Advances in Integrating Autonomy with Acoustic Communications for Intelligent Networks of Marine Robots

by

Toby Edwin Schneider

B.A., Physics, Williams College (2007)

Submitted to the Joint Program in Applied Ocean Science & Engineering
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Oceanographic Engineering
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
and the
WOODS HOLE OCEANOGRAPHIC INSTITUTION

February, 2013

Author .....
Joint Program in Oceanography/Applied Ocean Science & Engineering
Massachusetts Institute of Technology
and Woods Hole Oceanographic Institution
January 22, 2013

Certified by .....
Henrik Schmidt
Professor of Mechanical and Ocean Engineering
Massachusetts Institute of Technology
Thesis Supervisor

Accepted by .....
David E. Hardt
Chairman, Committee for Graduate Students
Massachusetts Institute of Technology

Accepted by .....
Henrik Schmidt
Chairman, Joint Committee for Applied Ocean Science & Engineering
Massachusetts Institute of Technology
Woods Hole Oceanographic Institution

Advances in Integrating Autonomy with Acoustic Communications for Intelligent Networks of Marine Robots

by

Toby Edwin Schneider

Submitted to the MIT/WHOI Joint Program in Applied Ocean Science & Engineering on January 22, 2013, in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Oceanographic Engineering

Abstract

Autonomous marine vehicles are increasingly used in clusters for an array of oceanographic tasks. The effectiveness of this collaboration is often limited by communications: throughput, latency, and ease of reconfiguration. This thesis argues that improved communication on intelligent marine robotic agents can be gained from acting on knowledge gained by improved awareness of the physical acoustic link and higher network layers by the AUV's decision making software.

This thesis presents a modular acoustic networking framework, realized through a C++ library called `goby-acomms`, to provide collaborating underwater vehicles with an efficient short-range single-hop network. `goby-acomms` is comprised of four components that provide: 1) losslessly compressed encoding of short messages; 2) a set of message queues that dynamically prioritize messages based both on overall importance and time sensitivity; 3) Time Division Multiple Access (TDMA) Medium Access Control (MAC) with automatic discovery; and 4) an abstract acoustic modem driver.

Building on this networking framework, two approaches that use the vehicle's "intelligence" to improve communications are presented. The first is a "non-disruptive" approach which is a novel technique for using state observers in conjunction with an entropy source encoder to enable highly compressed telemetry of autonomous underwater vehicle (AUV) position vectors. This system was analyzed on experimental data and implemented on a fielded vehicle. Using an adaptive probability distribution in combination with either of two state observer models, greater than 90% compression, relative to a 32-bit integer baseline, was achieved.

The second approach is "disruptive," as it changes the vehicle's course to effect an improvement in the communications channel. A hybrid data- and model-based autonomous environmental adaptation framework is presented which allows autonomous underwater vehicles (AUVs) with acoustic sensors to follow a path which optimizes their ability to maintain connectivity with an acoustic contact for optimal sensing or communication.

Thesis Supervisor: Henrik Schmidt
Title: Professor of Mechanical and Ocean Engineering

Biography

Toby Schneider grew up in Woodbridge, CT where he attended Amity Regional High School, engaging himself in his free time with gardening, model rocketry and programming graphing calculators. He received his Bachelor's degree in Physics from Williams College in Williamstown, MA, which is a beautiful small town no one has heard of (but is pretty much like Woods Hole with mountains instead of an ocean). During his summers he first worked on the dispersal of corn pollen at the CT Agricultural Experiment Station, and then in the subsequent summers performed experimental research in laser physics at Williams. This research went on to form the basis of an undergraduate thesis on "Precision phase shift spectroscopy in thallium".

After college, Toby entered the MIT/WHOI Joint Program to perform the work which comprises this thesis. He has participated in at least fifteen sea trials throughout North America and Italy, and enjoys seeing his engineering efforts working in the real world. His hobbies now including hiking and working in his backyard mini-farm.

Acknowledgment

Many thanks to Henrik for guiding the overall arch of my work and for providing an abundance of opportunities for doing real engineering at sea. I also greatly appreciate the advice of my thesis committee members, Jim and Hanu, which has undoubtedly made this a better work.

Thanks to all the past and present members of my lab at MIT (LAMSS). Ocean robotics is truly a team effort and you've all contributed important pieces. To my fellow adventurers in Italy for the first times in 2008: Kevin, it was a real pleasure working with you and I'm very sad we live on opposite coasts now; Arjuna, thanks for the pizza on the late nights soldering the payload before SWAMIS09; and Joe, I will never forget hearing your life story over the world's longest lunch in Campiglia. To the software core: Mike, thanks for all the spirited debates on middleware and for the IvP Helm, which is a solid piece of work; Henrik, for simulating enough that my bugs always showed up; Alon, for all the bits of fascinating trivia; and Ian, for inspiring me to continue to try to fix MOOS. And to my contemporaries: Steph ("the original"), for being my wingmate on every cruise (and in life); Erin and Sheida, for taking up the operational duties when I had to focus on finishing this thesis, and Stephanie (Fried), for reminding me that not everyone wants to be a Linux guru.

The NATO Centre for Maritime Research and Experimentation (up until 2012 known as NURC) in La Spezia, Italy, deserves special recognition for the GLINT08 through GLINT10 and SWAMSI11 sea trials they coordinated and for which they provided valuable material and intellectual resources. Thanks also to my mentor Tom Pastore during my summer 2009 internship at the Centre for his the advice and support.

The WHOI Acoustic Communications group (the Micro-Modem folks) headed by Lee Freitag also deserve kudos for their quick and helpful guidance pertaining to all my (often

ignorant) requests for assistance and for their help on the GLINT08 and TIGER12 experiments.

Also, I want to thank the Hanu's SeaBED group at WHOI for making available the data from the AGAVE07 expedition. Also, I specifically want to thank Chris Murphy for his valuable critique of Goby-Acomms and for providing me a chance to participate in his CAPTURE11 experiment.

Bluefin Robotics deserves credit for their many hours of operations support, often going beyond the required or expected. Wes, thanks for helping make the Bluefin/MIT team work smoothly on all our trials.

Many thanks to all the folks who wrote and contributed to the open-source projects used in my work, from GNU/Linux to MOOS-IvP, the Acoustics Toolbox, and all the libraries used in Goby, especially the Boost C++ library.

Furthermore, thanks to my many good friends in the Joint Program, who have helped pass many cold winters and tourist-infested summers with cheer and good fun. Also, I would not be here without the love and encouragement of my family, and for that, I am deeply thankful and blessed.

Finally, I wish to acknowledge the sponsors of this research for their generous support of my tuition, stipend, and research:

- the WHOI/MIT Joint Program
- the MIT Presidential Fellowship
- the Office of Naval Research (ONR) # N00014-08-1-0011, # N00014-08-1-0013, and the ONR PlusNet Program Graduate Fellowship
- the Defense Advanced Research Projects Agency (DARPA) (Deep Sea Operations: Applied Physical Sciences (APS) Award # APS 11-15 3352-006, APS 11-15-3352-215 ST 2.6 and 2.7)

Contents

1 Introduction

> Il semble que la perfection soit atteinte non quand il n'y a plus rien à ajouter, mais quand il n'y a plus rien à retrancher.[1]
>
> Antoine de Saint-Exupéry, *Terre des Hommes* (1939)

1.1 Motivation

In recent years, autonomous underwater vehicles (AUVs) and other marine robots have gone from being nifty toys for engineers to serving valuable scientific, military, and exploration roles, often with real social consequences. To highlight a few examples, in the last decade, AUV researchers have

- surveyed the Arctic with implications for future extraterrestrial exploration [1].
- measured and surveyed the Deepwater Horizon oil spill plume [2].
- performed passive ranging to a sub-sea target (e.g. submarine) using an underwater hydrophone array [3].
- located the downed Air France Flight [4] to recover valuable post-mortem data.
- discovered the pink terraces of New Zealand [5].

AUVs provide spatial coverage unattainable by fixed sensors, and sensor payload flexibility at very favorable costs compared to manned systems. Rapid advances in consumer electronics have allowed AUVs to be outfitted with increasingly powerful computational ability, leading a trend toward more autonomy in navigation, data acquisition, and data

[1] Perfection is achieved, not when there is nothing more to add, but rather when there is nothing more to take away.

processing. Sensors, such as sonars and conductivity-temperature-depth (CTD) probes, have been miniaturized and fitted on AUVs of all sizes. Industry maturation has led to reduced vehicle cost, allowing more institutions to own and operate fleets of vehicles, rather than maintaining a single, costly robot. This increased ability to realize autonomy and affordably field clusters of AUVs is desirable as it can lend spatial diversity to sensing tasks and redundancy to failure.

All of these advances motivate an increased need for underwater wireless communication. Collaboration between autonomous assets requires a certain exchange of information, almost by definition. Sensor data, either raw or processed, is often of a timely nature and cannot wait until the end of the mission to reach the human scientists or operators. Data that reaches operators during the mission can allow a certain parallelism between robotic and human intelligence: the people can make decisions based on these data to influence the robotic system's future behavior. These decisions must be propagated to the deployed nodes, which again requires the ability to send underwater telegrams.

AUVs are generally operated wirelessly for obvious practical reasons, and this is what distinguishes them from remotely operated vehicles (ROVs). Wireless telemetry in the ocean over any significant range is a difficult task to accomplish. The usual suspect for a carrying data signals, electromagnetic radiation, is rapidly attenuated in sea water. Thus, most systems based on light or radio waves are only practical for ranges on the order of tens of meters (see results in [6–8]). While such systems have some specific niche uses (such as "tetherless" ROVs and "data mule" systems), it is unlikely that groups of AUVs will be deployed so densely in the foreseeable future.

Therefore, undersea communication over any significant range is widely accepted to be only practical using an acoustic carrier [9]. The quality of acoustic communications is often poor (low baud rates with high latency) due a number of physical realities of acoustic waves discussed in Section 1.2.1. This has led to a significant push to provide increased autonomy capabilities on AUVs to compensate for the lack of available communication throughput which precludes direct human teleoperation (which is common in land and air robotics). However, there has been little crossover between underwater autonomy and acoustic communications, with the former community generally treating the physical link as a "black box" that sends bytes from one point to another.

This thesis argues that improved communication can be gained from acting on knowledge gained by improved awareness of the physical acoustic link and higher

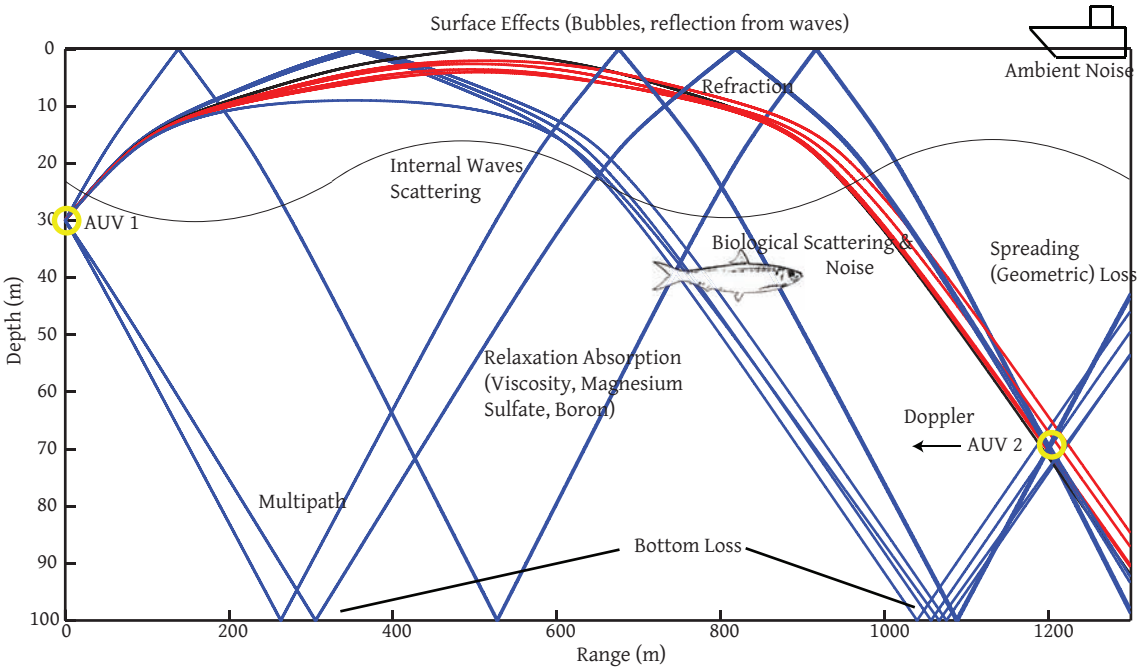


Figure 1.1: Sketch of the relevant oceanographic effects on the performance of acoustic links.

network layers by the AUV's decision making software. Here, "improving communications" is defined as increasing the unit information throughput per unit power ratio. Before getting into detail about the contributions that support this assertions, it is instructive to examine the historical record of work in both of the disciplines this thesis aims to knit together: underwater acoustic telemetry and vehicle autonomy.

1.2 Historical Background

1.2.1 Physical acoustic links

Underwater acousticians and signal processing researchers have characterized many of the detrimental effects of the ocean acoustic environment on successful transmission of datagrams; the difficulty of obtaining high rate acoustic transmissions due to the ocean environment and how this impacts acoustic networking is summarized by many researchers such as Baggeroer [10], Kilfoyle [11], Preisig [12], Stojanovic [13], Chitre [14], and Partan [15].

A summary of these limiting factors (sketched in Fig. 1.1) includes:

- Low bandwidth: due to physical and oceanographic processes (primarily viscous absorption and chemical relaxation from magnesium sulfate and boric acid [16]), the ocean is a low pass filter for acoustic waves.
- Slow propagation speeds: the speed of sound (which depends on the ocean pressure, temperature, and salinity) is about 1490 m/s at a temperature of 10° C, salinity of 35, and depth of 100 meters [17]. Compared to the speed of light in a vacuum ($3.0 \cdot 10^8$ m/s), sound in the ocean travels five orders of magnitude slower.
- Multipath: multiple delayed copies of the signal reaches the receiver primarily due to interface (surface / bottom) reflections and significant refraction from the stratified ocean.
- Doppler effects: due to relative platform motion (typically on the non-negligible order of 0.1% of the speed of sound) the received signal is typically shifted in frequency (which is a narrowband approximation as the Doppler shift is a function of the emitted frequency). Also, due to refraction and waves (both sea surface and internal), different paths have different Doppler shifts, which leads to a varying Doppler shift in the received impulse response as a function of arrival delay. Finally, due to fluctuations in multiple paths with similar delays (such as caused by focusing from sea surface waves [18]), the received signal shows a frequency spread.
- Signal-to-noise ratio (SNR): As is inherent in the name, the SNR is determined by the source level (which is limited by available power and transducer technology) reduced by the transmission loss (a combination geometrical spreading, absorption, and refraction) and the noise level (anthropogenic, biological, physical).

All of these effects combine to make the design of wireless acoustic links challenging. The broad arch of the underwater communications signal processing community has been from reasonably reliable but inefficient incoherent modulation schemes to phase coherent modulation. Coherent modulation schemes require significant effort to remove intersymbol interference (ISI) from the aforementioned channel effects, but achieve more efficient use of the available bandwidth.

Incoherent techniques, such as frequency shift keying, simply measure the energy in a given frequency range to determine the received symbol. By adding pseudo-random hopping sequences to allow longer channel clearing times, multipath effects are reduced (at least in channels with short enough decay times), and by making the frequency bin for each symbol wide enough, Doppler effects are mitigated. However, this technique is

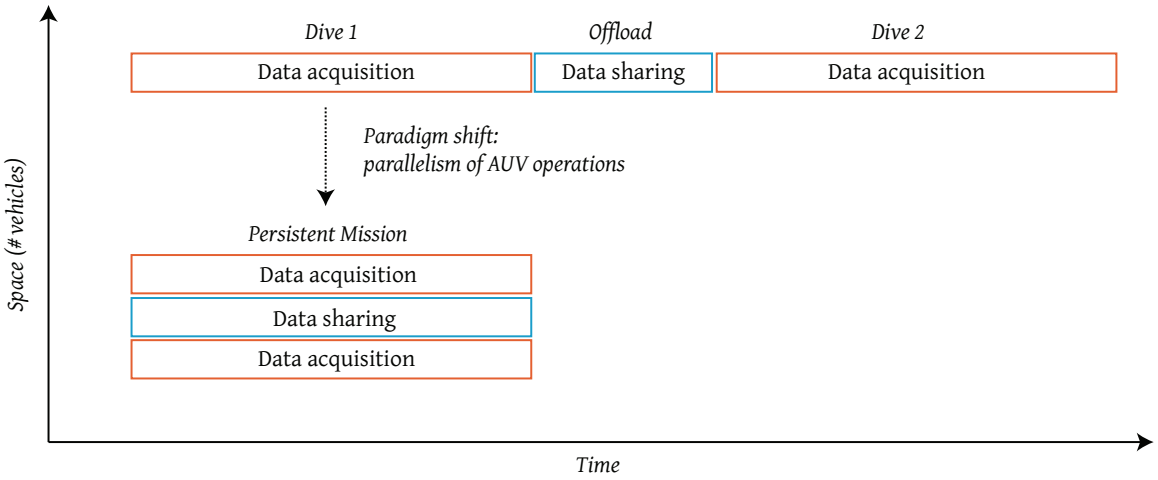


Figure 1.2: Tradeoff between number of vehicles and time required to complete a mission. With an effective wireless communications network, parallelization of vehicle operations has the potential to more efficient as well as being quicker.

an inefficient use of already highly limited bandwidth.

Approaches that attempt to improve on this inefficiency quickly run into the fundamental time-frequency tradeoff combined with the time and frequency spreading nature of the acoustic channel. Coherent single carrier modulation is Doppler resistant but must compensate for ISI from multipath due to the short (in time) symbol size. On the other hand, many groups are looking at using multiple carrier (e.g. OFDM [19]) which is inherently less susceptible to multipath, but must compensate for ISI from Doppler (synchronization) caused by the narrow (in frequency) subcarrier width.

Regardless of the signaling approach taken, the theoretical (and practical, if time and frequency spreading effects can be sufficiently removed) limit on throughput will always be bounded by the bandwidth and the SNR, as shown by the famous Shannon-Hartley theorem.

1.2.2 Underwater vehicle autonomy

Autonomous decision making concerning navigation of underwater robots is still a nascent field. The challenges and costs of designing, controlling, and fielding vehicles has consumed much of the research effort in underwater robotics thus far. Missions are largely pre-programmed surveys, possibly with basic runtime redirection from a human operator in light of an event. After the mission completes, the data are offloaded and analyzed. Then the robot is reprogrammed and deployed again. This paradigm shift is

illustrated in Fig. 1.2.

As a result of increased commercialization of vehicles (and thus improved robustness and somewhat reduced costs), it is becoming more practical to field clusters of robots at once. This opens up the possibility of reduced spatial/temporal aliasing of sampling for oceanographic work, and collaborative missions for reduced mission time for time-critical applications such as mine countermeasures or target detection. Along with this shift towards increased parallelism of operations comes a requirement for improved communications performance. Essentially by definition, robotic collaboration requires communication.

The limited communications available due to the effects outlined in Section 1.2.1 have led to advances in autonomous decision making, such as the Interval Programming (IvP) Helm. Such systems have only had limited success in removing the need for human interaction, as years of artificial intelligence (AI) work in non-marine domains has also shown. Humans are incredibly good at heuristic judgment, while machines are highly effective at mathematical computation and data storage. Pragmatic AI will leverage the best of both aspects of a computer/human system, rather than seek to supplant humans with robots. In the ocean environment, safety and cost considerations lead to the conclusion that the humans should stay ship (or shore) side. This again leads to the need for effective wireless networking.

Thus far, the need for networking for real marine robotic systems has led to a number of ad-hoc projects that serve specific goals (primarily the monitoring of one vehicle's state during a pre-planned mission). Carryovers from terrestrial networking (an incredibly widely studied problem since the advent of the global Internet) do not work well in the marine domain because of two main factors (illustrated in Fig. 1.3):

- Very low throughput - traditional packet designs such as the Internet Protocol suite (IP) do not function efficiently or effectively. Further more, the assumption of total throughput is generally invalid (TCP breaks down). Vehicles have more data to share than will ever be able to be transmitted acoustically.
- High latency - Challenges for MAC, routing, and guaranteed delivery.

Several researchers report on fielded networks of AUVs, each with its own advances and limitations:

- Persistent Littoral Undersea Networking (PLUSNet) [20]: The PLUSNet project demonstrated transfer of three message types (in the Compact Control Language (CCL)

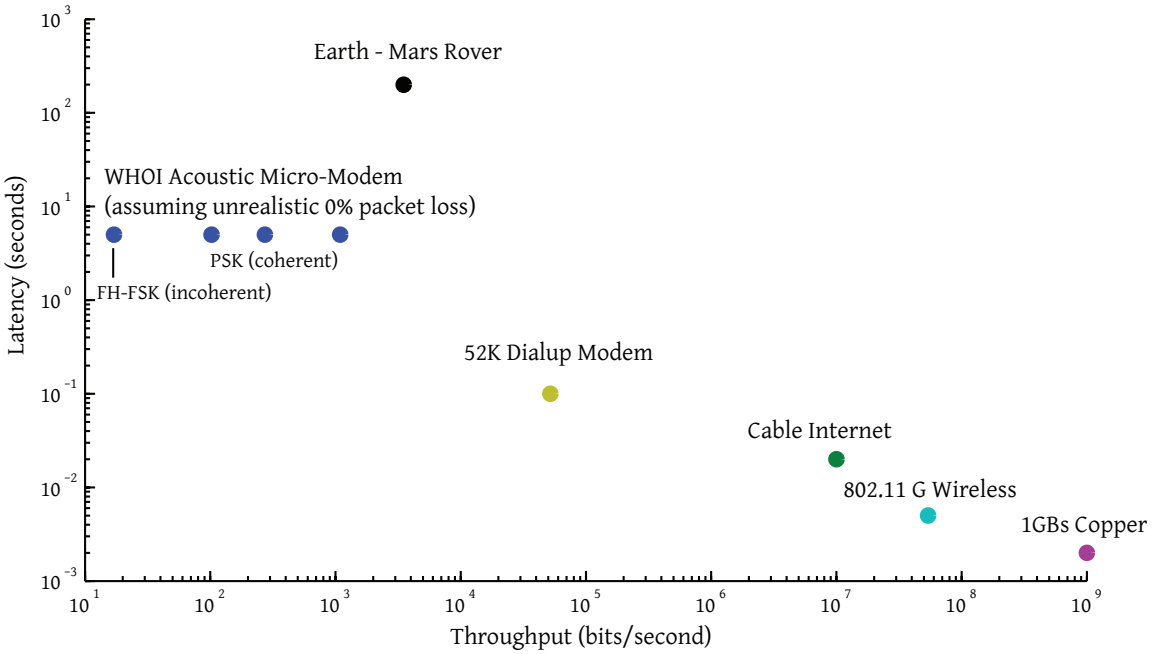


Figure 1.3: Log-log comparison of a widely used acoustic modem (the WHOI Micro-Modem) with various other terrestrial and extraterrestrial wireless and wired physical links. The underwater acoustic link (even with newer advances in modulation) is the most throughput constrained and has the highest latency, save for inter-planetary links.

[21]) between a mixed network of AUVs, buoys and moored nodes. However, the networking code was not especially robust, and no tractable mechanism for sending messages outside of the three predefined hard-coded types was available. Furthermore, the entire system was tied to one acoustic modem (the WHOI Micro-Modem). To a large degree, the experiences from PLUSNet led to the development of Goby-Acomms (Chapter 2).

- SeaWeb [22, 23]: SeaWeb demonstrated selective automated repeat-request (ARQ) which improves upon basic ARQ, especially on high-latency links. A large network of mostly stationary nodes was used, and routing based on neighbor sensing was performed, though the details in the papers are sparse. In this case, solely the Teledyne Benthos modem was used. Little information is suggested about data presentation (source coding).

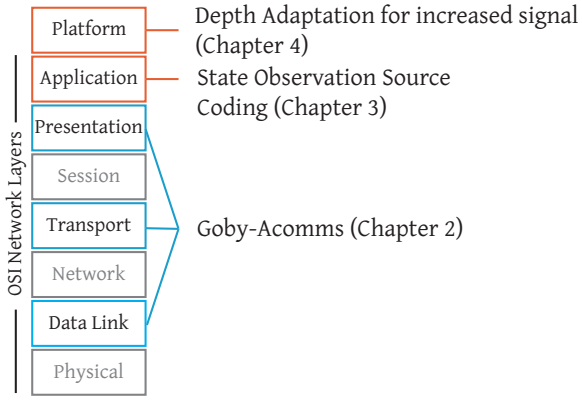


Figure 1.4: Outline of this thesis' contributions in the context of the Open Systems Initiative (OSI) network layers [24]. While potentially misleading (at least some cross-layering is necessary for efficiency), the OSI model provides a common starting point for discussion of networks. The so-called "platform" layer is an innovation for robotics work and represents the participation of an entire vehicle (e.g. movement) in the network.

1.3 Contributions

As shown in Fig. 1.4, the contributions in this thesis come together to form an incomplete, but nonetheless valuable networking system for fielded AUVs. All of the chapters contain work performed or demonstrated using AUVs operated under the "Unified C2" paradigm, much of which was developed by the author. See Appendix A for technical details on this operation setup.

Starting from closest to the hardware, the Goby-Acomms library, a networking suite specifically designed for marine links with high latency and low throughput, is presented. Next, two techniques are presented (and others are suggested) for improving communications by making use of the vehicle's intelligence; they are split into non-disruptive (no negative effect on the mission objectives) and disruptive (requires movement of the vehicle that may be orthogonal to the overall mission objectives). A summary of the methods discussed here is given in Table 1.1. These advances are complementary to advances in signal processing and acoustic modem design, and can be thought of comprising the application and platform layers of the network, respectively.

1.3.1 Goby-Acomms: Rethinking network protocol design for the underwater environment

It may be tempting to believe that the design of such a protocol suite is a solved problem; after all, the ubiquitous Internet Protocol Suite (which includes the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP) built upon the Internet Protocol (IP)) has successfully demonstrated its success. However the IP suite was specifically designed for links that achieve total throughput and are typically significantly underutilized with reasonably low latencies (order of milliseconds or less), such as Ethernet. In contrast, typical underwater links, such as those built using acoustic modems, rarely achieve total throughput in real situations and are thus run at full capacity. Furthermore, they have high latencies (order of seconds or more). Thus, the design decisions made for the IP suite do not work for marine links and in some cases are counterproductive.

The Goby-Acomms suite presented in Chapter 2 provides four key advances (listed in order from closest to the application to closest to the physical link) intended to address the limits of traditional networking systems in light of the extreme bandwidth and latency constraints of underwater links:

1. The **Dynamic Compact Control Language (DCCL)** is a marshalling (or synonymously serialization) scheme that creates highly compressed small messages suitable for sending over links with very low maximum transmission units (order of 10s to 100s of bytes) such as typical underwater acoustic modems. DCCL provides greater efficiency (i.e. smaller messages) than existing marine (CCL, Inter-Module Communication) and non-marine (Google Protobuf, ASN.1, boost::serialization, etc.) techniques by pre-sharing all structural information and bounding message fields to minimum and maximum values (which then create messages of any bit size, not limited by integer multiples of octets such as int16, int32, etc.). The DCCL structure language is independent of a given programming language and provides compile-time type safety and syntax checking, both of which are important for fielding complex robotic systems. Finally, DCCL is extensible to allow user-provided source encoders for any given field or message type, which enabled the work in Chapter 3 to be easily demonstrated and used on real, fielded vehicles.

2. The transport layer of Goby-Acomms provides time dynamic priority queuing (**Goby-Queue**). In our experience, acoustic links on fielded vehicles have been typically run at over-capacity; that is, there are more data to send than will ever be send over

the link. Thus, the data that are to be sent must be chosen in some fashion. Historically, priority queues are widely used to send more valuable data first. However, different types of data also have different time sensitivities, which Goby-Queue recognizes via the use of a (clock time based) time-to-live parameter. Finally, the demand for a given type of data can increase over time since last receiving a message of that type. Goby-Queue extends the traditional priority queue concept to balance these various demands and send the most valuable data under this set of metrics.

3. Acoustic modems such as the WHOI Micro-Modem do not provide any shared access of the acoustic channel. Coordinating shared access can be accomplished by assigning slots of time in which each vehicle can transmit, which is the time-division multiple access (TDMA) flavor of medium access control (MAC). The **Goby-Acomms acoustic MAC (AMAC)** extends the basic TDMA idea to include passive (i.e. no data overhead) auto-discovery of vehicles in a small, equally time-shared network. Thus, AMAC simplifies the amount of pre-deployment configuration required to configure small networks of AUVs.

4. The **Goby ModemDriver** provides an abstract interface for acoustic modems (and other "slow link" devices, such as satellite modems), as there is no standard for interfacing to such devices. Many acoustic modems provide functionality beyond the strict definition of a modem (which is defined as sending data from one point to another). Examples of these extra features include navigation (long base line or LBL, ultra-short base line or USBL) and ranging measurements ("pings"). Goby ModemDriver allows an application intent only on transmitting data to operate on any implemented modem without concerning itself with the details of that device. On the other hand, if the application needs to use some of the extra features, it can do so via a set of well-defined extensions.

1.3.2 Non-disruptive Techniques

Some methods that do not require potentially unacceptable changes to the mission include selecting modem parameters based on propagation models and use of entropy source coding, such arithmetic coding.

Rather than moving the vehicle as suggested in the next section and detailed in chapter 4, the vehicle can select an optimal bit-rate, transmit power or frequency band. Given that the acoustic absorption per unit distance increases with frequency, whereas the am-

Table 1.1: Techniques to improve acoustic communications available to an artificially intelligent AUV

Description	Disruptive	Improvement	Requires	Advantages	Disadvantages
Move to close range with receiver ("data muling")	yes	higher SNR or reduced multipath	receiver position	large gains	extremely disruptive
Change depth to expected beneficial position (Chapter 4)	yes	higher SNR or reduced multipath	receiver position, propagation model	potentially large gains (e.g. SOFAR channel)	quite disruptive
Stop to transmit or receive	yes	less Doppler, lower noise	source transmit time, self-noise model	minimally disruptive	synchronization with transmitter
Frequency selection	no	higher SNR	receiver position, propagation model, broadband or multi-channel hardware		extra hardware
Transmit power selection	no	lower power	receiver position, propagation model	easy to implement	accurate modeling difficult, cannot use Class D amplifiers
Entropy-based source encoding (Chapter 3)	no	increased information per bit	data model	complementary to other techniques	data modeling time consuming or difficult

bient noise level generally decreases with frequency, a maximum in the expected signal-to-noise ratio exists varies as a function of carrier frequency as shown in [25]. Thus, a vehicle aware of the range to its communicating partner can select the available carrier frequency that is expected to be closest to this maximum SNR. Similarly, the power can be adjusted to reach an expected acceptable threshold for a given modulation scheme and bit rate. Due to the realities of designing broadband transducers, very wide band modems required to make this technique feasible are not presently available. However, a vehicle equipped with two transducers with different bands could select between the two based on the range to the receiver.

Autonomous modeling to improve source coding

A second way of non-disruptively improving acoustic communications is via source coding or compression of the vehicle's data. This thesis shows that the vehicle's intelligence can be used to make significant strides in source coding of telemetered data. Physical models of source data can be compared to the measured data to provide small delta messages to existing entropy encoders, such as arithmetic coding. The resulting encoded message is substantially smaller due to the resulting increase in information content. This technique is non-disruptive and can be complementary to the disruptive techniques. Chapter 3 presents such a system built on the Goby-Acomms library, and is thus suitable for real deployments.

The concept developed in chapter 3 provides a shared (between sender and receiver) decoupled physical model and an arithmetic entropy encoder, which encodes the innovations between the model and the measured data. This concept is applicable to numerous data sources on a AUV, and since the two components (model and encoder) are isolated, the work on the encoder need not be duplicated for future applications of this technique.

Several potential applications of this concept include:

- Scalar sensor data, such as from conductivity-temperature-depth (CTD), turbidity, pH, oxygen, or pollutants (e.g. hydrocarbons) instrumentation. These measurements can be compared to a shared dynamic environmental forecasting model and efficiently encoded using an entropy encoder.
- Estimated positions of an unknown moving object (such as the bearing and range to a sonar target) combined with a model of the target's dynamics.
- In a similar concept to the unknown target, but for a known vehicle, the position of the sending AUV (the output of its navigation system) can be compared to a shared model of the AUV's motion.

The third of these, telemetry of a time series of an AUV's position, is the application explored in chapter 3. In this case, two dynamics models are developed, a computationally inexpensive fixed-speed motion model that is applicable to a large class of torpedo-shaped AUVs, and a more general purpose Kalman Filter model that introduces vehicle maneuvers via process noise to preserve generality to a large set of vehicles. These two models are compared to the measurements and the differences ("deltas") produced are then provided to a modified integer arithmetic encoder to encode. This technique was

used on a two widely different vehicle types using simulation on experimental data as well as in-situ testing using Goby-Acomms. In all cases, compression ratios (relative to using standard 32-bit integers) exceeding 85% were realized using this technique.

Even in the case where only the newest position of the vehicle is desired (that is, a historical back fill of "missed" positions due to link dropouts is unneeded), this technique provides improved compression for highly lossy links (up to those exceeding about 65% packet loss).

1.3.3 Disruptive Techniques

Since, by definition, AUVs are mobile, the possibility exists for motion (or lack thereof) to effect a change in the physical communications situation. In general, the movement required would be partially orthogonal to the movement goals of the overarching mission (e.g. environmental sampling, hazard detection), hence the use of the term "disruptive."

The disruptive technique diagrammed in chapter 4 aims to improve communication performance by placing the AUV in a position where the acoustic channel is more favorable for telemetry, such as by increasing the expected signal-to-noise ratio (SNR) for use with an incoherent modulation scheme (such as frequency-hopping frequency shift keying).

In order to accomplish this, the vehicle must model the acoustic channel. Existing acoustic models are not designed for "online" use by an autonomous embedded computing system, having been developed for human-based "offline" use, typically using powerful computational hardware. To make such models usable in this new context, the Generic Robotic Acoustic Modeling (GRAM) concept was developed. GRAM provides an abstract remote-procedure call (RPC) interface to legacy (but still valuable) acoustic modeling codes; as well, it provides an incremental update mechanism the reduces the overhead involved with each invocation of the acoustic model code.

GRAM is then used in conjunction with the existing BELLHOP code from [26] to provide forecasts of the predicted acoustic transmission loss between an AUV and its receiver. The inputs to the model include sound speed data calculated from measurements on-board the AUV. Using the results of the model, an autonomy behavior developed for this work produces a function of utility for the vehicle over all reachable depths. This behavior is then solved using the existing IvP Helm multi-objective decision engine presented in [27]. Thus, the specific contributions of this chapter to the thesis are the GRAM concept and implementation ("online" acoustic modeling by a robotic agent), the depth-

adaptive autonomous behavior (BHV_AcommsDepth and the derived BHV_MaxSNRDepth), and the environmental-feedback framework (see Fig. 4.1) that utilizes these components in a fielded AUV.

Another technique that is not explored in detail in this thesis involves slowing or stopping the vehicle to mitigate self-noise and/or Doppler effects. Certain modulation schemes, such as orthogonal frequency division multiplexing (OFDM), are highly sensitive to Doppler shifts. Since vehicle speeds (order of 10^0 m/s) are not negligible compared to the speed of sound in sea water (order of 10^3 m/s), normal vehicle motion can be disruptive to successful communications. Whether this technique is useful or not depends on the autonomy system understanding the requirements of the acoustic modem; it makes no sense to stop the vehicle if the modem's modulation is immune to the relevant frequency shifts. A second reason to arrest the motion of the vehicle is just prior to receipt of a telegram to reduce the vehicle's self noise. Zimmerman, et al [28] found that the self-noise of a typical mid-size AUV (the Bluefin Odyssey IIb) was 20 to 50 dB higher than the ocean background noise levels in the 10Hz to 10kHz range; the upper end of this band is commonly used for AUV communication. Most of this noise is motion related (propeller and turbulence), suggesting that much of this noise could be removed by stopping the vehicle temporarily to receive communications. However, such a scheme requires knowledge of incoming transmissions, which can be pre-arranged (e.g. fixed time division multiple access (TDMA) medium access control (MAC)) or negotiated (e.g. request-to-send/clear-to-send style MAC schemes).

Both of these techniques have the obvious disadvantage of taking time and power away from the core mission for the purpose of communications. However, many missions (especially detection of mines or submarines) are inherently useless without timely reports of collected data. Thus, when using these disruptive techniques some form of multi-objective optimization needs to be used. Otherwise, there is a risk of the mission collapsing to the degenerate case where all of the mission time is spent communicating useless data.

2 Goby-Acomms: A modular acoustic networking framework for short-range marine vehicle communications

2.1 Introduction

For successful collaboration of autonomous underwater vehicles (AUVs) in tasks ranging from the scientific (e.g. oceanographic sensing; see [29]) to commercial and military (e.g. harbor surveillance; see [30]), transmission of datagrams is essential to propagate state and sensor data. However, the only practical long range communications link, one carried by acoustics, has extremely low throughput and high latencies due to the physics governing propagation of sound in the ocean (primarily little available bandwidth, low speed, and multipath due to boundary interactions). For an overview of these challenges see [14] and [12].

Another significant reality caused by the acoustic link's low data rates is that total throughput is rarely or never a possibility. Ideally all collaborating vehicles and the topside human operator would have the entire data set of all vehicles in the network in order to make the best mission decisions. Instead, only a miniscule subset of the data generated by an AUV can be relayed acoustically. This leads to a significant prioritization problem; some of this queuing can be automated by the Goby dynamic priority queuing module (*queue*, see section 2.3).

The final significant challenge for underwater telemetry addressed by Goby is the lack of standardization or even significant commonality between different acoustic modems. In order to allow communication that appears seamless to the application in a variety of hardware environments, the Goby interface to the link layer (*modemdriver*, see section 2.5) is extensible and polymorphic, allowing Goby to communicate over any device that

can send bytes from point A to point B. At the same time, the application can still use hardware-specific features as desired. In collaboration with the *modemdriver*, Goby provides basic time division multiple access (TDMA) medium access control (MAC) with a novel peer autodiscovery mechanism (section 2.4).

Networking is a well studied problem in the terrestial domain; a prime example is the Internet Protocol (IP) Suite, which is comprised of TCP and UDP. However, the limitations to throughput and latency in an underwater acoustic network suggest we should perform careful analysis before applying terrestial networking solutions to the marine environment. Specifically, we suggest that certain tradeoffs of efficiency for abstraction that are desirable on high throughput, low latency links involving thousands of computers are not desirable for the low throughput, high latency acoustic links involving at most tens of autonomous underwater vehicles (AUVs).

A common form of networking abstraction is the concept of "layers" (together, the layers form a network "stack"). The Open System Interconnection Reference Model (OSI Model) presented in [31] provides a framework for this type of abstraction. In the OSI Model, each layer is abstracted from the previous layer; that is, higher levels do not need to concern themselves with the implementation details of lower levels. This abstraction allows for complicated systems to be broken into more manageable pieces and is likely a contributer to the success of the internet. However, such layering comes with tradeoffs. Higher layers duplicate header information (such as addressing) and error checking that may be already implemented on one or more of the lower layers. Hence, with `goby-acomms`, in order to produce shorter messages, we chose to maintain the general concept of networking layers, but with more explicit and implicit interactions between layers. Due to the success and widespread knowledge of the IP suite, it will be used as a comparison to `goby-acomms` where possible to elucidate design decisions.

The layers (or modules) of `goby-acomms` are summarized in Table 2.1 with an approximation of the corresponding layer(s) of OSI Model, and illustrated generally in Fig. 2.1. While each layer is dependent on one or more of the other layers, any layer could be replaced as long as the replacement fulfills the necessary interface requirements. This modularity of `goby-acomms` should improve its flexibility for use in a variety of future acoustic networks, as needs change and new research comes to light. For example, a future project that just needs encoding could use *dccl* alone. Or an acoustic network with an improved buffering system could replace *queue* while making use of the remaining layers of `goby-acomms`.

Table 2.1: Comparison of `goby-acomms` layers with those of the OSI Model

OSI Model layer	IP Suite layer	`goby-acomms` layer	Provides
Application	Provided by the IP user	Provided by the `goby-acomms` user.	Configuration and data.
Presentation	Provided by the IP user	*dccl* [a]	Encoding and decoding.
Session	Provided by the IP user	Not used. Sessions are passive.	
Transport	TCP or UDP	*queue* [b]	Priority buffering, concatentation of multiple DCCL messages, and optional guarantee of receipt.
Network	IP	Not provided, but could be added by user into the *dccl* header.	
Data Link	e.g. Ethernet driver	*amac* [c]	Division of time into slots for multiple vehicles over the half-duplex link.
		modemdriver [d]	Configuration of, interaction with, and abstraction of the physical layer.
Physical	e.g. Ethernet	e.g. WHOI Micro-Modem	Transmission and receipt of messages.

[a] section 2.2
[b] section 2.3
[c] section 2.4
[d] section 2.5

2.1.1 A note about Goby versions 1 and 2

Goby grew out of a MOOS project (see section 2.6.1) to become a standalone C++ suite of libraries which was called version 1 (Goby1). After numerous field trials and feedback from external collaborators, a significant overhaul and complete rewrite of much of Goby was completed in light of this knowledge. This became know as Goby version 2 (Goby2). Unless otherwise specified, this chapter refers to the current state of the art, Goby2, but in a number of places the design history of the project is mentioned to explain certain decisions. Often the history of a design choice is critical to understanding its rationale. Successful engineering of real systems is an iterative, incremental process, and the intellectual contribution of this work rests on the ultimate outcome of a several year history of making mistakes or subpar choices, learning from them, and improving the work.

The overall goals of Goby2 versus Goby1 are to

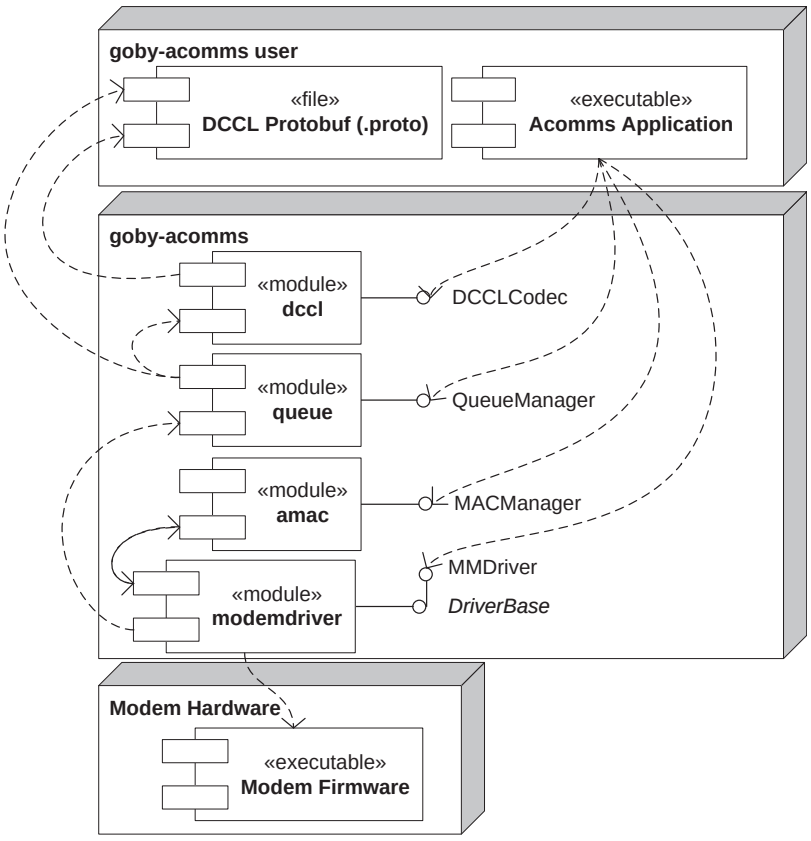


Figure 2.1: Unified Modeling Language (UML) component model of the `goby-acomms` library. Dependencies are indicated with a dotted arrow pointing from an object to its dependency. The interface class to each library is given as a line terminated by a semicircle (e.g. DCCLCodec). UML is presented in [32].

