Pingo d'água: ICMP para Internet das Coisas Aquáticas

Francisco H. M. B. Lima[1], Luiz F. M. Vieira[1], Marcos A. M. Vieira[1], Alex B. Vieira[2], José Augusto M. Nacif[3]

[1]Departamento de Ciência da Computação - Universidade Federal de Minas Gerais (UFMG)

[2]Departamento de Ciência da Computação - Universidade Federal de Juiz de Fora (UFJF)

[3]Universidade Federal de Viçosa (UFV) - Campus UFV Florestal

{francisco.lima, lfvieira, mmvieira}@dcc.ufmg.br

alex.borges@ufjf.edu.br, jnacif@ufv.br

Abstract. *High latency, low transmission rate and the presence of limited computional devices are characteristics present in scenarios of underwater communications. In this paper we present the Water Ping application, which implements the ICMP (Internet Control Message Protocol) protocol with header compression, to work in this scenario. The proposed protocol allows underwater devices to receive ping messages, in this way participating in the Internet and forming the Internet of Underwater Things (IoUT). Results show that underwater network devices can now be remotely monitored by the Internet, and also significative gains in the compression, consuming 93.75% less energy.*

Resumo. *Alta latência, baixa taxa de transmissão e presença de dispositivos computacionalmente limitados são características presentes no cenário da comunicação em ambientes aquáticos. Neste trabalho apresentamos a aplicação Pingo d'água, que implementa o protocolo ICMP (Internet Control Message Protocol) com compressão de cabeçalhos, a fim de atender o cenário. O protocolo proposto permite que dispositivos aquáticos possam receber mensagens ping, dessa forma participando da Internet, e formando a Internet das Coisas Aquáticas (IoUT). Resultados mostram que dispositivos de redes de sensores aquáticas podem ser monitorados remotamente pela Internet e o ganho na compressão é significativo, consumindo 93,75% menos energia.*

1. Introdução

Redes sem fio aquáticas, caracterizadas por terem em sua totalidade ou maior parte dispositivos computacionalmente limitados, tem se tornado foco crescente de pesquisas. Há inúmeras aplicações que podem se beneficiar de pesquisas nesta área, como por exemplo, monitoração de recursos hídricos, monitoração de oleodutos, bacias hidrográficas, reservatórios de hidrelétricas e plataformas de petróleo [Vieira et al. 2010a]. Parte do crescente interesse em redes de sensores aquáticas deve-se ao desenvolvimento de dispositivos capazes de se comunicarem entre si abaixo da superfície aquática, conhecidos como dispositivos aquáticos e nós sensores [Viana et al. 2015].

Apesar da existência de projetos concretos de redes sem fio aquáticas, observa-se uma lacuna quanto à interoperabilidade dessas redes com as redes tradicionais que operam

na Internet. De fato, soluções de comunicação, roteamento e até mesmo de aquisição e análise de dados são criadas fortemente acopladas a um contexto ou tecnologia. Assim, mudanças são necessárias para que esse cenário se torne ubíquo e, análogo à Internet das Coisas, se forme uma Internet das Coisas Aquáticas (IdCA ou IoUT - *Internet of Underwater Things*).

Dada a baixa capacidade computacional dos dispositivos envolvidos e o desafiante canal de comunicação (geralmente acústico), redes sem fio no ambiente aquático não suportam tráfego semelhante ao das redes tradicionais, que operam sobre a arquitetura TCP/IP. Mesmo redes sem fio tradicionais não experimentam características semelhantes a esse meio, como pequena largura de banda, limitação da velocidade do som na água –que representa uma latência cinco ordens de magnitude maior que comunicações na velocidade da luz– e uma notória maior taxa de bits errados [Vieira et al. 2010a]. Além disso, os dispositivos que compõem redes aquáticas apresentam fortes limitações quanto a energia, em especial, por conta de estarem localizados em locais de difícil acesso e manutenção [Coutinho et al. 2016d]. Em suma, nesse cenário faz-se necessária uma comunicação composta por mensagens que sejam curtas e que não ocorram com muita frequência [Sun and Melodia 2013].

Assim, neste trabalho apresentamos uma proposta de compressão para mensagens do protocolo ICMP [Postel 1981] e ICMPv6 [Conta and Gupta 2006] que servirá como bloco base de construção de um arcabouço de IdCA. Note que, a versão do IPv6 para redes sem fio pessoais e de baixo consumo (*6LoWPAN*) [Kushalnagar et al. 2007] apresenta solução para um problema semelhante que ocorre em redes sem fio terrestres e assim, serve como inspiração para o problema tratado em redes sem fio aquáticas. Por fim, as alterações na pilha de protocolos proposta permitirá a monitoração ubíqua dos nós sensores aquáticos, exemplificadas neste trabalho –mas não restrito– pela monitoração realizada por *ping*.

Nós avaliamos as alterações propostas através de testes em um sistema realista, composto por um dispositivo aquático, um dispositivo intermediário (*gateway*) e um computador. Um outro computador remoto foi utilizado para executar o comando Ping, do protocolo ICMP. Assim uma mensagem pode vir da Internet, passar pelos dispositivos do sistema de rede sem fio aquática, sendo comprimida e chegando ao dispositivo aquático final. Este dispositivo aquático gera uma mensagem de resposta que foi descomprimida no *gateway* e enviada de volta ao remetente remoto na Internet, por meio do protocolo ICMP tradicional. Os resultados mostram que dispositivos de redes de sensores aquáticas, que antes não conversavam com a Internet, podem ser monitorados remotamente. Mais ainda, o ganho na compressão é significativo, chegando a 93,75% no consumo energético, permitindo assim a criação da Internet das Coisas Aquáticas.

O restante deste trabalho é composto como a seguir. Na Seção 2 apresentamos os trabalhos relacionados. Na Seção 3 apresentamos a solução proposta, contendo o cenário considerado, a compressão de cabeçalhos e a descrição da implementação. Na Seção 4 apresentamos os testes e resultados. Finalmente, na Seção 5 apresentamos a conclusão e trabalhos futuros.

