TOPICS: A Modular Software Architecture for High-Latency Communication Channels

Chris Murphy
Bluefin Robotics Corporation
553 South Street
Quincy, Massachusetts 02169 USA
cmurphy@bluefinrobotics.com

***Abstract*—This paper presents TOPICS, a production software architecture for communicating between unmanned maritime vehicles over high-latency channels, including acoustic and Iridium Short Burst Data (SBD) modems. The architecture is modular and extensible to facilitate development of mission-specific capabilities in addition to standard vehicle behaviors. TOPICS supports multiple redundant communication devices per vehicle, low-level modem abstraction for "backseat driver" applications, and file transfer when supported by the modem.**

I. Introduction

Autonomous Underwater Vehicles (AUVs) have long been reliant on acoustic modems for communicating with surface operators while submerged. The acoustic channel presents numerous and well-documented challenges to communication[1], [2], which results in low communication bandwidth, high message latency, and a rapidly changing channel. Increasingly, AUVs and other unmanned maritime vehicles also rely on Iridium Short Burst Data (SBD) for remote monitoring and command when on the surface but "over the horizon." While the Iridium channel is significantly more reliable than the acoustic one, both present long communication latencies and expose software interfaces that rely on small fixed-length packets. Acoustic modems, such as the widely used Woods Hole Oceanographic Institution (WHOI) Micromodem[3], typically support packets of tens or hundreds of bytes in size, and a typical Iridium SBD modem supports packets up to two thousand bytes in size.

TOPICS is a production software architecture for communicating between unmanned maritime vehicles over these high-latency and packetized channels. TOPICS, which stands for "Tele-Operation via Packetized and Intermittent Communication Subsystems," supports redundant communication devices on each vehicle, provides a "backseat driver" interface for low-level modem control, uses the open-source Dynamic Compact Control Language (DCCL)[4] for data encoding, and supports modem-driven file transfer when available. TOPICS has been developed, tested, and deployed on Bluefin AUVs, and integrates with Bluefin's SOMA[5] publish-subscribe software architecture.

There are several recent examples of software architectures for acoustic communication, including [6], [7], [8] and [9]. Common to these architectures are several philosophies, including software isolation between data sources and modem interfaces, reliance on some form of compressed message encoding, and (typically) a time-based Medium Access Control (MAC) strategy. In addition to sharing those philosophies, TOPICS key architectural points include explicit support for:

- The Dynamic Compact Control Language (DCCL)[4]
- Multiple modems per vehicle
- Modem control by "backseat driver" applications
- Modem-driven file transfer

In the following sections the TOPICS architecture is described at a high level(Sec. II), followed by a discussion of the key architectural points (Sec. III) relative to previous work. Examples of common usage patterns (Sec. IV) are then presented, before concluding with a more detailed view of the interfaces (Sec. V) presented by each component of the architecture and a discussion of future work (Sec. VI).

Fig. 1. Bluefin 21" AUV with dual Sonardyne AvTrak6 HP modems for redundant acoustic communication.

II. Architecture

TOPICS must fill the needs of diverse vehicles and missions, across the spectrum of military, commercial survey, and academic research applications. To maintain future flexiblity across all these configurations, including those not yet identified, the TOPICS architecture was designed to be highly modular and extensible – with clearly defined interfaces between only a few types of components. The relationship between these key components is shown in Figure 2, including the software interfaces between them. There are four software components that make up TOPICS: Medium Access Control