1. improve extensibility by third-party authors. Release 1 was primarily focused on a specific marine vehicle middleware (the MOOS project), and did not offer many expansion opportunities except the ability to define one's own DCCL messages.

2. promote the development of a system that provides high correctness assurance as far before deployment as possible, since ship time is highly valuable and vehicles are costly. The communications system is, almost by definition, an essential part of collaborative vehicle missions, and thus it cannot fail.

Some details about Goby1 are given in the appendix section B.1.

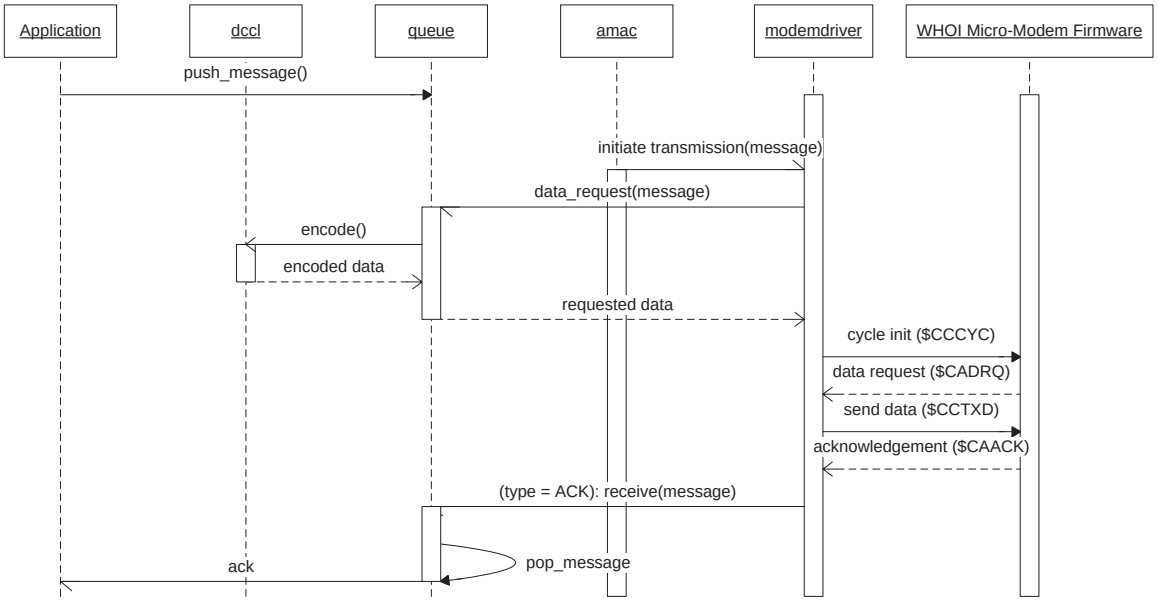


Figure 2.2: The UML sequence diagram for sending a message using all the `goby-acomms` components.

2.2 *DCCL*: data marshalling (or source coding)

The Dynamic Compact Control Language (DCCL) is comprised of two parts designed to make creating very small datagrams straightforward and reliable: a structure language and an encoding library. The language provides a way of representing "methodless classes" or "dumb data objects" that are similar to what can be represented in a C struct, but with the addition of meta-data that provides information allowing optimized encoding and strict upper bounding of the message size.

2.2.1 Motivation

DCCL fulfills the role of data presentation by providing a framework for defining object-based messages and a commmon framework for source encoding and decoding them. Furthermore, each DCCL message is intended to be extended by the user to provide the equivalent of the Internet Protocol's (IP) header information to allow for multiple hop routing. It does not make sense to use IP on acoustic links, largely because the size of the IP header (minimum 20 bytes) is the same order of magnitude of the maximum transmission unit (MTU) of common acoustic modems. For example, the WHOI Micro-Modem uses frames of 32 to 256 bytes. Basagni et al. [33] show that for a representative link

with a bit-error rate of 10^{-4} and bit rate of 200 bits per second (bps), obtaining maximum throughput efficiency of a multi-hop network requires packet sizes of 500 bytes or less. Thus, it seems reasonable to assume that the minimum IP header will comprise at least 5% of an acoustic modem packet satisfying the 500 bytes or less constraint. In the case of the Micro-Modem, this could as much as 62.5% overhead. Given the very low throughput of these links to begin with, this is an unacceptable amount of the message to be used for potentially unneeded information. Once out of the highly restricted domain of acoustic links, it is straightforward and likely desirable to wrap DCCL messages as a payload on IP networks.

In addition to minimizing header size, the limited throughput constraint of acoustic communications suggests that compressing message payloads as much as possible is a useful goal. Due to the high error rates caused by the acoustic channel combined with high latencies (order of seconds to minutes), guaranteeing receipt of multiple frames can often take an unacceptable amount of time for AUV collaboration. Thus, all of `goby-acomms` deals with data frames smaller than or equal to the size of the hardware layer's frame size, and does *not* perform automatic fragmentation. This requires that the application layer produce data that are useful or at least potentially useful as standalone frames. Thus, for the Dynamic Compact Control Language, the user must strictly specify and name the fields that a given message can take. Furthermore, all numeric fields must have tight bounds that represent the realistic set of values that field will take. For example, it is inefficient to use a 32-bit integer to represent the operation depth which might vary at most from 0-11021 meters on Earth and thus fit in 14 bits or less.

2.2.2 Design overview

DCCL is comprised of two components: 1) a structure language based on Google Protocol Buffers (protobuf) with which to define messages (described in section 2.2.3); and 2) a module in the `goby-acomms` C++ library (*dccl*, detailed in section 2.2.4) that validates the DCCL extensions of protobuf and implements consistent encoding and decoding of each message.

In order to produce messages as small as possible, DCCL offers these features:

- Defined bounded field types with customizable ranges. For example, an integer with minimum value of 0 and maximum value of 5000 takes 13 bits instead of the 32 bits often used for the integer type, regardless of whether the full integer type

is needed.

- Dissolved byte boundaries (unaligned messages): fields in the message can be an arbitrary number of bits. Octets (bytes) are only used in the final message produced. There is no delimiter between fields in the encoded bitstream; each encoder is required to produce the exact number of bits consumed by the corresponding decoder.
- Delta-difference encoding of correlated data (e.g. CTD instrument data): rather then sending the full value for each sample in a message, each value is differenced from both a pre-shared key and the first sample within the message. This feature is described in more detail in section 2.6.5.

We also wanted to remove some of the complexity and potential sources of human error involved in binary encoding and bit arithmetic. To make DCCL straightforward, we made several design choices:

- All bounds on types can be specified as any number, such as powers of ten, rather than restricting the message designer to powers of two. This leads to a small inefficiency since the message is encoded by powers of two, but this drawback is balanced by the value of simplicity since the human mind is much more comfortable with powers of ten than powers of two.
- Protobuf is the basis of the markup language that defines the structure of a DCCL message. Protobuf is widely used within Google and is an open source project with excellent documentation and quality.
- Encoding and decoding for basic types are predefined and handled automatically by the DCCL C++ library (*dccl*), meaning that in the vast majority of the cases no new code needs to be written to create or redefine a DCCL message. Writing code on cruises is always a risky endeavor, and minimizing that risk is important to maximizing use of ship time. However, flexibility to define custom algorithms to assist with encoding is provided for the fairly rare case when the basic encoding does not satisfy the needs of a particular message.

2.2.3 DCCL Structure Language

The DCCL structure language (defined in Table 2.2) is based on an extension of the Google Protocol Buffers ("protobuf") language (see [34]). Protobuf was chosen to replace XML,

Table 2.2: Definition of the DCCL Structure Language

Message Extensions[a]				
Extension Name	Extension Type		Explanation	Goby1 equivalent[b]
(dccl.msg).id	int32		Unique identifying integer for this message	<id>
(dccl.msg).max_bytes	uint32		Enforced upper bound for the encoded message	<size>
Field Extensions[c]				
Extension Name	Extension Type	Applicable Fields	Explanation	Goby1 equivalent
(dccl.field).codec	string	all	Codec to use for this field (omit for default)	N/A
(dccl.field).omit	bool	all	Do not include field in encoded message (default = false)	N/A
(dccl.field).precision	int32	double, float	Decimal digits to preserve.	<precision>
(dccl.field).min	double	(u)intN[d], double, float	Minimum value that this field can contain (inclusive)	<min>
(dccl.field).max	double	(u)intN, double, float	Maximum value value that this field can contain (inclusive)	<max>
(dccl.field).max_length	uint32	string, bytes	Maximum length (in bytes) that can be encoded	<max_length>
(dccl.field).max_repeat	uint32	all (repeated)	Maximum number of repeated values.	<array_length>

[a] Extensions of `google.protobuf.MessageOptions`
[b] See section B.1 for legacy Goby1 XML tag definitions.
[c] Extensions of `google.protobuf.FieldOptions`
[d] (u)intN refers to any of the integer types: int32, int64, uint32, uint64, sint32, sint64, fixed32, fixed64, sfixed32, sfixed64

which was used in DCCL1, because it provides static (compile-time) type safety and syntax checking. Developing systems for AUVs involves integrating large codebases from multiple research centers, and without a high degree of compile-time correctness assurance, costly ship time will be wasted tracking down avoidable software bugs. Since DCCL1 was defined in XML (see appendix section B.1), all correctness checking was done at runtime, deferring detection of syntactical and type errors.

Furthermore, Protobuf messages (and thus their derivatives, DCCL messages) use a compiler that produces native C++ classes, allowing for high efficiency which is critical for embedded systems. Due to the fundamental tradeoff between power and longevity (as detailed in [35]), AUVs can only support low performance (but low power) hardware, such as ARM based computers. C++ provides an excellent balance between programming ease and runtime efficiency.

Finally, Protobuf provides class introspection, which allows DCCL to operate on arbitrary user-provided messages, which can be compiled into the application or loaded dynamically either through shared libraries or runtime compilation of the DCCL message definition.

An example of the syntax of the structure language is given in Fig. 2.3. Also in that figure is the encoded size of the message.

Message Design

When designing a DCCL message, a few considerations must be made. Each message needs to be given a identification (ID) number unique within the DCCL network that this message is intended to live by assigning a value to the `option (dccl.msg).id`. By default, DCCL provides a variable integer encoder for the ID that uses one byte for IDs in the range $[0, 128)$ and two bytes for $[128, 32768)$. Sometimes messages may have limited scope or may be mutually exclusive, in which case duplicate IDs may be assigned. Like any other field, this DCCL ID codec can be redefined by the user. For example, a restricted network where only eight message types are needed could use a 3 bit ID field. The value of being able to redefine the header based on the network size is illustrated in Fig. 2.4.

Furthermore, the overall maximum size of the message needs to be determined (`option (dccl.msg).max_bytes`). This may be a constraint imposed by the hardware layer that this message is intended to traverse. In the case of the WHOI Micro-Modem, this should match the frame size of the intended data rate to be used (32 bytes for rate 0, 64 bytes for rate 2, and 256 bytes for rates 3 and 5). The size of the message is given by the sum

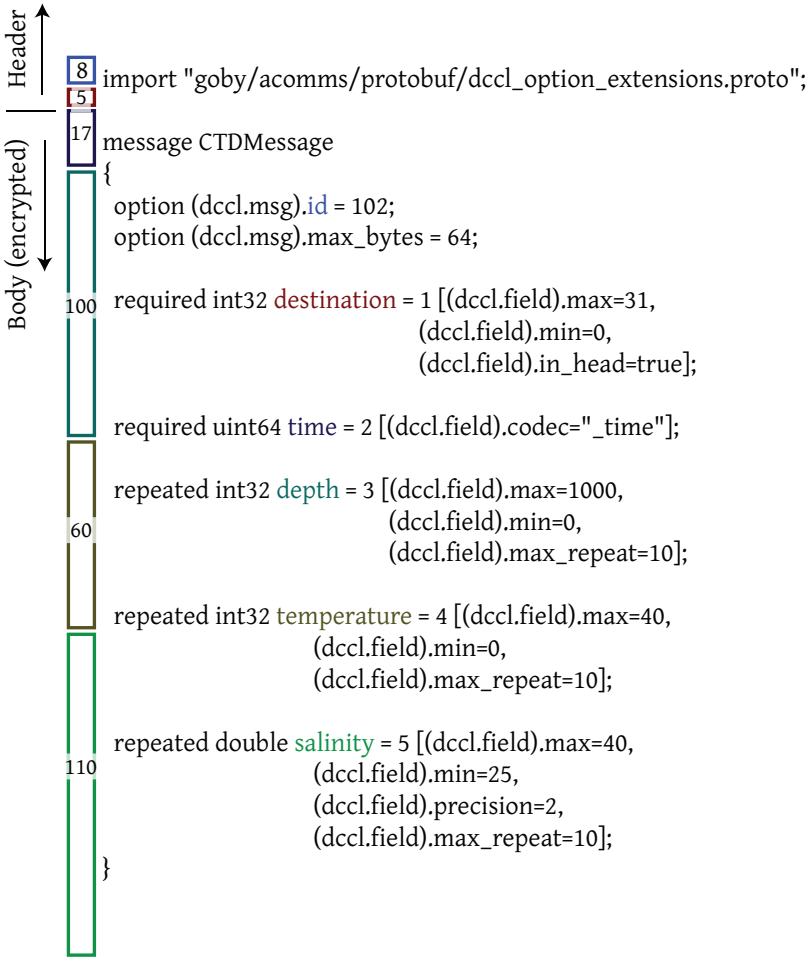


Figure 2.3: Definition of a DCCL message for sending ten samples from a Conductivity-Temperature-Depth (CTD) sensor. On the left is the size of each encoded field in bits; the whole message is 280 bits (35 bytes) including required bit padding on the header and body. For comparison, the default Protobuf encoding uses 81 bytes to encode this message with a representative set of values.

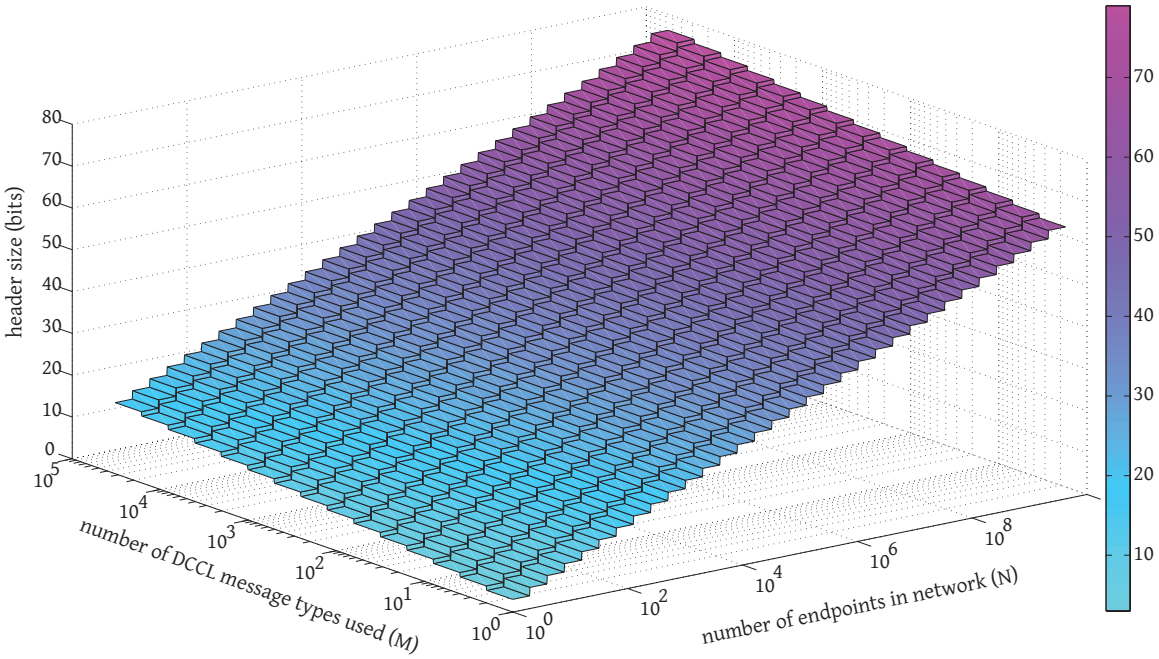


Figure 2.4: Size of a customized DCCL header (containing a type identifier (DCCL ID), source address, and destination address) for varying network sizes N and number of message types M used. In comparison, IP uses a fixed 64 bits total for the source and destination addresses and does not provide any type identification (equivalent on this plot to $N = 2^{32} = 10^{9.6}$, $M = 0$). Since AUV networks typically reside near the origin on this plot, it is valuable to have the ability to target a given network and thus minimize the header size.

of the user defined fields, plus the DCCL ID field. The field sizes are calculated using the expressions given in the "Size" column of Table 2.3. These sizes are calculated at runtime with *dccl*, so it is rarely necessary to calculate these by hand. However, these expressions give a sense of how much space a given field will typically take, which is important when considering how to type and bound the data.

2.2.4 Algorithms and Implementation

Along with the message structure defined in section 2.2.3, DCCL provides a set of consistent encoding and decoding tools in the C++ *dccl* library. The tools provided by *dccl* include:

- Calculation of message field sizes and comparison to the mandated maximum size

(`max_bytes`). Messages exceeding this size are rejected and the designer must choose to remove and/or reduce fields or increase the message `max_bytes`.

- Encoding of DCCL messages using the expressions given in Table 2.3. The user passes an instantiation of a Protobuf message and receives an encoded string of bytes back.
- Decoding of DCCL messages using the reciprocal of the expressions used for encoding.

2.2.5 Encryption

dccl provides pre-shared key encryption of the body portion of the message using the Advanced Encryption Standard (AES or Rijndael) [36]. AES is a National Institute of Standards and Technology (NIST) certified cipher for securely encrypting data. It has been certified by the National Security Agency (NSA) for use encrypting top secret data.

dccl uses a SHA-256 hash of a user provided passphrase to form the secret key for the AES cipher (see [37] for the specification of SHA-256). In order to further secure the message, an initialization vector (IV) is used with the AES cipher. The IV used for DCCL is the most significant 128 bits of a SHA-256 hash of the header of the message. It is a requirement of the DCCL message designer that the message header contain a nonce (such as the time of day) so that it provides the continually changing value required of an IV. This ensures that the ciphertext created from the same data encrypted with the same secret key will only look the same in the future on a given day on the exact second it was created. The open source Crypto++ library available at [38] is used to perform the cryptography tasks.

2.2.6 Comparison to prior work

In the marine community, two contributions provide a similar framework to DCCL: the Compact Control Language (CCL) and Inter-Module Communication (IMC). Furthermore, other application domains have developed approaches to a similar problem. These include various forms of Text Encoding, Abstract Syntax Notation One (ASN.1), and Google Protocol Buffers (whose encoding library DCCL does not use).

Table 2.3: Formulas for encoding the DCCL types.

Protobuf Type	Size (bits)	Encode[a]
bool (required)	1	$x_{enc} = \begin{cases} 1 & \text{if } x \text{ is true} \\ 0 & \text{if } x \text{ is false} \end{cases}$
bool (optional)	2	$x_{enc} = \begin{cases} 2 & \text{if } x \text{ is true} \\ 1 & \text{if } x \text{ is false} \\ 0 & \text{if } x \text{ is undefined} \end{cases}$
enum (required)	$\lceil \log_2(\sum \epsilon_i) \rceil$	$x_{enc} = i$
enum (optional)	$\lceil \log_2(1 + \sum \epsilon_i) \rceil$	$x_{enc} = \begin{cases} i+1 & \text{if } x \in \{\epsilon_i\} \\ 0 & \text{otherwise} \end{cases}$
string	$8 + min(length, max\_length) \cdot 8$	$x_{enc} = length + \sum_{n=0}^{min(length, max\_length)} s(n) \cdot 2^{8(n+1)}$
(u)intN (required)	$\lceil \log_2(x_{max} - x_{min} + 1) \rceil$	$x_{enc} = \begin{cases} x - x_{min} & \text{if } x \in [x_{min}, x_{max}] \\ 0 & \text{otherwise} \end{cases}$
(u)intN (optional)	$\lceil \log_2(x_{max} - x_{min} + 2) \rceil$	$x_{enc} = \begin{cases} x - x_{min} + 1 & \text{if } x \in [x_{min}, x_{max}] \\ 0 & \text{otherwise} \end{cases}$
double, float (required)	$\lceil \log_2((x_{max} - x_{min}) \cdot 10^{prec} + 1) \rceil$	$x_{enc} = \begin{cases} \text{nint}((x - x_{min}) \cdot 10^{prec}) & \text{if } x \in [x_{min}, x_{max}] \\ 0 & \text{otherwise} \end{cases}$
double, float (optional)	$\lceil \log_2((x_{max} - x_{min}) \cdot 10^{prec} + 2) \rceil$	$x_{enc} = \begin{cases} \text{nint}((x - x_{min}) \cdot 10^{prec}) + 1 & \text{if } x \in [x_{min}, x_{max}] \\ 0 & \text{otherwise} \end{cases}$
bytes	$max\_length \cdot 8$	$x_{enc} = x$

· x is the original (and decoded) value; x_{enc} is the encoded value.

· x_{min}, x_{max}, max_length, $prec$ are the value of the (dccl.field).min, (dccl.field).max, (dccl.field).max_length, and (dccl.field).precision options, respectively. ϵ_i is the ith child of the enum definition (where $i = 0, 1, 2, \ldots$), *not* the value assigned to the enum.

· nint(x) means round x to the nearest integer.

· $s(n)$ is the ASCII value of the nth character of the string.

[a] if data are not provided or they are out of range (e.g. $x > max$), they are encoded as zero ($x_{enc} = 0$) and decoded as not present.

Compact Control Language

dccl owes inspiration and part of the name to the Compact Control Language (CCL) developed at WHOI by Roger Stokey and others for the REMUS series of AUVs. An overview of CCL is available in [39], and the specification is given in [21]. In our experience, before DCCL, CCL was the *de facto* standard data marshalling scheme for acoustic networks based on the WHOI Micro-Modem.

DCCL is intended to build on the ideas developed in CCL but with several notable improvements. DCCL provides the ability for messages to adapt quickly to changing needs of the researchers without changing software code (i.e. *dynamic*). CCL messages are hard coded in software while DCCL messages are configured using Protobuf. Furthermore, CCL has no mechanism to including network level header information required for routing. DCCL provides a user-extensible header for such tasks, without increasing the overhead when such information is not required.

Also, significantly smaller messages are created with DCCL than with CCL since the former uses unaligned fields, while the latter, with the exception of a few custom fields (e.g. latitude and longitude), requires that message fields fit into an even number of bytes. Thus, if a value needs eleven bits to be encoded, CCL uses two bytes (sixteen bits), whereas DCCL uses the exact number of bits (eleven in this case). DCCL also offers several features that CCL does not, including encryption, delta-differencing, and data parsing abilities.

To the best of the authors' knowledge (which is supported by Chitre, et al. in [14]), CCL is the only previous effort to provide an open structure for defining messages to be sent through an underwater acoustic network. Other attempts have been ad-hoc encoding for a specific project. In order not to trample on Stokey's work and maximize interoperability, we have made DCCL optionally compatible with a CCL network, giving DCCL the CCL initial byte flag of 0x20 (decimal 32). This allows vehicles using CCL and DCCL to interoperate, assuming all nodes have appropriate encoders for both message languages. When interoperability is not necessary, this byte is omitted, saving message header space.

Inter-Module Communication (IMC)

IMC (see [40]) uses XML with XSLT transformations into the native language code (e.g. C++) that gives a similar language-neutral data object (but with compile-time type safety) to DCCL2. However, IMC uses the standard system primitive types (e.g. 32- and 64-bit integers) and does not allow arbitrary bounding or user-defined codecs.

Text Encoding

Two approaches to encoding that have proven useful in other applications for compressing textual data are dictionary coders (e.g. LZW [41]) and entropy coders (e.g. Huffman coding [42]). Both of these are successful on sparse data, such as human readable text. Their utility for the types of messages encountered commonly in marine robotics is limited, however. These messages tend to be short and full of numeric values, whose information entropy is much greater than that of human generated text. However, careful application of entropy coders to non-text data can produce significant gains, as explored in chapter 3.

Furthermore, the overhead cost incurred by these text encoders means that the compressed message may not be more efficient than the original message until a sizable amount of data (perhaps several kilobytes) has been encoded. This exceeds the size of individual frames in the WHOI Micro-Modem, meaning that in messages would have be split across frames and reassembled. Given the low throughput and high error rate of the acoustic channel, it is impractical to attempt to send a message that is more than several frames before being decodable. Furthermore, the resulting message from these text encoders is variable length, as the compressibility depends on the input data. This can cause further difficulties transporting these data across the acoustic network.

Given these considerations, we decided that currently available text encoders would not an acceptable solution to the problem at hand, i.e. creating short messages for acoustic communications.

Abstract Syntax Notation One

Abstract Syntax Notation One (ASN.1) is a mature and widely used standard for abstractly representing data structures (or messages) in a human-readable textual form. It also specifies a variety of rules for encoding data using the ASN.1 structures. In both these areas, ASN.1 is similar to DCCL: DCCL also provides a structure language (based on XML in this case), and a set of encoding rules. In fact, the rules used by DCCL are very similar to the ASN.1 unaligned Packed Encoding Rules (PER). For a good treatment of ASN.1, see Larmouth's book [43].

Given the severe restrictions on message size due to the acoustic modem hardware, existing ASN.1 structures are unlikely to be useful, unless the designers were originally careful in specifying bounds on numerical types (e.g. INTEGER) and minimizing use of string types (e.g. UTF8STRING). Thus, for simplicity, the authors prefer the protobuf

specification given in Table 2.2 and currently used by DCCL.

Support for ASN.1 may become a desirable goal in the future to take advantage of the knowledge base and experience of this well accepted standard. However, we will likely have to choose a tightly reduced subset of the ASN.1 specification to meet the restrictive demands of the underwater acoustic channel, possibly rendering any gains of being standards-compliant moot.

Google Protocol Buffers

While DCCL only uses Google Protocol Buffers (Protobuf) as a starting point for its language definition, Protobuf also provides a binary encoding. The encoding is reasonably compact, but does not take into account the origin of each field's data, which allows DCCL to provide a more compact encoding. The DCCL encoding of the message given in Fig. 2.3 is 56% more compact than the Protobuf built-in encoding because the DCCL message designer can incorporate information about the fields' physical origins (e.g. salinity is bounded between 25 and 40 in the world's oceans).

DCCL Summary

The design of DCCL can be summarized in a few key points:

- It is worth spending significant human and CPU time designing highly compressed messages since acoustic links are often operated at capacity ("total throughput is rarely achievable on real AUV acoustic networks").
- DCCL provides a minimal header size (in theory as small as 0 bytes) because acoustic modems have small MTUs (order of 10s to 100s of bytes).
- DCCL allows variable length encoders, but enforces a known upper bound on message size to allow a designer to target a given link-layer packet size.

2.3 *queue*: Dynamic priority based buffering

2.3.1 Motivation

Field experience has taught us that in a network of AUVs, desired throughput almost always exceeds the available channel capacity. Based on the available capacity, the engineers and scientists topside prefer to see as much data as possible. The upper limit on

desired throughput would perhaps be a real-time feed of all sensor data, but this can easily be order of megabits per second or higher (especially if video is involved). Maximum data throughput of available commercial modems, such as the WHOI Micro-Modem, is order kilobits per second or much lower in realistic environments. Given that this spread is unlikely to close due to the physical limitations of the acoustic carrier, users will always have be selective about which data are sent over the network.

One solution to this problem is to fix (before launch) a small subset of data that will be transmitted acoustically. Approaches to acoustic networking before *goby-acomms* such as the approaches in [44] and [45] use this solution, typically only sending a vehicle status and maybe a single sensor data type that is most relevant to the experiment at hand. This technique is generally suboptimal, given the designer must account for the worst case communications scenario or risk filling the sending buffers faster than messages can be transmitted over the channel. Due to the highly variable communications environment experienced using acoustics in the ocean, this minimax approach will under-utilize the available capacity.

To better utilize the channel, we need a solution that dynamically scales with the moment-to-moment available capacity. When we have poor throughput, we want to send highly valuable messages. These may correspond to status messages or time sensitive mission specific messages such as target or event detection alerts. When we have good throughput, we also want to send less critical, but still useful data. *queue* provides a prioritized set of buffers with time varying values to effect this desired behavior. Each DCCL message type is assigned a buffer. When the Medium Access Control requests data from *queue*, a priority contest is performed between all the buffers that contain messages. The winning buffer provides data from either its front or back, based on the user's desire for a first-in / first-out (FIFO) or first-in / last-out (FILO) queue respectively.

2.3.2 Prior work

Priority Queues

Priority queues are a widely available container type in modern programming languages. (For example, C++, Java, and Python all provide implementation in their standard libraries). In a priority queue, messages are added with some priority value. When data are requested from the queue, the highest priority data are given first. *queue* provides a dynamic priority queue of (ordinary) double-ended queues (or "deques"). The dynamic part

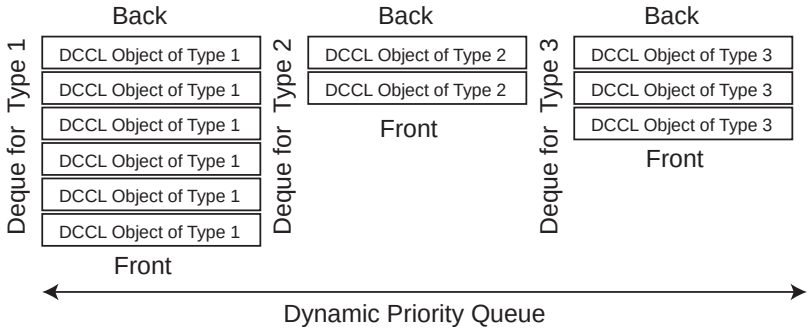


Figure 2.5: Data structure of *queue* for three declared DCCL types. All objects within a deque are of the same DCCL type and each deque is dynamically prioritized using Equation 2.1. Whether a deque is accessed from the back or front is configurable for each DCCL type.

is how *queue* differs from ordinary priority queues. Rather than having a fixed priority, entries in *queue* have a priority that varies in time. The structure of this dynamic priority queue of deques is outlined through an example in Fig. 2.5.

2.3.3 Implementation

Each buffer (one buffer for each DCCL type[1]) is given a base value (V_{base}) and a time-to-live (ttl) that create the priority ($P(t)$) at any given time (t):

$$P(t) = V_{base}\frac{(t - t_{last})}{ttl} \tag{2.1}$$

where t_{last} is the time of the last time an object was sent from this buffer.

This means for every buffer, the user has control over two variables (V_{base} and ttl). V_{base} is intended to capture how important the message type is in general. Higher base values mean the message is of higher importance. However, with only the V_{base}, higher value messages would always supercede lower value messages. AUV operators know, however, that messages of some types become more valuable if one has not been received in a long period of time, where "long" is defined by the preferences of the operators and the goals of the mission. For example, the value of a vehicle's status message grows in time as the operators become increasingly concerned with the health and location of the vehicle. The ttl parameter works to incorporate this notion of time varying value.

As the name suggests, the ttl governs the number of seconds the message lives from creation until it is destroyed by *queue*. But more importantly, the ttl also factors into the

[1] as uniquely defined by (dccl.msg).id

Table 2.4: Example DCCL types for *queue*

type id	V_{base}	ttl	example
1	1	1000	position report
2	2	2000	event message
3	1	3000	sensor (CTD) message

priority calculation. More time sensitive messages (those with lower ttl values) grow in priority faster. The reason that the ttl parameter is chosen to be the same as the priority time constant is simplicity. Managing complexity of configuration is one of the largest challenges facing field robotics. By grouping these two concepts together, only a single value (with a concrete meaning) needs to be configured for a given message queue's time sensitivity. When a vehicle can have dozens of message types (and communications is one of dozens of subsystems), reducing configuration needs reduces complexity.

So with these two parameters, the user can capture both overall value (i.e. V_{base}) and latency tolerance (ttl) of the message buffer. An example of how queuing manifests itself for different spacing of the Medium Access Control cycles is given in Fig. 2.6.

Another way to think of this dynamic priority buffering is in analogy to the economics of supply and demand. DCCL messages are analogous to perishable goods (such as food). The message sender has certain supply of each type of message. The receiver demands messages at a fixed price based on the type of good (V_{base}) that grows as time passes since the last "shipment" (successful transmission). The "perishability" of goods is reflected in the ttl. The sender uses Equation 2.1 to maximize his "profit" (assuming linear[2] laws).

2.3.4 Performance evaluation

In order to evaluate the performance of *queue*, a marketplace for data types was established where the receiver's demand for data is governed by three parameters: a base demand D_b, a growing demand after some time without data of that type (D_s, for **s**carcity), and a decreasing demand based on the age of the data (D_e for value **e**rosion). This hypothetical marketplace is intended to simulate the needs of an AUV topside operator or collaborating autonomous vehicle. The "score" (S^i) upon receipt of a given DCCL message i of type j is then given by

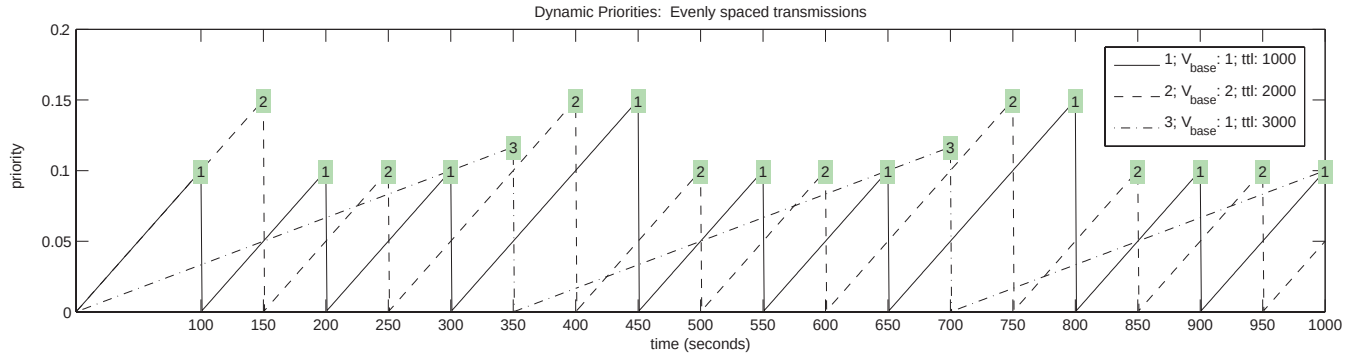
$$S^i = D_b^j + D_s^j - D_e^i \tag{2.2}$$

[2]other functional forms (e.g. exponential) were tested but the author could not find any substantial benefit versus linear, so linear was chosen for simplicity.

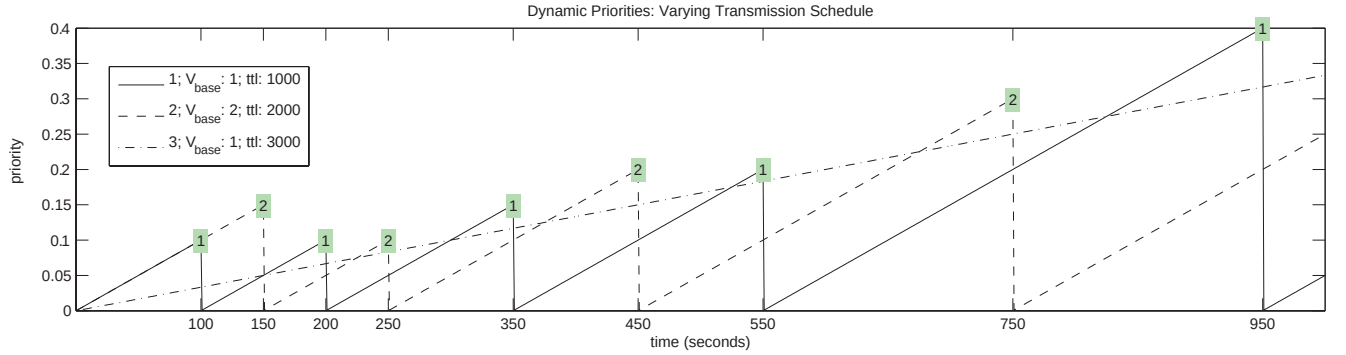
Table 2.5: Configuration of the Goby Dynamic Priority Queues

Message configuration		
Name	Type	Explanation
ack	bool	Acknowledgment required for this message type.
blackout_time	uint32	Absolute minimum time (seconds) between transmissions of this type.
max_queue	uint32	Size of the queue (in messages).
newest_first	bool	true=first-in-last-out (FILO), false=first-in-first-out (FIFO).
ttl	int32	time-to-live in seconds (also governs time sensitivity).
value_base	double	Base value of this message type.
Field roles [a]		
Role Name	Extension Type	Explanation
TIMESTAMP	double or uint64	Tags this DCCL field as source of time.
SOURCE_ID	(u)intN	Tags this DCCL field as containing the source address.
DESTINATION_ID	(u)intN	Tags this DCCL field as containing the destination address.

[a] Allows the message designer to designate any field in the DCCL message with the appropriate type as a given role in the queuing and addressing of this message. This is in lieu of a traditional header (such as IP uses) and thus allows the DCCL "header" to conform to the specifics of the network: see Fig. 2.4.



(a) TDMA Cycles are evenly spaced (period of 100 seconds). Types **1** and **2** alternate until **3** grows enough (long time since last t_{last})



(b) TDMA Cycles are unevenly spaced (increasing period of 50, 100, and then 200 seconds). **3** never gets to send in this scenario.

Figure 2.6: Comparison of message priority selection for the three different example types given in Table 2.4 using *queue*. Types **1** and **2** are equally valuable (since **1** is more time sensitive with its lower ttl and **2** is more valuable overall with a higher V_{base}). **3** is the least valuable. While clearly dependent on the spacing of transmissions, *queue* ensures a mix of all types of messages are sent, weighting the valuable ones more.

Table 2.6: Data marketplace parameters used for performance evaluation

parameter	value
D_b	V_{base}
τ_e	ttl
τ_s	See Fig. 2.7 & Fig. 2.8

where the scarcity parameter is given as

$$D_s^j = \frac{t - t_{last}^j}{\tau_s^j} \tag{2.3}$$

and

$$D_e^i = \frac{t - t_{create}^i}{\tau_e^j} \tag{2.4}$$

The time constants τ_s^j and τ_e^j govern the rate of demand growth and rate of message depreciation, respectively. The value t is the current time, t_{last}^j is the last time a message was sent from queue j, and t_{create}^i is the creation time of the message i.

Assuming that each queue is empty with Bernoulli probability P_{empty}, and that all queues have fresh data, the total score for $N = 100$ messages sent with evenly spaced interval $\tau = 100s$ is shown in Fig. 2.7 for the *queue* algorithm and two comparison queuing schemes: a standard priority queue which always picks the queue with the highest V_{base} and a time-sensitive priority queue that picks the queue with the lowest ttl amongst all full queues. The *queue* algorithm performs best when the queues are full nearly all the time, as the need for this type of queuing decreases when the link is no longer over-capacity. However, in real operations, P_{empty} is near zero for many of the queues as the vehicle is generating far more data than could ever traverse an acoustic link.

As shown in Fig. 2.8, when the scarcity time constant τ_s is increased substantially (thus reducing the demand for different types of data based on the last time they were received), the standard V_{base} priority queue performs better in this market. However, for types of data such as vehicle status reports, event reports (target contact and track reports), and health reports, the value of these data relatively decreases the more that are received in a short time frame as the reports are essentially the same, suggesting the "market" for AUV data is better represented by the one used for the results in Fig. 2.7.

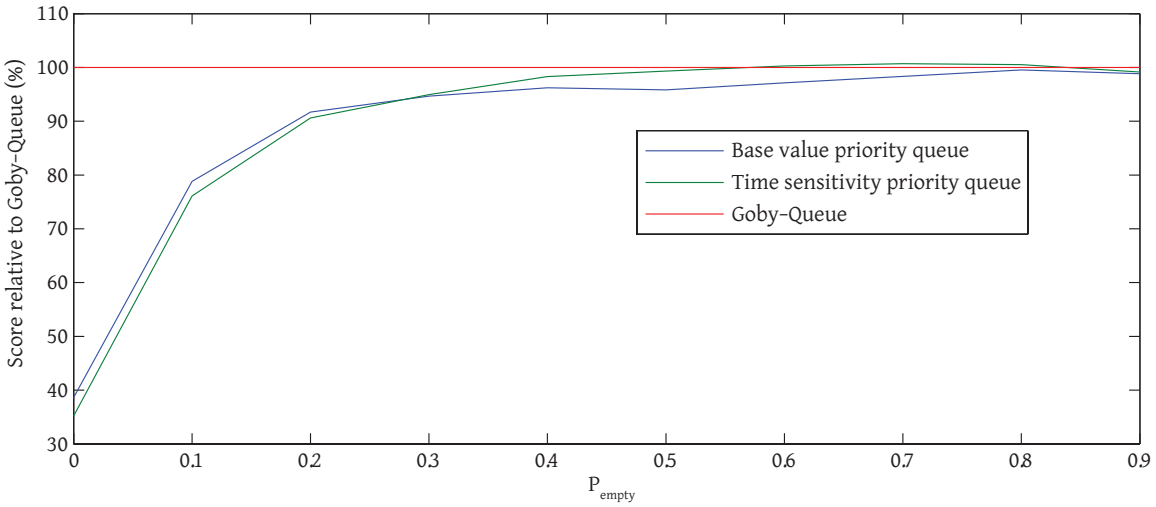


Figure 2.7: Relative performance of queuing techniques where $\tau_s = \tau/10$ and the other marketplace parameters are given in Table 2.6 with the queues from Table 2.4. In this case, the rate of growth in demand for a given type is an order of magnitude greater than the rate messages are sent (period τ). Thus, messages of a type that hasn't be received recently are significantly favored over types that have recently been received. In this case, especially when the network is over-capacity ($P_{empty} \simeq 0$), the Goby *queue* technique outperforms the two variants of a standard priority queue.

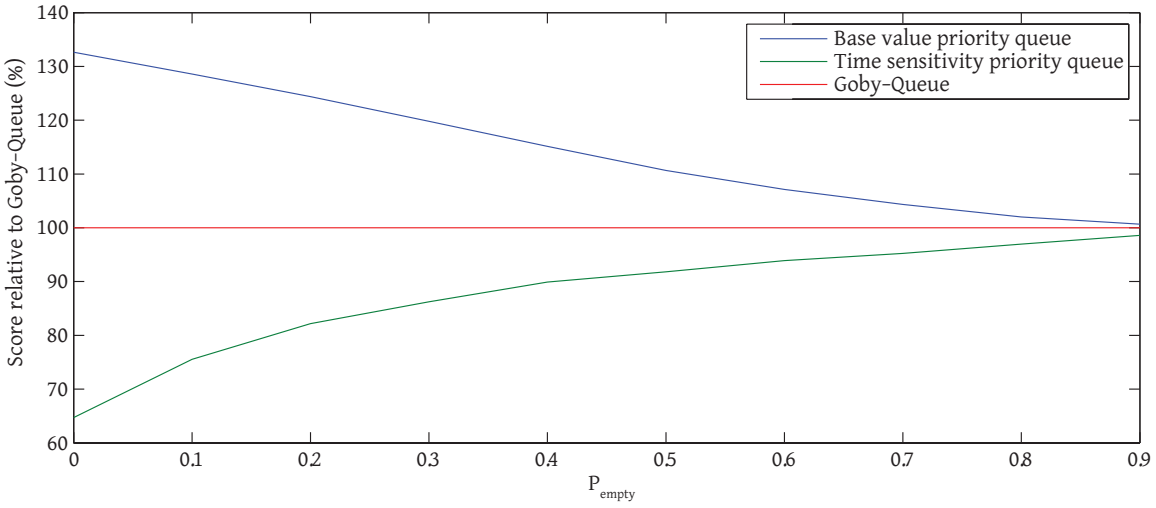


Figure 2.8: Relative performance of queuing techniques where $\tau_s = 10\tau$ (and other parameters as in Table 2.6). That is, the growth of demand for a given type is slow relative to the rate ($1/\tau$) at which messages are sent. In this case, a regular priority queue outperforms the *queue* algorithm.

