uFldNodeBroker: Brokering Node Connections

June 2018

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

1 Overview

The uFldNodeBroker application is a tool for brokering connections between a node (a simulated or real vehicle) and a shoreside community. It is used primarily in coordination with uFldShoreBroker to discover and share host IP and port information to automate the dynamic configurations of pShare. Inter-vehicle communications over the network are handled by pShare in both simulation with single or multiple machines as well as on fielded vehicles using Wi-Fi or cellphone connections. The pShare application simply needs to know the IP address and port number of connected machines. Often these aren't known at run-time and even if they were, maintaining that information in configuration files may be cumbersome, especially for large sets of vehicles. This tool is meant to automate the configuration by letting the nodes and shoreside community discover each other by giving the node (uFldNodeBroker) some initial hints on where to find the shoreside community on the network. The typical layout is shown in Figure 1.

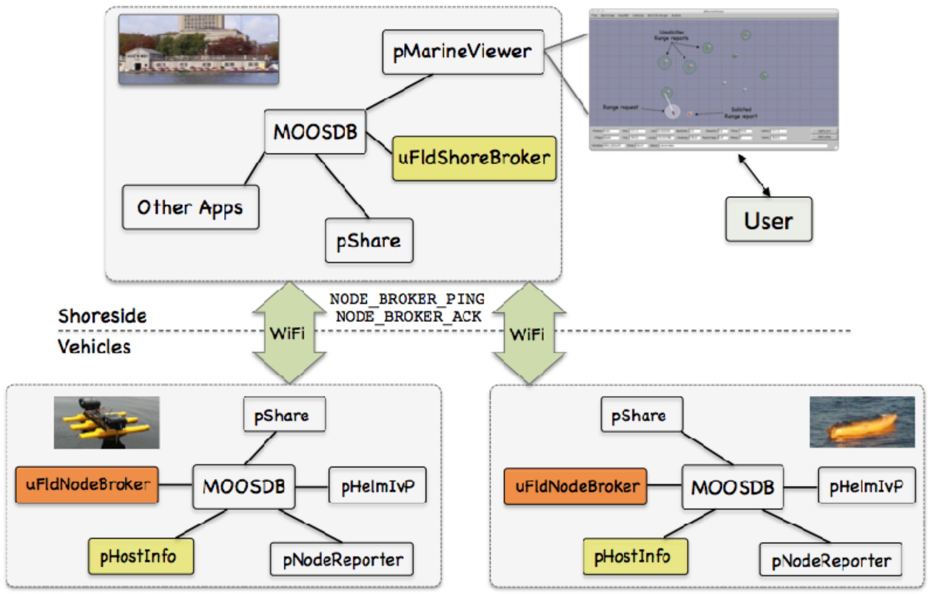

Figure 1: **Typical uFldNodeBroker Topology:** A vehicle (node) sends information about itself (IP address and port number) to possible shoreside locations. Once a connection is made, further sharing is established between the node and shoreside communities.

The functionality of `uFldNodeBroker` paraphrased:

- Discover the node's host information (typically from `pHostInfo`).
- For candidate shoreside hosts, request a new share configuration in `pShare` to each candidate for the variable `NODE_BROKER_PING`.
- Publish `NODE_BROKER_PING` with the node's host information.
- Await a reply in the form of incoming `NODE_BROKER_ACK` mail, presumably from the shoreside community running `uFldShoreBroker`.
- Now that a shoreside community is known, request new share configurations from the node's local `pShare` for all the info otherwise wanted for sharing to the shoreside.
- Keep sending pings periodically in case the shoreside community is re-started and needs to re-establish connections to nodes.

2 The uFldNodeBroker Interface and Configuration Options

The `uFldNodeBroker` application may be configured with a configuration block within a `.moos` file. Its interface is defined by its publications and subscriptions for MOOS variables consumed and generated by other MOOS applications. An overview of the set of configuration options and interface is provided in this section. If one has access to a command line where `uFldNodeBroker` has been built, interface information may also be seen by typing `"uFldNodeBroker --interface"`, and configuration information by typing `"uFldNodeBroker --example"`.

