An Architecture for Underwater Networks

Mandar Chitre
Acoustic Research Laboratory, National University of Singapore

Lee Freitag
Woods Hole Oceanographic Institution

Ethem Sozer
Massachusetts Institute of Technology

Shiraz Shahabudeen
Acoustics Research Laboratory, National University of Singapore

Milica Stojanovic
Massachusetts Institute of Technology

John Potter
Acoustic Research Laboratory, National University of Singapore

***Abstract* – As electromagnetic waves do not propagate well underwater, acoustics plays a key role in underwater communication. Due to significant differences in the characteristics of electromagnetic and acoustic channels, networking protocols for underwater systems differ from those developed for wired and wireless radio networks. Various schemes have been proposed for one or many aspects of underwater networks. However, no widely accepted common framework exists for underwater acoustics to unify these proposed schemes into a functional underwater network. The availability of such a framework will enable easy integration of independently developed techniques and thus accelerate the pace of research in underwater acoustic networking.**

In order for a common framework to be successful, it needs to have a wide acceptance. To gain such an acceptance, a framework needs to take into account a wide variety of different constraints and requirements for various underwater applications. This requires inputs from various research groups and end users. To help define the use cases and a common framework for underwater networking, a joint effort has been initiated between acoustic communication research groups at the Acoustic Research Laboratory (National University of Singapore), Woods Hole Oceanographic Institution and the Massachusetts Institute of Technology. In this paper, we discuss the first draft of the framework specifications from this effort. We welcome feedback from the underwater acoustic research community and potential end users of underwater networking systems.

I. INTRODUCTION

Communication between a set of underwater systems such as remote sensors, autonomous underwater vehicles and control vessels would enhance the effective use of such systems tremendously. As electromagnetic waves do not propagate well underwater, acoustics plays a key role in underwater communication. Due to significant differences in the characteristics of electromagnetic and acoustic channels, networking protocols for underwater systems differ from those developed for wired and wireless radio networks. As the sound waves are much slower than the electromagnetic waves, the latency in communication is typically much higher. Due to the multi-path propagation and ambient noise, the effective data rates are lower and packet loss rate is usually much greater. Comprehensive reviews of underwater acoustic communications are presented in [1][2][3].

In order to develop a fully functional network, several aspects of a protocol need to be defined. This typically includes modulation, synchronization, packet formatting, error correction, medium access control, addressing, routing, etc. Over the past few decades, various schemes have been proposed for one or many of these aspects of underwater networks (e.g. [4][5]). However, no widely accepted common framework exists for underwater acoustics to unify these proposed schemes into a functional underwater network. The availability of such a framework will enable easy integration of independently developed techniques and thus accelerate the pace of research in underwater acoustic networking.

In order for a common framework to be successful, it needs to have a wide acceptance. To gain such an acceptance, a framework needs to take into account a wide variety of different constraints and requirements for various underwater applications. This requires inputs from various research groups and end users. To help define the use cases and a common framework for underwater networking, a joint effort has been initiated between acoustic communication research groups at the Acoustic Research Laboratory (National University of Singapore), Woods Hole Oceanographic Institution and the Massachusetts Institute of Technology.

Many traditional approaches to network design are based on the Open System Interconnection (OSI) model [6] or its variants. We define an underwater networking framework loosely based on the OSI model – the Underwater Network Architecture (UNA). The primary goal of the initiative is to define a layered architecture for underwater networking research.

In this paper, we discuss the first draft of the specifications from this effort. We welcome feedback from the underwater acoustic research community and potential end users of underwater networking systems.

II. UNDERWATER NETWORK ARCHITECTURE

A. Overview

The UNA takes into account underwater networking needs and is specific enough to allow easy integration between implementations of different layers by different research groups. At the same time, the architecture is flexible enough to accommodate different application requirements and new ideas. In addition to defining a layered architecture, the architecture definition specifies the primitives that define communication between layers. Additionally, a UNA Framework Application Programming Interface (FAPI) is defined to enable layer implementations to be easily incorporated into various stacks. To ensure flexibility, the architecture also defines an extension framework so that the architecture can be expanded and cross-layer optimization can be taken into account.