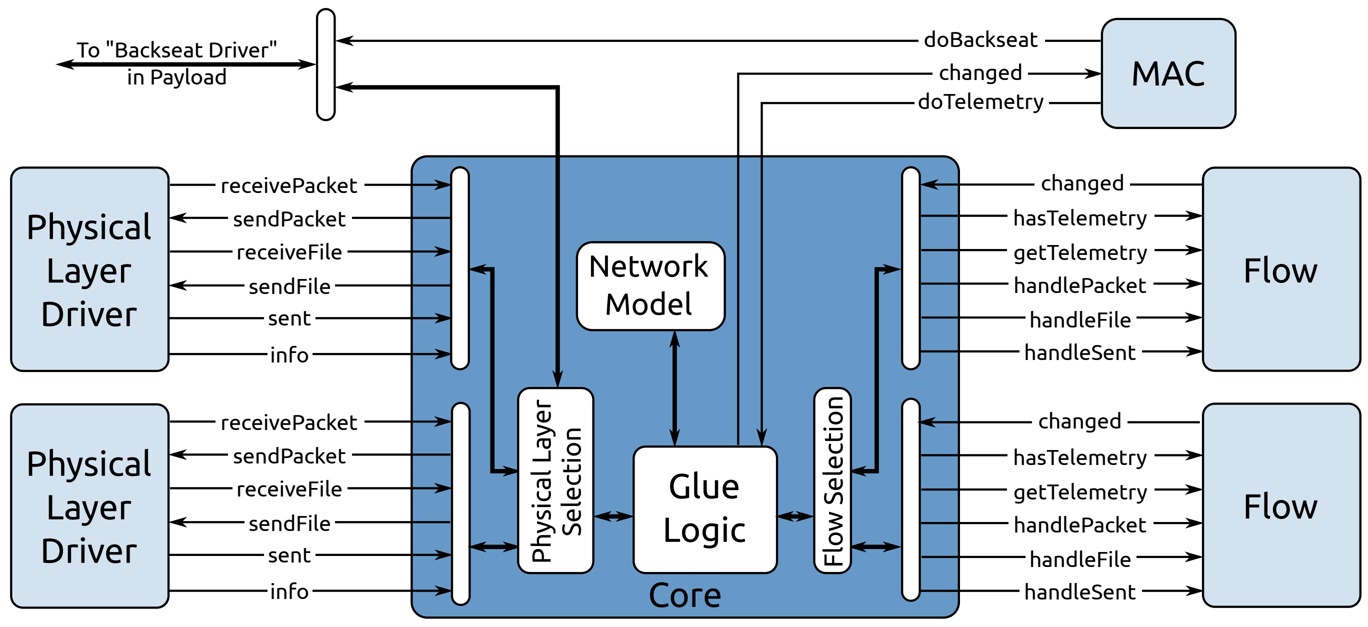

Fig. 2. Key software components of the TOPICS architecture, including the MAC, Data Flows, Physical Layers and TOPICS Core, and an overview of key interfaces between them.

(MAC), Data Flows, Physical Layers, and the TOPICS Core which connects the other three types of component. Each type of component is discussed in greater detail below, in Sections II-A through II-D.

A. Medium Access Control (MAC)

MAC modules are responsible for determining when telemetry can be transmitted and notifying the TOPICS Core that it can transmit data, along with a deadline by which that transmission must be completed. It also can notify the TOPICS Core that control of a Physical Layer should be loaned to a "backseat driver" for temporary use, as described in greater detail below (Sec III-C). In the current implementation, the MAC is driven by a schedule of time blocks assigned to specific vehicles, though there is the explicit goal of supporting more exotic MAC approaches such as Multiple Access with Collision Avoidance (MACA)[10], MACA for Wireless (MACAW)[11] or the Distance Aware Collision Avoidance Protocol (DACAP)[12].

B. Data Flows

TOPICS uses modular Data Flows to organize types of traffic. A Flow describes a single stream of related data which can prioritize its own traffic. Examples of a Data Flow include a file transfer manager, a queue of operator commands, traffic from a fault management subsystem, or a set of vehicle status messages. Upon creation, Data Flows must register the type(s) of message that each will transmit, and each is responsible for handling any messages that come in with that same type. Message types are specified using the Dynamic Compact Control Language (DCCL)[4], in concert with TOPICS-specific Protocol Buffer extensions.

