The Dynamic Compact Control Language Version 3

MTS/IEEE OCEANS Conference
Genova, Italy

May 21, 2015

Toby Schneider

Stephanie Petillo

GobySoft, LLC, N. Falmouth, MA, USA

Chris Murphy

Bluefin Robotics Corporation, Quincy, MA, USA

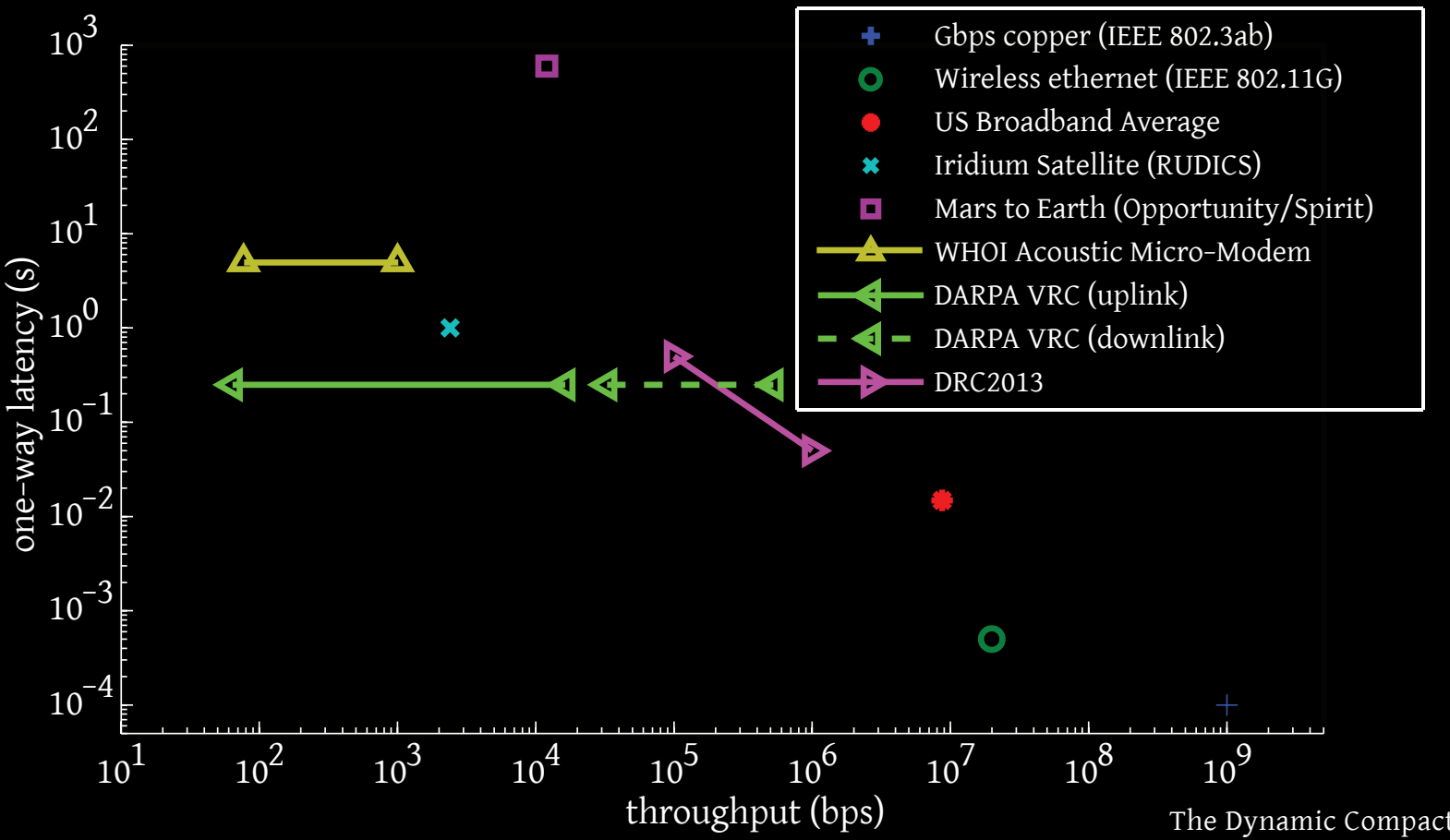
Henrik Schmidt

Massachusetts Institute of Technology, Cambridge, MA, USA

Problem

Robotic platforms are becoming increasingly widely deployed - growing need for data transfer.

