uFldNodeComms: Simulating Inter-vehicle Communications

June 2018

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139

1 Overview

The `uFldNodeComms` application is a tool for handling node reports and messages between vehicles. Rather than directly sending node reports and messages between vehicles, `uFldNodeComms` acts as an intermediary to conditionally pass a report or message on to another vehicle, where conditions may be the inter-vehicle range or other criteria. The assumption is that `uFldNodeComms` is running on a topside or shoreside computer, and receiving information about the present physical location of deployed vehicles through node reports. The typical layout is shown in Figure 1.

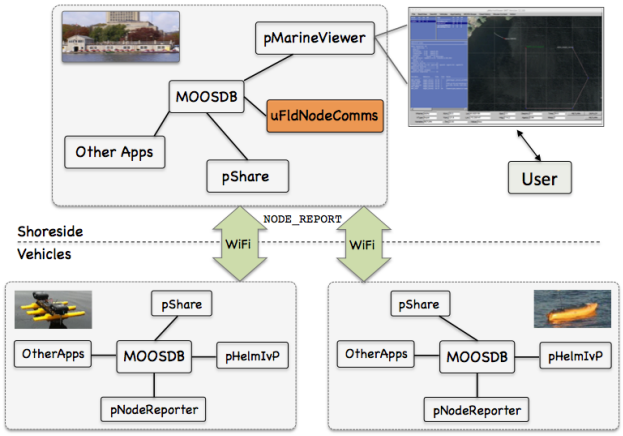

Figure 1: **Typical uFldNodeComms Topology:** A shoreside or topside community is receiving information from several deployed vehicles, in the form of node reports. The node reports contain time-stamped updated vehicle positions, from which the speed and distance measurements are derived and posted to the shoreside MOOSDB.

In short, uFldNodeComms subscribes for incoming node reports for any number of vehicles, and keeps the latest node report for each vehicle. On each iteration, for a each vehicle, if the node report has been updated, the report is published to a specially created MOOS variable for the other n-1 vehicles. A user-configured criteria is applied before publishing the new information. Typically this criteria involves the range between vehicles, but the criteria may be further involved. The idea is conveyed for three vehicles *alpha*, *bravo*, and *charlie*, shown below in Figure 2.

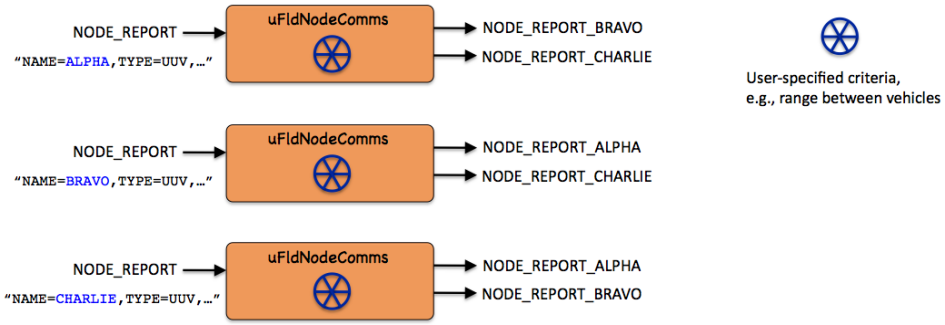

Figure 2: **Brokering by uFldNodeComms:** Each incoming node report is sent out to a specially named variable corresponding to one each of the other n-1 vehicles.

1.1 The Difference Between a Node Report and Node Message

There are two distinct message types in play with uFldNodeComms:

- Node reports - concise vehicle trajectory/state, in NODE_REPORT
- Node messages - open ended inter-vehicle content, in NODE_MESSAGE

A *node report* contains concise information regarding a vehicle's state, including its position, speed, heading and depth for underwater vehicles. It may also include information regarding the helm,

including the current all-stop state and helm mode. It is typically published by pNodeReporter running on each vehicle, and generated by default at 4Hz. This may be faster for higher speed vehicles operating in close proximity, and lower for slower moving vehicles operating in expansive waters. A node report is typically contained in one of two variables, either NODE_REPORT_LOCAL or NODE_REPORT. The former is used by pNodeReporter to indicate the report has been generated locally about ownship. When this information is sent off-board, to the shoreside, uFldNodeComms, and eventually other vehicles, it converted to the variable NODE_REPORT.

