The Internet Underwater: An IP-compatible Protocol Stack for Commercial Undersea Modems

Yifan Sun
Department of Electrical Engineering
State University of New York at Buffalo
ysun24@buffalo.edu

Tommaso Melodia
Department of Electrical Engineering
State University of New York at Buffalo
tmelodia@eng.buffalo.edu

Abstract

Recent underwater sensor network research has focused on developing physical, medium access control, and network layer protocols to enable high data rate, energy-efficient and reliable acoustic communications. However, it is now essential to design and standardize architectures that will enhance the usability and interoperability of underwater networks.

This paper proposes a networking architecture to efficiently provide interoperability with traditional TCP/IP protocol stacks for commercial underwater modems. The proposal is based on an adaptation layer located between the data link layer and the network layer, such that the original TCP/IP network and transport layers are preserved unaltered to the maximum extent. The adaptation layer performs header compression and data fragmentation to guarantee energy efficiency. Furthermore, the proposed architecture includes mechanisms for auto-configuration based on router proxies that can avoid human-in-the-loop and save energy when broadcast is needed. The proposed architectural framework was implemented as a Linux device driver for a commercial underwater network modem SM-75 by Teledyne Benthos. Testing and simulation results illustrate that the architecture efficiently provides interoperability with TCP/IP.

1 Introduction

Underwater networks have been the object of intense research by the communications, signal processing, and networking communities in the past few years. Protocols at different layers have been designed to meet the challenges of underwater networking [1]. However, underwater devices can not yet be regarded as citizens of the traditional networking realm because of their limited interoperability with existing networks based on the TCP/IP architecture. A similar situation was also experienced by wireless sensor networks, until IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) [2] became part of the Internet suite of standardized protocols. It is now essential to design, deploy, and test a standardized protocol stack that can efficiently provide interoperability for underwater networking devices.

While one can envision a scenario where users can address and access underwater nodes from any Internet-connected terminal, including workstations or smartphones, as of today existing underwater acoustic sensor network (UW-ASN) cannot be reconfigured or reprogrammed once the modems have been deployed. On the contrary, with support for the TCP/IP protocol stack, applications such as FTP and SSH could be utilized to reconfigure the network. Similarly, monitoring applications (e.g., traceroute) could also be used to diagnose network problems.

There are several challenges that need to be addressed to provide smooth interoperability with TCP/IP networks. Underwater networks are in fact unique in that they combine the challenges of severely limited resources of wireless sensor network (WSN) and very long propagation delays of delay-tolerant networks (DTN) [3]. The main challenges from a protocol perspective lie in the following aspects:

- The long propagation delays of underwater networks [1] require protocols to be delay-tolerant. However, TCP/IP is not designed to be such.
- Compared to traditional RF wireless devices, underwater modems consume much more energy for transmission than for reception and idle listening [4]. Instead of the traditional energy consumption structure $E_T > E_R \approx E_L > E_P$, where E_T, E_R, E_L and E_P represent the energy consumption for transmitting, receiving, idle listening and processing, the energy consumption structure is like $E_T \gg E_R \approx E_L > E_P$. Since batteries are, in many cases, not rechargeable, energy saving is a high priority. Therefore, architectural principles should be informed by energy saving schemes designed based on these premises.
- TCP/IP is designed for reliable channels and high-performance communication. However, the underwater channel is time-varying and demands robust and channel-aware packet routing, forwarding and fragmentation schemes.

Acknowledgment: This work was partially supported by the National Science Foundation under grants CNS-1055945 and CNS-1126357.

WUWNET'13, November 11–13, 2013, Kaohsiung, Taiwan.

http://dx.doi.org/10.1145/2532378.2532407

Existing protocols solve some of the aforementioned challenges through ad-hoc design. However, most of them regard the underwater network as a system isolated from the Internet. Tunneling at a border router is usually required and assumed implicitly to bridge protocols on both sides. However, tunneling results in unnecessary overhead and may degrade the performance due to inconsistency between protocols and lack of full session information, e.g., information about congestion and delay. Moreover, existing protocols use proprietary addressing schemes. As a consequence, traditional devices can not access underwater nodes with exiting tools based on TCP/IP protocols.

To address these problems, Hui et al. [5] proposed the 6LoWPAN architecture for wireless sensor network, which provides an IPv6 protocol stack for resource-constrained devices. 6LoWPAN provides a solution for header compression, data fragmentation and auto-configuration. However, Hui's architecture is designed for terrestrial networks and is not optimized for the underwater energy consumption structure. Moreover, the data fragmentation that fits the long IPv6 packet (at least 1280 Bytes) into fixed-sized IEEE 802.15.4 frames (127 octets) is not suitable for time-variant underwater channels. The high propagation delay of underwater networks and the channel characteristics require optimized, channel-dependent frame size to increase channel utilization and energy efficiency [6, 7].

Another candidate architecture, delay-tolerant networking (DTN) was proposed by Fall et al. [3]. DTN is designed to tackle high latencies and disconnections. However, the introduction of an adaptation layer between application layer and transport layer makes it difficult to store state information from physical layer and the medium access control (MAC) layer, so that it is impossible to guarantee energy efficiency.

