Lab #8
Simulation of Multi-Vehicle Operations
Part II

2.S998 Unmanned Marine Vehicle Autonomy, Sensing and Communications

Contents

1 Overview and Objectives

In today's lab, our objectives are dependent on your current progress in implementing the Lab 07 assignments. Depending on where you are in your work, today's focus will be on:

- Implementing the Lab 07 missions, with TA help if need be,
- Running your Lab 07 missions in the classroom simulation setup,
- An additional challenge problem - extending the lab 07 mission with inter-vehicle communications and offloading of mission components,
- An additional challenge problem - extending the Lab 07 mission with re-planning.

The goal of the first item above is self evident - we want to give everyone a chance to keep reasonably on pace in handing in lab 07 this Thursday, since there will be a new assignment going out on the same day.

The goal of the second item above is to begin migrating our lab work-flow to making use of an in-lab simulation environment where individual (or group) autonomy configurations are demonstrated. In short, the goal is to replace the shoreside MOOS community you have been running on your laptops with a single classroom shoreside community to which you connect your simulated vehicle. This is much closer to the setup that will be used during experimentation on the river.

The goal of the third item above is to begin experimenting with the uFldNodeComms and uFldMessageHandler modules for allowing inter-vehicle communications. These modules will be commonly used in most if not all later labs, so this provides a chance to begin getting familiar with them. They have not yet been discussed in lectures, but our objectives in this lab are fairly vanilla, and the documentation for both modules is available on the course wiki page.

The goal of the last item above is to gain further familiarity with behavior configurations and MOOS app development by implementing a further capability on your current Lab 07 assignment.

1.1 Preliminaries

Standard practice now before any lab should be to perform an svn update within your moos-ivp tree and rebuild if you see any updates pulled down from the SVN server. This lab does assume that you have a working MOOS-IvP tree checked out and installed on your computer. To verify this make sure that the following executables are built and findable in your shell path:

```
$ which MOOSDB
/Users/you/moos-ivp/MOOS/MOOSBin/MOOSDB
$ which uTimerScript
/Users/you/moos-ivp/bin/uTimerScript
$ which mykill
/Users/you/moos-ivp/scripts/mykill
```

1.2 More MOOS / MOOS-IvP Resources

The one document that may be new to the discussion in this lab is the documentation for the uField Toolbox linked below. See also the slides from today's lecture.

- The IvP Helm documentation.
  http://oceanai.mit.edu/moos-ivp-pdf/moosivp-helm.pdf.
- The uField Toolbox Documentation
  http://oceanai.mit.edu/moos-ivp-pdf/moosivp-ufield.pdf
- The moos-ivp.org website documentation.
  http://www.moos-ivp.org

2 Experimenting with Distributed Simulation

The first exercise in today's lab involves simulation distributed over more than one machine. Note that in lab 7, the mission configuration structure began to take the form of (a) single shoreside MOOS community running pMarineViewer, and (b) some number of vehicle communities. All of these communities are still typically running on a a single (e.g., your laptop) machine. When our labs move to the water, this of course will change, but the logical topology will be the same. As a step in this direction, in today's lab experiment with simulation distributed over multiple machines.

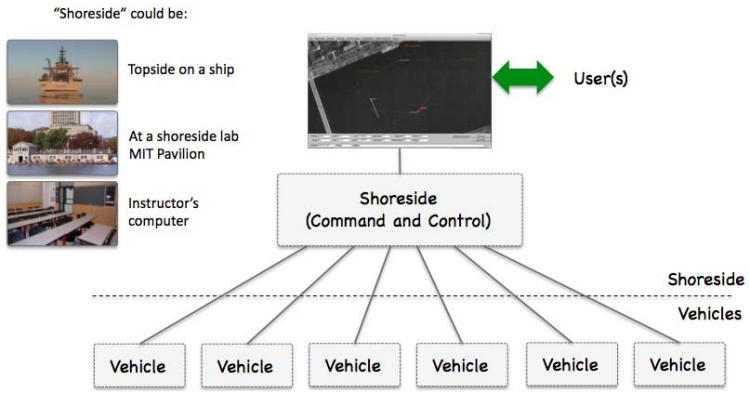

Figure 1: **Shoreside to Multi-Vehicle Topology:** A number of vehicles are deployed with each vehicle maintaining some level of connectivity to a shoreside command and control computer. Each node (vehicles and the shoreside) are comprised of a dedicated MOOS community. Modes and limits of communication may vary.