A *node message* contains open ended inter-vehicle message content, in the form of (a) a destination vehicle, (b) a MOOS variable name, and (c) the value of the MOOS variable. Not all messages sent will arrive at their destination and uFldNodeComms is the arbiter of which messages get through. There is no single app the generates a node message. Any app, by the configuration of the user, may publish an outgoing message to NODE_MESSAGE_LOCAL. This indicates the message was generated on ownship to be sent elsewhere. In the uField simulation scheme, using uFldNodeComms, it will arrive on the shoreside as NODE_MESSAGE and, if allowed through, will arrive on the destination vehicle(s) as NODE_MESSAGE. The receiving vehicle will unpack the incoming node message using an app called uFldMessageHandler.

In short, uFldNodeComms reasons about node reports and node messages, but it uses node reports to have situational awareness of all vehicle positions. It uses this situational awareness to be the arbiter of which node messages are allowed to travel between which vehicles, depending on the restrictions configured by the user as uFldNodeComms parameters.

2 Handling Node Reports

Node reports may contain a bundle of useful information for sharing between vehicles. Perhaps the most important is the present vehicle position and trajectory. This may be used by the receiving vehicle for collision avoidance and formation keeping and so on. The vehicle position is also used by uFldNodeComms to determine if the node reports themselves are to be shared between vehicles. The inter-vehicle ranges derived from the NODE_REPORT messages are also used to determine if other more generic information contained in the NODE_MESSAGE variable is to be shared between vehicles.

2.1 The Criteria for Routing Node Reports

The basic criteria for sharing node reports is range. By default, the node report received for any one vehicle is passed on to *all* other known vehicles within comms range of the originating vehicle. The comms range is set by the parameter comms_range as on line 9 in Listing 3. Starting with this as a baseline, a few other factors may be involved in determining whether a node report is shared.

Using Vehicle Group Information

If uFldNodeComms is configured with the parameter groups set to true, as on line 21 in Listing 3, then node reports are only shared between vehicles having the same group name. The group name is a field contained in the node report itself, so the onus is on the vehicle to include this information as part of its report. The pNodeReporter application contains an optional configuration parameter group=<group-name> where the group information is declared for inclusion in all node reports. The

motivation for the grouping option is to support multi-vehicle competitions where some vehicles want to convey positions to teammates, but not adversarial vehicles.

The groups feature only affects the passing of node reports between vehicles. It does not affect the passing of node *messages* discussed further below. Node messages contain their addressee information explicitly.

The msg_groups feature only affects the passing of node *messages* between vehicles. It does not affect the passing of node *reports* discussed above. This feature is introduced in the release following 19.8.2.

Establishing and Inter-Vehicle Critical Range

When the two vehicles are within a range deemed critical, as set by the critical_range configuration parameter as on line 10 in Listing 3, node reports are shared between vehicles regardless of the comms_range parameter and the groups parameter. The default for this parameter is 30 meters. The thought behind this feature is that, while it may be advantageous to not broadcast your own vehicle position to non group members for the purposes of a competition, it may be a good idea to share this information for the sake of collision avoidance.

Checking for Staleness

Since up-to-date inter-vehicle range information is used as part of the criteria in determining whether a vehicle receives a new node report from another, the position of the candidate recipient vehicle needs to reasonably up-to-date. The stale_time configuration parameter may be set as on line 11 in Listing 3, to determine the amount of elapsed time without receiving a node report from a vehicle before it is considered stale. If a recipient vehicle becomes stale, it will also not receive node messages.

Ignoring a Group

The user may specify a group name with the ignore_group parameter. All incoming node reports that match this group will simply be ignored. Introduced after Release 19.8.x.

2.2 Optional Limitation on the Node Report Share Rate

The min_report_interval, given in seconds, sets the minimum amount of time between node report messages from one vehicle to another. The default value is -1, disabling this feature.

Note that when the view_node_rpt_pulses parameter is set to true, then uFldNodeComms also publishes a visual artifact in a posting to VIEW_COMMS_PULSE. By limiting the node report share rate, the rate of this variable posting is limited too. This may also be a consideration in extreme missions of very high numbers of vehicles and very high time warp, for easing the burden on the viewer, e.g., pMarineViewer.

2.3 Node Report Transmissions and pShare

Node reports are communicated to recipient vehicles in coordination with the pShare application. For each unique vehicle name discovered by uFldNodeComms via received node reports, it will publish

a new MOOS variable NODE_REPORT_NAME. This variable will be published with the contents of other vehicles' node reports.

