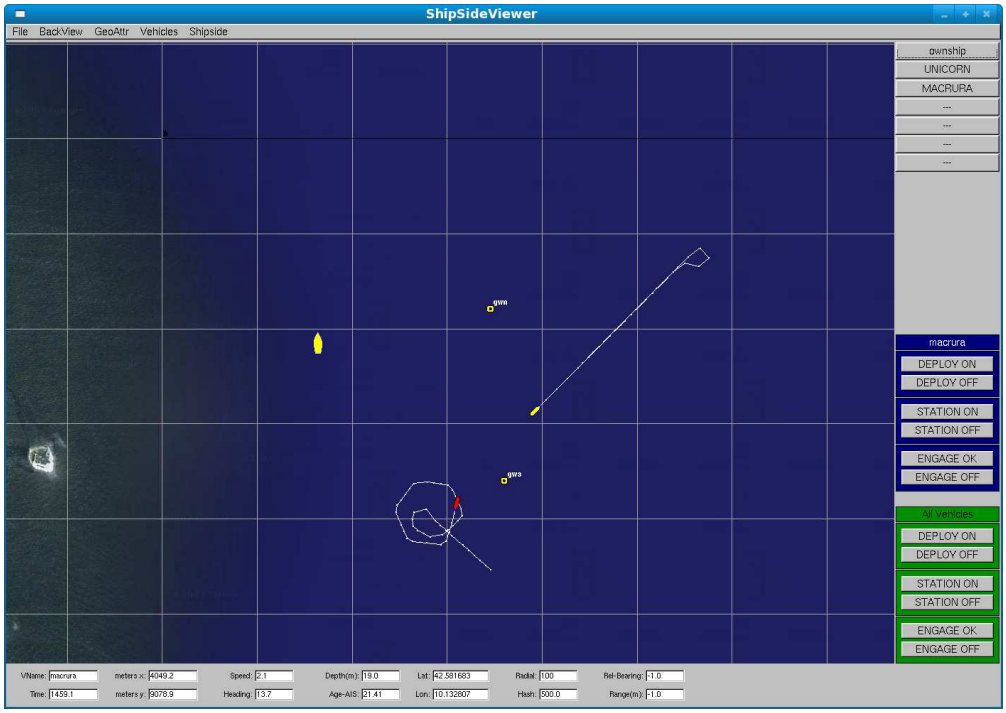

MOOS-IvP

Undersea Network Command and Control Topside

User's Guide

H. Schmidt, K. Cockrell, and T. Schneider

Laboratory for Autonomous Marine Sensing
Department Mechanical Engineering
Massachusetts Institute of Technology, Cambridge MA

December 8, 2008

Abstract

This paper provides a compact User's Guide for the topside MOOS-IvP module suite for autonomous communication, command and control AC^3 for an undersea network with limited communication capacity.

Contents

1 Overview

1.1 Purpose and Scope of this Document

The purpose of this document is to provide a catalog style overview of and user's Guide for the modules used for the MOOS-IvP Topside for command and control of an undersea autonomous sensing network with nested, adaptive and collaborative autonomy. The scope of discussion includes, for each module, a brief description of the module function, authorship, source for download, rough measure of complexity, and module dependencies.

Further, for use by developers of onboard processes and behaviors modules, the description includes a detailed listing and description of MOOS variables published by or subscribed to by each simulator module.

