MIT 2.680

UNMANNED MARINE VEHICLE AUTONOMY,
SENSING, AND COMMUNICATIONS

Lecture 6 – Multi-Vehicle Missions

February 26th, 2026

Web: http://oceanai.mit.edu/2.680

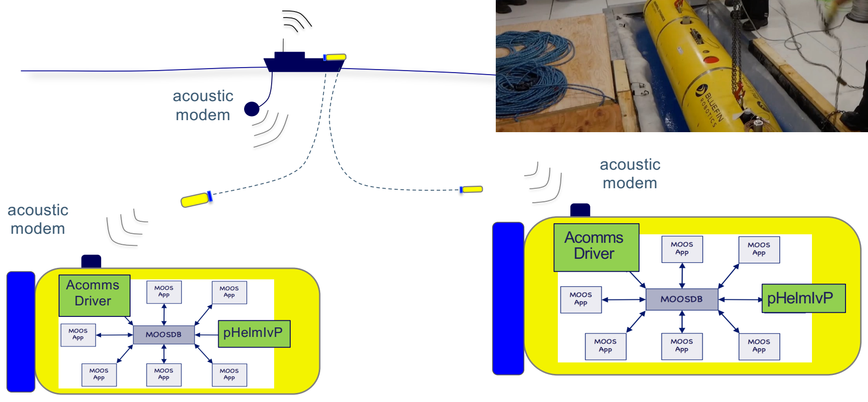
Email:
Mike Benjamin, mikerb@mit.edu

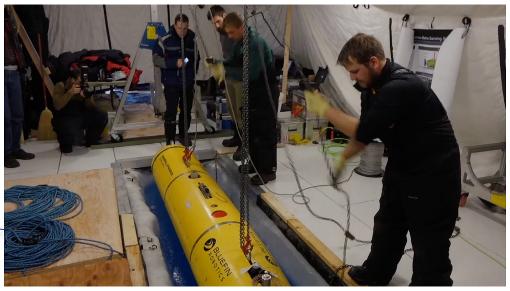
Photo by Arjan Vermeij
GLINT '09

1

Multi-Vehicle Operations

In this lecture:
- Inter-MOOSDB communications
- The uField Toolbox
- Launching Multi-Vehicle Missions





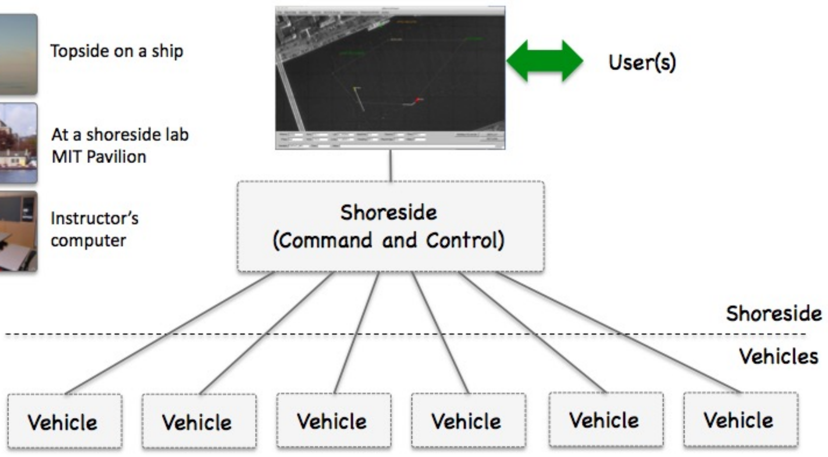
Inter DB Comms | pShare | uField Toolbox | Berta Mission | Launching Missions | Plug/Meta Files | Lab Preview

2

MITMECHE

The Shoreside/Vehicle Topology

"Shoreside" could be:



3

MITMECHE

Communications Between Machines / Vehicles

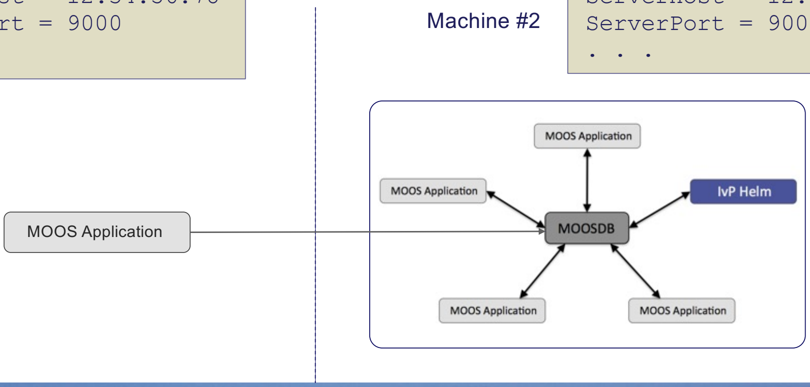
We've seen in our labs that MOOS apps do not necessarily have to be on the same physical machine running the MOOSDB.

