Setting up MOOS Development Environment

Version 1.1

10/10/2021

Outline

[1. Installing and Building MOOS system from source 3](#_Toc84833592)

[2. Downloading the moos-ivp-extend tree for our own application 3](#_Toc84833593)

[3. Creating your own MOOSapp in moos-ivp-extend --- An Odometry MOOS App 3](#_Toc84833594)

[4. Integrating AquaNet code into MOOS 6](#_Toc84833595)

[4.1 Building pOdometry-Publisher on machine A (192.168.80.129) 7](#_Toc84833596)

[4.2 Building pOdometry-Subscriber on machine B (192.168.80.130) 9](#_Toc84833597)

[5. Running AquaNet with MOOS 10](#_Toc84833598)

[5.1 Running aquanet on the machine A 10](#_Toc84833599)

[5.2 Running aquanet on the machine B 11](#_Toc84833600)

[5.3 Runing publisher on machine A 12](#_Toc84833601)

[5.4 Running subscriber on machine B 12](#_Toc84833602)

[5.5 Running results is showing as the following figure 12](#_Toc84833603)

[6. pShare based Inter-vehicle communication in MOOS (Two Machines) 13](#_Toc84833604)

[6.1 Machine A (shoreside computer) side 13](#_Toc84833605)

[6.2 Machine B (vehicle) side 14](#_Toc84833606)

[6.3 Test communication between two moos community 14](#_Toc84833607)

[7. Creating our own share application--- pAquaMessage 16](#_Toc84833608)

[7.1 Single way communication configuration (AUV -> Offshore PC) 18](#_Toc84833609)

[7.2 Two-way communication configuration (AUV <-> Offshore PC) 19](#_Toc84833610)

Ubuntu 20.04

# 1. Installing and Building MOOS system from source

-Downloading the moos-ivp and building

svn co https://oceanai.mit.edu/svn/moos-ivp-aro/trunk/ moos-ivp

cd moos-ivp

svn update

sudo apt-get install g++ cmake xterm

sudo apt-get install libfltk1.3-dev freeglut3-dev libpng-dev libjpeg-dev

sudo apt-get install libxft-dev libxinerama-dev libtiff5-dev

./build-moos.sh

(let it build)

./build-ivp.sh

(let it build)

-Modifying the bash file to run the script within MOOS

cd ~

vim .bashrc

export PATH=”$PATH:/home/moos/moos-ivp/bin”

export MOOSROOT=/home/moos/moos-ivp

source .bashrc

cd $MOOSROOT/ivp/missions/s1\_alpha

pAntler alpha.moos

- If the “pAntler alpha.moos” can run, and the map with a vehicle shows up, the MOOS has been installed successfully

# 2. Downloading the moos-ivp-extend tree for our own application

svn co https://oceanai.mit.edu/svn/moos-ivp-extend/trunk moos-ivp-extend

cd moos-ivp-extend

./build.sh

export PATH="$PATH:/home/moos/moos-ivp-extend/bin"

# 3. Creating your own MOOSapp in moos-ivp-extend --- An Odometry MOOS App

-In order to use GenMOOSApp\_AppCasting , the following path need to be included

export PATH="$PATH:/home/moos/moos-ivp/scripts"

GenMOOSApp\_AppCasting Odometry p "Jane Doe"

-Add Your New Application to the Build System

vim moos-ivp-extend/src/CMakeLists.txt

#==========================================================================

# List the subdirectories to build...

#==========================================================================

ADD\_SUBDIRECTORY(lib\_behaviors-test)

ADD\_SUBDIRECTORY(pXRelayTest)

ADD\_SUBDIRECTORY(pExampleApp)

ADD\_SUBDIRECTORY(pOdometry) <-- Add this line

-Build the new created application

cd moos-ivp-extend

./build.sh

-Verify the pOdometry is in your shell path with:

which pOdometry

/home/you/moos-ivp-you/bin/pOdometry

--Write the application

cd moos-ivp-extend/src/pOdometry

vim Odometry.h (see the figure “Odometry.h”)

vim Odometry.cpp (see the code “Odometry.cpp”)

Odometry.h

https://github.com/shuaidong-networking/AquaNet-MOOS/blob/main/pOdometry/pOdometry.h

Odometry.cpp

https://github.com/shuaidong-networking/AquaNet-MOOS/blob/main/pOdometry/pOdometry.cc

--Every time you modify the code, you need to run the “./build.sh” under the “moos-ivp-extend/”

-Testing the new created pOdometry app in the Alder misstion

--Run the un-modified Alder Mission

cd moos-ivp-extend/missions/alder

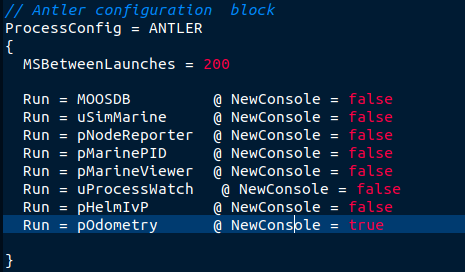
pAntler --MOOSTimeWarp=10 alder.moos



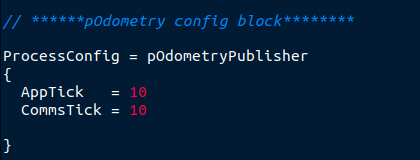
--Run the modified Alder Mission

--Modify the “alder.moos” file

--Add the “Run = pOdometry @ NewConsole = true”



--Add pOdometry config block (setting the running frequency)