Marine links tend to have (extremely) low throughput.



Solutions

Three broad (complementary) approaches to improving marine robotic performance in “poor” link scenarios:

- Improve the throughput and latency of the underlying physical link *(advances in signal processing).*
- Improve the autonomous capabilities of the individual robots *(advances in perception, autonomous control, machine learning).*
- Improve the information value of the data transmitted *(advances in source encoding).* <-- Our focus in this talk

What do *we* want?

- Vehicle position / attitude
  - latitude, longitude, depth, altitude
  - Euler angles: pitch, roll, yaw
- Vehicle health
  - battery remaining
  - sensors' status
- Sensor output
  - scalar data (pressure, temperature, turbidity)
  - imagery (sea floor, sidescan sonar)
  - processed data (beam/time record from sonar)
- Commands
- Shared state (handshaking, mission updates)

All data are not created equal

Clue from physics:

- You will not run into a `double` on the street.
- Sensors measure quantities with dimensions (mass, temperature, salinity, pressure), not *numbers*.
- These quantities are governed by rules, which are often understood, at least probabilistically:
  - Prior measurements in the region (for sensor data)
  - Motion prediction / tracking (for navigation data)
  - Conops requirements (for command messages)

All data are not created equal

Clue from object-oriented programming:

- Group collection of primitive types into sensible unit (object)
  - e.g. CTDMessage, DeployCommand, NodeReport, SonarSettings

Clue from design:

- Perfection is achieved, not when there is nothing more to add, but rather when there is nothing more to take away
  -- Antoine de Saint-Exupéry, *Terre des Hommes* (1939)

DCCL Overiew

The Dynamic Compact Control Language v3 contains:

- an interface description language (IDL) with static (compile-time) units of measure support
- a flexible data marshalling (source encoding/decoding) C++ library (which allows user extension).

Open source (LGPL 2) implementation

(http://libdccl.org).

(Prior releases part of Goby project / DCCL3 is stand-alone)

Interface Description Language (IDL)

```
message CommandMessage
{

 required int32 destination = 1

 optional string description = 2

 enum SonarPower { NOMINAL = 10; LOW = 5; OFF = 0; }
 optional SonarPower sonar_power = 10;
 required double speed = 11


 repeated int32 waypoint_depth = 12


}
```

Base Protobuf message example

DCCL acts as an "invisible" extension to Google Protocol Buffers (Protobuf).

- Code unaware of DCCL can still use the same messages (e.g. internal vehicle data)
- Extensions provide additional information (e.g. numeric bounding).

Protobuf is widely used and provides syntax checking and multi-language support (C++, Python, Java, C, ...)

IDL: Field bounds

```
import "dccl/protobuf/option_extensions.proto";

message CommandMessage
{
 option (dccl.msg) = { id:  125  max_bytes:  32  codec_version: 3 }

 required int32 destination = 1
                [(dccl.field)  = { max: 31 min: 0 in_head: true }];
 optional string description = 2
                [(dccl.field).omit = true];
 enum SonarPower { NOMINAL = 10; LOW = 5; OFF = 0; }
 optional SonarPower sonar_power = 10;
 required double speed = 11
                [(dccl.field) = { units { base_dimensions: "LT^-1" }
                                  max: 2.0  min: -0.5 precision: 1 }];
 repeated int32 waypoint_depth = 12
                [(dccl.field) = { units { base_dimensions: "L" }
                                  max: 40  min: 0 max_repeat: 4 }];
}
```

DCCL message example

Numerics are bounded by

- *max*: largest value
- *min*: smallest value
- *precision*: *decimal* digits of precision preserved.

Message level information

- id: identifies message (CommandMessage == 125)
- max_bytes: enforced upper bound for MTU targetting

IDL: Static units of measure

For scientific data, unit safety is as important as type safety.

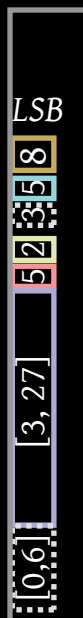
Each numeric field is given a unit system (e.g. SI) and specified dimension (e.g. length, power, speed) or a unit outside a consistent system (e.g. nautical mile).

BASE DIMENSIONS IN DCCL

Physical dimension	Symbol character
length	L
time	T
mass	M
plane angle	A
solid angle	S
current	I
temperature	K
amount	N
luminous intensity	J
information	B
dimensionless	-

Optionally, DCCL produces "unit safe" accessors and mutators for fields with units using the Boost Units library.