2.1 Configuration Parameters of uFldNodeBroker

The following parameters are defined for `uFldNodeBroker`.

Listing 2.1: Configuration Parameters for uFldNodeBroker*.*

auto_bridge_appcast:	Suppress the normal automatic bridging of the APPCAST variable. The default is false.
auto_bridge_realmcast:	Suppress the normal automatic bridging of the REALMCAST and REALMCAST_CHANNELS variables. The default is false.
bridge:	A variable to register with pShare for bridging to a shoreside MOOS community once a shoreside connection has been established.
try_shore_host:	A candidate route to send send initial pings in hopes of establishing a connection. The route specifies an input route for pShare running in a shoreside community.

Since appcasting and realmcasting are common and recommended practices for users of the uField Toolbox, key variables that are bridged are done automatically. Normally the APPCAST variable is auto bridged to support appcasting. And the REALMCAST and REALMCAST_CHANNELS variables are auto bridged to support realmcasting. The user has the option to opt out of this configuration by setting the auto_bridge_* configuration parameters described above. Note that realmcasting was first introduced to the uField Toolbox and MOOS-IvP code in the first release after 19.8.1.

An Example MOOS Configuration Block

An example MOOS configuration block can be obtained by entering the following from the command-line:

```
$ uFldNodeBroker --example or -e
```

Listing 2.2: Example configuration of the uFldNodeBroker *application.*

```
===============================================================
uFldNodeBroker Example MOOS Configuration
===============================================================

ProcessConfig = uFldNodeBroker
{
  AppTick   = 4
  CommsTick = 4

  keyword        = lemon

  try_shore_host = pshare_route=localhost:9200
  try_shore_host = pshare_route=192.168.0.122:9301
  try_shore_host = pshare_route=multicast_8

  bridge = src=VIEW_POLYGON
  bridge = src=VIEW_POINT
  bridge = src=VIEW_SEGLIST

  bridge = src=NODE_REPORT_LOCAL, alias=NODE_REPORT
}
```

3 Publications and Subscriptions for uFldNodeBroker

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

```
$ uFldNodeBroker --interface or -i
```

3.1 Variables Published by uFldNodeBroker

The primary output of uFldNodeBroker to the MOOSDB are the requests to pShare for registrations, and the outgoing pings to candidate shoreside communities.

- 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 4.
- NODE_BROKER_PING: A message written locally but bridged to a candidate shoreside MOOS community, containing IP address and pShare route information about the local community.
- PSHARE_CMD: message to pShare to add a new bridge for a given variable and given target MOOS community at a specified IP address and port number.

3.2 Variables Subscribed for by uFldNodeBroker

The uFldNodeBroker 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_BROKER_ACK: Information published presumably by uFldShoreBroker running in a separate shoreside community. Message has information about the shoreside host including the community name, IP address and port numbers for the MOOSDB and its local pShare process.
- PHI_HOST_INFO: Information about the local host IP address, the MOOS community name, the port on which the DB is running, and the port on which the local pShare is listening for UDP messages.

3.3 Command Line Usage of uFldNodeBroker

The uFldNodeBroker 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

```
$ uFldNodeBroker --help or -h
```

Listing 3.3: Command line usage for the uFldNodeBroker tool.

```
==========================================================
Usage: uFldNodeBroker file.moos [OPTIONS]
==========================================================

Options:
  --alias=<ProcessName>
      Launch uFldNodeBroker with the given
      process name rather than uFldNodeBroker.
  --example, -e
      Display example MOOS configuration block.
  --help, -h
      Display this help message.
  --interface, -i
      Display MOOS publications and subscriptions.
  --version,-v
      Display release version of uFldNodeBroker.
```

4 Terminal and AppCast Output

The uFldNodeBroker application produces some useful information to the terminal on every iteration of the application. An example is shown in Listing 4 below. This application is also appcast enabled, meaning its reports are published to the MOOSDB and viewable from any uMAC application or pMarineViewer. The counter on the end of line 2 is incremented on each iteration of uFldNodeBroker, and serves a bit as a heartbeat indicator. The "0/0" also on line 2 indicates there are no configuration or run warnings detected.