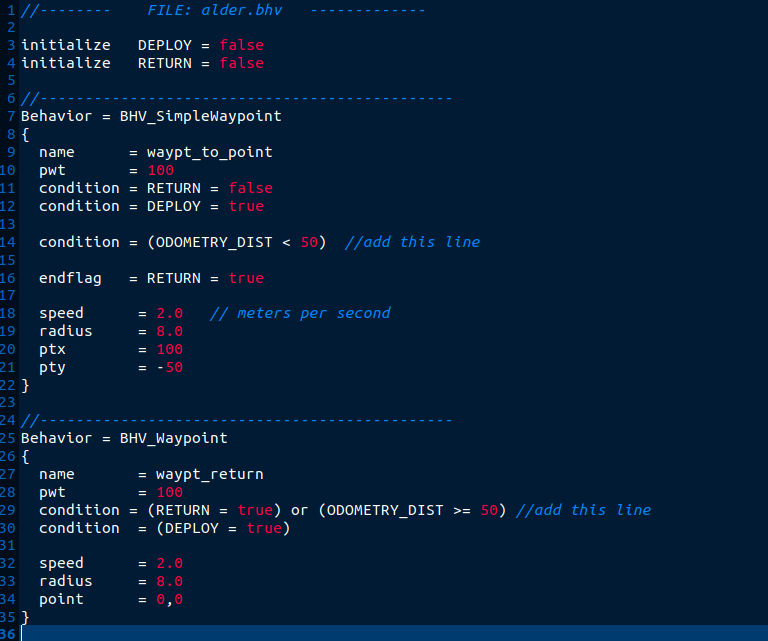


--Modify the alder.bhv file (configurations on Helm)

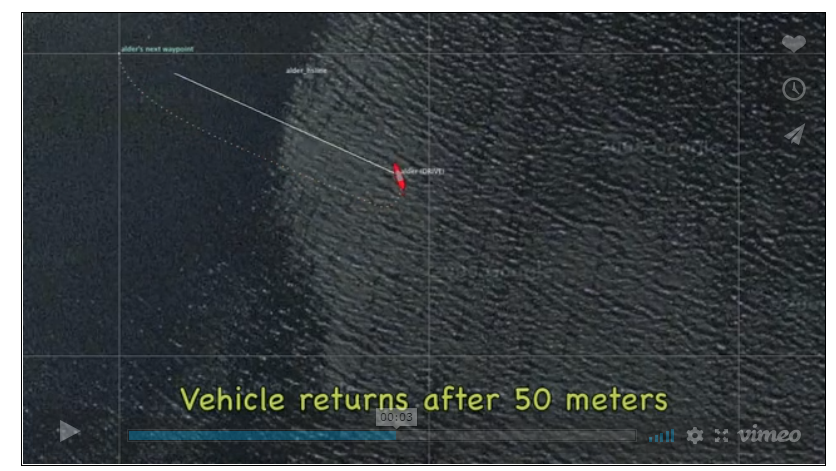
--Add two conditions into alder.bhv file

condition = (ODOMETRY\_DIST < 50)

condition = (RETURN = true) or (ODOMETRY\_DIST >= 50)

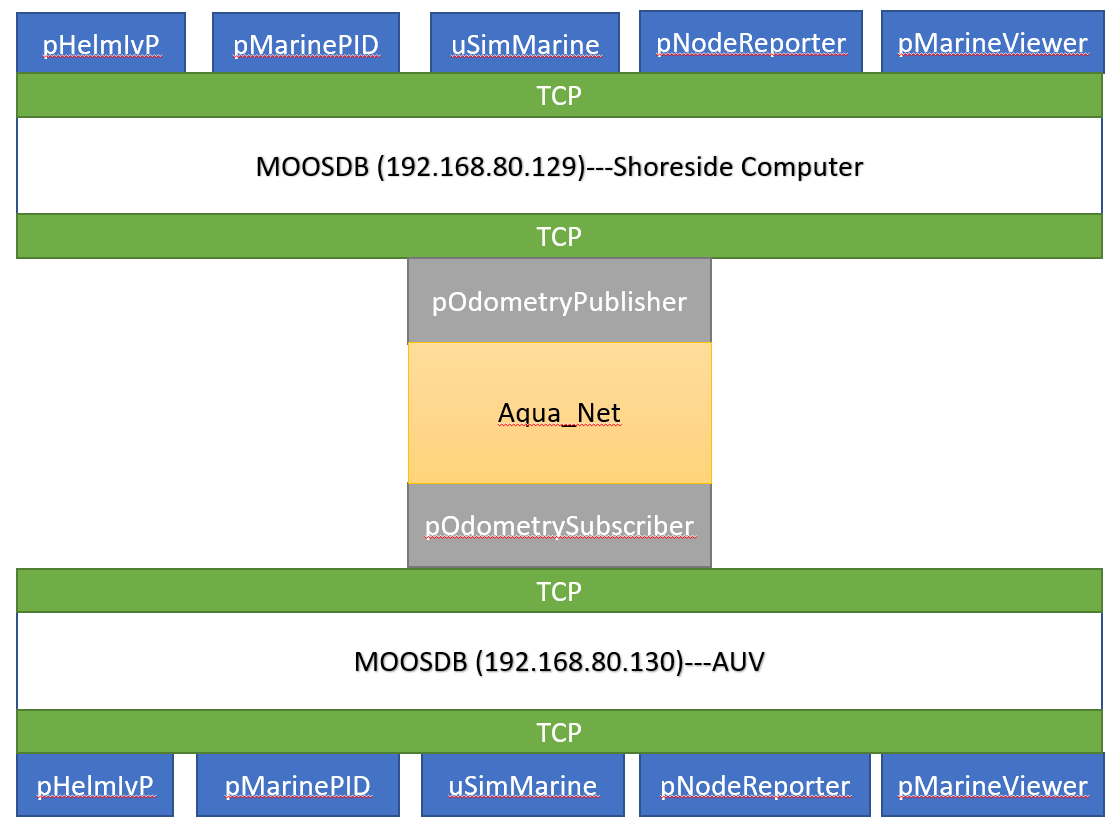


--Rebuild the code and run the mission again



# 4. Integrating AquaNet code into MOOS

In this section, we will talk about how to integrate AquaNet into MOOS communication system. The interface is shown as the following figure.