As shown in Figure 2, if three vehicles, *alpha*, *bravo*, and *charlie* become known, three corresponding MOOS variables NODE_REPORT_ALPHA, NODE_REPORT_BRAVO, and NODE_REPORT_CHARLIE will be published. The variable NODE_REPORT_ALPHA will be published with reports from *bravo* and *charlie* and so on. To make this happen, uFldNodeComms needs the corresponding share relationships from, for example, NODE_REPORT_ALPHA in the shoreside community to the variable NODE_REPORT in the alpha community on the alpha vehicle. This sharing relationship is established separately by the uFldShoreBroker application running in the shoreside community.

3 Asymmetric Node Report Sharing

Under normal circumstances, if one vehicle is within range to send a node report or message to another, it should also be able to receive node reports and messages from the same vehicle. In this section we discuss ways to make this relationship asymmetric. This is done in one of two ways:

- Increasing a vehicle's *stealth*, i.e., requiring other receiving vehicles to be closer than normal to receive reports and messages, or
- Increasing a vehicle's *earange*, i.e., allowing other receiving vehicles to be farther than normal and still receive node reports and messages.

3.1 Bestowing a Vehicle with an Enhanced Stealth Property

By using the *stealth* parameter, one vehicle may be given a bit of an advantage. The stealth parameter is assigned to a particular vehicle and is a number in the range [0.5, 1]. A message from source to destination is sent if:

$$actual\_range(src, dest) < comms\_range * stealth(src) \tag{1}$$

By default, the stealth factor for each vehicle is 1. A stealthy vehicle with a factor of one half, means the receiving vehicle needs to be twice as close as it would otherwise to receive the stealthy vehicle's node report or node message. The critical_range parameter may be used to override a vehicle's stealthiness with respect to sending node reports. That is, a message from source to destination is sent if:

$$actual\_range(src, dest) < max((comms\_range * stealth(src)), critical\_range) \tag{2}$$

The stealth value for a particular vehicle may be set in the MOOS configuration file as shown on line 18 in Listing 3. It may also be set dynamically by receiving mail on the variable UNC_STEALTH. For example the posting

```
UNC_STEALTH = vname=alpha, stealth=0.75
```

would immediately reset the stealth factor for vehicle *alpha* on the next iteration. The motivation for exposing this parameter via incoming MOOS mail is that another shoreside application, monitoring vehicle speed for example, may reward a vehicle operating in the field with increased stealth based on any criteria. This can be useful in constructing vehicle competitions.

3.2 Bestowing a Vehicle with an Enhanced Listening Property

In addition to the stealth property, a complementary property may be bestowed upon a vehicle using the earange parameter. The earange parameter is assigned to a particular vehicle and is a number in the range [1, 2]. A message from source to destination is sent if:

$$actual\_range(src, dest) < comms\_range * earange(dest) \tag{3}$$

By default, the earange factor for each vehicle is 1. A vehicle with a factor of two may receive node reports of another vehicle at twice the range it may otherwise. In the end the stealth and earange properties may cancel each other out when their extreme values are factored together. Taken together a message from source to destination is sent if:

$$actual\_range(src, dest) < comms\_range * stealth(src) * earange(dest) \tag{4}$$

The earange value for a particular vehicle may be set in the MOOS configuration file as shown on line 19 in Listing 3. It may also be set dynamically by receiving mail on the variable UNC_EARANGE. For example the posting

```
UNC_EARANGE = vname=alpha, earange=1.5
```

would immediately reset the earange factor for vehicle *alpha* on the next iteration. The motivation for exposing this parameter via incoming MOOS mail is that another shoreside application, monitoring vehicle speed for example, may reward a vehicle operating in the field with increased earange based on any criteria. This can be useful in constructing vehicle competitions.

4 Handling Node Messages

Node messages are of a generic structure for sharing a MOOS variable-value pair between a source node and a destination node. A node message from vehicle *alpha* to vehicle *bravo* would be posted locally by *alpha* with something like:

```
NODE_MESSAGE = "src_node=alpha,dest_node=bravo,var_name=FOOBAR,string_val=hello"
```

The local NODE_MESSAGE posting is shared to the shoreside community running uFldNodeComms where it is considered for re-routing to the destination vehicle.