2 Topside Command and Control

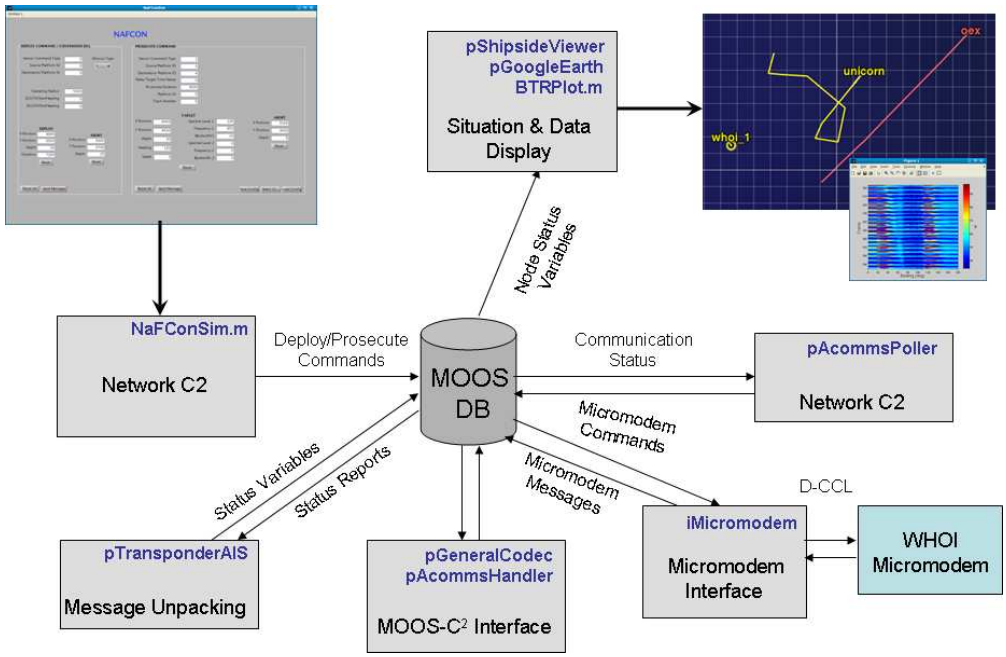

Figure 1: MOOS-IvP community for MIT autonomous network command and control topside.

2.1 Topside MOOS Community

The topside MOOS community provides the interface middle-ware between the field command and control operators and the network communication infrastructure. The topside MOOS community is shown schematically in Fig. 1.

The topside community is similar to the vehicle communities, except for the lack of an autonomy helm. Thus the communication stack is identical to that applied on the vehicles, including the modules `pGeneralCodec`, `pAcommsHandler`, `pAcommsPoller`, and `iMicroModem`.

Since the topside, lacking the helm, only has one state, the state transition manager module `pNaFCon` is not in general used on the topside.It is only used if the topside is broadcasting its own status reports, which may be desirable if operated from a ship, where the navigation information may be needed for the collision avoidance behaviors of the underwater assets.

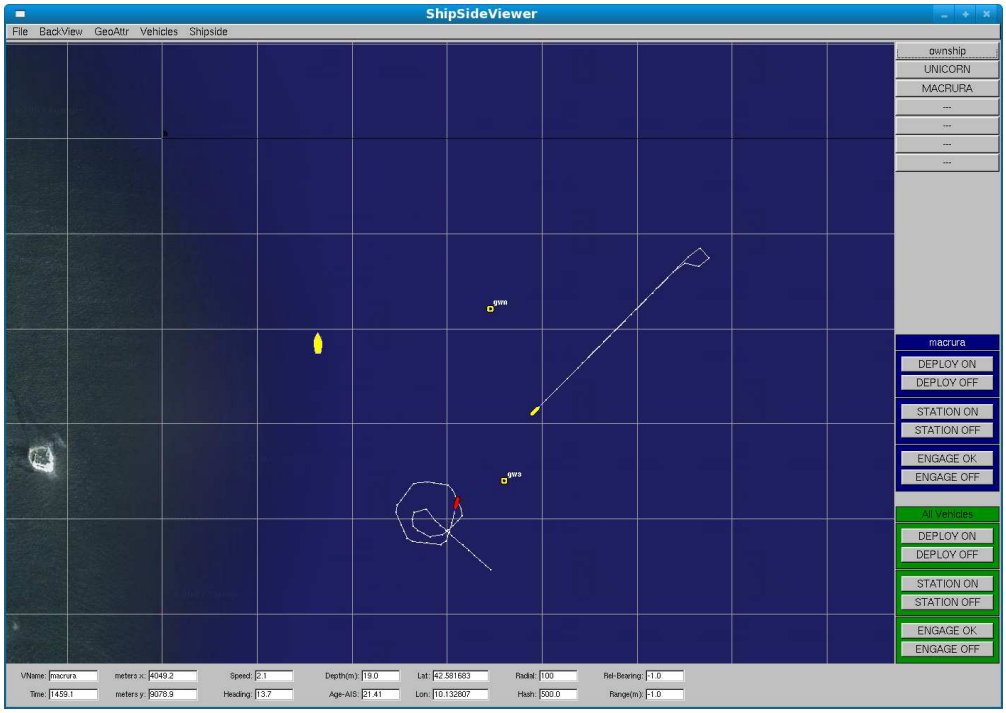

Figure 2: pShipSideViewer topside situational display

The principal interpreter of the incoming messages containing node status and contact information is the standard `pTransponderAIS` process, configured for accepting incoming messages published in the MOOS variable `NAFCON_MESSAGES`. It parses the message into status variables that are compatible with the topside situational display driver, `pShipSideViewer`, which will display the node information geographically as shown in Fig. 2.