In order to use the AquaNet socket interface inside MOOS, the following steps should be made:

(Two virtual machines, each machine should install the MOOS according to the instructions mentioned above)

-Get and compile the latest aquanet code:

Install subversion:

sudo apt-get install subversion

-Get the code

svn co svn+ssh://dmitrii@hudson.ccny.cuny.edu/var/svn/repos/aquanet aquanet

export AQUANET\_FOLDER=/home/ubuntu/aquanet

-Install dependencies

sudo apt-get install libglib2.0-dev

sudo apt-get install libgsl-dev

-Build AquaNet

cd $AQUANET\_FOLDER/trunk

make

Now, the AquaNet is built and downloaded successfully in a separate folder. The step should be executed on both machines.

## 4.1 Building pOdometry-Publisher on machine A (192.168.80.129)

-Create the pOdometryPublisher moos application

GenMOOSApp\_AppCasting OdometryPublisher p "Jane Doe"

-Modify the CMakeList

vim moos-ivp-extend/src/CMakeLists.txt

ADD\_SUBDIRECTORY(lib\_behaviors-test)

ADD\_SUBDIRECTORY(pXRelayTest)

ADD\_SUBDIRECTORY(pExampleApp)

ADD\_SUBDIRECTORY(pOdometryPublisher) **<-- Add this line**

-Build the new created application

cd moos-ivp-extend

./build.sh

-Verify the pOdometryPublisher is in your shell path with:

which pOdometryPublisher

/home/you/moos-ivp-you/bin/pOdometryPublisher

--Write the application

cd moos-ivp-extend/src/pOdometryPublisher

vim OdometryPublisher.h (see the figure “OdometryPublisher.h”)

vim OdometryPublisher.cpp (see the code “OdometryPublisher.cpp”)

--pOdometryPublisher.h could be found at:

https://github.com/shuaidong-networking/AquaNet-MOOS/blob/main/pOdometryPublisher/pOdometryPublisher.h

-- pOdometryPublisher.cc could be found at:

https://github.com/shuaidong-networking/AquaNet-MOOS/blob/main/pOdometryPublisher/pOdometryPublisher.cc

-In order to use AquaNet API, some header files from aquanet should be link to the pOdometryPublisher folder.

-Header files

ln -s $AQUANET\_FOLDER/trunk/aquanet\_log.h moos-ivp-extend/src/pOdometryPublisher/aquanet\_log.h

ln -s $AQUANET\_FOLDER/trunk/aquanet\_netif.h moos-ivp-extend/src/pOdometryPublisher/aquanet\_netif.h

ln -s $AQUANET\_FOLDER/trunk/aquanet\_pdu.h moos-ivp-extend/src/pOdometryPublisher/aquanet\_pdu.h

ln -s $AQUANET\_FOLDER/trunk/aquanet\_socket.h moos-ivp-extend/src/pOdometryPublisher/aquanet\_socket.h

ln -s $AQUANET\_FOLDER/trunk/aquanet\_time.h moos-ivp-extend/src/pOdometryPublisher/aquanet\_time.h

-Configuration files

ln -s $AQUANET\_FOLDER/trunk/test\_example/mesh/node1/config\_add.cfg moos-ivp-extend/src/pOdometryPublisher/config\_add.cfg

ln -s $AQUANET\_FOLDER/trunk/test\_example/mesh/node1/config\_arp.cfg moos-ivp-extend/src/pOdometryPublisher/config\_arp.cfg

ln -s $AQUANET\_FOLDER/trunk/test\_example/mesh/node1/config\_conn.cfg moos-ivp-extend/src/pOdometryPublisher/config\_conn.cfg

ln -s $AQUANET\_FOLDER/trunk/test\_example/mesh/node1/config\_net.cfg moos-ivp-extend/src/pOdometryPublisher/config\_net.cfg

-Compile the new created application

cd moos-ivp-extend

./build.sh

-Copy alder.moos file to pOdometryPublisher folder and modify it

cd moos-ivp-extend/missions/alder

vim alder.moos

-Code for alder.moos files could be found at

https://github.com/shuaidong-networking/AquaNet-MOOS/blob/main/pOdometryPublisher/alder.moos

## 4.2 Building pOdometry-Subscriber on machine B (192.168.80.130)

-Create the pOdometrySubscriber moos application

GenMOOSApp\_AppCasting pOdometrySubscriber p "Jane Doe"

-Modify the CMakeList

vim moos-ivp-extend/src/CMakeLists.txt

# List the subdirectories to build...

#==========================================================================

ADD\_SUBDIRECTORY(lib\_behaviors-test)

ADD\_SUBDIRECTORY(pXRelayTest)

ADD\_SUBDIRECTORY(pExampleApp)

ADD\_SUBDIRECTORY(pOdometrySubscriber) <-- Add this line

-Building the new created application

cd moos-ivp-extend

./build.sh

-Verify the pOdometryPublisher is in your shell path with:

which pOdometryPublisher

/home/you/moos-ivp-you/bin/pOdometrySubscriber

--Write the application

cd moos-ivp-extend/src/pOdometrySubscriber

vim pOdometrySubscriber.h (see the figure “pOdometrySubscriber.h”)

vim pOdometrySubscriber.cpp (see the code “pOdometrySubscriber.cpp”)

--Code for pOdometrySubscriber.h can be found at:

https://github.com/shuaidong-networking/AquaNet-MOOS/blob/main/pOdometrySubscriber/pOdometrySubscriber.h

--Code for pOdometrySubscriber.cc can be found at:

https://github.com/shuaidong-networking/AquaNet-MOOS/blob/main/pOdometrySubscriber/pOdometrySubscriber.cc

-In order to use AquaNet API, some header files from aquanet shoulde be link to the pOdometryPublisher folder.