4.1 The Criteria for Routing Node Messages

The basic criteria for sharing node messages is (a) the message addressee, and (b) the range between the source and destination vehicles. Since uFldNodeComms is receiving and keeping track of incoming node reports from all vehicles, it has ready access to the inter-vehicle range between the source and destination nodes. The same criteria used for sending node *reports* is used for sending node messages. It must meet the range criteria as perhaps modified by the stealth and earange factors discussed earlier. The only difference is that the critical_range parameter is irrelevant for the issue of sending node messages. This parameter was only used for node reports in the interest of safety and collision avoidance. There are however two other factors that may affect node message transmission, discussed next, message frequency and message size.

4.2 Enforcing a Minimum Time Between Node Messages

A maximum send frequency is enforced by requiring a minimum wait time between successful sends from a given source node. This minimum time is given by the parameter `min_msg_interval` as on line 13 in Listing 3. The default is 30 seconds. This interval is defined by the time starting with a successful transmission of a node message from a source to any destination. A separate log is kept by `uFldNodeComms` for each known vehicle.

Further clarification: Suppose for example, the `min_msg_interval` is set to 60 seconds. If a vehicle attempts an outgoing message 58 second after the previous message, it will not go through. If attempted 4 seconds later or 62 seconds after the previous successful message, then it will go through. It's as if the unsuccessful attempt never happened.

Continuing this example, consider a vehicle that attempts a message 62 seconds after the previous message was sent. Suppose this new message fails due to the inter-vehicle range being to large, the message length being too long, or the destination node not being recognized. In this case the failure results in another full 60 seconds being needed before the next message will be handled.

4.3 Enforcing a Maximum Node Message Length

The length of node messages may be limited with the parameter `max_msg_length`, as on line 14 in Listing 3. The default maximum length is 1000 characters. The length of a message refers to the number of characters in the `string_val` field. For example, the length of the message

```
NODE_MESSAGE = "src_node=alpha,dest_node=bravo,var_name=FOOBAR,string_val=hello"
```

is five. Limiting the message length is a proxy for inter-vehicle communications where message packet length is constrained, as with acoustic communications for example.

4.4 Posting Messages to a Vehicle Group

A node message is typically addressed to another named vehicle. The sender may also address the message by group name. All vehicles in the group meeting the prevailing range criteria will receive the message. The group associated with the vehicle is declared in the node report sent by that vehicle. A node message using a group address is similar except for the use of the `dest_group` parameter

```
NODE_MESSAGE = "src_node=alpha,dest_group=red_team,var_name=FOOBAR,string_val=hello"
```

The sender has the option of indicating *both* a group name and vehicle name as the message destination. It may also specify more than one vehicle name explicitly. Thus the following is allowed:

```
NODE_MESSAGE = "src_node=alpha,dest_name=bravo:charlie:gilda,dest_group=red_team,
               var_name=FOOBAR,string_val=hello"
```

If a destination vehicle is specified twice in the list of destinations or implicitly in the named group, the message will be sent only once to the destination vehicle.

5 Visual Artifacts for Rendering Inter-Vehicle Communications

Each time a node report or message is sent to a vehicle by uFldNodeComms, another posting is made to the variable VIEW_COMMS_PULSE. This message may be subscribed for by another application using it to visually render the communications events. The screen shot in Figure 3 below shows four vehicles. The two bottom vehicles are sharing node reports indicated by the red and blue comms pulses.