Another useful process running on the topside is `pNafconMessageViewer` which shows the complete content of the status and contact reports arriving from each node in the network, with an example from GLINT'08 in Fig. 3.

Finally, dedicated graphical tools are available for displaying measured and processed data received from the nodes, such as BTR and CTD records.

The topside command console is driven by the MATLAB script `NaFConSim.m`, the GUI of which is shown in Fig. 4. The name is an acronym for *Network and Field Control Simulator* because it originates as an MIT simulation of a comprehensive network control console developed by Penn State for the PLUSNet program. The GUI has two sections, one for issuing *Deploy Commands*, the other for issuing *Prosecute Commands*, with each allowing for a number of optional sub-states or mission types. Thus, a deploy may be a hexagonal loiter pattern or a racetrack. Also, deploy missions include environmental sampling missions involving vertical YoYo-s or horizontal zig-zag surveys. Similarly, prosecute missions may be selected to be target-adaptive or simple classical

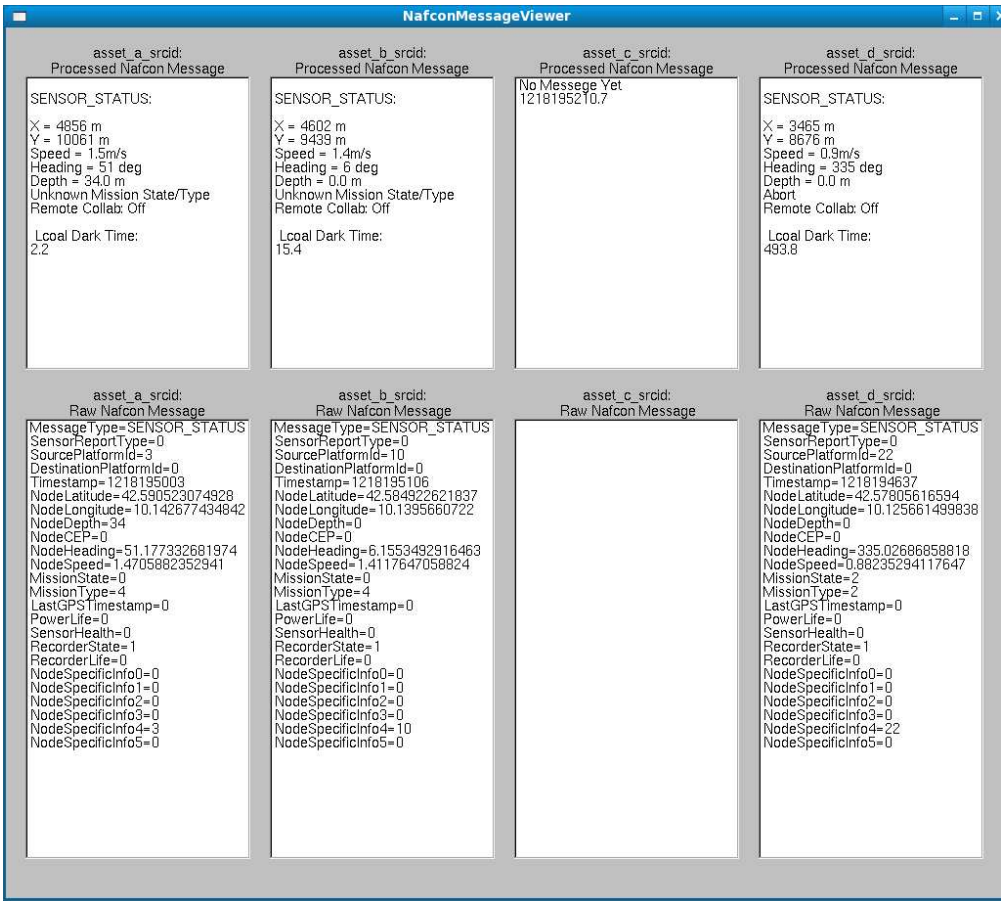

Figure 3: `pNafconMessageViewer` display shows the status reports in completeness, received from the network nodes.