The UNA is based on a five-layer model. Each of the nodes consists of the layers shown in Fig. 1. The application layer is not defined in the UNA specifications, but is rather a client of the four layers (transport, network, data link and physical) defined in the UNA. The UNA does not define the algorithms used in each of the four layers. It only defines the service access point interface (SAPI) to be implemented by each of the layers.

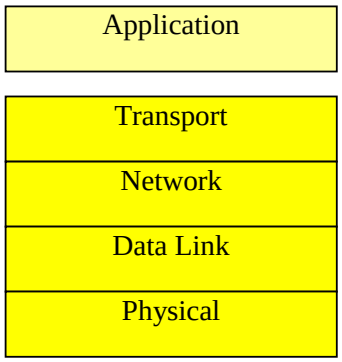

Fig. 1. Layers in the UNA

As typical underwater systems have limited processing capability, the protocol has been kept as simple as possible without significantly compromising performance. The UNA specifications currently do not include any recommendations for authentication and encryption. These may be easily implemented at the application layer or via a spreading scheme at the physical layer. The UNA will be expanded later to explicitly address these requirements.

Each layer is described by a SAPI. The SAPI is defined in terms of messages being passed to and from the layer. The clients (usually higher layers) of a layer invoke the layer via a request (REQ). The layer responds to each REQ by a response (RSP). Errors are reported via an ERR RSP with error codes. If the layer needs to send unsolicited messages to the client, it does so via a notification (NTF). A layer communicates logically with its peer layer via protocol data units (PDU). As the peer-to-peer communication is symmetric, a layer may send a REQ PDU to its peer layer at any time. It would optionally respond to such a PDU with a RSP PDU. This is logically depicted in Fig. 2.

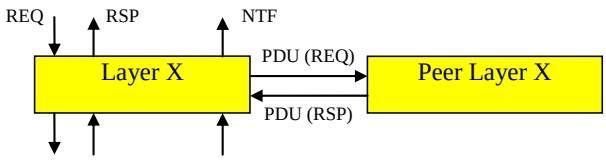

Fig. 2. Message Nomenclature in the UNA

It may be desirable in some cases, that non-neighboring layers communicate with each other to achieve cross-layer optimization. This may be implemented by allowing REQ and RSP PDUs between any two layers in the protocol stack.

The UNA specifications define detailed message structures for all SAPI messages. These message structures include message identifiers, data formats to be used, parameters and their possible values. In the interest of keeping this paper brief, we concentrate on the messages and not the detailed message structures in the following sections.

B. Addressing

Each node must be issued a unique address by the administrator. The address will consist of two parts – a network ID (8 bit) and a node ID (8 bit). The network ID defines a network that the node is a part of; all nodes in the network have the same network ID. The node ID must be unique within a network.

A special network ID (all bits 1) and node ID (all bits 1) are defined as a broadcast address. This address may be used in primitives that support broadcast. The network ID and node ID with all bits 0 are reserved and must not be assigned to any network or node.

The UNA does not define any method for dynamic assignment of addresses to new nodes. It may be expanded later to explicitly address this requirement.

C. Transport Layer

The transport layer specifications support two modes of communications – connection oriented and datagram. A connection oriented mode allows for persistent reliable connection with the open, write and close primitives and incoming data notifications. The datagram mode allows for reliable or unreliable delivery of datagrams via send primitives and incoming datagram notifications. In case of reliable datagram service, an undelivered datagram must be reported to the client layer. To enable multiple applications, the transport layer must provide the concept of ports for both modes of communication. Ports allow the transport layer to divert incoming data to multiple applications. All ports are open for connections or datagram receipts. If no application processes the data on these ports, the data is simply lost.