Listing 4.4: Example terminal or appcast output for uFldNodeBroker*.*

```
 1  ===================================================================
 2  uFldNodeBroker henry                                       0/0(129)
 3  ===================================================================
 4
 5    Total OK  PHI_HOST_INFO    received: 13
 6    Total BAD PHI_HOST_INFO    received: 0
 7    Total HOST_INFO changes    received: 1
 8    Total          PSHARE_CMD    posted: 7
 9    Total BAD NODE_BROKER_ACK  received: 6
10
11  ===========================================================
12               Vehicle Node Information:
13  ===========================================================
14
15       Community: henry
16          HostIP: 10.0.0.5
17     Port MOOSDB: 9001
18       Time Warp: 4
19        IRoutes: 10.0.0.5:9301
20
21  ===========================================================
22              Shoreside Node(s) Information:
23  ===========================================================6
24
25  Community                  Pings  Pings  IP        Time
26  Name       Route           Sent   Acked  Address   Warp
27  ---------  --------------  -----  -----  --------  ----
28  shoreside  localhost:9300  128    120    10.0.0.5  4
29  Phase Completion Summary:
30  ------------------------------------
```

```
31   Phase 1: (Y) Valid Host information retrieved (iroutes).
32   Phase 2: (Y) Valid TryHosts (1) configured.
33   Phase 3: (Y) NODE_BROKER_PINGS are being sent to TryHosts.
34   Phase 4: (Y) A Valid NODE_BROKER_ACK has been received.
35   Phase 5: (Y) pShare requested to share user vars with shoreside.
36  All Phases complete. Things should be working as configured.
```

On line 5, the number of incoming mail messages for PHI_HOST_INFO is tallied where the host information bundle is deemed complete. It is complete if it contains the host community, IP address, time warp and pShare input route information. If an incomplete host information packet is received, it is tallied in line 6. The number on line 6 should always be zero. If the number on line 6 is not zero, or the number on line 5 never increments, you should check the operation of the pHostInfo.

The number on line 7 indicates the number of time the detected host information changes. This number should stabilize very quickly maxing out at 1 or 2 typically. The number on line 8 indicates the number of dynamic bridge requests posted to pShare. These are in the form of postings to the variable PSHARE_CMD, which occur first for outgoing pings to candidate shoreside hosts, and then for the user bridge configuration variables after a shoreside host has been connected. Invalid NODE_BROKER_ACK messages are tallied on line 9. An invalid ack may be due to a mismatch in time warp between node and shoreside, or a mismatch in keywords if keywords are being used.

Self node information is displayed in the next group, lines 11-19. The community name, MOOSDB port, and time warp are all read from the .moos mission configuration file. The node's IP address (line 16) and the local pShare input routes (line 19) are obtained from output received from pHostInfo.

Information about candidate and connected shoreside communities is shown next in lines 21-28. In this case there is only one entry, line 27. For each candidate entry, the community name, shoreside pShare input route, number of pings sent and acknowledged are in the first four columns. The last two columns indicate the IP address and time warp of the shoreside learned from NODE_BROKER_ACK messages received from the shoreside. When a candidate shoreside community has *not* been connected, the entry in this table will something like:

```
Community                    Pings  Pings  IP       Time
Name       Route             Sent   Acked  Address  Warp
---------  ---------------   -----  -----  -------  ----
           localhost:92003   385    0
```

The last block of information in the report is the Phase Completion Summary, lines 29-36. It lists the rough sequence of events typical to reach a shoreside community connection. If any one of these phases is incomplete, the output on line 36 will be replaced with a few hints on where to troubleshoot.