to evaluate the performance for this situation. In the delay / time diagram in Figure 5.2 and in the Table 5.3 we can see the important delays and timing results measured during the experiment.

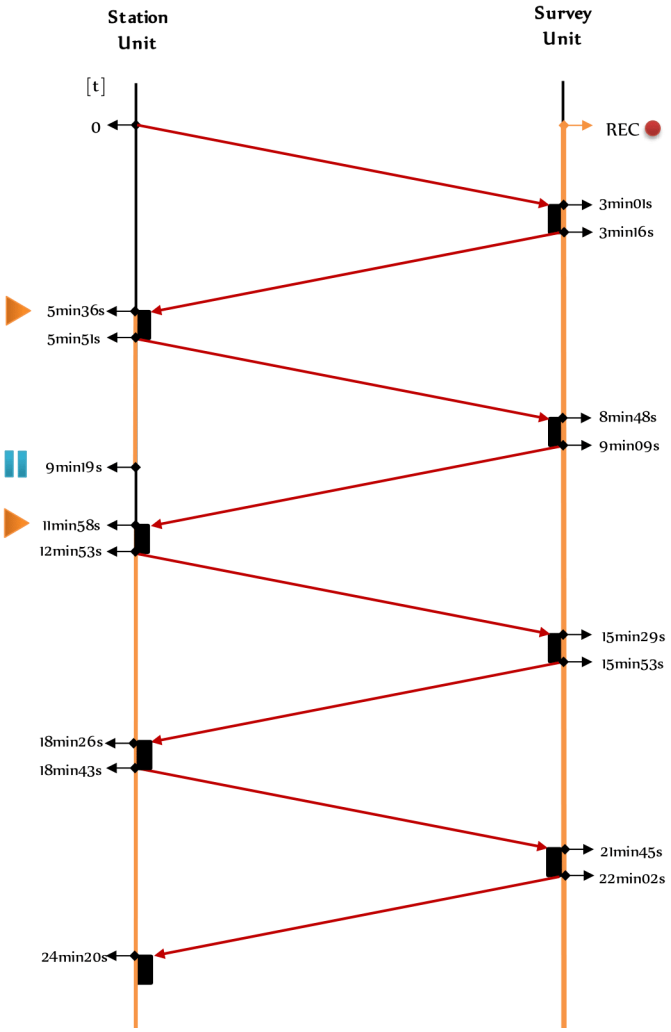


Figure 5.2: Delay / Time Diagram with experimental timing results (Exp.#1)

Using the values in the Table 5.3 we were able to plot the graph in Figure 5.3. This enables us to better perform an analysis of our experimental results both in performance of the system and QoE/QoS parameters. As we can see, the time elapsed to receive the first chunk of video by the user and display it on VLC is equal to the time that takes the Data Mule to travel from the Central Station to the Survey Unit, to upload the video data being streamed, and then to travel again to the Central Station Unit to deliver the new chunks, which in this case was:

$$t_{recFirstChunk} = t_{travelToSurvey} + t_{uploadStream} + t_{travelToCentralStation} = 5min36s \quad (5.1)$$

Also, we can easily get the value for the maximum stall time ($t_{Stall_{max}}$). In this case, $t_{Stall_{max}}$ is the interval between the moment where the part of the video streaming that was delivered in the

Table 5.3: Experimental timing results (Exp.#1)

t	**Remaining video time in receiver node buffer**
0min0s	0min0s
Data Mule Unit arrives bringing +3min43s	
5min36s	3min43s
9min19s	0min0s
Data Mule Unit arrives bringing +6min40s	
11min58s	6min40s
*18min25s**	0min13s
Data Mule Unit arrives bringing +6min03s	
18min26s	6min15s
*24min19s**	0min22s
Data Mule Unit arrives bringing +6min03s	
24min20s	6min24s

* Refers to the immediate moment before the new chunks are transferred by the data mule that arrived

first Data Mule arrival to the Central Station ended after displaying, and the moment that the Data Mule arrives again at the Central Station bringing new video data:

$$t_{Stall\,max} = 18min26s - 9min19s = 9min07s \tag{5.2}$$

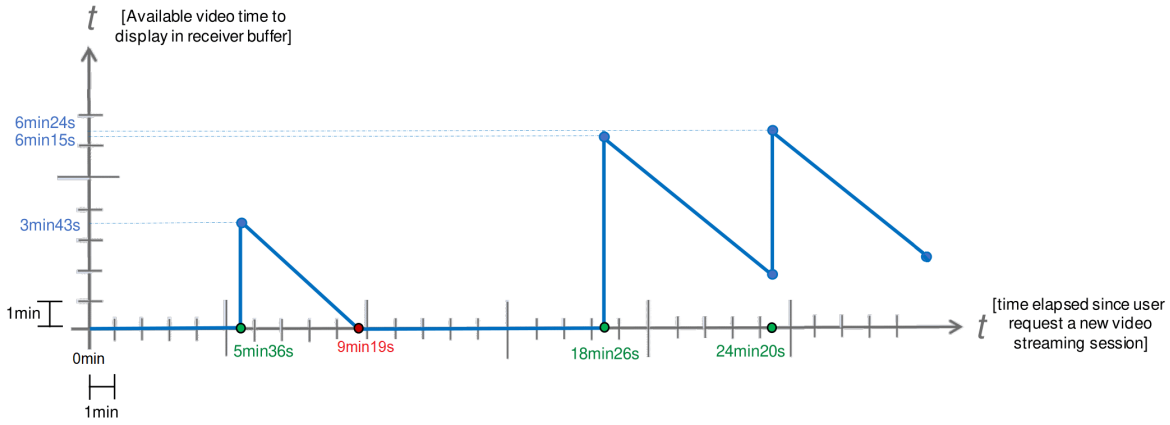


Figure 5.3: Available video time in receiver buffer
(Situation with one mule and no fails)

Finally, we can verify that after two Data Mule Unit complete travels, that can be translated to approximately 18 minutes and 26 seconds, the video streaming service became stabilized. In other words, after that moment we got continuous video streaming and the user no longer experienced video stalling or "frozen" video image. This was the expected result that we predicted in our theoretical assumptions, presented in Chapter 3.

