An Extensible Networking Architecture for Autonomous Underwater Vehicles

Ricardo Martins
João Borges de Sousa

LSTS – Underwater Systems and Technology Laboratory
Universidade do Porto - Faculdade de Engenharia
Rua Dr. Roberto Frias I203
4200-465 Porto, Portugal
{rasm,jtasso}@fe.up.pt

***Abstract*—In this paper we present a loosely-coupled software architecture, designed to abstract the particularities of the different network interfaces usually found in Autonomous Underwater Vehicles and communication gateways.**

Our architecture was implemented as part of a broader on-board software framework utilized in several autonomous systems. It improves on previous efforts, by having a simpler and more extensible design, reduced data overheads, and incorporating delay-tolerant networking concepts.

An overall description of the supported network interfaces and systems is also given.

I. Introduction

Autonomous Underwater Vehicles (AUVs) are commonly equipped with several network interfaces, each of which suitable for a particular operational scenario. Line-of-sight (LOS) high-bandwidth radio frequency (RF) links are used when the vehicle is surfaced and at close range. Satellite and cellular communications introduce higher latencies and impose stricter limits on the amount of data that can be transferred, but can be used when the dimensions of the area of operation preclude near range RF communication. Underwater acoustic communications are used during mission execution, with the vehicle underwater, and are notoriously prone to interferences, disruption and unpredictable delays. Furthermore, the available bandwidth is usually very limited and the latency is large [1].

Creating heterogeneous networks has proved to be challenging and usually requires special tailored software solutions [2] [3]. Several groups have developed and deployed solutions to address this problem [4] [5] [6] [7] [8], but few have been implemented as part of the on-board software of autonomous vehicles. A tight integration between communications, sensing, navigation and maneuvering is desirable in order to provide each subsystem with the maximum amount of information about its surroundings.

Tolerance to long communication delays and disruptions is of particular importance in the type of scenarios AUVs typically operate. The Delay/Disruption Tolerant Networking (DTN) architecture [9] was conceived to simplify the deployment of "challenged internets", in particular, interplanetary internets which suffer high delays. However, it can also be used in much broader operational environments, including sensor networks, wireless mobile networks and underwater networks. One of the cornerstones of the DTN architecture is the assumption that persistent storage is well distributed across the network, and data are saved to persistent storage until delivery or expiration.

In this paper we present a loosely-coupled software architecture, designed to abstract the particularities of the aforementioned network interfaces while providing a uniform and highly capable Application Programming Interface (API). The architecture incorporates DTN concepts and improves on previous efforts [10] [11], by having a simpler and more extensible design while reducing data overheads.

Our implementation was tested with a set of Light Autonomous Vehicles (LAUVs) [12], developed by the Underwater Systems and Technology Laboratory (LSTS) and depicted in Figure 1. LAUV is a torpedo shaped vehicle made of composite materials with one propeller and 4 control fins. It has an advanced miniaturized computer system and runs the Linux operating system. The vehicle is easily configurable for multiple operation profiles and sensor configurations to facilitate test and evaluation of new technologies. The LAUV is an open system which lends itself to the integration of new systems and technologies developed in Portugal and also COTS state-of-the-art equipment. In what concerns network interfaces, the LAUV is equipped with one 802.11n radio, one Iridium Short Burst Data (SBD) transceiver, one Global System for Mobile (GSM) modem, one Woods Hole Oceanographic Institution (WHOI) Micro-Modem, and one EvoLogics (Sweep-Spread Carrier) S2C acoustic modem.

On the shore-side we used a set of Manta Gateways, depicted in Figure 2, also developed by LSTS. The Manta Gateway is a centralized communications hub for maritime assets, supporting several types of wireless networks. The system is equipped with 802.11n and 802.11ac radios, one Iridium SBD transceiver, one High-Speed Downlink Packet Access (HSDPA) modem, one WHOI Micro-Modem, and one EvoLogics S2C acoustic modem. The system is fully portable and can be used stand-alone or installed on buoys, boats and ships.

The remainder of this paper is organized as follows. Section

978-0-933957-40-4

Fig. 1: LAUVs: Light Autonomous Vehicles.

Fig. 2: Manta Gateway.

IMC messages are divided in header, payload and footer. The header contains among other fields, a synchronization number which allows us to detect different byte order serializations and/or protocol versions; a message identifier, a source and a destination. The message payload varies according to the message identifier (as described by the IMC protocol specification) and can also include other (inline) messages recursively. The basic structure of an IMC message is given in Table I.