2.3.5 Message stitching

While goby-acomms does not provide splitting and subsequent restitching of hardware layer frames to allow transmission of large DCCL messages[3], it does provide the opposite. *queue* will stitch small DCCL messages together to form a larger hardware layer frame. For example, goby-acomms *will not* break a 256 byte DCCL message into parts to fit a 32 byte WHOI Micro-Modem frame, but it *will* stitch 2 16 byte DCCL messages to fit a 32 byte WHOI Micro-Modem frame. This stitching is simply concatenation which works since the DCCL field decoders must consume exactly the same number of bits that the encoder produced.

The reasoning behind this is acoustic telemetry is so error prone that each received hardware frame should be useful in its own right. The size of hardware frames are chosen as a decent compromise between size and acceptable frame error rates. Hence, we feel providing abstraction of multiple frames per DCCL message would lead to unacceptable error rates (or unacceptable delays waiting for successful receipt and acknowledgement). However, providing facilities to fully utilize the entire hardware frame when the DCCL messages are small is useful and efficient as it maximizes the ratio of data to overhead (various headers and physical layer synchronization).

2.4 *amac*: Medium Access Control

2.4.1 Motivation

The goby-acomms acoustic Medium Access Control module is intended to provide a robust and easily usable MAC layer. MAC is perhaps the most widely studied question in acoustic networking; Partan does a good job summarizing the various options [15]. *amac* focuses on providing collision free communications with acceptable utilization of the available bandwidth under the following assumptions:

- All nodes are within broadcast range of one another for much of the time. (i.e., there are no hidden nodes).
- The hardware layer can support time division multiple access (TDMA).

These assumptions may seem rather strong, but in our experience they are practical for numerous present day AUV applications. goby-acomms supports two variants of the TDMA MAC scheme: centralized and decentralized. As the names suggest, Centralized

[3] for example, TCP provides such a message splitting feature

TDMA involves control of the entire cycle from a single master node, whereas each node's respective slot is controlled by that node in Decentralized TDMA.

2.4.2 Centralized TDMA (Polling)

Centralized TDMA involves a master node (usually aboard the Research Vessel or on land) which initiates every transmission for the entire communcations cycle (i.e. "polls" each node for data). Thus, the other nodes are not required to maintain synchronized clocks as the timing is all performed on the master node.

This style of MAC has been widely used for small AUV operations using the WHOI Micro-Modem. Its principal advantages are that it has 1) no requirement for synchronized clocks, 2) full control over the communications cycle at runtime (assuming the master is accessible to the vehicle operators, as is usually the case); and 3) a master who can acknowledge "broadcast" messages.

However, centralized TDMA has a number of substantial disadvantages. In order for a third-party master to initiate a transmission, an acoustic packet must be sent for this initialization. This additional "cycle initialization" packet, like any acoustic message, has a high chance of being lost (after which the data are never sent because the sending node did not receive a cycle initialization message), consumes power, and lengthens the time of the communications slot. See Fig. 2.9 for the various parts of the communication cycle with (for Centralized TDMA) and without (for Decentralized TDMA) the cycle initialization message. The additional time required for each slot of Centralized TDMA is

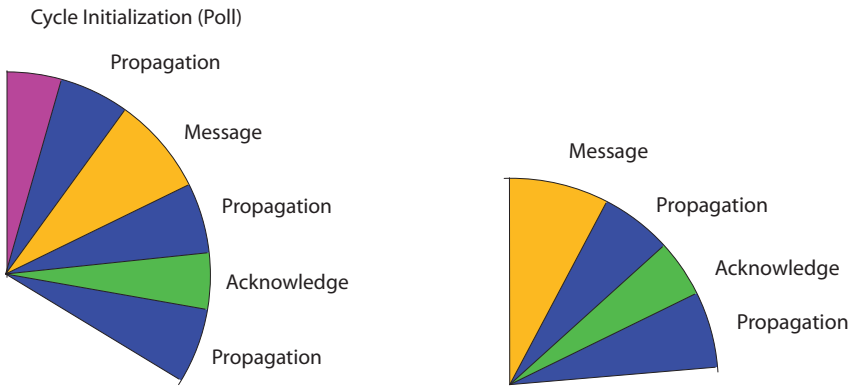
$$\tau_{ci} + r_{max}/c \tag{2.5}$$

where τ_{ci} is the length (in seconds) of the cycle initalization packet (about one second for the WHOI Micro-Modem), r_{max} is the maximum range of the network (typically of order 1000s of meters), and c is the compressional speed of sound (nominally 1500 m/s).

2.4.3 Decentralized TDMA with passive auto-discovery

Decentralized TDMA removes the cycle initialization packet and thus reduces the length of each slot and the chance of errors. However, it introduces the constraint of synchronized clocks[4] for all nodes, which can be somewhat tricky to maintain underwater. See

[4] the accuracy of the clock synchronization can be low relative to other timing needs such as bi-static sonar. Generally, accuracy better than 0.1 seconds is acceptable; higher inaccuracies can be handled by increasing the guard time on both sides of each slot.



(a) Centralized TDMA (b) Decentralized TDMA

Figure 2.9: Comparison of the time needed for a single slot for the two types of TDMA supported by `goby-acomms` *amac*. Eq. 2.5 gives the additional length of time required by the Centralized variant.

Eustice et al. [46] for an example of maintaining synchronization for navigation and communication.

Decentralized TDMA gives each vehicle a single slot in which it transmits. Each vehicle initiates its own transmission at the start of its slot. Collisions are avoided by each vehicle following the same rules about slot placement within the time window (based on the time of day). All slots are ordered by ascending acoustic MAC address (or "modem identification number"), which is an unsigned integer unique for each network.

During the runtime of the network, it is often desirable to add or remove nodes. Since the MAC is spread throughout the nodes, there is no easy way to change the cycle during runtime. *amac* supports passive auto-discovery (and subsequent expiration) of nodes to provide a solution to this problem. This auto-discovery is passive because it requires no control messaging beyond the normal communications between nodes.

Vehicles are discovered by shifting a blank slot in each cycle based on their knowledge of the world and the time of day. If a new vehicle is heard from during the blank, it is added to the listening vehicle's knowledge of the world and hence their cycle. In the simplified situation (which is really a worst case scenario) discovery is defined by a single vehicle transmitting during a cycle and all the others silent (the current slot is not equal to each vehicle's acoustic MAC address).

The auto-discovery works in a manner analogous to excitation of electrons in an atom. The blank slot is inserted somewhere in the middle of the cycle ("ground state"). The ground state is moved around pseudorandomly (but in the same place for a given known

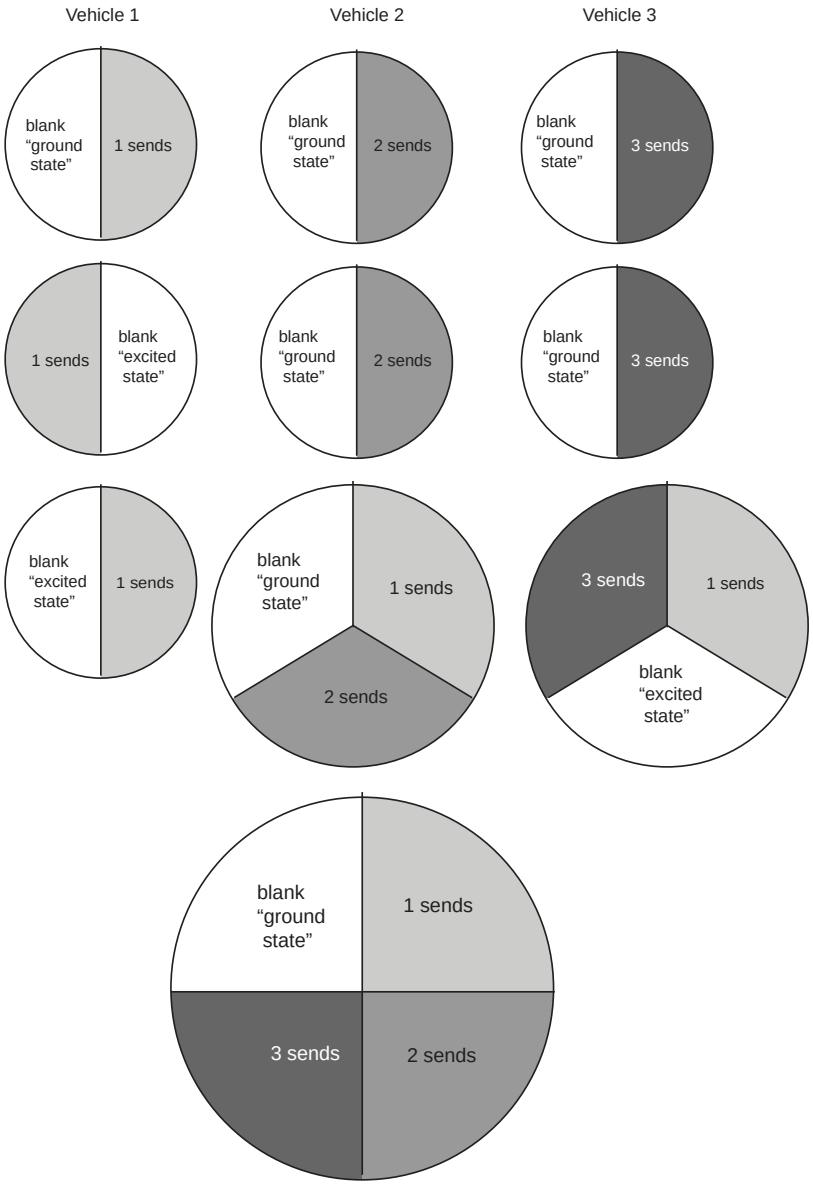


Figure 2.10: Graphical example of auto discovery for three nodes launched at the same time. Each circle represents the vehicle's cycle at each time step (represented by horizontal rows) based on the vehicle's current knowledge of the world. In the first row, all vehicles only know of themselves and put the blank slot in the last slot; thus, all communications collide and no discoveries are made. In the second row, vehicle 1's blank is moved (by pseudo-chance, see equation 2.6) to the penultimate (first) slot, so vehicles 2 and 3 discover 1. Then, in the third row vehicles 2 and 3 are discovered by the others because vehicle 3 moves its blank slot. By the fourth row all vehicles have discovered the others and continue to transmit without collision following the cycle diagrammed on this row.

time	vehicle 1	vehicle 2	result
0	send	send	collision
15	blank	blank	nothing
30	blank	send	success: 1 discovers 2
45	cycle wait	blank	nothing
60	cycle wait	send	success
75	cycle wait	blank	nothing
90	send	blank	success: 2 discovers 1
105	listen for 2	cycle wait	nothing
120	blank	cycle wait	nothing
135	send	listen for 1	success
150	listen for 2	send	success
165	blank	blank	nothing
180	send	listen for 1	success
195	blank	blank	nothing
210	listen for 2	send	success

Table 2.7: Example initialization for the Decentralized TDMA with autodiscovery. By 135 seconds, both vehicles have discovered each other and are synchronized. Thus, no more collisions will occur. This scenario assumes that both vehicles always have some data to send during their slot.

"world" of vehicles) so that any possible MAC address in the cycle will be eventually discovered. However, depending on the "temperature" (determined by the *coolness* parameter C) the blank slot may be "excited" and moved to the end (to the ultimate slot). How often this parameter is moved is pseudorandomly determined from the time of day and the current known world state (as evidenced by the sum of the acoustic MAC addresses of all known nodes). The higher the coolness parameter C, the less likely the blank slot will be "excited" from its normal position to the end of the cycle. Assuming all collisions are destructive to the data received, no vehicles would ever be discovered without this movement of the blank slot. By moving the blank slot, we improve the chances that two vehicles with dissimilar views of the world will eventually discover each other. Mathematically, this placement of the blank spot in the cycle can be expressed as

$$i_{blank} = \begin{cases} i_{max} & \text{if } \lfloor t_{UTC} / \sum_i \tau(i) \rfloor \pmod{C} = \sum_i a(i) \pmod{C} \\ i_{base} & \text{otherwise} \end{cases} \tag{2.6}$$

$$i_{base} = i_{max} - \lfloor t_{UTC} / \sum_i \tau(i) \rfloor \pmod{i_{max}} - 1 \tag{2.7}$$

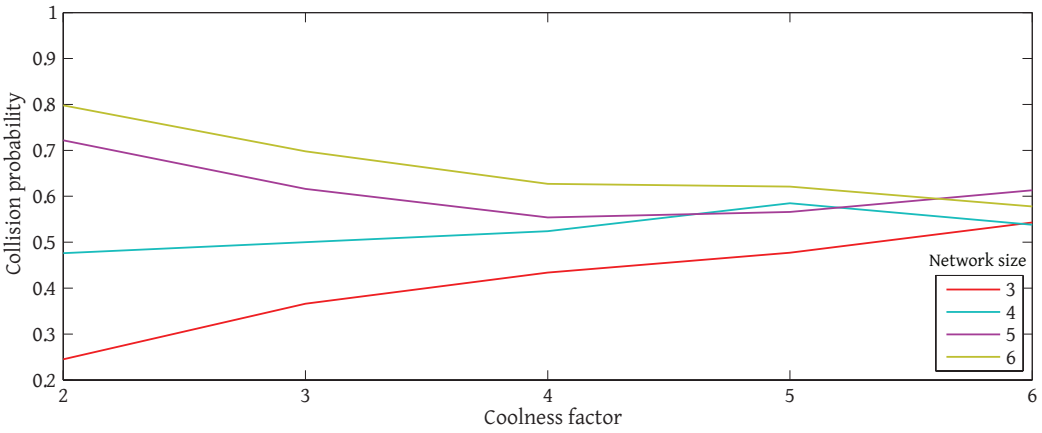
where i_{blank} is the position of the blank slot in the cycle, t_{UTC} is the number of seconds since midnight Coordinated Universal Time (UTC) of the current day, $\tau(i)$ is the length of the ith slot, C is the "coolness" parameter, and $a(i)$ is the acoustic MAC address of the node in the ith slot. Put in words, the blank slot is moved from its normal place in the

cycle (position i_{base}) to the excited position (i_{max}) when the number of cycles since the start of the day is congruent modulo C with the known world (sum of MAC addresses). Therefore, the higher C is, the less often these two values are congruent, and the less often the blank slot is "excited". The nominal (base) position of the empty slot i_{base} is rotated through the non-end positions of the cycle so that the non-excited vehicle cycles can discover all the different excited vehicle MAC addresses over time.

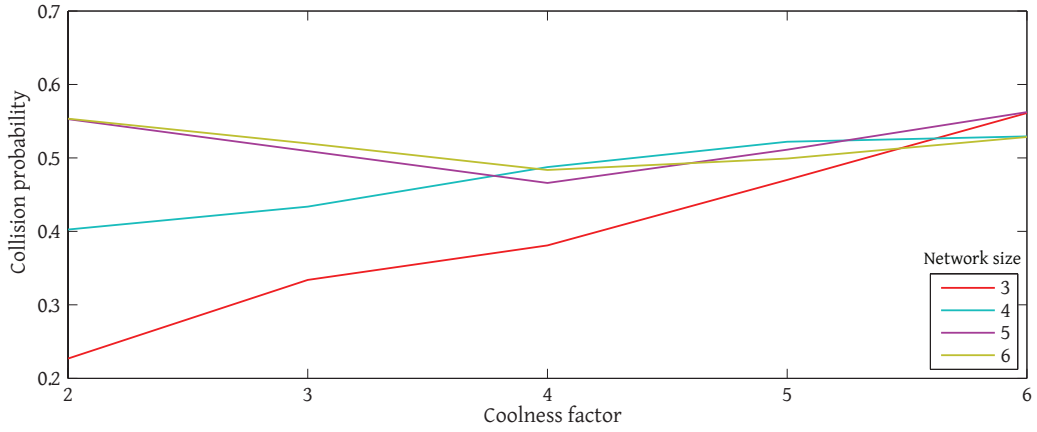
2.4.4 Performance evaluation

The meaning of the coolness parameter may be qualitatively intuitive, but it is worth quantitatively examining its effect on the performance of *amac*. First, a Monte Carlo simulation of randomly chosen networks (from three to six vehicles in size) drawn from the space of all seven bit MAC addresses was performed and the results plotted in Fig. 2.11. The collision probability represents the likelihood of collision during the *amac* learning phase (after which all vehicles have the same TDMA cycle and the chance of collision goes to zero). Fig. 2.11a shows the collision probability for these varying network sizes as a function of the coolness factor C for a network where all theoretical collisions actually occur ($p_{collide} = 1$). Real networks will not always have collisions as transmissions may be lost due to range, vehicles may choose not to transmit, or a sufficient amount of header data to perform discovery may survive the collision due to asymmetric travel times. Thus, in Figs. 2.11b and 2.11c, the same metric was plotted but for $p_{collide} = 0.75$ and $p_{collide} = 0.5$, respectively. When the probability of collision goes down, this reduces the optimal coolness parameter value for a given expected network size.

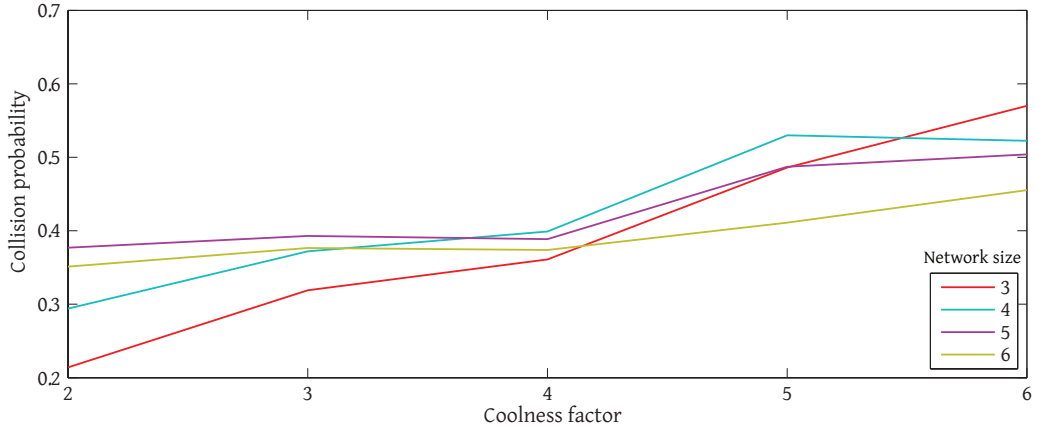
Next, a full simulation of the discovery process was created, using $p_{collide} = 1$ and a number of vehicles all entering the network simultaneously. This is the worst case scenario, as all the vehicles have different TDMA cycle states at once. All permutations of a network containing five MAC addresses were simulated. Again, a variety of small networks were examined for a range of coolness factors. The line graph in Fig. 2.12 shows the mean discovery duration (the time to reach a zero collision state) in multiples of the slot duration ($\tau = \tau(i)$ for all i) when such a discovery succeeded. The bar graph shows the percentage of permutations that failed for a given coolness factor. Thus, lower coolness factors generally reach a solution faster, if such a solution can be reached. Higher coolness factors lead to slower network discovery, but with increasing chance of success. Thus, a reasonable engineering value for C would be one more than the expected maximum size of the network.



(a) $p_{collide} = 1$



(b) $p_{collide} = 0.75$



(c) $p_{collide} = 0.5$

Figure 2.11: These plots show the mean collision probability for a given cycle during the discovery (learning) phase of *amac*, for a range of probabilities of a given transmission colliding ($p_{collide}$).

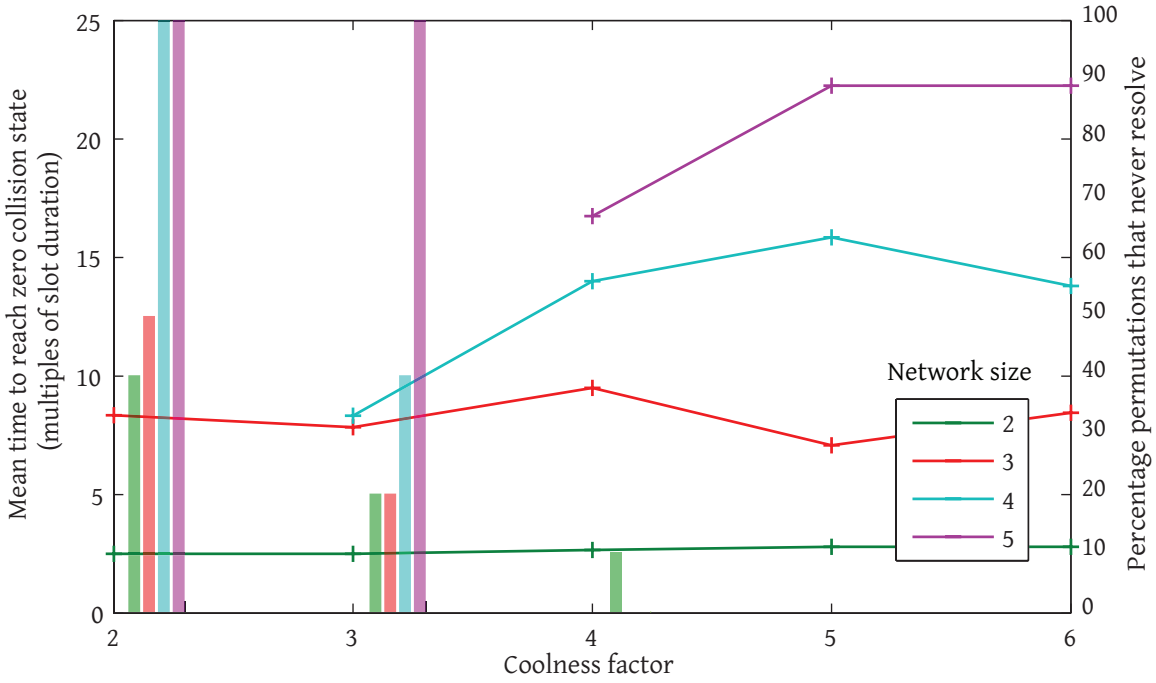


Figure 2.12: Full simulation of network discovery time for *amac* using passive autodiscovery. Line graph: mean discovery time in multiples of τ. Bar graph: percentage of MAC address permutations that fail to ever reach full discovery amongst all nodes.

2.5 *modemdriver*: Acoustic modem driver

2.5.1 Motivation

In `goby-acomms`, the physical layer is generally assumed to be an acoustic modem, as the tradeoffs made between efficiency and abstraction are intentionally highly biased towards efficiency in `goby-acomms` because of the very low throughput acoustic channel. However, the remainder of `goby-acomms` is agnostic to the choice of acoustic modem, or even that the physical layer is acoustic at all; other very low throughput channels (e.g. satellite) also work with the design paradigms of `goby-acomms`. *modemdriver* is responsible for communicating with the specific firmware of the acoustic modem of choice and abstracting that interface for the rest of `goby-acomms`.

No standard exists for the interfaces to acoustic modems, and it is unlikely one will arise in the near future. Even with a common interface, modems will likely continue to have useful special features (such as navigation and ranging functionality) that exist outside realm of the core functionality of a modem, which is to send data. Thus, in order to

make use of the rest of Goby (especially DCCL) on a variety of AUVs using different hardware, the link layer interface (ModemDriver) was written with a standard object-oriented design: a base class that provides an interface to the higher layers of Goby, and an increasing suite of derived classes to implement this interface for a specific piece of hardware. A beneficial side effect of this design is the ability to write drivers for a variety of non-acoustic links that have similar characteristics to acoustic links (low throughput), such as satellite communications or a faster-than-realtime underwater autonomy simulator, such as the MOOS-IvP "uField Toolbox" from [47].

The key to the success of this design is simplicity. A single function call initiates a transmission using the ModemTransmission class, which is typically a data telegram (`type == DATA`). Symmetrically, a single signal (again containing an instantiation of the ModemTransmission) is emitted upon asynchronous receipt of data. However, the ModemTransmission can be extended (even outside the Goby project) to support any number of additional data structures relating to an alternative type of transmission (e.g. long baseline (LBL) pings such as the WHOI Micro-Modem's `type == NARROWBAND_LBL`). If the application is aware that is using a specific piece of hardware, all these extensions become visible. If not, the application can still use the core functionality (data transmission). This design allows the system designer to choose abstraction when desirable or have full control of a modem's functionality as needed. See Fig. 2.13 for a diagram illustrating the core and extended functionality for several acoustic and other "slow" links.

This simplicity makes the task of writing a new driver easier. Rather than deal with a plethora of functionality (the superset of all supported modems would have been an alternative design), the new driver author must only deal with sending and receiving data. Once that is finalized, new functionality can be added as needed.

2.5.2 *DriverBase*: Abstract Acoustic Modem Driver

DriverBase provides an virtual interface to a generic modem (or any system that can transmit datagrams). The only requirement is that the modem supports transmission of fixed size datagrams with optional acknowledgment of the message receipt (variable size datagrams can be used to mimic fixed sizes). If the modem does not provide acknowledgment, this functionality must be built into the extension of DriverBase. The requirement is visualized by the `DATA` and `ACK` transmission types shown in Fig. 2.13.

DriverBase provides:

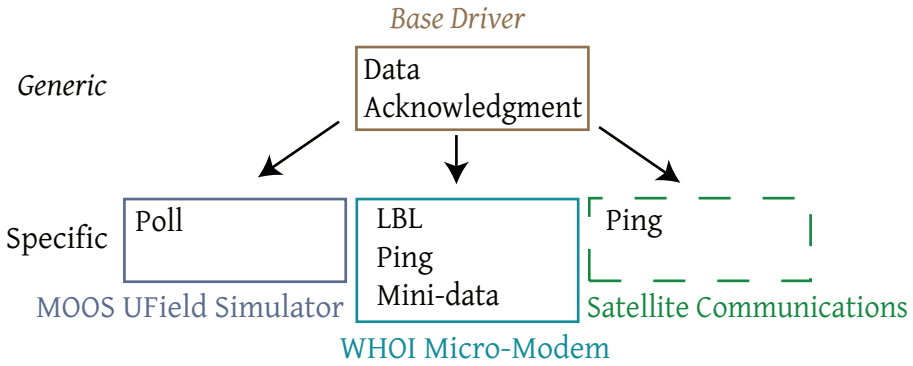


Figure 2.13: Diagram of base generic ModemDriver functionality versus extended modem-specific features for an acoustic modem (the WHOI Micro-Modem), a satellite link, and a faster-than-realtime virtual "modem"(the uField simulator). The application using ModemDriver can operate on either the generic level (if it only needs data and acknowledgment functionality) without knowing the details of the given hardware or on the specific level to take advantage of special features of a given modem.

- a class that reads serial port or TCP data into a buffer for use by the DriverBase derived class.
- methods to set all six callbacks provided by the derived class (receive, transmit result, data request, raw incoming message, raw outgoing message). Typically *queue* handles the receive, and data request callbacks. The raw messaging callbacks are optionally provided for the application layer to perform debugging directly on the modem, if desired.
- four virtual functions: for starting, stopping, and running the driver, and for initiating the transmission of a message.

2.5.3 *MMDriver*: WHOI Micro-Modem Driver

The MMDriver extends the DriverBase for the WHOI Micro-Modem acoustic modem. The WHOI Micro-Modem uses a serial RS-232 interface and an NMEA-0183 sentence structure.

The following features of the WHOI Micro-Modem are implemented, which comprise the majority of the Micro-Modem functionality:

- Frequency Hopping Frequency Shift Keying (FH-FSK) (rate 0) data transmission (`type == DATA`, `type == ACK`). See Fig. 2.14 for a sequence diagram of the Goby *MMDriver* using the Micro-Modem for this transmission type.

- Phase Shift Keying (PSK) (rates 1,2,3,4,5) data transmission (`type == DATA`, `type == ACK`).
- Narrowband transponder LBL ping (`type == MICROMODEM_NARROWBAND_LBL_RANGING`).
- REMUS transponder LBL ping (`type == MICROMODEM_REMUS_LBL_RANGING`).
- User mini-packet 13 bit data transmission (`type == MICROMODEM_MINI_DATA`).
- Two way ping (`type == MICROMODEM_TWO_WAY_PING`). Fig. 2.15 illustrates this transmission type.

Prior Work: iMicroModem

Given the number of users of the WHOI Micro-Modem, we believe that a number of special purpose "ad-hoc" Micro-Modem drivers have been written, but the details of which are not reported in the literature. Grund's iMicroModem is one of these that we have had a chance to work with since it was the Micro-Modem driver used with the MOOS automony architecture that preceeded `goby-acomms`. The MOOS is discussed in section 2.6.1.

MMDriver borrows a number of ideas from iMicroModem in terms of dealing with the specifics of the WHOI Micro-Modem firmware. The major difference is that MMDriver implements the interface provided by *DriverBase* and thus does not have to replicate any of the work done by *DriverBase*. This makes MMDriver shorter and simpler as it only contains details specific to the WHOI Micro-Modem. *DriverBase* handles the communication (RS-232 serial in this case), logging, and interface to the rest of `goby-acomms`.

This is standard object-oriented design, but it is mentioned here since such a design is critical for supporting multiple types of acoustic modems. Given that no standard exists for underwater acoustic telemetry, it appears the need to support various types of hardware through an abstracted generic interface will exist for some time. *MMDriver* with *DriverBase* provides that for the WHOI Micro-Modem.

2.6 Goby1 Field Case Studies

2.6.1 MOOS-IvP Autonomy system and middleware

`goby-acomms` was developed with and tested with the MOOS-IvP autonomy architecture [27]. MOOS-IvP is used on marine robots for autonomy level control. We use MOOS-

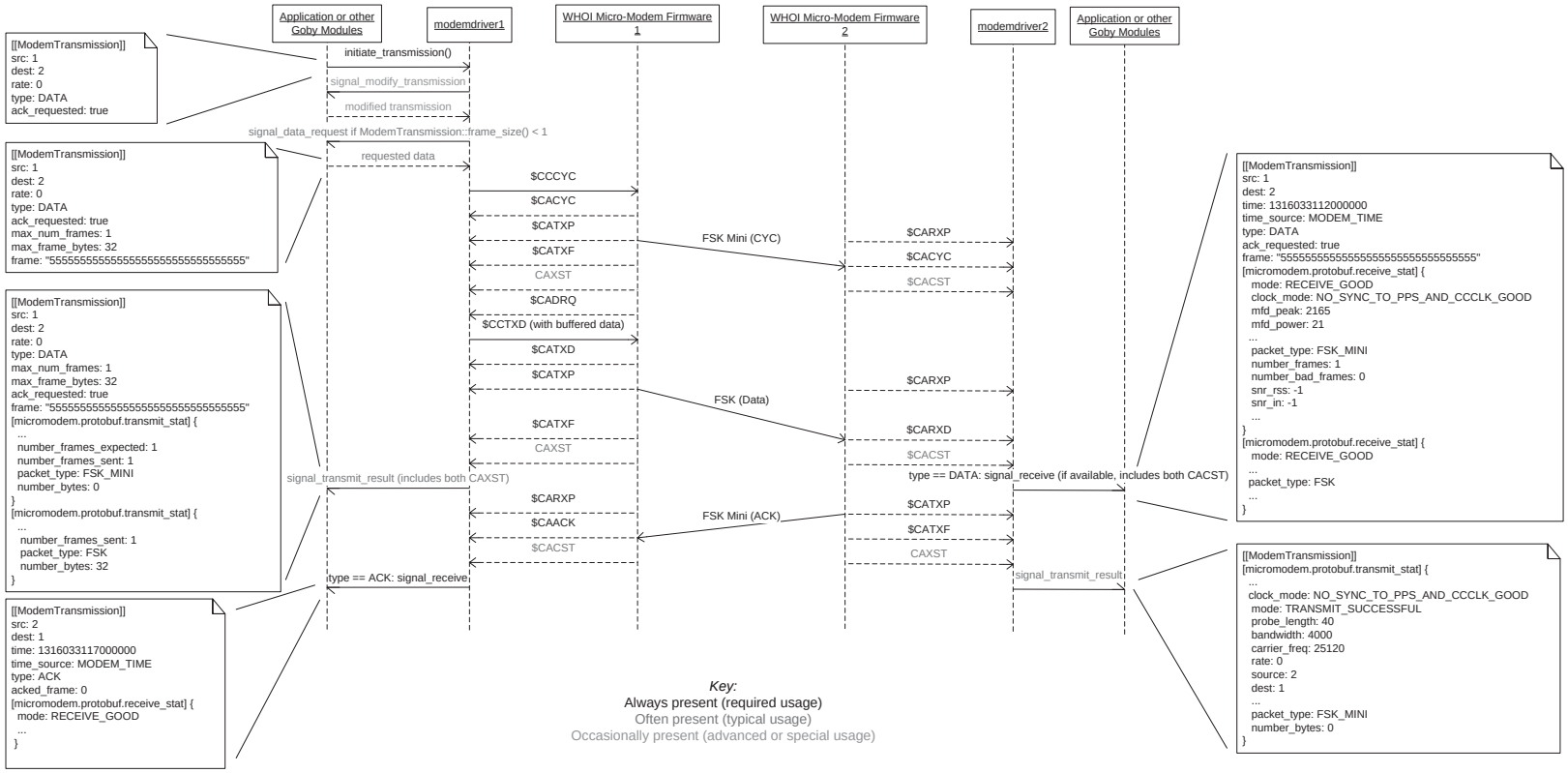


Figure 2.14: UML sequence diagram of the process of sending a unicast message with acknowledgment using Goby over the WHOI Micro-Modem at FH-FSK rate 0. The Goby driver abstracts numerous Micro-Modem NMEA-0183 messages into a single extensible transmission object (`goby::acomms::protobuf::ModemTransmission`). The example messages on the left and right would be produced and consumed respectively by the Goby *Queue* module or a suitable replacement.

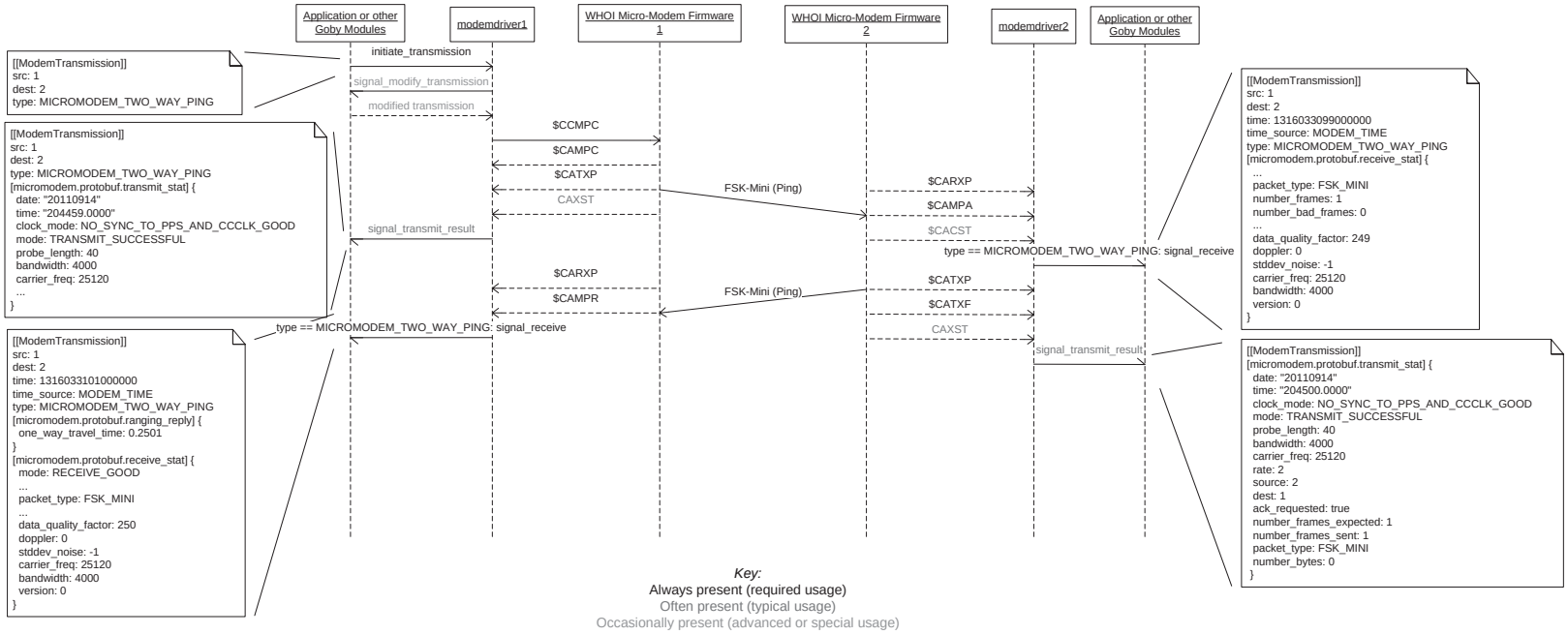


Figure 2.15: Sequence diagram for a two-way time-of-flight ("ping") measurement using the Goby `MMDriver` and the WHOI Micro-Modem. Goby can support such a special feature (outside the realm of data transmission) without requiring that all modems have it.

IvP in an abstracted manner such as that different vehicle types from different manufacturers appear the same to the autonomy system and communications network. This model of operations is called Unified Command and Control. The design of Unified Command Control as well as further details of all these trials except GLINT10 can be found in [48], as well as in Appendix A. The results presented here are using `goby-acomms` via pAcommsHandler, an interface process between MOOS-IvP and `goby-acomms`. The experiments referenced are summarized in Table 2.8. Since each experiment has different assets, different environmental conditions, and different objectives, it is difficult to make clear comparisons in performance from one sea trial to another. Thus, what follows is a series of case studies highlighting the development and testing of `goby-acomms`. For successful sea trials with AUVs, two goals are perhaps the most important: saving experimenters' time and improving safety of the vehicles. Any improvement in operations that touches upon these goals improves the productiveness of the experiment. These are often hard to quantify, but these case studies try to emphasize what `goby-acomms` has done on both of these fronts.

2.6.2 GLINT08

The three GLINT experiments (2008-2010) were designed to develop and test systems for multi-static active tracking of moving targets. For the 2008 experiment the first part of `goby-acomms`, the code which later became *queue*, was developed. We were using three CCL messages to communicate three types of data from the AUV: status (position and speed of the vehicle), contact (possible detection), and track (fused contacts) reports. The status reports were always generated so that we could monitor the health and activity of the vehicles. When an acoustic source was detected, contact reports were generated by the signal processing and then track reports from the tracker. At this point using a basic priority queue (before *queue*) we would only receive the higher priority track reports and contact reports and no status reports. This was because the number of contact and track reports exceeded the available throughput of the acoustic channel. This was unacceptable because we knew where the targets were, but no longer received updates as to the position of our AUV. Thus, we needed a way for messages with a lower base value (such as the status message) to occasionally become more valuable than those with higher base values (such as the contact and track messages). To solve this problem, *queue*'s dynamic priority queues, as described in section 2.3, were created.

With *queue*, the messages received were proportional to the base value (time sensi-

Table 2.8: Summary of field trials.

Name	Summary	Assets (vehicles all have WHOI Micro-Modem)	Experiment Datum[a]
GLINT08	Interoperability of marine vehicles for passive acoustic target detection	1 Bluefin 21 AUV, 1 NURC OEX AUV, 1 OceanServer Iver2 AUV, 2 Robotic Marine Kayaks, 1 WHOI Comm Buoy, 2 Ship-deployed WHOI Micro-Modems	42.5°N, 10.08333°E
SWAMSI09	Detection and tracking of seabed objects using bistatic acoustics.	2 Bluefin 21 AUVs, 1 WHOI Comm Buoy	30.045°N, 85.726°W
GLINT09	Interoperability of marine vehicles for multi-static acoustic target target tracking	1 NURC OEX AUV, 1 OceanServer Iver2 AUV, 2 Robotic Marine Kayaks, 2 Ship-deployed WHOI Micro-Modems	42.47°N, 10.9°E
CHAMPLAIN09	Thermocline gradient following.	1 OceanServer Iver2 AUV, 1 Ship-deployed WHOI Micro-Modem.	42.2511°N, 73.3612°W
GLINT10	Interoperability of marine vehicles for passive and active acoustic target tracking. Collaborative acoustic communications and environmental sampling.	1 Bluefin 21 AUV, 1 WHOI Comm Buoy, 2 NURC OEX AUVs, 2 Ship-deployed WHOI Micro-Modems.	42.47°N, 10.9°E

[a] The experiment datum is a location in the southwest corner of the operation region from which all vehicle positions are referenced using the Universal Transverse Mercator projection with the WGS 84 ellipsoid [49].

tivity, via the time-to-live (ttl) was introduced later). During a tracking event, track and contact messages were highest priority, but status messages were still occasionally sent.

2.6.3 SWAMSI09

SWAMSI09 was another acoustic sensing experiment, this time for sea floor mine-like targets. CCL had no messages for reporting contacts for this type of target. For this reason and the others given in section 2.2.6, we determined that CCL was no longer sufficient for our needs and developed DCCL and the corresponding encoding library, *dccl*.

The ease of defining and redefining DCCL messages allows for rapid prototyping of new experimental ideas during the field trial, rather than being rigidly confined to previously defined messages. We wrote five new messages on the experiment to greatly expand the flexibility of vehicle to topside, and vehicle to vehicle communications capa-

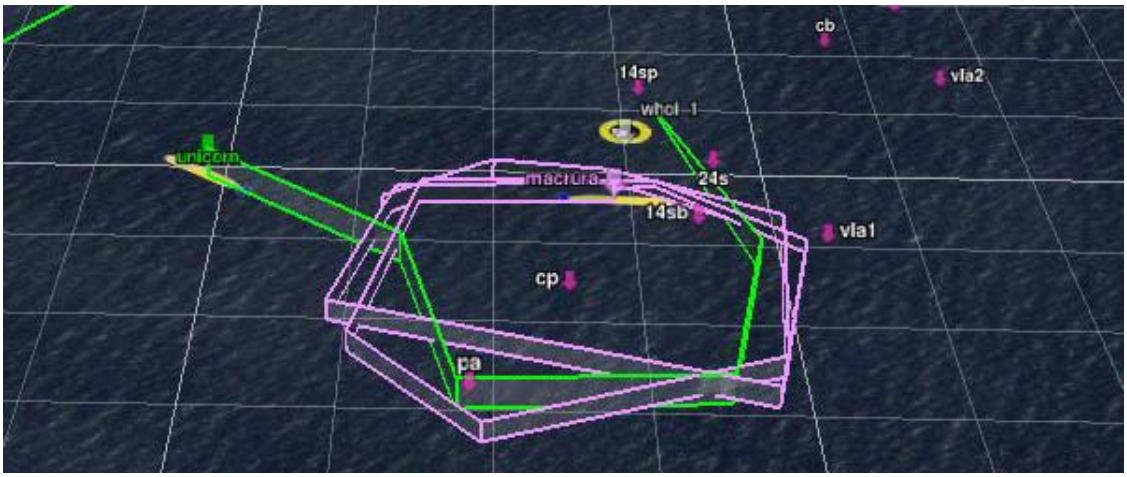
bility.

We used two AUVs to execute a variety of bistatic acoustic configurations for tracking of proud and buried seabed targets. Both AUVs traversed a circular pattern around the potential target, maintaining a constant bistatic angle (see Fig. 2.16a). Entering into this collaboration and maintaining the correct angle required handshaking and data transfer between both vehicles. We were able to command the vehicles into this collaborative state with `LAMSS_DEPLOY`, and the `LAMSS_STATUS` message (with additional fields added to support this experiment) was passed between vehicles to maintain the correct positioning autonomously.

