uFldShoreBroker: Brokering Shore Connections

June 2018

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

1 Overview

The `uFldShoreBroker` application is a tool for brokering connections between a shoreside community and one or more nodes (simulated or real vehicles). A shoreside community is collection of MOOS processes typically running a GUI providing a situational display and managing messages to and from fielded vehicles. This is depicted in the notional rendering in Figure 1 below. The shoreside community in practice is often situated on a ship with UUVs below, and is more aptly referred to as the topside community. The user interacts with the GUI or perhaps other communication modules, to send high-level messages to the vehicles.

The `uFldShoreBroker` application is used primarily in coordination with `uFldNodeBroker`, running on the vehicles, 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 unduly 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 letting (`uFldShoreBroker`) respond to incoming pings, i.e., initialization messages, from nodes on the network. The typical layout is shown in Figure 1.

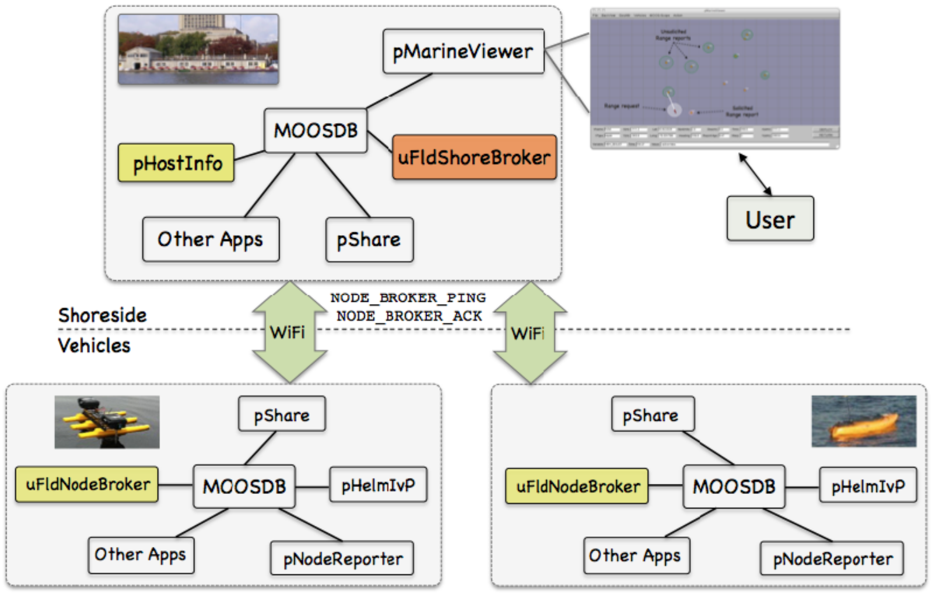

Figure 1: **Typical uFldShoreBroker Topology:** A vehicle (node) sends information about itself (IP address and port number) to the shoreside, received by uFldShoreBroker. It responds by (a) acknowledging the connection to the node, and (b) establishing user configured bridges of particular MOOS variables to the node.

The functionality of `uFldShoreBroker` paraphrased:

- Discover the shoreside's own host information (typically from `pHostInfo`).
- Await incoming `NODE_BROKER_PING` messages from non-local vehicles.
- Upon an incoming ping, respond to the nodes with a `NODE_BROKER_ACK` message to the location specified in the ping message.
- Establish new bridges to the nodes for variables specified previously by the user in the `uFldShoreBroker` configuration.
- Keep sending acknowledgments periodically to confirm to vehicles that they are still connected to the shoreside community.

2 Bridging Variables Upon Connection to Nodes

A primary function of `uFldShoreBroker` is to establish bridging relationships to a remote node community after it has received a ping from that community. These variables are specified in the configuration block with lines like `bridge = "src=DEPLOY_ALL, alias=DEPLOY"` as in Listing 3. This step is described next.

2.1 Inter-MOOSDB Bridging with pShare

Static bridging with `pShare` is done by specifying the desired route in the `pShare` configuration block with a line of the form:

```
output = src_name=VAR, dest_name=ALIAS, route=ROUTE
```

For example:

```
output = src_name=DEPLOY_HENRY, dest_name=DEPLOY, route=12.56.111.1:9200
```

The above connection may be used to send the vehicle Henry the deploy command from the shoreside community. The problem is that the shoreside may not know the IP address of Henry (or the port on which it's pShare is listening) until it presents itself to the shoreside at run time.

In this case, dynamic share registration needs to be used by sending `pShare` a message after it has been launched. For example, the above sharing relationship could be established by sending the following message:

```
PSHARE_CMD = "cmd=output, src_name=DEPLOY_ALL, dest_name=DEPLOY,
             route=2.56.111.1:9200
```

It is the job of `uFldShoreBroker` to post the above style dynamic requests once the node information becomes known to the shoreside community.