5.4 One Data Mule Unit Operation with Failure

In the second experiment, we followed the same reasoning of the previous test, but this time we have simulated a failure in the Data Mule Unit, causing bundles to get lost and force our reliable connection solution to retransmit them again. The Data Mule Unit, after it was reached by the out-of-band link, returned to the Survey Unit bringing the new bundles and the ones that were lost. In this case, at protocol and software level there was a need to request the lost bundles to the Survey Unit because since *dtnstream* already sent these packets to the mule, it has to be our reliable connection to ensure the retransmission of such lost chunks. The dela y /time diagram for this experimental situation is shown in Figure 5.4 and in Table 5.4 we can see the delay and timing results measured during this second experiment.

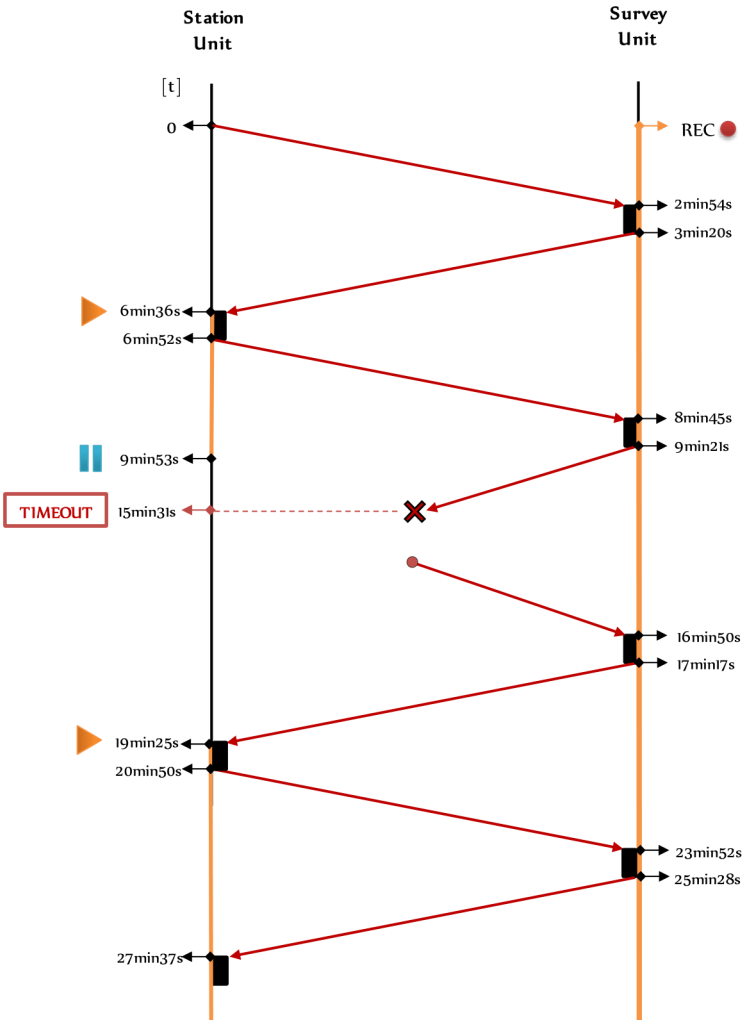


Figure 5.4: Delay/Time Diagram with experimental timing results (Exp.#2)

It is also relevant to mention that we defined out timeout to assume that the Data Mule was lost ($T_{timeout}$) to be 150% of the time it will normally take to the mule to return to the Central Station Unit for every experiment that evolves mule failure. $T_timeout$ can be calculated using Eq. 5.3.

Table 5.4: Resume of experimental timing results (Exp.#2)

t	**Remaining video time in receiver node buffer**
0min0s	0min0s
Data Mule Unit arrives bringing +3min17s	
6min36	3min17s
9min53s	0min0s
Data Mule Unit arrives bringing 13min43s	
19min25s	13min43s
*27min36s**	5min32s
Data Mule Unit arrives bringing +9min17s	
27min37s	14min49s

* Refers to the immediate moment before the new chunks are transferred by the data mule that arrived

$$T_{timeout} = 1.5 \times (t_{travelToSurvey} + t_{uploadStream} + t_{travelToCentralStation}) \tag{5.3}$$

As we may see in Figure 5.5, the time elapsed to receive the first chunk of video by the user and display it on VLC in this case was:

$$t_{recFirstChunk} = t_{travelToSurvey} + t_{uploadStream} + t_{travelToCentralStation} = 6min36s \tag{5.4}$$

Again, we can easily get the value for the maximum stall time:

$$t_{Stall\,max} = 19min25s - 9min53s = 9min32s \tag{5.5}$$

Now, since in this situation we simulated a fail in the Data Mule Unit, we also measured another parameter: $t_{recovery}$ which is the time to recover the system normal functioning after a Data Mule is lost. In this experiment it is:

$$t_{recovery} = 19min25s - 15min31s = 3min54s \tag{5.6}$$

We verified that our developed retransmission mechanism worked as expected, requesting and receiving the lost bundles successfully. In Figure 5.6 we show a preview of the output in the Central Station Unit terminal where our reliable connection script was running, performing requests for the lost bundles. It is important to remember that the Central Station Unit only knows that some bundles were lost because the Data Mule arrives again and brings new video data with Streaming Sequence Numbers higher than the ones expected. This makes the reliable connection script to send requests to the survey through the out-of-band link, being these lost chunks brought by the Data Mule in the next travel. So, in conclusion regarding this matter, the delay to recover the lost bundles can be seen in Eq. 5.7.

