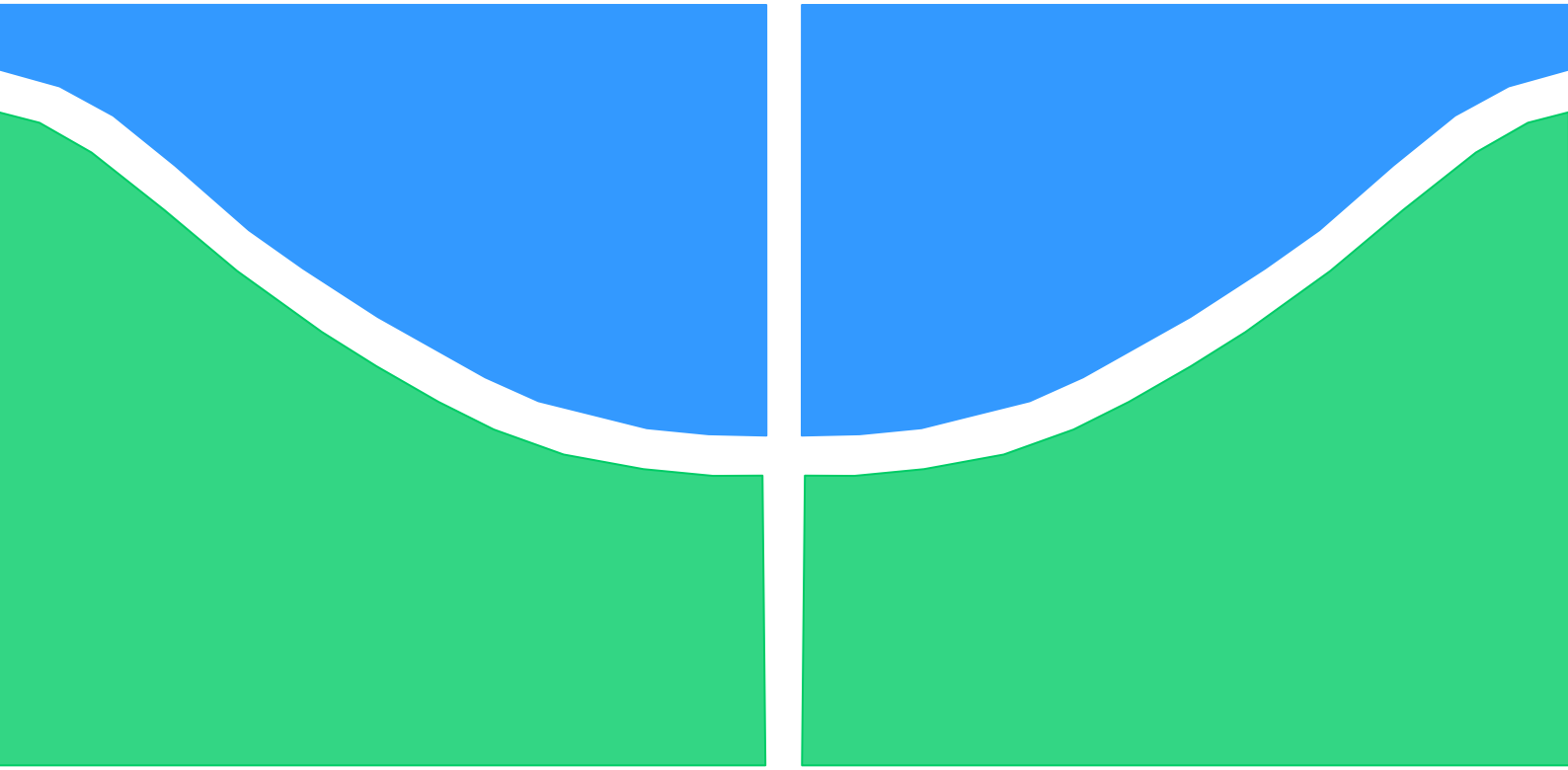


TRABALHO DE GRADUAÇÃO

PROTOCOLO DE CONTROLE DE ACESSO AO MEIO COM TRANSFERÊNCIA CONFIÁVEL PARA REDES DE COMUNICAÇÕES SUBMARINAS

Lucas Soares de Brito
Rodrigo Morais Silva

Brasília, dezembro de 2011

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

TRABALHO DE GRADUAÇÃO

PROTOCOLO DE CONTROLE DE ACESSO AO MEIO COM TRANSFERÊNCIA CONFIÁVEL PARA REDES DE COMUNICAÇÕES SUBMARINAS

Lucas Soares de Brito
Rodrigo Morais Silva

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro de Redes de Comunicação

Banca Examinadora

Prof. Marcelo M. de Carvalho, UnB/ ENE

Prof. Renato Mariz de Moraes, UnB/ ENE

Eng. Mestre Remo Z. Machado Filho, Centro de Pesquisas e Desenvolvimento da Petrobras

Dedicatória

Agradeço primeiramente a Deus, porque sem Ele eu não estaria aqui. Dedico este trabalho final à minha família, pois sem ela seria impossível ser o que sou hoje. A meus pais, Ari e Elci, por todo apoio e compreensão nos momentos difíceis. Agradeço também a todos os professores que me trouxeram os conhecimentos de engenharia que levarei para minha carreira profissional, em especial ao professor Marcelo, que com muita competência nos deu direção neste projeto, sempre nos dando a atenção e conhecimento necessários, além do auxílio e dicas do professor Renato. Aos amigos e companheiros que me ajudaram e incentivaram (e me suportaram...) durante todos esses anos. Obrigado a todos!

Lucas Soares de Brito

Primeiramente a Deus pela oportunidade de concluir este curso. À minha família pela orientação, apoio e dedicação. Aos meus amigos pela compreensão.

Rodrigo Morais Silva

RESUMO

O avanço das pesquisas sobre as redes submarinas tem o intuito de contornar as dificuldades impostas por esse canal tão adverso para as comunicações. Caracterizado pelos atrasos de propagação elevados e com alta variância, o canal submarino dificulta as comunicações por possuir banda limitada e uma baixa taxa de transferência de bits. Devido às limitações de energia e às características do canal submarino, o desenvolvimento de protocolos de controle de acesso ao meio (MAC, do inglês *medium access control*) para redes acústicas submarinas é um grande desafio. Buscando-se evitar a complexidade de sincronização de tempo de muitos protocolos de redes submarinas, desenvolveu-se aqui a extensão de um protocolo proposto na literatura, o MAC CW com recuo aleatório, que apresenta um bom desempenho em ambiente submarino.

O protocolo proposto adiciona o serviço confiável de entrega de dados na camada de enlace utilizando reconhecimentos e retransmissões. O serviço confiável de entrega de dados na camada de enlace é de extrema importância em canais submarinos, visto que as taxas de erros de bits nesse meio são imensamente superiores às encontradas em redes terrestres. Sabendo-se das limitações do canal submarino e dos recursos escassos desperdiçados quando um erro ocorre, o protocolo proposto fornece um serviço confiável de entrega de dados visando evitar uma retransmissão fim a fim dos dados por um protocolo da camada de transporte ou da aplicação. A avaliação de desempenho para o protocolo proposto é feita utilizando-se o simulador de código livre Network Simulator 3 (ns-3). Dos resultados obtidos, a vazão média da rede reduz em aproximadamente 17% em relação à vazão alcançada pelo protocolo MAC CW original. Além disso, o protocolo proposto apresenta uma menor variação relativa da vazão à medida que a densidade de nós na rede é alterada. O protocolo proposto apresenta um bom desempenho nas simulações se for considerado que o MAC CW não oferece o serviço de entrega confiável de dados.

ABSTRACT

The progress of research on underwater networks aims to overcome the difficulties imposed by a channel that is very inhospitable to communications. In addition to having high propagation delays with high variability, the underwater channel is characterized by a very limited bandwidth, which causes to low bit rates for data communications. Due to power constraints and the aforementioned characteristics, the development of medium access control (MAC) protocols for underwater acoustic networks is a very challenging problem. In this work, we make the effort to avoid the complexities of network time synchronization by proposing an extension to a simple MAC protocol proposed in the literature – the CW MAC with random backoff.

The proposed protocol extends the CW MAC by adding reliable delivery service mechanism through the use of acknowledgments and retransmissions. In fact, a reliable delivery service at the data link layer is of extreme importance in submarine channels, due to the high bit error rates that are typical of this environment (which are vastly superior to the ones found in terrestrial networks). Consequently, we avoid the waste of scarce channel resources by diminishing end-to-end retransmissions at the transport or application layers. The protocol is implemented and evaluated with the use of a discrete-time network simulator, the Network Simulator 3 (ns-3). Simulation results show that the average network throughput is only 17% lower compared to the throughput achieved by the original CW MAC protocol, which does not provide a reliable delivery service. In addition, the proposed protocol presents lower relative variations in throughput compared to the CW MAC protocol under different node densities on the network. Overall, the proposed protocol presents good performance considering that the original CW MAC does not provide a reliable delivery service of data.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS E ABREVIATURAS

Símbolos Gregos

σ - atraso na notificação de uma detecção pelo filtro casado do Micromodem WHOI.

Siglas

AUV – Autonomous Underwater Vehicles

ACK – Acknowledgement

API – Application Programming Interface

CBR – Constant Bit Rate

CDMA – Code Division Multiple Access

CSMA – Carrier Sense Multiple Access

CTS – Clear To Send

CW – Contention Window

DOTS – Delay-aware Opportunistic Transmission Scheduling

DSSS – Direct Sequence Spread Spectrum

FAMA – Floor Acquisition Multiple Access

FDMA – Frequency Division Multiple Access

FEC – Forward error correction

FHSS – Frequency Hopping Spread Spectrum

FSK – Frequency Shift Keying

ISI – Interference Intersymbol

MAC – Medium Access Control

MACA – Multiple Access with Collision Avoidance

MACAW – Multiple Access with Collision Avoidance for Wireless

NS-3 – Network Simulator 3

PPP – Point-to-Point Protocol

PSK – Phase Shift Keying

RTS – Request to Send

RX – Receptor

SNR – Signal-to-Noise Ratio

SYNC – Synchronization

TDMA – Time Division Multiple Access

TX – Transmissor

UAN – Underwater Acoustic Networks

WHOI – Woods Hole Oceanographic Institution

Capítulo 1

INTRODUÇÃO

1.1 Visão Geral

Apesar da importância dos oceanos para o ser humano, ainda sabe-se pouco sobre o ambiente que cobre a maior parte da superfície do planeta. Aproximadamente 90% do volume dos oceanos permanecem desconhecidos para o ser humano [1]. O cenário de desconhecimento é devido principalmente às suas dimensões e aos elevados custos das operações marítimas e equipamentos para coleta de dados.

A aquisição de dados oceanográficos é feita tradicionalmente através do lançamento de sondas equipadas com sensores no leito oceânico. As sondas, que são alimentadas por baterias, permanecem no fundo do oceano até que sejam coletados os dados da aplicação. Após esse período, as sondas são recolhidas e os dados armazenados em sua memória interna podem ser acessados para a análise e processamento em uma estação de controle na superfície.

Este tipo de técnica para aquisição de dados possui diversas desvantagens [2]: 1) O monitoramento não é em tempo real. Os dados coletados pelos sensores só podem ser acessados após a recuperação dos instrumentos. 2) A capacidade de armazenamento é limitada. 3) Não há detecção de falhas. Só é possível detectar uma falha nos equipamentos depois da recuperação das sondas do fundo do oceano. 4) Não há reconfiguração em tempo real do sistema. Não existe interação entre as sondas no fundo do oceano e a estação de controle na superfície.

Para superar essas limitações, é necessário que seja estabelecido um enlace de comunicação em tempo real entre os instrumentos debaixo d'água e a estação de controle na superfície. Implementações que possuem tal característica são conhecidas como *redes acústicas submarinas* (UAN, do inglês *Underwater Acoustic Networks*). Existem diversos tipos de aplicações que fazem uso deste tipo de rede. Dentre elas podemos citar:

• **Exploração submarina:** redes acústicas submarinas podem ser usadas para encontrar campos de petróleo em potencial e reservas de minerais valiosos no subsolo

marítimo. Além disto, podem ser usadas na determinação das rotas para o assentamento de cabos submarinos de comunicação.

• **Prevenção de desastres:** redes acústicas submarinas podem fornecer alertas de tsunami e maremotos para as zonas costeiras, evitando ou minimizando a proporção dos desastres.

• **Vigilância tática distribuída:** sistemas de detecção de intrusão compostos por sensores submarinos fixos em conjunto com veículos autônomos submarinos (AUV, do inglês *Autonomous Underwater Vehicles*) podem ser usados em áreas de vigilância.

• **Monitoramento Ambiental:** redes acústicas submarinas podem ser usadas no monitoramento biológico para estudo do efeito das atividades humanas sobre os ecossistemas marinhos causados principalmente pela poluição. Também podem ser usadas para o monitoramento climático através do acompanhamento das correntes marítimas.

Devido as suas propriedades físico-químicas, tais como temperatura, salinidade e densidade, o canal submarino possui características que o fazem um meio bem mais adverso para as comunicações acústicas do que o canal de rádio é para as ondas eletromagnéticas. Entre estas características podemos citar: muitas fontes de ruído; perda de caminho elevada e altamente dependente da frequência e distância; atraso elevado e com alta variância; propagação por multicaminhos e espalhamento Doppler, que levam a uma grande interferência intersimbólica (ISI, do inglês *intersymbol interference*). Essas características peculiares do meio submarino fazem com que o canal acústico tenha uma largura de banda limitada e força as redes submarinas a operarem com taxas de transmissão na ordem de quilobits por segundo (kbit/s), utilizando a tecnologia disponível atualmente.

Em comunicações onde o meio de transmissão é compartilhado por múltiplos dispositivos transmissores/receptores, é necessário algum mecanismo que controle as transmissões visando evitar colisões de sinais. Considerando um cenário de redes submarinas é possível prever a interação de múltiplos dispositivos desejando comunicar-se ao mesmo tempo utilizando o mesmo meio de comunicação. Portanto, quando existir mais de um par transmissor/receptor compartilhando o canal submarino, é necessário um protocolo de controle de acesso ao meio (MAC, do inglês *medium access control*), que coordene as transmissões. A importância na utilização de protocolos MAC se dá pela necessidade de evitar colisões no acesso ao canal, de forma a maximizar a eficiência da rede e minimizar o desperdício de energia. Tendo em vista as características severas do canal submarino, o projeto de um protocolo robusto de controle de acesso ao meio é um grande desafio.

Os protocolos MAC para redes de sensores terrestres, em suas formas originais, não funcionam bem em canais submarinos. No entanto, estes servem como embasamento teórico para a proposta de diversos protocolos de controle de acesso ao meio para o canal submarino. Na literatura científica, encontra-se que os protocolos baseados em ALOHA são bons candidatos para as redes esparsas com baixa taxa transmissão de dados, como é o caso das redes submarinas [1].

Dentre os diversos protocolos já propostos para redes submarinas, o protocolo MAC CW, proposto por Parrish et al. [1], chama a atenção pela simplicidade e pelo bom desempenho neste ambiente. Este protocolo é baseado no ALOHA com recuo aleatório. O mecanismo de recuo aleatório do MAC CW permite que a rede lide com rajadas de dados de curto prazo, como ocorre em uma rede de monitoramento utilizada para a detecção de eventos. Neste caso, embora a carga média de longo prazo seja baixa, quando ocorre um evento, a carga da rede aproxima-se das condições de saturação [1].

Apesar de conseguir bons resultados em simulações, o protocolo MAC CW não implementa a transmissão confiável de dados, característica fundamental na concepção de um projeto de redes. Com isso em mente, este trabalho propõe um protocolo de controle de acesso ao meio com transferência confiável para redes de comunicações submarinas. Motivados pelo desempenho alcançado pelo MAC CW, o protocolo proposto também possui recuo aleatório, baseado apenas no uso de informações simples sobre o estado do enlace. A transferência confiável de dados é feita através reconhecimentos e retransmissões. Acredita-se que este protocolo possa ser aplicável a uma ampla gama de redes.

1.2 Principais Contribuições

Entre as principais contribuições deste trabalho destaca-se a proposta de um protocolo de controle de acesso ao meio com transferência confiável para redes de comunicações submarinas. O protocolo desenvolvido visa evitar a complexidade do sincronismo de rede, uma vez que as características do canal submarino não favorecem essa prática. O protocolo proposto reúne a simplicidade e desempenho do protocolo MAC CW aliado à transferência confiável de dados.

Além do projeto do protocolo, implementou-se sua operação no simulador de rede de código livre ns-3. Com a implementação, foram executadas diversas simulações tendo em

vista a avaliação de desempenho do protocolo proposto frente ao MAC CW. Dos resultados obtidos, o protocolo proposto mostrou que a vazão global da rede reduziu em torno de 17% em relação à vazão alcançada pelo MAC CW original. Além do mais, o protocolo proposto apresentou bastante robustez às variações do terreno. O protocolo MAC CW apresentou, em simulação, uma variação relativa da vazão em torno de 0.13 entre o tamanho de terreno mínimo (200m x 200m) e o máximo (1.000m x 1.000m); resultado superior à variação relativa da vazão encontrada para o protocolo proposto, em torno de 0.07. Outro resultado importante, obtido em simulações, foi que o protocolo proposto apresenta uma menor variação relativa da vazão para diferentes densidades de nós na rede. Isso representa a robustez do protocolo, visto que um erro na definição do tamanho da janela de contenção ideal, não levaria a uma grande perda de vazão. Em linhas gerais, o protocolo proposto obteve bons resultados na avaliação de desempenho nos cenários estudados, principalmente se for levado em consideração que o protocolo MAC CW não implementa o serviço confiável de entrega de dados.

1.3 Apresentação do Trabalho

O presente trabalho está organizado da seguinte maneira. No Capítulo 2, é feito um estudo sobre os fundamentos de redes submarinas, apresentando uma visão geral sobre as características peculiares desse tipo de rede. O Capítulo 3 apresenta os trabalhos relacionados ao desenvolvimento de protocolos de controle de acesso ao meio para redes de comunicações submarinas. No Capítulo 4, descreve-se o protocolo MAC CW, no qual este trabalho se baseia. O Capítulo 5 apresenta a proposta de protocolo de controle de acesso ao meio com transferência confiável para redes de comunicações submarinas. No Capítulo 6 é apresentada uma breve introdução ao simulador ns-3. No Capítulo 7 é feita a avaliação de desempenho do protocolo desenvolvido através da execução de simulações e coleta de dados para diferentes cenários. Finalmente, o Capítulo 8 encerra este trabalho com as conclusões e sugestões de pesquisas futuras.

Capítulo 2

FUNDAMENTOS DE REDES SUBMARINAS

2.1 Características do Canal Submarino

A propagação de ondas eletromagnéticas no oceano só atinge grandes distâncias quando estão na faixa de frequência entre 30-300 Hz. Transmissões nessa faixa de frequência exigem potência elevada e antenas de grandes dimensões. Uma transmissão submarina na faixa de 433 MHz, utilizando modems comerciais de radiofrequência, é capaz de propagar-se por apenas 120 cm de distância [2]. Uma alternativa às ondas eletromagnéticas é a utilização de ondas ópticas, pois este tipo de onda sofre menos atenuação no meio submarino. Porém, a transmissão de sinais ópticos é afetada severamente pela dispersão, além de requerer uma precisão elevada no apontamento dos lasers. Desta forma, a comunicação acústica se torna a melhor alternativa para transmissões no canal submarino.

A comunicação acústica submarina é influenciada principalmente pelos seguintes fatores [2]:

• **Ruído:** são muitas as fontes de ruído que degradam o canal submarino. O ruído provocado pelo ser humano é causado principalmente por máquinas (bombas, redutores, usinas de energia), e a atividade de transporte, especialmente em áreas que apresentam tráfego de navios pesados. Além do ruído causado pelo ser humano, existe o ruído ambiental, que está relacionado com o movimento das águas, incluindo as marés, correntes, tempestades, ventos e chuvas.

• **Perda de Caminho:** a atenuação do sinal acústico é provocada principalmente pela conversão de energia acústica em calor. Quanto maiores forem à distância e a frequência, maior será a atenuação do sinal acústico, como pode ser visualizado na Figura 1. As reverberações na superfície e no fundo do oceano também causam a atenuação do sinal acústico, bem como o espalhamento, refração e dispersão (devido ao deslocamento do ponto de reflexão causado pelo vento na superfície). A expansão das frentes de ondas

resultantes da propagação acústica também provoca a perda de caminho, que neste caso aumenta com a distância da propagação e é independente da frequência.

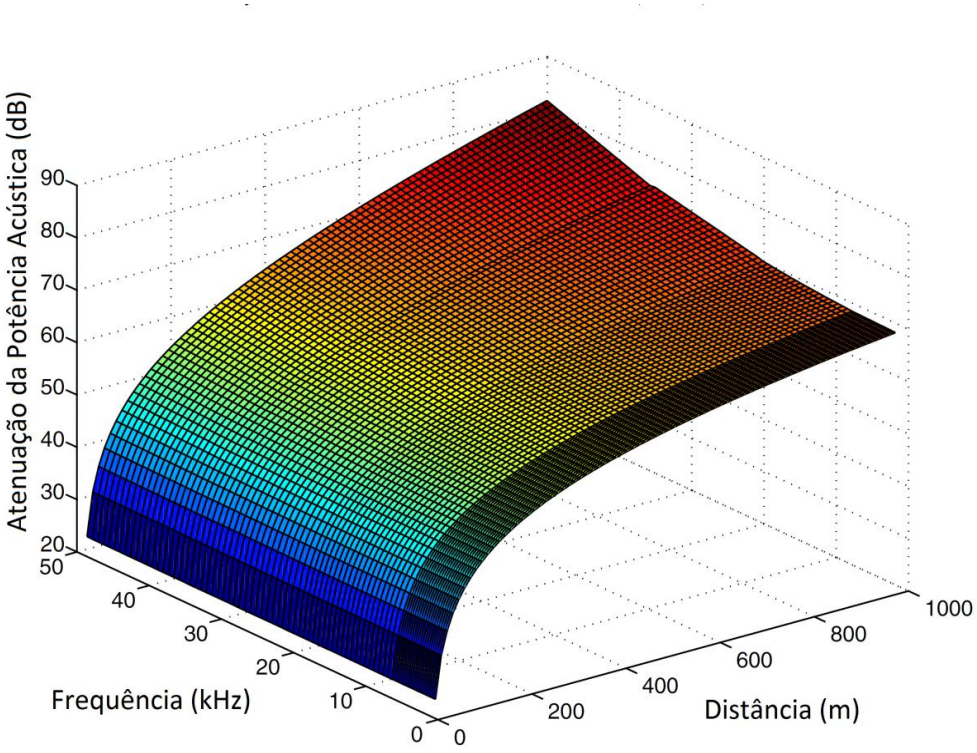


Figura 1 - Perda de caminho em canal submarino versus distância e frequência na faixa de 1 a 50 kHz. Adaptado de [2].