Therefore, we propose a novel architecture designed to address the aforementioned problems. The major contributions of this paper are as follows:

- We propose a network architecture for underwater acoustic sensor networks. The architecture is fully compatible with the traditional TCP/IP protocol and supports both IPv4 and IPv6.
- We propose a software architecture that cooperates with existing operating systems and can be easily reconfigured to cooperate with different underwater modems.
- The proposed architecture is optimized for underwater networks by introducing header compression, data fragmentation and router proxy.

The rest of the paper is structured as follows. Related work is discussed in Section 2. We introduce the system architecture and some basic assumptions in Section 3. Sections 4, 5, 6 and 7 discuss different aspects of the adaptation layer design, including frame size optimization, header compression, fragmentation, neighbor detection, autoconfiguration, routing and forwarding. Finally, performance evaluation results are discussed in Section 8.

2 Related Work

An increasing number of protocols have been proposed for underwater networks in the past ten years. However, most of them focus on designing protocols and algorithms at the data link or network layer without considering the whole protocol stack. Moreover, existing protocols are usually application-specific. For example, oceanographic data collection sensor networks usually assume a static topology and are delay insensitive, while disaster prevention systems are optimized for rare communication but delay sensitive cases. Also, the core challenges of network discovery, naming, addressing, configuration and diagnosing are not thoroughly discussed in recent underwater networking research.

Providing interoperability with the Internet requires an architectural framework. Candidate architectures include IP over 802.15.4 [5] and DTN [8]. Hui et al. proposed a 6LoWPAN-based wireless sensor network architecture that brings IPv6 to 802.15.4 networks [5]. The core idea of 6LoWPAN is to fragment long IPv6 packets to fit in much smaller 802.15.4 frames as well as compressing the transport and network layer headers. We use similar compression and fragmentation methods to create a frame format better suited for the underwater scenario. Furthermore, we enhance performance and efficiency for underwater channels by finding the optimal packet size through mathematical optimization techniques.

The DTN architecture introduces a robust solution to address long delays and predictable disconnections. The core of this architecture is the Bundle Protocol [9], which is an adaptation layer between the application layer and the transport layer. However, higher layer protocols cannot easily gather and leverage information from the physical and data link layers, which may lead to poor performance in underwater networks. Also, the requirements of application-layer softwares and the unique Endpoint Identifiers (EID) limit the interoperability of the architecture, which is again not designed for energy saving.

SEAWEB [10] is an existing experimental deployment of underwater networks. It defines its physical, link and network layers of underwater network. However, the system uses a specialized repertoire of communication gateways, which significantly limit its interoperability. SEAWEB requires human intervention to setup and monitor the system at the gateway. On the contrary, we propose an architecture that interfaces and makes the underwater network remotely accessible from the Internet.

3 System Architecture

We consider underwater networks composed of underwater nodes and border routers. Border routers are specialized nodes that are carried on ships, on buoys or onshore and provide Internet connectivity through cellular or satellite communications. Underwater nodes are responsible for collecting, processing and transmitting data, and can be configured and accessed from remote hosts. Border routers operate at the network layer and bridge the underwater sensor network segment with the traditional IP-based network.

There may be multiple border routers in an underwater network. We assume that each node is able to find the best border router, i.e, the border router that leads to minimal energy consumption. Hence, underwater nodes may be naturally divided into subnets based on different physical layer

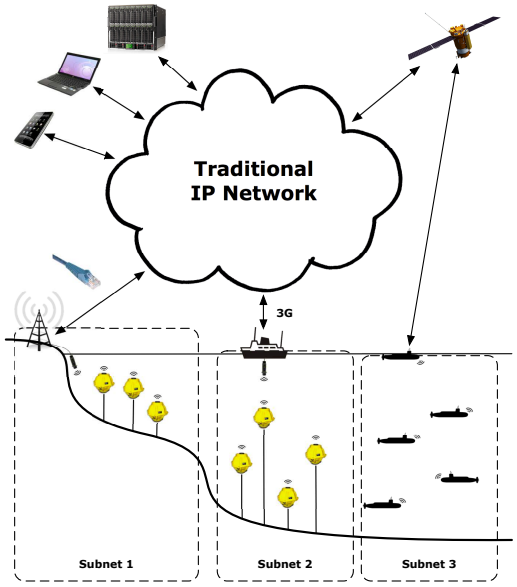

Figure 1. System Architecture.

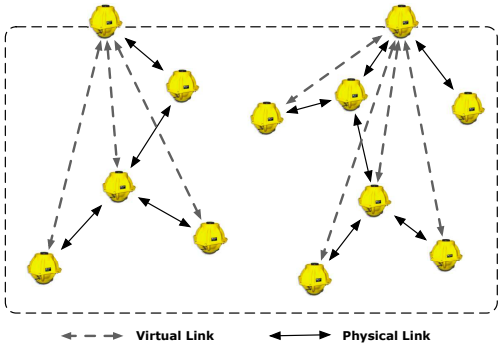

Figure 2. Interior Network Abstraction.

protocols or different border routers they are connected to. Dividing the network into subnets may save energy and reduce delay by limiting the maximum number of underwater hops. Furthermore, diagnosing may be made easier because errors are limited to one subnet and are therefore easier to locate.