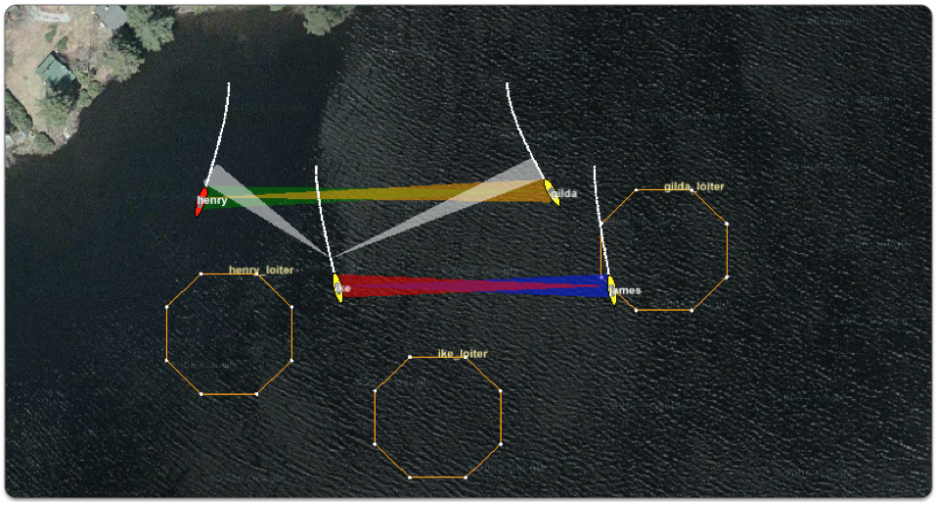

Figure 3: **Communication Pulses:** A visual artifact, a VIEW_COMMS_PULSE, is rendered between vehicles when either a node report or node message is generated. The white pulse indicates a node message, and the non-white pulses indicate a node report. The pulse widens from a point at the source vehicle to maximum width at the destination vehicle. A pulse will remain rendered in pMarineViewer for some number of seconds after initial post, unless it is replace by a new pulse with the same label.

The top two vehicles are also sharing node reports seen by the yellow and green pulses. The bottom left vehicle has also just sent a node message to the top two vehicles indicated by the two white pulse. Note the pair of white pulses were posted a few seconds prior to the present point in time since they point to vehicle positions just back in the vehicle history. The node reports are updated continuously, constantly replacing the pulse just previously posted.

The pulse colors are chosen automatically by uFldNodeComms. They may be toggled off in pMarineViewer with the '@' key. The comms pulse is conveyed in a posting to the variable VIEW_COMMS_PULSE. The format is shown with the following example.

```
VIEW_COMMS_PULSE = "sx=109.63,sy=-60.06,tx=30.96,ty=-60.22,beam_width=7,duration=10,
                    fill=0.35,label=JAMES2IKE,edge_color=green,fill_color=red,
                    time=2652414456.59,edge_size=1"
```

Comms pulses may be generated by other applications besides uFldNodeComms, and may be consumed and rendered by other applications besides pMarineViewer. The definition of the comms pulse object and the methods for serializing and de-serializing between object and string representation may be found in the lib_geometry library in the moos-ivp software tree.

6 Node Report Filtering and Sharing

As discussed above, a core input to uFldNodeComms are all the node reports arriving from each vehicle. Node reports are needed to:

- Send node reports back out to other vehicles, serving as proxy AIS system between vehicles
- Determine where vehicles are relative to each other, to decide whether inter-vehicle messages are sent from one vehicle to another.

The rate of node report arrivals may huge, especially when the shoreside is supporting a multi-vehicle mission of say 50 or more vehicles, and in simulations of say time warp 50 or higher. On each vehicle a node report is typically generated by pNodeReporter at 4Hz. Together this would amount to 10,000 node reports per second. The uFldNodeComms app can routinely handle this load without issue on any machine we have tested.

However, pMarineViewer also ingests this same stream of node reports, primarily for the purpose of updating and rendering vehicle trajectories. In a high time warp simulation resulting in a single vehicle generating a node report at 100-200 node reports per second, this rate is far more than what pMarineViewer needs to smoothly render a vehicle trajectory. Roughly 10-20 position updates per second is sufficient to give the human eye the feeling of a smoothly updated trajectory.

As an optional configuration, uFldNodeComms can be configured to re-post node reports under a different variable name, NODE_REPORT_UNC. The re-posting is generated at a rate that considers the prevailing time warp, and ensures a posting rate commensurate with what is needed for human visual updates, e.g. 10-20Hz. The idea is conveyed below in Figure 4.

