Inter-Vehicle Messaging

Nov 2023

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

`project-pavlab/memos/memo_swarm_messaging`

1 Inter-Vehicle Messaging

A common thread of missions involving the Swarm Autonomy Toolbox involves *inter-vehicle messaging*. Even in the cases where coordination involves purely knowledge of collaborator positions, i.e., no inter-vehicle auctions, position information messages need to be exchanged if the vehicles are not sensing each other's position. The inter-vehicle messaging scheme supporting the Swarm Autonomy Toolbox is derived partly from tools in the public MOOS-IvP codebase and partly from tools currently in the Swarm Autonomy Toolbox. They are described here.

1.1 General Messaging

An inter-vehicle message contains at least four critical parts:

- Source node
- Destination node
- Message name (MOOS variable)
- Message content

All messages can be serialized into a simple string. For example, if the vehicle *abe* wishes to share a message with vehicle *ben* about a temperature measurement at a particular location, it may look like:

```
src=abe, dest=ben, varname=TEMP_MEASUREMENT, value="lat=43.825300, lon=-70.330400, temp=68.4"
```

This message, if it is an outgoing message on vehicle *abe*, is itself published to the local MOOS variable `NODE_MESSAGE_LOCAL`:

```
NODE_MESSAGE_LOCAL = src=abe, dest=ben, varname=TEMP_MEASUREMENT,
                     value="lat=43.825300, lon=-70.330400, temp=68.4"
```

This message will be routed ultimately to vehicle *ben*, an incoming message under the MOOS variable name `NODE_MESSAGE`. A MOOS app on the receiving vehicle receives the `NODE_MESSAGE` and unpacks the intended message. In the above example, the variable `TEMP_MEASUREMENT` is published on vehicle *ben*. The idea is shown in Figure 1.

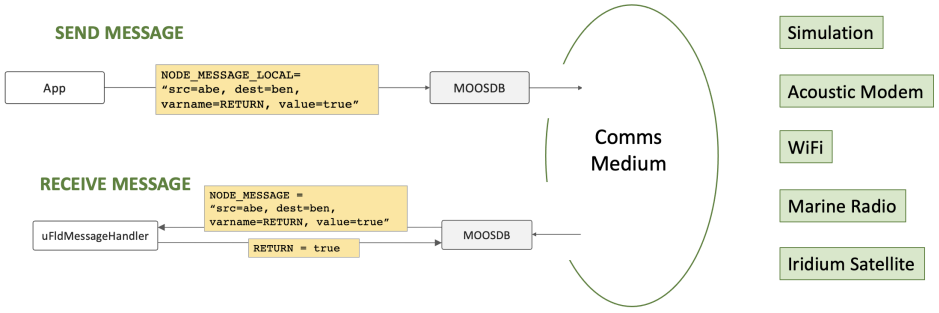

Figure 1: **General Abstract Flow of Messages**: Messages are packed as a single serialized string on the originating vehicle, and unpacked on the destination vehicle with intended message posted in the destination vehicle's local MOOSDB.

The above protocol and sequence of events is how inter-vehicle messaging is implemented at the very front end and back end the set of steps constituting a message from one vehicle to another. In between there may be queuing or prioritization, and any number of other intermediate steps depending on the comms medium, e.g., underwater acoustic comms, WiFi, marine radio, satellite and so on.

The above protocol existed prior to the Swarm Autonomy Toolbox, as part of the uField Toolbox, which is part of the MOOS-IvP public codebase.

1.2 Simulated Messaging

The inter-vehicle messaging scheme described above is implemented in simulation by routing all messages through a third-party (shoreside) MOOS community. The `uFldNodeComms` MOOS app is part of the public MOOS-IvP codebase and handles the routing of inter-vehicle messages. This app is aware of all vehicle positions and can simulate lossy and range-limit comms between vehicles. This configuration not only is used in pure simulation, but also on field deployments to simulate lossy comms while exercising actual vehicles.