3.1 Interior Network Abstraction

Since underwater sensor networks are usually sparsely deployed, a node may not be able to communicate with the border router through a direct acoustic link. Even if a node is within transmission range of a border router, forwarding through an intermediate node can significantly reduce energy consumption and interference to other nodes. Therefore, all nodes cooperate to route and forward data packets to the border router through multi-hop routes.

Different from IP routing, routing in the underwater subnet, which is performed beneath the network layer, is also defined as "mesh routing" or "layer-2" routing. The layer beneath the network layer maintains its own neighboring and routing tables and forwards data packets to the corresponding next hop. After a packet arrives at the destination, the

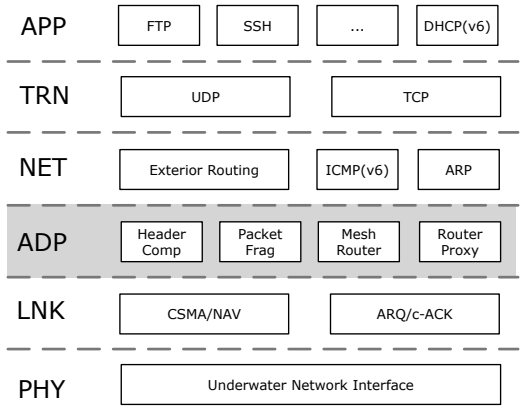

Figure 3. Layered Network Architecture.

packet is handed over to the network layer and the network layer may process it or forward it to another node. Mesh routing is combined with traditional routing at the border router, and packets are forwarded at the network layer to the next hop. We use the term "L2 Hop" to describe a mesh routing hop, where the source and destination are within each other's acoustic transmission range, and "L3 hop" to describe the traditional IP-level forwarding hop. Thus, from the network layer perspective, at each underwater node there is a *virtual link* established between the node and the border router, as illustrated in Fig. 2. Therefore, the data link layer regards only nodes within acoustic signal range as neighbors, while the network layer ignores nodes on the path and only treats the border router as a neighbor.

The introduction of mesh routing keeps clean and simple neighboring and routing tables at the network layer since most of the nodes only have information on the border router. At the same time, any routing algorithm at the data link layer can be implemented according to application requirements. From the network layer perspective, this is similar to a typical Wi-Fi segment, where an Access Point provides Internet connectivity to all the terminals in the subnet. From the perspective of the data link layer, the whole system is a typical underwater sensor network and application specific protocols and algorithms can be adopted.

3.2 Network Architecture

The system follows the traditional TCP/IP five-layer model. Our objective is to design a network architecture compatible with different physical and data link layer protocols. An adaptation layer is therefore inserted to interface the traditional IP network layer with a custom designed link layer. The adaptation layer serves as the core of the architecture. It is responsible for header compression, packet fragmentation, mesh routing and broadcast emulation. By utilizing cross layer header information, the long IPv4 or IPv6 header can be shortened, thus significantly saving energy when maintaining connectivity or exchanging information with small packets. For longer packets, fragmentation guarantees energy efficiency by balancing header overhead and retransmission overhead. Mesh routing is also performed at this layer and a packet is forwarded at intermediate nodes. Also, routing information exchange, such as router adver-

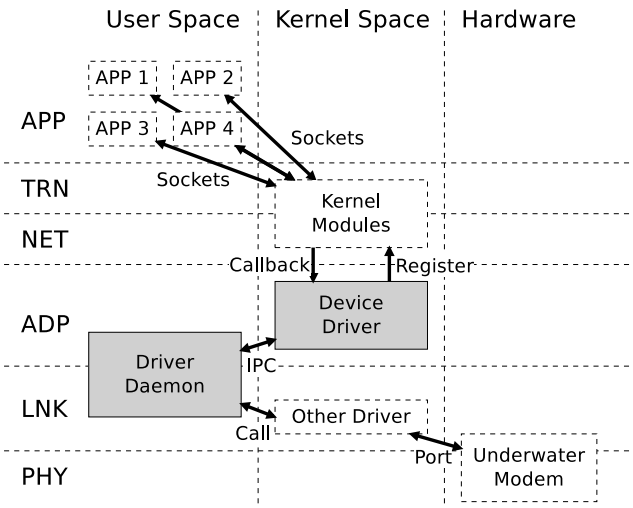

Figure 4. Software Implementation Architecture.

tisement, requires the capability of broadcasting over one L3 hop. To emulate L3 broadcast, nodes serve as router proxies to pass router information to terminals.

The architecture is designed to maintain the traditional TCP/IP protocol stack unaltered. In this way, the whole architecture can easily inter-operate with existing platforms. Thus, the network layer is kept unchanged and is responsible for L3 routing. Internet Control Message Protocol version six (ICMPv6) provides plenty of control and diagnosing functions as well as neighbor and router discovery capabilities. ICMP plus Address Resolution Protocol (ARP) can do similar work in an IPv4 environment. ICMP protocols works together with application layer DHCP or DHCPv6 protocols to provide auto-configuration. Moreover, the transport layer also provides user applications with TCP or UDP sockets.