• **Propagação por Multicaminhos:** a propagação do sinal acústico por multicaminhos é responsável pela degradação severa do sinal de comunicação, uma vez que gera interferência intersimbólica (*ISI*). O grau de interferência das várias cópias do sinal acústico que chegam no receptor depende da geometria do enlace acústico. Canais horizontais possuem dispersões extremas de multicaminhos, enquanto canais verticais são caracterizados pela dispersão curta. Assim, a extensão da propagação é uma forte função da profundidade e da distância entre o transmissor e o receptor.

• **Atraso de propagação elevado e com alta variância:** a velocidade de propagação média de ondas acústicas no canal submarino é de 1.500 m/s, o que é muito inferior à velocidade de 299.792.458 m/s das ondas eletromagnéticas no canal de rádio. O rendimento dos sistemas acústicos pode ser degradado substancialmente por atrasos de propagação altos da ordem de 0,67 s por quilômetro. No entanto, mais prejudicial ainda é o fato deste atraso ter alta variância. Assim, o projeto de um protocolo eficiente torna-se bastante difícil, uma vez que muitos dos protocolos de comunicação comuns dependem de uma estimativa precisa do tempo de viagem de ida e volta.

• **Espalhamento Doppler:** o espalhamento Doppler é uma medida do alargamento espectral causado pelo movimento relativo entre o transmissor e o receptor, ou até mesmo de objetos ao redor deles. Essas perturbações resultam em uma modulação aleatória em frequência que se faz presente em cada um dos componentes de multipercurso. No receptor, o conjunto somado dessas componentes causa um alargamento espectral do sinal, chamado de espalhamento Doppler. O espalhamento Doppler é bastante significativo em canais submarinos. A degradação no desempenho da comunicação digital causada pelo espalhamento Doppler é devido aos símbolos adjacentes que se interferem no receptor. Isso requer processamento de sinal muito sofisticado para lidar com a interferência inter-simbólica resultante.

2.2 Banda e Capacidade do Canal Submarino

A maioria das características descritas na Seção 2.1 é causada pelas propriedades físico-químicas do meio submarino, tais como temperatura, salinidade e densidade. Todos esses fatores fazem com que o canal acústico seja variável no tempo e no espaço, e determinam a largura de banda disponível como sendo limitada e dramaticamente dependente da distância e da frequência. Desta forma, os sistemas que operam em algumas dezenas de metros podem ter uma largura de banda de mais de uma centena de kHz, enquanto que sistemas que operam sobre várias dezenas de quilômetros podem ter uma largura de banda de poucos kHz, como pode ser observado na Tabela 1.

Tabela 1 – Largura de banda disponível para diferentes alcances em canais acústicos submarinos. Adaptado de [2].

Alcance	Distância (km)	Largura de banda (kHz)
Muito longo	1000	<1
Longo	10–100	2–5
Médio	1–10	≈ 10
Curto	0.1–1	20–50
Muito curto	<0.1	>100

Mesmo operando nas melhores condições e em distâncias curtas, as redes acústicas submarinas operam a uma taxa baixa de bits, na ordem de quilobits por segundo (kbit/s). Parrish et al. [1] realiza o cálculo para a capacidade do enlace acústico. Para isso, foram examinados canais que possuem três perfis de velocidade de som distintos. Os perfis de velocidade do som utilizados no cálculo são mostrados na Figura 2. O primeiro perfil apresenta uma camada de ar quente logo acima da superfície da água, o que faz com que a velocidade do som seja alta próximo à superfície. Logo abaixo dessa camada se encontra uma região chamada de termoclina que apresenta uma variação brusca de temperatura. Esta característica peculiar faz com que a velocidade do som diminua drasticamente nessa região. O segundo perfil apresenta velocidade do som decrescente com a profundidade, mas sem apresentar variações abruptas como no primeiro perfil. O terceiro apresenta velocidade do som aumentando com a profundidade. As diferenças entre os perfis são resultantes das propriedades físicas e químicas tais como salinidade, temperatura e densidade do oceano onde estes perfis foram medidos. Os perfis foram medidos no mês de setembro no Mar Mediterrâneo, em junho, na Baía de Griffin perto das ilhas de San Juan, e em fevereiro no Mar de Bering, respectivamente.

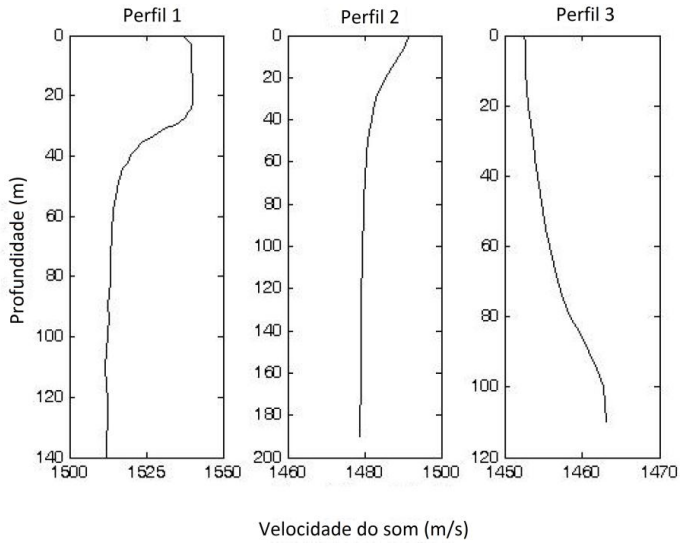


Figura 2 – Perfis de velocidade do som. Adaptado de [1].

As Figuras 3, 4, 5 e 6 mostram os cálculos de capacidade medida entre fonte e receptor em profundidades de 10, 20, 70, 105 e 170 metros, e sob condições de terreno de areia com grãos de médio diâmetro (entre 1,2 mm e 2,4 mm) e argila do tipo siltosa e velocidade do vento entre 2,5 m/s e 10 m/s. Através dos gráficos é possível visualizar como a capacidade do canal é influenciada pelo tipo de terreno. O terreno que possui areia com

grãos de diâmetro médio é menos atenuante que a argila do tipo siltosa. Pode-se notar também que quando a fonte e o receptor estão ambos acima ou abaixo da termoclina a capacidade é maior do que quando eles estão separados pela termoclina. Além disso, as perdas de superfície devido ao vento impactam mais na capacidade do canal quando a fonte e o receptor estão acima da termoclina do que quando a fonte e o receptor estão abaixo da termoclina.

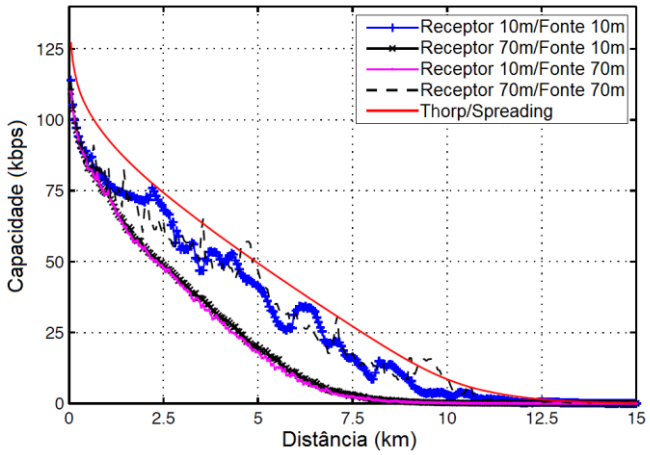


Figura 3 – Capacidade calculada para o perfil 1 da Figura 2 considerando um terreno submarino constituído de areia de grãos de médio diâmetro e com velocidade do vento de 2,5 m/s. Adaptado de [1].

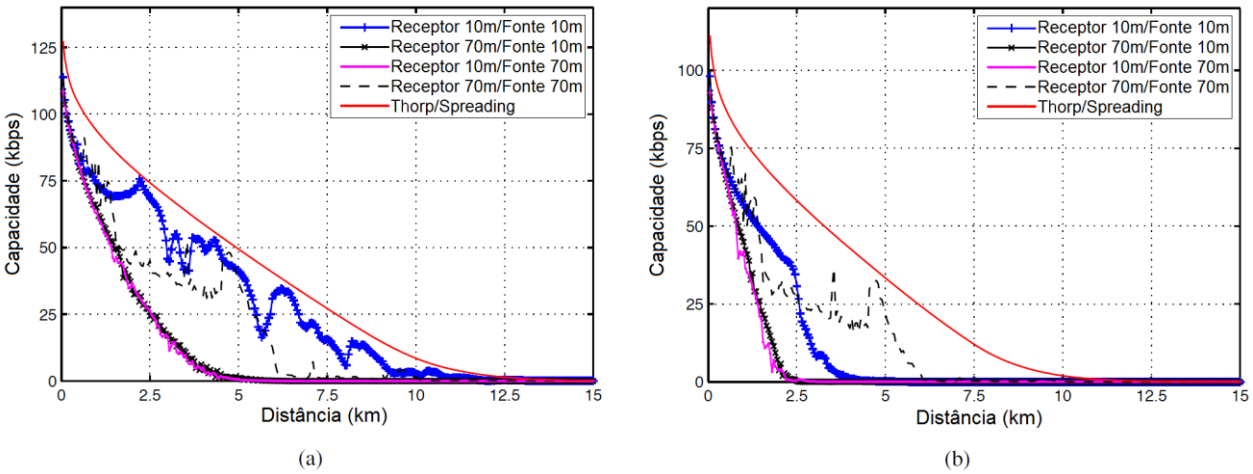


Figura 4 – Capacidade para o perfil 1 da Figura 2 considerando um terreno submarino constituído de argila do tipo siltosa e com velocidade do vento de 2,5 m/s (a) e 10 m/s (b). Adaptado de [1].

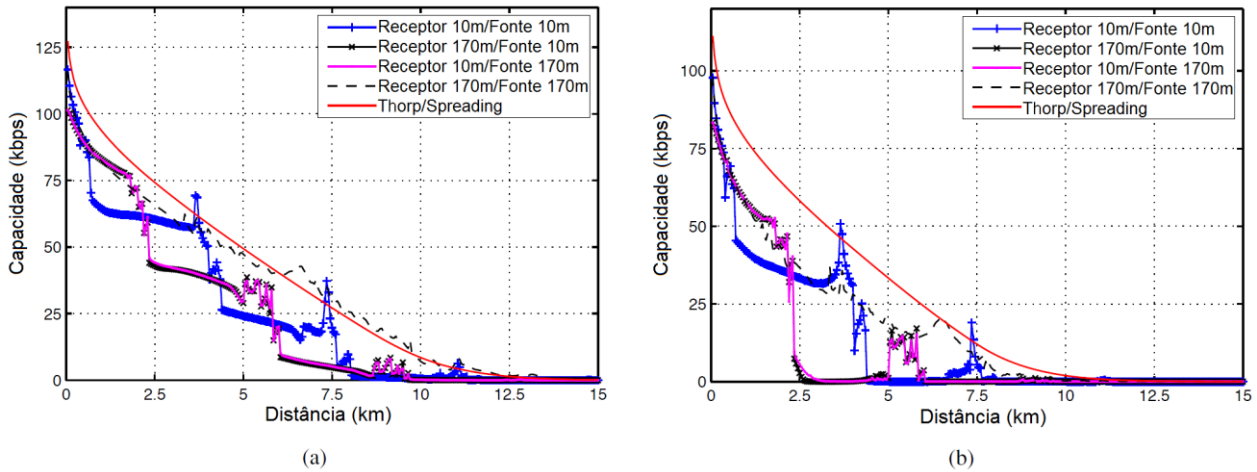


Figura 5 – Capacidade para o perfil 2 da Figura 2 considerando um terreno submarino constituído de argila do tipo siltosa e com velocidade do vento de 2,5 m/s (a) e 10 m/s (b). Adaptado de [1].

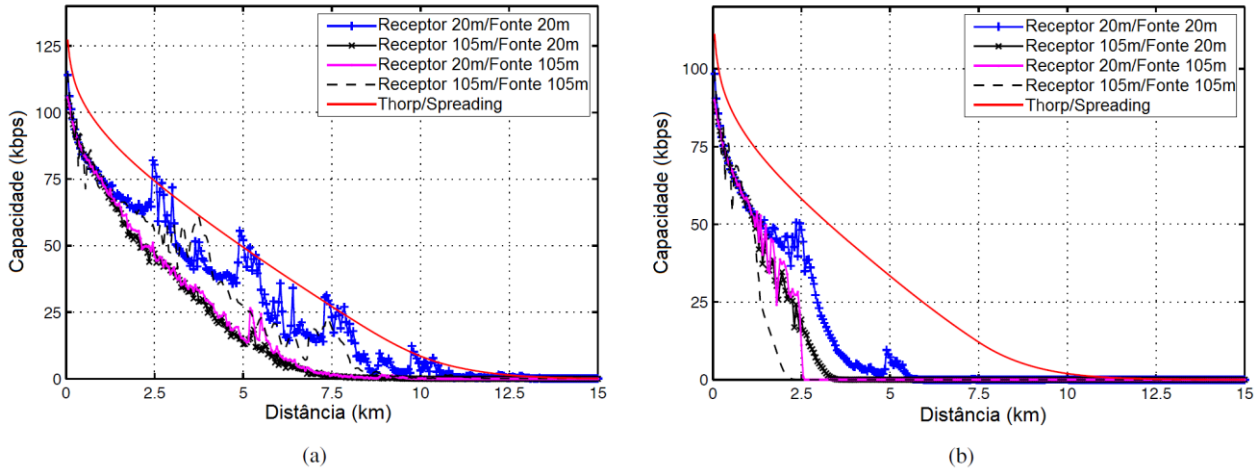


Figura 6 – Capacidade para o perfil 3 da Figura 2 considerando um terreno submarino constituído de argila do tipo siltosa e com velocidade do vento de 2,5 m/s (a) e 10 m/s (b). Adaptado de [1].

2.3 Modelos do Canal Submarino

O modelo do canal de comunicação é uma representação matemática que busca caracterizar as perdas de potência do sinal que se propaga, de modo que se possa estimar a intensidade com a qual um sinal transmitido chega ao seu receptor. Para tal, emprega-se a relação sinal-ruído (SNR, do inglês *signal-to-noise ratio*) que expressa à razão do sinal útil recebido oriundo do transmissor pela energia total do ruído. Assim, o modelo do canal de

comunicação leva em consideração a maioria das características da propagação do sinal através do meio, incluindo a atenuação e ruído ambiente.

A perda relativa à propagação do sinal é o somatório das perdas relativas à atenuação e à perda devido ao espalhamento. A perda por espalhamento é devido a um efeito geométrico que representa o enfraquecimento regular do um sinal à medida que ele se afasta da fonte. Perda por atenuação é provocada principalmente pela conversão de energia acústica em calor. Como foi visto na Figura 1, quanto maiores forem à distância e a frequência, maior será a atenuação do sinal acústico.

Modelar o canal acústico submarino é uma tarefa de extrema complexidade, devido principalmente as condições de contorno (perfis de superfície e fundo), que influenciam na perda de energia e na dispersão do sinal. Portanto, a modelagem do canal de comunicação submarino exige um equilíbrio entre a precisão do modelo e a complexidade computacional. Modelos muito complexos levariam muito tempo para serem executados, o que inviabiliza a análise de desempenho baseada em simulações.

Um dos modelos do canal acústico submarino mais utilizados em pesquisas é o modelo BELLHOP. O modelo BELLHOP foi desenvolvido originalmente no *Naval Ocean Systems Center*, e continuou a se desenvolver no *Naval Research Laboratory*. A partir daí, muitos institutos ao redor do mundo se interessaram pelo modelo que é considerado um dos mais fieis as características do canal submarino. Dentre as instituições que desenvolveram o modelo Bellhop, pode-se citar a *Ocean Acoustic Library*. O modelo BELLHOP desenvolvido pela *Ocean Acoustic Library* foi escrito em Fortran por Michael Porter e está disponível no sítio desta instituição [3]. O modelo foi desenvolvido para considerar a propagação dos raios acústicos em duas dimensões, tendo como saída às coordenadas cartesianas do raio acústico, baseado no tempo de viagem, amplitude, pressão acústica e na perda de caminho. O modelo BELLHOP requer a solução das equações dinâmicas dos raios acústicos, que são descritos em detalhes por Porter et al. [4]. Para um sistema com simetria cilíndrica as equações de raios podem ser escritas como:

$$\begin{aligned} \frac{dr}{ds} &= c\xi(s)\ , \qquad \frac{d\xi}{ds} = -\frac{1}{c^2}\frac{\partial c}{\partial r}\ , \\ \frac{dz}{ds} &= c\zeta(s)\ , \qquad \frac{d\zeta}{ds} = -\frac{1}{c^2}\frac{\partial c}{\partial z}\ , \end{aligned} \tag{1}$$

Onde $r(s)$ e $z(s)$ representam as coordenadas em coordenadas cilíndricas e s é o comprimento de arco ao longo do raio; o par $c(s)$ $[\xi(s), \zeta(s)]$representa o versor tangente ao longo do raio.

O modelo de propagação Bellhop lê informações de propagação de um banco de dados. Um arquivo de configuração descrevendo o local, e a resolução das informações arquivadas devem ser fornecidos através de atributos. Infelizmente o modelo de propagação Bellhop não está disponível na versão do simulador utilizado na avaliação de desempenho deste trabalho. Portanto utilizou-se o modelo de propagação Thorp para caracterizar a propagação da onda acústica no canal submarino. A utilização desse modelo implica caracterizar a onda acústica apenas pela sua perda de caminho e não considerando, portanto, os perfis de superfície e fundo que exercem considerável impacto através de perda de energia e dispersão de ondas acústicas; nem tão pouco considera-se as variações de pressão e de velocidade do som como são considerados no modelo Bellhop.

Modela-se a energia acústica dissipada em forma de calor através da atenuação Thorp seletiva em frequência [1]. A atenuação Thorp é representada pela equação

$$A(d, f) = d \cdot 10\log(a(f)), \qquad (2)$$

sendo

$$10\log(a(f)) = 0.1\frac{f^2}{1+f^2} + 40\frac{f^2}{4100+f^2},$$

onde *10log(a(f))* é Atenuação Thorp, dada em decibel (dB) por quilojarda. E *A(d, f)* é a perda por absorção em dB para um raio de frequência *f*, em kHz, que propaga-se por *d* quilojardas.

2.4 Topologia de Redes Submarinas

Em geral, as missões de observação submarinas são extremamente caras, devido principalmente à complexidade e custos das operações de lançamento dos equipamentos para coleta de dados no oceano. Por isso, é de extrema importância garantir confiabilidade da rede implantada de forma a prevenir o insucesso dessas missões. A concepção da topologia da rede submarina torna-se crucial de modo a evitar pontos únicos de falha e

gargalos na comunicação que possam comprometer o funcionamento global da rede. Além do mais, a topologia deve ser projetada de forma cuidadosa, pois ela influencia diretamente o consumo de energia, a capacidade e a confiabilidade de uma rede. Basicamente pode-se distinguir três topologias básicas para redes acústicas submarinas, conforme detalhado a seguir.

• **Redes submarinas estáticas bidimensionais:** neste tipo de rede os sensores ficam posicionados fixamente no fundo do oceano com o auxílio de âncoras. No mesmo nível dos sensores ficam equipamentos chamados de coletores, que são responsáveis por retransmitir os dados da rede do fundo do oceano para a superfície. Os nós sensores e o nó coletor são interligados através de enlaces acústicos. O nó coletor é equipado com dois transceptores acústicos, um horizontal para comunicação com os nós sensores e um vertical para enviar os dados para a estação na superfície. A estação de superfície é equipada com um transmissor acústico que é capaz de lidar com múltiplas comunicações em paralelo com os nós coletores implantados. A estação de superfície também possui equipamentos de radio frequência ou transceptores via satélite para comunicação com outras estações em terra, como pode ser observado na Figura 7.

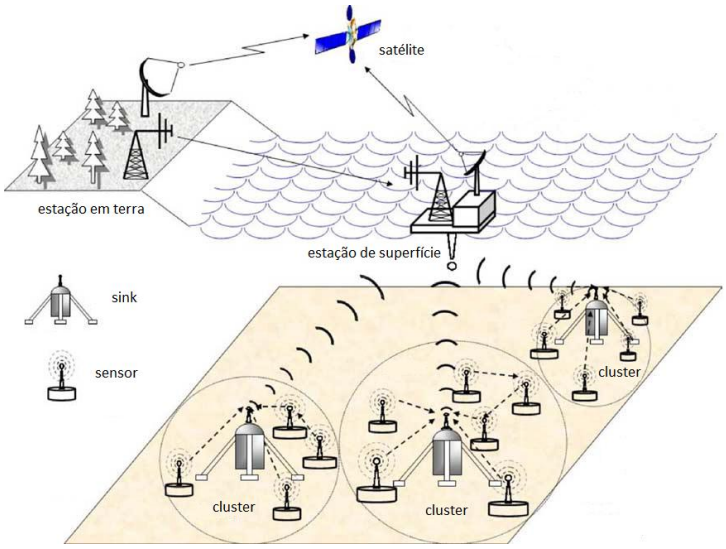


Figura 7 – Rede submarina estática bidimensional. Adaptado de [2].

• **Redes submarinas estáticas tridimensionais:** neste tipo de rede os nós sensores ficam posicionados em diferentes profundidades através da utilização de bóias e âncoras, como pode ser observado na Figura 8. A bóia empurra o nó sensor para a superfície, e um cabo de sustentação de tamanho regulável ligado a âncora permite o ajuste da profundidade. Um desafio a ser enfrentado neste tipo de topologia é o efeito das correntes oceânicas que podem alterar a posição dos nós sensores. Este tipo de topologia é usada

principalmente para observar fenômenos que não podem ser devidamente verificados por meio de nós sensores no fundo do oceano.