C. The Physical Layer

Physical Layer modules implement a common interface to each acoustic or Iridium SBD modem. The modules store common device parameters, such as the hardware address of the device, and present a common software interface for communication. While most acoustic modems expect some form of serial connection to the vehicle, the underlying software interfaces vary widely from modem to modem. By having device-specific drivers implement a common Physical Layer interface, many details of modem protocols are obscured from the rest of the software stack. The interface does maintain the packet-oriented nature of these channels, rather than attempting to craft a stream-oriented protocol, to maintain reliability in high packet loss environments. Some modems support built-in file transfer, typically relying on a built-in Automatic Repeat-reQuest (ARQ) scheme, and TOPICS exposes that functionality when available.

D. TOPICS Core

The primary role of the TOPICS Core is to mediate between the Data Flows (which want to send and receive data) and Physical Layers (which actually can send and receive data) when data is received, and when instructed to send data by the Medium Access Control. Key to this mediation is a mapping between hardware addresses and vehicles, since there will be multiple vehicles and each may have multiple Physical Layers that it can rely on. This mapping is contained in a network model internal to TOPICS. The network model tracks which vehicles exist, what the addresses are for any Physical Layers on board each of them, and which Physical Layers should be used preferentially. In a small network, this model might consist only of a single AUV and a surface ship running

```
message Status {
  option (dccl.msg).id = 127;
  option (dccl.msg).max_bytes = 5;

  optional double depth = 10 [
    (dccl.field).min = 0, (dccl.field).max = 11000,
    (dccl.field).precision = 1
  ];
}
```

Fig. 3. A minimal DCCL message for encoding the full ocean depth (0m to 11000m) to a precision of one decimal place. For details of the DCCL syntax, see [4] and [13].

```
message Status {
  option (dccl.msg).id = 127;
  option (dccl.msg).max_bytes = 5;

  optional double depth = 10 [
    (dccl.field).min = 0, (dccl.field).max = 11000,
    (dccl.field).precision = 1,
    (bluefin.field).somasub = "*/tracking.depth/*",
    (bluefin.field).units = "m"
  ];
}
```

Fig. 4. The message from Figure 3, annotated with additional TOPICS-specific metadata. The datasource that should be used to populate the message field is indicated by "somasub" - which must be a valid SOMA[5] subscription. The second annotation, "units," indicates the units of the encoded data. Many SOMA publications have units associated with the value.

topside monitoring equipment, though the model scales to larger networks as well.

In addition to the network model, the TOPICS Core contains logic for selecting a specific Flow when transmission is allowed. Currently, that logic is simply a priority queue of the Data Flows; more sophisticated schemes such as Weighted Fair Queueing (WFQ) are used within individual Data Flows to provide more sophisticated traffic management.

Finally, the TOPICS Core manages the life cycle of each subcomponent, ensuring that each is configured correctly, started, and stopped at appropriate times during the application life cycle. A main event loop is responsible for triggering any time-based events scheduled by subcomponents (such as MAC deadlines), and distributing SOMA messages to any subcomponents that have requested specific subscriptions.

III. Key Architectural Points

A. Dynamic Compact Control Language (DCCL)

The Dynamic Compact Control Language[4] provides algorithms and a software library for marshalling and compressing data on board AUVs. Figure 3 shows a trivial DCCL message containing a single field, the depth of a vehicle in meters. In addition to defining a structured message consisting of only the single field 'depth', the definition specifies the minimum and maximum value for the field and the number of decimal places desired. This information will be used to quantize the data during transmission for compression. More complex options for field encoding are built into DCCL, and others can be created to compress specific types of data. DCCL message types can be compiled into source code – useful where reliability is the overriding concern – or loaded at run-time if greater flexibility is needed.