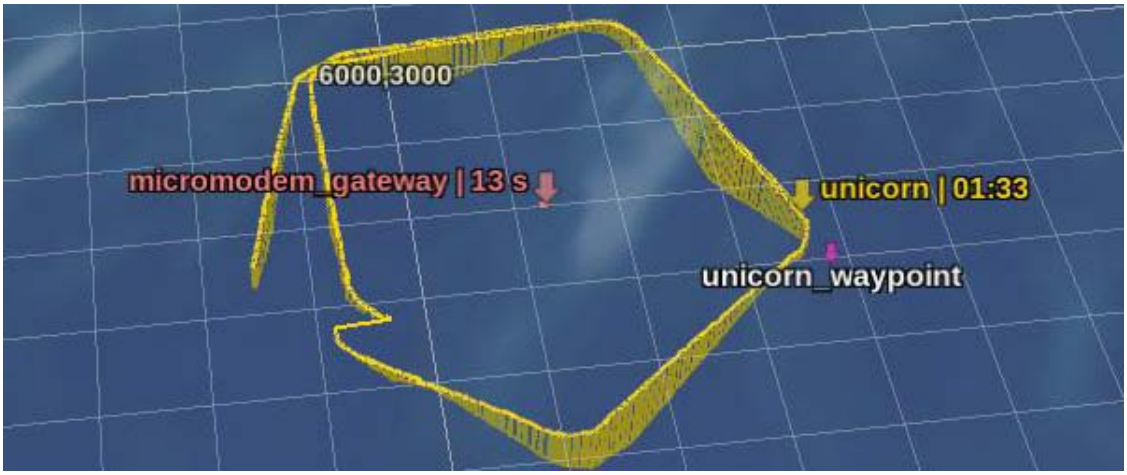
2.6.4 GLINT09

For the previous two experiments, we were using iMicroModem (section 2.5.3) as the driver for the WHOI Micro-Modem. Concerns about the robustness and extensibility of that software led to the development of *modemdriver*. While the features provided by *modemdriver* for the WHOI Micro-Modem did not vary much from iMicroModem, we saved significant time debugging.

Furthermore, we expanded our usage of *dccl*. DCCL messaging made another collaborative experiment possible. We had a mobile acoustic gateway (an autonomous surface craft with a WHOI Micro-Modem) available to stream high rate environmental and other data messages. By virtue of the surface craft staying near the AUV (made possible by the AUV's `LAMSS_STATUS` message), the AUV had a short acoustic propagation path to the surface craft. From there, the surface craft relayed data to the operators via IEEE 802.11 wireless ethernet. Also, the depth of the modem was controlled by a winch that the surface vehicle could command autonomously. Using the `WINCH_CONTROL` message, the AUV commanded the surface craft a depth at which to set the modem to improve communications. The AUV was performing a bistatic acoustic detection of a mid-water column depth target. The source, mounted on a buoy, was autonomously turned on and off by the AUV using the `SOURCE_ACTIVATION` message. The AUV, which was towing an acoustic array, was the receiver. None of this multi-robot collaboration would have been possible without the ability to define new messages quickly and with a high degree of confidence in their syntactical correctness provided by *dccl*.



(a) During SWAMSI09, the two AUVs "Macrura" and "Unicorn" perform a synchronous circular pattern with a constant angle of separation. However, due to the sporatic updates from the acoustic modem, it is hard to visualize the performance of the vehicles in executing this maneuver at runtime.



(b) A snapshot of the runtime visualization of the AUV "Unicorn" performing a sinusoidal depth excursion while performing a pentagon shape. While full, updates are delayed, the `LAMSS_STATUS_FILLIN` and `LAMSS_CTD` messages give a detailed history of the vehicle's track when the communication environment permits.

Figure 2.16: Comparison of the Google Earth interface for Ocean Vehicles (GEOV) [48] visualization available to the vehicle operator during runtime using data transmitted via `goby-acomms` early in its design at SWAMSI09 (a) and in the form `goby-acomms` is presented in this paper during GLINT10 (b). Vertical lines indicate acoustic position updates via the `LAMSS_STATUS` message and horizontal lines connect these updates.

Table 2.9: Formulas for delta difference encoding the DCCL <float> type.

DCCL Type	Encode[a]
<float> (*key*)	$x_{key} = \begin{cases} \text{nint}((x - x_{min}) \cdot 10^{prec}) + 1 & \text{if } x \in [x_{min}, xmax] \\ 0 & \text{otherwise} \end{cases}$
<float> (*delta*)	$x_{\Delta} = \begin{cases} \text{nint}((x + \Delta_{max} - x_{key}) \cdot 10^{prec}) + 1 & \text{if } x - x_{key} \in [-\Delta_{max}, \Delta_{max}] \\ 0 & \text{otherwise} \end{cases}$

· x_{min}, x_{max}, $prec$, Δ_{max} are the contents of the <min>, <max>, <precision>, and <max_delta> tags, respectively.
· nint(x) means round x to the nearest integer.
· The key value x_{key} is the same as the normal <float> type encoding given in Table 2.3.

2.6.5 CHAMPLAIN09

The third case study is the CHAMPLAIN09 adaptive environmental experiment. In this experiment, a small AUV outfitted with a Conductivity-Temperature-Depth (CTD) instrument was deployed to study the thermocline structure of Lake Champlain. The AUV was commanded, using a updated LAMSS_DEPLOY message, on the task of adaptively surveying the thermocline. The vehicle accomplished this task by performing series of sinusoidal ("yoyo") depth maneuvers and streamed its samples back using the delta-difference encoded LAMSS_CTD message. In this manner, the environmental data was made available in near realtime (i.e. delayed by no more than a few minutes) to the AUV operator.

The key feature used for this work and later in GLINT10 was Goby1's delta-differencing, originally applied only to CTD messages and later added as a general feature to *dccl*. Delta-difference encoding can be applied to <float> DCCL fields (and <int> since they are derived from <float>). It gives an even more compact way to losslessly encode correlated data. In this case, due to the AUVs finite speed and continuity of salinity and temperature values, CTD values are correlated in time. By estimating upper and lower bounds on this correlation, data can be compressed further than DCCL normally allows by sending the first sample in its entirety (still bounded by the usual DCCL <max> and <min> "global key") and sending the remaining samples in a frame by their difference to this first sample. The bound on the maximum that this difference can be (Δ_{max}) must be given in <max_delta> tag, using physical knowledge of the data to be sampled. See Table 2.9 for the corresponding formulas for the field size and encoded values. Diagrammatically, the process is explained in Fig. 2.17 and an example of the data available to the operator during runtime is shown in Fig. 2.18.

For example, perhaps it is known *a priori* by means of historical data or a ship CTD

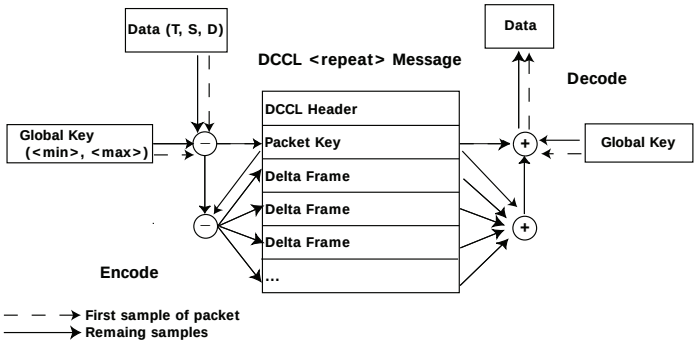


Figure 2.17: Schematic of encoding and decoding a DCCL message using delta-difference encoding. The DCCL header is diagrammed in detail in Fig. B.1.

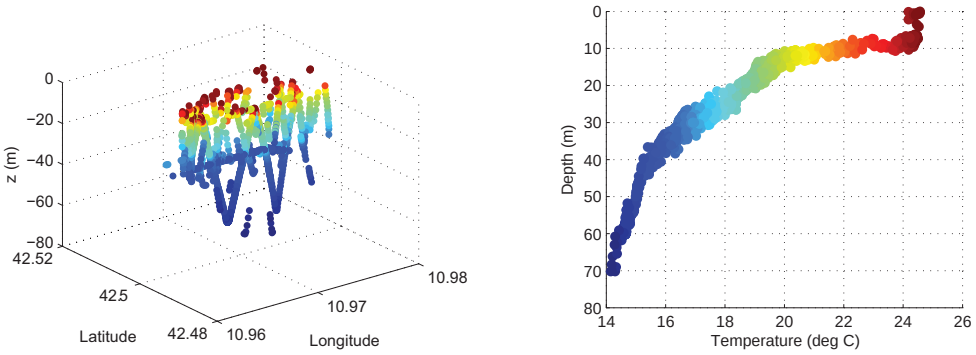
cast that the maximum temperature gradient in a given area is $0.05°C/m$ and the dive rate of our glider is $0.2m/s$. We also know that the water in the operation region does not exceed $(10, 30)°C$. Furthermore, we feel that tenths of a degree Celcius is sufficient precision. Finally, we want to sample the thermistor on our CTD at 1 Hz and we are using a 256 byte WHOI Micro-Modem frame. Putting this all together, we use for temperature (in $°C$) a DCCL <`float`> with a <`min`> of 10, a <`max`> of 30, a <`precision`> of 1. The <`max_delta`> must be calculated iteratively (such as using the Newton-Raphson method), as making a smaller <`max_delta`> creates a smaller message, increasing the number of samples that can be fit in a frame. This increases the window of sampling for a given frame, thus increasing the <`max_delta`>. That is, the optimum Δ_{max} for given bounds is the solution closest to equality to

$$l_{key} + l_{delta}(\Delta_{max}) \cdot (\Delta_{max}/r_{max} * f_s - 1) <= l_{frame} \tag{2.8}$$

where l_{frame} is the total message frame size, l_{key} and $l_{delta}(\Delta_{max})$ are the sizes for key and delta frames given in Table 2.9, r_{max} is the maximum expected rate of change of the physical parameter being encoded, and f_s is the sampling frequency. For this example $l = 2048$ bits, $r_{max} = 0.05°C/m \cdot 0.2m/s = 0.01°C/s$, $f_s = 1Hz$, and $l_{key} = 8$ bits, so solving for the smallest Δ_{max} (which provides the largest number of samples in the frame) is $3.1°C$, providing 310 samples per frame. This is a 21% improvement over the 256 (l/l_{key}) samples that would fit if the message was not delta-difference encoded.

Δ_{max} is a function of the size of the frame (l_{frame}), so l_{frame} can also a parameter for optimization based on the expected maximum rate of change (r_{max}) of the physical parameters to be sent if the physical layer supports a variety of frame sizes.

This delta-differencing was removed in Goby2 with the addition of user-defined codecs.



(a) Temperature variation with depth and position

(b) Temperature-Depth profile

Figure 2.18: Temperature data from the GLINT10 experiment a CTD instrument mounted on AUV Unicorn available at runtime via `goby-acomms` using delta-differenced encoding.

Instead, as a user of Goby2, a similar concept was defined using an entropy coder of delta-differenced samples. This idea is explored in detail in chapter 3. The Goby2 version can be thought of as a generalization to arbitrary probability distributions of the concept used during this experiment.

2.6.6 GLINT10

All the features and implementation of `goby-acomms` version 1 were in place for the GLINT10 sea trial. The DCCL messages and corresponding V_{base} and ttl used for dynamic priority queuing are given in Table 2.10.

The key new items for GLINT10 were

- Expansion of delta-difference encoding mentioned in Section 2.6.5 to support any arbitrary `<float>` field, not just those from a CTD instrument. This enabled the new "back-fill" `LAMSS_STATUS_FILLIN` message which keeps a history of vehicle positions regularly sampled (in this case twice per minute). These were queued with a low V_{base} of 0.4 relative to the other messages (see Table 2.10). Thus, in cases of low throughput due to unfavorable environmental conditions other data messages would be sent. However, when the throughput went up the queued up `LAMSS_STATUS_FILLIN` messages would be sent, giving the topside operators a somewhat delayed but still relevant history of the vehicle's manuevers. When developing complex adaptive autonomy, this is critical for debugging and understanding

Table 2.10: Summary of DCCL Messages used in the GLINT10 Experiment

Message Name	Category	DCCL size (bytes) [a]	Estimated Equivalent CCL Size (bytes) [b]	ttl [c]	V_{base} [c]	Description
LAMSS_DEPLOY	Command	31	40	300	1000	Underwater vehicle command message.
LAMSS_PROSECUTE	Command	31	40	300	1000	Underwater vehicle command message: prosecute detected target.
ACOUSTIC_MOOS_POKE	Command	32	31	300	10000	Underwater debugging / safety message.
LAMSS_STATUS	Data / Collaboration	27	35	300	1.5	Vehicle Status message (position, speed, Euler angles, autonomy state)
LAMSS_STATUS_FILLIN	Data	29	52	1800	0.4	Vehicle Status message historical "back-fill" (delta-difference encoded).
LAMSS_CONTACT	Data	29	34	600	2	Passive acoustic contact report message.
LAMSS_TRACK	Data	29	34	300	4	Passive acoustic track report message.
LAMSS_BTR	Data	64	63	7200	1	Beam-Time Record Data from a towed passive acoustic array.
LAMSS_CTD	Data	256	496	1800	1	Salinity, temperature, depth data from a CTD instrument (delta-difference encoded).

[a] For DCCL: see section 2.2.
[b] Since CCL does not implement these messages, these size estimates are based on the closest available message in the existed CCL message set.
[c] For Priority queuing: see section 2.3.

the vehicles' performance. Furthermore, all `LAMSS_CTD` messages were also decoded as status messages since they contain the three dimensional location of the

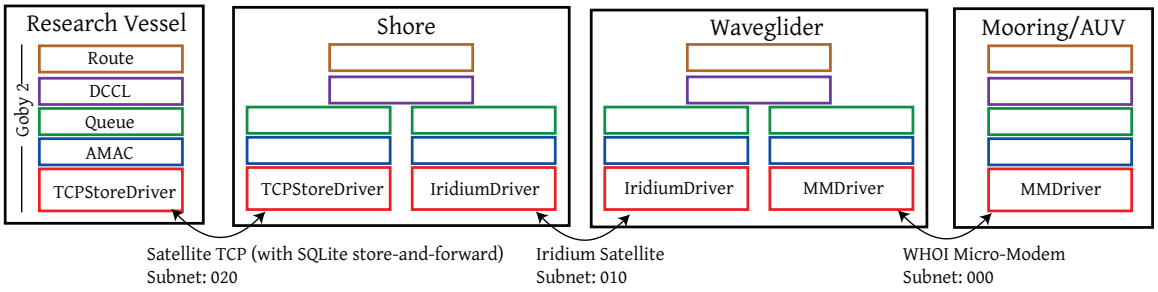


Figure 2.19: Structure diagram of the Goby components for the Tiger12 cruise to form a seamless link from the Research vessel to the Tiger mooring (and intermediate nodes) over a collection of different link types (satellite-TCP, satellite-Iridium "call", acoustic Micro-Modem).

vehicle at the time of the sample.

- The auto-discovery decentralized TDMA MAC described in section 2.4 was tested using two vehicles, one gateway buoy and one ship deployed WHOI Micro-Modem. Due to the lack of a cycle initialization packet that could be lost, transmissions from ranges of up to 4 km were successfully made. Using the standard centralized TDMA that we have used for the previous experiments, we saw transmissions up to 2 km. It is difficult to quantify in-water performance of MAC schemes without a robust understanding of the environmental effects on propogation.

2.7 Goby2 Field Trials

Since Goby2 is new, it has only been involved in three field trials (plus many hours of simulation and hardware-in-the-loop testing), compared to over a dozen or more that the authors are aware of that used Goby1:

- CAPTURE11 (August 2011): Chief scientist: C. Murphy (WHOI). Nodes: 2 OceanServer Iver2 AUVs (R. Eustice, University of Michigan), 1 WHOI SeaBED AUV (H. Singh, WHOI), 1 unmanned surface vehicle (F. Hover, MIT), and two research vessels, all equipped with an acoustic WHOI Micro-Modem. This experiment (using hardware and software assets from four different laboratories) successfully demonstrated multi-hop transmission of rich (e.g. imagery) datasets using C. Murphy's CAPTURE protocol. CAPTURE used Goby2 DCCL and ModemDriver, showing its extensibility

in the hands of several other research groups that do not collaborate on a daily basis. The design and results of CAPTURE are detailed in [50].

- Cyborg12 (May 2012): Chief scientist: A. Balasuriya (MIT). Nodes: 1 Bluefin 9" AUV, 1 WHOI Micro-Modem shallow water buoy, 1 research vessel, all equipped with an acoustic Micro-Modem. This trial was an engineering test in the process of developing a collaborative network of human experts and AUVs for mine countermeasures.
- Tiger12 (June 2012): Chief scientist: L. Freitag (WHOI). Nodes: 1 Tiger sonar mooring (prototype for an AUV), 1 Liquid Robotics WaveGlider, 1 research vessel. This cruise was a test of using Goby2 over a heterogenous mix of physical links for the purpose of sending control messages from the research vessel to the Tiger mooring and status messages in the opposite direction. The research vessel was out of acoustic range of the mooring, so the waveglider was used as a forwarding router between the acoustic (sub-sea) and Iridium subnets. Since Iridium does not well support calls directly to mobile nodes, a shore station was used as a store-and-forward intermediate. All messages were logged here and retrieved at the next opportunity by the research vessel. This system provided reliable end-to-end transmission of DCCL messages for several days of continuous operation. While DCCL was designed initially for acoustic networks, it is equally useful for satellite links because of the similar characteristics (low throughput and very high cost per bit). Fig. 2.19 shows the network setup of this experiment with its three different types of links.

These trials have made us confident that we were successful in the primary design themes (see section 2.1.1) of third-party extensibility and field reliability. Furthermore, only one of these (Cyborg12) would have even been possible with Goby1.

2.8 Conclusion

`goby-acomms` provides an acoustic networking suite that combines high usability at sea with techniques intended to make the most out of the very low throughput provided by acoustic telemetry. It is comprised of four modules that could be interchanged with a suitable replacement as research advances in a particular areas:

- *dccl*: provides encoding and decoding. The major contribution from DCCL is the ability to create custom objects that can be serialized to very short messages, with an emphasis on message size efficiency over features and abstraction.

- *queue*: deals with the common problem in acoustic networks of having too many messages. *queue* provides a way to prioritize messages based both on the time sensitivity and the overall value of the message.
- *amac*: implements a simple TDMA scheme with auto-discovery requiring no control messages to be sent and thus not using any bandwidth that might be better used for mission data.
- *modemdriver*: provides an abstract interface for an acoustic (or other low-bandwidth carrier) modem and an implementation of this interface for various transport devices or systems: the widely used WHOI Micro-Modem, the MOOS "uField" simulation environment, UDP/IP, and a store-and-forward server based on ZeroMQ and TCP.

`goby-acomms` emphasizes robustness through object-oriented design to provide a communications architecture that can support real field operations with underwater robots. The hope is that `goby-acomms`, or at least some of the ideas within, can move the field of collaborative underwater robotics and artificial intelligence forward. `goby-acomms` is freely available with the Goby Underwater Autonomy Project from `http://launchpad.net/goby`. The Goby libraries are licensed under the GNU Lesser General Public License (LGPL) and gladly accepts contributions from members of the marine acoustic networking and robotics community.

3 *Non-disruptive Technique: autonomous modeling to improve source coding*

3.1 Introduction

3.1.1 Motivation

Users of mobile marine platforms such as autonomous underwater vehicles (AUVs) and unmanned surface vehicles (USVs) are one of the major beneficiaries of improved acoustic communication capabilities, since the need to move often precludes the use of fiber optic communication tethers. These vehicles are also becoming increasingly "intelligent"; they are outfitted with substantial computational ability and are capable of fulfilling complex mission components or entire missions autonomously; for examples, see [4, 51, 52].

For many types of AUV missions it is required or desirable for the vehicle (here, the *sender*) to transmit accurate and frequent vehicle position measurements to collaborating vehicles or a human operator (the *receiver*). For example, oceanographic missions require the position where sensor samples were taken, and collaborative target detection tasks require a history of positions to avoid unnecessary redundant coverage, or facilitate coordinated control manuevers such as formation flying. Furthermore, as vehicle navigation decisions become increasingly automated, human operators desire increased assurance that their highly expensive vehicles are operating correctly and away from hazards.

This need for vehicle position knowledge can often consume much or all of the available acoustic link's throughput in fielded vehicles. In this chapter, a system is devised that uses a matched state observer on the *sender* and *receiver* to reduce the position vector to a vector of differences from the modeled state. The probability distribution of these differences is modeled *a priori* or adaptively built from prior statistics. The resulting distribution is coupled with an arithmetic entropy encoder to provide highly compressed

position vectors.

3.1.2 Related Work

Much work has been done on understanding the physical channel for acoustic telemetry; see [53] for a review of the last decade. While the focus has been on error-free channel coding and transmission of datagrams, little has been published in the marine domain on source coding of underwater measurements as evidenced by [54] and a dearth of coverage of source coding in the major underwater networking review papers [14, 55].

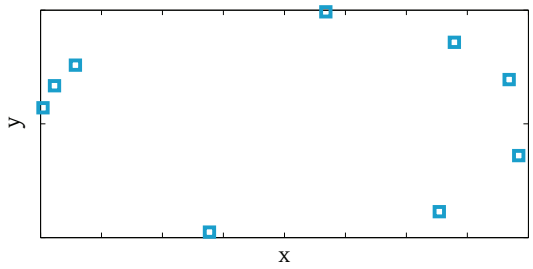
One exception is Murphy's work [50, 56], in which he uses transform compression (e.g. the discrete wavelet transform) to source encode imagery and historical time series of scalar data. While not specifically addressed, one could apply this technique to source encode vehicle state vectors. However, the transform codes provide at best an approximation of the original signal until the entire sequence is received. Furthermore, the performance of the transform compressor improves with longer sequences of data. These limitations make this technique less suitable when near-realtime telemetry of accurate vehicle positions is required, and more suitable for less time-sensitive transmission of previously collected data (such as scientific scalar instrumentation data as Murphy uses in his examples).

Outside the marine domain resides the closest related work, by Koegel and Mauve [57]. They investigate the information content of a moving urban or highway land vehicle trajectory (defined as a time series of vehicle positions). In this domain, the throughput is much less limited, but the desired number of trajectories to transmit is high. Thus, the ratio of trajectory number to available throughput is similar to the marine domain where we have a small number of trajectories, but a very low throughput link. Koegel and Mauve suggest the use of a Kalman filter for this problem but do not further investigate it, as is done in this chapter.

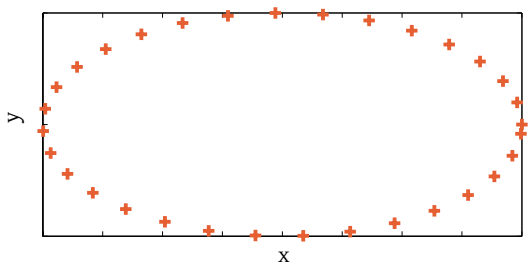
Others have looked at techniques to losslessly encode very large sets of trajectories from terrestrial GPS data, such as the linearization and clustering approach from [58] and the road-network algorithm in [59]. Many of these techniques are focused on the problem of efficiently storing and transmitting full datasets "offline". In the marine domain, it is typically far easier to offload previously collected datasets after vehicle recovery or over electromagnetic wireless links after the vehicle surfaces. Thus, this chapter focuses on a technique intended to telemeter realtime or near-realtime data ("online"), which is the more pressing problem for underwater systems due to the highly constrained acoustic

link.

Finally, the multi-vehicle control field has been looking at the problem of cooperative localization (CL). In cooperative localization, a group of robots shares some subset of their sensor data or internal state estimates so that each robot eventually has an estimate of the pose (position and orientation) of all the other collaborating robots. Some of the robots may specifically be tasked to maneuver to decrease the overall estimate error. Typically, cooperative localization researchers are not especially concerned with the extremely low-bandwidth constraints that we find in underwater communication links, but there are a few exceptions worth noting here. Trawny, et al. [60] uses a highly quantized (as few as one bit per time step) Kalman filter dubbed the Iteratively-Quantized Extended Kalman filter (IQEKF), which they show in simulation to provide acceptable performance relative to a real valued Kalman filter. Nerurkar and Roumeliotis [61] then extend the idea to incorporate asynchronous communications (where one node cannot necessarily talk bi-directionally to another). Bahr [62], and later Fallon, et al. [63] address the problem of CL on AUVs. They use a surface craft (with an accurate GPS fix) to improve the dead-reckoning ability of underwater vehicles (which are GPS-denied). All of these approaches group localization and control into a single problem. Thus, while this may lead to a overall more efficient solution, it is potentially less robust and is difficult to deploy on the heterogeneous networks that exist in real sea robotics (gliders, AUVs, surface vehicles: all potentially from different designers). This chapter shows a system for robustly transmitting well-defined (predetermined precision, e.g. 1 meter) quantizations of the vehicle's state at a known sampling rate (e.g. 1/10 Hz), and its accuracy is not contingent on the performance of a given tracking algorithm. This approach is decoupled from the vehicle's navigation system through use of a generalized dynamics model, in an analogous way that the "backseat" autonomy is decoupled from the "frontseat" control in the Unified Command and Control system (see Appendix A). One of the major challenges facing practical field robotics is the management of complexity, especially involving heterogeneous collections of vehicles. Such a separation means that (some, at least) of the gains provided in using delta-based technique can be had at the "backseat" level without intimate knowledge or specification of the low level ("frontseat") control.



(a) Traditional ("newest when possible") approach.



(b) The technique given in this chapter ("evenly sampled historical timeseries")

Figure 3.1: Positions of the *sender* while performing an elliptical pattern available at the *receiver* using two types of position telemetry systems in the case of 70% packet loss.

3.2 Approach

The usual approach of vehicle position telemetry systems is to send the newest available position (as an independent vector) when the acoustic link becomes available as governed by the network medium access control (MAC). This is the approach used by the field work done by a variety of groups: Webster, et al. [44], Marques, et al. [64], Rajala, et al. [65], Kunz, et al. [1], Grund, et al. [20], and in much of our prior work on "Unified Command and Control", which is described in Appendix A.

This "newest-when-possible" approach, however, can lead to significant gaps in the vehicle's position history as observed by the *receiver* when the packet loss of the link is high as sketched in Fig. 3.1a. A different approach (sometimes colloquially referred to as "backfilling" of positions) is to send an evenly sampled historical timeseries. When packets are lost, an automated repeat-request (ARQ) mechanism retransmits the packets until the missing packets are received. Thus, over time, the entire time-series is received as shown in Fig. 3.1b. Such a "backfilling" system is the goal of the present chapter's work.

The goal of this system is to transmit a sampling (at sample period τ) of a time series of vehicle positions $\mathbf{y}(t)$ where

$$\mathbf{y}(t) = \begin{bmatrix} t \\ x(t) \\ y(t) \\ z(t) \end{bmatrix} \tag{3.1}$$

is the Cartesian position of the vehicle with reference to a common known datum, with

z given as the negative of the vehicle's depth[1].

Position measurements are transmitted as one of two types of messages:

- Full transmissions: The vector $\mathbf{y_f}$ which includes the time and full position of the vehicle relative to the experiment datum where

$$\mathbf{y_f} = \begin{bmatrix} t_f \\ x(t_f) \\ y(t_f) \\ z(t_f) \end{bmatrix} \tag{3.2}$$

  This message is used once at the start of each mission to synchronize the states of the *sender* and *receiver*. t_f represents the time of the mission start.

- Delta transmissions:

$$\mathbf{dy}[n] = \begin{bmatrix} dx[n] \\ dy[n] \\ dz[n] \end{bmatrix} \tag{3.3}$$

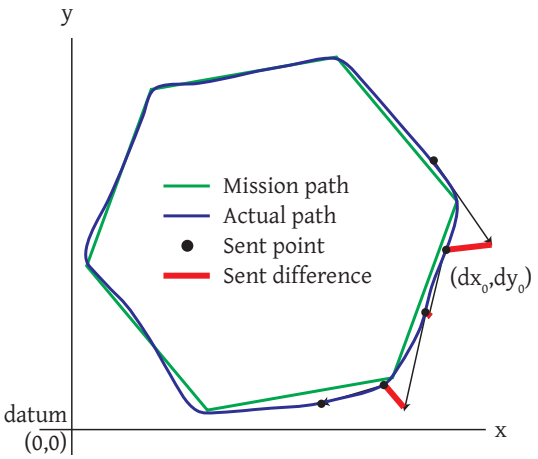
  where $n = 0, 1, 2, 3, \ldots$. This delta transmission is sent continuously following a full transmission or prior delta transmissions until the vehicle was removed from operation for greater than τ seconds, after which a full transmission is sent to reinitialize the receiver's state. Determining the values of **dy** via state observation is described in Section 3.3. The sample number n does not need to be transmitted, assuming the lower layers of the network stack can provide in-order receipt of messages without duplicate packets. In this case, the decoder simply increments n on each message received. This can be easily accomplished with automatic repeat request (ARQ) with a single alternating bit to discard duplicates. The sampled $\mathbf{y}(\mathbf{t})$ can be reconstructed at time $n = n_0$ using

$$\mathbf{y}(t_f + n_0\tau) = \begin{bmatrix} t_f + n_0\tau \\ \hat{\mathbf{y}}_{prior}[n_0] + \mathbf{dy}[n_0] \end{bmatrix} \tag{3.4}$$

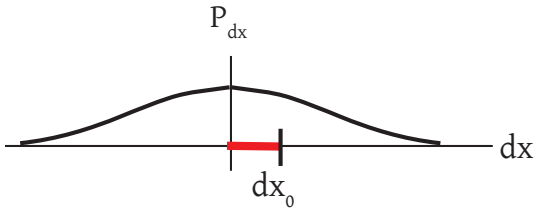
  where $\hat{\mathbf{y}}_{prior}[n]$ is the prior estimate of the state observer extrapolated to discrete time step n.

  Fig. 3.2 illustrates the process of generating these delta transmissions.

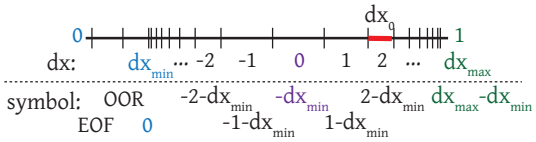
[1]For the purpose of this work, the transformation used from geodetic (latitude, longitude) to Cartesian (local) coordinates does not matter. One could use, for example, the Universal Transverse Mercator transformation with the WGS'84 ellipsoid [49] or the North-East-Down transformation of an earth-centered earth-fixed frame; see [66].



(a) Generation of differences (dx and dy) between the vehicle's actual position and the extrapolated position (generated on both sender and receiver using tracked positions previously transmitted). Two techniques for this process are explained in Section 3.3.



(b) Example of the probability distribution used to represent the error between the actual and extrapolated positions. These distributions are built up in Section 3.4.



(c) Arithmetic coding symbol intervals (each dx is mapped to a symbol with 1-meter precision).

Figure 3.2: Overview illustration for the delta transmissions showing the mapping of vehicle position (a) to a given probabilistic model (b) used to generate the symbol intervals required for arithmetic coding (c).

3.3 State Observation

A state observer is typically used in control systems to model the internal state of a system often in order to apply feedback to stabilize the system. Here, a state observer is used in a different way. A model of the system (in this case a vehicle in three-dimensional motion) is observed by both the *sender* and *receiver* of the communications link using a reduced set of the data, namely only the previously telemetered $\mathbf{y}[n]$. The difference between this reduced model and the (presumably more accurate) output of the vehicle's navigation system (which may incorporate other state observers and filters) is taken. This difference (which can be thought of as an error) is the value used to transmit. This operation is visualized in Fig. 3.2a.

Two state observers were used in this work: a deterministic ("fixed speed") model and a stochastic model based on the Kalman filter.

3.3.1 Fixed speed observer

The fixed speed model is useful for AUVs that drive at a roughly constant speed in the xy-plane while underway, which includes most of the torpedo-shaped vehicles such as the Bluefin and REMUS vehicles. This model uses the prior two transmitted positions to determine yaw Ψ where

$$\Psi = \tan^{-1} \frac{y[n-1] - y[n-2]}{x[n-1] - x[n-2]} \tag{3.5}$$

The vehicle's last position is extrapolated using this heading at the fixed speed s, and this is used as a reference for the difference (or error) to the actual vehicle position to be transmitted, such that

$$\begin{bmatrix} dx[n] \\ dy[n] \end{bmatrix} = \begin{bmatrix} x[n] - (x[n-1] + \tau|\mathbf{v}| \cos \Psi) \\ y[n] - (y[n-1] + \tau|\mathbf{v}| \sin \Psi) \end{bmatrix} \tag{3.6}$$

For depth, since manueuvers are less predictable, the last difference is used:

$$dz[n] = z[n] - (z[n-1] - z[n-2]) \tag{3.7}$$

The simplicity of this model means that it is computationally inexpensive thus adding negligible overhead to the limited resources on the vehicle. However, it is not applicable for AUVs that can change their speeds substantially while underway. For this, a general purpose model was developed, using the Kalman filter.

3.3.2 Kalman filter observer

Assumptions

To keep the model as general as possible for a moving vehicle, the following assumptions were made:

- Motion along each Cartesian dimension is independent of the other dimensions.
- The acceleration increment

$$d\ddot{x}[n] = \int_{\tau} \dddot{x} dt \tag{3.8}$$

is a normally distributed white noise process in all dimensions with $\sigma = \sigma_j$ (in the target tracking literature this is referred to as the Wiener-sequence acceleration model [67]).

These assumptions are somewhat unrealistic (e.g. motion in x and y are rarely independent), but serve to capture the dynamics of the vehicle sufficiently for the given task without introducing significant computational overhead or loss of generality.

State space model

Given these assumptions, a linear state space $\mathbf{x}[n]$ is defined as

$$\mathbf{x}[n] = \begin{bmatrix} \mathbf{y}[n] \\ \dot{\mathbf{y}}[n] \\ \ddot{\mathbf{y}}[n] \end{bmatrix} \tag{3.9}$$

where

$$\mathbf{y}[n] = \begin{bmatrix} x(t_f + n\tau) \\ y(t_f + n\tau) \\ z(t_f + n\tau) \end{bmatrix} \tag{3.10}$$

The dynamics of the vehicle in discrete time are thus given by

$$\mathbf{x}[n+1] = \mathbf{A}\mathbf{x}[n] + \mathbf{G}d\ddot{x}[n] \tag{3.11}$$

with state transition model

$$\mathbf{A} = \begin{bmatrix} \mathbf{I_3} & \tau\mathbf{I_3} & \frac{\tau^2}{2}\mathbf{I_3} \\ 0_3 & \mathbf{I_3} & \tau\mathbf{I_3} \\ 0_3 & 0_3 & \mathbf{I_3} \end{bmatrix} \tag{3.12}$$

where process noise $\mathbf{w}[n]$ is normally distributed

$$\mathbf{w}[n] = \mathbf{G}d\ddot{x}[n] \sim \mathcal{N}(0, \mathbf{Q}) \tag{3.13}$$

Given the Wiener-sequence acceleration model chosen above, the noise covariance $\mathbf{Q}$ is given as

$$\mathbf{Q} = \sigma_j^2 \mathbf{G}\mathbf{G}^T \tag{3.14}$$

where

$$\mathbf{G} = \begin{bmatrix} \frac{\tau^2}{2} & \frac{\tau^2}{2} & \frac{\tau^2}{2} & \tau & \tau & \tau & 1 & 1 & 1 \end{bmatrix}^T \tag{3.15}$$

Kalman filter

The Kalman filter [68] is a recursive Bayesian estimator for linear systems with normally distributed noise assumptions. In the marine robotics domain, Kalman filters have been typically used for two purposes: 1) tracking of unknown targets based on noisy and infrequent (often sonar) measurements as in [69, 70]; and 2) estimation of the vehicle's navigation solution from a variety of noisy sensors such as gyroscopes, inertial measurement units, pressure sensors and acoustic sensors (long baseline, Doppler velocity logging, altimeters), such as presented in [71–73].

For this work, the Kalman filter is used to predict the state of the system based on a reduced set of measurements $\{\mathbf{y}[n-1], \mathbf{y}[n-2], \ldots\}$, namely those measurements that have already been successfully transmitted to the *receiver*. In a sense, this is similar to the target tracking problem, except that the "target" (the *sender*) is an AUV controlled by the user of the system. This "target" is tracking itself using only the knowledge that the *receiver* (who is also tracking the AUV) has. The goal, as previously mentioned, is to efficiently communicate a more accurate state vector with as few bits as possible. The error between the prediction and the measured state of the system (which is typically a more accurate prediction from the *sender*'s navigation system, which may employ various stochastic filters as well) forms the delta transmission vector $\mathbf{dy}[n]$, also called the "innovation" or measurement residual.

The algorithm presented in this work can be described as a three-step process:

1. Both *sender* and *receiver* and predict the next state vector $\hat{\mathbf{x}}_{prior}[n]$ and *a priori* estimate covariance $\mathbf{P}_{prior}[n]$ where

$$\hat{\mathbf{x}}_{prior}[n] = \mathbf{A}\hat{\mathbf{x}}_{post}[n-1] \tag{3.16}$$

$$\mathbf{P}_{prior}[n] = \mathbf{A}\mathbf{P}_{post}[n-1]\mathbf{A}^T + \mathbf{Q} \tag{3.17}$$

2. The *sender* encodes (Section 3.4) and transmits the delta vector

$$\mathbf{dy}[n] = \mathbf{y}[n] - \mathbf{H}\hat{\mathbf{x}}_{prior}[n] \tag{3.18}$$

using the mapping of estimate to measurement state vectors given by

$$\mathbf{H} = \begin{bmatrix} \mathbf{I}_3 & 0_3 & 0_3 \end{bmatrix} \tag{3.19}$$

This delta vector is then received and decoded by the *receiver*. At this point, the "true" position of the vehicle can be recovered using (3.4).

3. Both ends update the filter state vector $\hat{\mathbf{x}}_{post}[n]$ and estimate covariance $\mathbf{P}_{post}[n]$ with their respective posteriors

$$\hat{\mathbf{x}}_{post}[n] = \hat{\mathbf{x}}_{prior}[n] + \mathbf{K}[n]\mathbf{dy}[n] \tag{3.20}$$

$$\mathbf{P}_{post}[n] = (\mathbf{I} - \mathbf{K}[n]\mathbf{H})\mathbf{P}_{prior}[n] \tag{3.21}$$

using the innovation in combination with the Kalman gain

$$\mathbf{K}[n] = \mathbf{P}_{prior}[n]\mathbf{H}^T\mathbf{S}[n]^{-1} \tag{3.22}$$

where

$$\mathbf{S}[n] = \mathbf{H}\mathbf{P}_{prior}[n]\mathbf{H}^T + \mathbf{R} \tag{3.23}$$

3.4 Arithmetic coding

In the previous section a method was discussed for producing a minimal set of (presumed independent) delta values to transmit. A source encoder can now be chosen to compress these differences.

In this work, arithmetic coding was chosen over various alternatives because of two main advantages:

- Assuming an accurate model, it produces a nearly optimal encoded bitset.
- The modeling process is separate from the coder design. This allows a single implementation of an arithmetic coder to function on many distinct sources of data. It also allows for various models to be evaluated on the source data without redesigning the coder.

The main drawback is that arithmetic coding has a reasonably high computational cost. This is generally not a concern for the underwater vehicle domain since available computing resources typically far outpace the throughput of the acoustic channel.

3.4.1 Generating a source model

The next step in this process is identifying a suitable model. The full transmissions (3.2) are encoded using a uniform probability distribution, since the vehicle could reasonably be redeployed anywhere in the operation region. The process of mapping the source delta data from Section 3.3 is sketched in Fig. 3.2b.

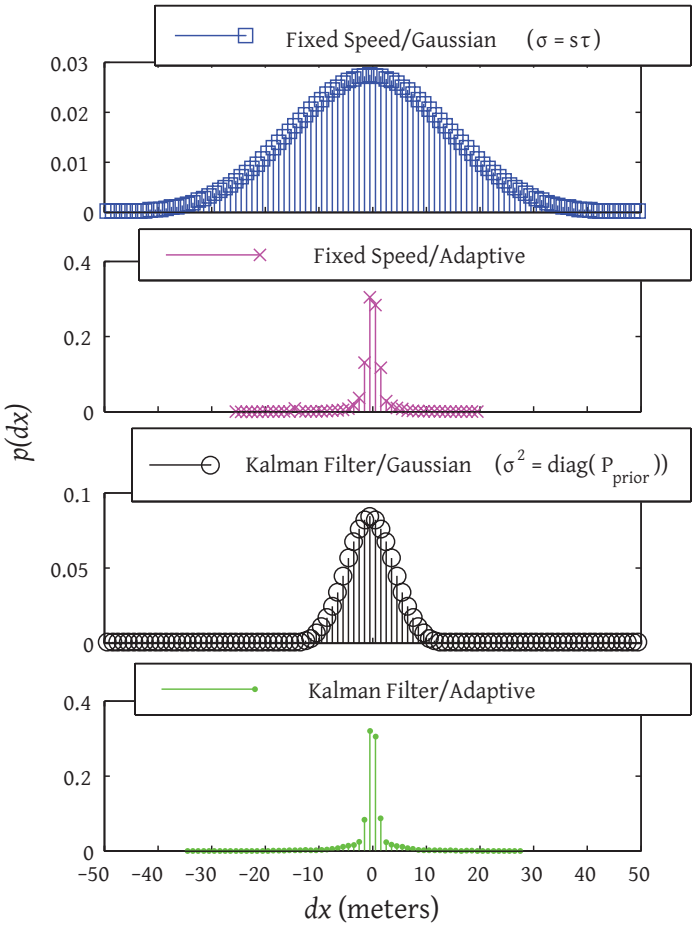


Figure 3.3: The probability distributions (given in (3.26-3.28)) used to arithmetically encode dx and dy in this work. These models were used to produce the experimental results given in Fig. 3.6. Not shown is the uniform distribution given in (3.24).

A priori, it seems logical that the probability distribution governing the delta values $\mathbf{dy}[n]$ would be zero mean, since any pattern the vehicle makes will have an equal number of negative and positive position differences. For example, see the hexagon in Fig. 3.2a. The negative dx on the east side will be offset by the positive dx on the west side. The shape of the distribution is unclear, however, and depends substantially on the maneuvering choices the vehicle makes (tight circles would lead to high error using the dynamic model given in (3.6), straight lines would be low error). Thus, the following distributions were compared (all the non-uniform distributions are shown in Fig. 3.3):

- Uniform (similar for $p[dy]$ and $p[dz]$):

$$p[dx] = \begin{cases} \frac{1}{dx_{max} - dx_{min} - 1} & dx \in [dx_{min}, dx_{max}) \\ 0 & dx \notin [dx_{min}, dx_{max}) \end{cases} \tag{3.24}$$

  where the limits

$$\mathbf{dy}_{min} = \begin{bmatrix} dx_{min} \\ dy_{min} \\ dz_{min} \end{bmatrix}, \mathbf{dy}_{max} = \begin{bmatrix} dx_{max} \\ dy_{max} \\ dz_{max} \end{bmatrix} \tag{3.25}$$

  must be determined *a priori* based on the tolerance for unencodable symbols if the state observer difference exceeds these bounds. In a real system a symbol can be reserved for out-of-range values and the encoder reset to send a new "full transmission" (3.2) when this occurs. The tighter the bounds, however, the less probability mass that is "wasted" on encoding values that will never or rarely occur.