3.3 Software Architecture

We implemented the proposed architecture as a Linux driver as shown in Fig. 4. Since we do not modify the TCP/IP transport and network layers, we benefit from the robustness of the Linux networking core. Data packets from the application layer are handled by the transport and network layers in the Linux kernel before they are passed to the adaptation layer. Our driver works at the adaptation layer and is divided into two parts. The driver part runs in Kernel mode and is responsible for device-independent functions such as fragmentation, router proxy and header compression. A second driver component runs in user mode so that it can be easily reconfigured to communicate with devices from different vendors. The inter-process communication (IPC) between these two components is achieved through Netlink Socket [11]. The data-link layer may be either defined in the daemon or in the modem itself.

By dividing the system driver into two parts, we provide the architecture with the flexibility to operate on different devices while at the same time keeping the architecture design coherent. The kernel portion of the driver is device-independent, and therefore the adaptation layer and higher layer protocols remain the same no matter what modem is used. Instead, the device-dependent part of the driver runs

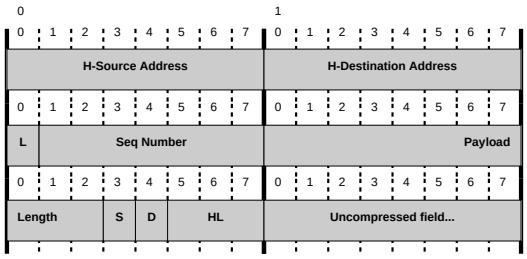

Figure 5. Frame Header Compression.

in the user space. Thus, it can be designed based on high level programming languages without the need to consider the complex challenges of Kernel programming.

4 Header Compression and Fragmentation

The header of the traditional IP protocol stack is designed for fast parsing and to be extensible. Also, different layers are kept separate. These design principles result in significant redundancy. Through header compression, the frame size can be significantly reduced, thus significantly enhancing efficiency, especially for small packets.

Since in each field of each layer header there exist common values, we can reduce the number of bits to represent those common values. If the field value does not fall into our common value set, we carry those fields after the header. For example, the *next header* field of IPv6 header is usually UDP, TCP or ICMPv6. Then, we can use 2 bits and assign 00 to UDP, 01 to TCP and 10 to ICMPv6. In this way, the 2-byte field is compressed into 2 bits. If a packet carries a different payload, we set this field to 11, and carry the original 2 bytes after the header. In addition, redundancy is further reduced by considering the headers of the 3 layers as a single "big header".

4.1 Frame Header Compression

The tasks of the frame header include marking the next hop address, helping reassemble the packet and carrying information for mesh routing. Although it is usually added to the frame at the data link layer, we add it in the adaptation layer since the adaptation layer maintains neighbor and mesh routing information.

The first two bytes represent hop-by-hop addresses. Instead of traditional 6-byte MAC addresses, we use a one byte hardware-like address. Hence, a subnet can contain up to 253 underwater devices (0 is for network identifier and 255 is reserved for broadcast). If more devices are involved in a deployment, one can divide them into different subnets each connected to a border router.

Hop-by-hop addresses are followed by fragmentation information. The first bit is a flag to mark if this frame is the last frame of a packet. The other 7 bits are for the sequence number of the packet, which assists the receiver with reassembly if frames do not arrive in order. As we set the minimum payload size to 32 bytes, the fragmentation can at most support a packet of $32 \times 2^7 = 4096$ bytes. We also use 11 bits to keep track of the payload length of the frame. The maximum transmit unit is $2^{11} = 2048$ bytes.

The last part is 6 bits long and keeps track of the routing information. Since the end-to-end source or destination

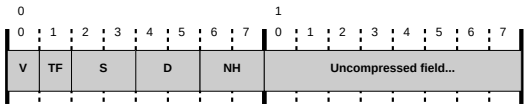

Figure 6. IPv6 Header Compression.

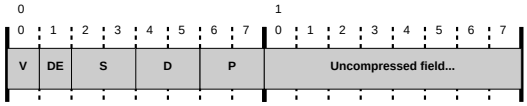

Figure 7. IPv4 Header Compression.

is usually the border router, we use one bit to denote if this field is the border router. If it is, we compress the 1 byte information into 1 bit only. Otherwise, we carry the hardware address in-line. We allocate 3 bits to "hop limit", which appears to be sufficient for most realistic deployments.

4.2 IPv6 Header Compression

By fully utilizing the common value of some fields, in the best case, we can compress the 40-byte IPv6 header into 1 byte. We use 1 bit for the version, to distinguish IPv4 and IPv6 header. Since the value of traffic class and flow label are usually 0, 1 bit is used to represent these 2 fields, otherwise they will be carried inline. The next 4 bits are assigned to denote source and destination addresses. For each address composed of 2 bits, the first bit is responsible for checking the status of IPv6 status. Supposing the value of this bit is 1, the IPv6 address prefix is the link-local prefix. The other bit is used to indicate if the IID can be derived from the hardware address. Even through an outbound packet does not use link-local prefix, nodes can use it before the packet is routed to the border router. Since the border router knows the global prefix, it can recover that field when forwarding packets to traditional IP networks. Since link-local prefix is used as one of the common values, the system benefits from greater probability of transmitting using link-local prefix. The next header field is reserved for UDP, TCP and ICMPv6.