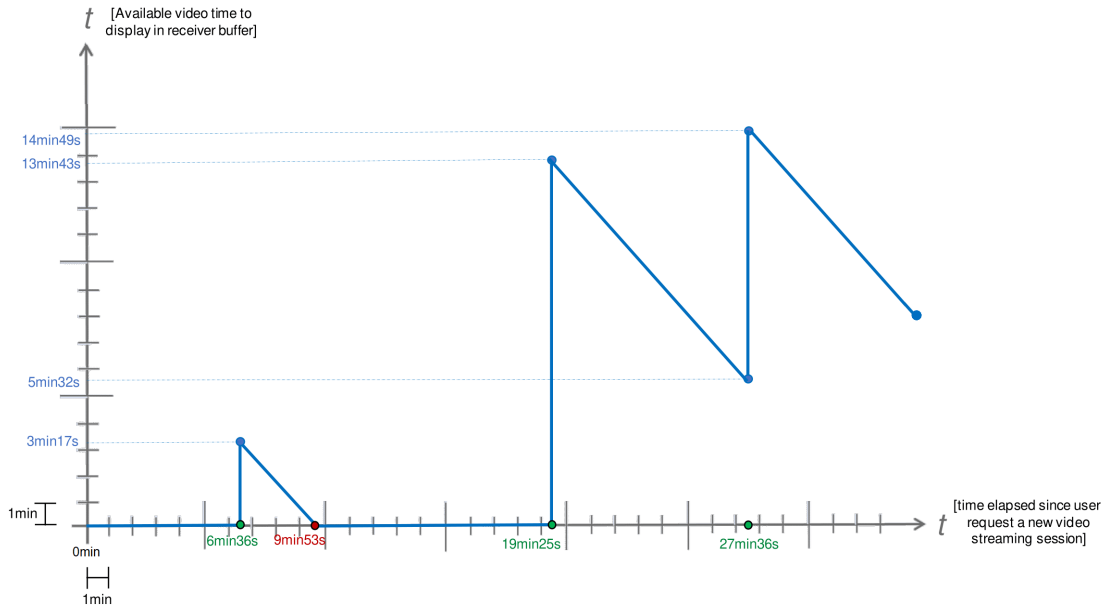


Figure 5.5: Available video time in receiver buffer
(Situation with one mule with failure)

$$t_{recoverLostBundles} = 27min37s - 19min25s = 8min12s \tag{5.7}$$

Concerning the stabilization of the video streaming service, we can see that almost three Data Mule Unit complete travels were needed to achieve such behaviour, being the time interval slightly different from the last experiment being approximately 19 minutes and 25 seconds. This is still the expected result that we predicted in our theoretical assumptions.

5.5 One Data Mule Unit Operation with a Second Data Mule for Retransmission

We will proceed now to the exploration of the third and last experiment. Here, we also followed some of the same reasoning and procedure of the previous tests, but in this case when we simulated a failure in the Data Mule Unit, we had a second mule ready to enter in action and replace the first and lost Data Mule, not only to recover the bundles that got lost but also to get newer streaming data. In this case, at protocol and software levels, there was no need to request the lost bundles to the Survey Unit because since all bundles generated from the beginning are transferred every time a new node (the second mule) reaches Survey Unit, even if they were transmitted previously to other nodes due to IBR-DTN implementation. We have only noticed this behaviour during these experiments and we give some analysis and implications that this caused in our tests in the last section of the current chapter. Similar to the other operation situations, the delay / time diagram

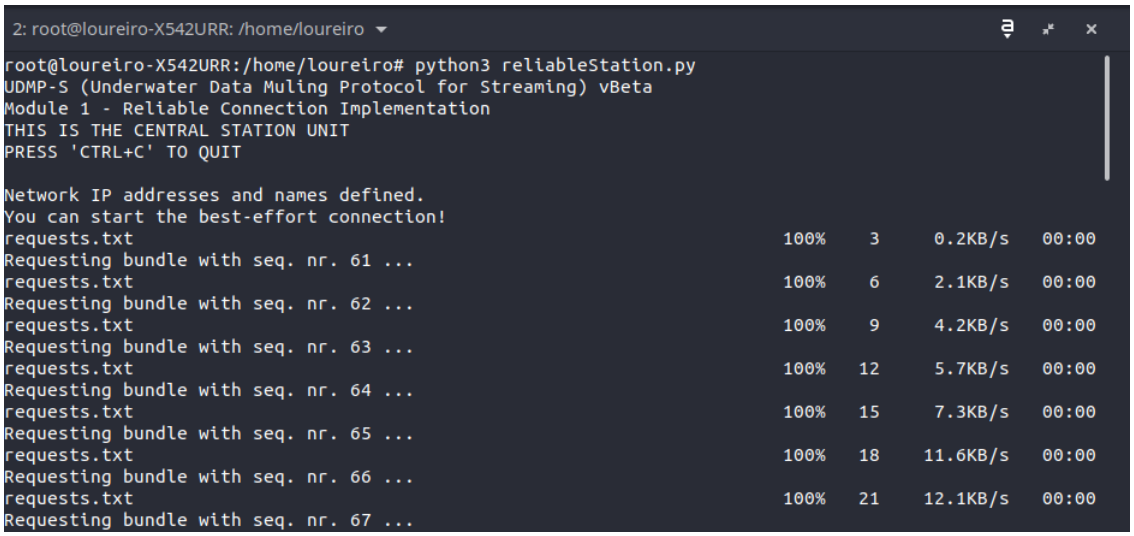


Figure 5.6: Preview of the Output in the Central Station Unit
(Reliable Connection Script Terminal)

for this experimental test is shown in Figure 5.7 and in Table 5.5 we can see the important delays and timing results measured during this last experiment.

