Practical Session 3

In this final practical session, we will take the DCCL message that we made yesterday and send it using pAcommsHandler and a very simple CTD simulator that we will write first.

CTD Simulator

Copy the code in `goby3-course/src/moos/pattern/` to a new directory (`goby3-course/src/moos/ctd`) and rename all the files and class names to `pGobyCourseCTDSim`.

Also add your new directory to the `CMakeLists.txt` in `goby3-course/src/moos`.

In the `loop()` method (called by `CMOOSApp::Iterate` at the AppTick frequency), generate a randomly populated CTD message using your DCCL message from yesterday and publish it (using `publish_pb`) to the MOOS variable "CTD_DATA_OUT".

Update the moos mission

In `goby3-course/launch/moos`, add your CTD Simulator (`pGobyCourseCTDSim`) to the ANTLER launch list and create a ProcessConfig block for it in `auv.moos`.

Make sure that it runs and that you see the CTD_DATA_OUT in uMS:

```
pAntler auv.moos


uMS


(connect to port 9000)
```

Add to pAcommsHandler

In the `ProcessConfig = pAcommsHandler` block of both `auv.moos` and `topside.moos`, add your new CTD message to the `queue_cfg` and `translator_entry` blocks. Use the "CTD_DATA_OUT" variable as the "trigger" and the "create" variable and the "CTD_DATA_IN" as the "publish" variable.

Some things to consider (which are often more important in a real system than in this simple example):

- How should you prioritize the CTD data relative to the other messages (currently just the NavigationReport)
- Do you want acknowledgments for each CTD message received or would it better to avoid the extra mini-packet in the water?
- Do you want to send the newest sample first (`newest_first: true`), or send them in the order they are generated (`newest_first: false`)?

Keep in mind that the queue_cfg must exist for both sides of the link, but in pAcommsHandler only the *sender's* (in this case the AUV) `queue_cfg` values matter (using Goby3's gobyd either the subscriber or the publisher can set these values).

See your data on the topside

Run both the AUV and the topside and check that your CTD_DATA_IN values are reaching the topside MOOS community (using uMS on port 9001).

Bonus

Switch to using the WHOI Micro-Modem instead of the UDP Multicast driver by changing the `driver_cfg` and `mac_cfg` on the AUV (`auv.moos`) to:

```
driver_cfg {
     driver_type: DRIVER_WHOI_MICROMODEM
     serial_port: "/tmp/ttymm0"
     serial_baud: 19200
     [goby.acomms.micromodem.protobuf.config] {
        reset_nvram: true
     }
}
mac_cfg {
    type: MAC_FIXED_DECENTRALIZED
    slot { src: 1 rate: 1 slot_seconds: 10 }
    slot { src: 3 rate: 1 slot_seconds: 10 }
}
```

and the same for the topside (`topside.moos`) except for the serial port (`/tmp/ttymm1` instead of `/tmp/ttymm0`).

Also:

- Make sure you're connected to NETSIM and using the `./netsim_pty.sh` script as we did yesterday.
- Set `MOOSTimeWarp = 1` as you can't use the real Micro-Modems faster than real speed.

Relaunch the AUV and topside and after a minute or so you should start seeing the vehicle (every 20 seconds):

To see what's happening in pAcommsHandler you can always attach the screen sessions:

```
screen -r auv.pAcommsHandler
screen -r topside.pAcommsHandler
```