2. Trabalhos Relacionados

Existem diversas propostas realizadas com o objetivo de superar obstáculos encontrados para a comunicação sem fio no ambiente aquático. Protocolos de enlace, como Aloha [Vieira et al. 2006], modelos de mobilidade [Caruso et al. 2008], serviços e técnicas de localização [Vieira et al. 2010b, Erol et al. 2007b, Erol et al. 2007a, Erol et al. 2008] protocolos de roteamento como o Pressure Routing [Lee et al. 2010], GEDAR [Coutinho et al. 2014], oportunístico [Vieira 2012, Coutinho et al. 2016d], geográfico [Coutinho et al. 2016a], com controle de profundidade [Coutinho et al. 2013], baseados em centralidade [Coutinho et al. 2016c] com controle para nós dormirem [Coutinho et al. 2015], além de técnicas para economizar energia [Coutinho et al. 2016b], existem para as redes de sensores aquáticas. Apesar dos evidentes avanços que estes trabalhos trazem, nenhum trata a questão de criar a Internet das Coisas Aquáticas e/ou permitir que mensagens da Internet trafeguem pelas redes de sensores aquáticas.

Existem propostas para o acesso remoto a redes sem fio aquáticas (RSSFAs) e a seus dispositivos (sensores e nós). Porém, ainda não há um cenário que permita que RSSFAs possam ser remotamente reconfiguradas [Sun and Melodia 2013]. De fato, [Sun and Melodia 2013] propõem uma nova arquitetura que visa interoperabilidade com a arquitetura TCP/IP, permitindo o acesso remoto a uma rede aquática através da Internet. A proposta tem como características de destaque a inclusão de uma camada de adaptação à pilha de protocolos TCP/IP tradicional e a presença de um dispositivo na superfície, responsável pela comunicação com a Internet. Além disso, o trabalho inclui a instalação e execução de *softwares* de *driver* a nível de *Kernel* e a nível de usuário para viabilizar o seu funcionamento. Trata-se de uma solução promissora que constitui um passo importante para o alcance da Internet das Coisas Aquáticas que pode colaborar com o nosso trabalho. No entanto, no presente trabalho, nós não propomos uma nova arquitetura, mas sim uma versão com compressão para o protocolo ICMP. Essa alteração não exige a instalação de *drivers* adicionais ou de aplicações executadas a nível de *Kernel*. Além disso, com as alterações propostas, RSSFAs contariam com suporte da pilha de protocolos da arquitetura TCP/IP e, assim, a reconfiguração poderia ser feita por aplicações como SSH e FTP. Mais ainda, além da reconfiguração, será possível também realizar o monitoramento utilizando ferramentas como *ping* e *traceroute*, úteis para planejamento e avaliação de desempenho de rotas.

O *6LoWPAN* [Kushalnagar et al. 2007], citado na introdução, propõe um princípio de compressão que servirá de inspiração para este trabalho. A ideia principal do *6LoWPAN* é fragmentar os longos pacotes do protocolo IPv6, em quadros menores, realizando também a compressão dos cabeçalhos das mensagens. A compressão toma como base os valores mais comuns que aparecem nos campos de uma mensagem, permitindo que seja reduzido o comprimento (em *bits*) de cada um dos campos.

Outros protocolos bem conhecidos também foram adaptados para redes limitadas, como RSSFAs. Por exemplo, [Choi et al. 2009] faz a adaptação do protocolo *Simple Network Management Protocol* - SNMP para redes pessoais e de baixo consumo. Trata-se de uma extensão modificada que torna possível a transmissão de mensagens do protocolo SNMP, operando, inclusive, sobre o 6LoWPAN, e que alcançou resultados positivos consideráveis. [Ng et al. 2013] também apresentam uma adaptação de protocolo, desta vez,

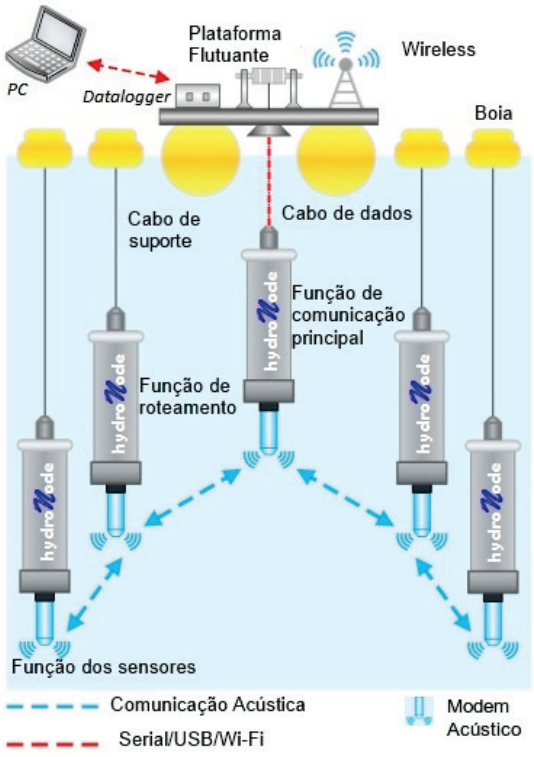


Figura 1. Cenário referência para este trabalho.

voltada ao ambiente de redes aquáticas. Os autores consideram os diversos obstáculos e características do cenário e trazem uma versão do protocolo MAC com *handshaking* assíncrono, conhecida como *Bidirectional-Concurrent MAC with Packet Bursting* (Bic-MAC). Note que, o foco do nosso trabalho é construir um bloco base para que diversos protocolos possam ser utilizados de forma indiscriminada entre Internet e RSSFA. Um dos exemplos adotados nesse trabalho é o monitoramento de dispositivos através da Internet, por meio do uso de mensagens mais curtas. A consequência natural dessa abordagem é uma melhor taxa de entrega de quadros na camada de enlace.