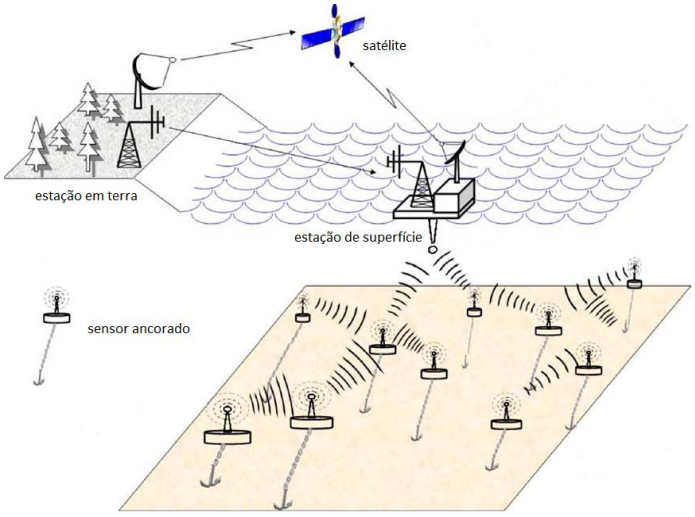


Figura 8 – Rede submarina estática tridimensional. Adaptado de [2].

• **Redes submarinas dinâmicas tridimensionais:** este tipo de rede faz uso de veículos submarinos autônomos (AUV). Os AUVs se assemelham à minisubmarinos equipados com sensores, tendo uma variedade de aplicações em oceanografia, monitoramento ambiental, e estudo dos recursos subaquáticos. Eles podem ser usados para melhorar as capacidades de redes submarinas. Estes veículos podem ser deslocados de forma dinâmica conforme a necessidade de coleta de dados. Também podem ser deslocados para sanar falhas na conectividade quando um nó fixo para de funcionar. Em alguns casos esses veículos podem ser usados na implantação e manutenção de novos nós sensores na rede.

Na Figura 9 é mostrado o SeaBED, um AUV desenvolvido pela equipe liderada pelo pesquisador Hanumant Singhe da *Woods Hole Oceanographic Institution* (WHOI) [5]. O SeaBED pode chegar a profundidade de 2.000 metros, tornando-o particularmente adequado para coletar imagens do fundo do mar. Esse AUV tem a capacidade de manter a profundidade constante e velocidade de um nó e meio.

Figura 9 – SeaBED – AUV desenvolvido pela WHOI. Fonte [5].

Dentro dessas topologias, os nós sensores podem enviar seus dados para os nós coletores através de ligações diretas ou de caminhos com múltiplos saltos. No caso de ligações diretas, cada nó sensor envia os dados coletados diretamente ao nó coletor mais próximo. Apesar da simplicidade dessa abordagem, esta solução não é a mais eficiente energeticamente uma vez que a potência de transmissão será em geral maior do que em redes com múltiplos saltos. Além disto, ligações diretas reduzem a vazão da rede devido ao aumento da interferência acústica causada pela potência de transmissão alta de outros transmissores. No caso de caminhos com múltiplos saltos, os dados coletados por um nó sensor são encaminhados para o nó coletor por meio de nós sensores intermediários. Apesar do aumento na complexidade do algoritmo de roteamento, esta abordagem tende a aumentar a vazão global da rede e a poupar energia dos nós sensores.

Capítulo 3

REVISÃO BIBLIOGRÁFICA

3.1 Protocolos de Controle de Acesso ao Meio

Em estudos de redes de comunicação, podemos identificar dois tipos de enlaces: ponto a ponto e compartilhado. O enlace ponto a ponto é caracterizado por um único remetente em uma extremidade do enlace e um único receptor na outra extremidade do enlace. Existem vários protocolos implementados para esse tipo de enlace, entre eles podemos citar o protocolo ponto-a-ponto (PPP, do inglês *point-to-point protocol*) [6]. Já o enlace compartilhado é caracterizado por ter muitos dispositivos usando o mesmo canal de comunicação. Entre os protocolos da camada de enlace do tipo compartilhado podemos citar o Ethernet [7]. Como o canal submarino é um canal do tipo compartilhado, concentrar-se-á a partir de agora neste tipo enlace.

Se houver apenas um par de dispositivos utilizando o canal submarino para se comunicar, o enlace será do tipo ponto-a-ponto e não haverá necessidade de mecanismos que controlem o acesso ao canal. No entanto, a maioria das aplicações submarinas faz uso de múltiplos sensores que necessitam se comunicar pelo mesmo canal de comunicação. Um dos principais problemas enfrentados na camada de enlace em canais compartilhados é conseguir coordenar as transmissões de forma a evitar colisões de quadros transmitidos. Protocolos que realizam esta função são denominados de protocolos de controle de acesso ao meio. Como os nós em uma rede agem, na maioria das vezes, de forma independente, dois nós podem querer transmitir seus quadros ao mesmo tempo. Quando isso ocorre, os quadros colidem e os demais nós da rede não conseguem decifrar nenhum dos dois quadros transmitidos. Em eventos assim o canal foi desperdiçado durante o período de colisão e os quadros colididos são descartados. Tal situação se agrava com o aumento da quantidade de nós ativos compartilhando o mesmo canal, podendo fazer com que a rede fique inoperante devido à quantidade de colisões.

A função de coordenar as transmissões é desempenhada pelos protocolos de controle de acesso ao meio. Durante anos, foram propostos e implementados dezenas de

protocolos de acesso múltiplo que foram sendo modificados de acordo com a evolução da tecnologia. A seguir serão descritos os principais métodos e protocolos de controle de acesso ao meio utilizados em redes sem fio terrestres e suas limitações no ambiente submarino.

O protocolo de Acesso Múltiplo por Divisão de Frequência (FDMA, do inglês *Frequency Division Multiple Access*) é o mais simples de todos os métodos de acesso ao meio, uma vez que ele apenas divide a faixa de frequências disponível em sub-faixas e as atribui a usuários individuais. Pelo fato de a largura de banda ser limitada no ambiente submarino, o FDMA não é adequado para redes acústicas submarinas, uma vez que os sistemas de banda limitada são vulneráveis ao desvanecimento multipercurso.

O protocolo de Acesso Múltiplo por Divisão de Tempo (TDMA, do inglês *Time Division Multiple Access*) divide o tempo disponível da comunicação em intervalos fixos e subsequentes, chamados de *time slots*. O canal acústico fica reservado para transmissão de dados para apenas um dos nós da rede durante cada *time slot*. Como discutido na Seção 2.1, o canal submarino possui atraso de propagação elevado, e com alta variância. O TDMA possui baixa eficiência de largura de banda em canais acústicos submarinos, uma vez que é necessário utilizar tempos de guarda grandes entre *slots*. O tempo de guarda deve ser utilizado para minimizar colisões de pacotes em *slots* de tempo adjacentes. Além disso, para a utilização do TDMA, faz-se necessário uma referência de tempo comum para sincronizar as estações, o que é muito difícil de implementar em ambientes submarinos com atrasos de propagação variáveis.

O protocolo de Acesso Múltiplo por Divisão de Código (CDMA, do inglês *Code Division Multiple Access*) distingue os sinais transmitidos simultaneamente por múltiplos dispositivos por meio de códigos pseudo-aleatórios que são usados para espalhar o sinal do usuário sobre toda a banda disponível. Essa característica faz com que o CDMA seja bastante robusto para desvanecimento seletivo em frequência, causado pela propagação por multicaminhos debaixo da água. Com isto, o CDMA evita retransmissões de pacotes, resultando em economia de energia e aumento de vazão da rede, fazendo do CDMA um método promissor de acesso múltiplo em redes submarinas [2]. O método para espalhamento no espectro pode ser o FHSS (do inglês *Frequency Hopping Spread Spectrum*), usando FSK (do inglês *Frequency Shift Keying*) para baixas taxas de transmissão, ou o DSSS (do inglês *Direct Sequence Spread Spectrum*), usando PSK (do inglês *Phase Shift Keying*) para altas taxas de transmissão [8].

No protocolo ALOHA original, toda vez que um nó possui dados para enviar ele transmite-o imediatamente. Se o pacote for recebido sem erros, o receptor envia um pacote de reconhecimento ACK (do inglês *acknowledgment*). Se o transmissor não receber o

pacote de reconhecimento devido a uma colisão, ele retransmite seu pacote de dados depois de esperar um tempo aleatório. Como os tempos de retransmissão são aleatórios e escolhidos independentemente, as chances da colisão repetir-se são baixas. Devido às retransmissões, o protocolo ALOHA tem uma vazão máxima de 18%. Para aumentar essa vazão, foi proposto o método slotted ALOHA. Como o próprio nome diz, neste protocolo cada nó da rede é sincronizado e o tempo é dividido em *slots*. Um nó só é permitido enviar pacotes no início de cada *slot* de tempo. Essa mudança no protocolo faz com que a vazão máxima seja aumentada para 36%. Pela simplicidade, protocolos baseados em ALOHA são bons candidatos para redes acústicas submarinas. Já o slotted ALOHA não é adequado devido à premissa de sincronização da rede. Como foi visto, o canal acústico submarino é caracterizado por atrasos de propagação elevados e com alta variância.

O protocolo ALOHA possui uma vazão baixa, mesmo em sua versão *slotted*. O fato de os nós não levarem o estado do canal em conta resulta em uma alta taxa de colisões. O protocolo CSMA (do inglês *Carrier Sense Multiple Access*) tenta evitar as colisões ouvindo o canal antes de uma transmissão [9]. Porém esse método não evita que uma colisão ocorra no receptor devido ao problema conhecido como terminal escondido, mostrado na Figura 10. Na Figura 10, o alcance da transmissão é representado por círculos em volta de cada nó. Suponha que o nó C esteja enviando um pacote para o nó B. Ao mesmo tempo, o nó A adquire um pacote para transmissão para o nó B. O nó A escuta o canal e não detecta a transmissão de C porque está fora de seu alcance; dessa forma o nó A inicia sua transmissão. Isso cria uma colisão em B. O nó C estava escondido do nó A. Essa situação é chamada de problema do terminal escondido. Para que não ocorram colisões em B, o nó A deve adiar a sua transmissão. No entanto, não há motivo de adiar a transmissão se o destino do pacote de A não é B e se o nó B possuir a capacidade de lidar com a interferência gerada pela transmissão do nó A. A situação onde um nó adia a transmissão devido à escuta de uma portadora que não irá causar interferência em sua transmissão é conhecida como problema do terminal exposto.

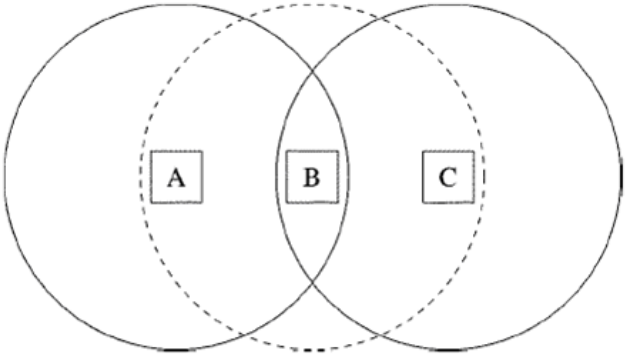


Figura 10 – O problema do terminal exposto e escondido. Adaptado de [4].

Uma forma de minimizar o problema do terminal exposto e escondido é a adição de um tempo de guarda entre as transmissões. Mas como já foi dito, os atrasos de propagação extensos dos canais submarinos podem tornar esse método muito ineficiente.

Em alternativa ao CSMA foi proposto o protocolo MACA (do inglês *Multiple Access with Collision Avoidance*) visando evitar colisões no receptor [10]. Este protocolo faz uso de dois pacotes de sinalização chamados de RTS (do inglês *request-to-send*) e CTS (do inglês *clear-to-send*). Considerando novamente a Figura 10, quando o nó A desejar enviar um pacote para o nó B, ele enviará um pacote de RTS contendo o comprimento do pacote de dados a ser enviado. Se B receber o pacote de RTS, ele enviará um pacote de CTS autorizando a transmissão e sinalizando a qualquer outro nó que esteja ao seu alcance (no caso o nó C) que adie sua transmissão pelo espaço de tempo do comprimento da mensagem de dados. Quando o pacote A recebe o CTS, ele inicia a transmissão do pacote de dados. Se um nó recebe um RTS, mas não um CTS, decide que ele está fora de alcance do receptor e transmite seus próprios pacotes. Portanto, este protocolo pode resolver tanto o problema do terminal oculto quanto o problema do nó exposto. Apesar de possuir a capacidade de evitar colisões, o protocolo MACA adiciona sobrecarga à rede e maior gasto energético devido às trocas de sinalização RTS-CTS. Em redes submarinas, que necessitam de protocolos eficientes energeticamente, a utilização do MACA pode inviabilizar algumas aplicações devido ao alto consumo de bateria. Porém, a redução de retransmissões pode compensar esses aumentos na sinalização.

Para melhorar o desempenho e a confiabilidade do MACA foi proposto o protocolo MACAW (do inglês *Multiple Access with Collision Avoidance for Wireless*) [11]. No MACA, a recepção do pacote não é reconhecida. Como o MACA foi desenvolvido para enlaces altamente confiáveis, onde os pacotes raramente contêm erros, o envio de uma confirmação leva a um desperdício de tempo. Se o canal de comunicação é de má qualidade, como é o caso dos canais submarinos, pacotes errados serão mais frequentes. Dessa forma, no MACAW, um pacote de reconhecimento é transmitido após cada recepção bem sucedida. A inclusão de um pacote extra na transação aumenta a sobrecarga, o que diminui o rendimento. No entanto, para os canais submarinos, o ganho em rendimento pode compensar o aumento da sobrecarga em sinalização.

O protocolo FAMA (do inglês *Floor Acquisition Multiple Access*) também promove o controle de acesso ao meio através de trocas dos pacotes de RTS e CTS [12]. Porém para prevenir as colisões entre os pacotes de sinalização e os pacotes de dados, o protocolo FAMA faz com que a duração dos pacotes de sinalização seja maior que o tempo máximo de propagação do canal. Desta forma a ausência de colisões é garantida se a duração do pacote RTS for maior que o atraso de propagação do canal e se a duração do CTS for igual

ao atraso na comutação do hardware entre transmissor (TX) e receptor (RX), mais duas vezes o maior atraso de propagação. Apesar destas condições garantirem a ausência de colisões, também introduzem uma grande latência na rede. Sabendo do elevado atraso do canal submarino, a latência gerada pelo FAMA pode reduzir substancialmente a vazão da rede, além de representar um alto consumo de energia.

O protocolo S-MAC (do inglês *Sensor-MAC Protocol*) é um protocolo projetado para redes de sensores terrestres sem fio, onde o principal objetivo é a redução no consumo de energia [13]. Uma das maiores fontes de desperdício energético é a escuta ociosa do canal de comunicação. O protocolo S-MAC reduz o tempo de monitoramento do canal, deixando o nó entrar em modo de hibernação, no qual o nó desliga seu rádio, e define um temporizador para acordar mais tarde. Todos os nós da rede mantêm uma tabela que contém a programação de horários de hibernação de todos os seus vizinhos. Para escolher a programação de horário de hibernação, o primeiro nó escuta o canal à espera de mensagens SYNC (do inglês *synchronization*) contendo a programação de outro nó. Pacotes SYNC são muito curtos, pois possuem apenas o endereço do remetente e seu período de hibernação. Depois de um período de escuta, se o nó não receber nenhum pacote SYNC, ele escolhe aleatoriamente um horário de hibernação e manda seu próprio pacote SYNC. Os outros nós da rede irão sincronizar com ele. Se um nó recebe um pacote SYNC de um vizinho antes de escolher seu próprio horário de sono, ele define sua programação como sendo igual ao do nó que lhe enviou um pacote SYNC. Em seguida, transmite sua programação. Se um nó recebe uma programação diferente depois que seleciona e difunde seu próprio horário, adota os dois horários.

Periodicamente, os nós da rede transmitem pacotes SYNC para seus vizinhos, permitindo que novos nós entrem na rede. O mecanismo de disputa pelo canal é feita através da troca de pacotes RTS-CTS. Depois que começam a transmissão de dados, os nós não seguem seus horários de sono até que a transmissão termine. O S-MAC requer sincronização periódica entre nós vizinhos. Apesar se a sincronização ser feita através de marcas de tempo relativos e não absolutos, a alta variância do atraso de propagação no canal submarino pode inviabilizar o uso desse protocolo em redes acústicas submarinas.

Além de coordenar o acesso ao meio compartilhado, o protocolo da camada de enlace deve disponibilizar uma funcionalidade de controle de erro para os dados transmitidos, uma vez que as características do canal submarino podem levar a taxas de erro de bit da ordem de 10^{-2} a 10^{-5} [2]. Técnicas de correção antecipada de erros FEC (do inglês *forward error correction*) devem ser empregadas em tal ambiente. A utilização de FEC tem como objetivo proteger os dados através da introdução de bits redundantes na transmissão de forma que o receptor possa detectar e corrigir os bits errados. Como os

recursos no canal submarino são escassos, há um compromisso entre a robustez da técnica FEC adotada (quantidade de bits redundantes) e a eficiência do canal.

3.2 Protocolos MAC para Redes Submarinas

Devido às limitações de energia e às características do canal submarino, o desenvolvimento de protocolos de controle de acesso ao meio (MAC) para redes acústicas submarinas é um grande desafio. Os protocolos MAC para canais submarinos foram desenvolvidos com base nos protocolos já existentes em redes de sensores terrestres. A importância na utilização de protocolos MAC se dá pela necessidade de evitar colisões no acesso ao canal, de forma a maximizar a eficiência da rede e minimizar o desperdício de energia. A seguir, será apresentada uma breve revisão bibliográfica sobre os principais protocolos de controle de acesso ao meio para redes submarinas encontrados na literatura.

Salva-Garau e Stojanovic et al. [14] propõem um protocolo para redes com veículos submarinos autônomos. O protocolo é baseado em uma rede organizada em *clusters*, dentro dos quais o esquema TDMA é usado. Tempos grandes de guarda são usados para superar a alta variância do atraso de propagação do canal submarino. Segundo os autores, o uso do TDMA não traz ineficiência à rede quando os veículos submarinos estão próximos uns dos outros dentro do mesmo cluster. Assim, o efeito do atraso de propagação variável não afeta tão dramaticamente o protocolo. Além do TDMA, o protocolo faz uso de códigos de espalhamento do sinal. Cada *cluster* possui seu próprio código de espalhamento, evitando assim interferência entre *clusters*. O protocolo proposto também possui mecanismos para lidar com a mobilidade do nó nos *clusters*.

O Slotted FAMA é uma modificação do protocolo FAMA original [15]. O FAMA exige que os pacotes de controle RTS e CTS sejam longos o suficiente para que não haja colisão, inviabilizando a aplicação do protocolo FAMA original em redes acústicas submarinas. O protocolo Slotted FAMA usa *slots* de tempo para limitar o impacto dos longos atrasos de propagação do canal acústico. Quando um nó possui um pacote para transmitir, ele espera até o início do próximo *slot* de tempo e transmite um pacote RTS, o qual é recebido por todos os nós em sua vizinhança. O nó de destino então envia um pacote de CTS no início do próximo *slot*, o qual também é recebido por todos os nós em sua vizinhança. Quando o nó de origem recebe o CTS, ele inicia sua transmissão de dados no início do próximo *slot*. Ao receber os dados, o nó de destino envia um pacote de reconhecimento no *slot*

subsequente ao término da transmissão de dados. No Slotted FAMA, bem como no FAMA original, os nós ficam monitorando o canal constantemente. Caso detectem uma portadora no canal, o nó entra em modo de recepção. Caso tenha um pacote para transmitir, o nó verifica se não há portadora no canal e então envia um pacote RTS e aguarda dois *slots* de tempo pelo pacote CTS. Caso o pacote CTS não seja recebido, o nó assume que ocorreu uma colisão e entra em estado de recuo aleatório, depois do qual tenta reenviar o RTS se nenhuma portadora for detectada no canal. Apesar de estarem cientes dos atrasos de propagação longos e com alta variância, os autores do Slotted FAMA não mencionam uma forma de implementar o sincronismo entre nós em redes acústicas submarinas.

O *Underwater MAC* (UW-MAC) é um protocolo de acesso ao meio proposto por Pompili et al. [11]. O UW-MAC utiliza o CDMA como método de acesso ao meio, e pode adaptar a potência de transmissão de forma a evitar interferência e economizar energia. Tais técnicas são adotadas visando minimizar o efeito “perto-longe” (*near-far*), que ocorre quando a comunicação com um nó distante fica prejudicada na presença da comunicação com um nó próximo em uma rede em que todos os nós apresentem a mesma potência de transmissão. O UW-MAC também faz a divisão da rede em *clusters* e os pacotes de controle são trocados através de um canal separado. Na fase de inicialização da rede, cada nó executa um algoritmo de agrupamento simples no qual ele inicia a descoberta de nós a fim de formar *clusters*.

Existem dois tipos de nós na rede. O primeiro tipo é chamado de “super-nó”. Este tipo de nó tem maior capacidade do que os demais nós da rede. Por possuir maior quantidade de baterias, este nó pode transmitir com maior potência de forma que sua transmissão alcance maiores distâncias. Esta maior robustez permite que este tipo de nó se comunique diretamente com a estação base na superfície da água. O segundo tipo de nó é chamado de “nó normal” e ocorre em maior quantidade na rede. Apesar de possuir maior potência de transmissão, quando um “super-nó” deseja se comunicar com um nó comum ele ajusta sua potência para não superar a potência de transmissão dos nós comuns. A potência só é aumentada quando o “super-nó” deseja se comunicar com a estação base. No período de inicialização os super-nós mediam a formação dos *clusters*. Em redes de sensores sem fios, os nós são muitas vezes implantados aleatoriamente e, portanto, pode existir nós da rede que não estão ligados a qualquer “super-nó”. Dessa forma, esse nó pode alcançar o super nó a partir de caminhos multisaltos. O protocolo UW-MAC utiliza ainda o método TDMA entre os “super-nós” para que eles possam comunicar-se com a estação base. É necessário, portanto, uma sincronização precisa entre os “super-nós”. Esta exigência pode inviabilizar seu uso no ambiente submarino devido às dificuldades de sincronização.