The transport layer defines messages for the several primitives for the connection oriented protocol – Open REQ, Close REQ, Write REQ, Connection Established NTF, Connection Lost NTF and Incoming Data NTF. It also defines messages for the primitives for the datagram protocol – Send Reliable Datagram REQ, Send Unreliable Datagram REQ and Incoming Datagram NTF. The requests, responses and notifications are summarized in Table 1 and Table 2.

D. Network Layer

The network layer provides routing capability to the protocol stack. It provides an unreliable packet delivery service over the routes. However, the layer may optionally implement some degree of reliability via retransmits. If the layer knows that a packet could not be delivered due to a lack of available route, it may inform the client layer via the no route notification.

The network layer defines messages for basic multi-hop communication primitives – Send Packet REQ and Incoming Packet NTF. It also defines messages to enable query of routing information for use in the application or transport layers. These primitives include Get Route REQ and No Route NTF. All network layer primitives are summarized in Table 3.

E. Data Link Layer

The data link layer provides single hop data transmission capability; it will not be able to transmit a packet successfully if the destination node is not directly accessible from the source node. It may include some degree of reliability. It may also provide error detection capability (e.g. CRC check). In case of a shared medium, the data link layer must include the medium access control (MAC) sub-layer. If carrier sensing, power control, etc. are desired, this functionality should be supported by the physical layer via the extension framework (defined in Section G).

Physical layer parameters may be tuned for optimal performance. If such functionality is included, it should reside in the Data Link layer as a sub-layer. The physical layer should support parameter recommendation primitives if this feature is to be implemented.

The primitives defined by the data link layer include Send Packet REQ and Incoming Packet NTF, as summarized in Table 4.

Table 1. Connection-oriented service primitives

Requests	**Responses**
Open REQ	Connection Established RSP Error RSP
Write REQ	Write Successful RSP Error RSP
Close REQ	Connection Closed RSP Error RSP
Notifications	
Connection Established NTF	
Connection Lost NTF	
Incoming Data NTF	

Table 2. Datagram service primitives

Requests	**Responses**
Send Reliable Datagram REQ	Datagram Delivered RSP Error RSP
Send Unreliable Datagram REQ	Datagram Accepted RSP Error RSP
Notifications	
Incoming Datagram NTF	

Table 3. Network layer primitives

Requests	**Responses**

Send Packet REQ	Packet Transmitted RSP Error RSP
Get Route REQ	Route Info RSP Error RSP
Notifications	
Incoming Packet NTF	
No Route NTF	

Table 4. Data link layer primitives

Requests	**Responses**
Send Packet REQ	Packet Transmitted RSP Error RSP
Notifications	
Incoming Packet NTF	
No Route NTF	

F. Physical Layer

The physical layer provides framing, modulation and error correction capability (via FEC). It provides primitives for sending and receiving packets. It may also provide additional functionality such as parameter settings, parameter recommendation, carrier sensing, etc.

The send packet primitives at the physical layer do not use destination addresses. In case of orthogonal channels, a parameter in the physical layer may be set to define the channel to be used. Then the send packet primitive may be used for the transmission. Incoming packets may be monitored on all or a few channels simultaneously. A parameter in the incoming packet notification may be included to denote the source address.

The primitives defined by the physical layer (see Table 5) are Send Packet REQ and Incoming Packet NTF. Additionally, a Recommend Parameters REQ enables higher layers to query the physical layer to help optimize the protocol parameters.

Table 5.Physical layer primitives

Requests	**Responses**
Send Packet REQ	Packet Transmitted RSP Error RSP
Recommend Parameters REQ	Recommended Parameters RSP No Recommendation RSP
Notifications	
Incoming Packet NTF	
No Route NTF	

G. Extension Framework

The extension framework allows capability query, arbitrary parameters and information to be passed around the layers for cross-layer optimization. This may include capability query such as broadcast capability and specific parameters (e.g. transmission frequency, data rate, etc.) It may also include generic data like BER, SNR, link quality, etc.