3. Solução Proposta

Nesta seção apresentaremos (i) o cenário considerado; (ii) nossa proposta de compressão de cabeçalhos e (iii) o Pingo d'água, uma adaptação do ICMP para monitoração de elementos em RSSFA.

3.1. Cenário Considerado

A Figura 1 apresenta um cenário típico de RSSFA. Em aplicações deste tipo, diversos nós sensores captam informações do meio. Comumente os nós se comunicam entre si. Estes nós sensores utilizam modens acústicos limitados para esta comunicação. Parte dos dados é recolhida por uma plataforma flutuante, que envia estes dados para posterior análise.

Neste trabalho, o modem acústico apresenta uma limitação de *buffer* de dados de apenas 24 bytes [Almeida et al. 2016]. Essa limitação é observada adiante, na proposta de compressão. Em outras palavras, desejamos alcançar uma compressão na qual as mensagens tenham, no máximo, 24 bytes de comprimento. Com relação a outros cabeçalhos que

podem compor a mensagem (cabeçalhos das outras camadas da arquitetura TCP/IP, por exemplo), não serão propostas alterações. Acreditamos que as soluções de compressão existentes na literatura, como o próprio *6LoWPAN*, poderão ser utilizadas em conjunto.

A Figura 1 também apresenta a *Datalogger Board*, um dispositivo que age como *gateway* de comunicação com o mundo exterior. Esse *gateway* é o dispositivo responsável pela comunicação dos dispositivos do ambiente aquático com os demais dispositivos terrestres, que operam segundo a pilha de protocolos da arquitetura TCP/IP. A comunicação com os dispositivos terrestres pode ocorrer, por exemplo, via cabo ou através de uma rede sem fio. O *gateway* está ligado via cabo a um nó sensor que, por sua vez, se comunica com os demais nós sensores pelo canal acústico. Por isso, a compressão das mensagens vindas da Internet para os dispositivos aquáticos será realizada no *gateway*, e nele também ocorrerá a descompressão das mensagens que fluirão no sentido oposto.

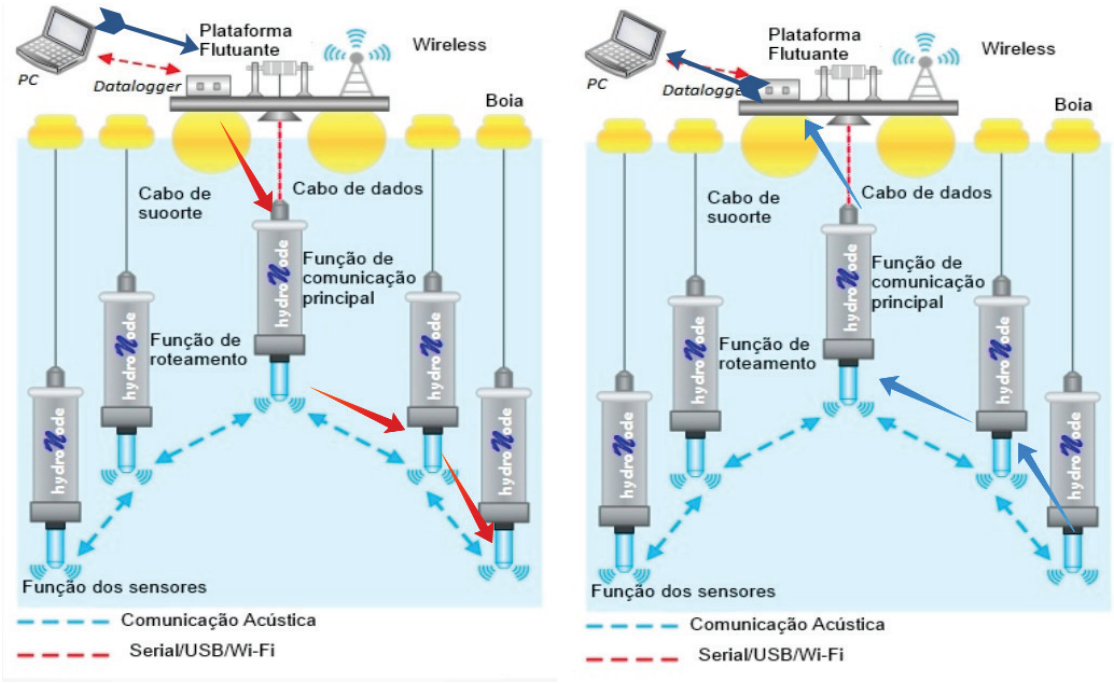
Seguindo esta especificação, temos como objetivo (mas não limitados a este) executar o comando Ping, composto pelas mensagens do tipo *Echo Request* e *Echo Reply* do protocolo ICMP, em qualquer computador na Internet. O fluxo de execução do comando pode ser ilustrado em duas etapas: a primeira, onde a requisição (*Echo Request*) flui de um dispositivo terrestre para os dispositivos aquáticos, mostrada na Figura 2(a); e a segunda, onde o nó destino da requisição gera a resposta (*Echo Reply*) e esta é enviada, fluindo no sentido contrário, como na Figura 2(b), com as respectivas compressão e descompressão sendo realizadas pelo *Gateway*.

3.2. Compressão de Cabeçalhos

Três campos estão presentes em toda mensagem do tipo ICMP, são eles: Tipo (*Type*), Código (*Code*) e *Checksum*. Os dois primeiros possuem 8 bits cada e o terceiro, 16. A composição e disposição do restante de uma mensagem ICMP depende dos valores do Tipo e do Código. Ou seja, dependem de qual mensagem do protocolo está sendo enviada. As mensagens trocadas pelo comando Ping possuem também os campos Identificador (*Identifier*) e Número de Sequência (*Sequence Number*), cada um tendo 16 bits, e o restante é composto pelos dados que serão "ecoados" pelos dispositivos, sendo também chamados de Carga (*Payload*) e tendo um tamanho usual de 56 bytes. Assim, é totalizado um tamanho usual de 64 bytes para as mensagens do comando Ping.