Looking to the Figure 5.8, we can verify that the time elapsed to receive the first chunk of video by the user and display it on VLC in this case was:

$$t_{recFirstChunk} = t_{travelToSurvey} + t_{uploadStream} + t_{travelToCentralStation} = 6min00s \tag{5.8}$$

Table 5.5: Experimental timing results (Exp.#3)

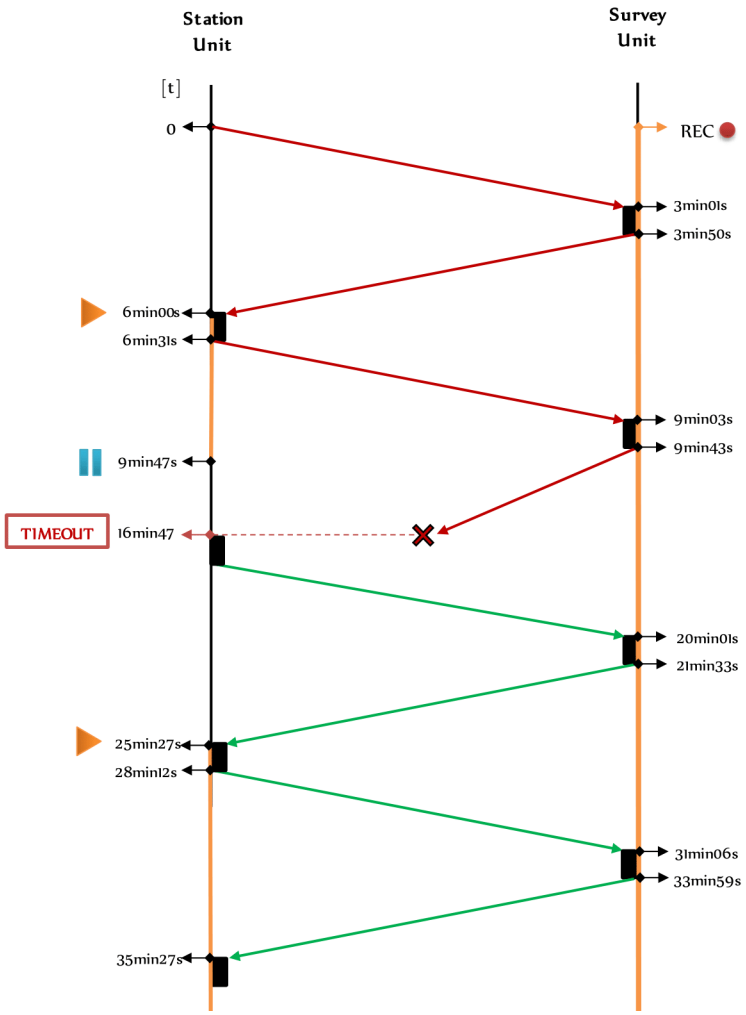
t	**Remaining video time in receiver node buffer**
0min0s	0min0s
Data Mule Unit arrives bringing +3min47s	
6min00s	3min47s
9min47s	0min0s
Data Mule Unit arrives bringing 17min51s	
25min27s	17min51s
*35min26s**	7min52s
Data Mule Unit arrives bringing +11min26s	
35min27s	19min18s

* Refers to the immediate moment before the new chunks are transferred by the data mule that arrived

Once more, we can easily get the value for the maximum stall time:

$$t_{Stall\,max} = 25min27s - 9min47s = 15min40s \tag{5.9}$$

Figure 5.7: Delay/Time Diagram with experimental timing results (Exp.#3)



As this situation has a simulation of a failure in one of the Data Mule Units, we also measured the system recovery time:

$$t_{recovery} = 25min27s - 16min47s = 8min40s \tag{5.10}$$

Concerning the stabilility of the video streaming service, we should consider that two complete travels were needed from the first Data Mule (even though one of the travels was not effective) and one from the second Data Mule Unit, making up to three complete mule travels translated in approximately 25 minutes and 27 seconds to achieve stable continuous video streaming. This also agrees with the expected result that we predicted in our theoretical assumptions for this situation although the delay was considerably bigger. In the next section we will present our reasoning to explain such result as well as for others acquired in the previous experimental situations.