-Header files

ln -s $AQUANET\_FOLDER/trunk/aquanet\_log.h moos-ivp-extend/src/pOdometrySubscriber/aquanet\_log.h

ln -s $AQUANET\_FOLDER/trunk/aquanet\_netif.h moos-ivp-extend/src/ pOdometrySubscriber /aquanet\_netif.h

ln -s $AQUANET\_FOLDER/trunk/aquanet\_pdu.h moos-ivp-extend/src/ pOdometrySubscriber /aquanet\_pdu.h

ln -s $AQUANET\_FOLDER/trunk/aquanet\_socket.h moos-ivp-extend/src/ pOdometrySubscriber /aquanet\_socket.h

ln -s $AQUANET\_FOLDER/trunk/aquanet\_time.h moos-ivp-extend/src/ pOdometrySubscriber /aquanet\_time.h

-Configuration files

ln -s $AQUANET\_FOLDER/trunk/test\_example/mesh/node2/config\_add.cfg moos-ivp-extend/src/ pOdometrySubscriber /config\_add.cfg

ln -s $AQUANET\_FOLDER/trunk/test\_example/mesh/ node2/config\_arp.cfg moos-ivp-extend/src/ pOdometrySubscriber /config\_arp.cfg

ln -s $AQUANET\_FOLDER/trunk/test\_example/mesh/ node2/config\_conn.cfg moos-ivp-extend/src/ pOdometrySubscriber /config\_conn.cfg

ln -s $AQUANET\_FOLDER/trunk/test\_example/mesh/ node2/config\_net.cfg moos-ivp-extend/src/ pOdometrySubscriber /config\_net.cfg

-Compile the new created application

cd moos-ivp-extend

./build.sh

-Moving the alder.moos file to pOdometrySubscriber floder and modify

cd moos-ivp-extend/missions/alder

vim alder.bhv

vim alder.moos

-Alder.bhv

https://github.com/shuaidong-networking/AquaNet-MOOS/blob/main/pOdometrySubscriber/alder.bhv

-Alder.moos

https://github.com/shuaidong-networking/AquaNet-MOOS/blob/main/pOdometrySubscriber/alder.moos

# 5. Running AquaNet with MOOS

## 5.1 Running aquanet on the machine A

-Running AquaNet Virtual Modem Server to emulate L1

cd $AQUANET\_FOLDER/trunk

./bin/aquanet-vmds 2021

-Create a bash-script inside the pOdometryPublisher folder to start the layers from L2-L4:

cd moos-ivp-extend/src/pOdometryPublisher

touch publisher.sh

-Insert the following code into publisher.sh file

#!/bin/sh

# Initialize L2-L4 modules on localhost

# start the protocol stack

$AQUANET\_FOLDER/trunk/bin/aquanet-stack &

sleep 2

# start the VMDM

# modify IP and Port number, if necessary

$AQUANET\_FOLDER/trunk/bin/aquanet-vmdc 192.168.80.129 2021 1 20 20 20 &

sleep 4

# start the MAC

$AQUANET\_FOLDER/trunk/bin/aquanet-uwaloha &

sleep 2

$AQUANET\_FOLDER/trunk/bin/aquanet-sroute &

sleep 2

# start the transport layer

$AQUANET\_FOLDER/trunk/bin/aquanet-tra &

-Run the script

chmod +x publisher.sh

./publisher.sh

## 5.2 Running aquanet on the machine B

-Create a bash-script inside the pOdometrySubscriber folder to start the layers from L2-L4:

cd moos-ivp-extend/src/pOdometrySubscriber

touch subscriber.sh

-Insert the following code into subscriber.sh file

#!/bin/sh

# Initialize L2-L4 modules on localhost

# start the protocol stack

$AQUANET\_FOLDER/trunk/bin/aquanet-stack &

sleep 2

# start the VMDM

# modify IP and Port number, if necessary

$AQUANET\_FOLDER/trunk/bin/aquanet-vmdc 192.168.80.129 2021 2 20 20 20 &

sleep 4

# start the MAC

$AQUANET\_FOLDER/trunk/bin/aquanet-uwaloha &

sleep 2

$AQUANET\_FOLDER/trunk/bin/aquanet-sroute &

sleep 2

# start the transport layer

$AQUANET\_FOLDER/trunk/bin/aquanet-tra &

-Run the script

chmod +x subscriber.sh

./ subscriber.sh

## 5.3 Runing publisher on machine A

-Start the pOdometryPublisher

cd moos-ivp-extend/src/pOdometryPublisher

pAntler alder.moos

## 5.4 Running subscriber on machine B

-Start the pOdometrySubscriber

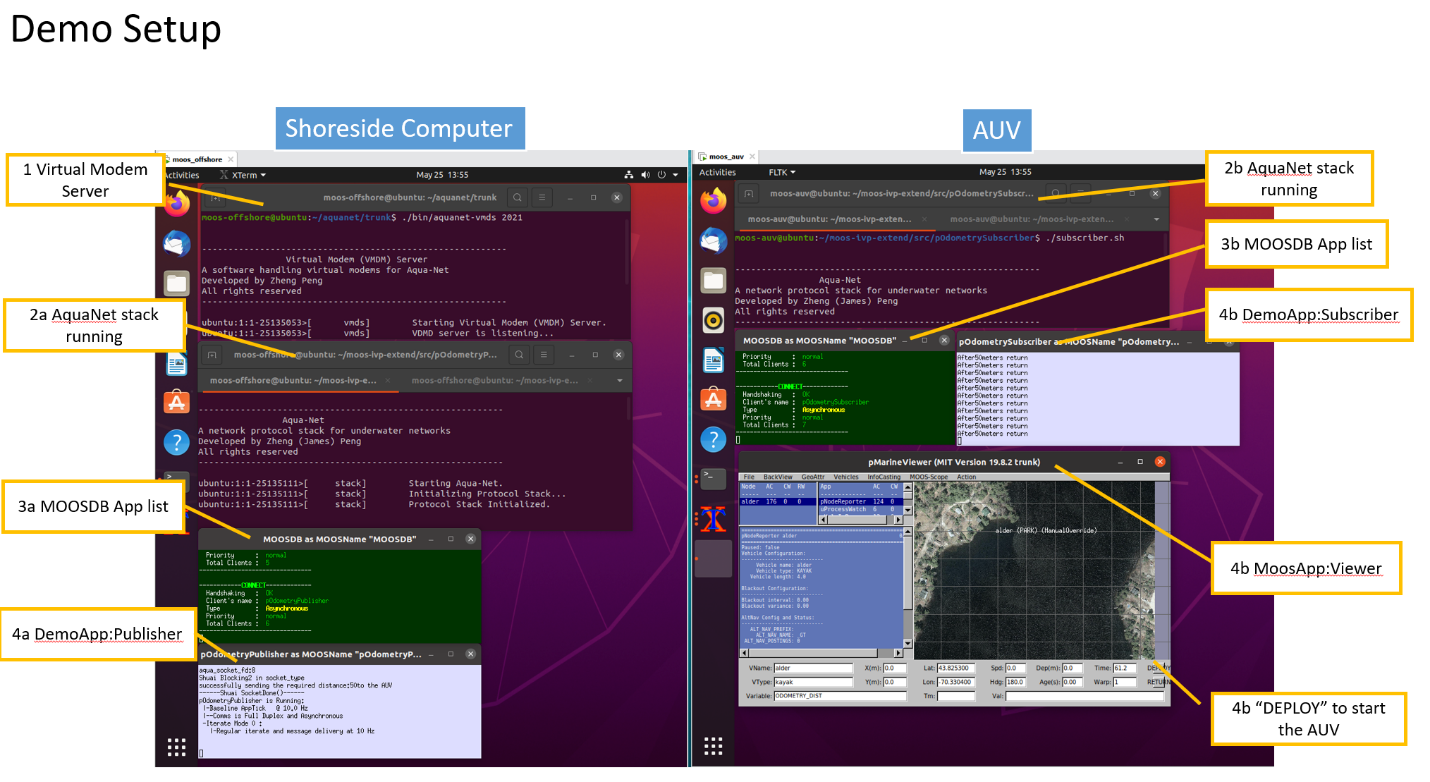
cd moos-ivp-extend/src/ pOdometrySubscriber