Default encoders (sizes)




```
import "dccl/protobuf/option_extensions.proto";

message CommandMessage
{
 option (dccl.msg) = { id:  125  max_bytes:  32  codec_version: 3 }

 required int32 destination = 1
                [(dccl.field)  = { max: 31 min: 0 in_head: true }];
 optional string description = 2
                [(dccl.field).omit = true];
 enum SonarPower { NOMINAL = 10; LOW = 5; OFF = 0; }
 optional SonarPower sonar_power = 10;
 required double speed = 11
                [(dccl.field) = { units { base_dimensions: "LT^-1" }
                                  max: 2.0  min: -0.5 precision: 1 }];
 repeated int32 waypoint_depth = 12
                [(dccl.field) = { units { base_dimensions: "L" }
                                  max: 40  min: 0 max_repeat: 4 }];
}
```

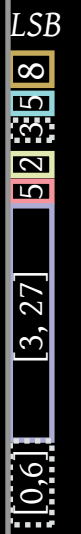
Header Fields:

- 8 bits header (dccl.id)
- 5 bits: destination [0, (2^5-1)]
- (3 bits padding to byte)

Body Fields (can be encrypted):

- 2 bits: sonar_power
- 5 bits: speed
- [3, 27] bits: waypoint_salinity vector (size varies on # of elements)
- ([0, 6] bits padding to byte)

Default encoders (example)




```
import "dccl/protobuf/option_extensions.proto";

message CommandMessage
{
 option (dccl.msg) = { id:  125  max_bytes:  32  codec_version: 3 }

 required int32 destination = 1
                [(dccl.field)  = { max: 31 min: 0 in_head: true }];
 optional string description = 2
                [(dccl.field).omit = true];
 enum SonarPower { NOMINAL = 10; LOW = 5; OFF = 0; }
 optional SonarPower sonar_power = 10;
 required double speed = 11
                [(dccl.field) = { units { base_dimensions: "LT^-1" }
                                  max: 2.0  min: -0.5 precision: 1 }];
 repeated int32 waypoint_depth = 12
                [(dccl.field) = { units { base_dimensions: "L" }
                                  max: 40  min: 0 max_repeat: 4 }];
}
```

Message definition

Example instantiation (x) and encoded values (x_{enc})

x_{enc} (bin)	x_{enc} (dec)	x
11111010	250	id: 125 (CommandMessage)
00011	3	destination: 3
000	(padding)	
10	2	sonar_power: LOW (i = 1)
10001	17	speed: 1.2
100	4 [10 15 10 12]	waypoint_depth: [10, 15, 10, 12]
[001010		
001111		
001010		
001100]		
000000	(padding)	

hex: fa 03 46 2a 8f c2 00
bin: 11111010 00000011 01000110 00101010 10001111 11000010 00000000

Compression performance

	DCCL Performance				
Message	Example Instantiation	GPB Size (bytes)	Python `struct` Size (bytes)	DCCL Size (bytes)	DCCL Compression
CommandMessage	destination: 3, sonar_power: LOW, speed: 1.2, waypoint_depth: [10, 15, 10, 12]	21	15	7	53-67%
AUVStatus	timestamp: 1427316658, source: 1, destination: 2, x: 2326, y: 1100, speed: 1.1, heading: 152.4, depth: 2150, altitude: 100, pitch: 0.01, roll: -0.02, mission_state: SEARCH, depth_mode: DEPTH_BOTTOM_FOLLOWING	90	48	19	60-79%
CTDMessage	temperature: 10, depth: 50, salinity: 32, sound_speed: 1485	29	18	7	61-76%

"GPB" is the default Protobuf encoding

Python *struct* is a formalized version of commonly used ad-hoc encoders (N-byte signed or unsigned integers byte-packed)

DCCL provides compression of 53-79% over these other approaches.

Application Examples

MOOS-IvP / Goby

- Numerous vehicle types (WaveGlider, Bluefin, OceanServer, CMRE OEX)
- Acoustic modems (primarily WHOI Micro-Modem), Iridium RUDICS/ Short-Burst Data
- Sonar data, oceanographic sensors data, command messages, multi-vehicle behavior synch.

TOPICS

- Bluefin vehicles with Iridium Short-Burst Data and/or acoustic modems.
- Command messages / vehicle navigation & status

DARPA Robotics Challenge (humanoid disaster relief robots)