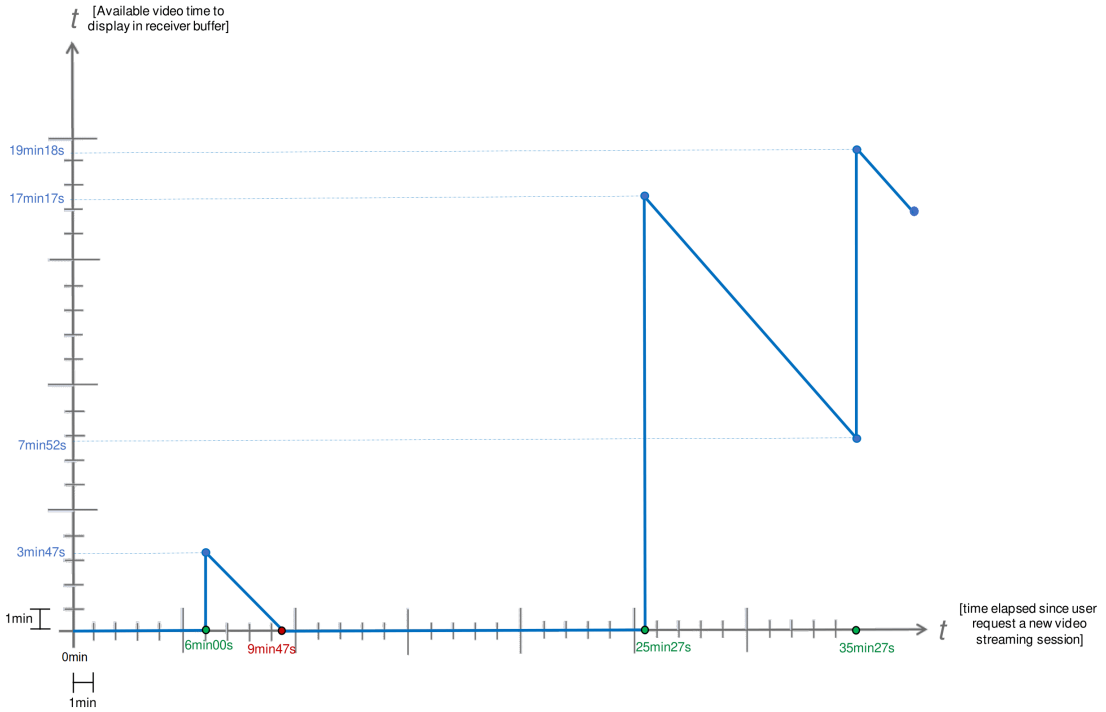


Figure 5.8: Available video time in receiver buffer
(Situation with one mule with failure and another mule for retransmission)

5.6 Discussion

In this section we conclude the analysis on the experimental results obtained and compare against the hypothetical acoustic solution we mentioned in the beginning of this chapter.

First we can say that the delay to receive the first chunk of video using our solution is independent from the factors we changed in every situation, being only dependent on the success, failure or even delay of the Data Mule Unit during the travel between the two nodes. Nonetheless, the average time that the user will wait to see the first chunk of video using the solution developed in this dissertation is given by Eq. 5.11.

$$t_{averageRecFirstChunk} = \frac{t_{recFirstChunk1} + t_{recFirstChunk2} + t_{recFirstChunk3}}{3} \approx 6min04s \tag{5.11}$$

Another parameter that is interesting to our discussion of results is the maximum consecutive stall time of the video streaming i.e. the maximum successive time interval where the receiver experienced "frozen" image, waiting for new data. The results for each of our experiments are summarized in Figure 5.9. As we can see, against the expected result, using our solution with two Data Mule Units, one for best-effort purposes and other for retransmission purposes has a higher experimental value for this parameter. This is basically due to two reasons. The first is that during our last experimental test, the second mule PVC cylinder was damaged and the simulation of the travels between Central Station Unit and Survey Unit took more than in the other cases. The second and most important reason for the evaluation of this solution is that the implementation

of IBR-DTN, as said above, transfers all bundles generated from the beginning of the *dtnstream* every time a new node approaches the Survey Unit, even if they were transmitted previously to other nodes, making the time to upload video to the second Data Mule much bigger. This is one of the upgrades that can be done as future work to improve our system's performance.

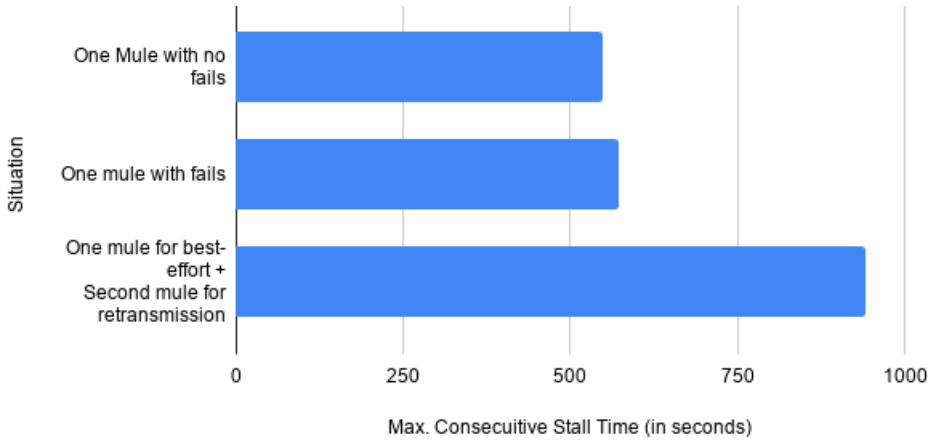


Figure 5.9: Maximum Consecutive Video Stall Time in different situations

To better show how much the solution developed in this dissertation is valuable and better alternative when compared with actual systems, we must look to the results in Figure 5.10. These charts represent the percentage of time where the video stream is effectively being displayed versus the time were the video is stalled. As we can see, even if we consider a mule failure, our solution has a clearly better exploitation of the time elapsed since the user requests a new video stream when compared to the theoretical acoustic system. This values were obtained through Eq. 5.12, using our experimental results in the three situations we tested and theoretical values already explained in previous sections for the hypothetical acoustic situation.

$$\%_{stallTime} = \frac{t_{stallTime}}{t_{totalStreamingSession}} \times 100 \tag{5.12}$$

It should be noted that these experimental results collected in a freshwater tank and conclusions that we have derived from them can also prove our system viability in real underwater scenarios. Of course, the initial delay to receive the first chunk will be much higher in real environments, since this initial delay depends directly of the distance between nodes that will be much bigger in that case. But, for example, the number of data mule complete travels demanded to achieve continuous video stream receiving at Central Station Unit is independent of the distance between nodes, even if we are talking about distances in the order of a few kilometers. This is because, despite the longer time interval spent traveling between receiver and sender, the amount of video being recorded at sender side will be also much higher, making the Data Mule to bring more minutes of video content to the buffer in the receiver side. Besides, the percentage of operation time where there is effective video streaming it will be very similar to the values obtained in our