pAntler alder.moos

## 5.5 Running results is showing as the following figure

When all the programs have been executed, several terminals will be shown as the following figure. The execution order of the program is:

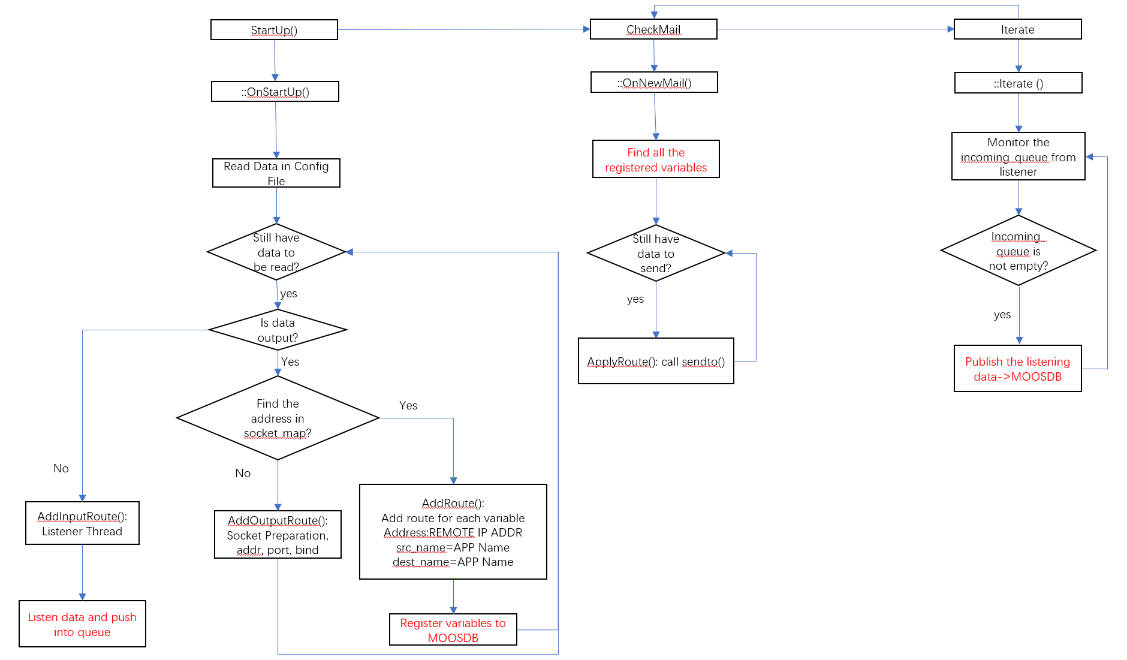
1. Running virtual modem server on shoreside pc side
2. Running AquaNet stack on both shoreside pc and AUV side by using the script mentioned earlier
3. Running the alder.moos file on both ends separately
4. MOOSDB and our demo will be started



-Click the DEPLOY button and you will see the AUV change the next waypoint at 50 meters instead of 100 meters.

# 6. pShare based Inter-vehicle communication in MOOS (Two Machines)

This section mainly talks about the interface called pShare that builds the connection between two machines. The structure of the pShare is shown as the following figure.



There are three components in the pShare source code including startup(), checkmail(), and iteratre(). In startup() function, it firstly reads the configuration file add it into socket according to the ip address and registers the required application names from MOOSDB. Then the checkmail() function is used to find the registered variables and use the sendto() function in internet socket to forward the data to the remote machine. Finally, the iterate() function starts working, it puts a thread called listener to receive the data from remote machine and publishes the received data to MOOSDB.

In this step, we create two separate MOOS communities: *a shoreside* community on machine A and *an alpha* community on machine B by creating two separate moos files. In the shoreside community there will be a MOOSDB and pMarineViewer. In the alpha community will be a MOOSDB and everything but pMarineViewer. In both community you will also need to add a pShare configuration block and add pShare to the Antler configuration block.