Uma ilustração do cabeçalho ICMP descrito para as mensagens *Echo Request* e *Echo Reply* [Postel 1981, Conta and Gupta 2006] pode ser vista na Figura 3 [Postel 1981]. Em cima deste cabeçalho, aplicamos um princípio de compressão semelhante ao existente no 6LowPAN. Assumiremos que os campos que compõem as mensagens possuirão com mais frequência alguns valores do que outros, e esses valores mais frequentes serão considerados valores comuns. Além disso, também iremos reduzir o tamanho dos campos que não possuem valores comuns (*Checksum*, Identificador, Número de Sequência e Dados), em função do novo cenário em que o protocolo será aplicado.

No corrente trabalho, consideramos como valores comuns os valores mais presentes nas mensagens *Echo Request*, *Echo Reply* e *Destination Unreachable*. Em nossos testes iniciais, as mensagens trocadas pelos dispositivos terão apenas os cabeçalhos e dados de mensagens ICMP. Os demais cabeçalhos, que fazem parte de uma mensagem na pilha de protocolos TCP/IP, não serão aqui incluídos. Dessa maneira, a compressão dos campos de uma mensagem ICMP será feita como a seguir:



(a) Comando Ping: *Echo Request* (b) Comando Ping: *Echo Reply*

Figura 2. Mensagens ping e pong através da Rede.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Identifier          |       Sequence Number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Data ...
+-+-+-+-+-
```

Figura 3. Cabeçalho das mensagens *Echo Request* e *Echo Reply* [Postel 1981].

- **Tipo** (*Type*): Para as mensagens do comando Ping, este campo recebe os valores 8 (*Echo Request*) e 0 (*Echo Reply*), no caso do protocolo ICMP. No caso do protocolo ICMPv6, os valores são, respectivamente 128 e 129, correspondentes às mesmas mensagens. Para a mensagem *Destination Unreachable*, o campo recebe os valores 3 e 1, para os protocolos ICMP e ICMPv6, respectivamente. Podemos assumir três valores comuns para o campo Tipo, um para cada mensagem mencionada, desprezando a diferenciação entre as versões do protocolo ICMP e definindo o valor deste campo como 2 bits. Assim, para 01 teremos *Echo Request*, para 00, *Echo Reply* e para 10, *Destination Unreachable* e o valor 11 será utilizado para indicar que este campo não está no seu valor comum.
- **Código** (*Code*): Este campo recebe sempre o valor 0 para as mensagens do comando Ping. No caso de uma mensagem do tipo *Destination Unreachable*, o valor 0 indica que não foi encontrada uma rota para o destino. Por isso, achamos prudente assumir 0 como sendo o valor comum do campo Código. Assim, quando em 0, o campo Código está no seu valor comum e, quando em 1, o campo está

com um valor diferente do comum.

- ***Checksum***: Ao invés do tamanho original de 16 bits, dado que a mensagem terá um tamanho menor do que o original, podemos reduzir o *Checksum* para 8 bits, deixando-o, assim, com metade do tamanho inicial.
- **Identificador** (*Identifier*): Uma vez que é baixa a quantidade de dispositivos que se comunicarão em baixo d'água e que, em uma mesma rede, não é viável que muitas mensagens estejam circulando no ambiente, podemos reduzir a 8 bits o identificador. Dessa forma, uma rede aquática funcionará com até 256 processos externos tentando acessá-la simultaneamente.
- **Número de Sequência** (*Sequence Number*): Pelas mesmas razões do item anterior e dado que uma característica do cenário aquático é a alta latência para a troca de mensagens, esta não pode ocorrer com muita frequência. Portanto, reduzimos este campo a 5 bits.
- **Dados** (*Payload*): Dada a necessidade de que as mensagens sejam curtas, a quantidade de dados a circular pela rede não precisa nem pode ser muito alta. Por isso, deixamos o campo de Dados com um tamanho padrão de 8 bits.

Temos assim uma redução de 40 bits – de 64 para 24 bits – no tamanho do cabeçalho e de aproximadamente 55 bytes – de usuais 56 para 1 byte – na parte de dados. A Figura 4 apresenta a nova estrutura da mensagem que circulará na rede subaquática.

Um campo com valor não comum terá o seu valor original concatenado ao final da mensagem. Assim, caso algum campo não tenha o valor comum, basta que o protocolo localize ao fim da mensagem o valor correto. Nosso protocolo preserva a ordem dos campos concatenados, tomando como referência sua posição original no cabeçalho.

A compressão do Identificador é feita através de uma espécie de mapeamento. O *gateway* é encarregado de manter uma relação entre um Identificador comprimido e o Identificador real correspondente. O *gateway* também cuida para que não haja repetição de Identificadores comprimidos.

Ao receber um identificador de 16 bits, o *gateway* escolhe o byte menos significativo deste identificador e, caso não exista um identificador comprimido com este valor, ele o armazena. Caso já exista, ele escolhe a parte mais significativa e repete o procedimento. Se ocorrer de também existir um identificador comprimido com o valor do byte mais significativo do campo original, o *gateway* passa a procurar sequencialmente por novos valores. Se não houver mais identificadores de 8 bits disponíveis, a transmissão é cancelada. Ao realizar o processo de descompressão, a relação entre o identificador real e o comprimido é descartada – tornando livre o valor que estava sendo utilizado – e uma nova relação é criada caso o mesmo identificador volte a aparecer. Durante os processos de compressão e descompressão, o *checksum* é recalculado, pois cada processo é responsável por gerar uma nova mensagem com tamanho e valores diferentes. A compressão do número de sequência é feita pegando-se a parte menos significativa do número de sequência original, tarefa que também cabe ao *gateway*.