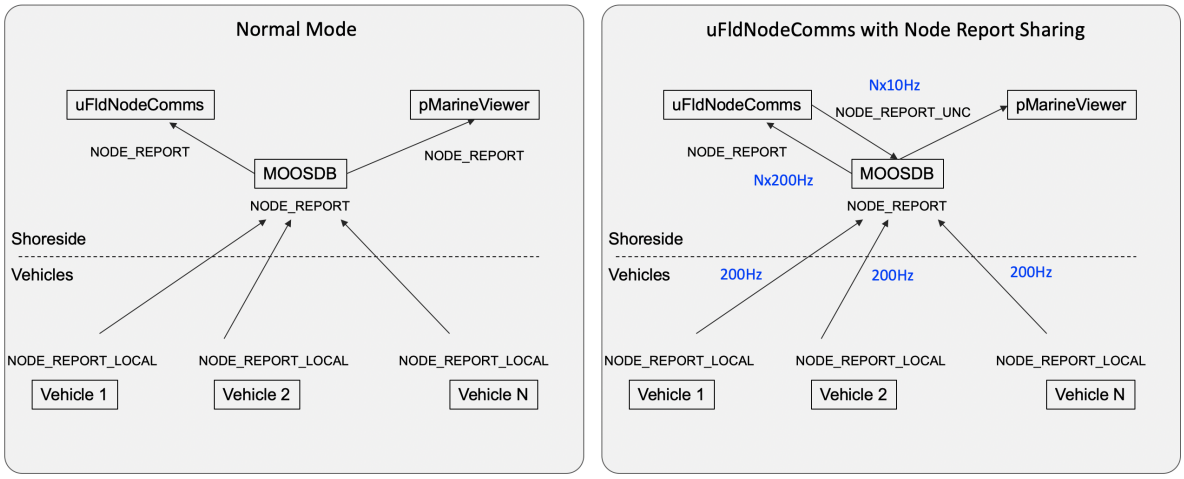

Figure 4: **Node Report Sharing:** With the optionally node report sharing configured, uFldNodeComms can act as a filter of node reports, re-posting node reports at a slower rate, in very high time warp situations.

An experienced MOOS user may note that pMarineViewer could be configured to simply register for NODE_REPORT at a slower rate, when the app registers for this variable. Of course the registration rate would have to also consider the prevailing time warp, but this is easy enough. This approach

would be problematic however because using the MOOSDB to filter out say 9 of every 10 node reports may result in un-even filtering per vehicle. Since all vehicles publish to NODE_REPORT and not NODE_REPORT_ABE for example, the above approach may end up filtering out 99 of 100 reports from one vehicle and 4 of 5 reports from another, leading to jumpy rendering and false stale report warnings.

By shifting node report filtering to uFldNodeComms, under such extreme high contact, high time warp situations, uFldNodeComms shifts computational burden away from pMarineViewer. The viewer is probably the most computationally burdened app running in these kinds of missions and uFldNodeComms is already processing the full set of node reports anyway as part of its basic job. Finally, most users are typically using a computer with multiple cores available, with uFldNodeComms and pMarineViewer running on different cores.

7 Terminal and AppCast Output

The uFldNodeComms application produces some useful information to the terminal and identical content through appcasting. An example is shown in Listing 1 below. On line 2, the name of the local community (usually shoreside) is listed on the left. On the right, "0/0(1822) indicates there are no configuration or run warnings, and the current iteration of uFldNodeComms is 1822. Lines 4-13 show the configuration requested from the mission configuation file.

Lines 15-30 convey a summary of received and sent node reports from each vehicle. The number in parentheses at the end of lines 18-22 indicate the elapsed time since the last node report was received. The number in brackets indicates the rate of received node reports (total node reports received per vehicle over time since app startup). This number should roughly equal the AppTick for pNodeReporter running on each vehicle.

Listing 7.1: Example uFldNodeComms console and appcast output.

```
 1  ============================================================
 2  uFldNodeComms shoreside                            0/0(1822)
 3  ============================================================
 4  Configuration:
 5          Comms Range: 1000
 6       Critical Range: 10
 7           Stale Drop: 5
 8          Stale Foget: 100
 9      Max Message Len: 1000
10      Min Message Int: 20
11      Min Report  Int: 60
12          Apply Group: false
13        Share Reports: true (0.1)
14
15  Node Report Summary
16  ====================================
17       Total Received: 9112      Elapsed      Rate
18                  ABE: 1828      (0.0)        [3.60]
19                  BEN: 1824      (0.0)        [3.59]
20                  CAL: 1820      (0.0)        [3.58]
21                  DEB: 1818      (0.0)        [3.58]
22                  EVE: 1822      (0.0)        [3.58]
```

```
23       ---------------
24           Total Sent: 90
25                  ABE: 27
26                  BEN: 18
27                  DEB: 18
28                  EVE: 27
29       ---------------
30       Stale Vehicles: 0
31
32  Node Message Summary
33  ====================================
34     Total Msgs Received: 3
35                      ABE: 3     (24.9)
36       ------------------
37              Total Sent: 3
38                      BEN: 3
39       ------------------
40      Total Blocked Msgs: 0
41                 Invalid: 0
42            Missing Info: 0
43           Stale Receiver: 0
44               Too Recent: 0
45             Msg Too Long: 0
46            Range Too Far: 0
47
48    ====================================================================
49    Most Recent Events (3):
50    ====================================================================
51    [96.22]: Msg rec'd: src_node=abe,dest_node=ben,var_name=UPDATE_LOITER,string_val=speed=2.4
52    [71.10]: Msg rec'd: src_node=abe,dest_node=ben,var_name=UPDATE_LOITER,string_val=speed=2.4
53    [56.46]: Msg rec'd: src_node=abe,dest_node=ben,var_name=UPDATE_LOITER,string_val=speed=2.4
```