Machine A IP address: 192.168.80.129

Machine B IP address: 192.168.80.130

## 6.1 Machine A (shoreside computer) side

-Create a folder for Alpha Mission’s MOOS file

cd moos-ivp-extend/missions/

mkdir alpha

-Make a copy of the Alpha Mission into the MOOS-IVP-EXTEND

cd ~

cp -rp moos-ivp/ivp/missions/s1\_alpha moos-ivp-extend/missions/alpha/

-Replace the name of alpha.moos file with shoreside.moos

cd moos-ivp-extend/missions/alpha/s1\_alpha

mv alpha.moos shoreside.moos

-Modify the shoreside.moos file

vim shoreside.moos

-Copy the following content to overwrite the original content in shoreside.moos, add the pShare block into the moos file.

ProcessConfig = ANTLER

{

MSBetweenLaunches = 200

Run = MOOSDB @ NewConsole = true

Run = pMarineViewer @ NewConsole = false

Run = pShare @ NewConsole = true //adding this line

}

//------------------------------------------

// adding pShare config block by Shuai

ProcessConfig = pShare //pShare configuration

{

AppTick = 4

CommsTick = 4

input = route = localhost:9201

output = src\_name=MOOS\_MANUAL\_OVERRIDE, route=192.168.80.130:9200

output = src\_name=DEPLOY, route=192.168.80.130:9200 //sending status of RETURN and DEPLOY and MOOS\_MANUAL\_OVERRIDE

output = src\_name=RETURN, route=192.168.80.130:9200

}

## 6.2 Machine B (vehicle) side

-Do the same operations as machine A side except the step of modifying the content of alpha.moos file.

-Copy the following content to overwrite the original one

ProcessConfig = ANTLER

{

MSBetweenLaunches = 200

Run = MOOSDB @ NewConsole = false

Run = pLogger @ NewConsole = false

Run = uSimMarine @ NewConsole = false

Run = pMarinePID @ NewConsole = false

Run = pHelmIvP @ NewConsole = false

Run = uProcessWatch @ NewConsole = false

Run = pNodeReporter @ NewConsole = false

Run = pShare @ NewConsole = false //add pShare configuration

}

//------------------------------------------

// adding pShare config block by Shuai

ProcessConfig = pShare

{

AppTick = 4

CommsTick = 4

input = route = localhost:9200

output = src\_name=NODE\_REPORT\_LOCAL, dest\_name=NODE\_REPORT, route=192.168.80.129:9201

output = src\_name=VIEW\_SEGLIST, route=192.168.80.129:9201

output = src\_name=VIEW\_POINT, route=192.168.80.129:9201

}

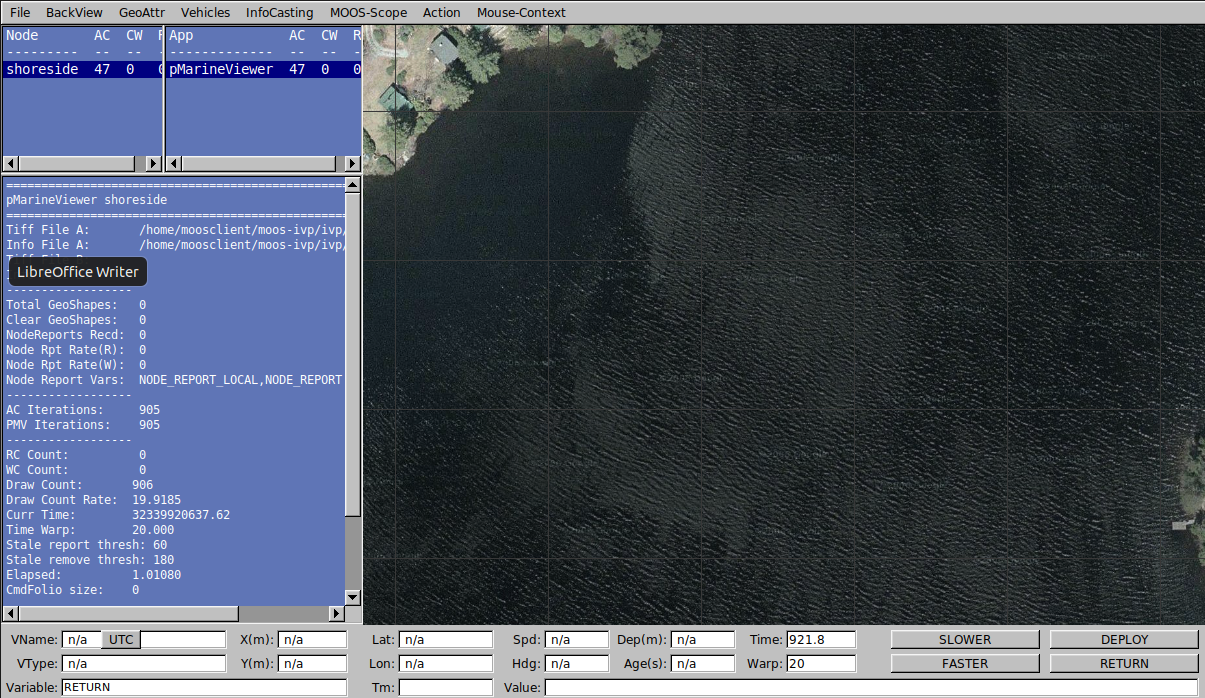
## 6.3 Test communication between two moos community