Syed e Heidemann et al. [17] propuseram o protocolo T–Lohi3, o qual é baseado em um esquema de contenção. Neste protocolo, os nós passam a maior parte do tempo monitorando o canal. Como a escuta ociosa exige baixo consumo de potência, o protocolo T–Lohi3 poupa energia dos nós. Outra forma de economia de energia implementada pelo protocolo é evitar as colisões de pacotes de dados através da alocação do canal. A alocação do canal é feita por períodos de disputa pelo meio. Durante esses períodos, os nós enviam tons curtos sinalizando que desejam usar o meio. Se o nó não receber outros tons disputando o canal durante o período de disputa, o nó ganhou o controle do canal e pode iniciar a transmissão. Se durante o período de disputa do meio houver mais tons de outros nós da rede que desejam obter o controle do canal, todos os nós desistem da transmissão e entram em estado de recuo aleatório definido de acordo com o número de tons recebidos durante aquele período. Acabado o período de recuo aleatório o nó tenta novamente obter o controle do meio. Após ganhar o controle do canal, o nó emite um tom de despertar a todos os nós. Através de um preâmbulo no tom, os nós que acordaram conseguem identificar se a mensagem está direcionada para ele. Caso seja verdade o nó entra em modo de recepção. Caso contrário, volta ao repouso. Para garantir a equidade do protocolo, um nó que acabou de efetuar uma transmissão não pode transmitir novamente durante um determinado período de tempo. Simulações promovidas pelos autores indicam uma sobrecarga de apenas 3 a 9% em relação à eficiência energética ótima (obtida com a transmissão de um pacote ponto a ponto, sem o protocolo) e uma taxa de utilização do canal, cerca de 30% da máxima teórica. [17]. Porém este protocolo não foi testado em ambientes submarinos reais. Dessa forma, os resultados obtidos podem diferenciar bastante quando o protocolo for testado no canal submarino, que apresenta variáveis que podem não ter sido consideradas na simulação.

O protocolo *UnderWater Acoustic Network MAC* (UWAN-MAC) faz uso do CSMA para alcançar eficiência energética [18]. Assim como o S-MAC mostrado na Seção 2.2, o protocolo UWAN-MAC reduz o tempo de monitoramento do canal, deixando o nó entrar em modo de hibernação, no qual o nó desliga seu rádio, e define um temporizador para acordar mais tarde. Isso gera economia de energia, pois o consumo de energia no modo hibernação é menor do que no modo de escuta ociosa do canal. Dessa forma, o nó passa a maior parte do tempo em hibernação seguidos de períodos de despertar curtos e coordenados. A cada despertar, o nó divulga para seus nós vizinhos o tempo que permanecerá em hibernação, de forma que os nós vizinhos possam se programar para que os despertares coincidam e a comunicação ocorra. O tempo de transmissão inicial é escolhido aleatoriamente e independentemente por cada nó. Porém, depois de escolhido o início do horário de transmissão, o nó irá seguir a sua programação e transmitirá em horários periódicos a partir da primeira transmissão. Assim, se o período entre as transmissões, ou seja, o período em que o nó permanece em modo de hibernação for muito maior do que a duração de uma

transmissão, a probabilidade de colisões será pequena. No período de inicialização, cada nó da rede transmite um pacote SYNC que contém a sua programação de hibernação e permanece acordado durante todo o ciclo para receber pacotes SYNC de seus vizinhos. Depois da fase de inicialização, cada nó saberá em que horário precisará estar acordado para receber os pacotes de seus vizinhos. O nó não entra em modo ocioso logo após uma transmissão. Antes disso, ele escuta o canal buscando receber pacotes SYNC de recém chegados à rede. Os nós podem ainda modificar o seu ciclo de transmissões para que possa se comunicar com nós vizinhos. Os autores do UWAN-MAC afirmam que apenas 3% da energia transmitida é desperdiçada em colisões, desde que cada nó possua pelo menos 5 nós vizinhos disponíveis a um salto. Porém, o atraso de propagação no ambiente submarino com alta variância pode afetar até mesmo esquemas de sincronização relativa. Esses esquemas são robustos a ambientes com atrasos de propagação altos, mas não com atrasos de propagação variáveis.

A situação peculiar do canal acústico submarino motivou a criação do protocolo *Delay-aware Opportunistic Transmission Scheduling* (DOTS) [19]. O alto atraso de propagação permite que vários pacotes propaguem-se simultaneamente no canal submarino. Esta condição pode ser explorada para melhorar o rendimento da rede. Um dos principais pressupostos do DOTS é a sincronização do relógio, para que os nós possam construir mapas locais de atraso de propagação e horários de transmissão esperados para os nós vizinhos através dos pacotes ouvidos. DOTS requer dois campos adicionais no cabeçalho MAC, ou seja, um *timestamp* preciso de relógio sincronizado de quando o quadro foi enviado e um estimativa do atraso de propagação entre a origem e o destino. Cada nó pode calcular o atraso de propagação de um vizinho para si mesmo, subtraindo o *timestamp* do quadro MAC a partir do momento da recepção do quadro MAC. O protocolo DOTS é baseado no protocolo MACA, visto na Seção 2.2, e utiliza pacotes RTS / CTS para acesso ao canal. Sempre que um nó tem um quadro para enviar, ele consulta seu banco de dados e decide se ele pode ou não transmitir. Se não forem detectados conflitos em transmissões correntes e em possíveis transmissões futuras, ele começa sua transmissão, caso contrário, recua para um período de tempo aleatório. Com esse esquema então, o nó pode começar uma transmissão mesmo que tenha acabado de receber um RTS referente a uma transmissão alheia a ele, desde que através dos *timestamp* ele verifique que todo o tráfego relativo à sua transmissão não irá interferir na transmissão corrente ou em possíveis transmissões futuras. DOTS promove dessa forma transmissão simultâneas que melhoram a vazão da rede. Porém como o DOTS exige uma sincronia da rede, este protocolo pode não apresentar bons desempenhos em ambientes submarinos que possuem atrasos de propagação elevados e com alta variância.

Dentre os protocolos propostos para redes submarinas referenciados, a maioria assume o sincronismo da rede. Buscando evitar essa complexidade busca-se aqui um protocolo simples, não dependente de sincronia e com uma boa vazão. O protocolo que se encaixou dentro dessas características foi o protocolo o MAC CW [1]. Implementado no modem comercial da WHOI, o protocolo MAC CW apresenta um bom desempenho em redes submarinas. O capítulo 4 procederá com o estudo detalhado desse protocolo, que serviu de base para o protocolo de transmissão confiável proposto.

Capítulo 4

O PROTOCOLO MAC CW

4.1 Introdução

O protocolo MAC CW foi proposto por Parrish et al. [1]. O protocolo MAC CW é baseado em um esquema simples de ALOHA com recuo aleatório. A escolha do ALOHA com recuo aleatório deveu-se à necessidade de se utilizar um mecanismo simples que evitasse colisões, uma vez que em ambientes submarinos a eficiência energética é um parâmetro bastante importante. Por sua simplicidade, o protocolo MAC CW pode ser implementado na maioria dos modems acústicos comerciais, atendendo dessa forma a uma grande variedade de aplicações que fazem uso de redes submarinas. O protocolo também busca evitar a complexidade de sincronização de tempo da rede, visto que o canal submarino apresenta atraso de propagação alto e com alta variância, como foi discutido no Capítulo 2.

O mecanismo de recuo aleatório é importante pelo fato de que o canal submarino é compartilhado e, portanto pode haver colisões na rede se múltiplos nós desejarem transmitir ao mesmo tempo. Quando isso ocorre, os quadros colidem e os demais nós da rede não conseguem decifrar nenhum dos quadros transmitidos. Em eventos assim, o canal é desperdiçado durante o período de colisão e os quadros colididos devem ser retransmitidos. Tal situação se agrava com o aumento da quantidade de nós ativos compartilhando o mesmo canal, podendo fazer com que a rede fique inoperante devido à quantidade de colisões. Assim, com a utilização do recuo aleatório, cada nó escolhe um valor de recuo aleatório independentemente. O valor escolhido para recuo aleatório é utilizado para inicializar um temporizador que funciona em contagem regressiva. Quando o temporizador termina a contagem regressiva o nó efetua a transmissão. Com isso, após uma colisão, é possível que o valor de recuo aleatório escolhido por um nó seja suficientemente mais curto do que os recuos aleatórios escolhidos pelos demais nós envolvidos na colisão. Isso possibilita que o nó retransmita seu pacote sem que haja uma colisão.

A Figura 11 mostra o diagrama de estados do protocolo MAC CW [1]. O protocolo possui um parâmetro chamado de CW, que define o tamanho da janela de contenção que

será usado no recuo aleatório. O valor CW representa o tamanho do intervalo dentro do qual o nó escolhe um valor aleatoriamente para inicializar o seu temporizador de recuo aleatório. O recuo aleatório no MAC CW funciona da mesma forma como explicado no parágrafo anterior. Assim, o nó escolhe aleatoriamente um valor de janela de contenção entre 0 e CW -1 todas as vezes que possui um pacote para ser transmitido. Em seguida o nó inicializa um temporizador com um valor escolhido aleatoriamente entre 0 e CW – 1 e começa a contagem regressiva. O temporizador é interrompido sempre que o nó detectar um pacote para recepção. Quando o temporizador termina a contagem regressiva, o nó transmite o pacote. Outro elemento do protocolo é o tempo de duração do slot σ, o qual é um parâmetro de implementação específica que deve ser longo o suficiente para permitir o acesso à informação de estado do canal. Simplificadamente, o tempo de duração do slot σ equivale a um atraso na notificação de uma detecção pelo filtro casado. No protocolo MAC CW, a duração do slot σ representa a discretização da janela de contenção. Dessa forma, o tamanho da janela de contenção fica limitada à múltiplos do valor do slot σ.

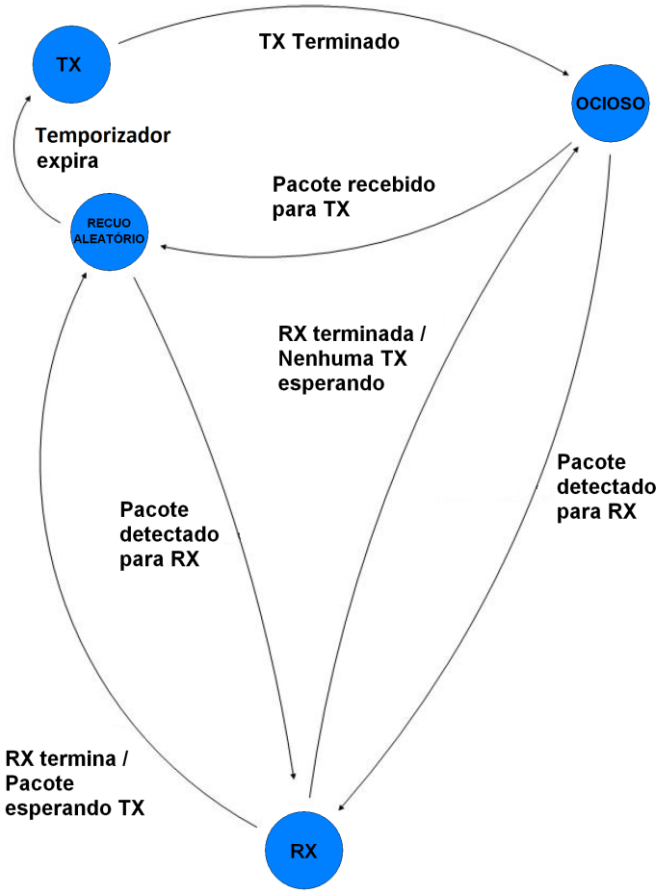


Figura 11 – Diagrama de estados do protocolo MAC CW. Adaptado de [1].

Como pode ser observado no diagrama de estados da Figura 11, o protocolo MAC CW não implementa o serviço de entrega confiável de dados. O nó no estado OCIOSO não possui nenhum pacote para ser transmitido e nenhum pacote para receber. Quando o nó detecta um pacote para recepção ele sai do estado OCIOSO e vai para o estado RX onde recebe o pacote. Após a recepção do pacote, se não houver chegado nenhum pacote para transmissão nesse intervalo, o nó volta para o estado OCIOSO. Se durante a recepção do pacote no estado RX houver chegado um pacote para a transmissão, o nó transita para o estado de RECUO ALEATÓRIO após a recepção do pacote. O nó ainda pode ir para o estado de RECUO ALEATÓRIO quando está no estado OCIOSO e um pacote chega para a transmissão. No estado de RECUO ALEATÓRIO, o nó escolhe aleatoriamente um valor de recuo aleatório entre 0 e CW – 1, e inicializa um temporizador com o valor escolhido. Se durante o tempo que passa pelo estado RECUO ALEATÓRIO, o nó detectar um pacote para recepção, ele pausa o temporizador e vai para o estado RX, onde prossegue com a recepção do pacote. Terminada a recepção do pacote o nó volta para o estado de RECUO ALEATÓRIO e retoma a contagem regressiva do temporizador. Quando o temporizador terminar a contagem regressiva, o nó passa para o estado TX onde efetivamente transmite o pacote de dados. Após a transmissão o nó volta para o estado OCIOSO.

4.2 Micromodem WHOI

A análise de desempenho do protocolo MAC CW foi feita utilizando o Micromodem da Woods Hole Oceanographic Institution (WHOI) [5]. O Micromodem no modo FH-FSK permite apenas dois tamanhos de pacotes: 21 bits e 32 bytes. Desta forma, se a informação não couber em um pacote de 21 bits deverá ser usado o pacote de 32 bytes. A Figura 12 mostra o Micromodem WHOI.

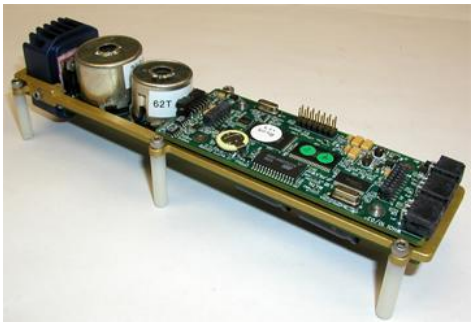


Figura 12 – Micromodem da WHOI. Fonte [5].

As potências utilizadas, no processo de transmissão e recepção de ondas eletromagnéticas, por modems em terra são semelhantes. Já comunicação acústica, a potência utilizada na transmissão pode chegar a dezenas de vezes a necessária para a recepção, a qual, por sua vez pode ser até 10 vezes maior que a de hibernação [8]. Além do mais, para transmissões entre dois pontos em terra a potência é muito inferior do que a potência utilizada em uma transmissão entre dois pontos submarinos separados pela mesma distância. Isso acontece porque o modem acústico utiliza vibrações mecânicas. A Tabela 2 reproduz a comparação entre o Aironet 350 (rádio da Cisco) e o Micromodem WHOI.

Tabela 2 – Comparação de potências (mW). Fonte [8].

Equipamento	TX	RX	Repouso	Hibernação
Aironet 350	2240	1350	1350	75
Micromodem	10000	3000	80	≈ 0

Como foi dito anteriormente, o protocolo MAC CW requer que os nós avaliem o estado do canal continuamente para definí-lo como ocioso ou ocupado. Porém, o Micromodem WHOI, não é equipado com uma capacidade separada de monitoramento de canal. Dessa forma, o nó pausa seu temporizador toda vez que o nó percebe um pacote para recepção. Assim, todos os pacotes adquiridos são recebidos na sua totalidade, o que efetivamente resulta em uma forma de avaliação do canal.

Diante dos princípios de funcionamento do protocolo MAC CW apresentados, é possível observar que esse protocolo não implementa a transferência confiável de dados. Observando novamente o diagrama de estados do MAC CW apresentado na Figura 11 nota-se que em nenhum estado foi implementado o uso de retransmissões e reconhecimentos, característica básica para protocolos que desejam fornecer um serviço confiável de entrega de dados. Cientes da importância de se adicionar confiabilidade nas transmissões, o Capítulo 5 apresenta uma proposta de protocolo de controle de acesso ao meio com transferência confiável para redes de submarinas.

Capítulo 5

PROPOSTA DE PROTOCOLO CONFIÁVEL

5.1 Introdução

O avanço de pesquisas sobre as redes submarinas tem o intuito de contornar as dificuldades impostas por esse canal tão adverso para as comunicações. Caracterizado pelos atrasos de propagação elevados e com alta variância, o canal submarino dificulta as comunicações possuindo banda limitada e uma baixa taxa de bits. Devido às limitações de energia e às características do canal submarino, o desenvolvimento de protocolos de controle de acesso ao meio (MAC) para redes acústicas submarinas é um grande desafio. Buscando evitar a complexidade de sincronização de tempo da rede submarina proposta em muitos protocolos, propõe-se aqui um protocolo de transferência confiável de dados baseado no MAC CW com recuo aleatório, que apresentou um bom desempenho no ambiente submarino.

O protocolo proposto visa fornecer serviço confiável de entrega de dados na camada de enlace utilizando reconhecimentos e retransmissões. O serviço confiável de entrega de dados na camada de enlace é de extrema importância em canais submarinos visto que a taxa de erro de bits nesse meio é imensamente superior à encontrada em redes terrestres. Sabendo das limitações do canal submarino e dos recursos escassos desperdiçados quando um erro ocorre, o protocolo fornece um serviço confiável de entrega de dados visando evitar uma retransmissão fim a fim dos dados por um protocolo da camada de transporte ou de aplicação.

A idéia central do protocolo é adicionar o envio de pacotes de reconhecimento para confirmar a recepção de um pacote de dados e de retransmissões caso a confirmação da recepção não ocorra. Acredita-se que a perda de vazão devido ao tráfego de pacotes de reconhecimento e às retransmissões na rede, seja compensada pela maior confiabilidade na entrega de dados.

O protocolo proposto também faz o monitoramento do canal antes de tentar transmitir um pacote. No protocolo MAC CW, o nó vai para o estado de recuo aleatório toda vez que possui um pacote para ser transmitido. Já no protocolo aqui proposto, quando um

nó recebe um pacote para a transmissão, o nó avalia o estado do canal como ocioso ou ocupado. Se o canal estiver ocupado, o nó entra em estado de recuo aleatório como no protocolo original. Mas se o canal é avaliado como ocioso, o nó já entra em modo de transmissão e envia o pacote sem passar pelo estado de recuo aleatório.

Em linhas gerais, o protocolo proposto funciona da seguinte maneira: a todo momento em que detectar um pacote para recepção e não estiver transmitindo, o nó entrará em modo de recepção. Após receber o pacote, caso identifique que o pacote é endereçado para ele e o pacote é do tipo dados, o nó transmite um pacote de reconhecimento para confirmar a recepção. Caso verifique que o pacote não é endereçado para ele ou o pacote recebido é um pacote de ACK endereçado a ele, o nó volta ao estado em que estava antes da recepção.

Quando um nó recebe um pacote para a transmissão o nó avalia o estado do canal como ocioso ou ocupado. Se o canal estiver ocioso, o nó entra em modo de transmissão e envia o pacote sem passar pelo estado de recuo aleatório. Se o canal é avaliado como ocupado o nó escolhe aleatoriamente um valor de janela de contenção entre 0 e CW -1 e entra em estado de recuo aleatório. Em seguida o nó define um temporizador com o valor da janela de contenção escolhido e começa a contagem regressiva. O temporizador é interrompido sempre que o nó detectar um pacote para recepção, ocasião na qual o nó pausa o temporizador e só volta a retomar a contagem regressiva depois da recepção do pacote. Quando o temporizador do recuo aleatório terminar a contagem regressiva, o nó transmite o pacote e incrementa o número de transmissões para este pacote. Depois de transmitir o pacote de dados, e incrementar o contador de transmissões, o nó verifica se o número de transmissões atingiu um determinado limite. Caso tenha chegado ao limite, o nó zera o contador de transmissões e volta a ficar ocioso. Caso contrário o nó espera por uma confirmação do nó receptor. Se a confirmação por parte do nó receptor não chegar dentro do período de espera conhecido como *time out ack* o nó retransmite o pacote de dados, incrementa o contador de transmissões e verifica novamente se o limite foi atingido. Esse ciclo se repete até que o número de retransmissões para o pacote chegue a um determinado limite definido para cada aplicação. Caso o pacote de reconhecimento chegue ou este limite de retransmissões seja atingido o nó volta a ficar ocioso e zera o contador de transmissões.

O diagrama de estados do modelo proposto é mostrado na Figura 13. A descrição detalhada dos estados é feita na Seção 5.2.

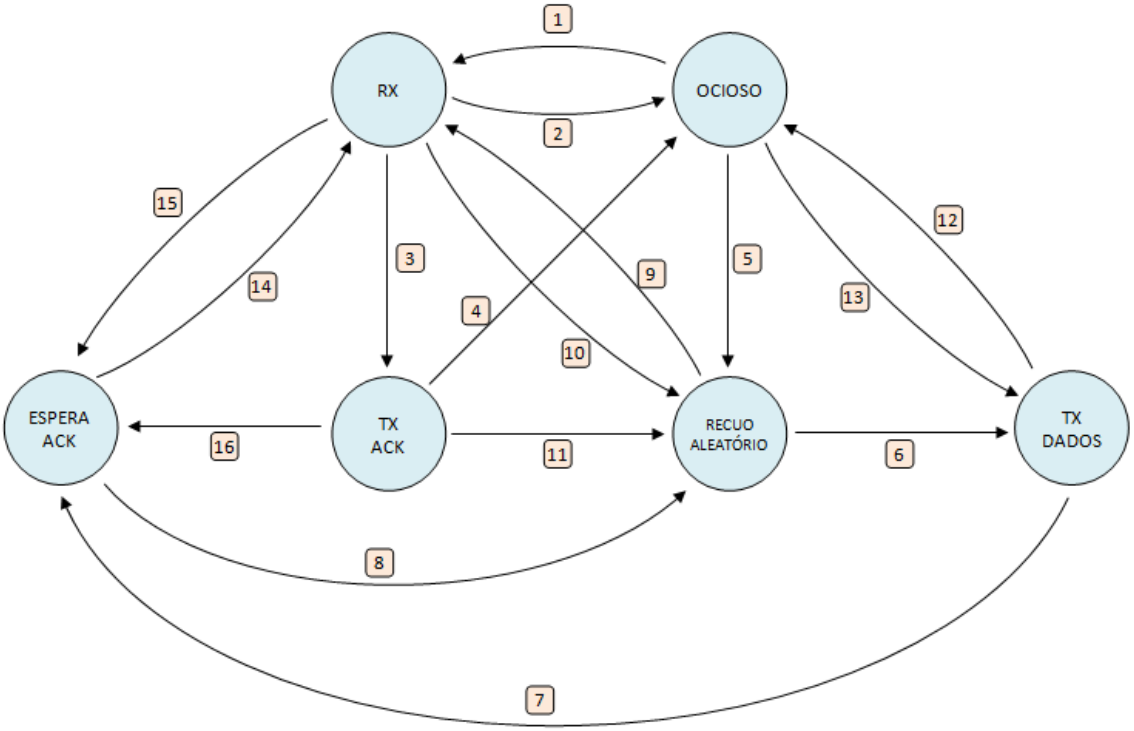


Figura 13 – Diagrama de estados do protocolo proposto.