2.1 Running the Double-Loiter Mission Across Multiple Machines

In this part we will:

- Obtain a copy of the lab 8 baseline mission configuration.
- Confirm that it is runnable on your own machine.
- Work with a lab partner to test it over two machines.

Obtain a copy of the lab 8 baseline mission configuration

The first step is to copy the baseline example mission from the moos-ivp tree and copy it into your own moos-ivp-jsmith tree.

```
% cp -rp moos-ivp/ivp/missions/lab8baseline  moos-ivp-jsmith/missions/lab08one
```

The `lab08one` mission file should like:

```
% ls lab08one
  clean.sh*                plug_origin_warp.moos     plug_pNodeReporter.moos
  launch_shoreside.sh*     plug_pHelmIvP.moos        plug_uFldNodeBroker.moos
  launch_vehicle.sh*       plug_pHostInfo.moos       plug_uProcessWatch.moos
  meta_shoreside.moos      plug_pLogger.moos         plug_uSimMarine.moos
```

```
  meta_vehicle.bhv          plug_pMOOSBridgeV.moos
  meta_vehicle.moos         plug_pMarinePID.moos
```

As with previous missions, the actual mission files are auto generated upon executing the launch scripts. Here there are *two* launch scripts - one (`launch_shoreside.sh`) if you are launching the shoreside community on your machine and serving others, and the other (`launch_vehicle.sh`) if you are launching a vehicle community and connecting to a remote shoreside community. Both scripts support the `-h` switch and both support the `--just_build` switch if you want to just build the target files without actually launching. For example:

```
% cd lab08one
% launch_shoreside.sh --just_build
% launch_vehicle.sh --just_build
% ls
  clean.sh*                 plug_pHelmIvP.moos        plug_uProcessWatch.moos
  launch_shoreside.sh*      plug_pHostInfo.moos       plug_uSimMarine.moos
  launch_vehicle.sh*        plug_pLogger.moos         targ_jsmith@machinename.bhv
  meta_shoreside.moos       plug_pMOOSBridgeV.moos    targ_shoreside.moos
  meta_vehicle.bhv          plug_pMarinePID.moos      targ_vehicle.moos
  meta_vehicle.moos         plug_pNodeReporter.moos
  plug_origin_warp.moos     plug_uFldNodeBroker.moos
```

Note the three target files newly created.

Confirm that the lab08one mission is runnable on your own machine

The goal of this step is to launch two communities with the two specialized shell scripts provided in the `lab08one` (baseline copy) mission. Although the intention is that typically you would launch just one or the other, launching both on the same machine should work just fine. To to this, first launch the shoreside community with a time warp of ten:

```
$ ./launch_shoreside.sh 10
```

Then launch a vehicle community with a matching time warp:

```
$ ./launch_vehicle.sh --shore=localhost 10
```

In launching the vehicle community, the IP address of the shoreside community must also be provided. In this case everything is running on "localhost".

It may seem strange that there are two pMarineViewer communities launched. But this is more sensible when these two communities are being launched on separate machines, as intended normally. The shoreside pMarineViewer will render all vehicles. The vehicle pMarineViewer will render just what is known locally. This may include other vehicles under certain configurations.

Note that the name of the vehicle is also auto-generated with a `username@machine` pattern.

Work with a lab partner to launch the lab08one mission

In the next step, find a lab partner to work with launching the `lab08one` mission. Take turns playing the vehicle role and the shoreside role. If you want to work with multiple partners to launch several vehicles connected to a single shoreside community, that's fine too.

As before, the first step is to launch the shoreside community. Pick a desired time warp and inform those connecting with a vehicle of this value:

```
$ ./launch_shoreside.sh  <time-warp>
```

Noting the shoreside IP address: To next launch a vehicle community on another machine, the person launching the vehicle community needs to know the IP address of the shoreside community. Once this is known, the vehicle community is launched with:

```
$ ./launch_vehicle.sh --shore=<ip-address> --vname=<vname> --mport=<port> --lport=<port> <time-warp>
```

If you are only launching one vehicle per machine, the `--vname`, `--mport`, and `--lport` arguments may be omitted. The default values will suffice. They only need to be specified to make sure that when launching more than one vehicle on a single machine that the vehicle name, MOOSDB port and MOOSBridge UDPListen ports are unique for each machine.