-Launch shoreside.moos file on machine A

cd moos-ivp-extend/missions/alpha/s1\_alpha

pAntler shoreside.moos

-After the shoreside.moos file is executed, you will see the following figure

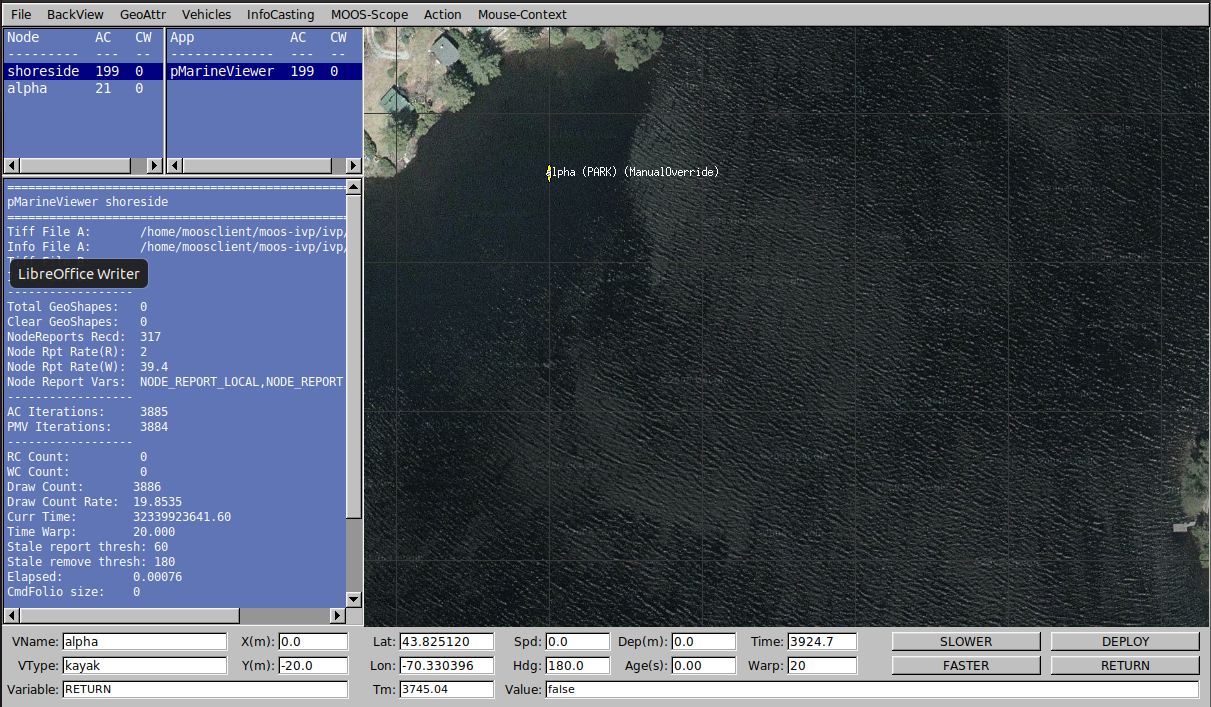


-Launch the alpha.moos file on machine B

cd moos-ivp-extend/missions/alpha/s1\_alpha

pAnlter alpha.moos

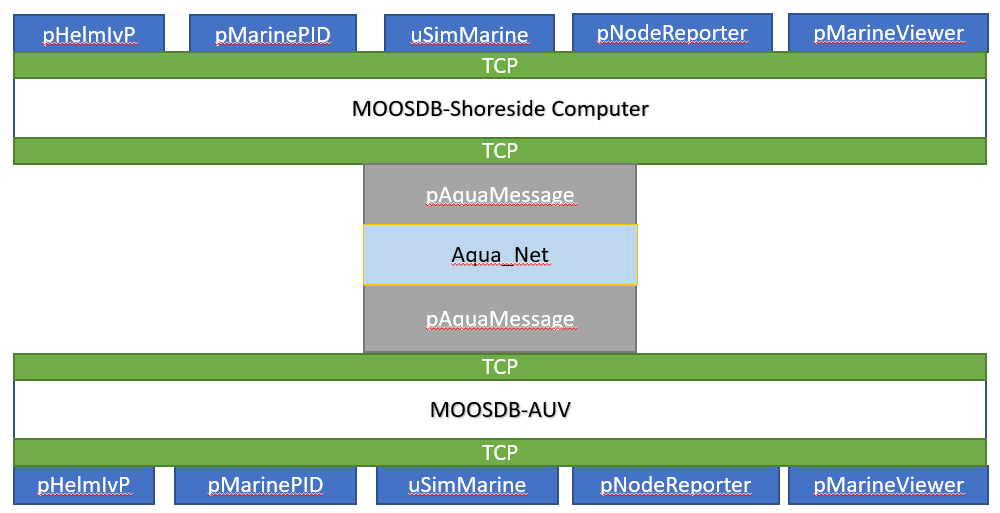
-After the alpha.moos is executed, a vehicle will be shown on machine A’s map.



-Click the “DEPLOY” button, you will see that the vehicle is moving

# 7. Creating our own share application--- pAquaMessage

This section creates a new MOOS application that has the same functionality with pShare that uses the internet socket. In this new application, we replace the internet socket with the AquaNet socket. The interface is shown as the following figure.



-Creating an application under “moos-ivp-extend/src/”

cd moos-ivp-extend/src

GenMOOSApp\_AppCasting AquaMessage p "Author\_Name"

-Make the pAquaMessage application executable

-Modify the CMakeList

vim moos-ivp-extend/src/CMakeLists.txt

ADD\_SUBDIRECTORY(lib\_behaviors-test)

ADD\_SUBDIRECTORY(pXRelayTest)

ADD\_SUBDIRECTORY(pExampleApp)

ADD\_SUBDIRECTORY(pOdometry)

ADD\_SUBDIRECTORY(pAquaMessage) //add this line

- Delete all files in pAquaMessage folder

cd pAquaMessage

rm –f \*

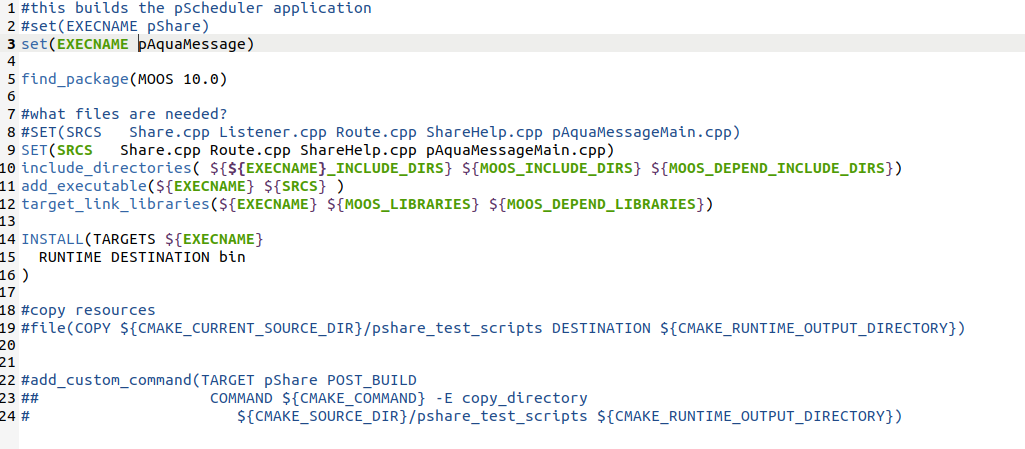
-Copy all the files in pShare into pAquaMessage

cp -rf /home/moos/moos-ivp/MOOS\_Dec3120/MOOSEssentials/Essentials/pShare/\* /home/moos/moos-ivp-extend/src/pAquaMessage/