DCCL, initially developed by Schneider et al. as part of the Goby Autonomy Framework[9], was selected for TOPICS due to its extensibility, and the excellent balance it strikes between flexibility and efficient data encoding. Figure 4 shows the same trivial DCCL message as in Figure 3, now annotated with TOPICS-specific metadata used by one Data Flow. The first field annotation, "somasub," indicates the name of a SOMA[5] publication that should be used as the onboard source for the field's data. SOMA publications essentially consist of key-value pairs, with a three-part "Name" representing the key (like shown) and the value being one of a few supported datatypes. The second field, "units," indicates the units that should be used when encoding and decoding the data. Many values published using SOMA are done so in the form of Quantities – which have both a numerical value and a unit. Value conversion between compatible units (e.g. feet to meters) is handled automatically, but attempted conversion between incompatible units (e.g. trying to convert feet to grams) will be flagged as an error. The TOPICS annotations allow us to extend that safety and type-checking to acoustic transmissions as well.

B. Multi-Device, Multi-Vehicle Support

Within TOPICS, Physical Layers are logically aggregated by vehicle - a single vehicle may have multiple devices it can send from (as in Figure 1), and there may be multiple modems on each target vehicle that it communicates with. TOPICS tracks the set of known vehicles it can communicate with, along with the "preferred" Physical Layer to send to when communicating with that vehicle. Incoming data from all devices is treated identically, allowing (e.g.) commands to be received by any operational interface. All of this information is stored internally in a network model. Physical Layer preferences, for either the local vehicle or a remote vehicle, can be updated by a command from the surface.

C. Backseat Driver Interface

AUVs typically have autonomy designed into the main vehicle computer. For some applications, vehicles now rely on a "backseat driver" division of labor to facilitate development of advanced autonomy algorithms while ensuring vehicle safety. The main vehicle computer, still responsible for vehicle control and interfacing with vehicle sensors and actuators, is augmented by a secondary "backseat driver" computer running high-level autonomy algorithms. Inspired by the layered autonomy approach described in [14], this insulates autonomy-layer tasks from the low-level manufacturer-specific details of each sensor, and allows advanced algorithms to be disabled in the case of a vehicle emergency. Many maritime autonomy frameworks, including MOOS-IvP[15], rely on this division of labor.

For autonomy applications with no specific communication requirements, TOPICS can operate as normal and communicate vehicle status and sensor data. If autonomy applica-

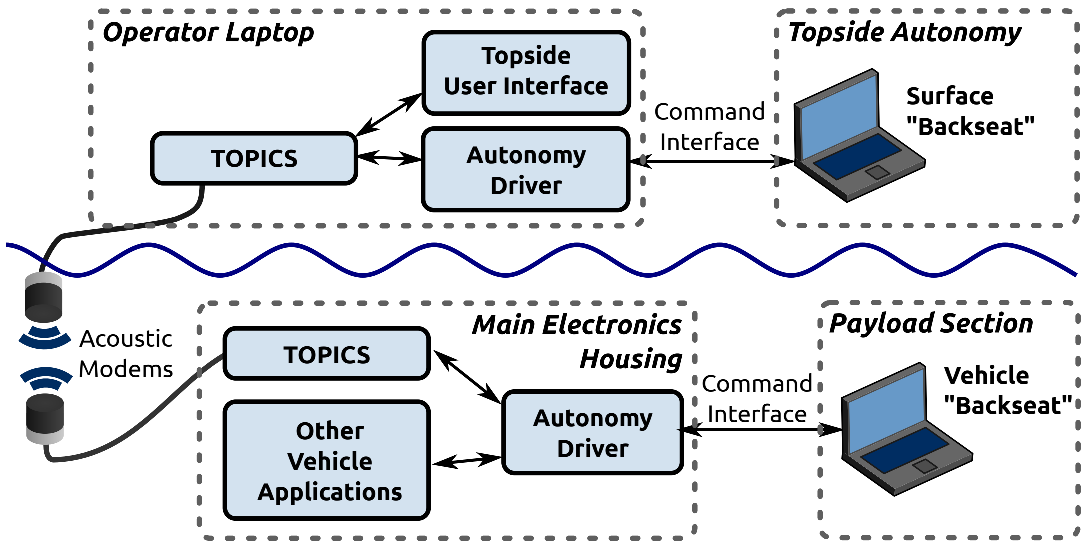

Fig. 5. Control of TOPICS from the payload section on a single AUV, and by a laptop onboard a surface ship, via a command interface.