2.2 Handling a Valid Incoming Ping from a Remote Node

The basic job of `uFldShoreBroker` is to await incoming pings, and use the information in a ping message to (a) decide if the ping should be accepted, and (b) send the appropriate response back to the sender, and (c) set up new outgoing `pShare` relationships if the ping is indeed accepted. The contents of a ping may look something like:

```
NODE_BROKER_PING = "community=henry,host=192.168.1.22,port=9000,time_warp=10
                    pshare_iroutes=192.168.1.22:9200,time=1325178800.81"
```

There must be a match in the MOOS *time warp* used by the shoreside MOOS community and any node connected to the shore. This is always 1 when operating vehicles in the field, but may be set to a much larger number in simulation. The time warp is set with the parameter `MOOSTimeWarp` at the top of the .moos configuration file.

The ping consists of three key pieces of information:

- The community name of the node,
- The IP address of the node,
- The input routes on which the node is listening for messages with its own local `pShare` running.

Once this information is known by the shoreside broker, new bridges can be established for variables identified by the user. The only other information needed is (a) the name of the variable in the local `MOOSDB`, and (b) the name (alias) of the variable as it is to be known in the remote `MOOSDB`. Once a valid ping has been received and accepted, `uFldShoreBroker` is ready to establish bridge arrangements with its local pShare running.

2.3 Vanilla Bridge Arrangements

The simplest bridge arrangement specifies (a) the variable as it is known locally, and (b) the variable name as it is to be known remotely. This is done with a uFldShoreBroker configuration line similar to:

```
bridge = src=DEPLOY_ALL, alias=DEPLOY
```

For *each* unique incoming ping, a new bridge arrangement will be requested. By unique, we mean having a distinct community (remote vehicle) name. For example, if pings are received and accepted from *henry*, *james*, and *ike*, uFldShoreBroker would make three separate posts, perhaps looking like:

```
PSHARE_CMD = "src_name=DEPLOY_ALL, dest_name=DEPLOY, route=2.56.111.1:9200"
PSHARE_CMD = "src_name=DEPLOY_ALL, dest_name=DEPLOY, route=2.56.111.3:9200"
PSHARE_CMD = "src_name=DEPLOY_ALL, dest_name=DEPLOY, route=2.56.111.6:9200"
```

At this point the behavior of pShare on the shoreside would be functionally equivalent to the scenario where the following three lines were in the pShare configuration block:

```
output = src_name=DEPLOY_ALL, dest_name=DEPLOY, route=12.56.111.1:9200
output = src_name=DEPLOY_ALL, dest_name=DEPLOY, route=12.56.111.3:9200
output = src_name=DEPLOY_ALL, dest_name=DEPLOY, route=12.56.111.6:9200
```

2.4 Bridge Arrangements with Macros

The user may configure uFldShoreBroker with bridge arrangements containing a couple types of macros. For example, consider the configuration:

```
bridge = src=DEPLOY_$V, alias=DEPLOY
```

The `$V` macro will expand to the name of the vehicle when it comes time to request a new bridge. If the newly received ping is from the node named *gilda*, the bridge request from the above pattern may look like:

```
PSHARE_CMD = cmd=output, src_name=DEPLOY_GILDA, dest_name=DEPLOY,
             route=2.56.111.1:9200
```

Note the vehicle name in the MOOS variable macro was expanded to be upper case, even though the ping information referred to the vehicle as *gilda*. This is just to aid in the convention that MOOS variable are typically all upper case. If one really want a literal expansion with no case altering, the macro `$v`, lower-case `v`, may be used instead. The macro is only respected as part of `src` field. In other words, if the bridge were configured with `alias=DEPLOY_$V`, the macro would not be expanded.

The other type of macro implemented is the `$N` macro, as in:

```
bridge = src=LOITER_$N, alias=LOITER
```

The `$N` macro will expand to the integer value representing number of unique pings received thus far. For example, if three pings are received and accepted from *henry*, *james*, and *ike*, uFldShoreBroker would make three separate posts, perhaps looking like:

```
 PSHARE_CMD = "src_name=LOITER_1, dest_name=LOITER, route=2.56.111.1:9200"
 PSHARE_CMD = "src_name=LOITER_2, dest_name=LOITER, route=2.56.111.4:9200"
 PSHARE_CMD = "src_name=LOITER_3, dest_name=LOITER, route=2.56.111.12:9200"
```

This may be useful when used in conjunction with another MOOS process generating output generically for N vehicles, without having to know the vehicle names in advance.