-Modify the CMakeLists.txt in pAquaMessage. Changing the line2 and line7 as the following figure.

mv pShareMain.cpp pAquaMessageMain.cpp

vim CMakeLists.txt



-Replace the pShare with the new created pAquaShare on machine A and machine B

-For machine A side: adding “run = pAquaMessage @ NewConsole = true” in the ANTLER block, and add a new block for pAquaMessage as the following figure



-For machine B, the same procedure. The corresponding block should be added into alpha.moos file.



Until now, the new created compliable pAquaMessage application used for message exchange between two MOOSDB has been run successfully based on internet socket.

## 7.1 Single way communication configuration (AUV -> Offshore PC)

--Modify the Share.cpp in offshore computer side. The code is uploaded into github.

https://github.com/shuaidong-networking/AquaNet-MOOS/blob/main/Offshore-pAquaMessage/Share.cpp

--Modify the Share.cpp in AUV side. The code is uploaded into github

https://github.com/shuaidong-networking/AquaNet-MOOS/blob/main/AUV-pAquaMessage/Share.cpp

--Add alpha.bhv into pAquaMessage on AUV side. The code is shown on github

https://github.com/shuaidong-networking/AquaNet-MOOS/blob/main/AUV-pAquaMessage/alpha.bhv

--Copy the publisher.sh script into pAquaMessage file on offshore side computer side

--Copy the subscriber.sh script into pAquaMessage file on AUV side

**Running the single way communication---offshore computer**

--Running virtual modem server on the offshore computer side.

cd $AQUANET\_FOLDER/trunk

./bin/aquanet-vmds 2021

--Running publisher.sh

chmod +x publisher.sh

./publisher.sh

--Running the alpha.moos file

pAntler alpha.moos

**--Running the single way communication—AUV side**

--Running subscriber.sh

chmod +x subscriber.sh

./subscriber.sh

--Running the alpha.moos file

pAntler alpha.moos

After all the procedures are done, you will see the screen on offshore computer showing the trajectory of AUV

## 7.2 Two-way communication configuration (AUV <-> Offshore PC)

--Modify the Share.cpp in offshore computer side. The code is uploaded into github.

https://github.com/shuaidong-networking/AquaNet-MOOS/blob/main/Offshore-pAquaMessage/Share.cpp

--Modify the Share.cpp in AUV side. The code is uploaded into github

https://github.com/shuaidong-networking/AquaNet-MOOS/blob/main/AUV-pAquaMessage/Share.cpp

--Add alpha.bhv into pAquaMessage on AUV side. The code is shown on github

https://github.com/shuaidong-networking/AquaNet-MOOS/blob/main/AUV-pAquaMessage/alpha.bhv

--Copy the publisher.sh script into pAquaMessage file on offshore side computer side

--Copy the subscriber.sh script into pAquaMessage file on AUV side

**Running the two-way communication---offshore computer**

--Running virtual modem server on the offshore computer side.

cd $AQUANET\_FOLDER/trunk

./bin/aquanet-vmds 2021

--Running publisher.sh

chmod +x publisher.sh

./publisher.sh

--Running the alpha.moos file

pAntler alpha.moos

**--Running the two-way communication—AUV side**

--Running subscriber.sh

chmod +x subscriber.sh

./subscriber.sh

--Running the alpha.moos file

pAntler alpha.moos

After all the procedures are done, the communication between two machines can exchange the command message and GPS information through AquaNet.

# 8. UnetStack

UnetStack is an agent-based network stack. Each *agent*is like a layerin a traditional network stack but has more flexibility to use the scarce resources (bandwidth, energy, etc) in the unet more efficiently. In order to develop unet over the subnero modems, we need to configure all environmental settings.

## 8.1 Unet installation

--Downloading the unet from GitHub with the following link

https://github.com/org-arl/unet-contrib

--If developers use c code, under the c folder, build the code

cd unet-contrib/unetsocket/c/

make

--Building the sample code under c/samples folder

make samples

## 8.2 Create our own code under samples/ folder for application

--Go to the makefile under the c/ folder

https://github.com/org-arl/unet-contrib

--we create two codes in makefile (tx\_string and rx\_string to send and receive between two modems)

samples: txdata txdata-reliable rxdata range setpowerlevel txsignal pbrecord bbrecord npulses execcmd wakeup rs232\_wakeup tx\_string rx\_string

    rm -rf \*.zip

    rm -rf fjage-\*

--Assign library to the created code

tx\_string.o: samples/tx\_string.c unet.h fjage.h

    $(CC) $(CFLAGS) -c samples/tx\_string.c -o samples/tx\_string.o

tx\_string: tx\_string.o

    $(CC) -o samples/tx\_string samples/tx\_string.o libunet.a libfjage.a -lpthread -lm

rx\_string.o: samples/rx\_string.c unet.h fjage.h

    $(CC) $(CFLAGS) -c samples/rx\_string.c -o samples/rx\_string.o

rx\_string: rx\_string.o

    $(CC) -o samples/rx\_string samples/rx\_string.o libunet.a libfjage.a -lpthread -lm

--Then go back to c/samples folder and make our code compliable

cd unet-contrib/unetsocket/c/samples

make samples

## 8.3 Complete sending code on Machine A