4.3 IPv4 Header Compression

Similar to the IPv6 header, we can also compress the IPv4 header from 20 bytes to 1 byte in the best case. Sometimes IPv4 requires fewer bytes due to the shorter addresses. However, several functions have to be truncated. The first bit is still used to identify the version of the packet. Then, nodes can parse the compressed frame header. The next bit is used for the field Differentiated Services Code Point (DSCP) and Explicit Congestion Notification (ECN). The value of this byte is usually 0. Source and destination use 2 bits respectively. The first bit is to indicate if the subnet address is the same as the border router and the second is 0 if it is the same with the nodes hardware address. Also, the protocol field has the same usage as the next header field for IPv6 packet and reserved for UDP, TCP and ICMP.

The IPv4 packet header also provides other functions. By setting them to their default value, those fields can be elided. For example, IPv4 can fragment packets to fit the limited maximum transmit unit. As our adaptation layer is responsible for fragmentation, we set the fragmentation flag to be always 1, and prevent the IPv4 protocol from fragmenting the packet. Then, the fragmentation offset can also be elided. For UDP, TCP and ICMPv6 header compression, we use the same scheme as in 6LoWPAN.

5 Frame Size Optimization

Choosing the optimal frame size is important in underwater networks because of the high transmission energy consumption and the time-varying underwater channel. Transmitting a frame of correct length can avoid retransmissions in bad channels and can reduce the overhead in good channels.

5.1 Bit-Error-Rate Based Optimization

If the bit error rate is p_e and the frame size is l bits, the packet can be successfully transmitted with a probability of

$$p_s = (1-p_e)^l. \tag{1}$$

The frame size can be further divided into two parts and be expressed as $l = l_p + l_h$, where l_p and l_h are the length of payload and header in bits respectively.

Assuming the node transmit one frame and then stops to wait for an ACK, and retransmits an unlimited amount of times until the packet is successfully delivered, the average number of transmission is N^{TX}, and can be expressed as

$$N^{TX} = \frac{1}{(1-p_e)^l}. \tag{2}$$

Each frame is transmitted after an RTS and CTS handshake. We assume that an RTS and CTS packet consume E_r and E_c units of energy, respectively, and the energy consumption of ACK or NACK frame is E_a. In addition, we assume a data frame costs E_d units of energy for its preamble and E_b for each following bit. Then, we can express the energy consumption to transmit one frame as

$$E = E_r + E_c + N^{TX} \cdot (E_d + (l_p + l_h) \cdot E_b) + N^{TX} \cdot E_a, \tag{3}$$

and the energy efficiency as $\eta = \frac{l_p}{E}$ $[bit/J]$.

We would like to choose the right packet size to maximize the energy efficiency subject to some basic constrains. The problem can be expressed as

$$\begin{aligned} &\text{Given}: p_e, l_h, MTU, E_r, E_c, E_d, E_b, E_a \\ \underset{l_p}{\text{maximize}} \quad & \eta \\ \text{subject to} \quad & l_p > 0 \\ & l_p + l_h < MTU \\ & l_p \in 8 \times \mathcal{Z}. \end{aligned}$$

This optimization problem is an integer program, which is not easy to solve in general. However, the limited feasible set makes this problem easy to solve in practice by enumeration.

5.2 Bit Error Rate Estimation

Frame size optimization relies on a good estimate of the bit error rate (BER). Exchanging information between nodes introduces extra overhead and wastes energy. However, by observing the sequence of ACKs and NACKs, one can infer whether a frame has been delivered or not. Thus, we can define an array of the frame delivery history as

$$H = \{(l_1, s_1), \ldots, (l_n, s_n)\}, \tag{4}$$

where l is the frame length in bits, and s represents whether or not the frame has been successfully delivered with +1 and -1, respectively. A frame is recorded as success if an ACK is received, or as fail if NACK is received or if there is an ACK/NACK timeout. We can then define a maximum likelihood estimator to find the BER with the highest likelihood. The likelihood can be expressed as

$$P(p_e|H) = \frac{P(H|p_e)P(p_e)}{P(H)}, \tag{5}$$

where

$$P(H|p_e) = \prod_{i=1}^{n} \left(\frac{1-s_i}{2} + s_i(1-p_e)^{l_i} \right). \tag{6}$$

In addition, we assume that $P(H)$ is independent of BER and that BER is uniformly distributed on $[0, 0.5]$, since we have no a priori information.