See Section 4 at the end of this lab for some tips on determining your machine's IP address in OS X and Linux, especially from the command line.

Regarding the network IP address on the two machines, note that it's important that either:

- Both machines are on the same local network. This is the case when the IP address is assigned by a router perhaps in the room. In this case the IP address may begin with one of a couple special numbers, `192.168.x.x`, or `10.1.x.x`.
- Both machines are on the global network, with an IP address known to the rest of the Internet.

One way to test your network setup between the two machines is both of you are able to ping each other using the `ping` command. This utility is found at `/sbin/ping` on most machines. Test the ping in both directions.

2.2 Running the Lab 7 Double-Traverse Mission Across Multiple Machines

The idea in this part is to morph your `lab07two` assignment mission with the `lab08one` mission such that it is launchable on different machines. *In most if not all future labs in this course, this style of configuration will be standard.* You must be able to both (a) test the autonomy developed for a particular vehicle by running everything on your own machine, shoreside and vehicle, simply to make progress in development, and then (b) demonstrate the autonomy developed by launching your vehicle connected to a remote shoreside machine.

In this part we will:

- Create a copy of the `lab07two` mission for modifying

- Edit your new version to support connection to a remote shoreside community
- Test that the modified version still works on a single machine
- Launch two versions of your vehicle connected to a remote shoreside community

2.2.1 Make a copy of your lab07two mission

As stated above, the current goal may be described as morphing your `lab07two` mission with the `lab08one` mission. It's easier to do this by starting with a copy of the `lab07two` mission and bring in the changes needed from the `lab08two` mission. So the first step here is to copy the `lab07two` mission you have been building (or have built) for the final lab 07 assignment due later this week. This folder, prior to modification here, should be capable of launching two vehicles and a shoreside community, *all on one machine*. We will modify this now to be able to launch one or more vehicles connected to a remote shoreside community.

```
% cp -rp moos-ivp-jsmith/missions/lab07two moos-ivp-jsmith/missions/lab08two
```

2.2.2 Edit the mission to support connection to remote shoreside community

Use the example of the `lab08one` mission to modify your `lab08two` mission. Namely:

- Create dedicated launch scripts for the shoreside and vehicle launches, called launch_vehicle.sh and launch_shoreside.sh. A good start is to copy these in from your `lab08one` mission.
- Modify, if need be, your `plug_uFldNodeBroker.moos` file such that the shoreside hostip is defined by a macro. For example you may have:

  ```
  TRY_SHORE_HOST = hostip=localhost, port_udp=9200
  ```

  You'll want to change this to:

  ```
  TRY_SHORE_HOST = hostip=$(SHOREIP), port_udp=9200
  ```

- Make sure the `uFldShoreBroker` on the shoreside is configured to bridge postings to `VISIT_POINT` out to the vehicles. In the `uFldShoreBroker` config block, make sure the following line is present:

  ```
  BRIDGE   = src=VISIT_POINT_$V,  alias=VISIT_POINT
  ```

- Add a launching of pMarineViewer to the Antler block of the vehicles. This can be done by copying the block from `meta_vehicle.moos` in the `lab08one` mission. Strictly speaking this step is optional, but often you will want a local version of the viewer running on your machine when connected to a remote shoreside community, so you can see a bit of what is going on with visual feedback on your own screen.

2.2.3 Test that the modified version still works on a single machine

Even though your modified mission and launch configuration is built with the goal of testing on multiple machines, this all should still be launchable on a single machine. Verify this:

We use `timewarp=15`, but use what you want. Launch the shoreside community with:

```
% ./launch_shoreside.sh 15
```

Launch the first vehicle:

```
% ./launch_vehicle.sh 15 --shore=<ipaddress> --vname=alpha --mport=9201 --lport=9301
```

Launch the second vehicle:

```
% ./launch_vehicle.sh 15 --shore=<ipaddress> --vname=bravo --mport=9202 --lport=9302
```

Be sure to give your two vehicles distinct names and port numbers. Recall that the shoreside is hardwired to use port 9000 for the MOOSDB and port 9200 for its UDPListen port.

(FYI: When launching a vehicle connected to a remote shoreside, you don't have to specify the two port numbers on the command line. The defaults for these two values are `9201` and `9301`. As long as you're only launching one vehicle on your remote machine, these arguments are not necessary.)