2.5 A Common Configuration Shortcut - the qbridge Parameter

A common usage pattern is to configure uFldShoreBroker to request two types of bridges for a given variable, for example:

```
bridge = src=DEPLOY_ALL, alias=DEPLOY
bridge = src=DEPLOY_$V,  alias=DEPLOY
bridge = src=RETURN_ALL, alias=RETURN
bridge = src=RETURN_$V,  alias=RETURN
```

This could be use in the shoreside community for easily commanding vehicles. When the user wishes to deploy all vehicles, a posting of `DEPLOY_ALL="true"` does the trick. If the user wishes only the vehicle *james* to return, a posting of `RETURN_JAMES="true"` may be be made. This pattern is so common that this shortcut is supported. This is done with the `qbridge`, ”quick bridge”, parameter. The above four configuration lines could be accomplished instead by:

```
qbridge = DEPLOY, RETURN
```

3 Usage Scenarios for the uFldShoreBroker Utility

The uFldShoreBroker was designed with a canonical command-and control scenario in mind. The idea is that the N deployed vehicles have a common autonomy protocol implemented. For example, a message to deploy or return a vehicle is the same message for each deployed vehicle. The idea is that two types of communication channels need to be established with pShare, (a) messages sent to all vehicles, and (b) messages sent to a particular named vehicle. The convention proposed here is to do this with the two types of bridging described in the discussion of the `qbridge` parameter, in Section 2.5. For a variable such as `DEPLOY`, a posting in the shoreside community to `DEPLOY_ALL` would go to all known vehicles, and a posting to `DEPLOY_HENRY` would only go to that particular vehicle.

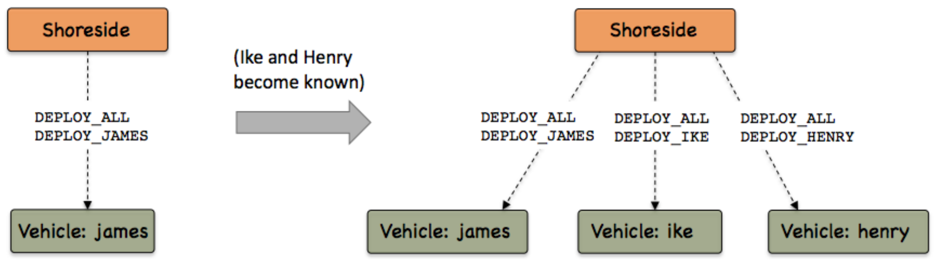

Figure 2: **Common uFldShoreBroker Usage Scenario:** As vehicles become known to the shoreside, each vehicle has two new bridges established. The first is the same for all vehicles to allow broadcasting, and the second bridge is unique to the particular vehicle, for individual command and control.

4 Terminal and AppCast Output

The uFldShoreBroker application produces some useful information to the terminal on every iteration of the application. An example is shown in Listing 1 below. This application is also appcast enabled, meaning its reports are published to the MOOSDB and viewable from any uMAC application or pMarineViewer. See the appcasting documentation for more on appcasting and viewing appcasts.

On line 1, the application iteration is shown, more as a heartbeat indicator. In lines 4-7, the primary variables consumed and posted by uFldShoreBroker are summarized in terms of how many posts have been made and received for each variable.

Listing 4.1: Example terminal output of the uFldShoreBroker tool.

```
 1  ===================================================================
 2  uFldShoreBroker_PS shoreside                                  (109)
 3  ===================================================================
 4
 5   Total PHI_HOST_INFO    received: 11
 6   Total NODE_BROKER_PING received: 180
 7   Total NODE_BROKER_ACK    posted: 180
 8   Total PSHARE_CMD         posted: 34
 9
10
11  ===========================================================
12              Shoreside Node(s) Information:
13  ===========================================================
14
15       Community: shoreside
16          HostIP: 128.30.27.202
17     Port MOOSDB: 9000
18       Time Warp: 6
19         IRoutes: localhost:9200
20
21  ===========================================================
22               Vehicle Node Information:
23  ===========================================================
24
25  Node   IP                       Elap  pShare
26  Name   Address         Status   Time  Input Route(s)      Skew
27  -----  -------------   ------   ----  ------------------  ------
28  henry  128.30.27.202   ok       0.0   128.30.27.202:9301  1.6542
```

```
29  gilda  128.30.27.202  ok      0.0   128.30.27.202:9302  1.7572
30
31  Recent Events (2):
32  [22.06]: New node discovered: gilda
33  [16.04]: New node discovered: henry
```

In lines 11-19, the key shoreside properties are listed. Typically, but not always, the shoreside community is name `"shoreside"` as indicated on line 15. The shoreside IP address, determined by `pHostInfo`, is shown on line 16. The time warp, and `MOOSDB` port are read from the shoreside .moos file and listed on lines 17 and 18. The input routes used by `pShare` are listed on line 19. If lines 16 or 19 are blank, `uFldShoreBroker` will not make any connections and the first place to look is whether or not `pHostInfo` is running and producing valid information.