- Normal, with variance $\sigma^2 = (s\tau)^2$:

$$p[dx] = \begin{cases} \mathcal{N}(0, \sigma^2) & dx \in [dx_{min}, dx_{max}) \\ 0 & dx \notin [dx_{min}, dx_{max}) \end{cases} \tag{3.26}$$

  The standard deviation $s\tau$ was chosen so that all possible manuevers including the "worst case" scenario have about 95% of the probability mass. The "worst case" is where the vehicle makes a $180°$ turn immediately after the preceeding transmission so that $dx[n] = 2s\tau$. This means that $\sigma = s\tau$ since

$$\Phi(\mu + 2.0\sigma) - \Phi(\mu - 2.0\sigma) \quad = \quad 0.95 \tag{3.27}$$

  where Φ is the cumulative mass function of the normal distribution. This distribution is used only in conjunction with the fixed speed observer (Section 3.3.1).

- Normal, with variance $\sigma^2 = diag(\mathbf{P}_{prior}[n])$: that is, the variances of the *a priori* estimate covariance from the Kalman filter. This distribution is used only in conjunction with the Kalman filter observer (Section 3.3.2).

- Adaptive: This model starts processing the dataset with the uniform distribution given above, and then equally incorporates the statistics of all previously transmitted symbols. Thus, an accurate model of the vehicle's prior positions is built up to encode future positions. At any sample m_0, the model is

$$p[dx] = \begin{cases} (N_{dx} + 1) / f_0 & dx \in [dx_{min}, dx_{max}) \\ 0 & dx \notin [dx_{min}, dx_{max}) \end{cases} \tag{3.28}$$

where N_{dx} is the number of prior transmissions over $[dx[0], \ldots, dx[m_0 - 1]]$ that had the value dx, and $m = 0$ is the start of the experiment. f_0 is a normalizing constant given as

$$f_0 = m_0 + dx_{max} - dx_{min} \tag{3.29}$$

The model is updated after encoding and after decoding so that the *sender* and the *receiver* can share the same state.

3.4.2 Implementing the arithmetic coder

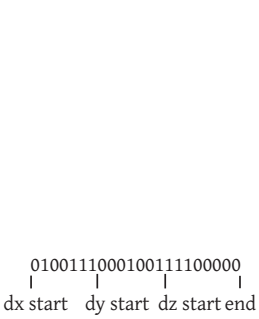
To ensure this work can be easily fielded on AUVs in the near future, the arithmetic coder was implemented in the Dynamic Compact Control Language (DCCL), part of the Goby2 project [74]. The details of the arithmetic coder were based on the widely used integer implementation by Witten, Neal, and Clearly [75] and further clarified in [76]. While the integer implementation is used in the code to avoid underflow, overflow, and precision problems, this thesis uses the floating point notation for clarity. This notation involves encoding a range from [0, 1) using normalized probability models.

The mapping from delta values to symbol space S (shown in Fig. 3.2c) is given by

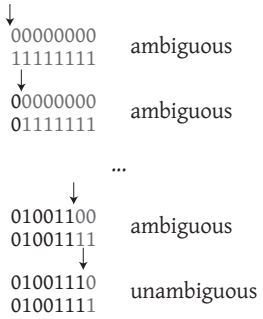
$$S[dx] = \begin{cases} dx - dx_{min} & dx \in [dx_{min}, dx_{max}) \\ \text{out-of-range} & dx \notin [dx_{min}, dx_{max}) \\ \text{end-of-file} & dx \in \emptyset \end{cases} \tag{3.30}$$

with two special symbols: end-of-file (EOF) used to indicate the end of encoding, and out-of-range (OOR) used to indicate any value outside $[dx_{min}, dx_{max})$. An EOF symbol is not required if the number of messages encoded per packet is arranged between sender and receiver ahead of time.

The algorithm for arithmetic coding is well known and will thus not be reprinted here for brevity's sake. However, one innovation was required to conform to the DCCL requirement that decoders consume exactly the same number of bits as the encoder produces. The implementation of an arithmetic coder given in [75] and elsewhere assumes that the decoder can safely read nonsense bits past the end of the file, until the actual end-of-file symbol is decoded. This will not work with DCCL since extra bits used in decoding end up being taken from those required for the *next* field in the message and thereby corrupting all following fields. Thus, in the DCCL implementation used here, the decoder tracks both the upper (current bitset followed by all ones) and lower (current bitset followed



(a) Example DCCL bitstream for a single encoded delta message. The "dy start" and "dz start" markers are given for illustration only; the way DCCL distinguishes the start of one field is where the last field's decoder left off.



(b) Example decoding dx from this example bitstream. Both the upper and lower bounds are tracked, consuming a single bit at a time until the range unambiguously identifies a symbol.

Figure 3.4: Example of the arithmetic decoder for DCCL, showing tracking of decoded ranges to ensure the number of bits consumed by the encoder and the decoder are identical.

by all zeros) bounds of the current symbol, adding bits one at a time until the symbol is unambigously decoded. An example of this process is given in Fig. 3.4. Relatedly, the end of the bitstream must be encoded exactly so that the decoder does not leave extra bits in the stream that would corrupt the next field of the message. The authors of [75] always use two bits to indicate which middle quarter (either [0.25, 0.5) or [0.5, 0.75)) is wholly contained by the final encoder range. However, when at least one end of the encoder range is at one of the bounds (low = 0 and/or high = 1), fewer bits may be required. The exact set of end bits (e) is given by

$$e = \begin{cases} \emptyset & \text{high } = 1, \text{low } = 0, \text{no follow bits} \\ 0 \text{ or } 1 & \text{high } = 1, \text{low } = 0, \text{follow bits} \\ 1 & \text{high } = 1, 0 < \text{low } < 0.5 \\ 0 & 0.5 < \text{high } < 1, \text{low } = 0 \\ 01 & \text{low } < 0.25, \text{high } \geq 0.5 \\ 10 & \text{low } < 0.5, \text{high } \geq 0.75 \end{cases} \tag{3.31}$$

plus any follow bits accrued from prior center expansions around [0.25, 0.75). This is consistent with the (rounded-up) information entropy

$$\lceil H_{bits} \rceil = -log_2(p) = \begin{cases} 0 & \text{high } = 1, \text{low } = 0 \\ 1 & \text{high } = 1, 0 < \text{low } < 0.5 \\ 1 & 0.5 < \text{high } < 1, \text{low } = 0 \\ 2 & \text{low } < 0.25, \text{high } \geq 0.5 \\ 2 & \text{low } < 0.5, \text{high } \geq 0.75 \end{cases} \tag{3.32}$$

for the cases in (3.31).

3.5 Results on experimental data

Here we will examine the performance of the system developed in the previous sections on transmitting hypothetical messages pulled from two experimental datasets:

- The shallow water GLINT10 experiment in the Tyrrhenian Sea containing in excess of sixty hours of cumulative dive time with a Bluefin 21" AUV.
- A dive from the Arctic Gakkel Vents expedition (AGAVE07) with a SeaBED AUV performing a survey at 4 km depth. The dive was twenty-one hours in duration.

These two datasets were chosen to contrast significantly different AUV classes performing different missions to demonstrate the broad applicability of this approach. Specific quantities from the experiments and values chosen here for these examples are given in Table 3.1.

Finally, this system was implemented and run in the field during the MBAT12 trial using a Bluefin 21" AUV and the WHOI acoustic Micro-Modem.

3.5.1 GLINT10

The desired transmission in this example is a Cartesian representation of the vehicle's position

$$\begin{bmatrix} x[n] & y[n] \end{bmatrix} = UTM_{WGS84}(lon[n], lat[n]) - UTM_{WGS84}(lon_d, lat_d) \tag{3.33}$$

and $z[n]$ is the negative of the pressure-derived vehicle depth. UTM_{WGS84} is the Universal Transverse Mercator transformation using the WGS'84 ellipsoid [49], $lon[n], lat[n]$ are

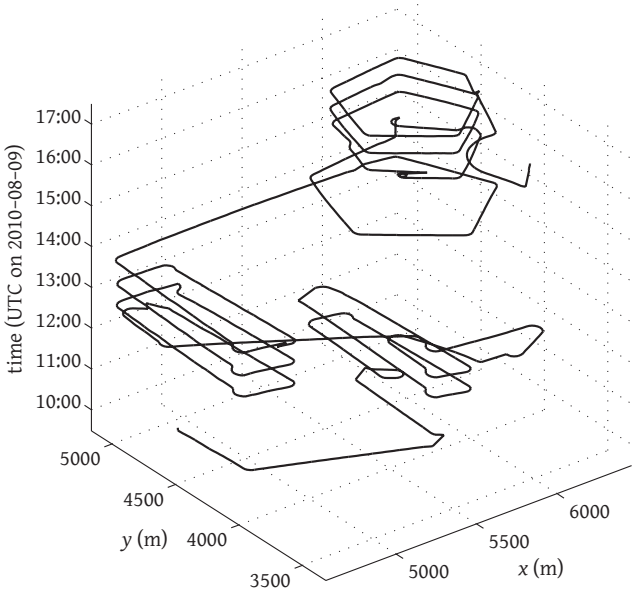
Table 3.1: Experimental Parameters

Parameter	GLINT10	AGAVE07	MBAT12
Delta model bounds: $[\mathbf{dy}_{min}, \mathbf{dy}_{max})$	[-50, 51) m, except Fig. 3.9a		
Transmitted x, y, z precision	1 m		
Jerk variance σ_j^2	10^{-3}, except Fig. 3.9b		
Measurement covariance $\mathbf{R}$	$25\mathbf{I_3}$		
Time between messages (τ)	10 s	10 s	5 s
Number of full transmissions (N_f)	199	34	1
Mean size of full transmission	60 bits	61 bits	55 bits
Number of delta transmissions (N_d)	24420	7370	660
Size of delta transmission	See Fig. 3.6 & Table 3.2		
Vehicle xy speed (s)	1.5 m/s	0.2 m/s	1.4 m/s
Water depth (D)	110 m	4140 m	20 m
Experiment datum (lat_d, lon_d)	42.45667°N, 10.875°E	85.61667°N, 85.75°E	42.38°N, 70.96°W

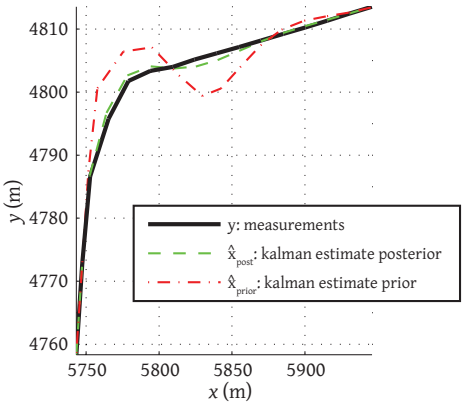
the vehicle's longitude and latitude, and lon_d, lat_d are the longitude and latitude of the experiment datum, a reference used for convenience (unrelated to the UTM zone datum).

Position data

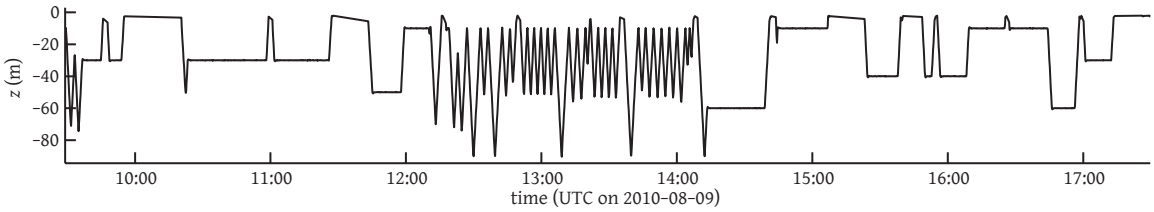
A representative subset of the data used is plotted in Fig. 3.5, and represents one AUV performing a variety of data collection and adaptive autonomy missions. The details of the missions are not of interest here, as the goal is to develop a technique for communicating position data regardless of the vehicle's mission. As can be seen from Fig. 3.5a, the AUV performed a variety of polygonal excursions interrupted by straight-line waypoints. In depth, both profiling "yoyo" and constant depth maneuvers were used. In Figs. 3.5b and 3.5d, the Kalman filter state vectors are plotted, as well as the measured position of the vehicle from the navigation system. The prior estimate deviates the most when the vehicle manuevers. This is expected since the causal motion model developed in Section 3.3.2 has no way to predict these maneuvers and once they occur they are tracked as random changes in the vehicle's jerk. A more specific motion model would likely improve the tracking here, but at the cost of loss of general applicability to a variety of vehicles and mission types. In any case, the required causality of the model will always limit the performance of this system to some degree.



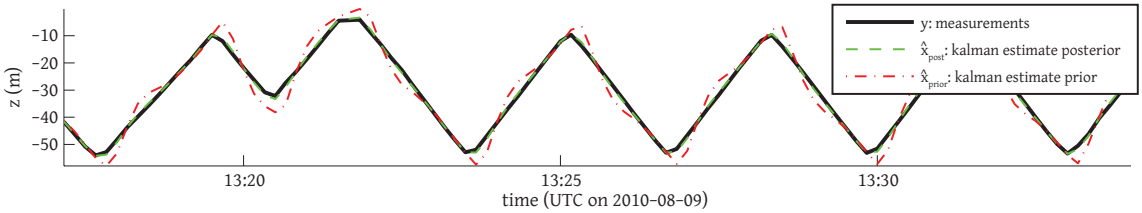
(a) Representative x and y positions of AUV Unicorn (showing 14% of the dataset used in the encoder results)



(b) Detail view of measurements compared to Kalman state vectors $\hat{\mathbf{x}}_{post}$ and $\hat{\mathbf{x}}_{prior}$ from time 16:47 to 16:50 as the AUV turns a corner.



(c) Depth excursion of the AUV Unicorn over the same data subset as in part (a).



(d) Zoom of part (c) showing the Kalman state vectors in z (negative depth). The value transmitted (dz) is the difference between the measurement and the $\hat{\mathbf{x}}_{prior}$ as given in (3.18). As shown here, this difference is highest following a sharp manuever.

Figure 3.5: Example subset of AUV Unicorn navigation data used for the experimental analysis from the GLINT10 cruise.

Encoder results

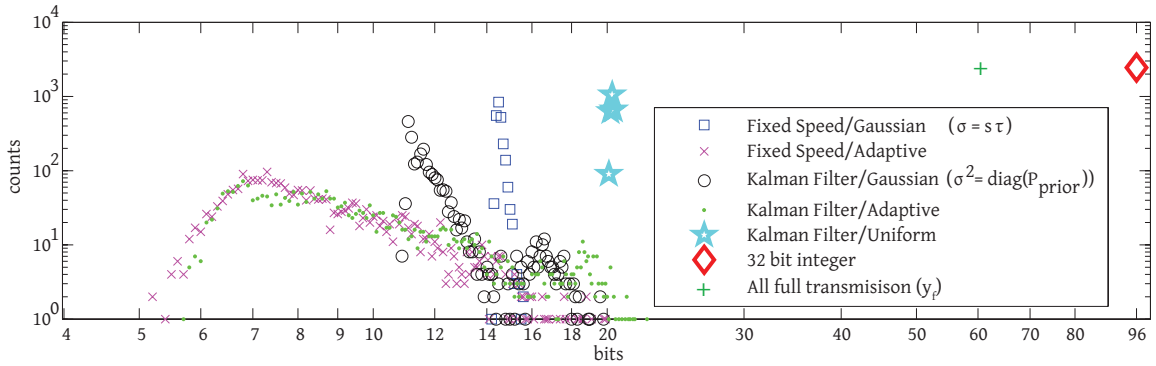
Each of the distributions given in Fig. 3.3 was used with the arithmetic coder discussed in the prior section to encode the dataset. The resulting size of each message was recorded and the statistics plotted in Fig. 3.6a, along with the uncompressed 32-bit integer as a reference point. The moments of these results are summarized in Table 3.2. As expected, the Gaussian model performed better than the uniform distribution since it makes use of the dynamic models from Section 3.3 where low errors are more probable than high errors (the vehicle in general continues on a similar path of motion). However, both of the Gaussian distributions overstate the error significantly from the adaptive distribution, as seen by the difference in standard deviation between the two models in Fig. 3.3. Once the adaptive distribution was initialized with sufficient data, it easily outperforms the results using the other distributions. Furthermore, the fixed speed dynamic model using the adaptive distribution is an improvement of 86% over the widely used Compact Control Language [39], which uses 61 bits to encode a vehicle position in the "MDAT_STATE" message.

Comparing the two dynamic models used, the Kalman filter has a significant edge with the Gaussian distribution since it produces an uncertainty model ($\mathbf{P}_{prior}$) as part of the state estimation process. For the adaptive distribution, however, the fixed speed model performs slightly better, especially since it has lower standard deviation due to a smaller number of large (i.e. 16-20 bit) messages.

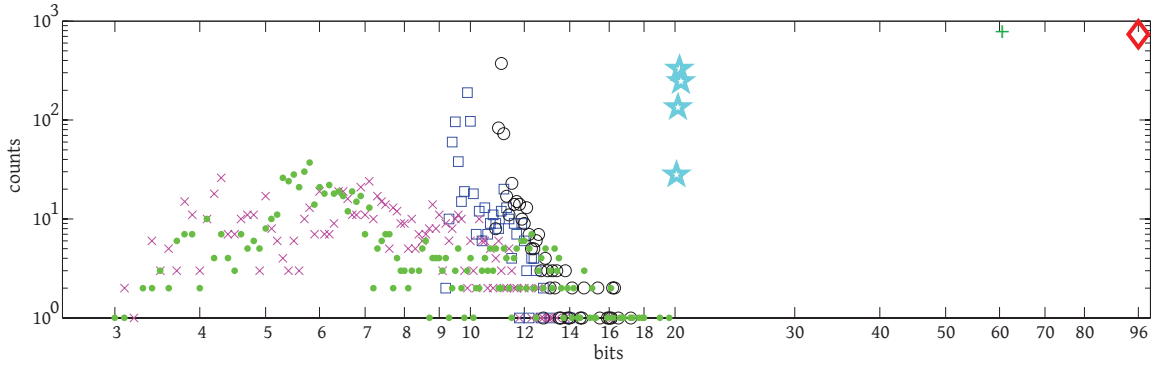
3.5.2 AGAVE07

As with the GLINT10 dataset, the vehicle's Cartesian position during AGAVE07 (Fig. 3.7) was hypothetically transmitted, but this time using the AlvinXY transformation from latitude and longitude to x and y [66]. The size of each delta message was computed for the same types of models as for the GLINT10 experiment; these results are plotted in Fig. 3.6b. In general, the results from the two datasets are similar. There are two main differences: 1) the overall size of the messages generated from the non-uniform distributions are smaller for the AGAVE07 dive than for the GLINT10 trial; and 2) the fixed speed model outperforms the Kalman filter model on the Gaussian distribution for the the AGAVE07 dataset.

Both differences are likely due to the difference in vehicle speeds. Since the resolution transmitted (1 m) and time step ($\tau = 10$ s) were kept constant between experiment



(a) GLINT10: The data are generated for the models given in Fig. 3.3 each operating on the full dataset from the Bluefin 21" AUV "Unicorn".



(b) AGAVE07: Results from the SeaBED AUV "Jaguar" dive on 2007-07-27 using the same distribution types.

Figure 3.6: Log-log plot of the number of delta messages generated with a given size (in bits) for each experiment. Two more data points are provided for comparison: 1) the size if only full transmissions were used (same as the Goby DCCL default algorithms given in Table 2.3); and 2) an uncompressed 32-bit integer representation.

datasets, the slower vehicle ("Jaguar" from AGAVE07) will diverge less from the expected position. The variance of the normal distribution for the fixed speed model is based on speed. In the case of the AGAVE07 results, this is a narrower distribution, more closely matching the adaptive model than in the GLINT10 case.

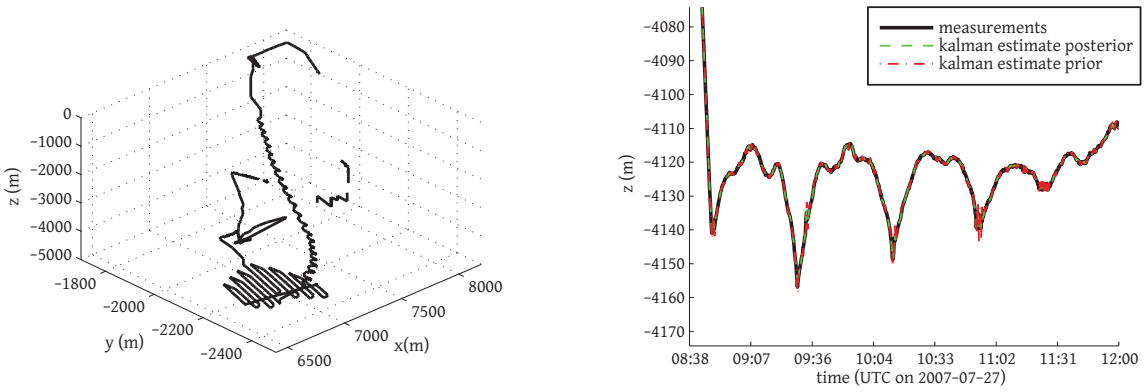
3.5.3 MBAT12

In this experiment, the technique described in this chapter was demonstrated *in situ* onboard a Bluefin 21" AUV (the *sender*) equipped with a WHOI acoustic Micro-Modem [77]. The *receiver* was a buoy equipped with both a Micro-Modem and a radio link to the research vessel. For comparison, the state of the vehicle was transmitted at each acoustic

Table 3.2: Experimental Results

Dynamic Model	Distribution	Mean[a]	Standard deviation[a]	Compression [b]
GLINT10				
Fixed speed	Gaussian ($\sigma = s\tau$)	14.6	0.157	85%
Fixed speed	Adaptive	8.68	2.22	91%
Kalman filter	Gaussian ($\sigma^2 = diag(\mathbf{P}_{prior}[n])$)	12.0	1.42	87%
Kalman filter	Adaptive	9.76	3.28	90%
Kalman filter	Uniform	20.3	0.0821	79%
AGAVE07				
Fixed speed	Gaussian ($\sigma = s\tau$)	10.1	0.746	89%
Fixed speed	Adaptive	7.11	2.23	93%
Kalman filter	Gaussian ($\sigma^2 = diag(\mathbf{P}_{prior}[n])$)	11.4	0.839	88%
Kalman filter	Adaptive	7.45	3.00	92%
Kalman filter	Uniform	20.3	0.0814	79%
MBAT12				
Kalman filter	Gaussian ($\sigma^2 = diag(\mathbf{P}_{prior}[n])$)	11.1	0.481	88%

[a] Mean and standard deviation given in bits.
[b] Relative to a 3 element 32-bit integer representation.



(a) Three-dimensional Cartesian position measurements for the full dataset.

(b) Subset of depth showing the Kalman state vectors.

Figure 3.7: SeaBED "Jaguar" dive used from the AGAVE07 experiment for the results in Fig. 3.6b.

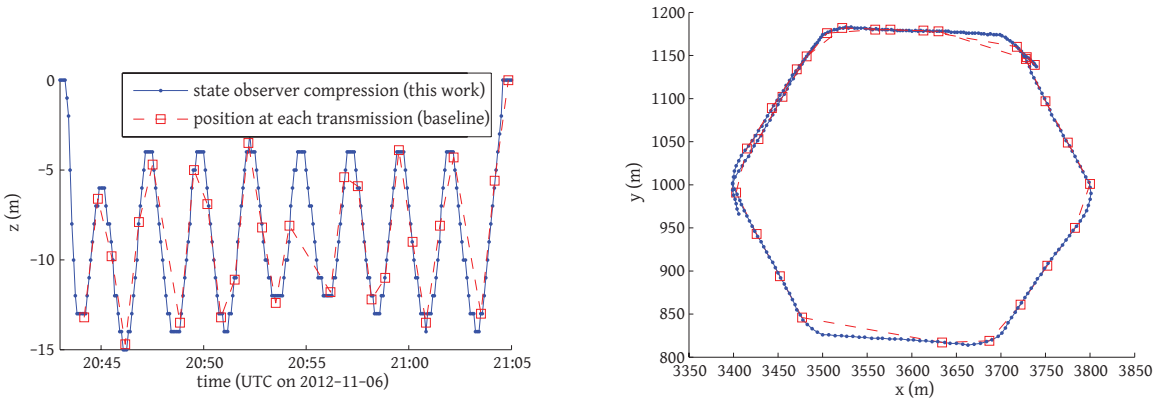
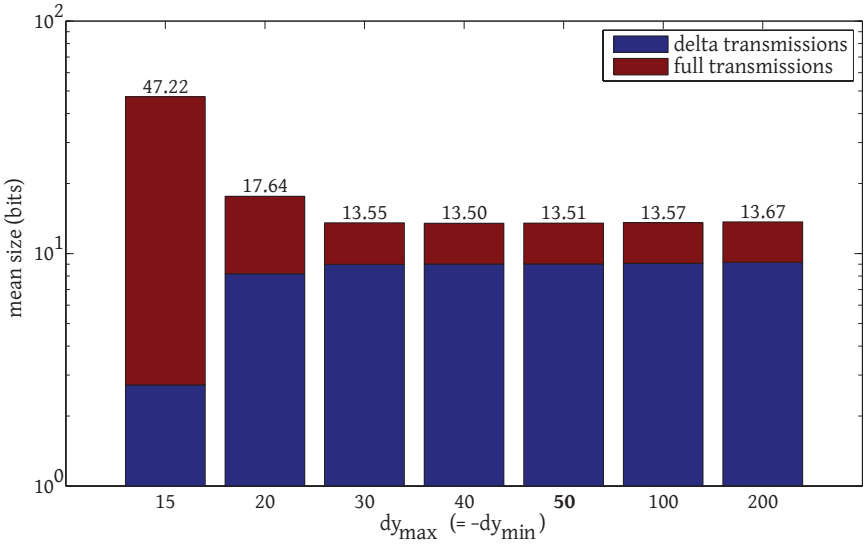


Figure 3.8: Position of the *sender* vehicle as seen by the *receiver* during the MBAT12 field trial. The state observer compression technique developed in this chapter is compared to the traditional position sent with each acoustic data transmission. Note that this new technique removes the aliasing present in the vehicle's depth excursions.

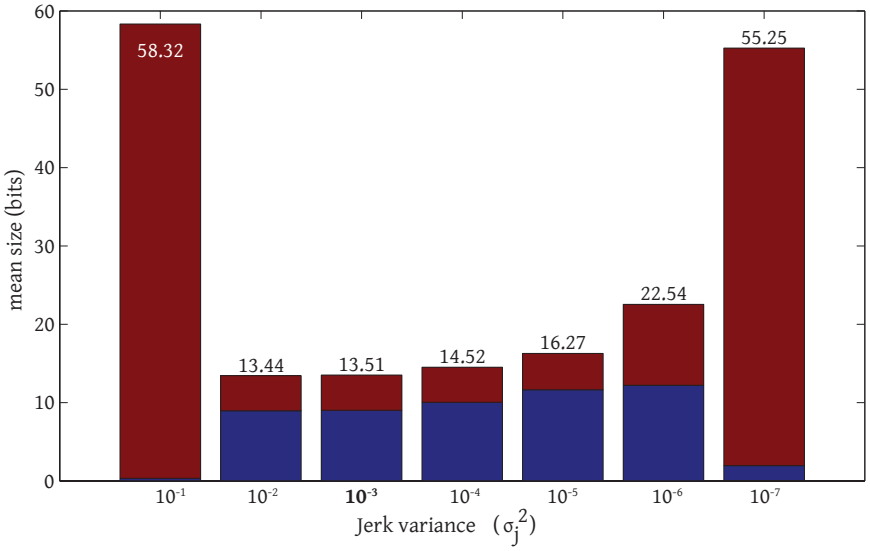
datagram transmission in addition to the state estimate innovations (with $\tau = 5$s). Fig. 3.8 shows the results from a mission during this experiment, which used the Kalman filter state observer coupled with the Gaussian distribution for encoding the delta transmissions. Using the lowest data rate available with the phase-shift-keying (PSK) modulation on the Micro-Modem ("rate 1"), only 12.6% of the available data throughput during this mission was used to transmit the state observer innovations (including addressing, duplicate rejection, and byte padding overhead). Thus, using this technique, it is possible to consider telemetry of vehicle positions as a reasonably small amount of overhead on the underwater communications system, rather than its primary purpose as has historically been the case.

3.6 Robustness

The design of the state observers is intentionally general to reduce the number of parameters to be "tweaked" or "tuned" and thus improve the robustness (for a broad range of maneuvering vehicles). However, there are two major parameters to determine that have values which are not clear *a priori*: the range of included delta values for the arithmetic entropy encoder (i.e. $[\mathbf{dy}_{min}, \mathbf{dy}_{max}]$) and, when using the Kalman filter estimator, the process noise (embodied in its variance: σ_j^2) which is used to model all vehicle manuevers.



(a) Performance over various values of the encoding range $[\mathbf{dy}_{min}, \mathbf{dy}_{max})$.



(b) Performance dependence on the variance of the process noise (which is how vehicle maneuvers are modeled)

Figure 3.9: Using the GLINT10 dataset, the performance (in mean size of messages) of this system over a range of parameter values. The contributions to the mean size from the delta and full transmission are shown separately. The values in bold were used for the rest of the analysis in this chapter and are also given in Table 3.1.

The first parameter ($[\mathbf{dy}_{min}, \mathbf{dy}_{max}]$) is illustrated in Fig. 3.9a, which shows the tradeoff in choosing these bounds. Too small, and the state observer consistently exceeds the bounds, and a full transmission must be resent to reinitialize both *sender* and *receiver* states. However, there is also a small price to pay for making them too large, which is the small amount of probability mass required for each discrete value with the range of $\mathbf{dy}_{min}$ to $\mathbf{dy}_{max}$, taking away mass from all the remaining values, making the more probable values (e.g. 1, 0, -1) slightly more costly to encode. However, this cost is small compared to having to reinitialize the states by sending full transmissions frequently. Thus, as is clear from Fig. 3.9a, it is preferable to err on the side of too loose bounds than too tight.

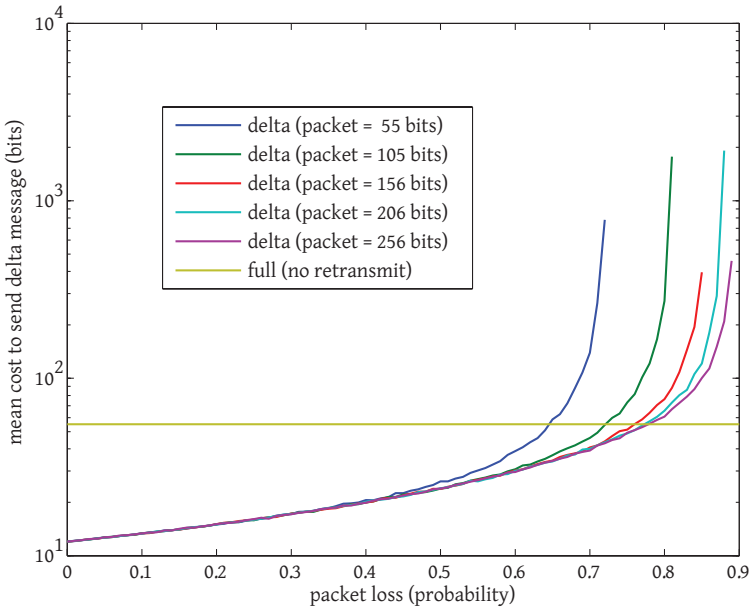
The second parameter (σ_j^2) was originally chosen using a subset of the experimental data from GLINT10 to determine a reasonable order-of-magnitude value ($\sigma_j^2 = 10^{-3}$). Figure 3.9b shows the overall performance (message size in bits, including full transmissions and delta transmissions) for a wide range of process noise values. This figure shows that the algorithm is robust over about four order of magnitude from 10^{-2} to 10^{-5} (too low or too high and the filter perpetually fails to track). It is also worth remembering that the fixed speed tracker does not require this parameterization and is applicable to a large number of classes of AUVs found in the field today.

Finally, by design, the tradeoff for inaccuracy in determining these parameters is *not* accuracy in the received telemetry, but rather cost (in bits) of sending these data. This means that the *receiver* is never uncertain about the quality of the data that it has received.

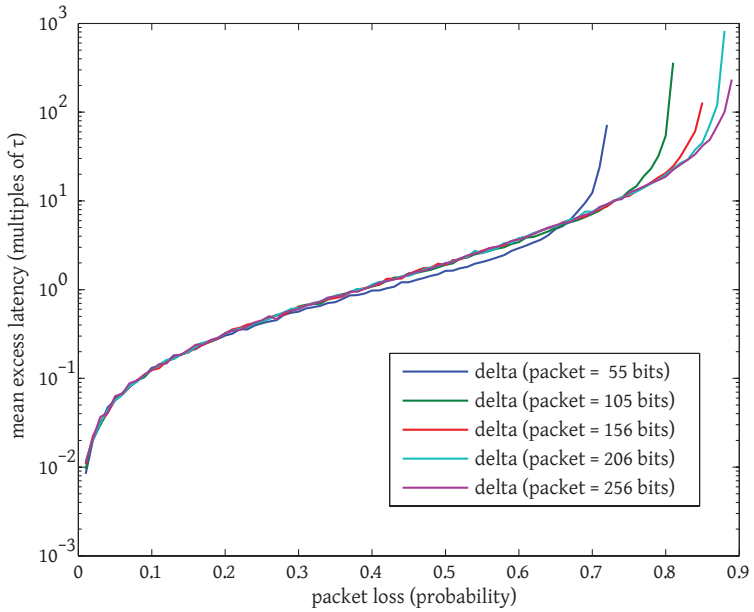
3.7 Performance comparison to traditional approach

As mentioned in Section 3.2, the delta technique developed in this chapter provides an evenly sampled historical timeseries (regardless of packet loss) by retransmitting dropped packets (using ARQ). Such an evenly sampled history is useful for in-situ analysis of instrument data attached to these position messages, or performance evaluation. Since ship time is highly expensive, it is valuable to do as much data analysis and debugging as possible while the vehicle is underway (rather than waiting until the end of the mission).

However, there are times when only the latest position of the vehicle is desired. In this case, does this state observation technique still provide any benefit? Using the experimental results from MBAT12 (Table 3.2), the simulated mean cost to send a delta message was plotted against the packet loss (modeled as an independent Bernoulli process) as Fig.



(a) Mean cost (in bits) to send a delta message compared to a full message system with no retransmits.



(b) Increased latency (over full system with no transmits) for a variety of packet sizes.

Figure 3.10: Performance comparison of traditional "newest when possible" system (full transmission with no retransmits) with the delta state observer technique *in the case when a full historical timeseries is unneeded.*

3.10a. This metric was computed for a variety of packet sizes, ranging from the full transmission size (55 bits) to 32 bytes (256 bits), a commonly used acoustic modem maximum transmission unit (MTU). In addition, a curve representing the traditional system using full transmissions with no retransmission upon loss is provided. Thus, depending on the packet size, this delta-based technique is cheaper (uses fewer bits on average) than a traditional full position system up to packet losses of 65% or higher.

On the other hand, the delta system requires that older innovations be correctly received before newer innovations can be used (otherwise the *sender* and *receiver* state observers do not share the same state). This leads to increased latency over the full position system that increases monotonically with packet loss. This excess latency is shown in Fig. 3.10b. The tolerance for excess latency versus increased bit cost is mission specific. A hybrid system that switches between this delta technique (for lower packet loss links) and a traditional full system (for higher packet loss links) may be necessary to reach the desired tradeoff of latency versus throughput for a given mission.

3.8 Conclusion

A technique for transmitting the position vector of an AUV at relatively high rates at very low cost (in bits) was developed and demonstrated on two experimental data sets and implemented in the field, leading to mean compression ratios as high as 93%. Such a system is suitable for use with the currently available acoustic modems that provide only on the order of 10^1 to 10^2 bytes per minute of throughput. This chapter shows that continuous telemetry of vehicle position is possible while staying well within the abilities of these modems, allowing for additional mission-specific messaging to take place as well.

4 Disruptive Technique: autonomous navigation approaches to improve the physical link

4.1 Introduction

Autonomous underwater vehicles (AUVs) are increasingly used in a variety of oceanographic and naval tasks, such as oceanographic surveys, target detection and classification, and seafloor imaging. Many of these tasks make use of acoustics as either a remote sensing tool or as a communications signal carrier. Propagation of acoustic signals are highly dependent on the acoustic environment: the sea surface, water column, and sea floor. Thus, accurate understanding of this environment may be exploited for improving the performance of sonars or acoustic modems, similarly to the optimization strategies used in the past by human platform and sonar operators. Computational acoustic modeling uses numerical methods for approximately solving the wave equation in real environments whose parameters are too complex to handle analytically. This chapter presents the Generic Robotic Acoustic Modeling (GRAM) concept for interfacing the artificial intelligence captaining the AUV to one or more existing acoustic models, such as the Acoustics Toolbox [78] or OASES [79] models. We present the use of the GRAM concept and software for two domains: improved acoustic communications in shallow water and deep sea localization and tracking of a near surface acoustic contact. A schematic of this concept is given in Fig. 4.1.

4.1.1 MOOS-IvP Autonomy Software

The MOOS-IvP Autonomy software presented in [27] provides two main components that form the underpinnings for this work:

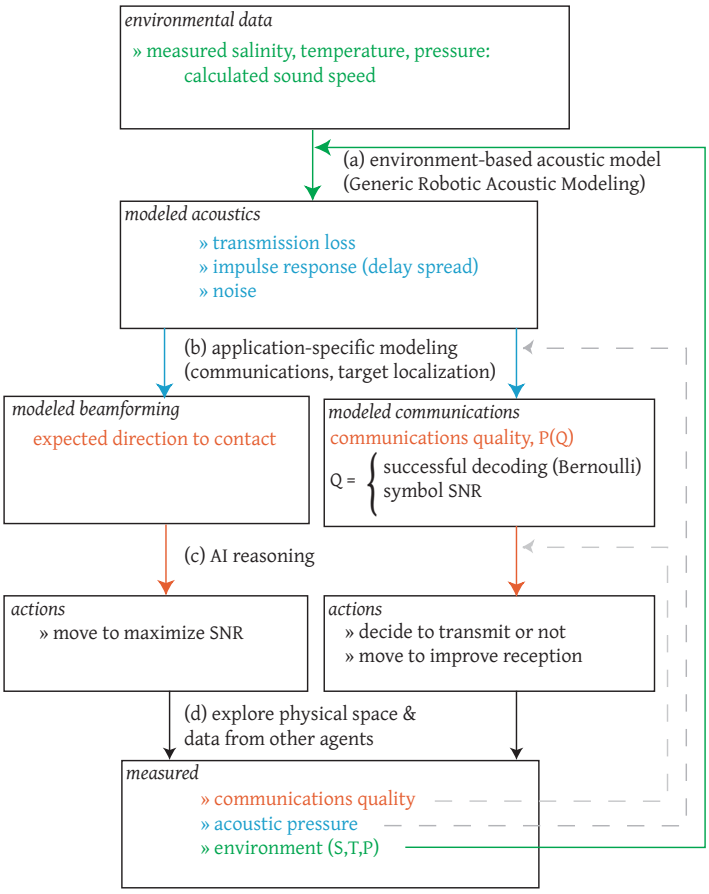


Figure 4.1: Block diagram overview of the model-based adaptivity framework presented in this work. The right path focuses on improving acoustic communications (demonstrated in section 4.3) whereas the left path models sonar performance for a target tracking application (section 4.4.2). Both applications (and others) can be run in parallel due to the RPC design of GRAM (see section 4.2). The dotted arrows represent areas of feedback not presented in this work that could be exploited for better performance.

- MOOS publish/subscribe middleware: provides interprocess communications (IPC) over TCP in a publish/subscribe manner via a central "bulletin-board" process which contains a database of the latest sample of each data type. This enables the robotic software to be split into many discrete subsystems that can be developed and debugged independently.
- The Interval Programming (IvP) Helm multi-objective decision engine: The IvP Helm provides an interface for a collection of behaviors to produce functions of utility (which are soft decisions and can be multi-modal) over one or more domains (usually heading, speed, and depth of the AUV). At a set frequency (typically one Hertz), the IvP Helm solves all the behaviors' functions for a single hard decision that is passed to the vehicle control system to execute. Unlike traditional behavior-based control such as that championed by Brooks [80], the IvP Helm behaviors have state and therefore can run models and act on collected data, as is done in the behaviors presented in this work.

MOOS-IvP has been used extensively in marine vehicle autonomy research, such as cooperative search tasks [30, 81, 82] and adaptive oceanographic sensing [83, 84].

4.1.2 Computational Acoustic Models

This work builds on several computational models that use differing techniques for approximately solving the wave equation in complex ocean environments. The theoretical treatment of the underlying approximations used for all these models is discussed in depth in [85]:

- Acoustics Toolbox: a collection of computational models and related tools. The two that are integrated into GRAM are:
  - BELLHOP [26]: a model that generates ray trajectories, transmission loss (using Gaussian beam tracing), and eigenray travel time outputs using the ray-based high-frequency approximation of acoustic propagation. This is the model used for the case studies in this work due to the rapid computation of ray tracing over other approaches. On embedded systems such as AUVs, fast computation is often more important than high fidelity due to power constraints and available computational resources. See Fig. 4.3 for an illustration of how much power can be saved by having a low duty cycle on the modeling system.

  - KRAKEN [86]: a normal modes based model. KRAKEN treats the ocean waveguide as a summation of modes and is thus best suited for more accurate modeling of somewhat lower frequency (fewer modes) problems than BELLHOP is suited for.

- OASES: a model that relies on wavenumber integration to solve problems involving propagation in one or more horizontally stratified layers, making it especially suited for seismo-acoustics problems.

Other acoustic models can be incorporated into the framework presented in section 4.2. The Acoustics Toolbox and OASES were chosen for their maturity, open source availability, and performance (both are written in Fortran, which is compiled to the platform's native machine code).