3.3. Pingo d'água

Por fim, precisamos de uma maneira de tornar possível que, a partir de um terminal qualquer, seja possível executar o comando Ping tendo como destino um dispositivo subaquático. Esse dispositivo não está acessível pela Internet e, assim, não tem um

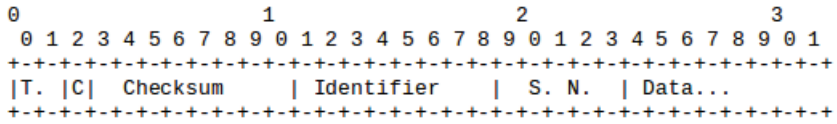


Figura 4. Cabeçalho ICMP Comprimido.

endereço IPV4 ou IPV6 tradicional. Além disso, pretendemos que não seja necessária uma alteração na pilha de protocolos ou em alguma aplicação já existente.

Como o cenário de rede aquática considerado neste trabalho não está em conformidade com a pilha de protocolos TCP/IP, é necessária uma maneira que torne possível identificar quando uma mensagem deve ou não ser encaminhada a uma rede aquática. Devemos também determinar o dispositivo destinatário no interior desta rede. Por exemplo, no cenário deste trabalho o *gateway* se conecta atualmente a um computador através da comunicação USB e, portanto, nesta configuração não tem o seu próprio endereço IP. Assim, cabe ao computador determinar se uma mensagem que chega até ele deve ou não ser encaminhada à rede aquática.

O envio de mensagens Ping é feito utilizando a ferramenta Ping normal, encontrada em diversos sistemas operacionais como Linux e Windows, e que pode ser executada via terminal. Dessa forma, em qualquer lugar da Internet se pode enviar uma mensagem para os dispositivos aquáticos. A ferramenta Ping tem uma funcionalidade que permite alterar os dados que serão "ecoados" na rede. Ou seja, é possível modificar o campo Carga (*Payload*). Para isto, basta executar o comando com o parâmetro "-p" seguido pelos valores com que se deseja preencher o campo, em hexadecimal, tendo um máximo de 16 bytes configuráveis. Utilizamos esta funcionalidade para possibilitar a identificação da mensagem que deveria ser enviada a rede aquática. Criamos um padrão que indica que a mensagem deve ser encaminhada à rede aquática, contendo também o ID do dispositivo destinatário. Portanto, para disparar um *ping* tendo como destino um dispositivo aquático, deve ser acrescentado o parâmetro "-p" com o seguinte padrão: "0FC000*ii*" onde cada *i* equivale a um dígito hexadecimal do ID do dispositivo, sendo o *i* mais à direita o menos significativo. Por exemplo, a execução do comando Ping tendo como destino um dispositivo aquático cujo ID é 15 deve ser assim: `ping <addr> -p 0FC0000F`. Onde *addr* é o endereço IP do computador conectado ao *gateway* da rede aquática.

Desenvolvemos um *software*, que chamamos de *ICMPforwarder*[1], que fica em execução no computador *gateway*. Esta aplicação, desenvolvida utilizando a Linguagem C de programação, é encarregada de verificar o padrão nos dados e encaminhar a mensagem para a rede aquática. O *gateway*, por sua vez, ao executar o processo de compressão, adicionará o ID recebido no padrão ao campo de dados da mensagem comprimida. Assim, um nó aquático saberá se é o destinatário da mensagem. Esta medida possibilita a operação em redes que não oferecem nenhum tipo de suporte ao protocolo IP.

4. Testes e Resultados

Para realização dos testes foi montada uma topologia. A topologia consiste em um computador, o *gateway* e um nó aquático. O processador presente no *gateway* é um

[1]https://github.com/franciscomilagres/PingoDagua

MSP430F247 e o dos dispositivos aquáticos, um MSP430F2274, ambos fabricados pela Texas Instruments®. O computador utilizado para execução da aplicação *ICMPforwarder* possui um processador Intel® Core *i7*®, 8GB de memória RAM e o seu sistema operacional é o Ubuntu® 14.04 LTS.

4.1. Testes sem ter como destino um dispositivo aquático

Primeiro testamos o Pingo d'água para demonstrar que o *ping* tradicional continua funcionando. As Figuras 5 e 6 ilustram a execução de um comando *ping* em uma máquina remota, sem ter como destino um dispositivo aquático. Como podemos ver, o *ICMPforwarder* identificou que a mensagem não deve ser encaminhada para a rede aquática via porta USB (figura 6) do *gateway*. A mensagem *Echo Request* recebida não é respondida e, a máquina com nome Eufrates, utilizada nos testes (figura 5), indica corretamente que nenhum pacote enviado foi respondido por dispositivo aquático.

```
eufrates:~> ping belohorizonte.lecom.dcc.ufmg.br -c 1
PING belohorizonte.lecom.dcc.ufmg.br (150.164.2.83) 56(84) bytes of data.

--- belohorizonte.lecom.dcc.ufmg.br ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

Figura 5. Execução do Ping em um computador.

```
Listening...
===4: ICMP - Received ===

0:  45 0 0 54 83 29 40 0 3f 1 82 bf 96 a4 6 25
10:  96 a4 2 53 8 0 7 bd 36 a8 0 1 80 7a 2f 58
20:  0 0 0 0 3d f4 d 0 0 0 0 0 10 11 12 13
30:  14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23
40:  24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33
50:  34 35 36 37
IPv4: hdr-size=20 pkt-size=84 protocol=1 TTL=63 src=150.164.6.37 dst=150.164.2.83
ICMP: type[8/0] checksum[1981] id[13992] seq[1]
Not for RS232...
```

Figura 6. Ping recebido e não redirecionado.