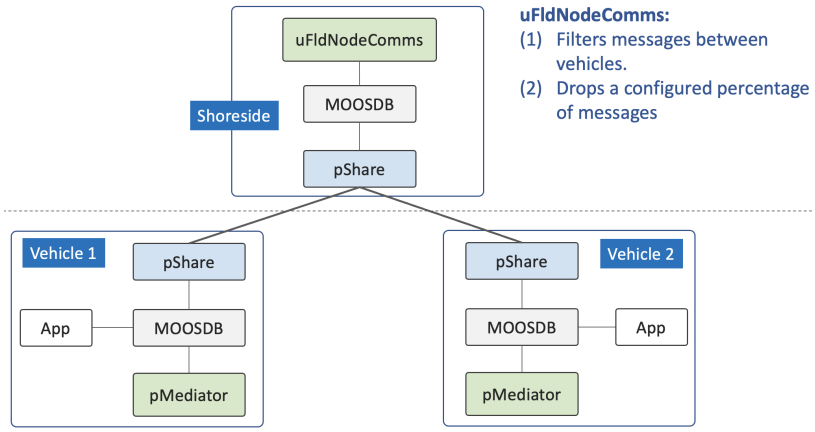

Figure 2: **Simulated Messaging**: In simulation, a shoreside community routes all inter-vehicle messages through the `uFldNodeComms`, which considers the range between vessels, time since a previous message, and other factors, to simulate lossy and range and bandwidth limited comms in the real world.

In the Swarm Autonomy Toolbox, a mediated messaging regime has been implemented in the `pMediator` app described in Section 1.3. The uField messaging regime of the public codebase has been augmented to handled mediated messages in the `uFldNodeComms` application.

1.3 Mediated Messaging

The Swarm Autonomy Toolbox contains a new app, `pMediator`, for adding a layer of send-acknowledge protocol on top of the bare inter-vehicle messaging. This app will address lossy-comms situations, enabling critical messages to be re-sent essentially until an acknowledgment has been received, or some fixed but configurable number of repeated tries. This app will also retain statistics on quality of communications between nodes to alert mission operators to worsening conditions before things become critical.

This app runs on each vehicle handling both outgoing and incoming messages. As an outgoing handler, it intercepts each outgoing message and assigns a unique message ID appended to the outgoing message. On the incoming handler, `pMediator` will send a return acknowledgment for each incoming message. The basic idea is shown in Figure 3.

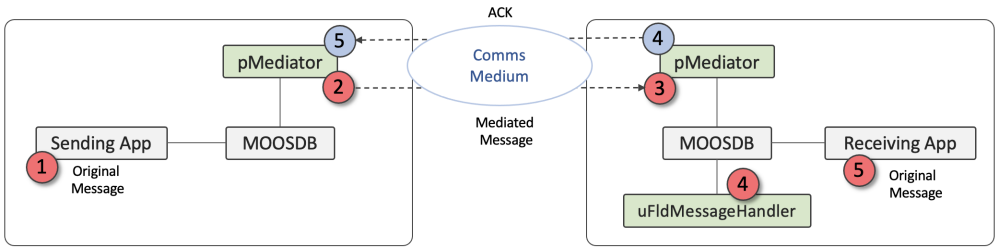

Figure 3: **Mediated Messaging**: In mediated messaging, (1) the originating app published an inter-vehicle message which (2) is intercepted by the `pMediator` app on the originating vehicle where a unique message ID is appended to the message. When (3) it is received on the destination vehicle, the message ID is noted, and (4) the message is passed on to the `uFldMessageHandler` app which then publishes the intended message to (5) the application that consumes the message. Upon receipt of the message on the destination vehicle, an acknowledgment message is sent back to the originating vehicle. The originating vehicle may repeat the message until an acknowledgment is received. If the destination vehicle receives the same message twice (same message ID), the subsequent messages are ignore and not posted locally more than once.

The first version of the `pMediator` app was completed in the winter of 2022, tested in simulation, and will be field-tested in the summer of 2022.