In lines 21-33, the status of each of the known vehicles is shown. The first two vehicles had their pings accepted. Their IP addresses are shown in the second column. Their status is shown in the third column. The elapsed time in the fourth column is the time since the last ping was received by the shoreside. The fifth column shows the input routes being used by `pShare` running on the vehicle node. If multiple routes are in use, this will be shown over multiple lines. The sixth column shows the time skew between the timestamp in the `NODE_BROKER_PING` message compared to the time it was received. Some of this is due to (a) latency in transmission, (b) latency due to App Ticks in brokers on both sides, and (c) clock discrepancy between the shoreside and the node computers. It's also worth mentioning that the skew will be magnified for higher time warps. Currently incoming ping connection requests are not denied due to a high clock skew, but this may be implemented in the future.

5 Configuration Parameters of uFldShoreBroker

The following parameters are defined for `uFldShoreBroker`.

Listing 5.2: Configuration Parameters for `uFldShoreBroker`*.*

`auto_bridge_appcast`:	Suppress the normal automatic bridging of the `APPCAST_REQ` variable. The default is false.
`auto_bridge_mhash`:	Suppress the normal automatic bridging of the `MISSION_HASH` variable. The default is false.
`auto_bridge_realmcast`:	Suppress the normal automatic bridging of the `REALMCAST_REQ` variable. The default is false.
`bridge`:	Names a MOOS variable to be bridged to a node community. Section 2.3.
`keyword`:	Optionally set a keyword. If set, incoming `NODE_BROKER_PING` messages must contain this keyword or else they ping will not get a response. Section TBD.
`qbridge`:	Shorthand notation for a common bridging pattern. Section 2.5.
`try_vnode`:	TBD
`warning_on_stale`:	If true, generate a warning if a previously known node has not been heard from in more than 10 seconds.

As an example, `bridge = src =DEPLOY_ALL, alias=DEPLOY`, will result in the bridging of variable

DEPLOY_ALL from the local MOOSDB, to the variable DEPLOY in a remote MOOS community. Further examples are given in Section 2.

An Example MOOS Configuration Block

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

```
$ uFldShoreBroker --example or -e
```

Listing 5.3: Example configuration of the uFldShoreBroker application.

```
===============================================================
uFldShoreBroker Example MOOS Configuration
===============================================================

ProcessConfig = uFldShoreBroker
{
  AppTick   = 4
  CommsTick = 4

  warning_on_stale      = false (default)
  auto_bridge_realmcast = true  (default)
  auto_bridge_appcast   = true  (default)
  auto_bridge_mhash     = true  (default)

  bridge =  src=DEPLOY_ALL, alias=DEPLOY
  bridge =  src=DEPLOY_$V,  alias=DEPLOY

  try_vnode = 192.168.4.24:9200
  try_vnode = 192.168.4.25:9200

  qbridge = RETURN
  qbridge = NODE_REPORT, STATION_KEEP

  bridge  = src=UP_LOITER_$N, alias=UP_LOITER

  // Note: [qbridge = FOO]  is shorthand for
  //       [bridge = src=FOO_$V,  alias=FOO] and
  //       [bridge = src=FOO_ALL, alias=FOO]

  app_logging = true  // false is default
}
```

6 Publications and Subscriptions for uFldShoreBroker

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

```
$ uFldShoreBroker --interface or -i
```

6.1 Variables Published by uFldShoreBroker

The primary output of uFldShoreBroker to the MOOSDB are the requests to pShare for registrations, and the outgoing acknowledgment replies to remote node/vehicle 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.
- PMB_REGISTER: A 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.
- NODE_BROKER_ACK: A message written locally but bridged to a remote vehicle MOOS community, containing IP address and port information about the local shoreside community.

6.2 MOOS Variables Subscribed for by uFldShoreBroker

The uFldShoreBroker application subscribes to the following MOOS variables:

- APPCAST_REQ: A request to generate and post a new apppcast report, with reporting criteria, and expiration.
- 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.
- NODE_BROKER_PING: Information published presumably by uFldNodeBroker running in a remote vehicle community. Message has information about the node host including the community name, IP address, the port number for the MOOSDB, input route(s) for the local pShare process.

6.3 Command Line Usage of uFldShoreBroker

The uFldShoreBroker 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:

```
$      uFldShoreBroker --help or -h
```

Listing 6.4: Command line usage for the uFldShoreBroker tool.

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

Options:
  --alias=<ProcessName>
      Launch uFldShoreBroker with the given
      process name rather than uFldShoreBroker.
  --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 uFldShoreBroker.
```