5.2 Descrição dos Estados e Transições

A Figura 13 apresenta o diagrama de estados do protocolo proposto. O estado OCIOSO representa o estado em que o nó não está recebendo pacotes nem possui pacotes para transmissão. O estado TX DADOS representa a transmissão de um pacote de dados pelo nó. O estado RECUO ALEATÓRIO representa a contenção quando o canal é sentido ocupado e o nó possui um pacote para transmissão. A espera pelo pacote de reconhecimento é representada pelo estado ESPERA ACK. O estado RX representa a recepção de um pacote pelo nó. As transições entre os estados na Figura 13 são descritas a seguir.

A transição 1 do estado OCIOSO para o estado RX ocorre sempre que um pacote é detectado para a recepção. O nó sai do estado RX e volta para o estado OCIOSO através da transição 2 quando o nó acaba de receber um pacote no estado RX e percebe que o endereço do pacote recebido não é igual ao seu ou que o pacote recebido está endereçado para ele e o pacote é do tipo ACK; e além do mais não há pacote esperando para ser transmitido.

Quando o nó acaba de receber um pacote no estado RX e o endereço do pacote recebido é igual ao endereço do nó e o pacote é do tipo DADOS, o nó efetua a transição 3 e vai para o estado de TX ACK onde ele envia um pacote de reconhecimento para o pacote de dados que acabou de receber. Quando o nó está no estado TX ACK e a transmissão do pacote de reconhecimento foi terminada, a volta para o estado OCIOSO, através da transição 4 , se dá apenas se o nó não possuir nenhum pacote de dados esperando para ser transmitido e o nó não esteja esperando por pacotes de reconhecimento.

A transição 5 é feita quando o nó, estando no estado OCIOSO, recebe um pacote para a transmissão e sente o canal ocupado. Nesta ocasião o nó muda para o estado de RECUO ALEATÓRIO. Neste estado o nó escolhe um valor entre 0 e CW – 1 para inicializar seu temporizador de recuo aleatório. Quando o temporizador do estado de RECUO ALEATÓRIO terminar, significa que o nó deve enviar seu pacote de dados. Para isso o nó efetua a transição 6 e vai para o estado TX DADOS. Quando a transmissão do pacote de dados termina o nó incrementa o número de transmissões para este pacote e verifica se o número de transmissões referentes aquele pacote já atingiu o limite estabelecido. Caso o limite de transmissões para aquele pacote não tenha sido atingido, o nó sai do estado TX DADOS e, através da transição 7, passa para o estado ESPERA ACK e espera por um pacote de reconhecimento para o pacote transmitido. Caso o número máximo de transmissões para o pacote tenha sido atingido, o nó efetua a transição 12 e volta para o estado OCIOSO. No estado ESPERA ACK pode acontecer que um pacote seja detectado para a recepção. Nesta situação a transição 14 é realizada e o nó passa para o estado RX. Caso o nó esteja no estado RX e o pacote recebido não possuir endereço igual ao do nó e o nó esteja esperando por um pacote de ACK a transição 15 é efetuada e o nó volta ao estado ESPERA ACK.

Quando o nó está esperando por um pacote de reconhecimento no estado ESPERA ACK e ocorre o *time out ack*, o nó deve retransmitir o pacote. Para retransmitir o nó volta para o estado de RECUO ALEATÓRIO através da transição 8. Se o nó estiver no estado de RECUO ALEATÓRIO e nesse momento ele detectar um pacote para recepção, ele pausa o temporizador do recuo aleatório e efetua a transição 9 a fim de receber o pacote no estado RX. Terminada a recepção do pacote no estado RX e o nó perceber que o endereço do pacote recebido não é igual ao seu ou que o pacote recebido está endereçado para ele e o pacote é do tipo ACK; e além do mais há pacote esperando para ser transmitido, o nó efetua a transição 10 e volta para o estado de RECUO ALEATÓRIO onde retoma a contagem regressiva do temporizador.

A transição 11 só ocorre quando o nó está no estado TX ACK e após o termino da transmissão do pacote de reconhecimento existe um pacote de dados esperando para ser

transmitido. Nessa ocasião o nó retorna para o estado RECUO ALEATÓRIO e volta a rodar o temporizador.

Quando um nó está no estado OCIOSO e recebe um pacote de dados para transmissão ele pode transmitir o pacote diretamente sem passar pelo processo de recuo aleatório. Isto é feito através da transição 13 quando o canal é sentido ocioso.

Finalmente, a transição 16 só ocorre quando o nó está no estado TX ACK e, após a transmissão do pacote de reconhecimento, ele ainda está esperando por um pacote de reconhecimento, ocasião na qual o nó volta para o estado ESPERA ACK.

Comparando o diagrama de estados do protocolo proposto com o diagrama de estados original do protocolo MAC CW, pode-se notar o acréscimo de dois estados – o de transmissão de pacote de reconhecimento e o de espera pelo pacote de reconhecimento, representados na Figura 13 pelos estados TX ACK e ESPERA ACK, respectivamente. O número de possíveis transições entre estados aumentou de sete para dezesseis. Dessa forma a complexidade do protocolo aumentou consideravelmente, mas em compensação o protocolo é mais confiável do ponto de vista de entrega de dados.

O tempo de espera pelo pacote de reconhecimento (ACK) antes da retransmissão foi chamado de *time out ack*. O valor mínimo necessário para espera pelo pacote de reconhecimento (ACK) foi calculado levando-se em consideração a Figura 14.

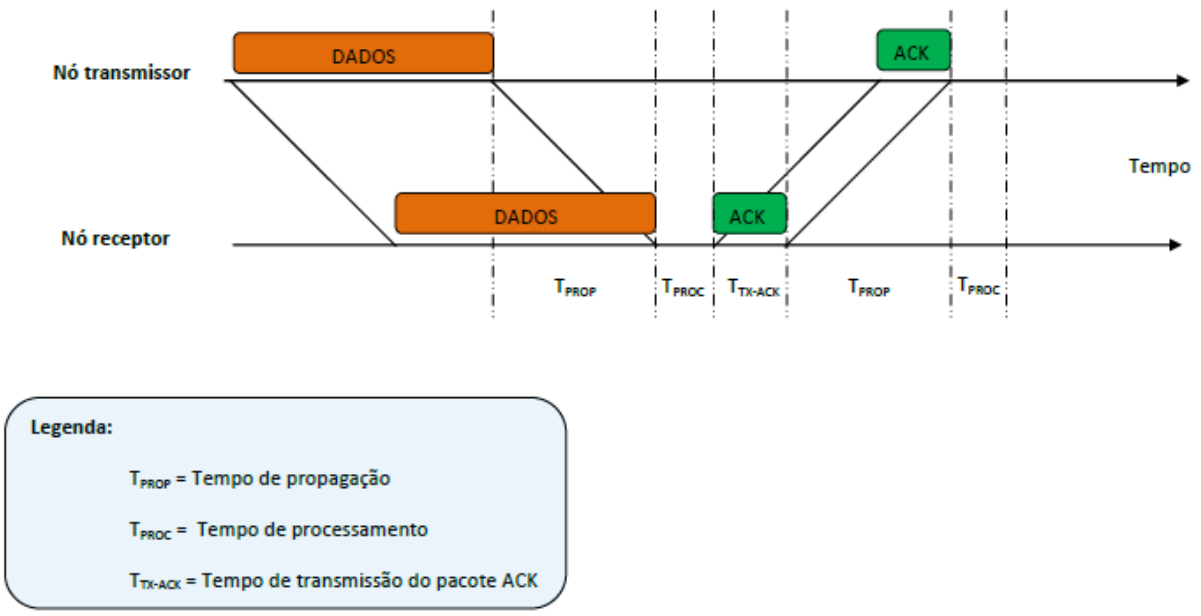
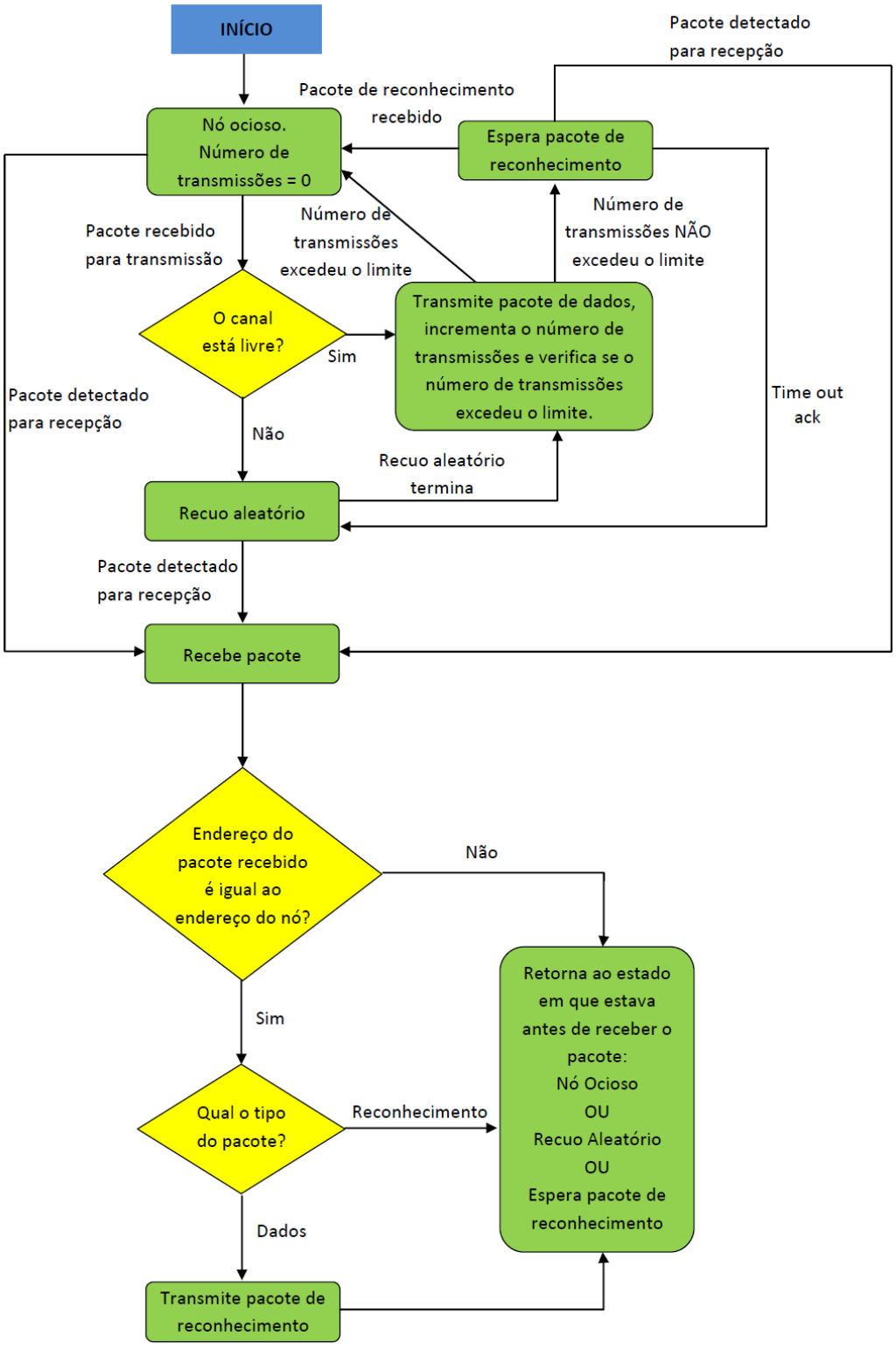


Figura 14 – Diagrama de tempos para cálculo do *time out ack*.

Dessa forma, de acordo com a figura acima o *time out ack* é dado por

$$T_{TIME\ OUT\ ACK} = 2 \cdot \left(T_{PROP} + T_{PROC}\right) + T_{TX-ACK}. \qquad (3)$$

O fluxograma do protocolo proposto é mostrado abaixo. Trata-se de uma forma mais simplificada de visualizar o protocolo em alternativa ao diagrama de estados apresentado na Figura 13.



Não preocupou-se aqui com o impacto das mudanças no consumo de energia dos nós submarinos. A eficiência energética é de fundamental importância em ambientes submarinos onde os nós sensores funcionam através de baterias. Mas, devido ao escopo do trabalho, deixamos esse estudo para investigações futuras.

Capítulo 6

INTRODUÇÃO AO NS-3

6.1 Visão Geral

A simulação do protocolo proposto foi feita no ambiente Network Simulator 3 – ns-3 [20]. O ns-3 é um simulador de rede a eventos discretos orientado principalmente para pesquisa e uso educacional [21]. O ns-3 é um software gratuito, distribuído sob a licença GNU GPLv2 e que é disponibilizado para investigação, desenvolvimento e utilização. O projeto ns-3 foi iniciado em 2006 e está em constante modificação.

É importante frisar aqui que o ns-3 é um novo simulador e não uma extensão do ns-2. Apesar de os dois simuladores serem escritos na linguagem de programação C++, o ns-3 é um novo simulador que não suporta as *application programming interface* (API) do ns-2. Muitos dos modelos desenvolvidos no ns-2 já foram importados para o ns-3.

O sítio principal do ns-3 está localizado em http://www.nsnam.org, onde se pode encontrar informações básicas sobre o funcionamento do simulador ns-3. Existe também a página http://www.nsnam.org/documents.html na qual há documentos relacionados com a arquitetura do sistema. Há ainda uma Wiki, que pode ser encontrada no endereço http://www.nsnam.org/wiki/, contendo informações para solução de problemas. O código-fonte pode ser encontrado e pesquisado em http://code.nsnam.org/, onde também encontra-se a árvore de desenvolvimento atual no repositório chamado ns-3-dev. Versões anteriores e repositórios experimentais do núcleo de desenvolvedores também podem ser encontradas lá.

Por ser um simulador complexo, o ns-3 precisa de algum mecanismo para gerenciar e organizar as alterações do código fonte. Para isso o ns-3 utiliza o Mecurial como sistema de gerenciamento de código fonte. Para a compilação é utilizado o Waf. Mais informações sobre o funcionamento desses softwares podem ser encontradas no Tutorial do ns-3 [20].

Normalmente o usuário do ns-3 utiliza o sistema operacional Linux para simular seus códigos fonte. Porém, os usuários que utilizam o sistema operacional Windows, da

Microsoft, podem utilizar o ns-3 através de um ambiente Linux emulado. Para estes usuários, os desenvolvedores do projeto ns-3 recomendam o ambiente Cygwin.

No ns-3 dispositivos de computação são chamados de nós, e são representados em C++ pela classe ns3::Node. A classe ns3::Node fornece métodos para gerenciar as representações de dispositivos de computação nas simulações [20]. Desta forma, em um nó é possível acrescentar aplicativos, pilhas de protocolos e dispositivos periféricos, como por exemplo, uma interface 802.11.

A classe ns3::Application representa, em C++, um programa de usuário que realiza alguma atividade a ser simulada. Esta classe fornece vários métodos para representar as funções da aplicação. Além do mais os desenvolvedores especializaram a classe ns3::Application de modo a criar outras classes como a ns3::OnOffApplication. Através dos parâmetros desta classe é possível testar uma rede em situações de saturação, utilizando tráfego CBR (*Constant Bit Rate*); ou ainda representar o funcionamento de uma rede em condições normais, utilizando tráfego poissoniano.

O canal de comunicação, no ns-3, é representado pela classe ns3::Channel. A classe ns3:: Channel também foi especializada pelos desenvolvedores do ns-3 e vários canais puderam ser representados, entres eles podemos citar: ns3::CsmaChannel, ns3::PointToPointChannel , ns3::WifiChannel e ns3::UanChannel.

Para permitir que os nós se comuniquem através do canal, foi criado no ns-3 a abstração de dispositivo de rede. Um nó pode se conectar em mais de um tipo de canal através de múltiplos dispositivos de rede. A abstração para dispositivos de rede no ns-3 é representada pela classe ns3::NetDevice. Esta classe oferece vários métodos para gerenciar as conexões do nó ao canal. Várias versões dessa classe foram implementadas para permitir o acesso aos diferentes tipos de canal. Desta forma a classe ns3::CsmaNetDevice foi projetada para se conectar no canal representado pela classe ns3::CsmaChannel; assim como a classe ns3::UanNetDevice foi projetada para se conectar no canal representado pela classe ns3::UanChannel.

O ns-3 fornece ainda classes chamadas de *topology helpers* que auxiliam a conexão das classes do tipo NetDevices às classes do tipo Nodes, bem como das classes do tipo NetDevices às classes do tipo Channels. Muitas operações seriam necessárias para criar um NetDevice, adicionar um endereço MAC, instalar esse dispositivo em um Node, configurar a pilha de protocolo do nó, e depois conectar o NetDevice ao Channel. Mais operações seriam necessárias para se conectar vários dispositivos em canais compartilhados [20]. Através dos *topology helpers* é possível combinar várias dessas operações para que se torne mais fácil essas conexões.

O ns-3 possui um módulo exclusivo para redes acústicas submarinas. Com esse módulo é possível modelar uma série de cenários nesse ambiente. O módulo UAN para o ns-3 inclui representações físicas de canal (PHY), além de modelos para protocolos MAC e para veículos autônomos submarinos. O código fonte do módulo UAN do ns-3 pode ser encontrado do diretório src/uan.

Com o avanço das pesquisas submarinas, o módulo UAN do ns-3 busca oferecer um ambiente de simulação completo para que os pesquisadores possam realizar experimentos, avaliações de desempenho e comparações. O módulo UAN, portanto, procura oferecer uma ferramenta confiável e realista do ambiente submarino. Para isso o módulo UAN oferece uma modelagem precisa do canal acústico submarino; um modelo para o modem acústico WHOI, além de alguns protocolos da camada de enlace.

Uma das áreas onde há mais pesquisas no ns-3 é na modelagem do canal acústico. Existe uma série de fatores que complicam uma modelagem precisa, tal como condições de contorno (superfície e terreno) dos oceanos, velocidade de propagação do som variável. Portanto, para uso em simulações em nível de rede é necessário realizar um compromisso entre complexidade do modelo utilizado na simulação e o tempo hábil para simulação. O modelo UAN para o ns-3 tenta ser o mais fiel possível ao ambiente submarino, dentro de limites de tempo viáveis computacionalmente.

Considerando o exposto, o módulo UAN possui três modelos de propagação para o canal acústico submarino: o modelo de propagação ideal, o modelo de propagação Thorp e o modelo de propagação Bellhop (disponível em um módulo separado).

Todos os modelos de propagação citados utilizam a classe ns3::UanPropModel para fornecer uma interface simples para expansão. O modelo de propagação ideal é representado no ns-3 pela classe ns3::UanPropModelIdeal, na qual não é assumida nenhuma perda de caminho ou atenuação no sinal transmitido.

O modelo de propagação Thorp é representado pela classe ns3:: UanPropModelThorp. Nesta classe, a atenuação Thorp é representada pela equação (1) apresentada na Seção 2.1 do Capítulo 2. A frequência utilizada no cálculo da atenuação Thorp, é a frequência central da modulação usada na classe ns3:: UanTxMode. A classe ns3::UanTxMode suporta alguns tipos de modulação, entre elas: PSK, QAM e FSK.

O modelo mais realista para a propagação de sinais no ambiente submarino é o modelo de propagação Bellhop que é representado pela classe ns3:: UanPropModelBh. Esta classe não está implementada na versão atual de desenvolvimento do ns-3 (versão 3.12), sendo disponível em um módulo separado. O modelo de propagação Bellhop lê

informações de propagação de um banco de dados. A classe ns3:: UanPropModelBh utiliza o software *Bellhop Acústico Ray Tracing* [3] para determinar as informações de propagação levando em consideração parâmetros do ambiente submarino tais como o perfil de velocidade do som, profundidade, velocidade do vento e as condições do terreno.

A principal classe do modelo PHY é a classe genérica ns3::UanPhyGen que é responsável pela manipulação de pacotes. Após a transmissão, é esta classe que decide se o pacote foi recebido com sucesso ou com erro. A classe ns3::UanPhyGen utiliza informação sobre a relação sinal-ruído (SNR) e sobre a taxa de erro de pacote para tomar a decisão de descarte de pacote. Esta classe também realiza o repasse dos pacotes recebidos com sucesso para as classes da camada MAC. A classe ns3::UanTransducer é responsável por rastrear todos os pacotes durante o evento de simulação.

O modelo padrão para o cálculo da taxa de erro de pacote é a classe ns3:.: UanPhyPerGenDefault. Esta classe testa a SNR do pacote recebido e compara com o limite de SNR definido previamente. Se a SNR do pacote estiver abaixo desse limiar o pacote é assumido com erro. Caso contrário, uma recepção bem sucedida é assumida. Já a classe ns3::UanPhyPerUmodem calcula a probabilidade de erro de pacote assumindo as características do Micromodem WHOI, que possui um código corretor de erro capaz de corrigir até um bit errado.

O modelo padrão para o cálculo de SINR no ns-3 é representado pela classe ns3:: UanPhyCalcSinrDefault, a qual assume que toda a energia transmitida é capturada no receptor e não existe interferência intersimbólica. Já a classe ns3::UanPhyCalcSinrFhFsk representa a SNR no Micromodem WHOI. O Micromodem WHOI operando no modo FH-FSK usa um padrão de salto predeterminado que é compartilhado por todos os nós.

Existe também a possibilidade de um nó possuir dois dispositivos transceptores. Esta situação é coberta através da classe ns3::UanPhyDual. A SNR nesse caso é calculada pela classe ns3::UanPhyCalcSinrDual que considera interferência apenas se os transceptores usarem a mesma frequência de transmissão.

O modelo UAN para ns-3 possui alguns protocolos MAC, entre eles está o protocolo ALOHA, que é representado pela classe ns3::UanMacAloha. Esta classe implementa o protocolo ALOHA tal como foi apresentado na Seção 2.3 do Capítulo 2.

Outro protocolo MAC submarino que foi implementado no ns-3 é o protocolo RC-MAC, representado pela classe ns3::UanMacRc. Este protocolo divide a banda disponível dinamicamente em dois canais: um para tráfego de dados e outro para sinalização. Este protocolo assume um nó gateway para o qual o trafego da rede é destinado. A classe

ns3::UanMacRc assume um único gateway. Também são usados pacotes RTS/CTS e o tempo é dividido em ciclos.

O protocolo MAC CW descrito no Capítulo 4 é representado no ns-3 através da classe ns3::UanMacCw. Esta classe representa o protocolo MAC CW diferentemente do modo que foi apresentado por Parrish et al. [1]. O protocolo MAC CW é representado no ns-3 como tendo um módulo de *carrier sense*.

No protocolo MAC CW o nó vai para o estado de recuo aleatório toda vez que possui um pacote para ser transmitido. Na classe ns3::UanMacCw, quando um nó recebe um pacote para a transmissão o nó avalia o estado do canal como estando ocioso ou ocupado. Se o canal estiver ocupado o nó entra em estado de recuo aleatório como no protocolo original. Mas se o canal é avaliado como ocioso o nó já entra em modo de transmissão e envia o pacote sem passar pelo estado de recuo aleatório.