4.2 GRAM: Low power in-situ Generalized Acoustic Modeling

The Generic Robotic Acoustic Model (GRAM) provides a set of tools implemented in C++ for performing *in-situ* modeling of the acoustic environment for use by autonomous decision making (such as the IvP Helm used in the experimental studies in sections 4.3 and 4.4.2). A graphical structure diagram of GRAM is provided in Fig. 4.2, showing also the suggested division of hardware systems based on the realtime and performance (and thereby power) requirements. GRAM is designed with several considerations that make it more suited for running on underwater embedded robotic systems than directly calling the underlying acoustic modeling code:

- asynchronous remote procedure call (RPC) design
- runtime reconfigurable
- abstracted interface

4.2.1 RPC design

GRAM is designed such that each "consumer" (an application or module that needs the result of an acoustic model) makes asynchronous requests independently of the other

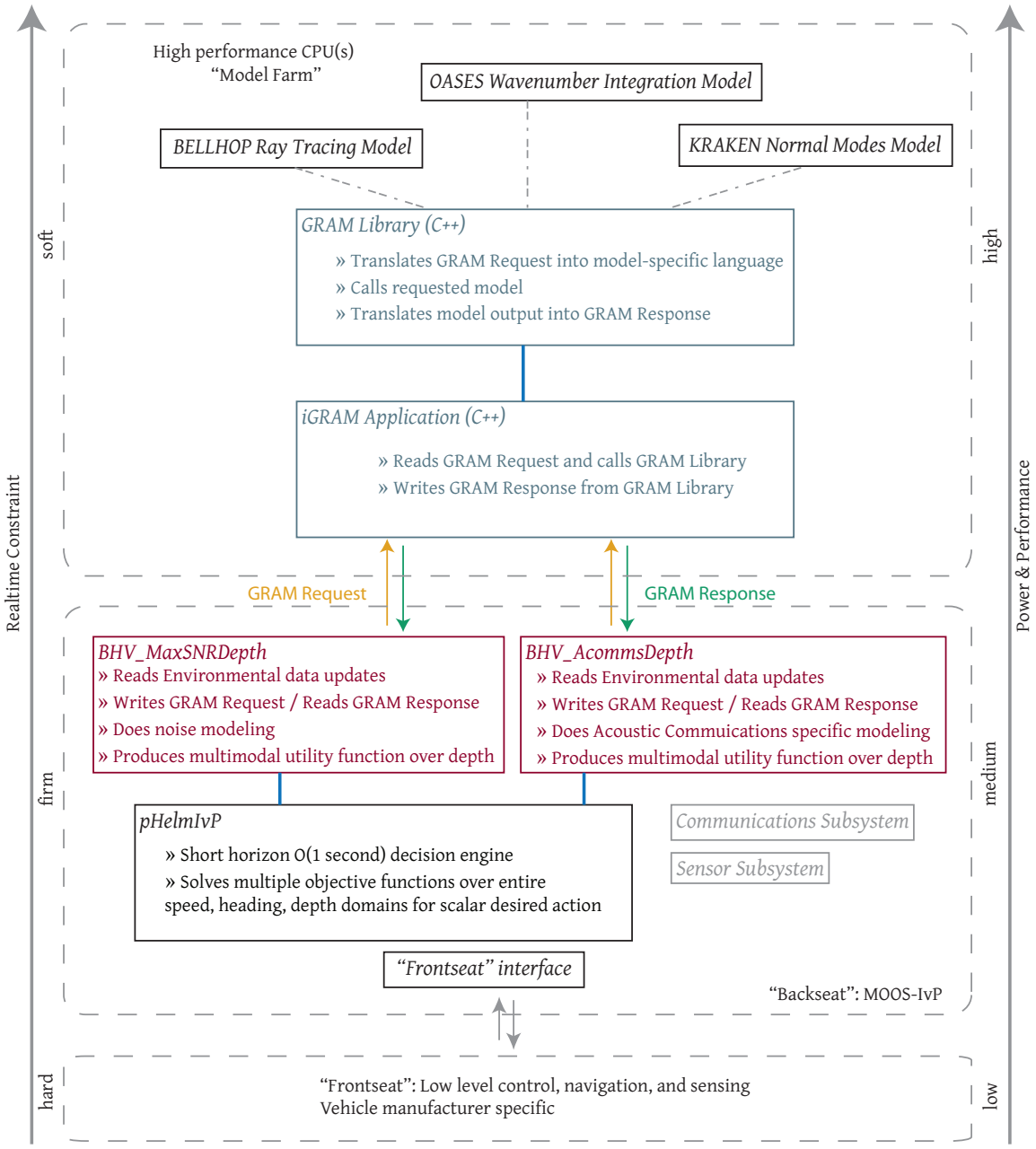


Figure 4.2: A structure diagram of an autonomous underwater vehicle using the GRAM tools and the MOOS-IvP autonomy middleware. See [87] for an overview of the "frontseat"-"backseat" paradigm (Oliveira, et al. also use a similar separation of hardware in [88]). The present work adds a third physical computing layer, the model farm, which can be thought of a one step further removed in terms of realtime requirements from the "backseat". While this separation is not required, it can be used to save power, as illustrated in Fig. 4.3.

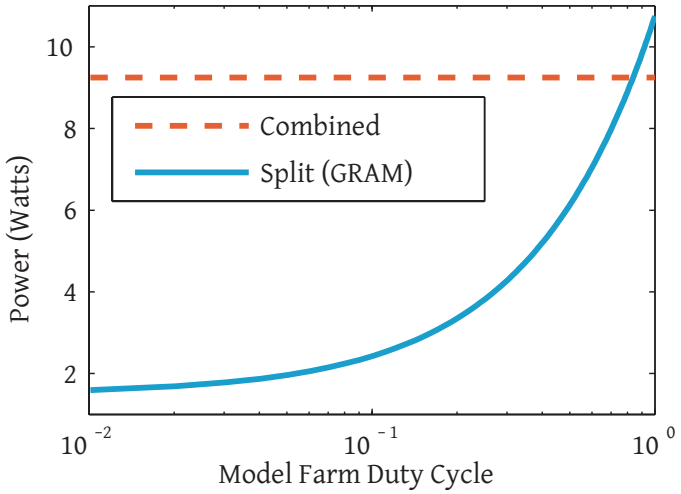


Figure 4.3: Comparison of computer board power usage for *split* low-power CPU (the "backseat") and high-performance CPU ("model farm") versus *combined* on a single high-performance board. Low-power board shown is the Eurotech Titan (Intel 520 MHz PXA270 XScale processor) [89]; high-performance board is the Advantech PCM-3363 (Intel 1.8 GHz Atom D525 Dual Core processor) [90]. The duty cycle is the fraction of time the acoustic models are being run, with the assumption the model farm can be shut off in the *split* case the rest of the time. Except when the models are being run near constantly, the *split* system (illustrated in Fig. 4.2) saves power, which is especially useful in longer slower missions where hotel power usage dominates propulsion power usage [35].

consumers. Each request is processed by the GRAM tools and a response is sent back containing the results of the model calculation. The requests and responses can be transmitted over a transport of choice (e.g. TCP, shared memory, RS-232); in the results presented in this chapter we use the MOOS middleware TCP-based transport. This design allows for the separation of soft realtime modeling computations from other firm or hard realtime systems ("backseat" autonomous decision making and "frontseat" low-level control and actuation), as consumers can continue to work using their most recent available data until the new model calculation is complete. This separation also allows for the hardware performing the modeling to be put into a low power state, saving significant amounts of energy when the required duty cycle of the modeling farm is somewhat less than one hundred percent. See Fig. 4.3 for a comparison of power usage for a split model/realtime system (such as diagrammed in Fig. 4.2) versus a more traditional single CPU board design. The specifics of the mission and dynamics of the environment can change the required duty cycle dramatically.

4.2.2 Runtime reconfigurable

Each request for a model calculation can contain any or all of the parameters of the acoustic environment. This allows for meshing fixed parameter values with real time updates of the environment available from on-board sensors (e.g. Conductivity-Temperature-Depth (CTD) sensor) and/or transmitted from a remote source (e.g. another AUV, satellite, surface craft).

4.2.3 Abstracted interface

GRAM uses an extensible object-oriented representation of the acoustic environment (written in a language-neutral Protocol Buffers representation [34]), which is translated into the specific input format required by the desired acoustic model. Many of these models use arcane input formats that are intolerant of syntactical mistakes. Given the costs of AUVs (hundreds of thousands of US dollars) and the operational costs of AUV experiments (thousands of dollars per day), accepted software quality practices that emphasize saving programming time and increasing reliability are of utmost concern to AUV researchers. The abstracted GRAM interfaces provides these quality checks that the native interface to the Acoustics Toolbox and OASES do not:

- compile-time type checking
- compile-time bounds checking on enumeration fields
- run-time bounds checking on numeric fields

4.3 GLINT10 Shallow water experiment

4.3.1 Acoustic Communications

Most autonomous vehicle tasks share a common element: collection of data that are only useful once they reach a human operator or a collaborating robot. Transmission of such data can easily be accomplished once the mission is completed and the vehicle is recovered. However, the time sensitivity of the data may preclude waiting hours or days before its recovery. Furthermore, offloading data during a mission guards against complete loss in the event of catastrophic vehicle failures. Finally, collaboration between two or more robots requires communication during the mission.

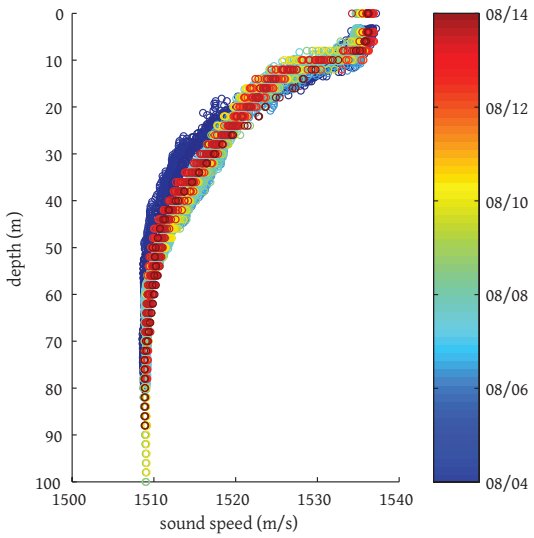


Figure 4.4: The 111 sound speed profiles calculated using the Chen/Millero equation [17] from the temperature, salinity, and pressure data collected by the AUV Unicorn throughout the GLINT10 experiment starting on 08/04/2010. Profiles were collected by the AUV performing one or more "yoyo" maneuvers in depth and are averaged over thirty minute windows. Initially the stratification is more pronounced before a storm early in the experiment caused some mixing of surface and bottom waters.

To this end, wireless acoustic communication systems have been developed to allow for subsea telemetry. Sound is used as a carrier rather than the more traditional radio or light waves due to the very short electromagnetic skin depth of sea water in all but very low frequencies (which require large antennas to efficiently generate). Acoustic waves are far from an ideal digital signal carrier, though. Attenuation due to absorption which increases with frequency puts a practical upper bound on the usable carrier frequencies and consequently available bandwidth. Multipath due to surface and bottom reflections as well as refraction caused by the often highly stratified vertical sound speed profile leads to intersymbol interference and thereby high packet loss. The low speed of sound in water (nominally 1500 m/s) leads to non-negligible Doppler effects. Stojanovic [91], Preisig [12], and Baggeroer [10] cover all these issues and how they influence the design of an acoustic modem physical layer.

Table 4.1: GLINT10 Experiment Parameters

Geometric	
Source (Buoy) depth	30 m
Receiver (AUV) depth	variable (primarily 0-60 m)
Source (Buoy) speed	0.03 m/s (σ = 0.02 m/s)
Receiver (AUV) speed	1.47 m/s (σ = 0.14 m/s)
Source beam pattern	azimuthally omni-directional; polar is 5dB reduced towards surface and bottom.
Environmental	
Sea state (Beaufort)	1-3
Sea floor depth	111 m (σ = 4.7 m)
Signaling[a]	
Source Level	190 dB re 1 μPa at 1 m
Frequency (carrier)	25120 Hz
Bandwidth	4160 Hz
Modulation	Frequency-hopping Frequency Shift Keying (FH-FSK)
Frequency hops	7
Symbol bin width	320 Hz
Symbol duration	6.25 ms
Symbol clearing time	6 symbols = 37.5 ms
Error correction coding	rate 1/2 convolutional code
Symbols / transmission	576
BHV_AcommsDepth	
Acoustic model window τ_a	120 s
Environmental window τ_e	1800 s

[a] See [92] for further details on coding and modulation and [93] for the packet specification.

4.3.2 Experimental Setup

The GLINT10 experiment took place in the shallow water (nominally 110 meters deep) off Porto Santo Stefano, GR, Italy in the Tyrrhenian Sea within ten kilometers of the experiment datum at $42°27'24''$ N, $10°52'30''$ E. The acoustic environment (see Fig. 4.4) was marked by a warm surface layer (corresponding to a high speed of sound) followed by a sharp thermocline and cooler water. From the perspective of this work, the experiment has two goals:

1. Collect statistics on acoustic modem performance as a function of range and depth for use in validating the utility of the adaptive behaviors and for developing feedback learning for future missions. On previous experiments in a similar environment, qualitative observations had been made about much improved modem performance at deeper depths. This experiment hopes to validate and quantify this observation.

2. Demonstrate an adaptive behavior `BHV_AcommsDepth` for tracking the modeled transmission loss minimum calculated using the sound speed profile obtained by the AUV using the thermocline detection and tracking behaviors developed as part of [84].

Both of these were performed using a single AUV ("Unicorn") and a communications buoy ("Buoy") fixed at 30 meters depth. Both assets were equipped with the WHOI acoustic Micro-Modem in the "C" frequency band using modulation rate "0": see Table 4.1 for the corresponding acoustical and modulation parameters. This choice of modem hardware was driven by availability and convenience; the adaptive behaviors in this work are based on fundamental acoustics that affect the performance of all acoustic modems. Different modulation schemes and adaptive equalization will cause improved results in certain environments, but they cannot remedy the underlying signal's quality. The behaviors developed here work to improve the underlying signal which should in turn improve modem performance regardless of the modem chosen. This work is complementary to that on the physical layer such as [94] and [95], and operates on a level above the traditional networking "stack" in a new layer called the "platform" layer as shown in Fig. 4.5. The timescales involved are widely different as well: `BHV_AcommsDepth` aims to improve communications taking into account environmental changes on the order of hours whereas physical layer communications work attempts to account for changes on the order of milliseconds.

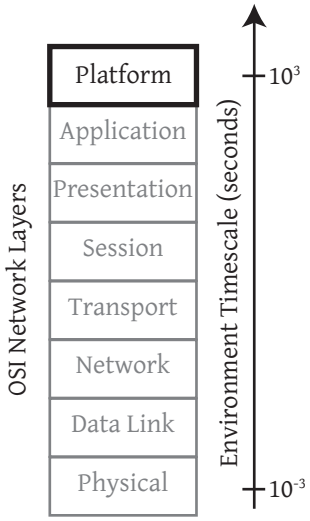


Figure 4.5: The work in this section can be thought of as comprising a new layer above the traditional seven-layer Open Systems Initiative (OSI) networking stack [24]. Another way of thinking about this compared to other work in the networking system is the timescale of environmental changes that are focused on; that is, the physical layer is concerned with symbol-to-symbol variation in the channel (milliseconds), whereas this work tackles hourly or longer scale variation in the environment.

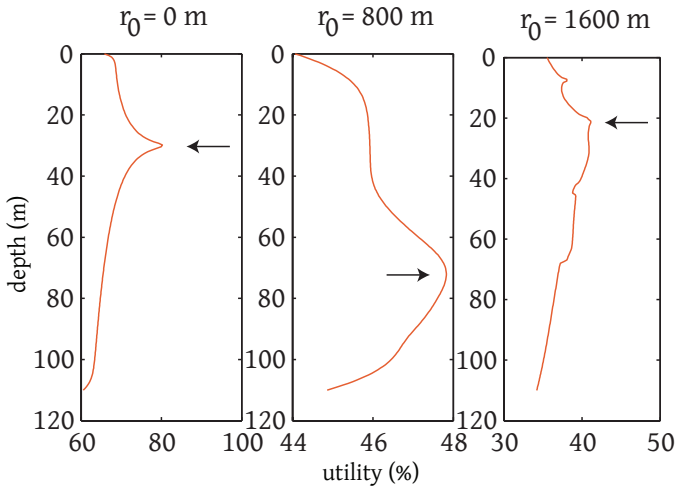
4.3.3 BHV_AcommsDepth: Autonomy Behavior for maximizing acoustic modem performance over vehicle depth

This IvP Helm behavior was written to arbitrate over the depth decision domain with the goal of improving acoustic communications reception between an AUV and a fixed (or slowly moving) receiver. It makes use of the modeled acoustics to form a soft decision based on the expected best communications throughput.

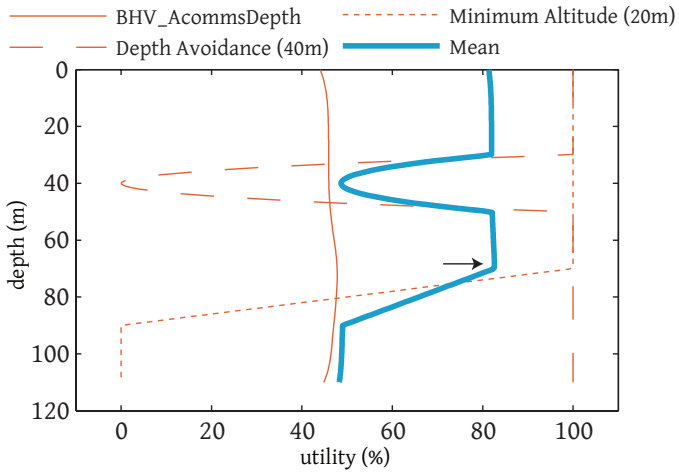
Using the newest available sound speed data `BHV_AcommsDepth` makes a request to GRAM for a transmission loss calculation for the range window Δr where

$$\Delta r = |\mathbf{v}_0| cos(\Theta) \tau_a \tag{4.1}$$

formed from the vehicle's current position r_0 for a predefined time horizon τ_a based on its current instantaneous velocity $\mathbf{v}_0$ and angle Θ with respect to the Buoy (where $\Theta = 0$ is defined as when the AUV's bow is pointing directly away from the buoy's position). This request is made at least every τ_a seconds so that the modeled region in range-depth space (with respect to the receiver) always contains the actual region that the vehicle currently occupies.



(a) Three example plots of the BHV_AcommsDepth objective function $O(d)$ (Eq. 4.3) using the ray trace shown in Fig. 4.11, Θ = 0, $H_{max} = 108$ dB, and remaining parameters as given in Table 4.1. In the absence of other behaviors, the vehicle's decision for depth is marked by an arrow.



(b) One possible interaction of BHV_AcommsDepth with other behaviors. The objective function for r_0 = 800m is shown along with two other behaviors (one for avoiding a hypothetical obstacle at 40m and a safety behavior to stay 20m off the sea floor). Again, the decision is given by an arrow.

Figure 4.6: Example BHV_AcommsDepth objective functions without (a) and with (b) concurrent depth-domain behaviors.

The sound speed is assumed homogeneous in range (i.e. the Northings/Eastings plane) given the infeasibility of sampling all points in the vehicle's future path. `BHV_AcommsDepth` then averages the modeled intensity over the range window to calculate the modeled transmission loss

$$\overline{H(d)} = -10 \log_{10} \left(\left[\sum_{r=r_0}^{r_0+\Delta r} \left| \frac{P(d,r)}{P_0} \right|^2 \right] / \Delta r \right) \tag{4.2}$$

where $P(d, r)$ is the acoustic pressure in AUV depth (d) and range (r), and P_0 is the pressure at the source. Using this averaged transmission loss, `BHV_AcommsDepth` seeks to maximize the expected acoustic signal level via an objective function ($O(d)$) over AUV depth

$$O(d) = \max \left(1 - \frac{\overline{H(d)}}{H_{max}}, 0 \right) \tag{4.3}$$

where H_{max} is a normalization constant representing the transmission loss threshold above which the vehicle is assigned no utility to be at that depth. This can either be the maximum H for a given window (as was used in the GLINT10 trial) or a global maximum determined based on the received signal statistics. For the data collected during this experiment, a H_{max} of 108 dB would be a reasonable choice as less than 1% of messages were received at measured transmission losses higher than this. This value was used in the objective functions plotted in Fig. 4.6. Another more aggressive choice could be the minimum probability of error decision rule for the binary hypothesis test between a packet being received successfully or a packet being dropped. Based on the GLINT10 data, this criterion would lead to $H_{max} = 94$ dB.

The completed objective function $O(d)$ is then passed to the IvP Helm to solve along with the other behaviors for the heading and speed domains. An example of how $O(d)$ interacts with other objective functions that also operate in the depth decision domain is illustrated in Fig. 4.6b.

4.3.4 Mission profile

This experiment was designed to test the effectiveness of model-based adaptivity on a single AUV without expensive equipment such as an upward-facing Acoustic Doppler Current Profiler (ADCP) which could measure sea-surface conditions. The required equipment for this experiment was only a CTD and enough computational power to run the MOOS-IvP and GRAM combined autonomy and modeling system.

Each mission was run with a basic straight-line "racetrack" in the Northings/Eastings local UTM Cartesian plane. The interesting part of the mission happens in depth, with the goal that the `BHV_AcommsDepth` would run simultaneously with other behaviors arbitrating over the Northings/Eastings plane (via a chosen desired speed and heading). In addition, it is expected that other behaviors will be added to influence the chosen depth of the vehicle, and the multi-objective solver of the IvP Helm resolving these multiple functions over the vehicle's utility for a given depth. Each mission followed this plan:

1. Gather a CTD profile by making sinusoidal excursions in depth, starting with close to the full water column and narrowing down to adapt to the thermocline region where the most changes in temperature (and by extension sound speed in this environment) are occuring. This thermocline adaptivity is described in [84].

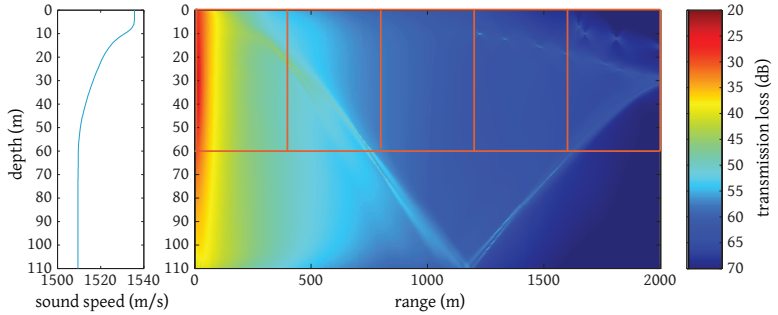
2. These CTD data are passed to `BHV_AcommsDepth` which generates an objective function for the IvP Helm to solve along with the other behaviors for the heading and speed domains. The BELLHOP model was used with GRAM for this work due to the high frequency (25 kHz) of the acoustic carrier. For this experiment, no behaviors besides `BHV_AcommsDepth` were running that produced an objective function over depth, so that we could evaluate the performance of `BHV_AcommsDepth` alone.

3. The vehicle moves to the optimal depth determined by `BHV_AcommsDepth` and at least every τ_a seconds reruns the GRAM model from step 2) taking into account changes in heading and speed. Less often (at the environmental interval τ_e) a reset to step 1) is made to remeasure the sound speed profile for any changes in the environment. τ_e should be much less than the timescale of changes to the environment. Since the shallow water Mediterranean sound speed profile changes significantly on the order of one day timescales, τ_e was chosen to about a two orders of magnitude below that, or 1800 seconds.

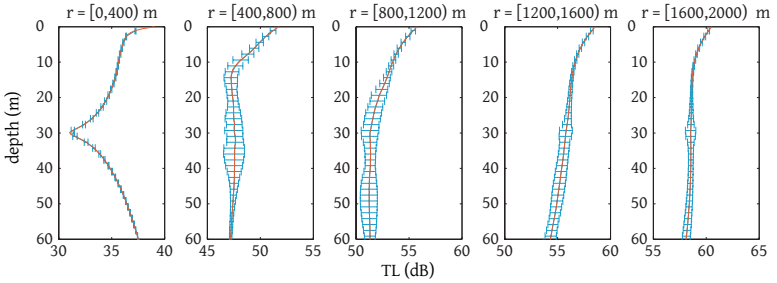
4.3.5 Results

Modeling and communications statistics

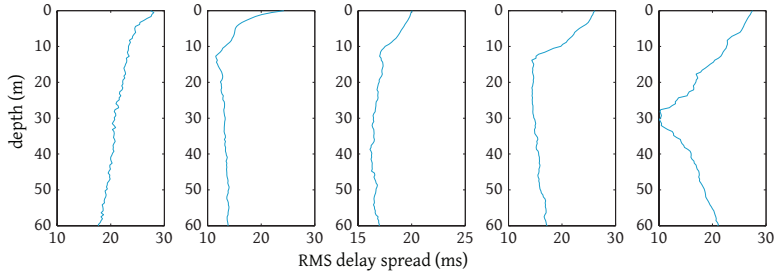
The modeling and statistical results of the GLINT10 experiment are summarized in Fig. 4.7, along with Fig. 4.8 which shows the modeled noise level. Fig. 4.7 (a) shows a BELLHOP ray tracing model for the average sound speed profile of the entire experiment to give a general overview of the acoustic environment from the perspective of the Buoy as source



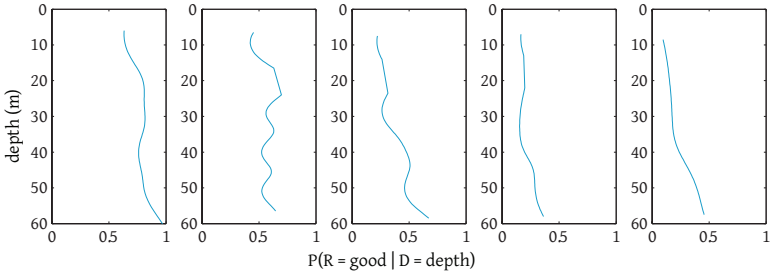
(a) Mean sound speed profile (left) used to compute representative transmission loss model (right). Boxes represent regions plotted in parts (b), (c), and (d) of this figure.



(b) Mean modeled transmission loss (H) for five range bins (each bin representing 4.4 minutes (τ_a) of averaging for a vehicle moving at 1.5 m/s.) The error bars represent the standard deviation σ for the H computed using all the sound speed profiles collected and displayed in Fig. 4.4.



(c) Modeled root-mean-square delay spread from the mean sound speed profile shown in part (a). Due to the 37.5 ms of clearing time used by the incoherent FH-FSK modulation of the modem, we expect that this delay spread will have little effect on the successful receipt of datagrams.



(d) Estimated conditional probability of successful receipt (R=good) plotted over the conditioning depth. Depths where the AUV was present less than 1% of the transmissions are excluded.

Figure 4.7: Modeled (using GRAM and BELLHOP) and measured data from the GLINT10 experiment. Note that at longer ranges ($r > 800$ m), there is a inverse correlation between the modeled transmission loss (b) and the estimated probability of successful receipt (d), as expected.

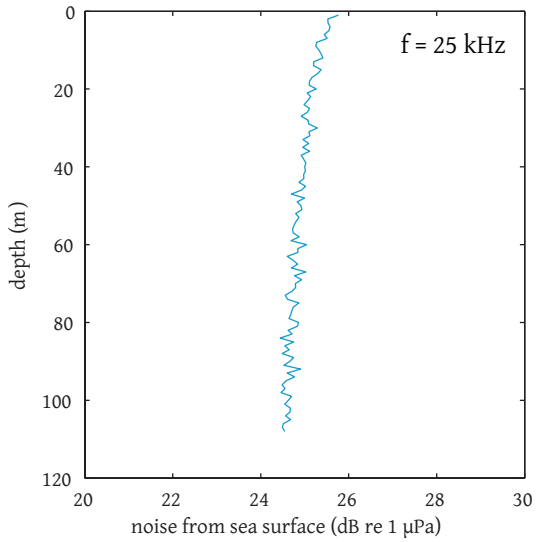


Figure 4.8: Modeled sea surface noise using for Beaufort sea state 2 (observed for most of the GLINT10 experiment) using [96] and OASES. Note that the noise profile in depth is nearly constant (about 1 dB of deviation), justifying the use of transmission loss H as a proxy for SNR in this analysis.

and show the significant downward refraction due to the thermocline from 10 to 30 meters depth. Note that this average erases some of the small scale features present in each actual profile (taken every τ_e seconds and plotted in Fig. 4.4) that the vehicle actually uses for its modeling.

For display purposes, the remaining plots are split into 400 meter range bins where all data shown within are averaged in range over these bins. Only the upper 60 meters of the water column are shown as this is where the AUV spent most of its time. Fig. 4.7 (b) gives the modeled transmission loss ($H_n(d)$) using the profiles (instantiations) taken by the vehicle and then averaged in intensity over the instantiations as well as within each range bin, that is

$$H_n(d) = -10 \log_{10} \left(\frac{1}{i_{max}} \sum_{i=0}^{i_{max}} 10^{-H(d,i)/10} \right) \tag{4.4}$$

where $H(d, i)$ is given by equation 4.2 for each instantiation of the sound speed profile i. In this case $\Delta r = 400$ and $r_0 = \Delta r(n-1)$ where $n = [1, 5]$ corresponds to each of the five displayed range bins. The error bars (standard deviation of the intensity over all the sound speed profile instantiations) show the sensitivity of various parts of the transmission loss plot to changes in the sound speed profile. The regions of higher standard

deviation are caused by caustics moving location due to small changes in the sound speed profile. In general, deeper depths have lower H in this environment.

Fig. 4.7 (c) gives the modeled root-mean-square delay spread τ_{RMS} calculated using the experiment mean sound speed profile (SSP) (Fig. 4.7 (a)) where

$$\tau_{RMS} = \sqrt{\frac{\int_0^\infty (\tau - \bar{\tau})^2 A(\tau) d\tau}{\int_0^\infty A(\tau) d\tau}} \tag{4.5}$$

and $A(\tau)$ is the intensity of the arrivals where $\tau = 0$ is the first arrival. In general, deeper water has a lower delay spread, leading to potentially reduced intersymbol interference.

The experimental data from all the transmissions ($N = 3350$) sent by the AUV Unicorn to the Buoy were used to compare against the modeled data. The data were split into two groups using a basic division interesting to AUV roboticists: was the message received correctly (R=good) or not (R=bad)? A message was considered to be received correctly if it had no errors after decoding (as verified by a cyclic redundancy check in the WHOI Micro-Modem). Any other problem with the message meant that it was considered not to be received correctly. A probability distribution of the vehicle's depth ($P(D)$) throughout the experiment was estimated from the data using an Epanechnikov kernel smoothing estimate. Also a conditional probability of depth given that the messages were received ($P(D|R = good)$) was computed in a similar manner. Of greater interest is the posterior probability ($P(R = good|D = d)$) which was computed for all depths d using the prior $P(D)$ and Bayes' rule:

$$P(R = good|D = d) = \frac{P(D = d|R = good)P(R = good)}{P(D = d)} \tag{4.6}$$

This posterior was plotted in Fig. 4.7 (d). For ranges greater than 800 meters (range bins 3-5), the data show a strong depth dependency, with modem performance doubling from twenty meters to fifty meters in the farthest range bin. At short ranges ($r < 800$ m), large percentages ($P(R = good|D = depth) > 0.75$) of the messages are received which is likely due to the strong direct arrival (first bottom bounce occurs at $r \simeq 1100$ meters) and generally high signal strength. As would be expected given that modem performance depends on signal strength, this is the inverse of the modeled transmission loss in Fig. 4.7(b), which shows a depth dependency in the same bins (decreasing H with increasing depth). These data also do not show a strong correlation with the modeled delay spread τ_{RMS}. This may be due to the fact that the real delay spread is significantly influenced by the sea surface, which was naively modeled using a flat pressure release surface in this

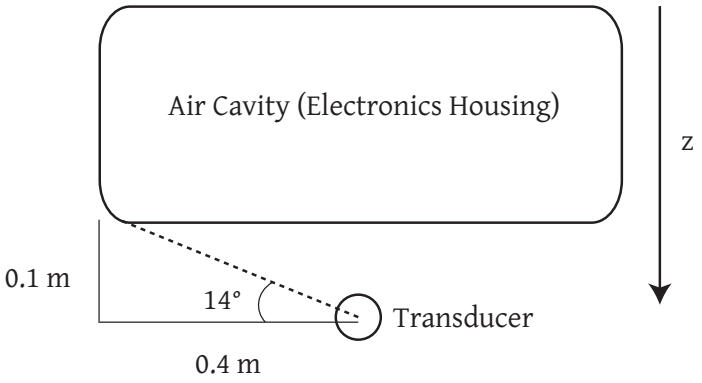


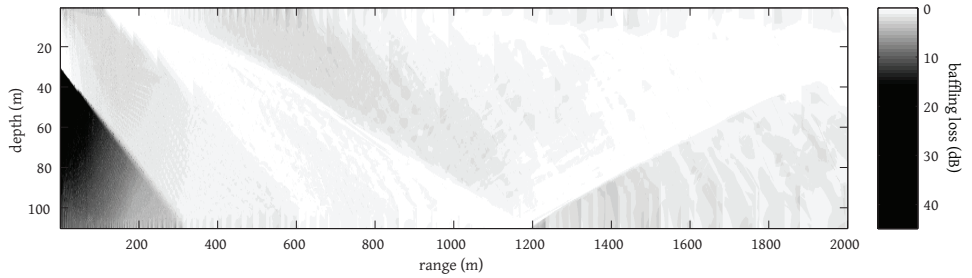
Figure 4.9: Side view of the payload electronics housing (modeled as an acoustic pressure release surface) and the modem transducer.

work due to the lack of onboard knowledge about the sea state. Furthermore, the symbol clearing time of the FH-FSK modulation employed is 37.5 ms, significantly longer than the delay spreads modeled here (the root-mean-square values are in the 10-20 ms range).

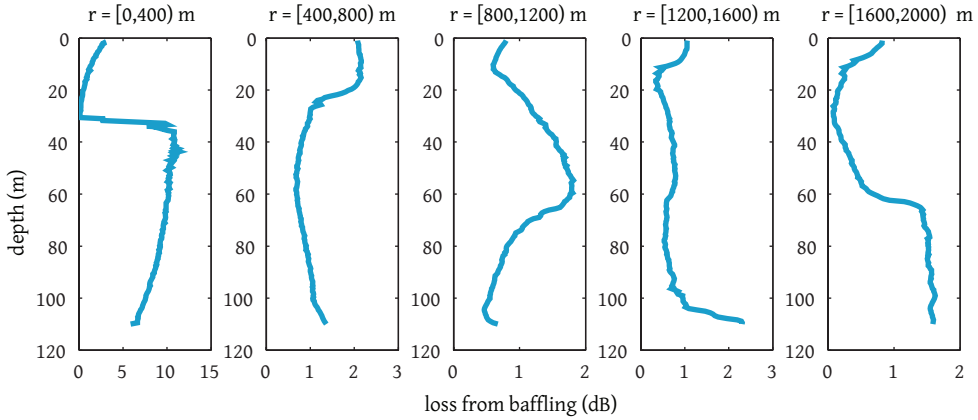
Compensation for baffling

The physical layout of the AUV Unicorn involves an air-filled electronics housing mounted directly above the acoustic modem transducer as shown in Fig. 4.9. *A priori* it seems reasonable that the subsequent (undesired) baffling caused by this housing may have an effect on the received signal strength. Thus, this effect was modeled by treating the housing as a pressure-release surface. The model in Fig. 4.7a was rerun, removing the rays that will not reach the transducer due to scattering by the electronics housing. The difference from the result without baffling (Fig. 4.7a) and this new result including the electronics housing is shown as Fig. 4.10.

The result is a significant increase in transmission loss when the AUV is directly under the Buoy and for a "shadow zone" reaching for few hundred meters from the Buoy at one hundred meters depth. Outside the region, the transmission loss is only mildly affected by this effect, and thus it is unlikely to play a major role in the performance of the communications system in this largely horizontal configuration. For deep sea systems that operate more vertically, however, this configuration could have a significant negative effect, and thus should be either considered in vehicle design where possible or incorporated in the vehicle's adaptation algorithms.



(a) Decrease in received signal strength between Fig. 4.7a and the same model including the electronics housing baffling. Dark regions indicate a larger baffling effect.



(b) The model from (a) with the same averaging bins used in Fig. 4.7.

Figure 4.10: Modeled (BELLHOP) results including the payload electronics housing baffling effect.

AUV Adaptivity

Fig. 4.11 shows the position of the AUV during its missions running `BHV_AcommsDepth` on 2010-08-08 overlaid with a single representative transmission loss ray trace from that day. The vehicle tracks the modeled downbeaming from 400-1000 meters well and also picks up the convergence zone off the first bottom bounce from 1400-2000 meters. As can be seen from the sensitivity analysis in Fig. 4.7(b), the region from 800-1200 has the highest sensitivity to changes in the sound speed profile. The AUV is responding to modeled caustics in this region that may or may not be real and are unlikely to be where they are predicted to be due to differences in the real environment from the model. This suggests an area of improvement for `BHV_AcommsDepth`: filter its objective function with a low pass filter with a cutoff inversely proportional to the measured sensitivity. By doing so, the `BHV_AcommsDepth` would make less certain choices in light of uncertainty, leaving depth decisions up to behaviors that have more knowledge.

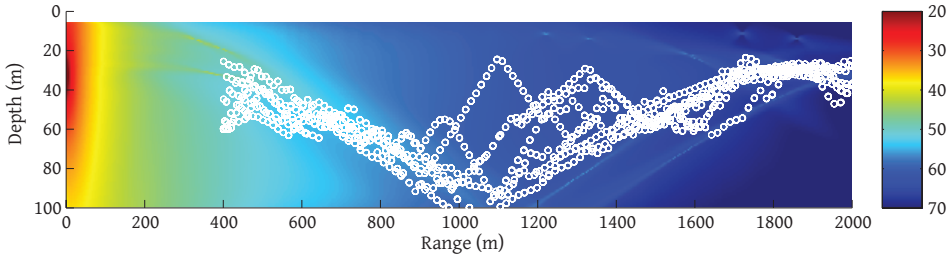


Figure 4.11: Depth position of the AUV Unicorn (white circles) with respect to range from the Buoy overlaid on a representative ray trace from 2010-08-08 08:28:19 UTC.

Applicability to other modulation schemes

The technique developed here (adaptively seeking the SNR maximum) is applicable to the FH-FSK modulation scheme since this modulation is resistant to Doppler and multipath effects by design. However, developers of the acoustic physical link are rapidly moving towards coherent modulation schemes, such as phase-shift-keying (PSK) due to the inherent inefficiencies of incoherent modulation. While any link is at least somewhat dictated by the SNR, this may not be the limiting factor in the case of PSK modems. Rather, the multipath structure (delay spread, stability of the arrivals, and the ratio of peak impulse power to the mean power) are generally more significant for PSK systems due to the short (time-domain) extent of the symbols. Thus, if analysis shows that one of these quantities provides a better proxy for the overall packet receipt probability, GRAM (with the appropriate acoustic model) can model that quantity and BHV_AcommsDepth will direct the vehicle to maneuver with respect to the utility function generated in that case. Therefore, the concept and system developed here is not specific to acoustic modems whose performance is SNR limited.

One caveat, however, is that the receiver in a phase-coherent system must compensate for the intersymbol interference caused by multipath at the physical layer. Thus, since there are numerous approaches to designing equalizers for this task (e.g. channel estimate based decision feedback equalizers, linear MMSE equalizers, passive time-reversal equalizers [97]), the resulting limitation of the system will likely be dependent on different aspects of the multipath structure. Of course, a perfect equalizer would be able to completely remove the multipath effect, and then the modem would be SNR-limited. Therefore, when applying this disruptive autonomy concept to vehicles equipped with phase-coherent modems, it is likely to be much more sensitive to the details of the spe-

cific modem implementation that is used.

4.4 Acoustic Connectivity in Deep Ocean Environments

In contrast to littoral environments, the bulk of the deep ocean is well isolated from atmospheric forcing, reducing the temporal variability on hourly and daily scale to a small fraction of volume close to the surface. Thus, the features of sound speed profile most significant to the acoustic environment are extremely stable, most notably the isothermal gradient dominating the sound speed profile below the SOFAR channel, controlling the dominant convergence zone propagation characteristic of deep ocean acoustics.

Another feature of the deep ocean which makes depth adaptation beneficial is the strong spatial diversity of the ambient noise. Thus, in shallow water, the ambient noise field tends to be dominated by the local, surface-generated noise. The noise from distant shipping and atmospheric disturbances will undergo significant attenuation due to the strong bottom interaction inherent to shallow water propagation. In contrast, the upward-refracting deep sea sound speed gradient allows noise from distant, natural and man-made sources of ambient noise to be carried over long distances with limited or no bottom interaction [85]. In very deep ocean environments in particular, this can result in an ambient noise field which is highly depth dependent. Thus, at depths above the critical depth, the ambient noise field has significant contributions from both local surface sources, and distant shipping and storms. Below the critical depth the acoustic field due to distant sources is evanescent, and the noise field is reduced to that produced by sources within a horizontal range of approximately half a convergence zone, the so-called Reliable Acoustic Path (RAP) cone. Consequently a reduction in noise of several dB can be expected near the bottom in such environments, which may be exploited by the platform autonomy.

Another feature associated with the interplay of signal and noise which may be exploited is the spatial diversity of the array gain. Thus, the performance of the acoustic array processing not only depends on the signal-to-noise ratio (SNR), but also on the angular distribution of the signal and noise components. For a deep receiver platform communicating with a shallow collaborator, the most reliable acoustic path will follow the convergence zone path, and the dominant elevation angle at the deep node will depend on the range: positive for short ranges, and negative for ranges beyond half a convergence zone (approx. 30 km). Similarly, the noise directionality will depend on the hori-

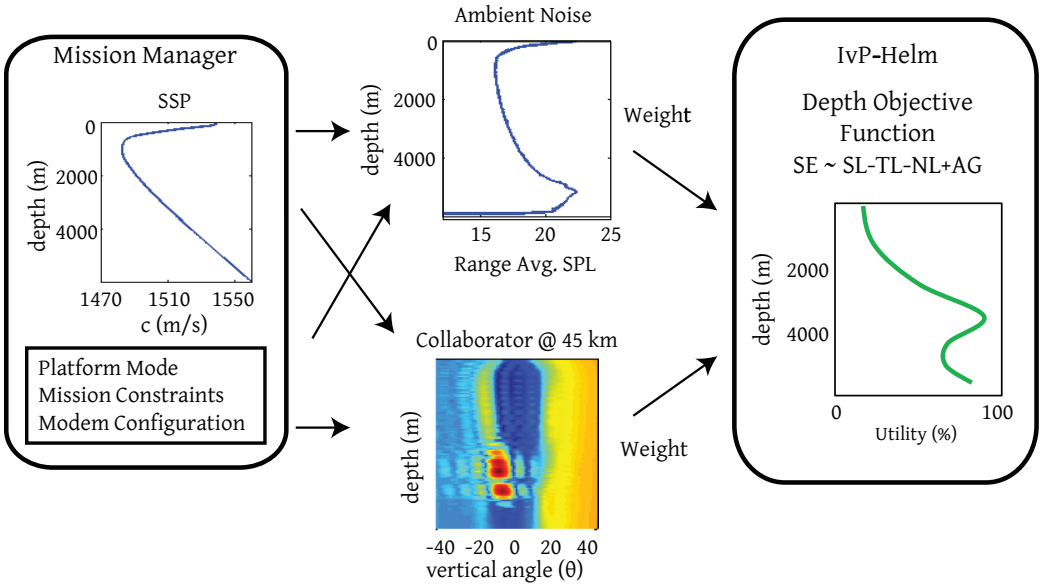


Figure 4.12: Functionality of model-based depth adaptation on a deep submersible for minimizing signal-noise ratio for maintaining optimal acoustic connectivity with near-surface acoustic contact. The on-board mission manager process will, based on the current environmental and situational information, request from GRAM a forecast of the estimated transmission loss. This forecast is then combined with estimates of the other terms in the sonar equation for planning the future depth trajectory, thus improving connectivity.

zontal source distribution, potentially leading to a strong depth dependence of the array gain which may be exploited for optimal system performance.

These features of the signal to noise trade-offs are conveniently captured in the classical *sonar equation*, which may be modeled by the GRAM infrastructure in combination with environmental information provided to the undersea platforms via the command and control infrastructure.

4.4.1 Depth Adaptation for Sonar Equation Optimization

In its simplest form the *Passive Sonar Equation* relevant to passive acoustic sensing and to underwater communication takes the form [85]

$$\mathrm{SE} = \mathrm{SL} - \mathrm{H} - \mathrm{NL} + \mathrm{AG}, \tag{4.7}$$

where SE is the resulting *signal excess*, SL is the *source level*, H the *transmission loss*, NL the *noise level* and AG is the *array gain*, all expressed in dB.

Figure 4.12 illustrates how an optimization of the system performance can be straightforwardly achieved in the MOOS-IvP autonomy architecture expanded by the GRAM environmental acoustic modeling framework.

The *Mission Manager* processes are responsible for maintaining the situational awareness of the autonomy system. To generate the current *environmental picture* including sound speed profile, noise profiles, and noise directionality, dedicated MOOS processes are fusing environmental information from all available sources, including i) historical data in on board databases; ii) environmental updates received from the *Field Control* via the acoustic communication network; iii) in-situ measurements by environmental sensors; and iv) on-board modeling.

The current environmental estimates resulting from this data fusion are assumed to be slowly varying and are therefore stored in the MOOSDB for use by the modeling infrastructure and the autonomy behaviors. For example, the overall depth-dependence of the SSP, and the noise level NG and array gain AG, will rarely be available from in-situ local measurements, and will therefore be fixed at mission start. On the other hand, occasional updates of the near-surface SSP, the location of local shipping traffic, and local noise measurements may be used to update the environmental picture, which is then published in the appropriate MOOS variables, allowing the platform to adapt to changes in the ambient noise field etc. For example the OASES model can be used through GRAM to estimate the current noise directionality, which may subsequently be used to update the estimate of the array gain for the current geometrical configuration.