TABLE I: IMC Message Format.

Data Field	Size (byte)	Type
Synchronization Number	2	16-bit unsigned int
Message Identification	2	16-bit unsigned int
Message Size	2	16-bit unsigned int
Time Stamp	8	64-bit floating point
Source Address	2	16-bit unsigned int
Source Entity	1	8-bit unsigned int
Destination Address	2	16-bit unsigned int
Destination Entity	1	8-bit unsigned int
Message Data	Message Size	-
CRC	2	16-bit unsigned int

II gives an overview of framework used to implement our solution. Section III presents the low-level software components and describes the API provided to higher-level modules. In Section IV we describe the supported network interfaces and their capabilities. Section V characterizes the high-level modules and the implemented API. Finally, Section V concludes and highlights directions for future work.

II. Software Framework

Our architecture was implemented in the DUNE: Uniform Navigational Environment (DUNE) software framework [13] using the C++ programming language.

DUNE is responsible for navigation, control, maneuvering, plan execution, vehicle supervision, and interaction with sensors, actuators and communication devices. It is CPU architecture independent (Intel x86 or compatible, Sun SPARC, ARM, PowerPC, and MIPS) as well as operating system independent (Linux, Solaris, Apple Mac OS X, FreeBSD, NetBSD, OpenBSD, eCos, RTEM, Microsoft Windows, and QNX Neutrino).

DUNE implements a message passing mechanism where independent tasks run in a separate thread of execution. All tasks are connected to a bus that allows publishing and subscribing messages. An example of a task is, for instance, a sensor driver, that can publish a message containing information about the sensor being controlled. This information may later on be consumed by a task whose purpose is to get the vehicle navigating in three dimensional space.

The set of messages supported by DUNE is formally defined in the IMC protocol [14], a message-oriented protocol targeting networked vehicle and sensor operations. IMC is fully defined and documented in a single XML file which is translated to C++ code using Python.

The basic DUNE/IMC infrastructure is in line with typical control infrastructures for autonomous vehicles [15] [16], enabling modular development of robotic applications. Using IMC, software components can run in logical isolation, interfacing with other modules only through the exchange of IMC messages. Moreover, a common message set strictly defines the interfaces of the different types of components.

III. Low-level Transports

In order to support a varying range of communication devices and protocols, several device drivers were implemented as DUNE tasks. These tasks are called low-level transports and interact with the devices and operating system facilities to provide a basic set of IMC primitives that are later used by higher-level transports. The data unit of these class of transports is the frame and the set of messages implemented by a low-level transport is the following:

A. Capability Advertisement

To make its presence known to other tasks, each low-level transport must advertise its capabilities and basic characteristics. These advertisements can be dispatched in response to queries or sent asynchronously when the characteristics of the communication device/interface change. The advertisement message contains the following basic information about the underlying communication interface:

- name of the device manufacturer;
- device model name;
- device serial number, IMEI or unique identifier;
- monetary cost of transmitting and receiving one byte of data;
- power required to transmit and receive one byte of data;
- maximum size of a single transmitted and received frame;
- average transmission bit rate and latency.

In addition, the low-level transport must specify which of the following capabilities are supported by the underlying communication interface:

- the interface has its own address scheme;
- reliable transmission of frames is supported;
- broadcasting is supported;
- node discovery is supported;
- measuring the distance to other systems is supported.

B. Notification of frame reception

When a frame is received on an interface, the controlling low-level transport decodes the data, and, if interface-level addressing is available, translates the source and destination interface-level addresses to canonical system names using a lookup table. A message containing the following fields is then dispatched to the message bus:

- canonical name of the originating and destination systems;
- sequence of data bytes.

C. Frame transmission requests

When a frame transmission request is received by a low-level transport, and interface-level addressing is supported, the transport translates the canonical name of the destination system to an interface-level address. The request is then put into a priority queue with ascending order of lifespan. The frame transmission request message has the following fields:

- 16-bit number that uniquely identifies the request;
- name of the destination system;
- request lifespan in seconds;
- sequence of data bytes.

D. Frame transmission request status

This message is dispatched when the transmission status of a request changes, the originating system is notified asynchronously of such events, and the message contains the following information:

- 16-bit number that uniquely identifies the request;
- status code indicating the status of the request: queued, expired, canceled, transmitting, transmitted, waiting, failed;
- additional human-readable information in case of failure.