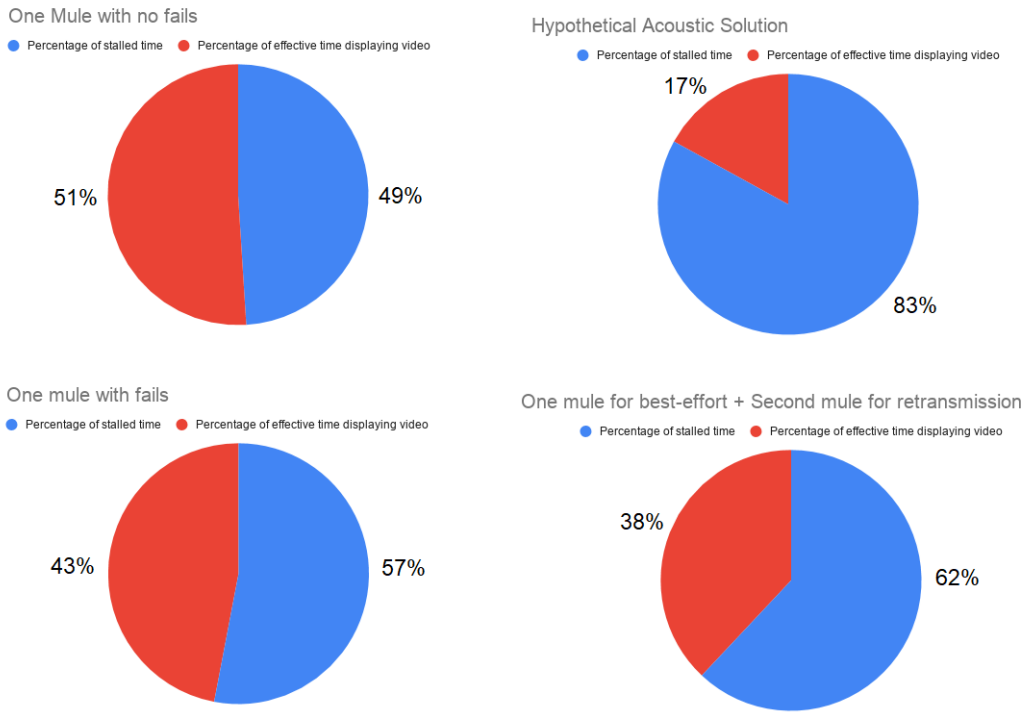


Figure 5.10: Percentage of Video Stall Time Vs Percentage of Effective Video Displaying Time

testbed. In short, our solution is capable of delivering video streaming in order of the hundreds of Kbit/s to few Mbit/s at a distance in the order of the km.

After evaluating the results we obtained and comparing them to existing systems for video streaming in underwater scenarios, we can conclude that the solution developed in this dissertation is certainly an evolution in this area, creating a new viable alternative with better QoE and QoS performance results than the average conventional solutions.

Chapter 6

Conclusions and Future Work

High definition video capture in underwater scenarios is becoming increasingly important in missions where Autonomous Underwater Vehicles (AUVs) are used. An AUV can collect video in deep water for different purposes such as emergency situations, weather analysis, and ecosystem monitoring in maritime environments. The three most popular technologies for underwater wireless communications - acoustic, optical and RF communications - are not able to stream high definition video over large distances, even if combined. This dissertation proposes a solution for the problem of underwater wireless video streaming using the data muling concept, successfully explored in the state of the art, to transport the video data between a Survey Unit, which collects the video, and a Central Station Unit, where the video can be played back and stored for later processing. Also, since these extreme environments pose significant challenges for the implementation of a wireless communications solution, we uses the concept of Delay-Tolerant Network (DTN), which is a type of networks capable of dealing with failures, intermitent connectivity, and high delays in the communication between sender and receiver nodes. Our system takes the advantage of the long range of acoustic communications to provide a out-of-band control channel between the nodes, and of the high data rate of RF or optical communications for short range video transmission between the data mules and the other nodes.

The main contribution of this dissertation is an adaptive high definition streaming-oriented solution for underwater scenarios, combining a retransmission mechanism for reliable video exchange, an adaptive video mechanism to cope with the available throughput provided at each moment by the data mule network, and an enhanced version of the Underwater Data Muling Protocol for streaming scenarios (UDMP-S). The new protocol was specified, including new state machine diagrams and the sequence message diagrams and implemented in Python. The reliable connection was designed to use the new messages types created for UDMP-S and to be run in both nodes in the system (Central Station and Survey Units).

The streaming-oriented solution as well as the reliable virtual connection were tested in a freshwater lab testbed through airtight enclosures. The experimental results show that the proposed solution is a viable option for underwater high definition video streaming, with the experimental delays and other metrics consistent with the theoretical assumptions for the considered scenario.

Making a comparison with other solutions, we see a clear improvement, not just in waiting time but also in the Quality of Experience for the user. As an example, we can recall that for a conventional underwater acoustic communications solution only 17% of the operation time is used for streaming video when with our solution we can achieve values higher than 50% for the same parameter.

To conclude, we present the future work that can be explored in order to advance the area of underwater high definition video streaming:

- **Optimize the IBR-DTN implementation** - IBR-DTN is a framework that is still under development and has several aspects that should be improved, such as bug corrections that led to malfunctioning during the work of this dissertation. Also, there should exist an option to pass the total custody to a network intermediary node using *dtnstream*, like our data mules. This revealed to affect our solution performance because every time a new node approaches the Survey Unit, all bundles generated since the beginning of the *dtnstream* session are sent again to this new node, even if other node has already delivered them.

- **Perform tests with several simultaneous Data Mules** - In this dissertation we have only performed tests with one and two mules, but in future experiments the use of simultaneous mules can be tested, for example, using 2 mules for the best-effort connection and another one reserved to retransmission purposes in order to reduce the delay of the first bit.