O ns-3 fornece ainda classes para gerenciar modelos de mobilidade de veículos autônomos submarinos e gerenciamento energético.

6.2 Modificações de Códigos

Parrish et al. [1] desenvolveu novos modelos de propagação MAC, PHY no popular simulador de rede livre, ns-2. A documentação e o código estão disponíveis em http://ee.washington.edu/research/funlab/uan. Para cada par de nós na rede, desenvolveu-se um perfil de atraso de potencia para as condições ambientais específicas. Esta informação foi usada para calcular a relação sinal ruído (SNR) no receptor, que por sua vez, especifica a taxa de erro de pacote para o enlace FH-FSK, o que fornece uma abstração da camada de enlace em ns-2 para determinar se um pacote transmitido é recebido com êxito ou perdido.

Infelizmente muitos dos modelos MAC e PHY que foram desenvolvidas no ns-2 não foram importados para o ns-3. Além do mais outros modelos que foram importados foram modificados; tal como a própria classe que representa o protocolo MAC CW. A classe que representa o protocolo MAC CW diferentemente do modo que foi apresentado por Parrish et al. [1] é representada no ns-3 como tendo um módulo de *carrier sense*.

No protocolo MAC CW [1], o nó vai para o estado de recuo aleatório toda vez que possui um pacote para ser transmitido. No ns-3, na classe que representa o protocolo MAC CW quando um nó recebe um pacote para transmissão o nó avalia o estado do canal como estando ocioso ou ocupado. Se o canal estiver ocupado o nó entra em estado de recuo aleatório como no protocolo original. Mas se o canal é avaliado como ocioso o nó já entra em modo de transmissão e envia o pacote sem passar pelo estado de recuo aleatório.

Como explicado no Capítulo 5, foi incorporado esse módulo de *carrier sense* no protocolo aqui proposto, pois acredita-se que seja uma evolução do protocolo e também por já possuir o modelo pronto em ns-3 para teste.

Outro grupo de classes de suma importância que não foram importadas para o ns-3 foram as classes que representam o modelo de propagação BellHop. Essas classes juntamente com o programa *Bellhop's Gaussian Ray Tracing* [3] geram a propagação da onda acústica levando em consideração as características do canal. A frequência utilizada, as condições de superfície e do fundo dos oceanos, bem como o perfil de velocidade do som e a profundidade do canal acústico influenciam a resposta ao impulso do canal. O perfil de velocidade do som faz os raios acústicos "dobrarem" de acordo com a Lei de Snell. As perdas de energia acústica são influenciadas principalmente pelas condições de superfície e do fundo e a frequência utilizada na transmissão como foi visto no Capítulo 2. O modelo Bellhop permite que o usuário insira as propriedades acústicas da superfície inferior, a fim de modelar essa perda com base nos coeficientes de transmissão e reflexão.

Usou-se em nossas simulações a classe ns3::UanPropModelThorp que modela a energia acústica dissipada em forma de calor através da atenuação Thorp seletiva em frequência já explicada anteriormente na Seção 2.2.

A relação sinal ruído foi calculada utilizando a classe ns3::UanPhyCalcSinrFhFsk, a qual modela a interferência para o Micromodem WHOI que considera que todos os nós usam o mesmo padrão de salto na modulação FH-FSK.

A taxa de erro de pacote é calculada através da classe ns3::UanPhyPerUmodem Essa classe calcula a taxa de erro de pacote para o Micromodem WHOI, assumindo um decodificador viterbi e FEC capaz de corrigir 1 bit errado.

Capítulo 7

AVALIAÇÃO DE DESEMPENHO

7.1 Simulações e Resultados

As simulações aqui realizadas seguem o mesmo padrão das simulações feitas por Parrish et al. [1]. Nas simulações que se seguem, foi colocado um nó receptor no centro de uma área quadrada de simulação, e os nós geradores de tráfego foram distribuídos aleatoriamente nessa região a 70 metros de profundidade. A taxa de transmissão usada foi de 80 bits/s (a menor taxa de transmissão suportada pelo Micromodem WHOI). Devido às restrições impostas pelo Micromodem WHOI descritas na Seção 4.2, o comprimento do pacote de dados foi definido como 32 bytes e o comprimento do pacote de reconhecimento foi definido como 21 bits. As simulações foram executadas 7 vezes para estimativa de vazão. O comprimento de *slot* foi definido como 0,2 s. Cada simulação teve a duração de 30 minutos, na qual foram transmitidos em média 500 pacotes. Defiu-se aqui a vazão normalizada como sendo a razão entre a vazão medida e a capacidade do enlace, dada por 1 pacote transmitido a cada 3,2 segundos.

Foram considerados três tipos de áreas quadradas (200m x 200m, 500m x 500m, 1.000m x 1.000m). Dessa forma, utilizando a Figura 14 e a equação (2), foi possível calcular o valor de *time out ack* mínimo para cada área considerando o pior caso (o nó estando em um dos vértices da área quadrada).

O tempo de processamento (T_{PROC}) utilizado foi de 0,05 segundos, considerando o número de operações que o simulador executa para desencapsular o pacote recebido e passá-lo para a próxima camada de rede. Como a capacidade do enlace é de 80 bps, então o tempo de transmissão do pacote reconhecimento é de 0,2625 segundos. Considerando a velocidade de propagação do som no meio submarino como sendo de 1.500 m/s, temos que os tempos de propagação da onda acústica para as áreas de 200m x 200m, 500m x 500m e 1.000m x 1.000m são de 0,094 s, 0,236 s e 0,471 s, respectivamente, para o pior caso. Dessa forma os tempos mínimos para a espera do pacote reconhecimento são de:

$$
\begin{aligned}
T_{TIME\ OUT\ ACK}\left(200m\ \times\ 200m\right) &= 0{,}5505\ s \\
T_{TIME\ OUT\ ACK}\left(500m\ \times\ 500m\right) &= 0{,}8345\ s \\
T_{TIME\ OUT\ ACK}\left(1.000m\ \times\ 1.000m\right) &= 1{,}3045\ s
\end{aligned}
\tag{4}
$$

A primeira simulação foi realizada para encontrar o tamanho da janela de contenção ideal, através do qual se obtém a máxima vazão da rede. Na primeira simulação foi utilizado tráfego saturado em uma rede contendo 10, 15 e 20 nós, os quais foram colocados aleatoriamente em uma região quadrada de 500m x 500m para que se pudesse encontrar o melhor valor para a janela de contenção (CW). A Figura 15 mostra o posicionamento aleatório dos nós na área de simulação. O nó receptor é representado pela estrela azul no centro do gráfico e os demais nós geradores de tráfego são representados por círculos vermelhos.

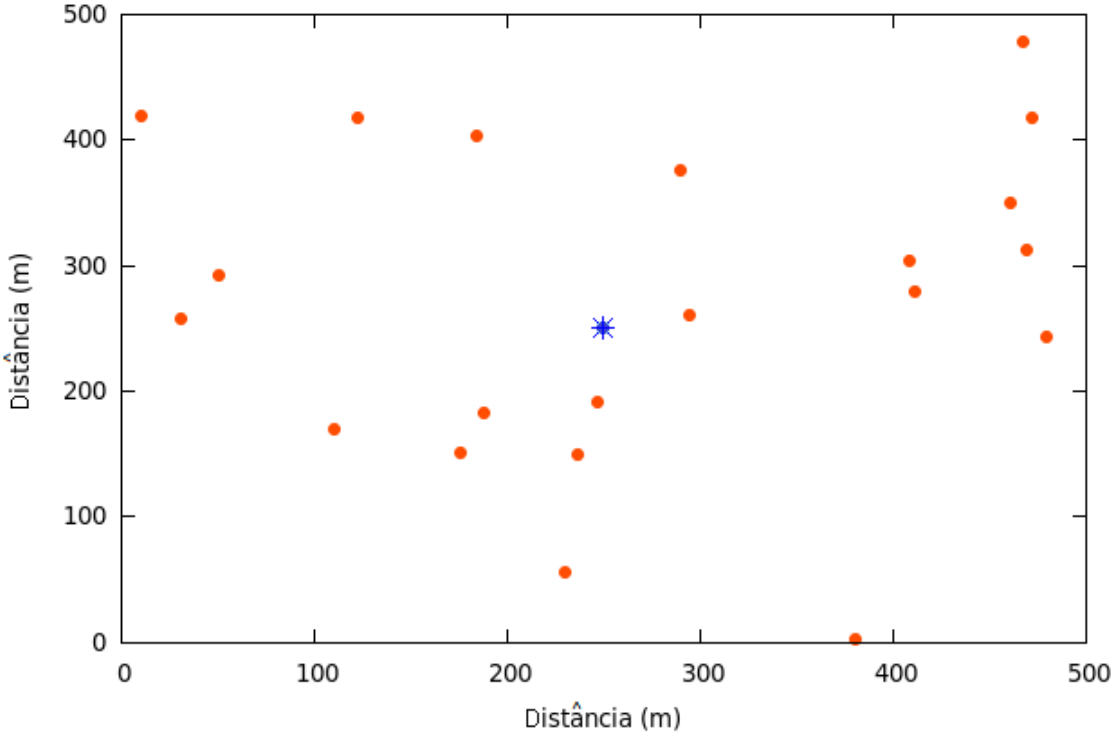


Figura 15 – Posicionamento aleatório dos nós para simulação em uma região quadrada de 500m x 500m. A estrela azul no centro da região representa o nó coletor, e os demais pontos vermelhos representam os nós geradores de trafego.

A Figura 16 mostra os resultados encontrados utilizando o protocolo MAC CW. A Figura 17 mostra os resultados obtidos com o protocolo proposto.

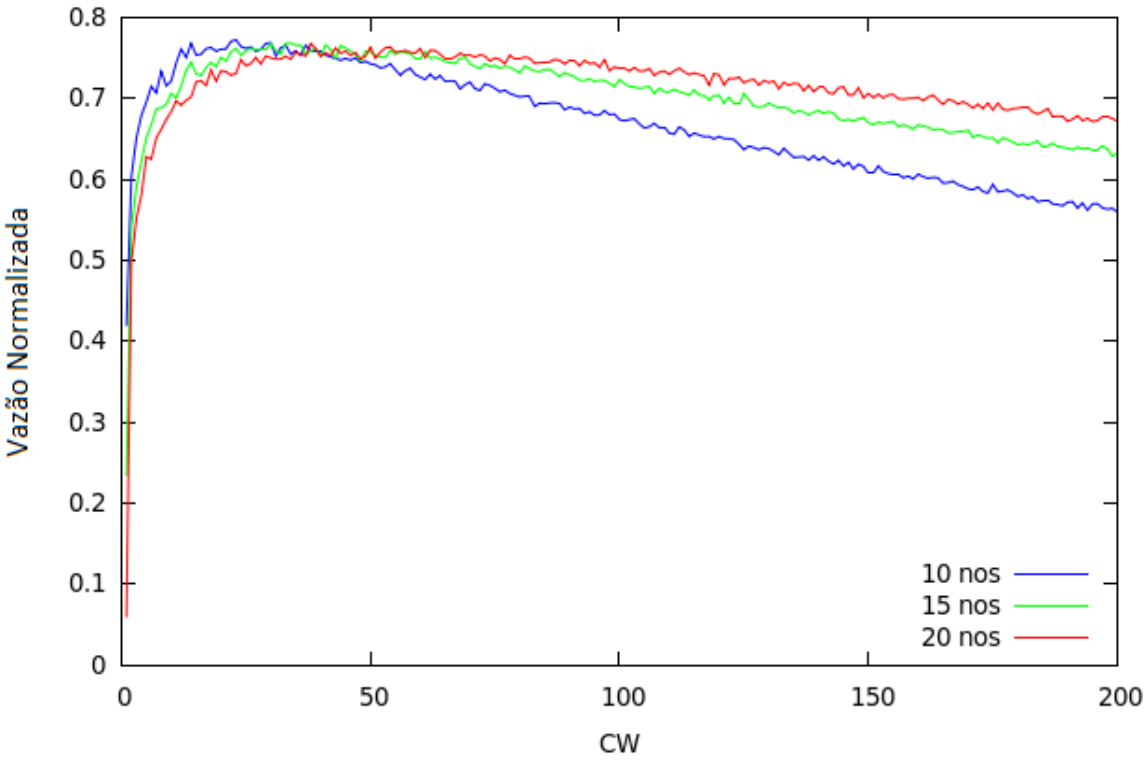


Figura 16 – Vazão Normalizada versus CW para 10, 15 e 20 nós implantados em uma região de 500m x 500m utilizando o protocolo MAC CW original e com tráfego saturado oferecido à rede.

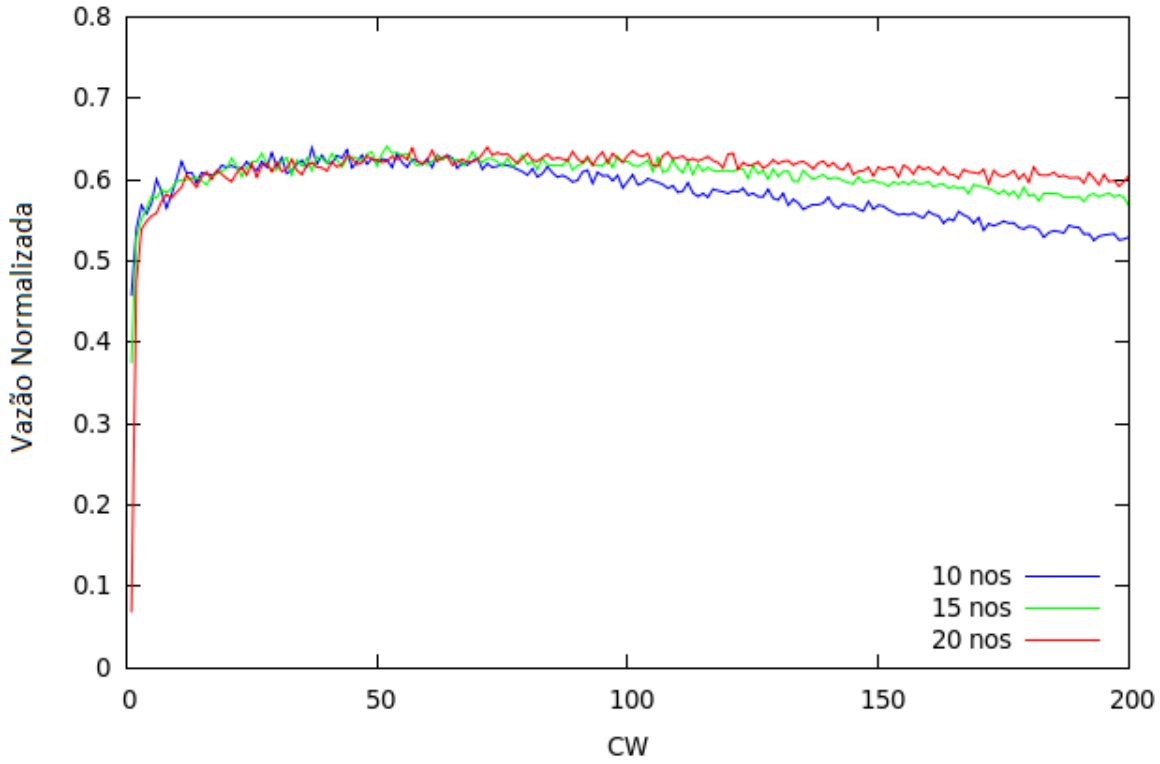


Figura 17 – Vazão Normalizada versus CW para 10, 15 e 20 nós implantados em uma região de 500m x 500m utilizando o protocolo proposto e com tráfego saturado oferecido à rede.

Comparando-se as Figuras 16 e 17 é possível notar que utilizando o protocolo proposto, a vazão média da rede reduz em torno de 17% em relação à vazão alcançada pelo protocolo MAC CW original. Além do mais o protocolo proposto apresenta uma menor variação relativa da vazão para densidades de nós diferentes na rede. Isso representa a robustez do protocolo, visto que um erro na definição do tamanho da janela de contenção ideal, não levaria a uma grande perda de vazão.

Através da Figura 16, pode-se encontrar o tamanho da janela de contenção ideal, através do qual se obtém a máxima vazão da rede. Dessa forma, utilizando o protocolo MAC CW, os valores ótimos de CW encontrados foram de CW = 23, CW = 34 , e CW = 38 para a rede contendo 10, 15 e 20 nós, respectivamente. Através da Figura 17, os valores ótimos de CW encontrados foram de CW = 44, CW = 52 e CW =72 para a rede contendo 10, 15 e 20 nós, respectivamente, utilizando o protocolo proposto.

O segundo conjunto de simulação deseja mostrar que a transferência máxima de dados é independente do número de nós na rede quando o tamanho da janela de contenção (CW) está definido corretamente para o número de nós e dimensões da rede. Os valores de CW ótimos encontrados através das Figuras 16 e 17 foram usados na segunda simulação, na qual a rede contendo 10, 15 e 20 nós foi implantada em uma área de 500m x 500m adotando o tráfego de Poisson. As Figuras 18 e 19 mostram os resultados encontrados para a vazão normalizada versus carga oferecida à rede com a utilização do MAC CW e do protocolo proposto, respectivamente.

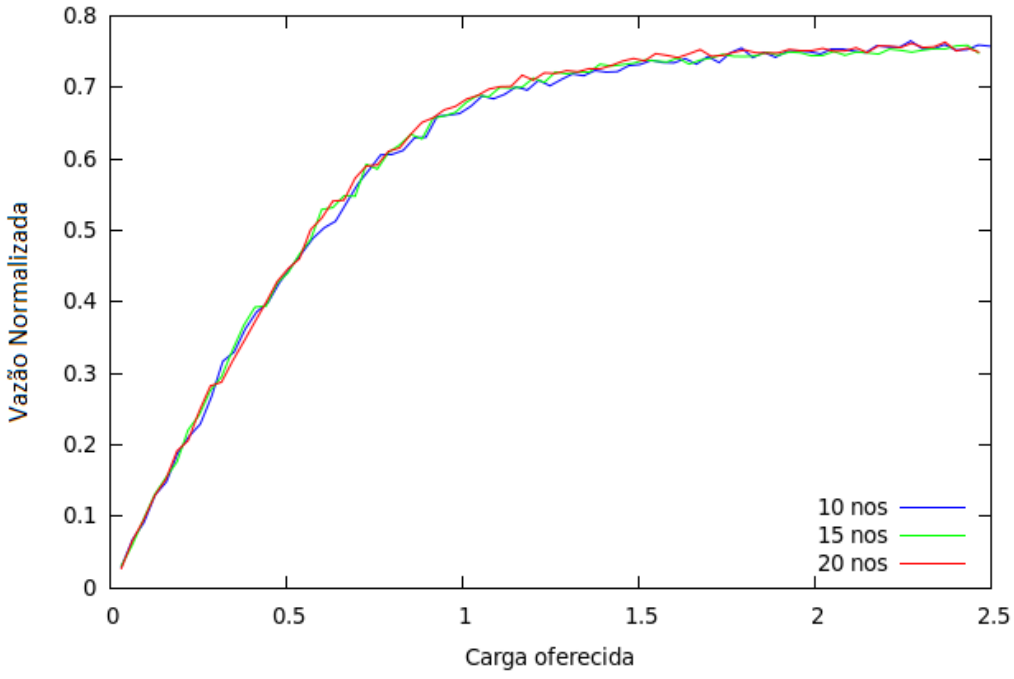


Figura 18 – Vazão Normalizada versus carga oferecida para 10, 15 e 20 nós implantados em uma região de 500m x 500m com tráfego poissoniano e utilizando o protocolo MAC CW original. O tamanho de CW foi definido como o valor encontrado para a maior vazão na Figura 16.

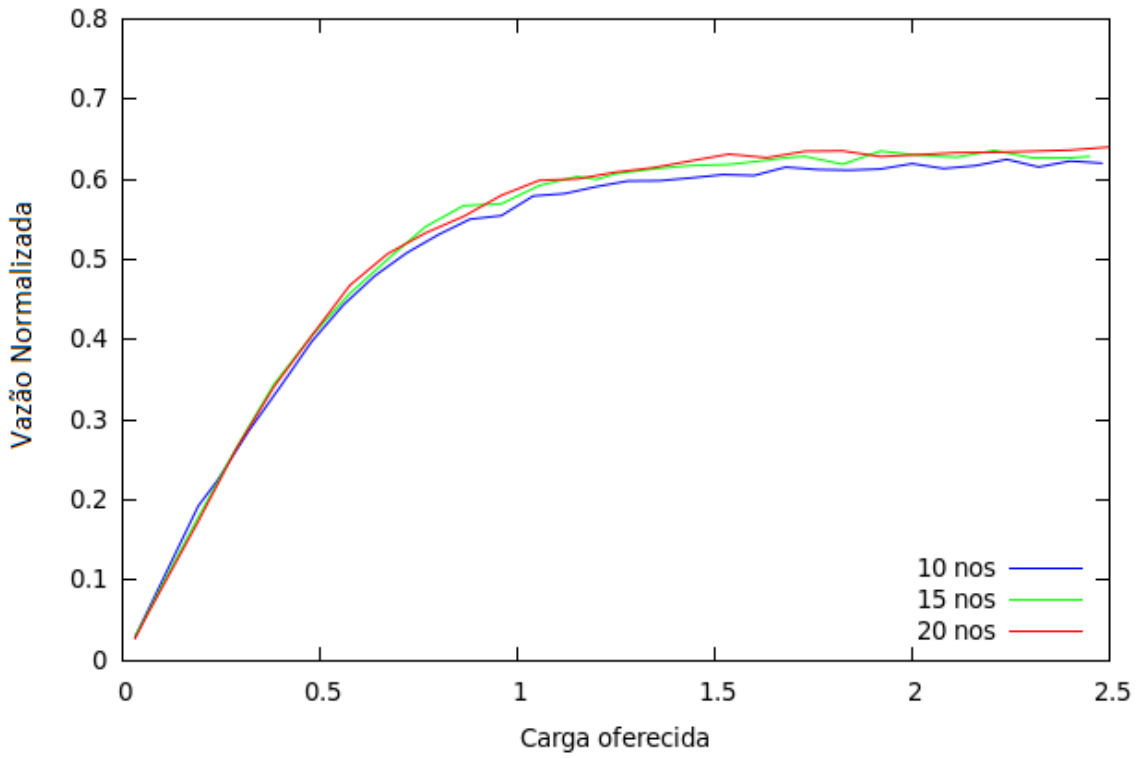


Figura 19 – Vazão Normalizada versus carga oferecida para 10, 15 e 20 nós implantados em uma região de 500m x 500m com tráfego poissoniano e utilizando o protocolo proposto. O tamanho de CW foi definido como o valor encontrado para a maior vazão na Figura 17.