E. Request ranging

If the interface has ranging capabilities, a feature mainly found in acoustic modems, the transport supports requests with the following information:

- 16-bit number that uniquely identifies the ranging request;
- list of canonical system names to range;
- request lifespan in seconds.

F. Notification of range reception

A notification sent in response to a successful range request, containing the following fields:

- 16-bit number that uniquely identifies the ranging request;
- list of canonical system names;
- list of distances.

G. Cancelation Request

This message will request cancelation of any type of transmission requests (frames or ranging) and contains the following fields:

- 16-bit number that uniquely identifies the request;
- human-readable message with cancelation reason.

H. Link Availability

This message is used by low-level transports that support link discovery, and is dispatched every time the transport discovers a new node or loses contact with a previously found node. This message contains the following fields:

- canonical name of the system
- link status: unknown, available, unavailable

IV. Supported Interfaces

In this section we provide an overview of the communication devices currently supported by our implementation.

A. WHOI Micro-Modem

The WHOI Micro-Modem is a compact, low-power acoustic modem developed for research purposes by WHOI. A detailed description of the modem can be found in [17] [18].

The modem has the capability to perform low-rate frequency-hopping frequency-shift keying (FH-FSK), with a bit rate of 80 bits per second (bps). High rate transmission is accomplished using variable rate phase-coherent keying (PSK), with a variable bit rate between 2,500 and 5,000 bps.

Reception of PSK data units requires an additional Floating Point Coprocessor board, which due to export restrictions is not available to our group and therefore has not been tested.

The FH-FSK WHOI Micro-Modem uses two classes of data units: a short packet which may contain 13 bits of user information, called a mini-packet, and a longer packet with 32 bytes of data. The mini-packet takes approximately 0.7 seconds to transmit and is used to acknowledge reception of data and to preface the transmission of 32 byte packets. Transmission of regular packets takes 4 seconds in addition to the time required to transmit the prefixing mini-packet.

The modem has built-in support for networks with up to 16 nodes and offers an acknowledgment capability which enables guaranteed delivery transactions. In our implementation the address 0 is used for node discovery and broadcasts.

Node discovery is implemented using an algorithm that dictates that every node that does not have any data to transmit within a certain amount of time must actively advertise itself, thus allowing other nodes to learn about its presence and reachability. This active advertisement takes the form of a mini-packet destined to the broadcast address.

Additionally, the WHOI Micro-Modem supports pinging (i.e. ranging) other modems and narrow band transponders. This capability is used by the navigation filter and ranging requests are interleaved with data transmissions.

B. EvoLogics S2C Acoustic Modem

The EvoLogics S2C series of underwater acoustic modems provides full-duplex digital communication using S2C technology. Several frequency ranges are available, balancing hardware size, bit rate and range. The LAUV is equiped with a 18-34 kHz model that can achieve data rates of up to 33 kbit/s and transmit acoustically up to 2 km [19] [20].

Interfacing with the modem is done through AT commands via a Ethernet or RS-232 serial port. The EvoLogics S2C modems offer two types of delivery algorithms: burst data and instant messages.

Burst data employs an adaptive delivery algorithm that adjusts S2C parameters in order to achieve and maintain the highest bit rate possible. Two communicating devices perform a handshaking procedure before transmitting actual data. The acoustic channel parameters are measured during the handshaking phase and data transfer in order to select the optimal data packet size and data are transparently buffered, split and reassembled.

Instant messages can be transmitted at all times, even during burst data transfers. Short instant messages have a maximum size of 64 bytes and are transferred with a constant bit rate of 976 bps. This type of delivery algorithm can also be used to broadcast data.

The modem has built-in support for networks with a configurable maximum number of nodes and offers an acknowledgment capability which enables guaranteed delivery transactions. In our implementation the address 14 is used for node discovery and broadcasts. Node discovery is implemented in a similar way to the WHOI Micro-Modem.

Ranging other modems is done automatically when using burst data but can also be explicitly requested.

C. User Datagram Protocol

The User Datagram Protocol (UDP) is a transaction oriented protocol where delivery and duplicate protection are not guaranteed. UDP uses the Internet Protocol (IP) as its underlying protocol [21].

The UDP transport implements a simple client/server infrastructure capable of transmitting and receiving datagrams and managing a queue of transmission requests.

Node discovery is implemented using periodic broadcast and multicast announcements.