4.2. Testes tendo como destino um dispositivo aquático

Nas Figuras 7 e 8 temos o disparo de um comando *ping*. Desta vez, tendo como destino um dispositivo aquático (figura 7). O *ICMPforwarder*, como mostrado na Figura 8, reconhece o padrão na mensagem e a encaminha para a porta USB do *gateway*, através do protocolo RS232. Ele aguarda a resposta do *gateway* e assim que recebe a mensagem de resposta do dispositivo aquático, o *ICMPforwarder* a envia para o remetente da mensagem.

O parâmetro "-c" recebido pelo comando *ping* indica quantas mensagens *Echo Request* deseja-se enviar. Vários testes foram feitos. Para facilitar o registro e visualização, realizamos estas imagens mostrando os testes enviando apenas uma mensagem.

O *gateway*, durante a execução desta tarefa, realizou a compressão da mensagem e a enviou a um dispositivo que compõe um nó aquático suportando o formato comprimido do Ping. Este, respondeu ao primeiro, que descomprimiu e encaminhou ao computador conectado.

```
eufrates:~> ping belohorizonte.lecom.dcc.ufmg.br -c 1 -p 0fc00001
PATTERN: 0x0fc00001
PING belohorizonte.lecom.dcc.ufmg.br (150.164.2.83) 56(84) bytes of data.
64 bytes from belohorizonte.lecom.dcc.ufmg.br (150.164.2.83): icmp_seq=1 ttl=63 time=300 ms

--- belohorizonte.lecom.dcc.ufmg.br ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 300.653/300.653/300.653/0.000 ms
```

Figura 7. Disparo do Ping com o parâmetro "-p".

```
===4: ICMP - Received ===

0:  45 0 0 54 f5 6 40 0 3f 1 10 e2 96 a4 6 25
10:  96 a4 2 53 8 0 f3 8 36 bc 0 1 f7 7a 2f 58
20:  0 0 0 0 3 dc 6 0 0 0 0 0 f c0 0 1
30:  f c0 0 1 f c0 0 1 f c0 0 1 f c0 0 1
40:  f c0 0 1 f c0 0 1 f c0 0 1 f c0 0 1
50:  f c0 0 1
IPv4: hdr-size=20 pkt-size=84 protocol=1 TTL=63 src=150.164.6.37 dst=150.164.2.83
ICMP: type[8/0] checksum[62216] id[14012] seq[1]

Forwarding to RS232.
-DONE...
Waiting for COM answer...
===16: COM - Received ===
Forwarding answer to host...
-DONE...
----------------------
```

Figura 8. Ping recebido e encaminhado ao *Gateway*.

4.3. Avaliação de Desempenho

Avaliamos o Pingo d'água com relação ao consumo energético, quantidade de bytes transmitidos e perda de pacotes na rede.

O gráfico presente na Figura 9 mostra a diferença de consumo energético entre transmissões contendo as mensagens comprimidas e transmissões sem compressão. Os valores de dados e cálculos utilizados para construção deste gráfico são semelhantes aos do trabalho de [Almeida et al. 2016], utilizando um modem acústico transmitindo

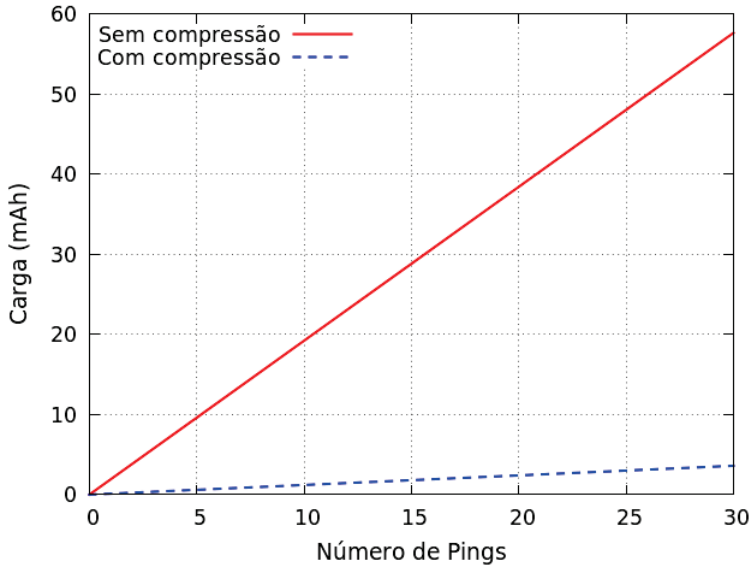


Figura 9. Consumo energético do tráfego na Rede Aquática.

na frequência de 33,8kHz. O consumo energético do modem é de 0,03 mAh por byte transmitido. A transmissão de uma mensagem ICMP com tamanho padrão e sem a nossa compressão exige uma quantidade de carga elétrica de 1,92 mAh, enquanto a transmissão de uma mensagem comprimida requer 0,12 mAh, consumindo 93,75% a menos. Ambas as curvas tem dependência linear com relação ao número de pings disparados. Mesmo assim, o consumo de energia com a utilização de nossa proposta é claramente menor.

Na Figura 10 está ilustrado o tráfego ICMP (em *bytes*) que ocorre na rede aquática, com a transmissão de mensagens ICMP e com a transmissão das mensagens comprimidas. A economia alcançada pela nossa compressão foi de aproximadamente 60 bytes por mensagem, dado que o tamanho padrão das mensagens ICMP trocadas pelo comando Ping é de 64 bytes. O desvio padrão (e intervalo de confiança) é desprezível pela quantidade de execuções e baixa variabilidade na compressão de dados tendo sido, por isso, omitido.

A Figura 11 apresenta a diferença na quantidade de pacotes perdidos entre transmissões compostas por mensagens comprimidas e transmissões compostas por mensagens ICMP no tamanho padrão, contando, neste caso, 3 pacotes por mensagem. O modem utilizado nos testes possui uma BER (*Bit Error Rate*) de 0,15% segundo a sua especificação, o que é responsável por gerar uma taxa de perda de pacotes de 4,69% para mensagens comprimidas e de 25,04% para as mensagens ICMP sem compressão, transmitidas em pacotes de, no máximo, 24 *bytes*.