Como pode ser observado nas Figuras 18 e 19, a transferência máxima de dados é independente do número de nós na rede quando o tamanho da janela de contenção (CW) está definido corretamente para o número de nós e dimensões da rede. A vazão média da rede utilizando o protocolo proposto apresentou a mesma redução (aproximadamente 17% em relação à vazão alcançada pelo protocolo MAC CW original).

O terceiro conjunto de simulações foi realizado para investigar o efeito das dimensões do terreno na vazão alcançável. Assim foram executadas simulações utilizando 15 nós em regiões quadradas de 200m x 200m, 500m x 500m e 1.000m x 1.000m. Do mesmo modo feito no primeiro conjunto de simulações, foram encontrados os valores ideais para a janela de contenção como sendo CW = 29, CW = 34 , e CW = 56 para as regiões quadradas de 200m x 200m, 500m x 500m e 1.000m x 1.000m, respectivamente, utilizando o protocolo MAC CW; e CW = 40, CW = 49 , e CW = 68 para as regiões quadradas de 200m x 200m, 500m x 500m e 1.000m x 1.000m, respectivamente, utilizando o protocolo proposto. Os valores encontrados de tamanho de janela ideal foram utilizados na simulação. As Figuras 20 e 21 mostram o gráfico da vazão normalizada versus carga oferecida à rede com a utilização do MAC CW e do protocolo proposto, respectivamente.

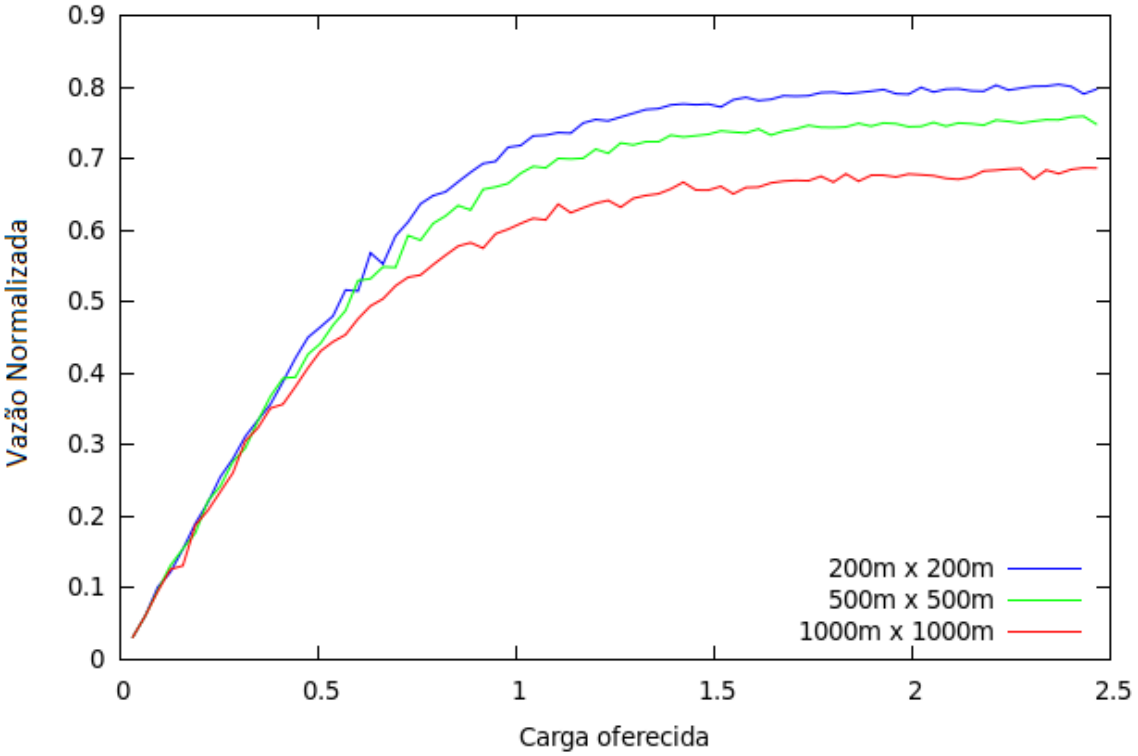


Figura 20 – Vazão Normalizada versus carga oferecida para uma rede de 15 nós implantados em uma região de 200m x 200m, 500m x 500m e 1.000m x 1.000m com tráfego poissoniano e utilizando o protocolo MAC CW original.

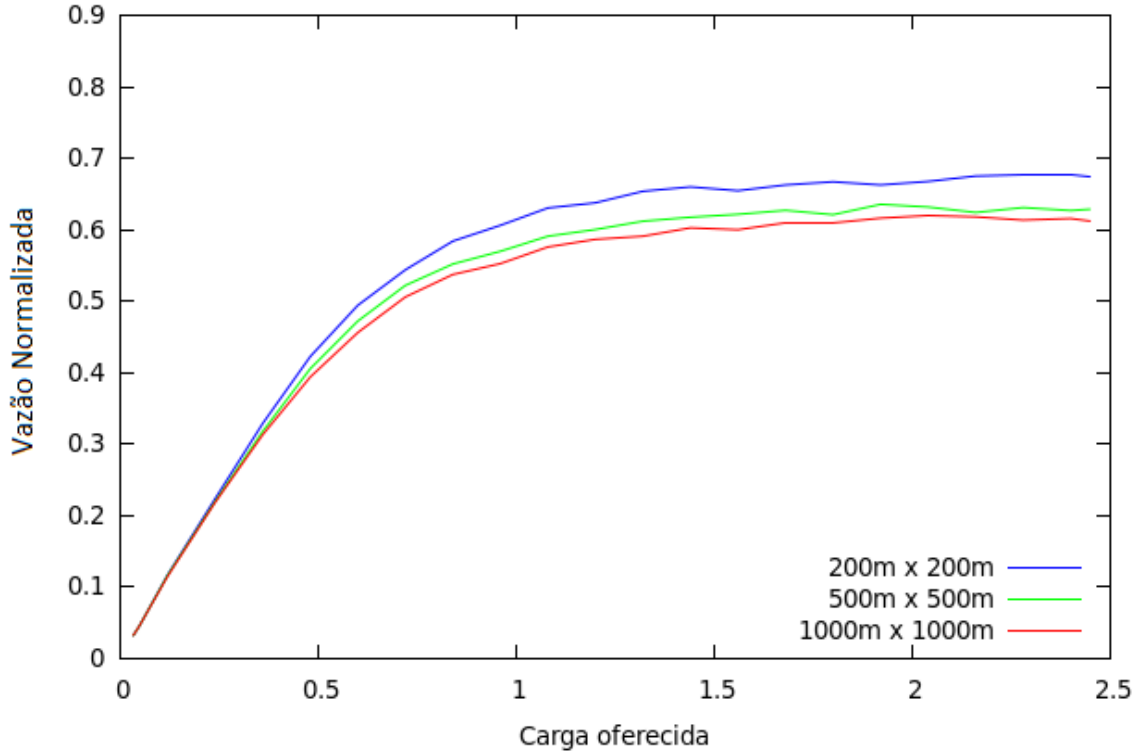


Figura 21 – Vazão Normalizada versus carga oferecida para uma rede de 15 nós implantados em uma região de 200m x 200m, 500m x 500m e 1.000m x 1.000m com tráfego poissoniano e utilizando o protocolo proposto.

Como pode ser visualizado nas Figuras 20 e 21, a vazão atingível é prejudicado pelo aumento no tamanho da rede. No entanto, parece que nosso protocolo, apesar de apresentar perda de vazão, apresenta uma variação de vazão relativa menor ao aumentarmos as dimensões do terreno. O protocolo MAC CW por outro lado, é mais sensível às variações de dimensão do terreno. O protocolo MAC CW apresenta uma variação relativa da vazão em torno de 0.13 entre o tamanho de terreno mínimo (200m x 200m) e o máximo (1.000m x 1.000m). Já no protocolo proposto essa variação relativa da vazão reduz para em torno de 0.07. Isso mostra a robustez do protocolo.

Capítulo 8

CONCLUSÃO

O recente crescimento de aplicações para redes de sensores submarinos motivou esse trabalho. Explorar uma ambiente tão diferente do que se está acostumado em meio terrestre foi um desafio para os autores desse trabalho. Lidar com tantas características adversas para o desempenho de uma rede nos fez os autores explorarem e preocuparem-se com cada detalhe do projeto, proporcionando aplicar uma ampla gama de conhecimentos adquiridos no decorrer desses anos.

Este trabalho apresentou a proposta de um protocolo de controle de acesso ao meio com transferência confiável de dados para redes submarinas. Sabendo das limitações do canal submarino e dos recursos escassos desperdiçados quando um erro ocorre, o protocolo fornece um serviço confiável de entrega de dados visando evitar uma retransmissão fim a fim dos dados por um protocolo da camada de transporte ou de aplicação. A análise de desempenho para o protocolo proposto mostrou que a vazão global da rede reduziu em torno de 17% em relação à vazão alcançada pelo MAC CW original. Além do mais, o protocolo proposto apresentou bastante robustez às variações do terreno. O protocolo MAC CW apresentou, em simulação, uma variação relativa da vazão em torno de 0.13 entre o tamanho de terreno mínimo (200m x 200m) e o máximo (1.000m x 1.000m); resultado superior à variação relativa da vazão encontrada para o protocolo proposto, em torno de 0.07. Outro resultado importante, obtido em simulações, foi que o protocolo proposto apresenta uma menor variação relativa da vazão para diferentes densidades de nós na rede. Isso ressalta a robustez do protocolo, visto que um erro na definição do tamanho da janela de contenção ideal, não levaria a uma grande perda de vazão. Em linhas gerais, o protocolo proposto obteve um bom desempenho na avaliação de desempenho, principalmente se for levado em consideração que o protocolo MAC CW não implementa o serviço confiável de entrega de dados.

Em trabalhos futuros deverá ser investigada a eficiência energética do protocolo proposto, uma vez que o consumo de energia é um limitante para o projeto de protocolos em ambientes submarinos onde os nós sensores funcionam através de baterias. Outro aspecto a ser destacado deve ser a adaptação em tempo real do tamanho da janela de contenção ideal, para que se possa sempre obter a vazão máxima de uma rede em que o número de nós e as dimensões são variáveis. Faz-se necessário também, a realização de avaliação de desempenho utilizando modelos mais precisos do canal submarino, tal como o

BellHop. Avaliação de desempenho para o caso em que todos os nós transmitem para qualquer nó e não apenas para o coletor deve também ser investigada.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Parrish N.; Tracy L.; Roy S. Arabshahi P.; e Fox, W., System Design Considerations for Undersea Networks: Link and Multiple Access Protocols , IEEE Journal on Selected Areas in Communications (JSAC), Special Issue on Underwater Wireless Communications and Networks, Dec. 2008.

[2] I. F. Akyildiz, D. Pompili, and T. Melodia, "Underwater Acoustic Sensor Networks: Research Challenges," Ad Hoc Networks (Elsevier), vol. 3, no. 3, pp. 257-279, March 2005.

[3] Ocean Acoustic Library. Acoustic Toolbox. Disponível em: http:// oalib.hlsresearch.com/Modes/AcousticsToolbox/. Acesso em 11 de setembro de 2011.

[4] Porter M.B. and Bucker H.P. Gaussian beam tracing for computing ocean acoustic fields. J. Acoust. Soc. America.

[5] Woods Hole Oceanographic Institution, Massachusetts, USA. http://www.whoi.edu/.

[6] W. Simpson (ed.) “The Point-to-Point Protocol (PPP)”, RFC 1661, jul. 1994. http://www.rfc-editor.org/rfc/rfc1661.txt.

[7] R. M. Metcalfe and D. R. Boggs, "Ethernet : Distributed Packet Switching for Local Computer Networks , " Comm. ACM, 19:7, July 1976, pp. 395-404.

[8] Penteado, Diorgenes. Redes Acústicas Subaquáticas na Monitoração de Correntes Marítimas. Rio de Janeiro: UFRJ/COPPE, 2010. Disponível em: http://www.gta.ufrj.br/ftp/gta/TechReports/Diorgenes10.pdf. Acesso em 23 de novembro de 2011.

[9] Simon S. Lam, ``A Carrier Sense Multiple Access Protocol for Local Networks,'' Computer Networks, Vol. 4, No. 1, January 1980.

[10] Phil Karn, "MACA - A New Channel Access Method for Packet Radio", Proceedings of the 9th ARRL/CRRL Amateur Radio Computer Networking Conference, London Ontario Canada, September 22, 1990, p. 134.

[11] Vaduvur Bharghavan, Alan Demers, Scott Shenker, and Lixia Zhang. MACAW: A Media Access Protocol for Wireless LAN's. In Proceedings of the ACM SIGCOMM '94 Conference, pages 212-225. ACM, August 1994.

[12] Fullmer, C. L., Garcia-Luna-Aceves, J. J. “Floor acquisition multiple access (FAMA) for packet-radio networks”. In: SIGCOMM ’95: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication, pp. 262–273, New York, NY, USA, 1995.

[13] Wei Ye, John Heidemann, and Deborah Estrin. An Energy-Efficient MAC protocol for Wireless Sensor Networks. In Proceedings of the IEEE Infocom, pp. 1567-1576. New York, NY, USA, USC/Information Sciences Institute, IEEE. June, 2002.

[14] F. Salva-Garau, M. Stojanovic, Multi-cluster protocol for ad hoc mobile underwater acoustic networks, in: Proceedings of IEEE OCEANS_03, San Francisco, CA.

[15] Molins, M., Stojanovic, M. “Slotted FAMA: a MAC Protocol for Underwater Acoustic Networks”, OCEANS 2006 - Asia Pacific, pp. 1–7, May 2006.

[16] Pompili, D., Melodia, T., Akyldiz, I. F. “A Distributed CDMA Medium Access Control in Underwater Acoustic Sensor Networks”. In: Proceedings of Med-Hoc-Net, Corfu, Greece, june 2007.

[17] Syed, A., Ye, W., Heidemann, J. “T-Lohi: A New Class of MAC Protocols for Underwater Acoustic Sensor Networks”, INFOCOM 2008. The 27th Conference on Computer Communications. IEEE, pp. 231–235, April 2008.

[18] Park, M. K., Rodoplu, V. “UWAN-MAC: An Energy-Efficient MAC Protocol for Underwater Acoustic Wireless Sensor Networks”, Oceanic Engineering, IEEE Journal of, v. 32, n. 3, pp. 710–720, July 2007.

[19] Youngtae Noh, Paul Wang, Uichin Lee, Dustin Torres, Mario Gerla. "DOTS: A Propagation Delay-aware Opportunistic MAC Protocol for Underwater Sensor Networks" IEEE ICNP'10, Kyoto, Japan, Oct. 2010 .

[20] NS-3 Developers, Reference Manual. Disponível em: http://www.nsnam.org/tutorias.html. Acessado em: Setembro de 2011.

[21] NS-3 Developers, Tutorial. Disponível em: http://www.nsnam.org/tutorias.html. Acessado em: Setembro de 2011.

APÊNDICE

Códigos de Simulação

I Código do Protocolo Proposto

```
#include "uan-mac-cw.h"
#include "ns3/attribute.h"
#include "ns3/uinteger.h"
#include "ns3/double.h"
#include "ns3/nstime.h"
#include "ns3/random-variable.h"
#include "ns3/uan-header-common.h"
#include "ns3/trace-source-accessor.h"
#include "ns3/log.h"

NS_LOG_COMPONENT_DEFINE ("UanMacCw");

namespace ns3 {
NS_OBJECT_ENSURE_REGISTERED (UanMacCw);

UanMacCw::UanMacCw ()
  : UanMac (),
    m_phy (0),
    m_pktTx (0),
    m_state (IDLE),
    m_cleared (false)
{
wait_ack = false;
retransmissao = false;
//LogComponentEnable ("UanMacCw", LOG_LEVEL_ALL);
}
UanMacCw::~UanMacCw ()
{
}

void
UanMacCw::Clear ()
{
  if (m_cleared)
    {
      return;
    }
  m_cleared = true;
  m_pktTx = 0;
  m_pktTx1 = 0;
  wait_ack = false;
```

```
  retransmissao = false;

  if (m_phy)
    {
      m_phy->Clear ();
      m_phy = 0;
    }
  m_sendEvent.Cancel ();
  m_txEndEvent.Cancel ();
  m_sendEventR.Cancel ();
}
void
UanMacCw::DoDispose ()
{
  Clear ();
  UanMac::DoDispose ();
}
TypeId
UanMacCw::GetTypeId (void)
{
  static TypeId tid = TypeId ("ns3::UanMacCw")
    .SetParent<Object> ()
    .AddConstructor<UanMacCw> ()
    .AddAttribute ("CW",
                   "The MAC parameter CW",
                   UintegerValue (10),
                   MakeUintegerAccessor (&UanMacCw::m_cw),
                   MakeUintegerChecker<uint32_t> ())
    .AddAttribute ("SlotTime",
                   "Time slot duration for MAC backoff",
                   TimeValue (MilliSeconds (20)),
                   MakeTimeAccessor (&UanMacCw::m_slotTime),
                   MakeTimeChecker ())
    .AddTraceSource ("Enqueue",
                     "A packet arrived at the MAC for transmission",
                     MakeTraceSourceAccessor (&UanMacCw::m_enqueueLogger))
    .AddTraceSource ("Dequeue",
                     "A was passed down to the PHY from the MAC",
                     MakeTraceSourceAccessor (&UanMacCw::m_dequeueLogger))
    .AddTraceSource ("RX",
                     "A packet was destined for this MAC and was received",
                     MakeTraceSourceAccessor (&UanMacCw::m_rxLogger)) ;
  return tid;
}
Address
UanMacCw::GetAddress ()
{
  return this->m_address;
}
void
UanMacCw::SetAddress (UanAddress addr)
{
  m_address = addr;
}
void
UanMacCw::SetForwardUpCb (Callback<void, Ptr<Packet>, const UanAddress&> cb)
{
```

```
  m_forwardUpCb = cb;
}
void
UanMacCw::AttachPhy (Ptr<UanPhy> phy)
{
  m_phy = phy;
  m_phy->SetReceiveOkCallback (MakeCallback (&UanMacCw::PhyRxPacketGood, this));
  m_phy->SetReceiveErrorCallback (MakeCallback (&UanMacCw::PhyRxPacketError, this));
  m_phy->RegisterListener (this);
}

Address
UanMacCw::GetBroadcast (void) const
{
  return UanAddress::GetBroadcast ();
}
void
UanMacCw::NotifyRxStart (void)
{
  if (m_state == RUNNING)
    {
      NS_LOG_DEBUG ("Time " << Simulator::Now ().GetSeconds () << " Addr " <<
GetAddress () << ": Switching to channel busy");
      SaveTimer ();
      m_state = CCABUSY;
    }
   if (wait_ack)
    {
      SaveTimer ();
    }
}
void
UanMacCw::NotifyRxEndOk (void)
{
  if (m_state == CCABUSY && !m_phy->IsStateCcaBusy ())
    {
      NS_LOG_DEBUG ("Time " << Simulator::Now ().GetSeconds () << " Addr " <<
GetAddress () << ": Switching to channel idle");
      m_state = RUNNING;
      StartTimer ();
    }
   if (wait_ack)
    {
      StartTimer ();
    }
}
void
UanMacCw::NotifyRxEndError (void)
{
  if (m_state == CCABUSY && !m_phy->IsStateCcaBusy ())
    {
      NS_LOG_DEBUG ("Time " << Simulator::Now ().GetSeconds () << " Addr " <<
GetAddress () << ": Switching to channel idle");
      m_state = RUNNING;
      StartTimer ();
    }
   if (wait_ack)
```

```
    {
      StartTimer ();
    }
}
void
UanMacCw::NotifyCcaStart (void)
{
  if (m_state == RUNNING)
    {
      NS_LOG_DEBUG ("Time " << Simulator::Now ().GetSeconds () << " Addr " <<
GetAddress () << ": Switching to channel busy");
      m_state = CCABUSY;
      SaveTimer ();
    }
   if (wait_ack)
    {
      SaveTimer ();
    }
}
void
UanMacCw::NotifyCcaEnd (void)
{
  if (m_state == CCABUSY)
    {
      NS_LOG_DEBUG ("Time " << Simulator::Now ().GetSeconds () << " Addr " <<
GetAddress () << ": Switching to channel idle");
      m_state = RUNNING;
      StartTimer ();
    }
   if (wait_ack)
    {
      StartTimer ();
    }
}
void
UanMacCw::NotifyTxStart (Time duration)
{
  if (m_txEndEvent.IsRunning ())
    {
      Simulator::Cancel (m_txEndEvent);
    }
  m_txEndEvent = Simulator::Schedule (duration, &UanMacCw::EndTx, this);
  NS_LOG_DEBUG ("Time " << Simulator::Now ().GetSeconds () << " scheduling
TxEndEvent with delay " << duration.GetSeconds ());
  if (m_state == RUNNING)
    {
      NS_ASSERT (0);
      m_state = CCABUSY;
      SaveTimer ();
    }
   if (wait_ack)
    {
      SaveTimer ();
    }
}
void
UanMacCw::EndTx (void)
```

```
{
  NS_ASSERT (m_state == TX || m_state == CCABUSY);
  if (m_state == TX)
    {
      m_state = IDLE;
    }
  else if (m_state == CCABUSY)
    {
      if (m_phy->IsStateIdle ())
        {
          NS_LOG_DEBUG ("Time " << Simulator::Now ().GetSeconds () << " Addr " <<
GetAddress () << ": Switching to channel idle (After TX!)");
          m_state = RUNNING;
          StartTimer ();
        }
    }
  else
    {
      NS_FATAL_ERROR ("In strange state at UanMacCw EndTx");
    }
   if (wait_ack)
    {
      StartTimer ();
    }
}
void
UanMacCw::SetCw (uint32_t cw)
{
  m_cw = cw;
}
void
UanMacCw::SetSlotTime (Time duration)
{
  m_slotTime = duration;
}
uint32_t
UanMacCw::GetCw (void)
{
  return m_cw;
}
Time
UanMacCw::GetSlotTime (void)
{
  return m_slotTime;
}
bool
UanMacCw::Enqueue (Ptr<Packet> packet, const Address &dest, uint16_t protocolNumber)
{
if (wait_ack){return true;}
else {
  switch (m_state)
    {
    case CCABUSY:
      NS_LOG_DEBUG ("Time " << Simulator::Now ().GetSeconds () << " MAC " <<
GetAddress () << " Starting enqueue CCABUSY");
      if (m_txEndEvent.IsRunning () == TX)
        {
```

```
        NS_LOG_DEBUG ("State is TX");
       }
      else
       {
        NS_LOG_DEBUG ("State is not TX");
       }
     NS_ASSERT (m_phy->GetTransducer ()->GetArrivalList ().size () >= 1 || m_phy-
>IsStateTx ());
     return false;
    case RUNNING:
     NS_LOG_DEBUG ("MAC " << GetAddress () << " Starting enqueue RUNNING");
     return false;
    case TX:
    case IDLE:
     {
      NS_ASSERT (!m_pktTx);
      UanHeaderCommon header;
      header.SetDest (UanAddress::ConvertFrom (dest));
      header.SetSrc (m_address);
      header.SetType (0);

      packet->AddHeader (header);
      m_enqueueLogger (packet, protocolNumber);
      if (m_phy->IsStateBusy ())
       {
         m_pktTx = packet;
         m_pktTxProt = protocolNumber;
         m_state = CCABUSY;
         UniformVariable rv (0,m_cw);
         uint32_t cw = (uint32_t) rv.GetValue ();
         m_savedDelayS = Seconds ((double)(cw) * m_slotTime.GetSeconds ());
         m_sendTime = Simulator::Now () + m_savedDelayS;
         NS_LOG_DEBUG ("Time " << Simulator::Now ().GetSeconds () << ": Addr " <<
GetAddress () << ": Enqueuing new packet while busy:  (Chose CW " << cw << ", Sending at
" << m_sendTime.GetSeconds () << " Packet size: " << packet->GetSize ());
         NS_ASSERT (m_phy->GetTransducer ()->GetArrivalList ().size () >= 1 || m_phy-
>IsStateTx ());
        }
      else
       {
         NS_ASSERT (m_state != TX);
         NS_LOG_DEBUG ("Time " << Simulator::Now ().GetSeconds () << ": Addr " <<
GetAddress () << ": Enqueuing new packet while idle (sending)");
         m_state = TX;
         m_phy->SendPacket (packet,protocolNumber);
         wait_ack = true;
         timer_ack = Seconds(1.5);
         m_sendTimeAck = Simulator::Now () + timer_ack;
         m_EventAck = Simulator::Schedule(timer_ack, &UanMacCw::TimeOutAck, this);
       }
      break;
     }
    default:
     NS_LOG_DEBUG ("MAC " << GetAddress () << " Starting enqueue SOMETHING
ELSE");
     return false;
    }
```

```
  return true;
}
}
void
UanMacCw::PhyRxPacketGood (Ptr<Packet> packet, double sinr, UanTxMode mode)
{
  UanHeaderCommon header;
  packet->RemoveHeader (header);
  if (header.GetDest () == m_address)
    {
      m_forwardUpCb (packet, header.GetSrc ());
     if(header.GetType () == 0){
      ns3::Ptr<ns3::Packet> packetAck = ns3::Create<ns3::Packet> (0);
      UanHeaderCommon headerAck;
      headerAck.SetDest (header.GetSrc ());
      headerAck.SetSrc (m_address);
      headerAck.SetType (1);
      packetAck->AddHeader (headerAck);
      m_state = TX;
      m_phy->SendPacket (packetAck,m_pktTxProt);
     }
     else {
    wait_ack = false;
       if (retransmissao)
        {
         Simulator::Cancel (m_sendEventR);
          m_sendTimeR = Seconds (0);
          m_savedDelaySR = Seconds (0);
          m_pktTx = 0;
          m_pktTx1 = 0;
          retransmissao = false;
        }
        else
        {
         Simulator::Cancel (m_EventAck);
         m_pktTx1 = 0;
         m_sendTimeAck = Seconds (0);
         timer_ack = Seconds (1.5);
        }
     }
    }
}
void
UanMacCw::PhyRxPacketError (Ptr<Packet> packet, double sinr)
{
}
void
UanMacCw::SaveTimer (void)
{
  if (wait_ack)
  {
        if (retransmissao)
         {
           m_savedDelaySR = m_sendTimeR - Simulator::Now ();
           Simulator::Cancel (m_sendEventR);
         }
        else
```

```
        {
        }
    }
  else {
  NS_LOG_DEBUG ("Time " << Simulator::Now ().GetSeconds () << " Addr " << GetAddress
() << " Saving timer (Delay = " << (m_savedDelayS = m_sendTime - Simulator::Now
()).GetSeconds () << ")");
  NS_ASSERT (m_pktTx);
  NS_ASSERT (m_sendTime >= Simulator::Now ());
  m_savedDelayS = m_sendTime - Simulator::Now ();
  Simulator::Cancel (m_sendEvent);
  }
}

void
UanMacCw::StartTimer (void)
{
  if(wait_ack){
        if(retransmissao)
        {
        m_sendTimeR = Simulator::Now () + m_savedDelaySR;
                if (m_sendTimeR <= Simulator::Now ())
                {
                        Retransmissao ();
                }
                else
                {
                        m_sendEventR = Simulator::Schedule (m_savedDelaySR,
&UanMacCw::Retransmissao, this);
                }
        }
        else
        {
        }
    }
  else {
  m_sendTime = Simulator::Now () + m_savedDelayS;
  if (m_sendTime == Simulator::Now ())
    {
      SendPacket ();
    }
  else
    {
      m_sendEvent = Simulator::Schedule (m_savedDelayS, &UanMacCw::SendPacket, this);
      NS_LOG_DEBUG ("Time " << Simulator::Now ().GetSeconds () << " Addr " <<
GetAddress () << " Starting timer (New send time = " << this->m_sendTime.GetSeconds ()
<< ")");
    }
  }
}
void
UanMacCw::SendPacket (void)
{
  if (wait_ack){}
  else {
  NS_LOG_DEBUG ("Time " << Simulator::Now ().GetSeconds () << " Addr " << GetAddress
() << " Transmitting ");
```

```
  NS_ASSERT (m_state == RUNNING);
  m_state = TX;
  m_phy->SendPacket (m_pktTx,m_pktTxProt);
  m_pktTx1 = m_pktTx;
  m_pktTx = 0;
  m_sendTime = Seconds (0);
  m_savedDelayS = Seconds (0);
  wait_ack = true;
  timer_ack = Seconds(1.5);
  m_sendTimeAck = Simulator::Now () + timer_ack;
  m_EventAck = Simulator::Schedule(timer_ack, &UanMacCw::TimeOutAck, this);
  }
}
void
UanMacCw::TimeOutAck (void)
{
wait_ack = true;
retransmissao = true;
m_sendTimeAck = Seconds (0);
timer_ack = Seconds (1.5);
UniformVariable rv (0,m_cw);
uint32_t cw = (uint32_t) rv.GetValue ();
m_savedDelaySR = Seconds ((double)(cw) * m_slotTime.GetSeconds ());
m_sendTimeR = Simulator::Now () + m_savedDelayS;
m_sendEventR = Simulator::Schedule (m_savedDelayS, &UanMacCw::Retransmissao, this);
}
void
UanMacCw::Retransmissao (void)
{
  m_state = TX;
  m_phy->SendPacket (m_pktTx1,m_pktTxProt);
  m_sendTimeR = Seconds (0);
  m_savedDelaySR = Seconds (0);
  m_pktTx = 0;
  m_pktTx1 = 0;
  wait_ack = false;
  retransmissao = false;
}
} // namespace ns3
```