D. Transmission Control Protocol

The Transmission Control Protocol (TCP) [22] provides reliable host-to-host communication between hosts over IP.

The TCP transport implements a simple client/server infrastructure capable of managing connections, transmitting and receiving segments and managing a queue of transmission requests.

Node discovery is implemented using the same mechanism used by the UDP transport.

E. Short Message Service (SMS)

The Short Message Service provides a means of sending messages of limited size to and from GSM mobiles. The provision of SMS makes use of a service center, which acts as a store and forward centre for short messages [23]. The maximum length of an uncompressed 8-bit SMS message is 134 bytes and this service is usually only available in coastal waters.

The SMS transport interacts with the GSM modem using AT commands [24] via an RS-232 serial port, and listens or polls, depending on the modem used, for incoming messages, and maintains a queue of transmission requests.

F. Iridium SBD Transceivers

Iridium SBD transceivers target application development in the areas of field force automation and remote asset tracking [25]. Presently, the Iridium satellite constellation consists of 66 Low Earth Orbiting (LEO) satellites at a height of approximately 780 km. The satellites complete a pole to pole orbit in around 100 minutes and the constellation covers the entire Earth's surface at every moment.

The elements that compose the Iridium SBD architecture are the Field Application, the Iridium Subscriber Unit (ISU), the Iridium satellite constellation, the Iridium Gateway, the Internet, and the Vendor Application. Machine to Machine (M2M) communication requires middleware to route messages between ISUs.

The maximum size of an SBD message is transceiver specific, ranging from 340 to 1960 bytes for mobile originated messages and 270 to 1890 bytes for mobile terminated messages.

The Iridium SBD transport maintains a queue of transmission requests and waits for notifications of pending messages at the Iridium Gateway. If there are no packets to transmit and no SBD pending notifications where received, the transport performs a mailbox check by sending an empty SBD message.

V. High-level Transport

Our architecture contains one high-level transport that abstracts details about the low-level transports and provides an simple interface to reliably transmit and receive large data-quantities. The data unit of the high-level transport is the packet.

This high-level transport listens for low-level transport advertisements and maintains a table of available links and reachable systems. This task is also responsible for:

- storing packets to persistent media until the packet is delivered or expires;
- finding the most suitable low-level transport to transmit data to a remote node;
- compressing and fragmenting packets into frames manageable by low-level transports and performing the converse operations;
- ensuring the delivery of frames and packets.

The API provided by this task is as follows:

A. Notification of packet reception

When all frames of one packet are received and the packet is reassembled successfully, a message containing the following fields is dispatched to the message bus:

- canonical name of the originating and destination systems;
- sequence of data bytes.

B. Packet transmission request

When a packet transmission request is received, the packet is saved to persistent storage. If a link to the destination system is available the packet is fragmented into frames with an appropriate size for the low-level transport and the task starts requesting frame transmissions. The packet transmission request message has the following fields:

- 16-bit number that uniquely identifies the request;
- name of the destination system;
- request lifespan in seconds;
- sequence of data bytes.

C. Packet transmission request status

This message is dispatched when the status of a packet transmission request changes, the originating system is notified asynchronously of such events.

- 16-bit number that uniquely identifies the request;
- Status code indicating the status of the request: queued, expired, transmitted.

VI. Conclusion

We described a software architecture capable of transparently and reliably interconnect several types of network interfaces in order to create an heterogeneous network.

The software framework we developed is suited for use in gateways and autonomous vehicles, and for mission critical transactions where data must be delivered reliably and traverse highly heterogeneous links in order to reach its destination.

In the short term, we intend to add support for multi-hop routing and implement Media Access Control (MAC) services on interfaces that lack such capabilities.

References

[1] I. F. Akyildiz, D. Pompili, and T. Melodia, "Challenges for efficient communication in underwater acoustic sensor networks," *ACM SIGBED Rev.*, vol. 1, no. 2, pp. 3–8, 2004.

[2] E. R. B. Marques, J. Pinto, S. Kragelund, P. S. Dias, L. Madureira, A. Sousa, M. Correia, H. Ferreira, R. Gonçalves, R. Martins, D. P. Horner, A. J. Healey, G. M. Gonçalves, and J. B. Sousa, "AUV control and communication using underwater acoustic networks," in *In Proc. IEEE Oceans Europe*, 2007.