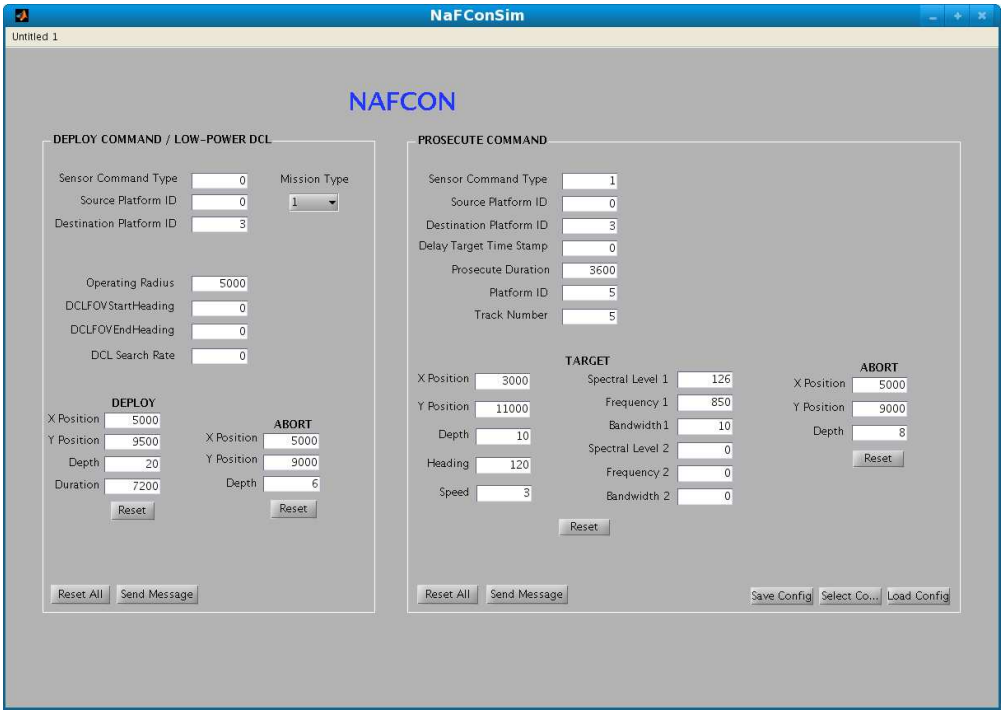

Figure 4: Network and Field Control (NaFCon) GUI for issuing Deploy and Prosecute commands to network nodes.

TMA survey patterns.

A complete list of the MOOS topside modules and utilities is given in Table **??**. A more detailed description is given in the following chapter.

2.2 Launching the Command and Control Topside

The topside situational display and communication infrastructure, a MOOS community AUV_Topside, Port 9123, is launched using the commands:

```
> cd ~/moos-ivp-local/missions/AUV_Topside/
> pAntler AUV_Topside_base.moos &
> pAntler AUV_Topside_CommsStack.moos &
```

This will bring up the situational display using **pShipSideViewer**, with an example shown in Fig. **??**.

The topside command and control GUI, shown in Fig. 4 is typically launched on a separate desktop using the commands:

```
> cd ~/moos-ivp-local/src/matlab/NaFConSimTop
> matlab
  >> NaFConSim
```

#	Module Name	Module Description	Author	Size
1	**NaFConSim.m**	MATLAB GUI for generating commnds to nodes in the autonomous undersea network	Dumortier	
2	**pHuxley**	Front-seat driver interface. Required for publishing of UTM Datum only. *Libraries: mbutil, MOOS, MOOSGen, MOOSUtility*	Balasuriya	
3	**pShipsideViewer**	Situational display *Libraries: mbutil, MOOS, MOOSGen, MOOSUtility*	Benjamin	
4	**uNafconMessageViewer**	Process for displaying messages received from active network nodes. *Libraries: MOOS, MOOSGen, MOOSUtility*	Cockrell	
5	**pGeneralCodec**	Generic module for Coding and Decoding of CCL modem messages *Libraries: MOOS, MOOSGen, MOOSUtility*	Schneider	
6	**pAcommsHandler**	Schedules acomms. *Libraries: mbutil, MOOS, MOOSGen, MOOSUtility*	Schneider	
7	**pAcommsPoller**	Manages and schedules polling of other nodes on the modem network. Usually run on topside only, but may be activated on mobile node for relaying. *Libraries: mbutil, MOOS, MOOSGen, MOOSUtility*	Schneider	
8	**iMicroModem**	Driver process for WHOI micromodems *Libraries: mbutil, MOOS, MOOSGen, MOOSUtility*	Grund	
Total unique lines of code Total aggregate lines of code				