The summary of node messages is shown in the second half of the report, in lines 32-46 in this case. The total messages received is shown on line 34, with a breakdown of where they have been received from in the following lines. Starting on line 37, a summary of sent node messages is given. First the total sent messages on line 37 and a breakdown of receivers in the following lines. A summary of blocked messages is given next, in this case in lines 40-46. The total number of blocked messages is given first, followed by the possible reasons for blocking in lines 41-46. Finally, the most recent messages are shown as events in the last lines of the report.

8 Configuration Parameters of uFldNodeComms

The following parameters are defined for uFldNodeComms.

Listing 8.2: Configuration Parameters for uFldNodeComms.

comms_range: Max range outside which inter-vehicle node reports and node messages will not be sent. Legal values: any numerical value. The default is 100 meters. Section 2.1.

critical_range: Range in meters within which inter-vehicle node reports will be shared even if group membership would otherwise disallow. Legal values: any numerical value. The default is 30 meters. Section 2.1.

debug: If true, further debugging information is produced to the terminal output. Legal values: true, false. The default is false.

earange: A parameter in the range of [1,10] for extending the range a vehicle may otherwise hear node reports from other vehicles. The default is 1. Section 3.2.

groups: If true, inter-vehicle node reports are shared only if two vehicles are in the same group. May be overridden if the two vehicles are within the critical range. The vehicle group designation is typically declared in pNodeReporter. The default is false. Section 2.1.

ignore_group: If specified, incoming node reports that match this group will be ignored. Introduced after Release 19.8.x.

min_msg_interval: The number of seconds required between messaged sends for any one source vehicle. The default is 30 seconds. Section 4.2.

max_msg_length: The total number of characters that may be sent in a string component of a node message. The default is 1000. Section 4.3.

min_rpt_interval: The number of seconds required between report sends for any one source vehicle. The default is -1, meaning no dropping of reports. Section 2.2.

msg_groups: If true, inter-vehicle node messages are shared only if the two vehicles are in the same group. The vehicle group designation is typically declared in pNodeReporter. The default is false. Section 2.1.

msg_color: The color of inter-vehicle message comms pulses. The default is "white". Section 5.

msg_repeat_color: The color of inter-vehicle message comms pulses when the message represents a repeat of a previously dropped message, under mediated message passing. The default is "light_green". Section 5.

stealth: A parameter in the range [0,1] for reducing the range other vehicles may otherwise hear the node reports from a source vehicle. The default is 1. Section 3.1.

pulse_duration: A duration, in seconds, that the comms or node pulse should be viewable. It is a field in VIEW_COMMS_PULSE message that is respected by at least pMarineViewer to render and slowly fade-out the rendering after this duration. The default is 10 seconds. Section 5.

shared_node_reports: When enabled with true, node reports are re-posted to the shoreside community as NODE_REPORT_UNC. The default is false. This feature may also be more commonly enabled via in incoming message. See UNC_SHARED_NODE_REPORTS. Section 6.

stale_time: Time in seconds after which a vehicle will not receive node reports or messages unless a node report has been received by that vehicle. The default is 5 seconds. Section 2.1.

verbose:	If true, status reports are displayed to the terminal during operation. The default is false.
view_node_rpt_pulses:	If true, comms pulses are rendered between vehicles whenever a node report successfully makes its way from one vehicle to another. The default is true. Section 5.

