Goby-Acomms version 2: extensible marshalling, queuing, and link layer interfacing for acoustic telemetry

Toby Schneider
MIT/WHOI Joint Program

Henrik Schmidt
MIT Laboratory for Autonomous Marine Sensing Systems
1. Many marine communication links (often acoustic) are extremely rate-limited (as low as 32 bytes / minute) with high latencies (seconds to minutes or more).
1. Marine communication links are low throughput and high latency.

2. Marine robots are increasingly used collaboratively: need to send positions, health, sensor data, commands, state, etc.

3. Research at sea demands robust & reconfigurable software.

How can we design a networking framework that satisfies these constraints?

Modular design with emphasis on efficiency instead of abstraction and where complexity is layered on simplicity.
Why not Internet Protocol (IP)?

Advantages

• Extremely widely adopted
• Extended by stateless (UDP) and connection-based (TCP) protocols.

Disadvantages (IP designed for much higher throughput and lower latency links)

• Header size = $O(10^1)$ bytes = MTU of typical acoustic modems. Little room for data!
• Vast amount of existing IP traffic uses TCP, but TCP is not designed for links for latency $O(10^1)$ seconds.
• Lack of data source “tracking” - leads to inability to do physics based source coding.
What is DCCL?

DCCL is the Dynamic Compact Control Language which is:

1. A structure language for defining *object-oriented* small messages:
   - \(O(10-100)\) bytes
   - suitable for *acoustic modems* (e.g. WHOI Micro-Modem)
   - extension of Google Protocol Buffers (protobuf)
     - compile-time syntax checking & type safety
     - compatible with non-DCCL protobuf messages
     - usable with many common languages (Java, C++, Python, ... )
What is DCCL?

1. A structure language for object-oriented small messages.

2. A set of encoding & decoding algorithms:
   - **efficient**: bounded numbers & unaligned packing
   - **simple**: encoding based on unsigned integers
   - **secure**: provides pre-shared key AES encryption
   - **customizable (v2)**: user-defined encoders (in C++) for any primitive or user-defined type where extra performance is desired.
## DCCL Encoder: Defaults

<table>
<thead>
<tr>
<th>Primitive Type</th>
<th>Size (bits)</th>
<th>Encode</th>
</tr>
</thead>
</table>
| bool           | 2           | \(x_{\text{enc}} = \begin{cases} 
2 & \text{if } x \text{ is true} \\
1 & \text{if } x \text{ is false} \\
0 & \text{if } x \text{ is undefined} 
\end{cases} \) |
| enum           | \(\lceil \log_2 (1 + \sum \epsilon_i) \rceil \) | \(x_{\text{enc}} = \begin{cases} 
i + 1 & \text{if } x \in \{ \epsilon_i \} \\
0 & \text{otherwise} \end{cases} \) |
| string         | length \cdot 8 | ASCII |
| int (all)      | \(\lceil \log_2 (x_{\text{max}} - x_{\text{min}} + 2) \rceil \) | \(x_{\text{enc}} = \begin{cases} 
nint(x - x_{\text{min}}) + 1 & \text{if } x \in [x_{\text{min}}, x_{\text{max}}] \\
0 & \text{otherwise} \end{cases} \) |
| double, float  | \(\lceil \log_2 ((x_{\text{max}} - x_{\text{min}}) \cdot 10^{\text{prec}} + 2) \rceil \) | \(x_{\text{enc}} = \begin{cases} 
nint((x - x_{\text{min}}) \cdot 10^{\text{prec}}) + 1 & \text{if } x \in [x_{\text{min}}, x_{\text{max}}] \\
0 & \text{otherwise} \end{cases} \) |
| bytes          | num_bytes \cdot 8 | \(x_{\text{enc}} = x \) |

Basically: arbitrary bit-sized integers and fixed point real numbers

(analogous to entropy source encoding with uniform probability distribution over a given range)
**DCCL Example: Defaults**

```protobuf
import "goby/common/protobuf/option_extensions.proto";

message CTDMessage {
  option (goby.msg).dccl.id = 102;
  option (goby.msg).dccl.max_bytes = 64;

  required int32 destination = 1 [(goby.field).dccl.max=31,
    (goby.field).dccl.min=0,
    (goby.field).queue.is_dest=true
    (goby.field).dccl.in_head=true];

  required uint64 time = 2 [(goby.field).dccl.codec="_time",
    (goby.field).queue.is_time=true];

  repeated int32 depth = 3 [(goby.field).dccl.max=1000,
    (goby.field).dccl.min=0,
    (goby.field).dccl.max_repeat=10];

  repeated int32 temperature = 4 [(goby.field).dccl.max=40,
    (goby.field).dccl.min=0,
    (goby.field).dccl.max_repeat=10];

  repeated double salinity = 5 [(goby.field).dccl.max=40,
    (goby.field).dccl.min=25,
    (goby.field).dccl.precision=2,
    (goby.field).dccl.max_repeat=10];
}
```

**10 Conductivity-Temperature-Depth samples (CTD), used for collaborative environmental sampling**

Unencoded: 1376 bits  
Protobuf: 648 bits (-53%)  
REMUS CCL: 344 bits (-75%)  
DCCL (default): 270 bits (-80%)

**Common acoustic modem MTUs:**  
256 - 2048 bits
In version 2, custom encoders can be loaded into DCCL.

E.g., arithmetic source coding provides near optimal encoding for a given data model.