tions need to communicate, perhaps to coordinate behaviors between vehicles, TOPICS can expose a low-level command interface on both the vehicle, and any surface ship, as shown in Figure 5. That device-level software interface provides cues (from the MAC) for when the backseat driver can communicate. During times where communication is allowed, the same Physical Layer interface that the main vehicle computer relies on is exposed via the command interface. This allows autonomy algorithms to "borrow" the Physical Layer and communicate their own messages, while isolated from low-level details of device protocols, and without affecting basic vehicle status monitoring, command, and control. This isolation from the physical hardware also allows emergency behaviors to rescind control from any backseat driver control.

D. File Transfer

When used with Physical Layers that support file transfer, Data Flows can request that a file be transmitted rather than a packet when *getTelemetry* is called. Rather than transferring a packet containing DCCL encoded data, a single DCCL message is included in the transmission as header metadata prior to the contents of the file. This allows files to be routed to appropriate Data Flows when they are received, based on the DCCL type of the metadata, and also for arbitrary metadata to be included with the file transmission. The destination is still mapped to a modem hardware address using the network model before transmission.

IV. Usage Examples

The following examples illustrate how components interact during the course of typical usage - packet transmission and receiption.

A. Transmission

When the TOPICS Core is notified by the MAC that a transmission can occur, the TOPICS Core does the following:

1) Selects a Physical Layer: First a Physical Layer is selected for transmission. This is done by consulting the network model to determine the 'preferred' Physical Layer for this vehicle. If that Physical Layer has failed or is otherwise unavailable, TOPICS will fall back to other available Physical Layers.

2) Selects a Data Flow: TOPICS next iterates through all available Data Flows, checking whether each has any telemetry to send.

3) Obtains a Send Request from Data Flow: Having determined that the selected Data Flow has telemetry to send, that telemetry is requested from the Data Flow along with a destination, unique request ID, telemetry mode, and whether or not to request an acknowledgement.

4) Converts Destination into Address: While Data Flows request transmission to specific vehicles, Physical Layers require hardware addresses to be used as destinations. TOPICS uses the network model to perform this mapping.

5) Passes Request to Physical Layer for Transmission: The actual telemetry (whether packet or file), destination hardware address, and request ID are all then passed to the Physical Layer for transmission.

B. Telemetry Reception

When telemetry is received by a Physical Layer, the TOPICS Core executes the following steps:

1) Converts the Source Address into a Vehicle: First, the hardware address of the originating modem is mapped (via the network model) into a vehicle identifier.

2) Looks Up the Appropriate Data Flow: The DCCL library is used to identify the type of message, which is then used to identify the appropriate flow for handling the telemetry.

3) Passes Telemetry to a Data Flow: The telemetry (whether a packet or file) is then passed to the appropriate Data Flow for handling.

V. Interfaces

Each component type has a small number of well-defined interfaces, as shown in Figure 2. Each interface is described below, grouped by component type.

A. Medium Access Control (MAC)

There are currently only three interfaces to MAC modules, as described below.

1) doTelemetry: When a telemetry transmission is allowed, the MAC instructs the TOPICS Core to initiate a telemetry cycle. This interface includes a deadline by which the telemetry cycle must be completed.

2) doBackseat: When the "backseat driver" is allowed to control the modem, ths interface instructs the TOPICS Core to notify the "backseat driver." A deadline for when control will be rescinded is included.

3) changed: The MAC is notified whenever there has been a change in flow status, such as telemetry now being available for transmission when it was not previously.

B. Data Flows

Not every Data Flow must implement every one of the below interfaces. A Data Flow containing a continuously updating status message, for instance, may only implement *getTelemetry* and *handlePacket*, and have *hasTelemetry* always report that telemetry is available.