- **Use an actual underwater acoustic solution in the testbed** - We have simulated the out-of-band acoustic link employing an Ethernet cable network, but as future work, the creation of a testbed with a real acoustic modems can help gather more accurate results to validate the real targeted scenario and further developments on the control network.

References

[1] J. Morgenroth, T. Pögel, and L. Wolf. Live-streaming in delay tolerant networks. Sep. 2011. doi:10.1145/2030652.2030673.

[2] F. B. Teixeira, N. Moreira, R. Campos, and M. Ricardo. Data muling approach for long-range broadband underwater communications. In *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1–4, 2019.

[3] S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. RFC 4838, Delay-Tolerant Networking Architecture. April 2007.

[4] N. F. Moreira. Data Muling for Broadband and Long Range Wireless Underwater Communications, MSc thesis, University of Porto, June 2019. URL: https://hdl.handle.net/10216/121806.

[5] GROW Project Website. [Accessed: 12/03/2021]. URL: https://grow.inesctec.pt/.

[6] K. A. Yau, A. R. Syed, W. Hashim, J. Qadir, C. Wu, and N. Hassan. Maritime networking: Bringing internet to the sea. *IEEE Access*, 7:48236–48255, 2019. doi:10.1109/ACCESS.2019.2909921.

[7] X. Che, I. Wells, G. Dickers, P. Kear, and X. Gong. Re-evaluation of RF electromagnetic communication in underwater sensor networks. *IEEE Communications Magazine*, 48(12):143–151, December 2010. doi:10.1109/MCOM.2010.5673085.

[8] M. W. Khan, Y. Zhou, and G. Xu. Modeling of acoustic propagation channel in underwater wireless sensor networks. In *The 2014 2nd International Conference on Systems and Informatics (ICSAI 2014)*, pages 586–590, Nov 2014. doi:10.1109/ICSAI.2014.7009354.

[9] H. Medwin. Speed of sound in water: A simple equation for realistic parameters, 1975. URL: https://calhoun.nps.edu/handle/10945/40176.

[10] Y. Kularia, S. Kohli, and P. Bhattacharya. Analyzing propagation delay, transmission loss and signal to noise ratio in acoustic channel for underwater wireless sensor networks. pages 1–5, 07 2016. doi:10.1109/ICPEICES.2016.7853300.

[11] P. Freitas. Evaluation of Wi-Fi Underwater Networks in Freshwater, MSc thesis, University of Porto, 2014. URL: https://hdl.handle.net/10216/75691.

[12] S. Sendra, J. Lloret, J. M. Jimenez, and L. Parra. Underwater acoustic modems. *IEEE Sensors Journal*, 16(11):4063–4071, June 2016. doi:10.1109/JSEN.2015.2434890.

[13] *Practical Guide to Radio-Frequency Analysis and Design*. All About Circuits, 2018. URL: https://www.allaboutcircuits.com/textbook/radio-frequency-analysis-design/rf-principles-components/what-is-rf-and-why-do-we-use-it/.

[14] B. Benson, Y. Li, R. Kastner, B. Faunce, K. Domond, D. Kimball, and C. Schurgers. Design of a low-cost, underwater acoustic modem for short-range sensor networks. pages 1 – 9, June 2010. doi:10.1109/OCEANSSYD.2010.5603816.

[15] S. I. Inácio, M. R. Pereira, H. M. Santos, L. M. Pessoa, F. B. Teixeira, M. J. Lopes, O. Aboderin, and H. M. Salgado. Dipole antenna for underwater radio communications. In *2016 IEEE Third Underwater Communications and Networking Conference (UComms)*, pages 1–5, Aug 2016. doi:10.1109/UComms.2016.7583457.

[16] B. C. Silva. Underwater Optical Communication: An Approach Based on LED, MSc thesis, University of Porto, 2015. URL: https://hdl.handle.net/10216/80478.

[17] N. Saeed, A. Celik, T. Y. Al-Naffouri, and M. Alouini. Underwater optical wireless communications, networking, and localization: A survey. *Ad Hoc Networks*, 94:101935, 2019. URL: http://www.sciencedirect.com/science/article/pii/S1570870518309776, doi:https://doi.org/10.1016/j.adhoc.2019.101935.

[18] C. Gussen, P. Diniz, M. Campos, W. Martins, F. Costa, and J. Gois. A survey of underwater wireless communication technologies. *Journal of Communication and Information Systems*, 31:242–255, Jan. 2016. doi:10.14209/jcis.2016.22.

[19] V. Venkataraman and H. Shah. Delay Tolerant Networking - A Tutorial. January 2009.

[20] M. S. Rahim, P. Casari, F. Guerra, and M. Zorzi. On the performance of delay — tolerant routing protocols in underwater networks. In *OCEANS 2011 IEEE - Spain*, pages 1–7, June 2011. doi:10.1109/Oceans-Spain.2011.6003388.

[21] K. Scott and S. Burleigh. RFC 5050, Bundle Protocol Specification. November 2007.

[22] W. Pöttner, F. Büsching, G. von Zengen, and L. Wolf. Data elevators: Applying the bundle protocol in Delay Tolerant Wireless Sensor Networks. In *2012 IEEE 9th International Conference on Mobile Ad-Hoc and Sensor Systems (MASS 2012)*, pages 218–226, Oct 2012. doi:10.1109/MASS.2012.6502520.

[23] H. Wang, C. Feng, and M. Luo. design and implementation of bundle protocol-based communication system. In *2010 International Conference on Educational and Information Technology*.

[24] L. Franck and F. Gil-Castineira. Using Delay Tolerant Networks for Car2Car communications. pages 2573 – 2578, July 2007. doi:10.1109/ISIE.2007.4375013.