The extension framework is common across all layers. It is implemented using generic primitives – Set Parameter REQ, Get Parameter REQ and Check Capability REQ. If a Set Parameter REQ provides an invalid value for a parameter, the old value is retained and returned via the Parameter RSP. Alternatively, a value close to the requested value may be set and returned via the Parameter RSP.

Table 6.Extension framework primitives

Requests	**Responses**
Set Parameter REQ	Parameter RSP Error RSP
Get Parameter REQ	Parameter RSP Error RSP
Check Capability REQ	Capability Implemented RSP Capability Not Implemented RSP

H. UNA Framework API

The UNA FAPI provides a C API with layer registration, message queue and timer functionality. This API provides an abstraction to the hardware and OS, allowing the implementation of each layer to be portable. Table 7 provides a summary of the C prototypes for FAPI.

The FAPI includes a function for registering layer implementations – FAPI_registerLayer. Layers registered with the FAPI may communicate with each other and utilize FAPI services. Possible values for the FAPI layer include LAYER_PHYSICAL (1), LAYER_DATALINK (2), LAYER_NETWORK (3) and LAYER_TRANSPORT (4).

FAPI supports message queues, which form the basis for communication between layers. The message queues are based on an event handling model, where a message handler callback in a layer is called when a message is available for it to process. This enables the FAPI to be easily implemented on operating platforms without multi-threading support. Messages are sent via a FAPI_sendMessage function. All

messages are considered to be of equal priority and are delivered in a first-in-first-out manner.

It is common for protocols to implement timeouts. This requires timer support from the operating platform. This support is encapsulated in the FAPI timer API using the FAPI_setTimer function. When a timer expires, a Timer Expired NTF is sent to the appropriate layer. Although the time out in the API is specified in milliseconds, some platforms may not be able to support this accuracy. In such cases, the FAPI implementation should provide as close a timeout as possible on the platform. Unexpired timers may be cancelled using the FAPI_cancelTimer function.

Table 7. UNA Framework API C Prototypes

Layer Registration
int FAPI_registerLayer(int layer, MessageHandler handler);
Message Queues
void MessageHandler(Message msg, int sender);
int FAPI_sendMessage(Message msg, int destination);
Timer
int FAPI_setTimer(int me, int ref, long timeout);
int FAPI_cancelTimer(int me, int ref);

III. CONCLUSIONS

We have outlined the UNA specifications in this paper. The primary aim of the specifications is to define a common framework that the underwater networking community (researchers and industry) may use. This would allow implementations of layers from different sources to interoperate and improve the pace of advances in the field.

The current specifications are based on commonly encountered requirements in underwater systems. With inputs from researchers at the Acoustic Research Laboratory (Singapore), Woods Hole Oceanographic Institution and the Massachusetts Institute of Technology, we have an initial draft of the specifications. We hope to further improve the specifications through interaction with other groups working in the area of underwater networking, or wishing to use the technology.

REFERENCES

[1] A. B. Baggeroer, "Acoustic telemetry - An overview," IEEE Journal of Oceanic Engineering, vol. 9, pp. 229-235, 1984.

[2] M. Stojanovic, "Recent advances in high-speed underwater acoustic communications," IEEE Journal of Oceanic Engineering, vol. 21, no. 2, pp. 125-136, 1996.

[3] D. B. Kilfoyle and A. B. Baggeroer, "The state of the art in underwater acoustic telemetry," IEEE Journal of Oceanic Engineering, vol. 25, no. 1, pp. 4-27, 2000.

[4] E. M. Sozer, M. Stojanovic, and J. G. Proakis, "Underwater acoustic networks," IEEE Journal of Oceanic Engineering, vol. 25, no. 1, pp. 72-83, 2000.

[5] S. Shahabudeen and M. Chitre, "Design of networking protocols for shallow water peer-to-peer acoustic networks," in Proceedings of IEEE Oceans'05 Europe, Brest, France, 2005.

[6] "ISO 7498: Open Systems Interconnection - Basic Reference Model,"1984.