1) hasTelemetry: This interface is called by the TOPICS Core to determine whether a given Data Flow would request data transmission, if *getTelemetry* were called. The Flow it is called on is provided the Physical Layer, along with the transmission deadline, so that the Flow is aware of the constraints on any transmission.

2) getTelemetry: This interface is called by the TOPICS Core to obtain data for transmission. It will only be called if *hasTelemetry* returned True, and should return valid data. The Flow it is called on is provided the Physical Layer, along with the transmission deadline, so that the Flow is aware of the constraints on any transmission. A request to send either a packet or a file is returned, along with any necessary metadata such as the destination.

3) changed: This interface is emitted every time the value of hasTelemetry might have changed, which may prompt the MAC or TOPICS Core to start telemetry transmission.

4) handlePacket: This interface is called by the TOPICS Core when a data packet that matches a registered DCCL type for this Data Flow is received. The received packet, the time of arrival, the vehicle which originated the packet, and the vehicle which was the destination (which may not be this vehicle) are all provided.

5) handleFile: This interface is called by the TOPICS Core when a file and associated metadata are received, and the metadata matches a registered DCCL type for this Data Flow. The received metadata, the filename, the time of arrival, the vehicle which originated the transmission, and the vehicle which was the destination (which again may not be this vehicle) are all provided.

6) handleSent: After a Physical Layer has completed transmission of a request from a Data Flow, it reports back on the success or failure of the task and any requested acknowledgement via this interface. The aforementioned statuses and the request identifier are provided.

C. Physical Layers

Each Physical Layer module must implement four interfaces for packetized communication - *sendPacket*, *receivePacket*, *sent*, and *info*. When a modem supports built-in file transfer, the additional interfaces *sendFile* and *receiveFile* can be implemented by Physical Layer modules to expose that functionality.

1) sendPacket: This is the primary interface implemented by Physical Layer modules – it is used to communicate packets from TOPICS to each individual Physical Layer module for transmission. Besides the underlying packet data for transmission, a few specific parameters describing the transmission request must be provided. These include the level of Forward Error Correction (FEC) applied to a packet (which is controllable on many acoustic modems), whether to request a low-level receive acknowledgement for a transmitted packet, the hardware address for the packet destination and a unique request identifier. The unique request identifier is used by the *sent* interface.

2) receivePacket: When a packet is received by a Physical Layer module, the data contained in the received packet, the time of arrival, the hardware address of the sender, and the hardware address of the destination are emitted. While some modems (such as the Sonardyne AvTrak6, or Iridium SBD modems) only report a packet if it was destined for the specific modem, other modems (such as the WHOI Micromodem) report all decodable packets - even those destined for another device. TOPICS forwards the packet to the appropriate Data Flow's *handlePacket* interface.

3) sent: When a packet or a file is transmitted, a unique request ID is provided to the Physical Layer by the Data Flow; when the Physical Layer has completed the sending task, it reports using the *sent* interface that the request has been completed by providing that same unique request ID. If an acoustic acknowledgement was requested for the packet (supported at the hardware level by several commercially available acoustic modems) the status of that acknowledgement is also reported. TOPICS forwards this information to the appropriate Data Flow's *handleSent* interface. This allows the originating Data Flow to track whether or not individual messages were received to, e.g., support implementing Automatic Retry reQuest (ARQ).

4) info: This interface communicates the set of capabilities that a physical layer supports, such as the Maximum Transmission Unit (MTU) for different telemetry modes, and whether or not file transfer is supported. It can be accessed by Data Flows to determine the bounds that telemetry must fit within.

5) sendFile: If a physical layer supports file transmission, this interface is used to request that a file and associated metadata be transmitted to a specific destination.

6) receiveFile: If file transmission is supported and a file is received, the file itself will be saved to a temporary location on disk. File metadata is forwarded by TOPICS to the appropriate Data Flow's *handleFile* interface along with the filename.