In addition, the *Mission Manager* is maintaining the *situational awareness*, including navigation information for the platform itself, collaborating platforms, and acoustic contacts, required for the acoustic modeling of the transmission loss, and keeps track of the current geometry of receiving acoustic arrays and sources on-board the platform, which is required for the array gain term in the sonar equation.

Based on the currently available *environmental and situational picture* the autonomy system can generate objective functions which optimize the sonar equation (Eq. 4.7), or components thereof. For that purpose a dedicated MOOS-IvP behavior, `BHV_MaxSNRDepth`, has been developed. Thus, as illustrated in Fig. 4.12, it will first retrieve the depth-dependence of the ambient noise NL from the MOOSDB. Then, it will submit a request to GRAM for a current estimate of all ray arrivals predicted for the acoustic contact of interest. Using a local plane-wave representation the behavior it will continuously use this ray expansion to generate an estimate of the current array response, representing the terms $AG - H$

in the sonar equation. This performance metric is shown in the lower, center plot as contours versus elevation angle and depth in the water column. The behavior then combines the two depth-functions into one depth objective function, representing the utility versus depth for the sensing objective. As described earlier, this objective function is then merged with other depth behavior objective functions by the IvP multi-objective optimization algorithm.

The reason for choosing variable weighting of the signal and noise components in the sonar equation is the fact that their reliability can vary significantly. Thus, the average historical ambient noise profile may have a significant uncertainty, in particular in regions with heavy seasonal shipping or atmospheric conditions. In such cases the weight of the NL terms should be reduced. Similarly, if the propagation environment is highly variable, more weight may be applied to the noise profile in choosing the optimal depth.

Although the principle of the depth-adaptation as described above is rather simple, there are a couple of subtle issues associated with the use of the concept in deep water.

The first is the nature of the caustics characteristic to the deep convergence zone propagation. As described earlier, caustics also play an important role for depth-adaptation in shallow water. However, the convergence zone caustics associated with a deep source or receiver can be modeled with high confidence level due to the extraordinarily stable nature of the deep sea SSP gradient. Further, in contrast to the shallow water case, the convergence zone caustics of relevance to deep platforms always have the shadow zone above the caustic. Therefore, robustness requires that a bias towards depth be built into the depth objective function. Parameter studies have shown that for realistic variations of the near surface SSP and realistic depth uncertainty of the acoustic contact, the depth of the caustic can be predicted with an error of order 50 m. Hence, a low-pass filtering of the raw depth-objective function with a spatial cutoff frequency of 1/50 m^{-1} , starting at the surface, will yield an optimal depth of order 100 m below the predicted caustic, well into the "safe zone".

Another issue of particular importance to the deep water application is the time scales associated with depth changes. The maximum pitch of an AUV is of order 20°, which for a typical platform speed of 1.5 m/s yields a maximum rate of depth change of 0.75 m/s. Hence, it takes more than 20 minutes required to change the depth by 1 km. In contrast, an AUV operating in 100 m deep shallow water can reach any new depth in less than a couple of minutes. Consequently, the forecasting horizon required for the adaptive depth change will have to be significantly larger. On the other hand, the decision

to change depth cannot be based on a fixed forecasting horizon. For example, to reach a certain depth in 30 minutes may require that the platform cross through a shadow zone, which is obviously detrimental to the acoustic connectivity. Thus, decisions about depth changes have to be made on the basis of the entire time from the present to the chosen maximum time horizon. In `BHV_MaxSNRDepth` this is done by forecasting the depth objective function over a set of ranges up to and including the forecasting horizon. Then, the depth decision is made based on the following, simple, strategy:

- If the platform is already at or near the optimal depth locally, it will remain there by choosing a short time forecast as a basis for the adaptation.
- If the local depth is not any longer near the optimal, e.g. when approaching a range where a convergence zone caustic is forming, the behavior will select as a target depth the optimal depth at a range within the forecast horizon, which can be reached with the minimum platform pitch.

The first criteria will ensure that once the platform has reached a stable optimum, it will track it until a discretely different optimum starts developing somewhere in the section of the water column allowed for the adaptation. The second criteria will ensure that the platform is not forced to change depth so fast that it crosses into a shadow zone to reach a future optimum. In other words, the platform will try to reach an optimum depth "ridge" tangentially. Of course this is all assuming that the contact motion continues at the current range rate and heading. Also, the depth control is not made in pitch directly, but in depth, so whenever significant depth changes are requested, the platform will increase the pitch to maximum. However, since this process is repeated at the rate of the IvP Helm updates, typically of order of a second, the depth adaptation will occur smoothly, as illustrated in the simulation example following.

4.4.2 Deep Sea Simulation Example

For demonstrating the deep ocean performance of the acoustic connectivity optimization behavior, we will use the high-fidelity simulation environment developed and established at MIT. This virtual environment provides a virtual ocean environment with high-fidelity simulation of the environmental acoustics, using the GRAM embedded modeling infrastructure. In addition to the environmental acoustic modeling, the simulator also incorporates hydrodynamic models of both the submersible and sonar arrays, as well as the ability to simulate the supporting acoustic communication networking. The fidelity of

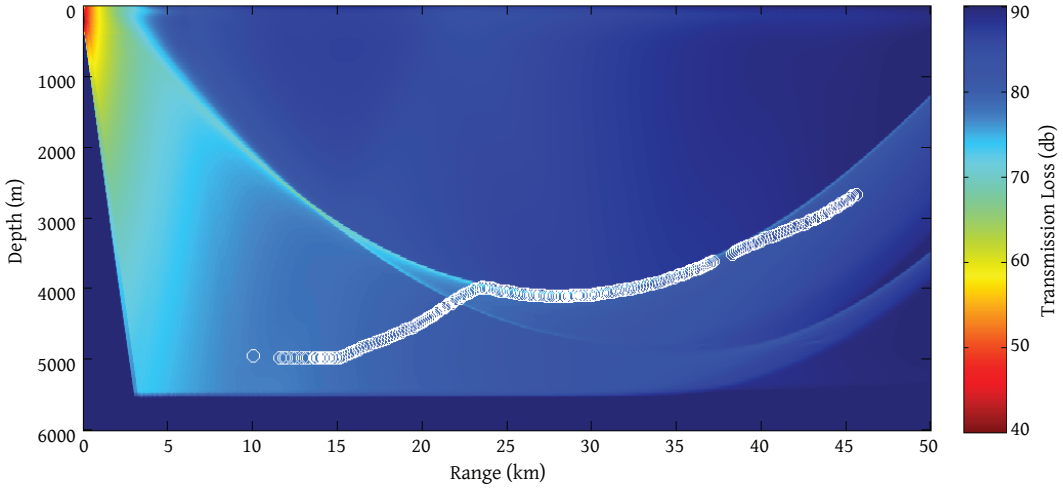
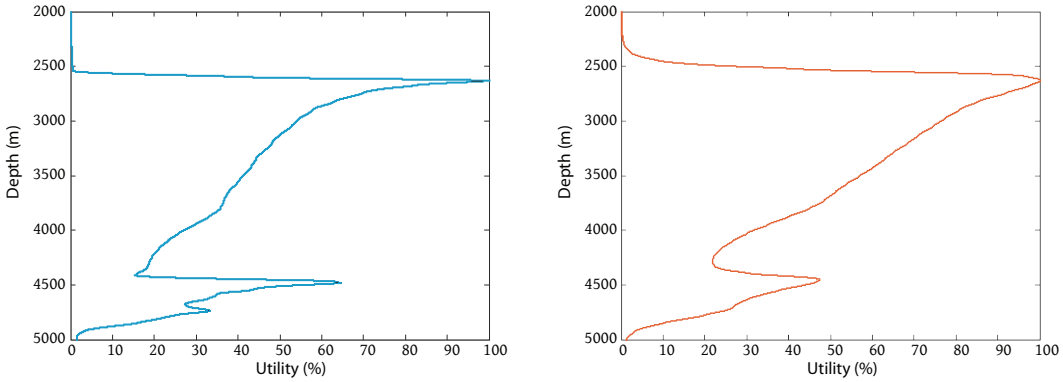


Figure 4.13: Contours of transmission loss in dB for a moving acoustic source at 200 m depth in deep water environment, plotted vs depth and the range from from the source. White markers show the adaptive path of a deep submersible optimizing acoustic connectivity with surface source, constrained by a maximum pitch of 20 degrees.

the simulation environment allows the testing of the exact same autonomy software and configuration as applied in actual field deployments.

To illustrate the performance of the MOOS-IvP behavior `BHV_MaxSNRDepth` in adapting to the local acoustic environment, we consider as an example the autonomous acoustic connectivity of a deep submersible in a Pacific environment with a water depth of 6000 m, with a critical depth of 4000 m. The submersible is tasked to maintain acoustic connectivity with an acoustic contact at 200 m depth, opening range at a rate of 5 m/s (10 kn). The ambient noise is assumed to be constant to the critical depth, and then decay linearly by 3 dB/km towards the bottom. The submersible is initially deployed at 5000 m depth, performing a hexagonal loiter pattern of radius 500 m. No particular array geometry is considered, and the depth adaptaion simply optimizes the signal-to-noise ratio.

When the contact range exceeds 15 km, the `BHV_MaxSNRDepth` behavior is activated and determines that a caustic, and associated optimal depth exists at a depth of 3000 m. The behavior in this case is configured to forecast 2 hours into the future, and determines that it can reach the caustic at a range of 30 km and a depth of 4000 m and initiates the depth change since it is not currently at an optimum depth. Because of range estimation uncertainty and the fact that the behavior uses a conservative contact range estimate, the submersible reaches the caustic at a range of 23 km, after which the adaptation strategy will make it track the optimal depth below the caustic, as evidenced by the actual track



(a) Raw objective function showing sharp caustic at 2500 m depth.

(b) Low-pass filtered corresponding to depth-uncertainty of caustic of 50 m.

Figure 4.14: Forecasted depth objective function at 43 km range.

shown in the figure by white circles. Note that the apparent depth rate of the submersible does not appear constant during the ascent. This is due to the loiter of the submersible, leading to variations in the range rate. When the range exceeds 40 km, the maximum pitch of the submersible does not allow it to continue to track the caustic, and it instead continues to climb at its maximum pitch.

To illustrate the performance of the low-pass filtering of the depth objective function, Fig. 4.14 shows the raw depth objective function at 43 km range for the example in Fig. 4.13. The left plot shows the raw depth objective function for minimizing the signal-to-noise ratio, while the right plot shows the low-passed filtered objective function with a maximum 100 m below the depth of the convergence zone caustic.

4.5 Conclusion

In this work, the new Generic Robotic Acoustic Modeling (GRAM) tool was introduced for successfully utilizing existing acoustic models on embedded processors onboard autonomous underwater vehicles. GRAM was applied to two representative problems: improving communication in an anisotropic shallow water environment, and maintaining contact with an acoustic target in the deep sea.

In addition to the examples given here, GRAM has applicability for software-only simulation of actual sonars as well as hardware-in-the-loop testing of modem systems (where signals from an existing hardware modem are delayed and convolved with the channel

measured by GRAM). These tools are available as part of the open source LAMSS project (https://launchpad.net/lamss) and access is available upon request.

5 *Closing remarks*

> La machine, qui semblait d'abord l'en écarter, le soumet avec plus de rigueur encore aux grands problèmes naturels.[1]
>
> Antoine de Saint-Exupéry, Terre des Hommes (1939)

By necessity, contributions from a thesis are typically narrowly focused and sometimes appear to be making a small advance in a subfield of a minor branch of engineering. Thus, in concluding, it is worth stepping back and examining the broader context and potential impact of the work presented here. In short, why does this work matter? Why should anyone care?

Earth's oceans cover a vast portion of the world's surface. Sea level rise and the melting of the Arctic ice and polar glaciers due largely to anthropogenic causes threatens to change the world we reside in: politically, socially and militarily. Oceans also contain present and potential sources of food and energy, and lifeforms that seem alien to us land-dwelling humans. Understanding and sensibly harvesting the resources of the ocean is thus of significant value to humanity. Both of these endeavors are and will continue to be aided by engineered systems. Robotic systems, as has been shown as well in space exploration, have the potential to collect far more information per dollar invested over longer durations than human explorers. These data are useless unless they reach people, however, and this, in short, is the reason why this work matters. The advances outlined here, however small, improve the ability of undersea robots to share information amongst themselves for more efficient missions, and to their human operators and scientists for slowly improving the collective knowledge of our species. We can choose

[1] The machine does not isolate us from the great problems of nature but plunges us more deeply into them.

to our detriment not to act on this knowledge, but we have no hope of making rational decisions without it.

The open source Goby-Acomms project (Chapter 2) is gaining traction as a *de facto* standard amongst the MOOS segment of the underwater robotics community, and increasingly outside it as well. Effecting internationally recognized standardization of DCCL, perhaps in collaboration with open JANUS physical layer, will be a valuable goal in the next few years. Practical routing and session protocols should be considered and will hopefully form part of the continued work. It is, after all, far easier to communicate if everyone speaks the same language. The design of the Goby project is intended to foster such collaboration: easily accessible source code, documentation, wikis, email lists, and bug report tracking.

The state observation technique for compression of vehicle positions (Chapter 3), which may be thought of as an extension of DCCL to include models present in the vehicle autonomy system, can and should be extended to telemeter innovations of scientific sensor data from modeled or prior data sets. Given the realities of the physical link, optimal compression (even at the cost of significant human effort and time) will pay off. Fielding vehicles is costly, time-consuming, and even at time dangerous, and it is essential to the mission to make the most of the available digital wireless throughput.

Taking the value of communications one step further, Chapter 4 shows that it appears to be possible, and potentially useful (depending on the specifics of the task to be performed) to incorporate improving communications into the autonomous navigation task of the vehicle(s). Validating these results in the deep sea and other varied environments will form the basis of the next stage of this work.

A Unified Command and Control for Heterogeneous Marine Sensing Networks

A.1 Introduction

A.1.1 Overview of the `Unified C2` architecture

Autonomous marine vehicles are becoming inexpensive enough (e.g. OceanServer's sub-$100,000 Iver2 [98]) and mature enough as single platform systems that use of these vehicles in groups is increasingly feasible. Groups of vehicles offer redundancy, specialization, and improved performance in certain underwater research tasks. However, even autonomous systems need to be commanded by a human, especially in the process of developing such systems. The concept of unified command and control (`Unified C2`) arose from the authors' need to deploy and command multiple autonomous underwater vehicles (AUVs) for target detection and environmental sampling. `Unified C2`, described in this paper, was developed from experiences during numerous field experiments involving AUVs and autonomous surface craft (ASCs) operated in shallow water (depths on the order of 100 meters) from 2008 to 2009. `Unified C2` is composed of three major components:

1. Hierarchical configuration that is set before the vehicles are in the water, as described in section A.2.

2. A network infrastructure that allows commands for an arbitrary number of vehicles to be sent by a single operator and data to be received while the vehicles are in the water, outlined in section A.3.

3. Data visualization by meshing the data with Google Earth satellite imagery and bathymetry. The process of interfacing the `Unified C2` network to Google Earth

is discussed in section A.4.

A.1.2 MOOS-IvP autonomy architecture

All the vehicles (and the operator topside computer) in the Unified C2 system run the MOOS-IvP autonomy architecture [99]. MOOS-IvP is comprised of two components, written entirely in the C++ programming language:

1. MOOS, the Mission Oriented Operating Suite, which is a publish-subscribe infrastructure for asynchronous interprocess communication between a number of discrete processes or MOOS Modules (collectively, a MOOS Community). Each MOOS Module communicates only by publishing data to the central data bus (the MOOSDB) and by receiving data from the MOOSDB for which it had previously subscribed. No process communicates directly with another process. This allows for rapid prototyping by partitioning the vehicle software system into modules that can essentially be developed and debugged individually.[1]

2. pHelmIvP, the Interval Programming Helm, which is a behavior-based decision engine that commands the low level control by producing a desired heading, speed, and depth for the vehicle. pHelmIvP allows for an arbitrary number of behaviors to compete for the vehicle's action, producing a "best option" by evaluating the entire objective *function* of each behavior over the entire (feasible) heading-speed-depth space, rather than just arbitrating over a *single* desired heading, speed, and depth from each behavior.[2]

Figure A.1 models the subsystems for the topside operator MOOS community and a sonar AUV MOOS community ("backseat" computer). Each subsystem is composed of one or more MOOS Modules (designated by a name starting with a lower case 'p' or 'i') that communicate through the central data bus (the MOOSDB). The desired actions produced by

[1] MOOS is like an office where all the employees (MOOS Modules) never talk to each other but post papers of their work (publish) and copy others' papers (subscribe) as needed on a central bulletin board (the MOOSDB).

[2] pHelmIvP works something like this: two people (behaviors) are trying to pick a movie to watch (by analogy, choose a heading, speed, and depth). If each tells only their top pick and, as is likely, these do not match, the best option is to pick one person's top choice at random, which leaves the other unhappy (50% utility). However, if both give a ranked list of their choices (objective functions), then a match might be found slightly down the list that reasonably satisfies both people (say 80% utility). pHelmIvP works like this second option; by looking at the entire utility function of all behaviors over the entire space, compromises can be made and more work gets done.

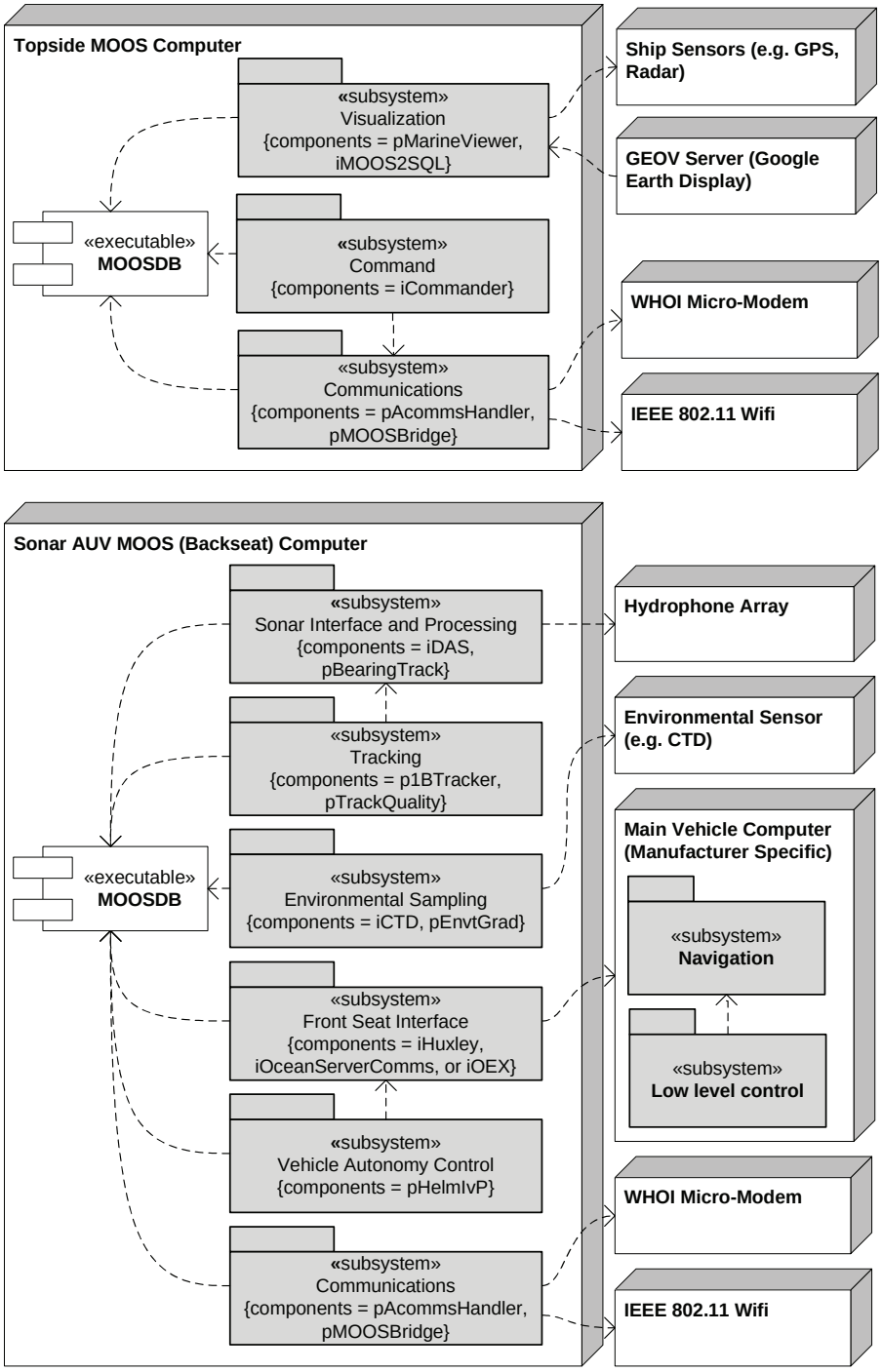


Figure A.1: Model of the subsystems of the operator topside (ship-based command computer) and a sonar Autonomous Underwater Vehicle (AUV) (e.g. the Bluefin 21 *Unicorn*). Each subsystem is comprised of one or more independent MOOS processes that communicate only through a central data bus (the `MOOSDB`). While processes do not communicate directly, key implicit dependencies between subsystems are indicated with dotted arrows.

pHelmIvP are passed to the vehicle's low level control computer ("frontseat" computer, details of which vary depending on the vehicle and manufacturer) which is responsible for actuating the commands. Precisely how the "frontseat" carries out the commands depends significantly on the details of the individual vehicle design. For example, a heading change might be implemented on one vehicle by tilting a single thruster but implemented on another vehicle by varying the thrust differential between two thrusters distributed some distance from the vehicle's center of gravity. By having this split "frontseat"-"backseat" structure, Unified C2 can command potentially very different vehicles via a single abstracted autonomy architecture (MOOS-IvP).

A.1.3 Field Exercises

All the work involved in developing Unified C2 was motivated by and tested at several field trials spanning from 2008 to 2009. These exercises took place in shallow water (depths on the order of 100 meters) and involved four different types of AUVs and ASCs, all running the MOOS-IvP autonomy in a frontseat-backseat configuration as discussed in section A.1.2. These trials are summarized in table A.1 and experiences from them will be referenced throughout the remainder of this paper.

A.1.4 Prior Art

Traditionally, underwater vehicles are controlled using platform specific interfaces where the vehicles communicate with the surfaced assets using dedicated wireless ethernet communication channels. Each mission is configured before the dive is initiated. After the vehicle is submerged, status and contact reports are transmitted via acoustic modem, and simple re-deploy commands are sent from the topside C2 console, provided such maneuvers are allowed in the mission script and coded into the acoustic datagrams. The fixed message set highly limits the ability of the human operator to command the vehicles while submerged. Also, almost uniformly, the command and control infrastructures have been proprietary and manufacturer-specific. Thus, there is little openly published literature on these systems.

For example, most REMUS vehicles (manufactured by Hydroid) are controlled within this paradigm using the REMUS Vehicle Interface Program (VIP) software infrastructure. Similarly, Bluefin Robotics' AUVs, apart from the MIT LAMSS fleet, are controlled using the Bluefin proprietary topside. Although some components of these systems are

Table A.1: Summary of field trials during which `Unified C2` was developed and tested. The column marked *Messages Used* refers to Dynamic Compact Control Language message names unless specified. The experiment datum is a location in the southwest corner of the operation region from which all vehicle positions are referenced using the Universal Transverse Mercator projection with the WGS 84 ellipsoid [49].

Name	Dates	Summary	Vehicles	Messages Used	Experiment Datum
CCLNET08	1.19.08 - 2.1.08	First Engineering test for GLINT	**AUV**: 1 NURC OEX. **ASC**: 2 Robotic Marine Kayaks	PlusNet CCL (DCCL not developed)	44.08892°N, 9.85054°E
SQUINT08	5.19.08 - 5.23.08	Second Engineering test for GLINT	**AUV**: 1 NURC OEX. **ASC**: 2 Robotic Marine Kayaks	PlusNet CCL (DCCL not developed)	44.08892°N, 9.85054°E
GLINT08	7.22.08 - 8.14.08	Interoperability of marine vehicles for passive acoustic target detection	**AUV**: 1 NURC OEX, 1 Bluefin 21 (Unicorn), 1 OceanServer Iver2. **ASC**: 3 Robotic Marine Kayaks	PlusNet CCL (DCCL not developed), compressed CTD messages (precursor to DCCL)	42.5°N, 10.08333°E
SWAMSI09	3.23.09 - 4.5.09	Mine detection using bistatic acoustics.	**AUV**: 2 Bluefin 21 (Unicorn, Macrura)	LAMSS_DEPLOY, LAMSS_STATUS, ACTIVE_CONTACT, ACTIVE_TRACK, CTD	30.045°N, 85.726°W
GLINT09	6.29.09- 7.21.09	Interoperability of marine vehicles for passive acoustic target detection	**AUV**: 1 NURC OEX, 1 OceanServer Iver2. **ASC**: 2 Robotic Marine Kayaks	LAMSS_DEPLOY, SURFACE_DEPLOY, LAMSS_STATUS, LAMSS_CONTACT, LAMSS_TRACK, CTD	42.47°N, 10.9°E
DURIP09	8.19.09 - 9.02.09	Engineering test for collaborative autonomy and towed array improvements	**AUV**: 2 Bluefin 21 (Unicorn, Macrura). **ASC**: 2 Robotic Marine Kayaks.	LAMSS_DEPLOY, SURFACE_DEPLOY, LAMSS_STATUS, LAMSS_CONTACT, LAMSS_TRACK, CTD, BTR, ACOUSTIC_MOOSPOKE	42.35°N, 70.95°W

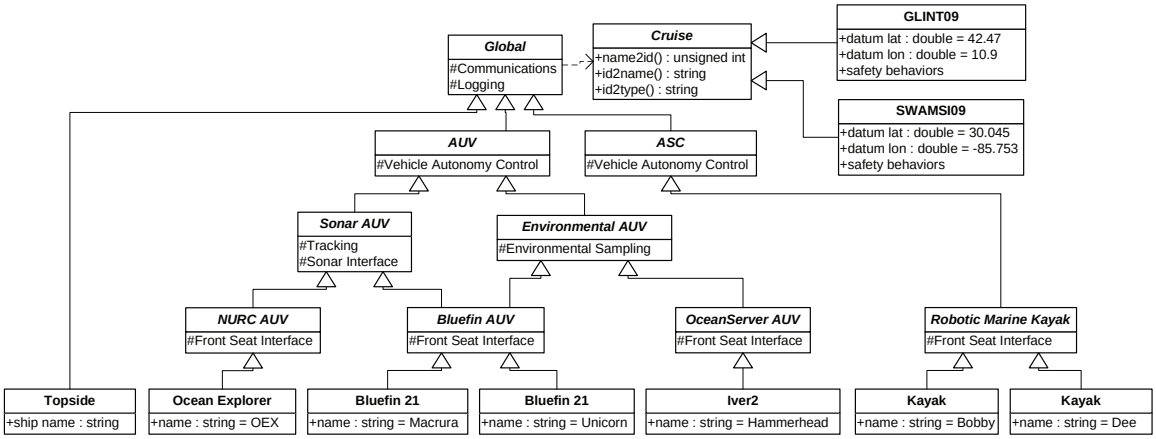


Figure A.2: Hierarchy of classes representing the software configuration of a collection of Autonomous Surface Craft (ASCs) and Autonomous Underwater Vehicles (AUVs) used in the field experiments summarized in table A.1. The open arrow indicates a generalization of another class. The bottom row of the hierachy represents configuration for an actual vehicle; the rest of the hierarchy is abstract. By putting configuration as high in this tree as possible, configuration redundancy is minimized and changes can be propagated easily to all the leaves. Each of these classes represents one or more text files that are assembled by a text preprocessor before being passed to the MOOS process launch manager (`pAntler`).

common, such as the CCL message coding, they are in general completely incompatible, requiring a separate topside C2 infrastructure for each vehicle type and often for each vehicle.

In contrast, the payload-centric autonomy and communication infrastructure integrated using `MOOS-IvP` allows `Unified C2` to be applied to controlling a completely heterogeneous network of vehicles and fixed nodes, all from a *single* topside command and control console on a surface ship or on shore. Furthermore, the Dynamic Compact Control Language (DCCL) messaging (detailed in section A.3.2) allows for a much richer and more quickly reconfigurable set of commands to be sent to the vehicles while in operation (i.e. underwater and out of reach of wireless ethernet).

A.2 Hierarchical configuration

`MOOS-IvP`, like many software systems, requires configuring all the processes with some initial state. Configuration values range from hardware settings (e.g. serial port names) to autonomy settings (e.g. behaviors) that govern the vehicles' initial mission and all pos-

sible states that can be commanded during runtime. In a research environment, many processes expose a good deal of initial state to the prelaunch configuration since optimum defaults are unknown *a priori* and are the subject of research themselves. This means there are a large number of prelaunch configuration values, many of which need to be consistent across vehicles. Furthermore, it is often desirable that changes to the configuration be able to be easily propagated throughout the set of vehicles without changing the value in more than one place. With a single vehicle, a single file defining the configuration is manageable, but as the number of vehicles grows, the authors have found that making sure that changes propagate is a logistical difficulty. The Communications subsystem is an example of one that requires a significant amount of consistent configuration throughout the network. If encoding/decoding templates (DCCL XML files: see section A.3.2) are inconsistent between vehicles, the messages may not be decoded properly.

To address this issue, a hierarchical configuration system was implemented where each vehicle's configuration is inherited from a number of parents up the "tree" of the hierarchy. Figure A.2 diagrams this hierarchy for a number of the vehicles used in the field exercises tabulated in table A.1. The configuration for each vehicle can be thought of as a class that inherits configuration much in the same way as `public` class inheritance (which expresses an "is-a" relationship) works in object-oriented programming languages (such as C++). For example, in Figure A.2, the Ocean Explorer (OEX) *is a* NURC AUV which *is a* Sonar AUV, etc. Any configuration for a Sonar AUV is inherited by all NURC AUVs which is then in turn inherited by the OEX.

Examples of the subsystems that are typically configured at each level are given in Figure A.2. Cruises are handled as a class that the `Global` configuration (i.e., the root of the tree) depends on. Information contained in each cruise is specific to the operations of that experiment such as a local datum for geodesic conversions, local obstacles to avoid, and a mapping of the vehicle names to a numeric id number for acoustic communications.

Since `MOOS` is limited to accepting plain text configuration files with key/value pairs, this hierarchical configuration structure is implemented through a series of small text files (that represent each "class" in Figure A.2) that are included to the main configuration file via a text preprocessor (`splug`), which is very similar to the C Preprocessor (`cpp`). The configuration is kept consistent throughout the vehicles by using version control software (in our case, Subversion) in the same manner that the source code is kept consistent.

This type of configuration was first developed and tested on ASCs at GLINT08 and then migrated to AUVs by SWAMSI09. The authors have found that ASCs make excellent testbeds for changes to the `Unified C2` architecture as they can be reprogrammed while deployed and are at less risk of loss due to errors in the new system as they do not dive.

A.3 Network

A.3.1 Subsea and surface networking

Reliable communications between underwater and surface nodes in the ocean are at best an uncertainty. Underwater communications are only practical using an acoustic carrier due to the rapid attenuation of most electromagnetic wavelengths in sea water [9]. Acoustics are subject to the variations of the physical propagation of sound, are slow (over five orders of magnitude slower than light), and have little usable bandwidth due to the low frequencies of the carriers, leading to unpredictability and low throughput when sending messages below the surface. Above-surface wireless ethernet (wifi) affords much higher bitrates but is subject to antennae line of sight issues, especially with small masted vehicles bouncing in the waves. In the field trials given in table A.1, the authors have found the range of acoustic communications to often be substantially better (factor of two or more) than the wifi connection to Bluefin 21 AUVs. However, ship-to-ship or ship-to-buoy networking over ranges of several kilometers can be reasonably reliable if properly installed. In the authors' experience, *reliable* communications are much more important to successful field operation of robotic vehicles than *fast* communications.

`Unified C2` networking has evolved from a single-hop underwater network to a single-hop underwater network with (potentially) multi-hop above-water network. This allows the operator's commands to a subsurface node to be forwarded to the nearest "gateway," which is either an acoustic modem on the ship, or a buoy or autonomous surface craft (ASC) with a modem. The last option allows the most flexibility, as the ASC can transit to improve its range to the underwater vehicle and improve the chance of reception.

One behavior that works well for cases when high throughput from the underwater vehicle is needed is to have an ASC trail the AUV by a short distance, making the acoustic transmission closer to vertical (as well as a shorter distance) and reducing the refraction (since the water is much more stratified in the vertical than the horizontal) and multiple reflections (multipath) that destroy acoustic packets. Figure A.3 shows two scenarios

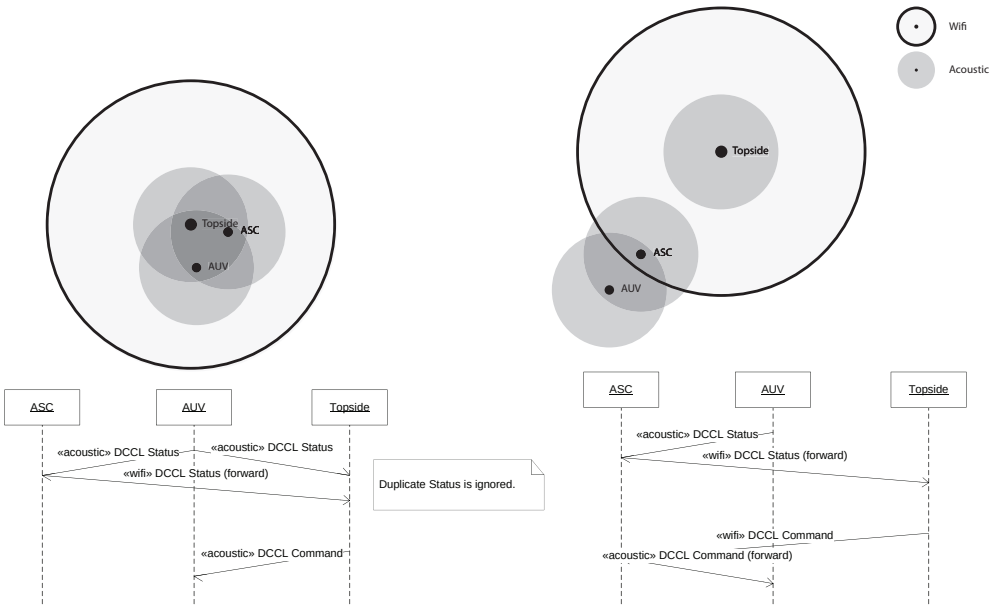


Figure A.3: Transmission coverage diagram (top) and sequence diagram for sending a command and receiving a status message from the topside to an underwater vehicle (below). On the left the AUV is within range of the topside for both networks but since the AUV is below the surface only the acoustic message is received. In the second scenario the AUV is out of communication directly with the topside so the messages are passed through a surface craft (ASC).

for a ship (topside), an AUV and an ASC, where the topside commander wishes to send a command to the AUV as well as receive a status message (e.g., position, orientation, speed, health) from the AUV.

In the first scenario (on the left), the AUV is in direct acoustic communication with the ship modem so that the messages pass directly between the topside and the AUV. In the second scenario (right side of Figure A.3), the AUV is out of acoustic range of the ship, but within the range of the ASC. The ASC is in range above the surface with the ship so that it can forward the messages between the two. This is a somewhat brute force technique to routing (all important packets are forwarded), but removes the need to maintain a very rapidly changing routing table (since almost everything is in motion) and since the bandwidth available above the sea is so much higher (three to four orders of magnitude above the acoustic underwater communications), there is no risk of saturating the wifi network. This trailing behavior as part of the `Unified C2` network was first successfully demonstrated at the CCLNET08 experiment in January 2008 (see Figure A.4). The Robotic Marine kayaks *Dee* and *Bobby* trailed both the research vessel (*Leonardo*) and the *OEX* AUV,

Figure A.4: GEOV screenshot showing ASCs *Bobby* and *Dee* trailing the *OEX* AUV to improve networking throughput at the CCLNET08 experiment. Acoustic communications are only robust over short distances with largely vertical propagation, which makes these "mobile gateways" effective, as they can always maintain a relatively short distance to the AUV. The ASCs were commanded to trail at 100 meters behind the AUV at 170° and 190° relative to the AUV's bow. The ASCs were also running a high priority collision avoidance behavior with the *RV Leonardo* ("leo"), which accounts for the shift to port from the normative tracking positions.

forwarding acoustic messages to the topside. Similar behavior was used as part of the GLINT08 and GLINT09 networks.

The underwater network can be thought of as a local network broadcast, where every vehicle hears every transmission within range (a few kilometers for the 25 kHz WHOI Micro-Modem (see [100]) employed in all the authors' experiments, though highly variable within the spatial and temporal environment). The above sea network is based off TCP/IP and UDP protocols, the latter used for cases when throughput is so bad that TCP/IP does not adequately handle all the missed packets and what is most desirable is the *newest* message rather than *all* the messages.

In fact, throughput of all messages is rarely achievable. Using the Band C WHOI Micro-Modem with a carrier of 25 kHz, the authors have observed that actual throughput in good conditions ranges from about 20 bits-per-second (bps) using the low-rate frequency-shift keying (FSK rate 0) modulation to 2000 bps using the high-rate phase shift keying (PSK

rate 5) modulation. However, this throughput is highly dependent on the environmental conditions (sound speed profile, water depth, surface waves), with the higher rates the most sensitive. It is also dependent on the orientation of the vehicle and the mounting position of the modem. Communication performance can change substantially over the course of one day to the next due to changes in the environment. Communication in very shallow water ($\sim$20 meters depth), such as that in the operation region of SWAMSI09, posed the most difficulty. Communications would fail for tens of minutes at a time, especially at ranges of more than several hundred meters. However, in GLINT08 with a water depth of 100 meters, even high rate (PSK rate 5) transmissions would occur successfully at ranges up to 1.6 km. To maximize the throughput, but ensure some consistent messaging, the authors will command the vehicles to alternatingly send messages at low and high rates.

All messages are prioritized to compensate for the reality of low and variable throughput. The message with the highest priority is sent at each opportunity provided by the medium access control (MAC) scheme. The priorities grow in the time since the last message of that type was sent. Thus, higher priority messages do not continuously overwhelm lower priority messages. This is implemented through a set of queues: one queue for each DCCL message (e.g. LAMSS_STATUS has one queue, ACTIVE_CONTACTS has another). Each queue i has a priority value ($P_i(t)$) which is calculated using

$$P_i(t) = V_i \left(\frac{t - \tau_i}{ttl_i} \right) \tag{A.1}$$

where V_i is the base value of the queue (i.e. the importance of that type of message), ttl_i is the time-to-live (in seconds) of messages in that queue, τ_i is the last time a message was sent from the ith queue and t is the current time. Messages with a short ttl are presumably more time sensitive, so their priorities grow faster than messages with a longer ttl. When the MAC scheme permits a message to be sent, the queue is chosen with the highest $P(t)$.

One example of why this dynamic growth of priorities is desirable is during a subsea detection. The AUV generates track and contact reports (high priority messages) for the detected object, but the operator still desires an occasional lower priority status message to ensure the vehicle is performing correctly. Were priorities not to grow, the operator would never receive a status message while track reports were being generated.

Messages received from underwater nodes are forwarded by the surface nodes so that all interested parties (typically all the assets in the experiment) can log and use these data. This allows for redundancy when certain vehicles are out of range of the initial sender.

A.3.2 Dynamic Compact Control Language (DCCL)

The Dynamic Compact Control Language (DCCL) provides a structure language for defining fixed-length short messages primarily to be sent through an acoustic modem (but which can be sent through TCP/IP or UDP as well). The messages are configured in XML and are intended to be easily reconfigurable, unlike the original CCL framework (see [101]) used in the REMUS vehicles that DCCL aims to replace. DCCL can operate within a CCL network, as the most significant byte (or CCL ID) is 0x20. DCCL messages can be easily reconfigured because they are defined in plain text XML files which are encoded using a set of standard rules given in Table 2.3. Creating a new CCL message, on the other hand, requires defining each field and its encoding in software code and then testing to ensure its correctness. By using an XML schema and other structural checks, DCCL greatly reduces the chance of undetected error when defining new messages. A more in-depth explanation of DCCL, including example XML files, can be found in [102]. The source code and documentation for DCCL, provided as the C++ library *libdccl*, is available as part of the open-source `goby-acomms` project at `http://launchpad.net/goby`.

One example that demonstrates the flexibility of DCCL occured in SWAMSI09. In this experiment, two AUVs were to perform bistatic acoustic detection of mine-like targets on the seafloor. In order to do this, both AUVs needed to traverse a circular pattern around the potential target, maintaining a constant bistatic angle. However, entering into this collaboration and maintaining the correct angle required handshaking and data transfer between both vehicles. At the time of the experiment there was no available fields in the current message set to perform this handshake. By adding some new fields (i.e. several lines of XML text) to the `LAMSS_STATUS` message, the vehicles were able to perform the handshake underwater as needed. This would not have been possible with the hard-coded CCL messages without substantially more planning, coding, and testing.

DCCL is similar to the ASN.1 unaligned Packed Encoding Rules [103]. DCCL messages are packed based on bit boundaries (as opposed to bytes or words) determined with knowledge of the XML file. They are not self-describing, as this would be prohibitively expensive in terms of data use. Thus, the sender and receiver must have a copy of the same XML file for decoding a given message. Also, each message is defined by an identification number that must be unique within a network.

DCCL messages can be used for any kind of data expressible as a combination of one or more customizably bounded integers or fixed point real numbers, strings, enumerations, booleans, or uncategorized hexadecimal. Thus, among other uses, they can be

used to encode commands and status messages for AUVs. The algorithms used for encoding the DCCL types are provided in Table 2.3. DCCL is currently limited to these data types, which cover the needs for vehicle status, command, and collaboration messages. However, it is not well suited for large vectors or arrays of data, since each field must be specified separately in the XML file. Presently, other processes are used, such as `pCTDCodec`, to encode data messages into hexadecimal that can form part or all of a DCCL message. `pCTDCodec` uses delta-difference encoding to compactly store samples from a Conductivity-Temperature-Depth (CTD) sensor. Delta-difference encoding makes use of the correlation between samples by sending only the difference values from an initial "key" sample.

Within the `MOOS-IvP` autonomy infrastructure, all the acoustic communications are handled by a process called `pAcommsHandler`. This involves encoding/decoding (using DCCL), message queuing and transmission by priority, and interaction with the modem firmware (driver). The sequence for sending a command underwater is modeled in Figure A.5. Above the surface, over wifi, the process is simpler since most of the networking is handled by TCP/IP (see figure A.6). Here, pAcommsHandler is used just for encoding and decoding (again with DCCL). DCCL is perhaps unnecessary with the higher rate wifi connectivity, but by encoding all messages with the same scheme, a vehicle can switch easily and seamlessly from being commanded on the surface to being commanded below the surface.

The command process (`iCommander`) is an NCurses terminal graphical user interface that allows a user to type in values for the fields for any number of DCCL messages. Since the fields displayed to the human operator for entry are directly read from the XML message files, any change to the command message contents are immediately available to the operator without changing software code. In the authors' experience it is preferable to avoid changing code on experiments whereever possible without sacrificing the ability to make changes to the messages (e.g., to allow a new experiment to take place).

DCCL loads the message configuration (given as XML) into C++ objects at runtime. Thus, when a message is changed, the vehicle and topside software needs merely to be restarted, not recompiled, for the change to propagate through the communications software (`pAcommsHandler`) and the command software (`iCommander`). This is advantageous for embedded computers, such as those deployed on marine vehicles, since compilation can be a time consuming process.