[3] E. R. B. Marques, G. M. Gonçalves, and J. B. Sousa, "Seaware: A Publish/Subscribe Communications Middleware for Networked Vehicle Systems," in *In Proc. IFAC Conference on Manoeuvring and Control of Marine Craft (MCMC)*, 2006.

[4] J. A. Rice, "Enabling undersea forcenet with seaweb acoustic networks," in *SSC San Diego Biennial Review 2003*, 2003.

[5] C. Petrioli and R. Petroccia, "SUNSET: Simulation, emulation and real-life testing of underwater wireless sensor networks," in *Proceedings of IEEE UComms 2012*, Sestri Levante, Italy, September, 12–14 2012.

[6] T. Schneider and H. Schmidt, "Goby-acomms version 2: extensible marshalling, queuing, and link layer interfacing for acoustic telemetry," in *9th IFAC Conference on Manoeuvring and Control of Marine Craft, Arenzano, Italy*, 2012.

[7] T. E. Schneider, "Advances in integrating autonomy with acoustic communications for intelligent networks of marine robots," Ph.D. dissertation, Massachusetts Institute of Technology, 2013.

[8] S. Shahabudeen, M. Chitre, M. Motani, and A. Siah, "Unified simulation and implementation software framework for underwater mac protocol development," in *OCEANS 2009, MTS/IEEE Biloxi-Marine Technology for Our Future: Global and Local Challenges*. IEEE, 2009, pp. 1–9.

[9] K. Fall, "A delay-tolerant network architecture for challenged internets," in *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 2003, pp. 27–34.

[10] M. Demmer, E. Brewer, K. Fall, S. Jain, M. Ho, and R. Patra, "Implementing delay tolerant networking," in *Intel Research Technical Report IRB-TR-04-020*, 2004.

[11] D. Merani, J. Potter, A. Berni, and R. Martins, "An underwater convergence layer for disruption tolerant networking," in *Baltic Congress on Future Internet Communications (BCFIC 2011)*. IEEE, 2011, pp. 103–108.

[12] L. Madureira, A. Sousa, J. Sousa, and G. Gonçalves, "Low cost autonomous underwater vehicles for new concepts of coastal field studies," *Journal of Coastal Research*, pp. 238–242, 2009.

[13] R. Martins, "DUNE: Uniform navigational environment," 2013. [Online]. Available: http://lsts.pt/software/dune

[14] R. Martins, P. S. Dias, E. R. B. Marques, J. Pinto, J. B. Sousa, and F. L. Pereira, "IMC: A communication protocol for networked vehicles and sensors," in *OCEANS 2009-EUROPE, 2009. OCEANS '09.*, 11-14 2009, pp. 1 –6.

[15] B. Gerkey, R. Vaughan, and A. Howard, "The Player / Stage Project : Tools for Multi-Robot and Distributed Sensor Systems," in *Proceedings of the 11th international conference on advanced robotics*, 2003, pp. 317–323.

[16] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[17] L. Freitag, M. Grund, I. Singh, J. Partan, P. Koski, and K. Ball, "The whoi micro-modem: an acoustic communications and navigation system for multiple platforms," in *In Proc. IEEE OCEANS'05 Conf*, 2005.

[18] W. H. O. I. Acoustic Communications Group, "Micro-modem software interface guide, version 3.01," 2009.

[19] V. K. Kebkal, R. Bannasch, and K. Kebkal, "Performance of hydroacoustic data transmission in horizontal channels using sweep-spread signaling and turbo-coding," in *OCEANS, 2011 IEEE - Spain*, 2011, pp. 1–6.

[20] EvoLogics, "EvoLogics S2C Acoustic Modems," 2013. [Online]. Available: http://www.evologics.de

[21] J. Postel, "User datagram protocol," Internet Engineering Task Force, RFC 768, August 1980. [Online]. Available: http://www.rfc-editor.org/rfc/rfc768.txt

[22] ——, "Transmission control protocol," Internet Engineering Task Force, RFC 793, September 1981. [Online]. Available: http://www.ietf.org/rfc/rfc793.txt

[23] European Telecommunications Standards Institute, "3rd generation partnership project; technical specification group terminals; technical realization of the short message service (sms)," (3GPP TS 03.40, Version 7.5.0 Release 1998), December 2001.

[24] G. ETSI, "07.07: Digital cellular telecommunication system (phase 2+) at command set for gsm mobile equipment (me)," 1997.

[25] Iridium Satellite LLC, *Iridium Short Burst Data Service Developers Guide Release 3.0*, 2012.