AUV position sample (in Cartesian) used in most collaborative missions

message MinimalNodeStatus
{
  option (goby.msg).dccl.id = 21;
  option (goby.msg).dccl.max_bytes = 32;

  required int32 t = 1 [(goby.field).dccl.codec = "_arithmetic",
                        (goby.field).dccl.arithmetic.model = "lamss.time"];
  required int32 x = 2 [(goby.field).dccl.codec = "_arithmetic",
                        (goby.field).dccl.arithmetic.model = "lamss.dist.horizontal"];
  required int32 y = 3 [(goby.field).dccl.codec = "_arithmetic",
                        (goby.field).dccl.arithmetic.model = "lamss.dist.horizontal"];
  required int32 z = 4 [(goby.field).dccl.codec = "_arithmetic",
                        (goby.field).dccl.arithmetic.model = "lamss.dist.vertical"];
}
## DCCL: Related work

<table>
<thead>
<tr>
<th></th>
<th>Multiple languages</th>
<th>Static type/syntax safety</th>
<th>Easily extensible</th>
<th>Custom encoders</th>
<th>Default compression</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Goby DCCL version 2</strong></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>Goby DCCL version 1</strong></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td><strong>Compact Control Language (CCL)</strong></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>Inter-Module Communication (IMC)</strong></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Abstract Syntax Notation One (ASN.1)</strong></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td><strong>Google Protocol Buffers (protobuf)</strong></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Problem:

- No standard API to acoustic modems
- Modems provide useful features beyond strict functionality of a modem (send bytes from point A -> B)

Goby approach (complexity layered on simplicity):

- Define core requirement of modem driver as sending data (simple to implement)
- All other features (ranging pings, LBL, USBL) are extensions to modem driver
- User can ignore extensions if only data transmission is required (simple), and later add in extensions (complex) as needed.
ModemDriver

- can work with any system that can transmit datagrams (including existing systems such as UDP/IP)
- preserves access to unique & advanced hardware features.
Experiments: Case Studies

- SWAMSI09: Bistatic detection of mine-like targets with two AUVs. **Messages**: LAMSS_DEPLOY, LAMSS_STATUS, ACTIVE_CONTACTS, ACTIVE_TRACKS

Operator Display:
AUVs are green / purple track
Experiments: Case Studies

- **GLINT09**: Active detection of submarine-like target with surface craft (gateway), AUV, and source buoy.

  **Messages**: LAMSS_DEPLOY, LAMSS_STATUS, WINCH_CONTROL, SOURCE_ACTIVATION

![Beam-Time Record](image)

**Figure 1**

- **Frame # (~4 seconds)**
- **Bearing (deg)**
- **dB**

Schneider & Schmidt: Goby 2
MCMC 2012 - Arenzano, Italy
Experiments: Case Studies

- **CHAMPLAIN09**: Adaptive sampling of physical data (T, S, c) using an AUV

  **Messages**: LAMSS DEPLOY, LAMSS STATUS, CTDCODEC

  ![](Temp Variation with Depth and Time.png)

  ![](Temp-Depth Profile.png)
Experiments: Case Studies

- GLINT10: Historical “back-fill” of position to provide accurate track of vehicle position.
  - **Messages**: LAMSS_STATUS, LAMSS_CTD, LAMSS_STATUS_FILL_IN
Experiments: Case Studies

SWAMSI11:

• 3 AUVs collaborative multistatic seafloor target detection.

• REMUS LBL navigation.
Experiments: Case Studies

- **BF9_MAY11:**
  Backend for C. Murphy’s sidescan image sending CTD samples for real-time lat / lon slice display:
Experiments: Case Studies

• MBAT11
  Shallow-water mock-up test for deep sea collaborative passive/active target tracking.
Experiments: Case Studies

Tiger 12

Heterogeneous mix of physical links:

- Deep sea mooring <-> Waveglider: WHOI acoustic Micro-Modem
- Waveglider <-> Shore: Iridium satellite (RUDICS)
- Shore <-> Research Vessel: Iridium satellite (OpenPort)
Acknowledgments

- Goby-Developers group
- Office of Naval Research
- L. Freitag and the WHOI Micro-Modem group
- P. Newman / M. Benjamin (MOOS-IvP)
- Field trial support: NATO Undersea Research Centre, NUWC (Newport), NAVSEA (Panama City), Bluefin Robotics, Robotic Marine Systems
- Open source projects used by Goby: Boost, Crypto++, NCurses, ASIO, Google Protobuf