The authors have developed about a dozen DCCL messages that are widely used in

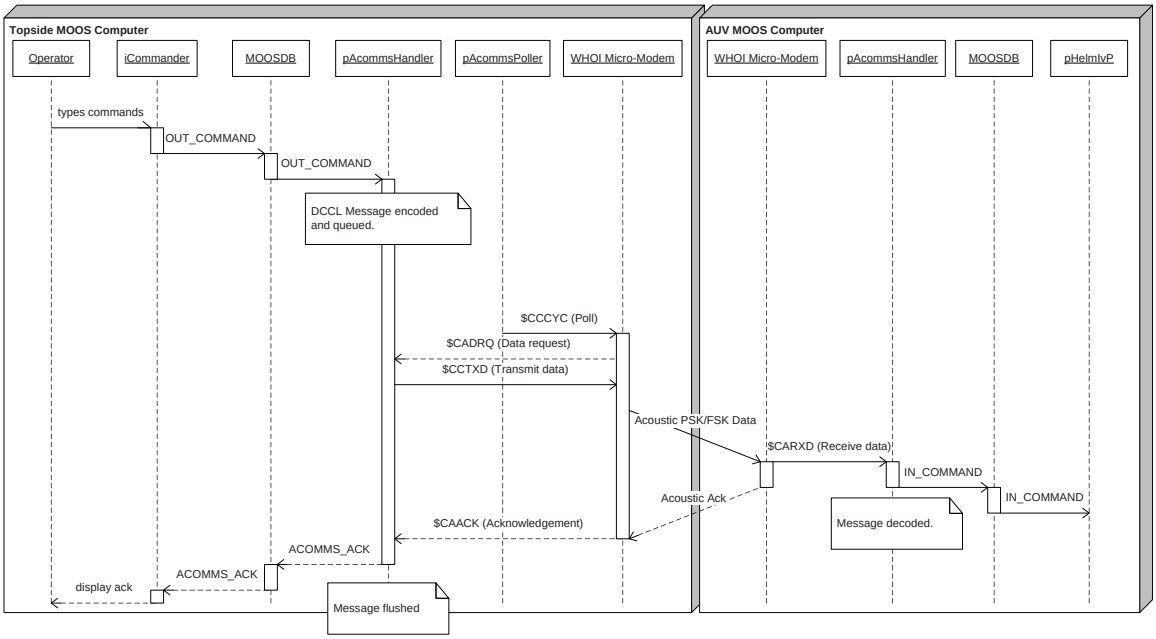


Figure A.5: Sequence diagram for sending a command to an AUV using the `Unified C2` infrastructure. The operator types a command into the `iCommander` application, which is configured with the desired set of DCCL messages (defined in XML). This message is encoded using DCCL, queued by `pAcommsHandler`, and sent through the water using a 25 kHz acoustic carrier (the WHOI Micro-Modem). On the receiving end, the message is decoded and an acknowledgment is generated and displayed to the operator. `pAcommsPoller` handles access of the acoustic channel by a centralized time division MAC scheme.

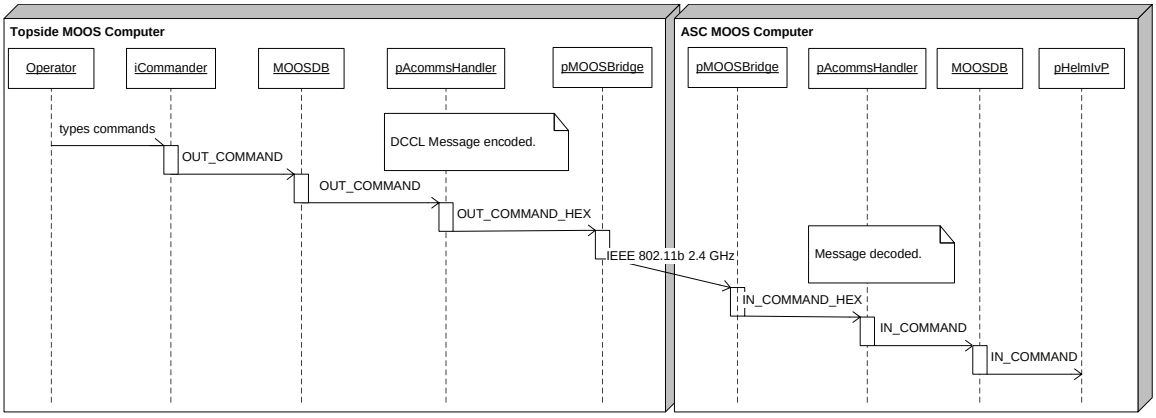


Figure A.6: Sequence diagram for sending a command to an ASC using the Unified C2 infrastructure. The TCP transport layer handles reliability so no acknowledgment is produced for the operator.

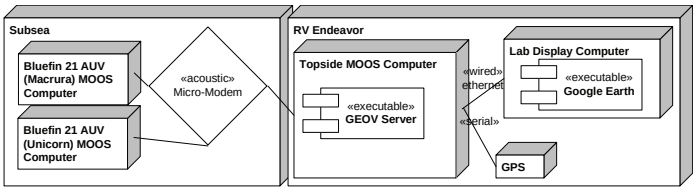


Figure A.7: Network structure for the SWAMSI09 experiment in Panama City, FL.

our experiments, as well a number of test messages. The messages used in field trials are summarized in Table 2.10. They are broken into three rough categories: Data, Command, and Collaboration. Data messages are sent from the vehicles to the topside operator with some type of measured or calculated data. Commands are sent to change the mission state of the vehicles. Collaboration messages are sent between robots to provide data or handshaking relevant to a multi-vehicle experiment.

A.3.3 Network examples

Figure A.7 shows the network connectivity for a simple two-AUV / no-ASC experiment (SWAMSI09). The topside (through a buoy) is connected to the vehicles by an acoustic network using the WHOI Micro-Modem. A wifi network (not shown) is used to upload configuration and code changes and download data logs, but this is primarily done before the day's operations. Once the vehicle is underway, the vehicles are deployed and redeployed through the acoustic network alone.

Figure A.8 gives the network connectivity for a larger experiment with both surface and subsea nodes (GLINT08/09). This network allows forwarding of messages through mobile gateways (the ASCs) and visualization of data shoreside via the internet and GEOV.

A.4 Google Earth interface for Ocean Vehicles (GEOV)

Visual feedback for the AUV operators is provided through a set of continually updated Keyhole Markup Language (KML) files in Google Earth. This Google Earth Interface for Ocean Vehicles (GEOV) provides a database for vehicle positions, speeds, headings and depths as well as a set of add-on modules for displaying other types of data (e.g, an operational grid, target track lines). GEOV is a client/server architecture (see Figure A.9) where the user configures what data he or she wishes to display from a web browser. Then any number of clients running Google Earth can see these data. GEOV has three

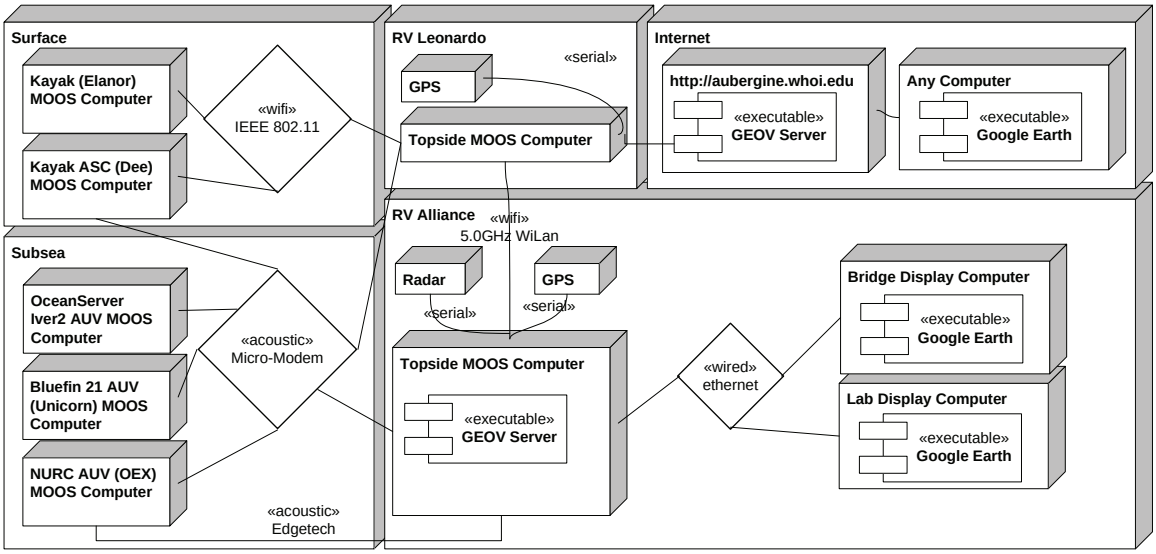


Figure A.8: Collective network structure of the GLINT08 and GLINT09 experiments south of Elba, Italy. GLINT08 did not have the *RV Leonardo*-to-Internet connection in place and GLINT09 did not have the *Unicorn* AUV present.

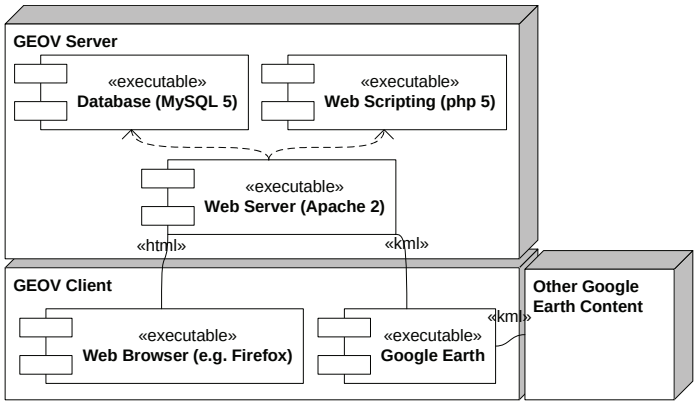


Figure A.9: Model of the client/server interaction for the Google Earth interface for Ocean Vehicles (GEOV).

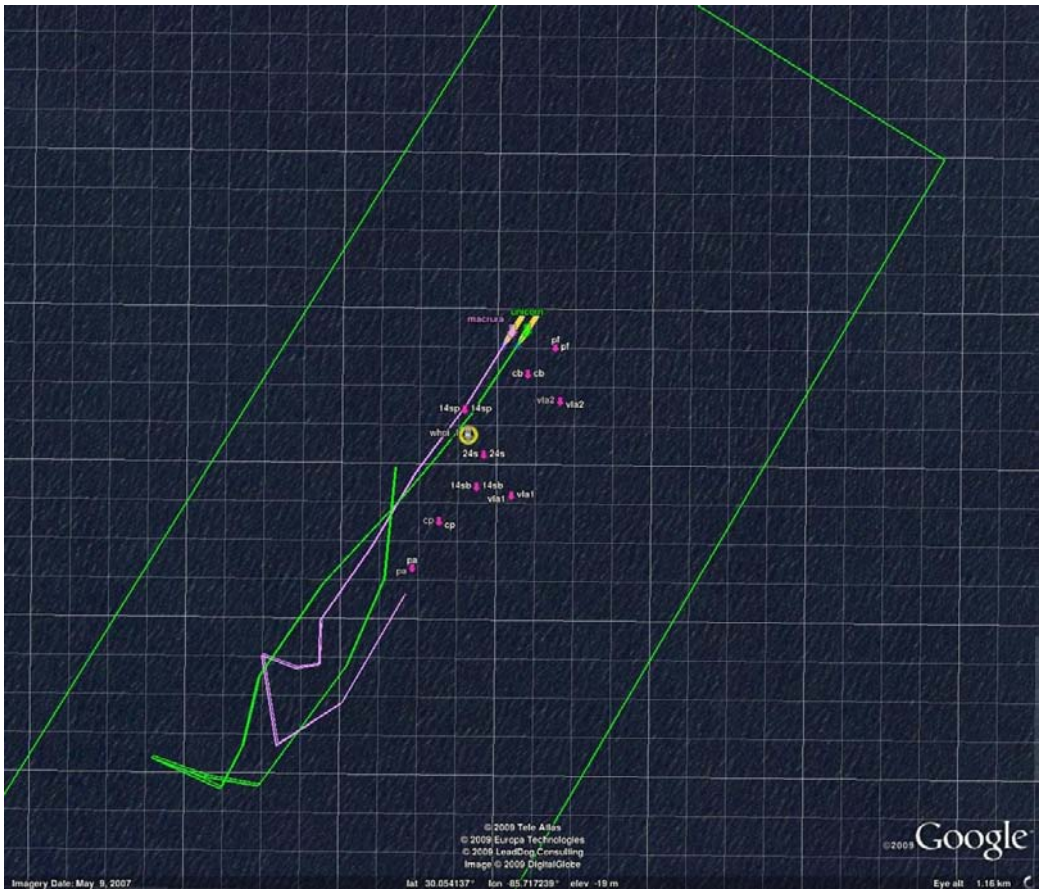


Figure A.10: Screenshot of GEOV from the SWAMSI09 experiment. Bluefin 21 AUVs *Unicorn* and *Macrura* are performing a synchronized racetrack maneuver for bistatic acoustics (both vehicles have sources and nose arrays). This synchronized behavior was commanded using the LAMSS_DEPLOY DCCL message and is autonomously coordinated using the LAMSS_STATUS message. The history of the vehicles is also provided by the LAMSS_STATUS message. The subsea targets are represented by purple arrows and each grid box is 25 meters square. The solid colored lines indicate the position history of each vehicle, while the name and icon show the current location.

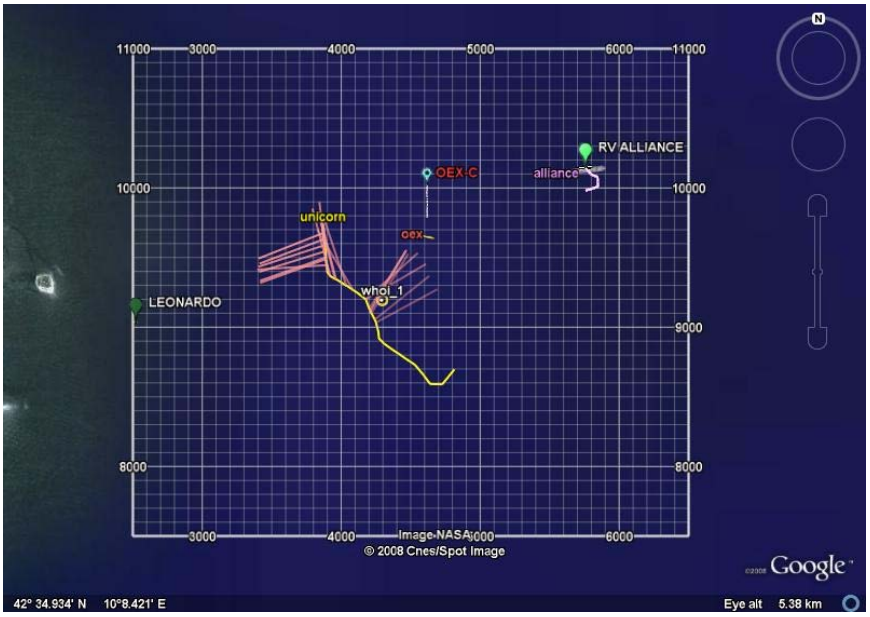


Figure A.11: Target tracking by the AUV *Unicorn* at the GLINT08 experiment visualized using GEOV. The pink lines indicate the direction of a detected source (the *RV Leonardo* was towing an acoustic source). These contact data are provided from the vehicles acoustically by the DCCL LAMSS_CONTACT message. Vehicles in capital letters were merged onto the GEOV display using AIS reports through a separate data feed.

modes of operation:

1. Realtime: displays up-to-date position and position history for vehicles selected. For an example from the SWAMSI09 experiment, see Figure A.10.

2. Playback: similar to the realtime display but for a chosen period of time in the past. Also allows for playback at faster than real speed.

3. History: displays the vehicle tracks for a period of time in the past. This differs from playback in that the history is incorporated into the KML file information, allowing history files to be saved and used separately from the GEOV server. The other two modes (realtime and playback) require a continuous connection to the GEOV server.

Google Earth was chosen due to its ubiquity, availability of continuously updated satellite data, and ease of meshing data from numerous sources. On collaborative experiments, it is possible to display other KML data along with GEOV data, allowing several research groups to work off the same display with minimal integration effort. In the field

experiment SQUINT08 (May 2008: La Spezia, SP, Italy), data from a sidescan sonar on a surface craft were displayed using the manufacturer's KML parser on top of GEOV position data. In GLINT08 (Pianosa Island, LI, Italy), Automatic Identification System (AIS) data for nearby ships was displayed using a collaborator's Perl script to convert AIS to KML, again along with the GEOV data (see Figure A.11 for an example).

The ability to have any number of clients for a given GEOV server allows individual scientists to visualize data on their own laptops while not disturbing the main lab display. Furthermore, AUV operations requires significant interaction with the bridge of the research vessel. Having a GEOV display on the bridge gives the officers a much better sense of what is happening underwater and allows for safer and more efficient operations.

A.5 Summary

In summary, the authors have found through numerous field trials that successful command and control of multiple research marine vehicles requires several components which are addressed by the `Unified C2` architecture:

- **Abstraction of the differences between vehicle types** (each manufacturer is different) - This is accomplished through the "frontseat"-"backseat" operation paradigm. After this abstraction, to the system designer, each vehicle is similar: a MOOS community running `pHelmIvP`.

- **Hierarchical configuration** - Changes in configuration must be propagated throughout the network of vehicles, ideally by making a change in a single location. By organizing the vehicle configuration into a hierarchy, much redundant configuration is removed.

- **Sufficiently robust but flexible communications** - Acoustic communications are often unreliable due to the physical constraints of the medium. To compensate for this reality, the subsea network can be connected to the more robust surface network by means of mobile gateways (surface craft). Very small messages are provided by DCCL, whose XML configuration allows for rapid reconfiguration when new data or commands need to be sent. Time-varying priorities allow for messages to be sent based on importance, while allowing for all message types to have some access to the communications channel.

- **Meshed visualization** - GEOV allows for visualization of vehicle data for runtime performance evaluation and safety. Google Earth allows easy meshing of data from multiple sources, allowing GEOV to interface with data from collaborators who do not need to be running the same `MOOS-IvP` infrastructure and display everything on a single display.

Together these elements fight the ever increasing complexity inherent in multi-vehicle robotic systems to create a manageable, yet readily reconfigurable, system.

B Goby-Acomms Details

B.1 Goby1 DCCL XML Specification

Version 1 of DCCL used an eXtensible Markup Language (XML) language definition. The current (version 2) definition of DCCL is in Google Protocol Buffers option extensions (see Table 2.2).

The full XML schema is available with the source code at <`http://launchpad.net/goby/1.1`>; here we give a summary of the tags. A DCCL message file always consists of the root tag <`message_set`> which has one or more <`message`> tags as its children. The <`message`> children are as follows:

- <`id`>: an identification number (9 bits, so <`id`> $\in [0, 511]$) representing this message to all decoding nodes [`unsigned integer`].
- <`name`>: a name for the message. This tag and <`id`> must *each* be a unique identifier for this message. [`string`].
- <`size`>: the maximum size of this message in bytes [`unsigned integer`]. DCCL may produce a smaller message, but will not validate this message XML file if it

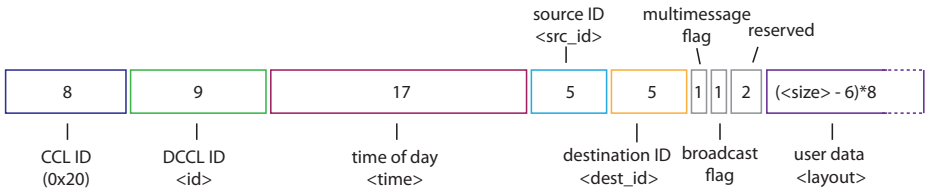


Figure B.1: Layout of the DCCL (version 1) header, showing the fixed size (in bits) of each header field. The user cannot modify the size of these header fields, but can access and set the data inside through the same methods used for the customizable data fields specified in <`layout`>. The multimessage and broadcast flags are not used by DCCL, but are included for use by priority queuing (see Section 2.3).

exceeds this size.

- <repeat>: empty tag that can be specified to tell DCCL to repeatedly create the entire message to fill the entire <size> of the message.
- <header>: the children of this tag allow the user to rename the header parts of the DCCL message. See Fig. B.1 for a sketch of the DCCL header format. These names are used when passing values at encode time for the various header fields.
  - <time>: seconds elapsed since 1/1/1970 ("UNIX time"). In the DCCL encoding, this reduced to seconds since the start of the day, with precision of one second. Upon decoding, assuming the message arrives within twelve hours of its creation, it is properly restored to a full UNIX time.
    - <name>: the name of this field; optional, the default is "_time". [string]
  - <src_id>: a unique address (<src_id> $\in [0, 31]$) of the sender of this message. For a given experiment these short unique identifiers can be mapped on to more global keys (such as vehicle name, type, ethernet MAC address, etc.).
    - <name>: default is "_src_id". [string]
  - <dest_id>: the eventual destination of this message (also an unsigned integer in the range [0,31]). If this destination exists on the same subnet as the sender, this will also be the hardware layer destination id number.
    - <name>: default is "_dest_id". [string]
- <layout>: the children of this tag define the generic data fields of the message, which can be drawn from any combination of the following types:
  - <bool>: a boolean value.
    - <name>: the name of this field. [string]
    - <array_length>: optional; specifying this makes this field an array of bool instead of a single bool [unsigned integer].
  - <int>: a bounded integer value.
    - <name>: see <bool><name>.
    - <max>: the maximum value this field can take. [real number].
    - <min>: the minimum value this field can take. [real number].

    * <max_delta>: gives the maximum value for the difference of delta fields when using delta-difference encoding. Optional; the use of this tag enables delta-differencing encoding. This feature is explained where it is motivated in as part of the CHAMPLAIN09 experiment in section 2.6.5 [real number].
    * <array_length>: see <bool><array_length>.
- <float>: a bounded real number value.
    * <name>: see <bool><name>.
    * <max>: see <int><max>.
    * <min>: see <int><min>.
    * <max_delta>: see <int><max_delta>.
    * <precision>: specifies the number of decimal digits to preserve. For example, a precision of "2" causes 1042.1234 to be rounded to 1042.12; a precision of "-1" rounds 1042.1234 to 1.04e3. [integer].
    * <array_length>: see <bool><array_length>.
- <string>: a fixed length string value.
    * <name>: see <bool><name>.
    * <max_length>: the length of the string value in this field. Longer strings are truncated. <max_length>4</max_length> means "ABCDEFG" is sent as "ABCD". [unsigned integer].
    * <array_length>: see <bool><array_length>.
- <enum>: an enumeration of string values.
    * <name>: see <bool><name>.
    * <value>: a possible value the enumeration can take. Any number of values can be specified.
    * <array_length>: see <bool><array_length>. [string].
- <hex>: a pre-encoded hexadecimal value.
    * <name>: see <bool><name>.
    * <num_bytes>: the number of bytes for this field. The string provided should have twice as many characters as <num_bytes> since each character of a hexadecimal string is one nibble (4 bits or $\frac{1}{2}$ byte). [unsigned integer].

Bibliography

[1] C. Kunz, C. Murphy, H. Singh, C. Pontbriand, R. A. Sohn, S. Singh, T. Sato, C. Roman, K.-i. Nakamura, M. Jakuba, R. Eustice, R. Camilli, and J. Bailey, "Toward extraplanetary under-ice exploration: Robotic steps in the arctic," *Journal of Field Robotics*, vol. 26, no. 4, p. 411–429, 2009. [Online]. Available: http://onlinelibrary.wiley.com.libproxy.mit.edu/doi/10.1002/rob.20288/abstract

[2] R. Camilli, C. M. Reddy, D. R. Yoerger, B. A. S. V. Mooy, M. V. Jakuba, J. C. Kinsey, C. P. McIntyre, S. P. Sylva, and J. V. Maloney, "Tracking hydrocarbon plume transport and biodegradation at deepwater horizon," *Science*, vol. 330, no. 6001, pp. 201–204, Oct. 2010. [Online]. Available: http://www.sciencemag.org/content/330/6001/201

[3] K. Cockrell and H. Schmidt, "Robust passive range estimation using the waveguide invariant," *The Journal of the Acoustical Society of America*, vol. 127, p. 2780, 2010.

[4] M. Purcell, D. Gallo, G. Packard, M. Dennett, M. Rothenbeck, A. Sherrell, and S. Pascaud, "Use of REMUS 6000 AUVs in the search for the air france flight 447," in *OCEANS 2011*, Sep. 2011, pp. 1 –7.

[5] WHOI Media Relations Office, "Scientists find part of New Zealand's submerged "pink terraces"," News Release, February 2011. [Online]. Available: http://www.whoi.edu/main/news-releases/2011?tid=3622&cid=89648

[6] D. Anguita, D. Brizzolara, and G. Parodi, "Building an underwater wireless sensor network based on optical communication: Research challenges and current results," in *Sensor Technologies and Applications, 2009. SENSORCOMM'09. Third International Conference on*, 2009, p. 476–479. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5210866

[7] M. Doniec, I. Vasilescu, M. Chitre, C. Detweiler, M. Hoffmann-Kuhnt, and D. Rus, "Aquaoptical: A lightweight device for high-rate long-range underwater

point-to-point communication," in *OCEANS 2009, MTS/IEEE Biloxi-Marine Technology for Our Future: Global and Local Challenges*, 2009, p. 1–6. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5422200

[8] N. Farr, A. Bowen, J. Ware, C. Pontbriand, and M. Tivey, "An integrated, underwater optical /acoustic communications system," in *OCEANS 2010 IEEE - Sydney*, May 2010, pp. 1 –6.

[9] C. Clay and H. Medwin, *Acoustical oceanography: Principles and applications*. John Wiley & Sons, Inc., 1977.

[10] A. Baggeroer, "Acoustic telemetry–An overview," *IEEE J. Ocean. Eng.*, vol. 9, no. 4, pp. 229–235, 1984.

[11] D. Kilfoyle and A. Baggeroer, "The state of the art in underwater acoustic telemetry," *IEEE J. Ocean. Eng.*, vol. 25, no. 1, pp. 4–27, 2000.

[12] J. Preisig, "Acoustic propagation considerations for underwater acoustic communications network development," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 11, no. 4, pp. 2–10, 2007.

[13] M. Stojanovic, "Recent advances in high-speed underwater acoustic communications," *IEEE J. Ocean. Eng.*, vol. 21, no. 2, pp. 125–136, 1996.

[14] M. Chitre, S. Shahabudeen, and M. Stojanovic, "Underwater acoustic communications and networking: Recent advances and future challenges," *The State of Technology in 2008*, vol. 42, no. 1, pp. 103–114, 2008.

[15] J. Partan, J. Kurose, and B. N. Levine, "A survey of practical issues in underwater networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 11, no. 4, pp. 23–33, 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1347372

[16] M. A. Ainslie and J. G. McColm, "A simplified formula for viscous and chemical absorption in sea water," *The Journal of the Acoustical Society of America*, vol. 103, p. 1671, 1998. [Online]. Available: http://link.aip.org/link/jasman/v103/i3/p1671/s1

[17] C. Chen and F. Millero, "Speed of sound in seawater at high pressures," *J. Acoust. Soc. Am*, vol. 62, no. 5, pp. 1129–1135, 1977.

[18] J. C. Preisig and G. B. Deane, "Surface wave focusing and acoustic communications in the surf zone," *The Journal of the Acoustical Society of America*, vol. 116, p. 2067, 2004. [Online]. Available: http://link.aip.org/link/jasman/v116/i4/p2067/s1/html

[19] M. Stojanovic, "OFDM for underwater acoustic communications: Adaptive synchronization and sparse channel estimation," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, 2008, p. 5288–5291. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4518853

[20] M. Grund, L. Freitag, J. Preisig, and K. Ball, "The PLUSNet underwater communications system: acoustic telemetry for undersea surveillance," in *OCEANS 2006*, 2006, p. 1–5. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4099155

[21] R. P. Stokey, "A compact control language for autonomous underwater vehicles," WHOI, Tech. Rep. Public Release 1.0, 2005. [Online]. Available: http://acomms.whoi.edu/ccl/

[22] J. Rice, "SeaWeb acoustic communication and navigation networks," in *Proceedings of the International Conference on Underwater Acoustic Measurements: Technologies and Results*, 2005. [Online]. Available: http://dtnrg.org/docs/papers/UAMeasurements2005Rice2.pdf

[23] J. Rice and D. Green, "Underwater acoustic communications and networks for the US navy's seaweb program," in *Second International Conference on Sensor Technologies and Applications, 2008. SENSORCOMM '08*, Aug. 2008, pp. 715 –722.

[24] International Telecommunication Union, "Information technology - open systems interconnection - basic reference model: The basic model," Internation Telecommunication Union, Tech. Rep. X.200, 1994.

[25] M. Stojanovic, "On the relationship between capacity and distance in an underwater acoustic communication channel," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 11, no. 4, pp. 34–43, 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1347373

[26] M. Porter and Y. Liu, "Finite-element ray tracing," in *Proc. Int. Conf. on Theoretical Comp. Acoust*, vol. 2, pp. 947–956.

[27] M. R. Benjamin, H. Schmidt, P. M. Newman, and J. J. Leonard, "Nested Autonomy for Unmanned Marine Vehicles with MOOS-IvP," *Journal of Field Robotics*, vol. 27, no. 6, pp. 834–875, November/December 2010.

[28] R. Zimmerman, G. D'Spain, and C. Chadwell, "Decreasing the radiated acoustic and vibration noise of a mid-size AUV," *IEEE Journal of Oceanic Engineering*, vol. 30, no. 1, pp. 179 – 187, Jan. 2005.

[29] S. Petillo, H. Schmidt, and A. Balasuriya, "Constructing a distributed AUV network for underwater plume-tracking operations," *International Journal of Distributed Sensor Networks*, vol. 2012, 2012.

[30] A. Shafer, "Autonomous cooperation of heterogeneous platforms for sea-based search tasks," Master's thesis, Massachusetts Institute of Technology, 2008.

[31] H. Zimmermann, "OSI reference model–The ISO model of architecture for open systems interconnection," *Communications, IEEE Transactions on*, vol. 28, no. 4, pp. 425–432, 2002.

[32] G. Booch, J. Rumbaugh, and I. Jacobson, "The unified modeling language," *Unix Review*, vol. 14, no. 13, p. 5, 1996.

[33] S. Basagni, C. Petrioli, R. Petroccia, and M. Stojanovic, "Choosing the packet size in multi-hop underwater networks," *Proceedings of IEEE OCEANS 2010*, p. 24–27, 2010. [Online]. Available: http://dandelion-patch.mit.edu/people/millitsa/resources/pdfs/oc10-roberto1.pdf

[34] Google, "Protocol buffers: Developer guide." [Online]. Available: http://code.google.com/apis/protocolbuffers/docs/overview.html

[35] A. Bradley, M. Feezor, H. Singh, and F. Yates Sorrell, "Power systems for autonomous underwater vehicles," *Oceanic Engineering, IEEE Journal of*, vol. 26, no. 4, pp. 526–538, 2001.

[36] J. Daemen and V. Rijmen, "AES proposal: Rijndael," *AES Proposal*, 1999. [Online]. Available: http://www.cryptosoft.de/docs/Rijndael.pdf

[37] "Secure hash signature standard," NIST, Tech. Rep. FIPS PUB 180-2, 2002. [Online]. Available: http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

[38] W. Dai, "Crypto++ library 5.6.0." [Online]. Available: http://www.cryptopp.com/

[39] R. P. Stokey, L. E. Freitag, and M. D. Grund, "A compact control language for AUV acoustic communication," *Oceans 2005-Europe*, vol. 2, p. 1133–1137, 2005.

[40] R. Martins, P. Dias, E. Marques, J. Pinto, J. Sousa, and F. Pereira, "IMC: A communication protocol for networked vehicles and sensors," in *OCEANS 2009-EUROPE*. IEEE, 2009.

[41] T. Welch, "Technique for high-performance data compression." *Computer*, vol. 17, no. 6, pp. 8–19, 1984.

[42] D. Huffman, "A method for the construction of minimum-redundancy codes," *Resonance*, vol. 11, no. 2, pp. 91–99, 2006.

[43] J. Larmouth, *ASN.1 Complete*. Elsevier, 2000. [Online]. Available: http://www.oss.com/asn1/larmouth.html

[44] S. Webster, R. Eustice, C. Murphy, H. Singh, and L. Whitcomb, "Toward a platform-independent acoustic communications and navigation system for underwater vehicles," in *OCEANS 2009, MTS/IEEE Biloxi - Marine Technology for Our Future: Global and Local Challenges*, oct 2009.

[45] L. Freitag, M. Grund, C. von Alt, R. Stokey, and T. Austin, "A shallow water acoustic network for mine countermeasures operations with autonomous underwater vehicles," *Underwater Defense Technology (UDT)*, 2005.

[46] R. Eustice, L. Whitcomb, H. Singh, and M. Grund, "Experimental results in synchronous-clock one-way-travel-time acoustic navigation for autonomous underwater vehicles," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 4257–4264.

[47] M. R. Benjamin, "The MOOS-IvP uField Toolbox for Multi-Vehicle Operations and Simulation," Massachusetts Institute of Technology, Tech. Rep. 12.2, 02 2012.

[48] T. Schneider and H. Schmidt, "Unified command and control for heterogeneous marine sensing networks," *Journal of Field Robotics*, 2010.

[49] NIMA, "Department of defense world geodetic system 1984: Its definition and relationships with local geodetic systems. second edition, amendment 1," NIMA,

Tech. Rep. TR8350.2, 2000. [Online]. Available: http://earth-info.nga.mil/GandG/publications/tr8350.2/wgs84fin.pdf

[50] C. Murphy, "Progressively communicating rich telemetry from autonomous underwater vehicles via relays," Ph.D. dissertation, Massachusetts Institute of Technology and Woods Hole Oceanographic Institution, 2012.

[51] Y. Zhang, M. Godin, J. Bellingham, and J. Ryan, "Using an autonomous underwater vehicle to track a coastal upwelling front," *IEEE Journal of Oceanic Engineering*, vol. 37, no. 3, pp. 338 –347, Jul. 2012.

[52] S. Petillo, A. Balasuriya, and H. Schmidt, "Autonomous adaptive environmental assessment and feature tracking via autonomous underwater vehicles," in *OCEANS 2010 IEEE - Sydney*, May 2010, pp. 1 –9.

[53] A. Baggeroer, "An overview of acoustic communications from 2000-2012," in *Underwater Communications: Channel Modelling & Validation*, 2012.

[54] M. Chitre, personal communication, UComms 2012 conference, 2012.

[55] I. F. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: research challenges," *Ad Hoc Networks*, vol. 3, no. 3, pp. 257–279, 2005.

[56] C. Murphy and H. Singh, "Human-guided autonomy for acoustically tethered underwater vehicles," in *OCEANS 2008*, Sep. 2008, pp. 1 –8.

[57] M. Koegel and M. Mauve, "On the spatio-temporal information content and arithmetic coding of discrete trajectories," *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pp. 13–24, 2012. [Online]. Available: http://www.springerlink.com/index/P1032Q10R1542638.pdf

[58] D. Feldman, C. Sung, and D. Rus, "The single pixel GPS: learning big data signals from tiny coresets," in *Proc. 20th ACM International Conference on Advances in Geographic Information Systems*, 2012.

[59] A. Civilis, C. Jensen, and S. Pakalnis, "Techniques for efficient road-network-based tracking of moving objects," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 5, p. 698–712, 2005.

[60] N. Trawny, S. I. Roumeliotis, and G. B. Giannakis, "Cooperative multi-robot localization under communication constraints," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, 2009, pp. 4394–4400. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5152606

[61] E. D. Nerurkar and S. I. Roumeliotis, "Asynchronous multi-centralized cooperative localization," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2010, pp. 4352–4359. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5650133

[62] A. Bahr, "Cooperative localization for autonomous underwater vehicles," Ph.D. dissertation, Massachusetts Institute of Technology, 2009.

[63] M. Fallon, G. Papadopoulos, and J. Leonard, "Cooperative AUV navigation using a single surface craft," in *Field and Service Robotics*, 2010, pp. 331–340. [Online]. Available: http://www.springerlink.com/index/E82JH73176245368.pdf

[64] E. R. B. Marques, J. Pinto, S. Kragelund, P. S. Dias, L. Madureira, A. Sousa, M. Correia, H. Ferreira, R. Goncalves, and R. Martins, "AUV control and communication using underwater acoustic networks," in *OCEANS 2007-Europe*, 2007. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4302469

[65] A. Rajala, M. O'Rourke, and D. B. Edwards, "AUVish: an application-based language for cooperating AUVs," in *OCEANS 2006*, 2006. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4098924

[66] C. Murphy and H. Singh, "Rectilinear coordinate frames for deep sea navigation," in *Autonomous Underwater Vehicles (AUV), 2010 IEEE/OES*. IEEE, 2010.

[67] X. Rong Li and V. Jilkov, "Survey of maneuvering target tracking. part i. dynamic models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1333 – 1364, Oct. 2003.

[68] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[69] S. Rao, "Modified gain extended kalman filter with application to bearings-only passive manoeuvring target tracking," in *Radar, Sonar and Navigation, IEE Proceedings-*, vol. 152, 2005, p. 239–244.

[70] R. Lum and H. Schmidt, "Exploiting adaptive processing and mobility for multi-static tracking by AUV networks," in *Proceedings of 4th International Conference on Underwater Acoustic Measurements: Technologies and Results*, Kos, Greece, Jun. 2011.

[71] M. Blain, S. Lemieux, and R. Houde, "Implementation of a ROV navigation system using acoustic/Doppler sensors and kalman filtering," in *OCEANS 2003. Proceedings*, vol. 3, 2003, p. 1255–1260.

[72] D. Loebis, R. Sutton, J. Chudley, and W. Naeem, "Adaptive tuning of a kalman filter via fuzzy logic for an intelligent AUV navigation system," *Control Engineering Practice*, vol. 12, no. 12, pp. 1531–1539, Dec. 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0967066103002582

[73] S. E. Webster, R. M. Eustice, H. Singh, and L. L. Whitcomb, "Advances in single-beacon one-way-travel-time acoustic navigation for underwater vehicles," *International Journal of Robotics Research*, vol. 31, no. 8, p. 935–950, Jul. 2012.

[74] 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.

[75] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, 1987. [Online]. Available: http://dl.acm.org/citation.cfm?id=214771

[76] K. Sayood, *Introduction to Data Compression*. Elsevier, Dec. 2005.

[77] L. Freitag, M. Grund, S. Singh, J. Partan, P. Koski, and K. Ball, "The WHOI Micro-Modem: an acoustic communications and navigation system for multiple platforms," in *IEEE Oceans Conference*, 2005.

[78] M. B. Porter, "The acoustics toolbox." [Online]. Available: http://oalib.hlsresearch.com/Modes/AcousticsToolbox/

[79] H. Schmidt, "Oases: Ocean acoustic and seismic exploration synthesis." [Online]. Available: http://lamss.mit.edu/lamss/pmwiki/pmwiki.php?n=Site.Oases

[80] R. Brooks, "Intelligence without reason," *Artificial intelligence: critical concepts*, vol. 3, 1991.

[81] T. Schneider, H. Schmidt, T. Pastore, and M. Benjamin, "Cooperative autonomy for contact investigation," in *OCEANS 2010 IEEE-Sydney*. IEEE.

[82] D. Hughes, S. Kemna, M. Hamilton, and R. Been, "Sensible behaviour strategies for AUVs in ASW scenarios," in *OCEANS 2010 IEEE-Sydney*. IEEE, 2010.

[83] D. Eickstedt, M. Benjamin, H. Schmidt, and J. Leonard, "Adaptive control of heterogeneous marine sensor platforms in an autonomous sensor network," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 5514–5521.

[84] S. Petillo, A. Balasuriya, and H. Schmidt, "Autonomous adaptive environmental assessment and feature tracking via autonomous underwater vehicles," in *OCEANS 2010 IEEE-Sydney*. IEEE.

[85] F. Jensen, W. Kuperman, M. Porter, and H. Schmidt, *Computational ocean acoustics*. Springer London, Limited, 2011.

[86] M. B. Porter, "The KRAKEN normal mode program," SACLANT Undersea Research Centre, Tech. Rep., May 2001. [Online]. Available: http://oalib.hlsresearch.com/Modes/kraken.pdf

[87] 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. [Online]. Available: http://dx.doi.org/10.1002/rob.20346

[88] P. Oliveira, A. Pascoal, V. Silva, and C. Silvestre, "Mission control of the Marius AUV: System design, implementation, and sea trials," *International journal of systems science*, vol. 29, no. 10, pp. 1065–1080, 1998.

[89] *Titan PXA270 RISC based PC/104 Single Board Computer Technical Manual*, EuroTech, Ltd. [Online]. Available: http://www.eurotech.com/en/pb.aspx?tab=download&pg=titan

[90] *User Manual: PCM-3363*, Advantech Co., Ltd.

[91] M. Stojanovic, "Underwater acoustic communications: Design considerations on the physical layer," in *Wireless on Demand Network Systems and Services, 2008. WONS 2008. Fifth Annual Conference on*. IEEE, 2008.

[92] L. Freitag, "FHFSK coding and modulation specification," Woods Hole Oceanographic Institution, Tech. Rep. 401003-SPEC, 2005. [Online]. Available: http://acomms.whoi.edu/publications/

[93] L. Freitag and S. Singh, "Compact data layer for acoustic communications," Woods Hole Oceanographic Institution, Tech. Rep. 401002-SPEC, 2004. [Online]. Available: http://acomms.whoi.edu/publications/

[94] W. Li and J. Preisig, "Estimation of rapidly time-varying sparse channels," *Oceanic Engineering, IEEE Journal of*, vol. 32, no. 4, pp. 927–939, 2007.

[95] L. Freitag, M. Stojanovic, S. Singh, and M. Johnson, "Analysis of channel effects on direct-sequence and frequency-hopped spread-spectrum acoustic communication," *Oceanic Engineering, IEEE Journal of*, vol. 26, no. 4, pp. 586–593, 2001.

[96] G. M. Wenz, "Acoustic ambient noise in the ocean: Spectra and sources," *The Journal of the Acoustical Society of America*, vol. 34, no. 12, pp. 1936–1956, 1962. [Online]. Available: http://link.aip.org/link/?JAS/34/1936/1

[97] L. Freitag, M. Stojanovic, D. Kilfoyle, J. Preisig, and M. Stojanovic, "High-rate phase-coherent acoustic communication: A review of a decade of research and a perspective on future challenges," in *Proc. 7th European Conf. on Underwater Acoustics*, 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.115.358&rep=rep1&type=pdf

[98] J. Crowell, "Small AUV for hydrographic applications," in *Proceedings of the IEEE Oceans Conference 2006*, Boston, MA, 2006.

[99] M. R. Benjamin, J. J. Leonard, H. Schmidt, and P. M. Newman, "An overview of MOOS-IvP and a brief users guide to the IvP Helm autonomy software," MIT, Tech. Rep. MIT-CSAIL-TR-2009-028, 2009. [Online]. Available: http://hdl.handle.net/1721.1/45569

[100] L. Freitag, M. Grund, S. Singh, J. Partan, P. Koski, and K. Ball, "The WHOI Micro-Modem: an acoustic communications and navigation system for multiple platforms," in *Proceedings of the IEEE Oceans Conference 2005*, Washington, DC, 2005.

[101] R. Stokey, L. Freitag, and M. Grund, "A Compact Control Language for AUV acoustic communication," in *Proceedings of the IEEE Oceans Conference 2005*, Brest, France, 2005.

[102] T. Schneider and H. Schmidt, “The Dynamic Compact Control Language: A compact marshalling scheme for acoustic communications,” in *Proceedings of the IEEE Oceans Conference 2010*, Sydney, Australia, 2010.

[103] O. Dubuisson and P. Fouquart, *ASN. 1: communication between heterogeneous systems*. Morgan Kaufmann Pub, 2000.