6 Neighbor Detection and Autoconfiguration

6.1 Neighbor Discovery in IPV6

IPv6 uses ICMPv6 for neighbor discovery and router discovery. Also, by maintaining a neighbor table, the IPv6 addresses of neighbor nodes are mapped to their hardware address. Routers use router advertisement to announce theirs existence periodically, and reply to router solicitations from nodes. Nodes also use neighbor solicitation and neighbor advertisement to exchange information. However, all these functionalities are defined within one IP hop.

If a mesh-under scheme is used, all the nodes are within one IP hop. However, a flooding-like broadcast in the whole underwater subnet can create excessive overhead. Instead of broadcasting the packets among the whole underwater subnet, we limit the router advertisement (RA) multicast to one underwater hop. Those nodes that received the RA packet become router proxies. If a node does not have the information of any router, it multicasts a router solicitation (RS) packet. When router solicitations are grabbed by a router proxy, the router proxy will unicast router information to the source of the router solicitation, together with some mesh routing information. Then, this node becomes a new router proxy and waits for further router solicitations from other nodes. Every router proxy will also request updated router information from other router proxies periodically.

Figure 8 illustrates this procedure. Alice does not have border router information, so she multicasts a router solicitation. Bob and Charlie, as router proxies, reply. Since Charlie is on the best route to the border router, Alice chooses Charlie, and then Alice becomes a new router proxy.

6.2 Autoconfiguration in IPv6

IPv6 defines both stateless and stateful autoconfiguration mechanisms. The latter one is the well known DHCPv6. Stateless autoconfiguration assigns a prefix and an IID to each node. The default prefix is fe80::/64 and the IID is based on the node's MAC address. Unlike IPv4, even when a routable IP address is assigned, each node will still have the link-local address. Before global prefix is assigned, each node locally generates an address with link-local prefix and IID derived from the hardware address. The global prefix is assigned by router advertisement packets, which can also

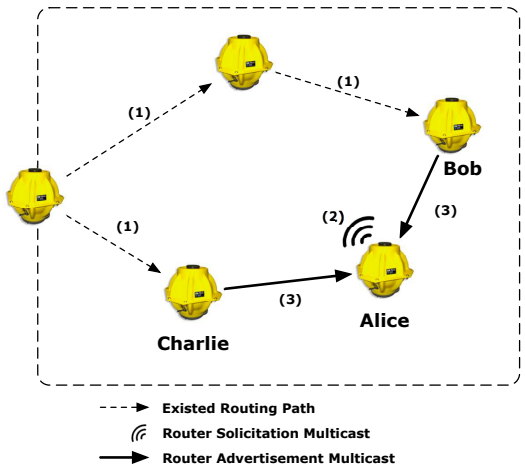

Figure 8. Autoconfiguration with Layer 2 Routing.

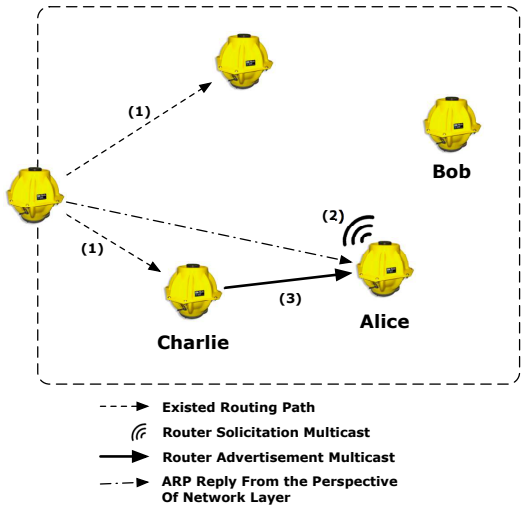

Figure 9. ARP with router proxy.

be sent from the router proxy in our architecture. The border router can also implement a DHCPv6 server, which provides flexibility and ensures no duplicated IP address in the whole underwater subnet exist. The border router can authorize each router proxy with a portion of the IP address space and let proxies serve as DHCPv6 server.

6.3 ARP in IPv4

IPv4 uses ARP for mapping a link layer address to an IP address. ARP is designed for broadcast within a single IP-hop, and is not to be routed inter-network. Since broadcast can involve excessive overhead, ARP is unsuited for underwater wireless networks.

One of the key points of mesh routing is to create virtual links between the border router and each node. Like typical WiFi networks, each node only regards the router as its neighbor and ignores the existence of the other nodes. Each node, if it knows router information, serves as a router proxy and can send ARP packets on behalf of the border router. Other nodes that do not have router information will only keep silent.

The ARP procedure is illustrated in Fig. 9. Alice sends an ARP request. Bob does not know where the border router is, so he keeps silent. Charlie, as a router proxy, replies with an ARP reply on behalf of the border router. Thus, from the network layer perspective, the ARP reply is from the border router and Charlie is not a neighbor of Alice.

6.4 DHCP in IPv4

The most serious problem of IPv4 is the address exhaustion problem, and Net Address Translation (NAT) is the state-of-art solution to this problem. Borrowed from IPv6, the idea of assigning IPv4 IID the same identifier as the node MAC/hardware address can be used for the stateless autoconfiguration. By doing so, each node is its own proxy and the adaptation layer replies a DHCP request message, so that broadcast is avoided. For mobile nodes, the adaption layer just sends that DHCP packet out to avoid address duplication.