An Example MOOS Configuration Block

Listing 3 shows an example MOOS configuration block produced from the following command line invocation:

```
$ uFldNodeComms --example or -e
```

Listing 8.3: Example configuration of the uFldNodeComms application.

```
1  ============================================================
2  uFldNodeComms Example MOOS Configuration
3  ============================================================
4
5  ProcessConfig = uFldNodeComms
6  {
7    AppTick   = 4
8    CommsTick = 4
9
10   comms_range    = 100       // default (in meters)
11   critical_range = 30        // default (in meters)
12   stale_time     = 5         // default (in seconds
13
14   max_msg_length   = 1000    // default (# of characters)
15   min_msg_interval = 30      // default (in seconds)
16   min_rpt_interval = -1      // default (in seconds)
17
18   verbose    = true          // default
19   groups     = false         // default
20   msg_groups = false         // default
21
22   stealth  = vname=alpha, stealth=0.8
23   earange  = vname=alpha, earange=4.5
24
25   shared_node_reports = true/false or any non-neg number
26
27   pulse_duration = 10;         // default (in seconds)
27   view_node_rpt_pulses = true  // default
28 }
```

9 Publications and Subscriptions for uFldNodeComms

The interface for uFldNodeComms, in terms of publications and subscriptions, is described below. This same information may also be obtained from the terminal with:

```
$ uFldNodeComms --interface or -i
```

9.1 Variables Published by uFldNodeComms

The primary output of uFldNodeComms to the MOOSDB are the node reports and node messages out to the recipient vehicles, and visual artifacts to be used for rendering the inter-vehicle communications.

- APPCAST: Contains an appcast report identical to the terminal output. Appcasts are posted only after an appcast request is received from an appcast viewing utility. Section 7.
- NODE_MESSAGE_<VNAME>: Node messages destined to be sent to a destination vehicle <VNAME> indicated as the recipient in the node message.
- NODE_REPORT_<VNAME>: Node reports destined to be sent to a given vehicle <VNAME> about the vehicle named in the node report.
- NODE_REPORT_UNC: When node report sharing is enabled, a subset of all node reports will be re-posted to this variable. See Section 6.

- VIEW_COMMS_PULSE: A visual artifact for rendering the sending of a node report or node message between vehicles.

9.2 Variables Subscribed for by uFldNodeComms

The uFldNodeComms application subscribes to the following MOOS variables:

- APPCAST_REQ: A request to generate and post a new apppcast report, with reporting criteria, and expiration.
- NODE_MESSAGE: A node message from one vehicle to another.
- NODE_REPORT: A node report for a given vehicle from pNodeReporter.
- NODE_REPORT_LOCAL: Another name for a node report for a given vehicle from pNodeReporter.
- UNC_STEALTH: The extra stealth allowed a given vehicle to "hide" node reports to others.
- UNC_EARANGE: The extra range allowed a given vehicle to "hear" node reports of others.
- UNC_FULL_REPORT_REQ: For a given vehicle name, the next node report from each of the other vehicles will be received, regardless of all other criteria such as range, frequency, group. This only applies for the very next report only, then all criteria reverts. This is a convenient debugging tool.
- UNC_SHARED_NODE_REPORTS: If enabled, set to true, uFldNodeComms will re-post node reports from all vehicles to the variable NODE_REPORT_UNC but will clamp the publication rate, regardless of time warp, to a rate suitable for visual consumption, typically no more than 10 updates per second per vehicle. This feature is used in coordination with pMarineViewer to improve performance in high time warp missions with a high number of contacts. This variable is typically published by pMarineViewer requesting this service from uFldNodeComms. Section 6.

- UNC_VIEW_NODE_RPT_PULSES: A way for external apps to tell uFldNodeComms to turn off or on the rendering of comms pulses whenever a node report is sent from one vehicle to another. Section 5.

Command Line Usage of uFldNodeComms

The uFldNodeComms application is typically launched with pAntler, along with a group of other shoreside modules. However, it may be launched separately from the command line. The command line options may be shown by typing:

```
$ uFldNodeComms --help or -h
```

Listing 9.4: Command line usage for the uFldNodeComms tool.

```
1  ============================================================
2  Usage: uFldNodeComms file.moos [OPTIONS]
3  ============================================================
4
5  Options:
6    --alias=<ProcessName>
7        Launch uFldNodeComms with the given process
8        name rather than uFldNodeComms.
9    --example, -e
10       Display example MOOS configuration block.
11   --help, -h
12       Display this help message.
13   --interface, -i
14       Display MOOS publications and subscriptions.
15   --version,-v
16       Display the release version of uFldNodeComms.
17
18 Note: If argv[2] does not otherwise match a known option,
19       then it will be interpreted as a run alias. This is
20       to support pAntler launching conventions.
```