Table 1: MOOS modules for Topside Command and Control of Undersea Network

The GUI will publish the commands to the topside MOOSDB in the topside MOOS community AUV_Topside on Port 9123, and will be broadcast to the virtual network by the modem communication stack.

3 Topside Command and Control Modules

3.1 NaFConSim

3.1.1 Brief Overview

`NaFConSim.m` is a MATLAB GUI which serves as the principal command and control console for the undersea network.

3.1.2 Parameters for the NaFConSim Configuration Block

The following configuration parameters are defined for `NaFConSim`.

`SerialComms:` The `NAFCONSIM` GUI may be configured for either communicating through a serial port to a MOOS community on another computer, or directly to the MOOSDB. The latter is the normal operation, with `SERIALCOMMS = false`.

`MOOSComms:` Must be set to `MOOSComms = true` for communication directly to the MOOSDB. Must always be the negated of `SerialComms`.

`Port:` Serial port used for communication. Only significant for `SERIALCOMMS = true`.

`BaudRate:` Baudrate used for communication via serial port. Only significant for `SERIALCOMMS = true`.

`Verbose:` Verbosity. Only true for testing and dbugging..

`Streaming:` Serial protocol.Only significant for `SERIALCOMMS = true`.

`SERIAL_TIMEOUT:` Timeout for serial communication in seconds .Only significant for `SerialComms = true`.

`SUBSCRIBE:` Identifies MOOS variables to be subscribed to by `NaFConSim`..

Below is an example configuration block for `NaFConSim`,

Listing 3.1 - An example for `pNaFConSim` *configuration block .*

```
1 /////////// Global Variables ////////////////
2 ServerHost = localhost
3 ServerPort = 9123
4 ////////Configuration Block for pNaFConSim ///////////////
5 ProcessConfig = NaFConSim
6 {
7   AppTick     = 10
8   CommsTick   = 10
9   Port     = COM6
10  BaudRate    = 4800
11  Verbose     = false
12  Streaming   = false
13  MOOSComms = true
14  SerialComms = false
15  SERIAL_TIMEOUT = 10.0
16  SUBSCRIBE   = DB_TIME @ 0
17  SUBSCRIBE   = LAT_ORIGIN @ 0
18  SUBSCRIBE   = LONG_ORIGIN @ 0
19 }
```

3.1.3 MOOS variables subscribed to by pNaFConSim:

.

MOOS variable	Type	Description	Published by
DB_TIME	D	UTC time for time stamping of commands	MOOSDB
LAT_ORIGIN	D	Latitude of local UTM Datum	pHuxley
LONG_ORIGIN	D	Longitude of local UTM Datum	pHuxley

3.1.4 MOOS variables published by NaFConSim:

.

MOOS variable	Type	Description	Format
PLUSNET_MESSAGE	$	Human-readable deploy or prosecute message to be coded, scheduled and transmitted by the communication stack.	

3.2 pHuxley

3.2.1 Brief Overview

This module or one of the other front-seat driver interface modules the solely responsible for publishing the local UTM Datum to the MOOSDB, and must therefore be part of the topside community. Since there is no frontseat driver on the topside, `pHuxley` should be operated in the simulation mode. The `pHuxley` process is decribed in detail in Section **??**, and we shall here only describe the configuration particular to the topside use.