2.2.4 Launch the vehicles connected to a remote shoreside community

In the next step, find a lab partner to work with launching the `lab08two` mission. Take turns playing the vehicle role and the shoreside role. If you want to work with multiple partners to launch several vehicles connected to a single shoreside community, that's fine too.

Note: some extra care may be needed to make sure that the view points sent to connected vehicles are sent only after the remote vehicles are connected. Also you may need to consider how your pPointAssign application on the shoreside knows the vehicle names of the remote vehicles, so that `VIEW_POINT_VNAME` *reflects that actual vehicle name used in the bridging. If you update from the* `moos-ivp` *tree, the* `uFldNodeBroker` *app has been augmented to additionally publish* `NODE_BROKER_VACK=alpha` *when the vehicle alpha has been connected to the shore. You can use this information in your* `pPointAssign` *app to dynamically configure the names of the vehicles to which it assigns points.*

If you would like, come see me and I will play the role of the shoreside community.

As before, the first step is to launch the shoreside community. Pick a desired time warp and inform those connecting with a vehicle of this value:

```
$ ./launch_shoreside.sh  <time-warp>
```

Noting the shoreside IP address: As before in `lab08one`, the person launching the vehicle community needs to know the IP address of the shoreside community. Once this is known, the vehicle community is launched with:

```
$ ./launch_vehicle.sh --shore=<ip-address> --vname=<vname> --mport=<port> --lport=<port> <time-warp>
```

If you are only launching one vehicle per machine, the `--vname`, `--mport`, and `--lport` arguments may be omitted. The default values will suffice. They only need to be specified to make sure that when launching more than one vehicle on a single machine that the vehicle name, MOOSDB port and MOOSBridge UDPListen ports are unique for each machine.

3 Adding Re-planning to the Lab 7 Double-Traverse Mission

The next exercise in today's lab is to augment the lab 7 mission (`lab07two`) with a re-planning component. A video of one solution to this mission (without re-planning) may be found in clip 13

Note that when the vehicle makes sharp turns, it often "achieves" its next waypoint by a wide margin. This is due to fortuitous settings of the capture radius and slip radius parameters in the waypoint behavior.

In this exercise, we will define a separate notion of what it means to have successfully visited a waypoint. We augment the `pGenPath` application to keep track of which points were visited by its own observation and standard for achieving a visit. Once the vehicle has completed its first pass on the waypoints, it will repeatedly go back to visit waypoints it missed the first time.

Here's what we need to do:

- Augment the `pGenPath` application to keep track of which points a vehicle actually visits. The `pGenPath` app may be configured with a parameter `visit_radius=N`, setting the criteria for visiting a waypoint to be within N meters. The default should be 10. (Note the app will have to subscribe to `NAV_X` and `NAV_Y` if it isn't doing so already.)
- Configure your autonomy mission such that when it ends and returns to home, it posts a message to `GENPATH_REGENERATE`.
- The `pGenPath` app registers for this message and regenerates a new path consisting of all the points missed on the previous pass.
- Once a new path has been regenerated, the vehicle resumes its mission until the new path has been traversed.
- The mission continues until there are no points left unvisited (as determined by the `pGenPath` application).

A video of one solution to this mission may be found in clip 21

4 Help in Determining Your Machine's IP Address

In Linux, the IP address may address may be found by the following means (among perhaps many ways), from the command line:

```
$ ifconfig eth1
```

This is assuming that `eth1` is your network interface. Other possibilities are `eth0`, `wlan0`, and so on, depending on your particular computer. (Type `ifconfig` with no argument to see a list of interfaces). If you want to use grep to prune out some of the verbiage, try the following on the command line:

```
$ ifconfig eth1 | grep 'inet addr:'| grep -v '127.0.0.1' | cut -d: -f2 | awk '{ print $1}
```

On OS X, try the following

```
$ networksetup -getinfo <interface>
```

where `interface` is either `"Wi-Fi"` on OS X Lion, or `"Airport"` on pre-Lion machines, or `"Ethernet"` if you're connected by an ethernet cable. The IP address may also be found by using the System Preferences GUI interface under the Network section. Select the active interface (the one with the green button at the top of the list). Select Advanced. Then select TCP/IP. The IP address should be then visible.

2.S998 Marine Autonomy, Sensing and Communications

Spring 2012