5. Conclusão e Trabalhos futuros

Neste trabalho, apresentamos o Pingo d'água, uma versão com compressão de cabeçalho para o protocolo ICMP, desenvolvida para a ferramenta Ping funcionar em ambientes com comunicação aquática. Mensagens ICMP enviadas de qualquer computador da Internet podem alcançar os dispositivos aquáticos, e estes responderem de volta. A solução apresentada pode ser utilizada, por exemplo, em monitoração de lagos.

Através de um dispositivo intermediário, um *gateway*, as mensagens do proto-

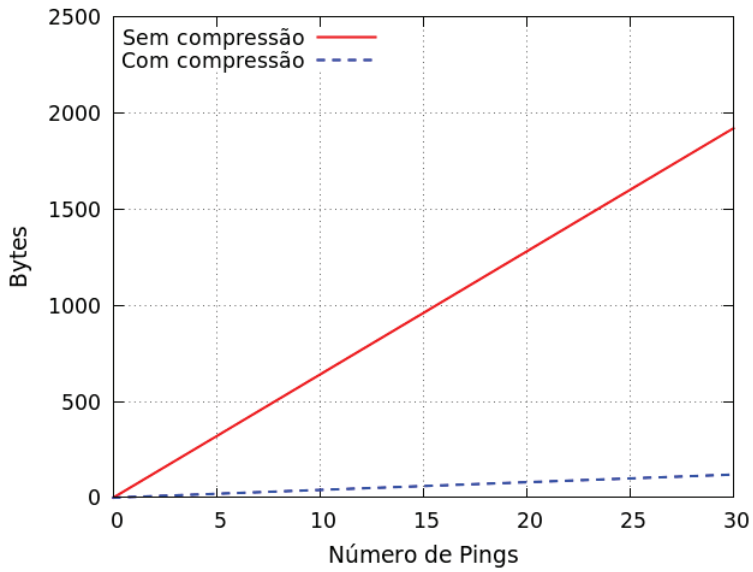


Figura 10. Tráfego ICMP na Rede Aquática.

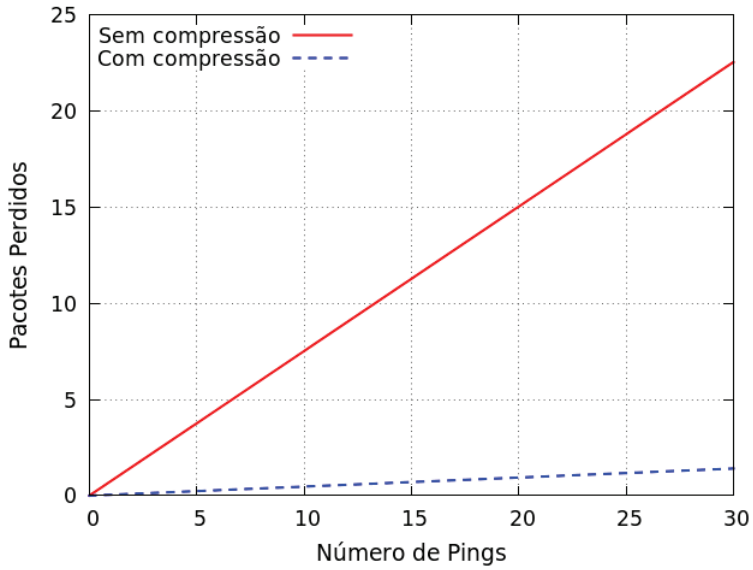


Figura 11. Perda de Pacotes na Rede Aquática.

colo ICMP enviadas de qualquer parte da Internet são comprimidas e transmitidas para os dispositivos aquáticos, cuja comunicação é limitada devido ao pouco poder computacional dos dispositivos e às características peculiares da comunicação aquática, como alta latência, baixa taxa de transmissão, alta taxa de perda de pacotes e outras, que se transformam em obstáculos para a criação da Internet das Coisas Aquáticas.

Em cenários como o descrito, é de fundamental importância possibilitar uma redução dos dados trafegados. Por isso, a compressão proposta pode ser um grande ganho se comparada à utilização do protocolo ICMP original, para o alcance da IoUT. Como proposto, há interoperabilidade tanto com o protocolo ICMP quanto com a versão ICMPv6.

Os resultados confirmam que os objetivos foram alcançados. Foi apresentado neste trabalho o envio de mensagens da Internet para dispositivos aquáticos a fim de facilitar a sua monitoração, sem necessitar modificar a arquitetura da pilha de protocolos TCP/IP ou mesmo ter que instalar drivers ou softwares adicionais. O trabalho apresentado tornou possível o acesso a uma rede sem fio aquática remotamente utilizando mensagens ICMP. Os testes mostraram a viabilidade e eficiência do protocolo proposto, permitindo a criação da Internet das Coisas Aquáticas.

Como trabalhos futuros podemos também operar com os demais cabeçalhos da pilha de protocolos da arquitetura TCP/IP, sobre o *6LoWPAN*. Também podemos utilizar uma versão do *6LowPAN* com baixo consumo de memória, como em [Santos et al. 2013]. Outras possibilidades são: dispensar a necessidade de parâmetros adicionais ao comando Ping em redes já atingíveis pela Internet; a integração com o protocolo DNS para o envio de mensagens por nome no lugar de identificadores; a adaptação do trabalho para operação com o *Wireshark*®; a inclusão de uma política de roteamento na rede, para testar a circulação das mensagens entre os nós; ou o uso de um mecanismo de endereçamento hierárquico com múltiplos saltos [Peres et al. 2016] que facilite o encaminhamento de mensagens.