[25] W. Pöttner, J. Morgenroth, S. Schildt, and L. Wolf. Performance comparison of DTN Bundle Protocol implementations. Sep. 2011. doi:10.1145/2030652.2030670.

[26] DTN2 Project. [Accessed: 12/03/2021]. URL: https://github.com/delay-tolerant-networking/DTN2.

[27] S. Burleigh. Interplanetary Overlay Network: An Implementation of the DTN Bundle Protocol. In *2007 4th IEEE Consumer Communications and Networking Conference*, pages 222–226, Jan 2007. doi:10.1109/CCNC.2007.51.

[28] S. Schildt, J. Morgenroth, W. Pöttner, and L. Wolf. IBR-DTN: A lightweight, modular and highly portable Bundle Protocol implementation. *Electronic Communications of the EASST*, 37:1–11, Jan 2011. URL: https://www.ibr.cs.tu-bs.de/papers/schildt-ibrdtn.pdf.

[29] OpenWrt Project Official Website. [Accessed: 12/03/2021]. URL: https://openwrt.org/.

[30] M. Dunbabin, P. Corke, I. Vasilescu, and D. Rus. Data muling over underwater wireless sensor networks using an autonomous underwater vehicle. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2091–2098, May 2006. doi:10.1109/ROBOT.2006.1642013.

[31] V. Kebkal, K. Kebkal, O. Kebkal, and M. Komar. Experimental results of Delay-Tolerant Networking in underwater acoustic channel using S2C modems with embedded sandbox on-board. In *OCEANS 2015 - Genova*, pages 1–6, May 2015. doi:10.1109/OCEANS-Genova.2015.7271702.

[32] Y. Su, R. Fan, and Z. Jin. ORIT: A Transport Layer Protocol Design for Underwater DTN Sensor Networks. *IEEE Access*, PP:1–1, May 2019. doi:10.1109/ACCESS.2019.2918561.

[33] F. B. Teixeira, R. Campos, and M. Ricardo. IEEE 802.11 Rate Adaptation Algorithms in Underwater Environment. In *Proceedings of the 10th International Conference on Underwater Networks Systems*, WUWNET '15, New York, NY, USA, 2015. Association for Computing Machinery. URL: https://doi.org/10.1145/2831296.2831312, doi:10.1145/2831296.2831312.

[34] L. Soares. Wireless Underwater Broadband and Long Range Communications using Underwater Drones as Data Mules, MSc Thesis, University of Porto, June 2017. URL: https://hdl.handle.net/10216/106809.

[35] *Advanced video coding for generic audiovisual services*. International Telecommunication Union - Joint Video Team, 2003. URL: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-200305-S!!PDF-E&type=items.

[36] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, 2003.

[37] G. J. Sullivan and T. Wiegand. Video Compression - From Concepts to the H.264/AVC Standard. *Proceedings of the IEEE*, 93(1):18–31, 2005.

[38] K. Pal, M. Govil, M. Ahmed, and T. Chawla. A Survey on Adaptive Multimedia Streaming. Dec. 2019. doi:10.5772/intechopen.86125.

[39] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann. A Survey on Bitrate Adaptation Schemes for Streaming Media Over HTTP. *IEEE Communications Surveys Tutorials*, 21(1):562–585, 2019.

[40] R. Pantos and Ed. W. May. RFC 8216, HTTP Live Streaming. August 2017.

[41] ISO/IEC 23009: Dynamic Adaptive Streaming over HTTP. April 2012.

[42] L. Yitong, S. Yun, M. Yinian, L. Jing, L. Qi, and Y. Dacheng. A study on Quality of Experience for adaptive streaming service. In *2013 IEEE International Conference on Communications Workshops (ICC)*, pages 682–686, 2013.

[43] IBR-DTN Tools. [Accessed: 12/03/2021]. URL: https://openwrt.org/packages/pkgdata_lede17_1/ibrdtn-tools2.

[44] P. U. Tournoux, E. Lochin, J. Leguay, and J. Lacan. Robust Streaming in Delay Tolerant Networks. In *2010 IEEE International Conference on Communications*, pages 1–5, May 2010. doi:10.1109/ICC.2010.5502070.

[45] S. Lenas, S. Burleigh, and V. Tsaoussidis. Reliable Data Streaming over Delay Tolerant Networks. June 2012. doi:10.1007/978-3-642-30630-3_33.

[46] S. Lenas, S. Burleigh, and V. Tsaoussidis. Bundle Streaming Service: Design, implementation and performance evaluation. *European Transactions on Telecommunications*, Nov. 2013. doi:10.1002/ett.2762.

[47] IBR-DTN Project Wiki. [Accessed: 12/03/2021]. URL: https://github.com/ibrdtn/ibrdtn/wiki.

[48] C. Raffelsberger and H. Hellwagner. A multimedia delivery system for delay-/disruption-tolerant networks. In *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 530–536, 2015.

[49] A. Thakur. SORT: A System for Adaptive Transmission of Video Over Delay Tolerant Networks. *International Journal of Wireless Networks and Broadband Technologies (IJWNBT)*, 9(2):22–49, July 2020. URL: https://ideas.repec.org/a/igg/jwnbt0/v9y2020i2p22-49.html.

[50] ALIX.3d3 Document Specification. [Accessed: 12/03/2021]. URL: https://www.pcengines.ch/pdf/alix3d3.pdf.

[51] IBR-DTN Issue 283 - [fix] dtnstream timeout mechanism for lost bundles. [Accessed: 12/03/2021]. URL: https://github.com/ibrdtn/ibrdtn/issues/283.

[52] Iperf Offical Page. [Accessed: 12/03/2021]. URL: https://iperf.fr/.