II Biblioteca do Código do Protocolo Proposto

```
#ifndef UANMACCW_H
#define UANMACCW_H
#include "ns3/uan-mac.h"
```

```
#include "ns3/nstime.h"
#include "ns3/simulator.h"
#include "ns3/uan-phy.h"
#include "ns3/uan-tx-mode.h"
#include "ns3/uan-address.h"

namespace ns3 {

class UanMacCw : public UanMac,
                 public UanPhyListener
{
public:
  UanMacCw ();
  virtual ~UanMacCw ();
  static TypeId GetTypeId (void);

  virtual void SetCw (uint32_t cw);

  virtual void SetSlotTime (Time duration);

  virtual uint32_t GetCw (void);
  virtual Time GetSlotTime (void);

// Inherited methods
  virtual Address GetAddress ();
  virtual void SetAddress (UanAddress addr);
  virtual bool Enqueue (Ptr<Packet> pkt, const Address &dest, uint16_t protocolNumber);
  virtual void SetForwardUpCb (Callback<void, Ptr<Packet>, const UanAddress&> cb);
  virtual void AttachPhy (Ptr<UanPhy> phy);
  virtual Address GetBroadcast (void) const;
  virtual void Clear (void);

  // PHY listeners
  /// Function called by UanPhy object to notify of packet reception
  virtual void NotifyRxStart (void);
  /// Function called by UanPhy object to notify of packet received successfully
  virtual void NotifyRxEndOk (void);
  /// Function called by UanPhy object to notify of packet received in error
  virtual void NotifyRxEndError (void);
  /// Function called by UanPhy object to notify of channel sensed busy
  virtual void NotifyCcaStart (void);
  /// Function called by UanPhy object to notify of channel no longer sensed busy
  virtual void NotifyCcaEnd (void);
  /// Function called by UanPhy object to notify of outgoing transmission start
  virtual void NotifyTxStart (Time duration);
Time timer_ack;
bool wait_ack;
EventId m_EventAck;
void TimeOutAck (void);
void Retransmissao (void);
bool retransmissao;
Time m_sendTimeR;
Time m_sendTimeAck;
Time m_savedDelaySR;
EventId m_sendEventR;
private:
  typedef enum {
```

```
    IDLE, CCABUSY, RUNNING, TX
  } State;
  Callback <void, Ptr<Packet>, const UanAddress& > m_forwardUpCb;
  UanAddress m_address;
  Ptr<UanPhy> m_phy;
  TracedCallback<Ptr<const Packet>, UanTxMode > m_rxLogger;
  TracedCallback<Ptr<const Packet>, uint16_t  > m_enqueueLogger;
  TracedCallback<Ptr<const Packet>, uint16_t  > m_dequeueLogger;
  // Mac parameters
  uint32_t m_cw;
  Time m_slotTime;
  // State variables
  Time m_sendTime;
  Time m_savedDelayS;
  Ptr<Packet> m_pktTx;
  Ptr<Packet> m_pktTx1;
  uint16_t m_pktTxProt;
  EventId m_sendEvent;
  EventId m_txEndEvent;
  State m_state;
  bool m_cleared;
  void PhyRxPacketGood (Ptr<Packet> packet, double sinr, UanTxMode mode);
  void PhyRxPacketError (Ptr<Packet> packet, double sinr);
  void SaveTimer (void);
  void StartTimer (void);
  void SendPacket (void);
  void EndTx (void);
protected:
  virtual void DoDispose ();
};
}
#endif // UANMACCW_H
```

III Código da Simulação

```
#include "uan-cw-example.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/mobility-module.h"
#include "ns3/tools-module.h"
#include "ns3/applications-module.h"
#include <fstream>
using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("UanCwExample");
Experiment::Experiment ()
  : m_numNodes (10),
    m_dataRate (80),
```

```
    m_depth (70),
    m_boundary (500),
    m_packetSize (32),
    m_bytesTotal (0),
    m_cwMin (1),
    m_cwMax (200),
    m_cwStep (1),
    m_avgs (7),
    m_slotTime (Seconds (0.2)),
    m_simTime (Seconds (1800)),
    m_gnudatfile ("uan-cw-example.gpl"),
    m_asciitracefile ("uan-cw-example.asc"),
    m_bhCfgFile ("uan-apps/dat/default.cfg")
{
}
void
Experiment::ResetData ()
{
  NS_LOG_DEBUG (Simulator::Now ().GetSeconds () << "  Resetting data");
  m_throughputs.push_back (m_bytesTotal * 8.0 / m_simTime.GetSeconds ());
  m_bytesTotal = 0;
}
void
Experiment::IncrementCw (uint32_t cw)
{
  NS_ASSERT (m_throughputs.size () == m_avgs);
  double avgThroughput = 0.0;
  for (uint32_t i=0; i<m_avgs; i++)
    {
      avgThroughput += m_throughputs[i];
    }
  avgThroughput /= m_avgs;
  m_data.Add (cw, avgThroughput/m_dataRate);
  m_throughputs.clear ();
  Config::Set ("/NodeList/*/DeviceList/*/Mac/CW", UintegerValue (cw + m_cwStep));
  SeedManager::SetRun (SeedManager::GetRun () + 1);
  NS_LOG_DEBUG ("Average for cw=" << cw << " over " << m_avgs << " runs: " <<
avgThroughput);
}

void
Experiment::UpdatePositions (NodeContainer &nodes)
{
  NS_LOG_DEBUG (Simulator::Now ().GetSeconds () << " Updating positions");
  NodeContainer::Iterator it = nodes.Begin ();
  UniformVariable uv (0, m_boundary);
  for (; it != nodes.End (); it++)
    {
      Ptr<MobilityModel> mp = (*it)->GetObject<MobilityModel> ();
      mp->SetPosition (Vector (uv.GetValue (), uv.GetValue (), 70.0));
    }
}
void
Experiment::ReceivePacket (Ptr<Socket> socket)
{
  Ptr<Packet> packet;
  while (packet = socket->Recv ())
```

```
    {
      m_bytesTotal += packet->GetSize ();
    }
  packet = 0;
}
Gnuplot2dDataset
Experiment::Run (UanHelper &uan)
{
  uan.SetMac ("ns3::UanMacCw", "CW", UintegerValue (m_cwMin), "SlotTime", TimeValue
(m_slotTime));
  NodeContainer nc = NodeContainer ();
  NodeContainer sink = NodeContainer ();
  nc.Create (m_numNodes);
  sink.Create (1);

  PacketSocketHelper socketHelper;
  socketHelper.Install (nc);
  socketHelper.Install (sink);

#ifdef UAN_PROP_BH_INSTALLED
  Ptr<UanPropModelBh> prop = CreateObjectWithAttributes<UanPropModelBh>
("ConfigFile", StringValue ("exbhconfig.cfg"));
#else
  Ptr<UanPropModelThorp> prop = CreateObjectWithAttributes<UanPropModelThorp> ();
#endif //UAN_PROP_BH_INSTALLED
  Ptr<UanChannel> channel = CreateObjectWithAttributes<UanChannel>
("PropagationModel", PointerValue (prop));
  //Create net device and nodes with UanHelper
  NetDeviceContainer devices = uan.Install (nc, channel);
  NetDeviceContainer sinkdev = uan.Install (sink, channel);

  MobilityHelper mobility;
  Ptr<ListPositionAllocator> pos = CreateObject<ListPositionAllocator> ();
  {
    UniformVariable urv (0, m_boundary);
    pos->Add (Vector (m_boundary / 2.0, m_boundary / 2.0, m_depth));
    double rsum = 0;
    double minr = 2 * m_boundary;
    for (uint32_t i = 0; i < m_numNodes; i++)
      {
        double x = urv.GetValue ();
        double y = urv.GetValue ();
        double newr = sqrt ((x - m_boundary / 2.0) * (x - m_boundary / 2.0)
                            + (y - m_boundary / 2.0) * (y - m_boundary / 2.0));
        rsum += newr;
        minr = std::min (minr, newr);
        pos->Add (Vector (x, y, m_depth));
      }
    NS_LOG_DEBUG ("Mean range from gateway: " << rsum / m_numNodes
                                      << "    min. range " << minr);
    mobility.SetPositionAllocator (pos);
    mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
    mobility.Install (sink);
    NS_LOG_DEBUG ("Position of sink: "
                  << sink.Get (0)->GetObject<MobilityModel> ()->GetPosition ());
    mobility.Install (nc);
    PacketSocketAddress socket;
```

```
socket.SetSingleDevice (sinkdev.Get (0)->GetIfIndex ());
socket.SetPhysicalAddress (sinkdev.Get (0)->GetAddress ());
socket.SetProtocol (0);
OnOffHelper app ("ns3::PacketSocketFactory", Address (socket));
app.SetAttribute ("OnTime", RandomVariableValue (ConstantVariable (1)));
app.SetAttribute ("OffTime", RandomVariableValue (ConstantVariable (0)));
app.SetAttribute ("DataRate", DataRateValue (m_dataRate));
app.SetAttribute ("PacketSize", UintegerValue (m_packetSize));
ApplicationContainer apps = app.Install (nc);
apps.Start (Seconds (0.5));
Time nextEvent = Seconds (0.5);
for (uint32_t cw = m_cwMin; cw <= m_cwMax; cw += m_cwStep)
 {
   for (uint32_t an = 0; an < m_avgs; an++)
    {
      nextEvent += m_simTime;
      Simulator::Schedule (nextEvent, &Experiment::ResetData, this);
      Simulator::Schedule (nextEvent, &Experiment::UpdatePositions, this, nc);
    }
   Simulator::Schedule (nextEvent, &Experiment::IncrementCw, this, cw);
 }
apps.Stop (nextEvent + m_simTime);
Ptr<Node> sinkNode = sink.Get (0);
TypeId psfid = TypeId::LookupByName ("ns3::PacketSocketFactory");
if (sinkNode->GetObject<SocketFactory> (psfid) == 0)
 {
   Ptr<PacketSocketFactory> psf = CreateObject<PacketSocketFactory> ();
   sinkNode->AggregateObject (psf);
 }
Ptr<Socket> sinkSocket = Socket::CreateSocket (sinkNode, psfid);
sinkSocket->Bind (socket);
sinkSocket->SetRecvCallback (MakeCallback (&Experiment::ReceivePacket, this));
m_bytesTotal = 0;
std::ofstream ascii (m_asciitracefile.c_str ());
if (!ascii.is_open ())
 {
   NS_FATAL_ERROR ("Could not open ascii trace file: "
                   << m_asciitracefile);
 }
uan.EnableAsciiAll (ascii);
Simulator::Run ();
sinkNode = 0;
sinkSocket = 0;
pos = 0;
channel = 0;
prop = 0;
for (uint32_t i=0; i < nc.GetN (); i++)
 {
   nc.Get (i) = 0;
 }
for (uint32_t i=0; i < sink.GetN (); i++)
 {
   sink.Get (i) = 0;
 }
for (uint32_t i=0; i < devices.GetN (); i++)
 {
   devices.Get (i) = 0;
```

```
    }
   for (uint32_t i=0; i < sinkdev.GetN (); i++)
    {
      sinkdev.Get (i) = 0;
    }
   Simulator::Destroy ();
   return m_data;
  }
}
intmain (int argc, char **argv)
{
  LogComponentEnable ("UanCwExample", LOG_LEVEL_ALL);
  Experiment exp;
  std::string gnudatfile ("cwexpgnuout.dat");
  //std::string perModel = "ns3::UanPhyPerGenDefault";
  std::string perModel = "ns3::UanPhyPerUmodem";
  //std::string sinrModel = "ns3::UanPhyCalcSinrDefault";
  std::string sinrModel = "ns3::UanPhyCalcSinrFhFsk";
  CommandLine cmd;
  cmd.AddValue ("NumNodes", "Number of transmitting nodes", exp.m_numNodes);
  cmd.AddValue ("Depth", "Depth of transmitting and sink nodes", exp.m_depth);
  cmd.AddValue ("RegionSize", "Size of boundary in meters", exp.m_boundary);
  cmd.AddValue ("PacketSize", "Generated packet size in bytes", exp.m_packetSize);
  cmd.AddValue ("DataRate", "DataRate in bps", exp.m_dataRate);
  cmd.AddValue ("CwMin", "Min CW to simulate", exp.m_cwMin);
  cmd.AddValue ("CwMax", "Max CW to simulate", exp.m_cwMax);
  cmd.AddValue ("SlotTime", "Slot time duration", exp.m_slotTime);
  cmd.AddValue ("Averages", "Number of topologies to test for each cw point", exp.m_avgs);
  cmd.AddValue ("GnuFile", "Name for GNU Plot output", exp.m_gnudatfile);
  cmd.AddValue ("PerModel", "PER model name", perModel);
  cmd.AddValue ("SinrModel", "SINR model name", sinrModel);
  cmd.Parse (argc, argv);
  ObjectFactory obf;
  obf.SetTypeId (perModel);
  Ptr<UanPhyPer> per = obf.Create<UanPhyPer> ();
  obf.SetTypeId (sinrModel);
  Ptr<UanPhyCalcSinrFhFsk> sinr = obf.Create<UanPhyCalcSinrFhFsk> ();
  UanHelper uan;
  UanTxMode mode;
  mode = UanTxModeFactory::CreateMode (UanTxMode::FSK, exp.m_dataRate,
                                       exp.m_dataRate, 12000,
                                       exp.m_dataRate, 2,
                                       "Default mode");
  UanModesList myModes;
  myModes.AppendMode (mode);
  uan.SetPhy ("ns3::UanPhyGen",
              "PerModel", PointerValue (per),
              "SinrModel", PointerValue (sinr),
              "SupportedModes", UanModesListValue (myModes));
  Gnuplot gp;
  Gnuplot2dDataset ds;
  ds = exp.Run (uan);
  gp.AddDataset (ds);
  std::ofstream of (exp.m_gnudatfile.c_str ());
  if (!of.is_open ())
    {
      NS_FATAL_ERROR ("Can not open GNU Plot outfile: " << exp.m_gnudatfile);
```

```
    }
  gp.GenerateOutput (of);
  per = 0;
  sinr = 0;
}
```

IV Biblioteca para o Código da Simulação

```
#ifndef UAN_CW_EXAMPLE_H
#define UAN_CW_EXAMPLE_H
#include "ns3/network-module.h"
#include "ns3/tools-module.h"
#include "ns3/uan-module.h"
using namespace ns3;

/**
 * \class Experiment
 * \brief Helper class for UAN CW MAC example
 *
 */
class Experiment
{
public:
  Gnuplot2dDataset Run (UanHelper &uan);
  void ReceivePacket (Ptr<Socket> socket);
  void UpdatePositions (NodeContainer &nodes);
  void ResetData ();
  void IncrementCw (uint32_t cw);
  uint32_t m_numNodes;
  uint32_t m_dataRate;
  double m_depth;
  double m_boundary;
  uint32_t m_packetSize;
  uint32_t m_bytesTotal;
  uint32_t m_cwMin;
  uint32_t m_cwMax;
  uint32_t m_cwStep;
  uint32_t m_avgs;
  Time m_slotTime;
  Time m_simTime;
  std::string m_gnudatfile;
  std::string m_asciitracefile;
  std::string m_bhCfgFile;
  Gnuplot2dDataset m_data;
  std::vector<double> m_throughputs;
  Experiment ();
};
#endif /* UAN_CW_EXAMPLE_H */
```