Referências

Almeida, T. T., Bragança, L., Antônio, R. A. A. A., Lima, F., Vieira, M. A. M., Júnior, J. G. R., Vieira, L. F. M., and Nacif, J. A. (2016). Avaliação de desempenho na comunicação em redes acústicas aquáticas utilizando modems reais de baixo custo. In *Simpósio Brasileiro de Computação Ubíqua e Pervasiva (SBCUP)*. SBC.

Caruso, A., Paparella, F., Vieira, L. F. M., Erol, M., and Gerla, M. (2008). The meandering current mobility model and its impact on underwater mobile sensor networks. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 221–225.

Choi, H., Kim, N., and Cha, H. (2009). 6LoWPAN-SNMP: Simple network management protocol for 6LoWPAN. In *High Performance Computing and Communications, 2009. HPCC '09. 11th IEEE International Conference on*, pages 305–313.

Conta, A. and Gupta, M. (2006). Internet control message protocol (ICMPv6) for the internet protocol version 6 (IPV6) specification.

Coutinho, R. W., Boukerche, A., Vieira, L. F., and Loureiro, A. A. (2015). Modeling and analysis of opportunistic routing in low duty-cycle underwater sensor networks. In *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 125–132. ACM.

Coutinho, R. W., Boukerche, A., Vieira, L. F., and Loureiro, A. A. (2016a). Geographic and opportunistic routing for underwater sensor networks. *IEEE Transactions on Computers*, 65(2):548–561.

Coutinho, R. W., Boukerche, A., Vieira, L. F., and Loureiro, A. A. (2016b). On the design of green protocols for underwater sensor networks. *IEEE Communications Magazine*, 54(10):67–73.

Coutinho, R. W., Boukerche, A. F., Vieira, L., and Loureiro, A. (2016c). A novel centrality metric for topology control in underwater sensor networks. In *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 205–212. ACM.

Coutinho, R. W., Vieira, L. F. M., and Loureiro, A. A. F. (2013). DCR: Depth-controlled routing protocol for underwater sensor networks. In *2013 IEEE Symposium on Computers and Communications (ISCC)*, pages 453–458. IEEE.

Coutinho, R. W. L., Boukerche, A., Vieira, L. F. M., and Loureiro, A. A. F. (2014). GEDAR: geographic and opportunistic routing protocol with depth adjustment for mobile underwater sensor networks. In *2014 IEEE International Conference on Communications (ICC)*, pages 251–256. IEEE.

Coutinho, R. W. L., Boukerche, A., Vieira, L. F. M., and Loureiro, A. A. F. (2016d). Design guidelines for opportunistic routing in underwater networks. *IEEE Communications Magazine*, 54(2):40–48.

Erol, M., Vieira, L. F., Caruso, A., Paparella, F., Gerla, M., and Oktug, S. (2008). Multi stage underwater sensor localization using mobile beacons. In *Sensor Technologies and Applications, 2008. SENSORCOMM'08. Second International Conference on*, pages 710–714. IEEE.

Erol, M., Vieira, L. F., and Gerla, M. (2007a). Localization with Dive'N'Rise (DNR) beacons for underwater acoustic sensor networks. In *Proceedings of the second workshop on Underwater networks*, pages 97–100. ACM.

Erol, M., Vieira, L. F. M., and Gerla, M. (2007b). AUV-aided localization for underwater sensor networks. In *Wireless Algorithms, Systems and Applications, 2007. WASA 2007. International Conference on*, pages 44–54. IEEE.

Kushalnagar, N., Montenegro, G., and Schumacher, C. (2007). IPv6 over low-power wireless personal area networks (6lowpans): overview, assumptions, problem statement, and goals. Technical report.

Lee, U., Wang, P., Noh, Y., Vieira, L. F. M., Gerla, M., and Cui, J.-H. (2010). Pressure routing for underwater sensor networks. In *INFOCOM 2010. The 29th Conference on Computer Communications. IEEE*, pages 1676–1684.

Ng, H. H., Soh, W. S., and Motani, M. (2013). A bidirectional-concurrent mac protocol with packet bursting for underwater acoustic networks. *IEEE Journal of Oceanic Engineering*, 38(3):547–565.

Peres, B. S., Souza, O. A. d. O., Santos, B. P., Junior, E. R. A., Goussevskaia, O., Vieira, M. A. M., Vieira, L. F. M., and Loureiro, A. A. F. (2016). Matrix: Multihop address allocation and dynamic any-to-any routing for 6LoWPAN. In *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 302–309. ACM.

Postel, J. (1981). [RFC792] Internet control message protocol.

Santos, E. R. d. S., Vieira, M. A. M., and Vieira, L. F. M. (2013). Routing IPv6 over wireless networks with low-memory devices. In *Personal Indoor and Mobile Radio Communications (PIMRC), 2013 IEEE 24th International Symposium on*, pages 2398–2402. IEEE.

Sun, Y. and Melodia, T. (2013). The internet underwater: An ip-compatible protocol stack for commercial undersea modems. In *Proceedings of the Eighth ACM WUWNET*, WUWNet '13, pages 37:1–37:8, New York, NY, USA. ACM.

Viana, S. S., Vieira, L. F., Vieira, M. A., Nacif, J. A. M., and Vieira, A. B. (2015). Survey on the design of underwater sensor nodes. *Design Automation for Embedded Systems*, pages 1–20.

Vieira, L., Loureiro, A., Fernandes, A., and Campos, M. (2010a). Redes de sensores aquáticas. *XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, Gramado, RS, Brasil*, 1:199–240.

Vieira, L. F. M. (2012). Performance and trade-offs of opportunistic routing in underwater networks. In *2012 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 2911–2915. IEEE.

Vieira, L. F. M., Kong, J., Lee, U., and Gerla, M. (2006). Analysis of aloha protocols for underwater acoustic sensor networks. *Extended abstract from WUWNet*, 6.

Vieira, L. F. M., Lee, U., and Gerla, M. (2010b). Phero-trail: a bio-inspired location service for mobile underwater sensor networks. *IEEE Journal on Selected Areas in Communications*, 28(4):553–563.