7 Routing and Forwarding

7.1 Mesh Under vs. Route Over

Since the adaptation layer sits between the data link and the network layer, from an architectural perspective it can either hand all packets that need to be forwarded over to the network layer, or it can forward packets itself "under" the network layer. The case where the adaptation layer gives incoming packets to the network layer is referred to as *route-over*. On the contrary, we refer to *mesh-under* as the case where the adaptation layer forwards packets until they arrive at their destination.

Mesh-under is a so-called layer-2 solution to route from one node to the border router without the help of the IP layer. One of the most serious challenges is that it is difficult to find the global optimal route since the underwater segment does not have global state information about the other segments of the network. However, routing in the underwater segment of the network is typically much more "expensive" in terms of delay, energy consumption, and bandwidth. Since the cost of packet forwarding in terrestrial network is negligible, optimized routing in the underwater segment is typically the main priority.

As discussed before, since the whole underwater subnet can be represented with link-local addresses, the adaptation layer can better compress the header to significantly reduce the overhead. Furthermore, if the adaptation layer can monitor ICMPv6 or ARP packets to form its own routing table, there is no extra cost to enable mesh routing. Moreover, the idea of router proxy fits a mesh-under scheme better, since when a node sends a router solicitation but the router is too far away to receive it, other nodes cannot help to forward the router solicitation.

In the *route-over* scheme, each interface of any node is seen as a separate network and a routable IP address must be assigned and used to transmit packets. The advantages of route-over include that it can utilize network layer capabilities provided by the IP protocol. For example, ICMPv6 provides a full set of neighbor detection and router solicitation schemes that can help find the best route. Moreover, some well-known tools used to diagnose IP links, e.g., traceroute, can only be used with the network-layer routing. However, header compression will become much less efficient since link-local prefixes can only be used for single hop communication. Due to the lack of router proxy, when a node wants to fetch some configuration, packets have to be forwarded many times before they reach the border router. Basically, broadcasting in the whole underwater segment becomes nearly impossible or very complex. For these reasons, a mesh-under scheme was selected in our architecture.

8 Evaluation

To evaluate the IP compatible architecture, we implemented it in a Linux driver as described in Figure 4. It was tested on both PC and an ARM-based single-board computer Gumstix [12]. The device dependent part is designed to drive Teledyne Benthos [13] Telesonar SM-75 Modem with Modem Management Protocol (MMP) [14] over RS-232 Serial Port. In addition, for experiments that require a controllable and repeatable channel, we have developed a Java based subpacket level real-time underwater network simulator. The simulator uses the same parameters as the SM-75 modem.

We validated our driver by using it with existing applications. To test interoperability with the UDP protocol, we used a local area network (LAN) instant messaging software called IPTUX to exchange messages between two nodes. We were able to successfully transfer files between an server and a client with FTP protocol. Our architecture is implemented without major modification to the typical TCP and UDP protocols. The only parameters that need to be tuned are the initial retransmission timeout (RTO) of the TCP protocol, for one second [15] is too short for underwater communications even when routing and forwarding is not required. However, since the control algorithm of TCP is designed for fast networks, the round-trip-time estimation and congestion control algorithm may not be optimal for underwater network. For example, three-way handshaking wastes too much energy and time, and sessions can be easily terminated due to the poor channel conditions. Hence, UDP with application layer session and congestion control schemes may be a better option.

One of the most significant advantages of our architecture is the higher energy efficiency. The combined effect of the data fragmentation and streaming scheme saves energy by trading-off between header overhead and retransmission overhead. This is especially notable when large data packets are transmitted. Moreover, header compression reduces the overhead for short control packets as well as large data packets. The energy consumption for control and information exchange is further cut down by the routing proxy.

Since our fragmentation scheme heavily relies on a good estimate of the bit error rate, we first examine how well the estimator works. The simulation is performed in our simulator where the sender and the receiver are 1500 m apart. The application layer generates packets following a Poisson distribution; the packet arrival rate is much higher than the channel capacity. The sender transmits at a power of 3.4 W.

Figure 10 illustrates the estimated BER on a specific link. The estimator evaluates the BER with sufficient accuracy and is rather stable. In addition, the estimator finds the right order of BER in less than 10 transmissions and the estimated

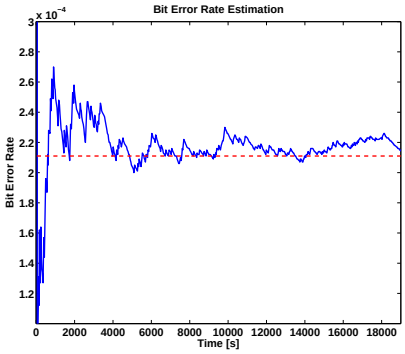

Figure 10. Estimation of the Bit Error Rate.