VI. Ongoing Work

TOPICS is now in active use on vehicles produced by Bluefin Robotics, so the development focus is now on implementing additional, and more advanced, types of Flow and Physical Layers. Development on the underlying architecture continues as well, however, with emphasis on:

- Advanced vehicle payload monitoring and control
- Balancing communication needs and mission objectives
- Implementing more advanced MAC schemes

References

[1] M. Chitre, S. Shahabudeen, and M. Stojanovic, "Underwater acoustic communications and networking: Recent advances and future challenges," *Marine Technology Society Journal*, vol. 42, no. 1, pp. 103–116, 2008.

[2] I. F. Akyildiz, D. Pompili, and T. Melodia, *Underwater Acoustic Sensor Networks: Research Challenges*. Elsevier, Mar. 2005, vol. 3, pp. 257–279.

[3] L. Freitag, M. Grund, S. Singh, J. Partan, P. Koski, and K. Ball, "The WHOI Micro-Modem: An Acoustic Communcations and Navigation System for Multiple Platforms," in *Oceans 2005, Proceedings of the MTS/IEEE*, vol. 2, 2005, pp. 1086 – 1092.

[4] T. Schneider and H. Schmidt, "The Dynamic Compact Control Language: A compact marshalling scheme for acoustic communications," in *Proceedings of MTS/IEEE OCEANS 2010*, Sydney, Australia, 2010, pp. 1–10.

[5] D. Goldberg, "Huxley: A flexible robot control architecture for autonomous underwater vehicles," in *OCEANS, 2011 IEEE - Spain*, 2011, pp. 1–10.

[6] S. E. Webster, R. M. Eustice, C. Murphy, H. Singh, and L. L. Whitcomb, "Toward a platform-independent acoustic communications and navigation system for underwater vehicles," in *Proc. IEEE/MTS OCEANS Conf. Exhib.*, Biloxi, MS, Oct. 2009, pp. 1–7.

[7] M. Jakuba, J. C. Kinsey, D. R. Yoerger, L. L. Whitcomb, R. Camilli, C. Murphy, A. Bowen, and C. R. German, "Communications and Control for Enhanced Autonomy in Underwater Vehicles for Deep Oceanographic Research," *AGU Fall Meeting Abstracts*, Dec. 2010.

[8] C. Murphy, J. Walls, T. Schneider, R. Eustice, M. Stojanovic, and H. Singh, "CAPTURE: A communications architecture for progressive transmission via underwater relays with eavesdropping," *Oceanic Engineering, IEEE Journal of*, vol. In Publication, pp. 1–11, 2013.

[9] T. Schneider and H. Schmidt, "Unified command and control for heterogeneous marine sensing networks," *Journal of Field Robotics*, vol. 27, no. 6, pp. 876–889, 2010.

[10] P. K. Karn, "MACA - a new channel access method for packet radio," in *9th ARRL Computer Networking Conference*, London, Ontario, Canada, 1990.

[11] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A medium access protocol for wireless LAN's," in *Proceedings of ACM SIGCOMM Conference (SIGCOMM '94)*, Aug. 1994, pp. 212–225.

[12] B. Peleato and M. Stojanovic, "Distance aware collision avoidance protocol for ad-hoc underwater acoustic sensor networks," *Communications Letters, IEEE*, vol. 11, no. 12, pp. 1025–1027, 2007.

[13] Google, "Protocol buffers," http://code.google.com/apis/protocolbuffers/, 2011.

[14] R. Brooks, "A robust layered control system for a mobile robot," *Robotics and Automation, IEEE Journal of*, vol. 2, no. 1, pp. 14–23, Mar. 1986.

[15] M. Benjamin, H. Schmidt, P. Newman, and J. Leonard, "Nested autonomy for unmanned marine vehicles with MOOS-IvP," *Journal of Field Robotics*, vol. 27, no. 6, pp. 834–875, November 2010.