3.2.2 Parameters for the Topside pHuxley Configuration Block

Below is an example configuration block for the topside `pHuxley` process,

Listing 3.2 - An example of a topside `pHuxLey` *configuration block .*

```
 1 ProcessConfig = pHuxley
 2 {
 3    AppTick   = 4
 4    CommsTick = 4
 5
 6    VarNamePrefix = GATEWAY
 7 // Huxley IP address & port
 8    HuxleyHost = localhost
 9    HuxleyPort = 29500
10    HuxleySim = true
11    Verbose = true
12 }
```

3.3 pGeneralCoDec

3.3.1 Brief Overview

This MOOS module is the generic Codec uased for in- and outgoing acomms messages, used consistently throughout the network. It is described in detail in Section **??**. The configuration for the coding and decoding of the messages, defined in xml-files, should be set identically on all platforms.

3.4 pAcommsHandler

3.4.1 Brief Overview

This MOOS module is the crucial queueing handler for in- and outgoing acomms messages, used consistently throughout the network. It is described in detail in Section **??**. We shall here only give an example of a configuration block that includes items unique to the topside, such as the queueing andscheduling of *Deploy* and *Prosecute Commands* to be transmitted to the mobile network nodes.

3.4.2 Topside pAcommsHandler Configuration Block

Below is an example configuration block for the topside `pAcommshandler` process, Note that this confguration assumes that the message coding and decoding is performed using the generic CoDec `pGeneralCodec`. If used together with the dedicated CCL CoDec `pFramer`, the configuration should use the `send_CCL` and `receive_CCL` variables instead of the generic ones, as decribed in Section **??**.

Listing 3.3 - An example of a topside `pAcommsHandler` *configuration block .*

```
ProcessConfig = pAcommsHandler
{
   AppTick    = 4
   CommsTick  = 4
   verbosity = verbose
// all case insensitive
   modem_id = 0

// information about iMicroModem
   micromodem_command_var = MICROMODEM_COMMAND
   micromodem_data_var = MICROMODEM_DATA
// Commands
// send = VarName,
                          //     VariableID
                          //        [Ack]
                          //           [BlackoutTime]
                          //              [MaxQueue]
                          //                  [NewestFirst]
                          //                     [Priority]
                          //                         [Priority Time Constant]
   send = OUT_DEPLOY_HEX_30B,    10, 1,  0,  1,   1,  10,  120
   send = OUT_PROSECUTE_HEX_30B, 11, 1,  0,  1,   1,  10,  120
// Reports
// receive = VarName, VariableID
   receive = IN_STATUS_HEX_30B,  12
   receive = IN_CONTACT_HEX_30B, 13
   receive = IN_TRACK_HEX_30B,   14
   receive = IN_CTD_HEX_254B,     3
   receive = IN_BTR_HEX_254B,     4
}
```

3.5 pAcommsPoller

3.5.1 Brief Overview

This MOOS module does a darn good job at doing what it is supposed to do

3.5.2 Configuration MOOS-block

3.5.3 MOOS variables subscribed to by pAcommspoller:

.

3.5.4 MOOS variables published by pAcommsPoller:

.

3.6 iMicroModem

3.6.1 Brief Overview

This MOOS module does a darn good job at doing what it is supposed to do

3.6.2 Configuration MOOS-block

3.6.3 MOOS variables subscribed to by iMicroModem:

.

3.6.4 MOOS variables published by iMicroModem:

.

3.7 uNafconMessageViewer

3.7.1 Brief Overview

This MOOS module displays summarized versions of NaFCon messages, along with the entire original message.

3.7.2 Topside uNafconMessageViewer Configuration Block

uNafconMessageViewer can display messages from up to 4 sources at once. Each source is denoted by its source's identification number, which is in the Nafcon message.

Below is an example configuration block for `uNafconMessageViewer`.

Listing 3.4 - An example of `uNafconMessageViewer` *configuration block .*

```
1 {
2  AppTick   = 5
3  CommsTick = 25
4
5  asset_a_srcid = 1
6  asset_b_srcid = 2
7  asset_c_srcid = 3
8  asset_d_srcid = 4
9 }
```

3.7.3 MOOS variables subscribed to by uNafconMessageViewer:

.

MOOS variable	Type	Description	Published by
NAFCON_MESSAGES	$	Nafcon messages from communications stack	pFramer
LAT_ORIGIN	D	Latitude of local UTM Datum	pHuxley
LONG_ORIGIN	D	Longitude of local UTM Datum	pHuxley

3.7.4 MOOS variables published by uNafconMessageViewer:

None