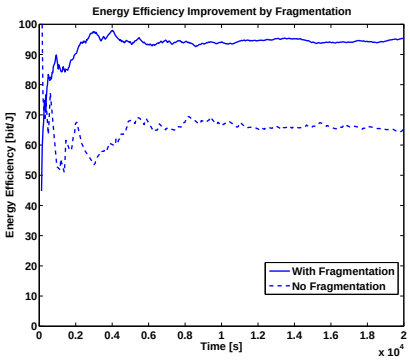

Figure 11. Energy Efficiency of Data Fragmentation.

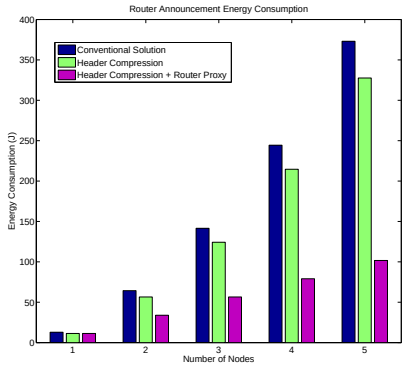

Figure 12. Energy Consumption for Information Exchange.

BER is in the range of 1×10^{-4} around the theoretical BER since about 30 transmissions. Figure 11 shows that the energy efficiency is increased by 50% when compared with non-fragmentation based schemes.

Next, we demonstrate the energy savings obtained through header compression and router proxy. We assume that there are n nodes in a line topology, and a node can only communicate with its nearest neighbor. The router is at one end. We calculate the energy consumption for all the nodes to get to know the router information using Router Advertisement and Router Solicitation. Figure 12 shows that header compression reduces the energy by shrinking the transmission size. When the number of hops increases, the router proxy shows its ability in saving energy. If the depth of the routing tree is greater than 5, the energy consumption is decreased by 70%.

9 Conclusions

We proposed a new underwater networking architecture that enables support for a traditional TCP/IP protocol stack. The proposal is based on an adaptation layer located between the data link layer and the network layer, such that the original TCP/IP network and transport layers are preserved unaltered. The adaptation layer performs header compression and data fragmentation to guarantee energy efficiency. Furthermore, the proposed architecture includes mechanisms for auto-configuration based on router proxies that can avoid human-in-the-loop and save energy when broadcast is needed. The proposed architectural framework was implemented as a Linux device driver for a commercial underwater network modem SM-75 by Teledyne Benthos. Testing and simulation results illustrate that the architecture efficiently provides interoperability with TCP/IP.

10 References

[1] T. Melodia, H. Kulhandjian, L. Kuo, and E. Demirors. Advances in underwater acoustic networking. In S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, editors, *Mobile Ad Hoc Networking: Cutting Edge Directions*, pages 804–852. John Wiley and Sons, Inc., Hoboken, NJ, 2013.

[2] N. Kushalnagar, G. Montenegro, and C. Schumacher. IPv6 over low power wireless personal area networks (6LowPAN): Overview, assumptions, problem statement, and goals. Technical report, RFC 4919, August 2007.

[3] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proc. of ACM SIGCOMM'03*, pages 27–34, Karlsruhe, Germany, August 2003.

[4] A. F. Harris III, M. Stojanovic, and M. Zorzi. When underwater acoustic nodes should sleep with one eye open: Idle-time power management in underwater sensor networks. In *Proc.of ACM Intl. Workshop on Underwater Networks*, pages 105–108, Los Angeles, CA, USA, September 2006.

[5] J. W. Hui and D. E.Culler. IP is dead, long live IP for wireless sensor networks. In *Proc. of ACM Conference on Embedded Networked Sensor Systems (Sensys)*, pages 15–28, Raleigh, NC, USA, Nov. 2008.

[6] D. Pompili, T. Melodia, and I. F. Akyildiz. Distributed Routing Algorithms for Underwater Acoustic Sensor Networks. *IEEE Trans. Wireless Communications*, 9(9):2934–2944, September 2010.

[7] S. Basagni, C. Petrioli, R. Petroccia, and D. Spaccini. Channel-aware Routing for Underwater Wireless Networks. In *Proc. of MTS/IEEE OCEANS 2012*, pages 1–9, Yeosu, South Korea, May 2012.

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

[9] S. Burleigh and K. Scott. Bundle protocol specification. Technical report, RFC 5050, November 2007.

[10] J. Rice and D. Green. Underwater acoustic communications and networks for the US navy's SEAWEB program. In *Proc. of IEEE Second International Conference on Sensor Technologies and Applications (SENSORCOMM)*, pages 715–722, August 2008.

[11] Kevin Kaichuan He. Kernel korner: Why and how to use netlink socket. *Linux Journal*, 2005(130):11–11, February 2005.

[12] Gumstix Inc. [Online]. Available: http://www.gumstix.com.

[13] Teledyne-Benthos, Acoustic Modems. [Online]. Available: http://www.benthos.com.

[14] H. Kulhandjian, L. Kuo, T. Melodia, D. A. Pados, and D. Green. Towards Experimental Evaluation of Software-Defined Underwater Networked Systems. In *Proc. of IEEE Underwater Communications Conf. and Workshop (UComms)*, Sestri Levante, Italy, Sep. 2012.

[15] V. Paxson and M. Allman. Computing TCPs retransmission timer. Technical report, RFC 2988, November, 2000.