Machine #1

```
ServerHost = 12.34.56.78
ServerPort = 9000
. . .
```

Machine #2

```
ServerHost = 12.34.56.78
ServerPort = 9000
. . .
```



4

Communications Between Machines / Vehicles

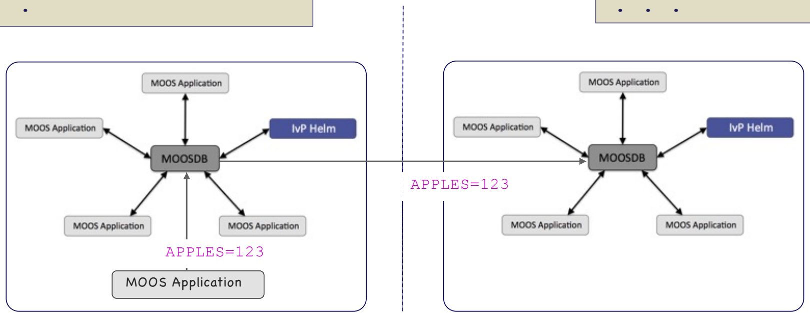
How do we get two MOOSDB's (communities) to talk to each other?

Machine #1

```
ServerHost = 12.34.56.78
ServerPort = 9000
. . .
```

Machine #2

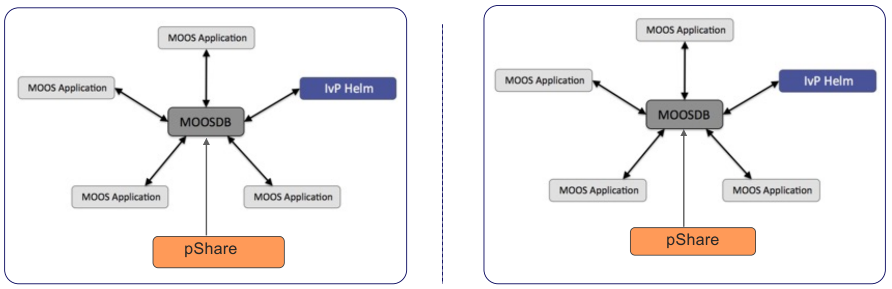
```
ServerHost = 12.34.56.78
ServerPort = 9000
. . .
```



When the two machines are on the same network, we can use pShare, (written by Paul Newman)

Inter MOOSDB Communications with pShare

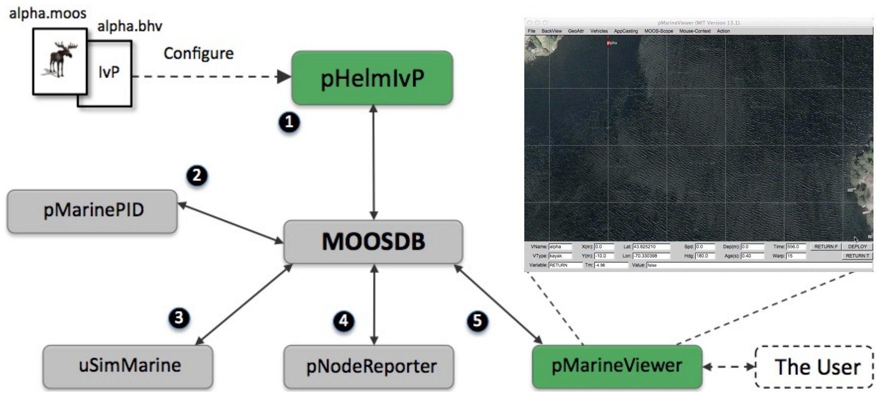
We use pShare for communications between two MOOS communities on the same network.



The pShare app is launched on *both* machines as part of their respective communities.

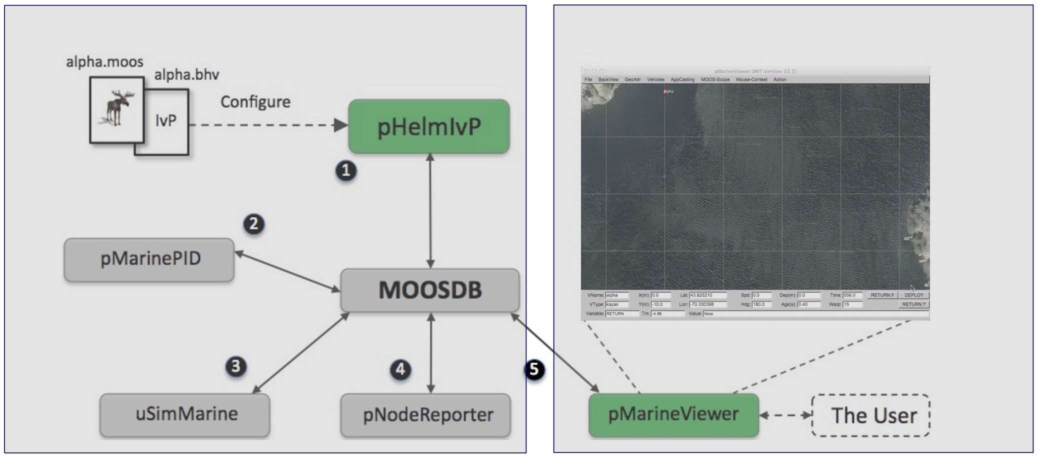
Revisiting the Alpha Mission

- In the next lab (Lab 6), one of the first exercises is “Alpha pShare”.
- We split the Alpha mission onto two machines: A shoreside machine and robot machine.

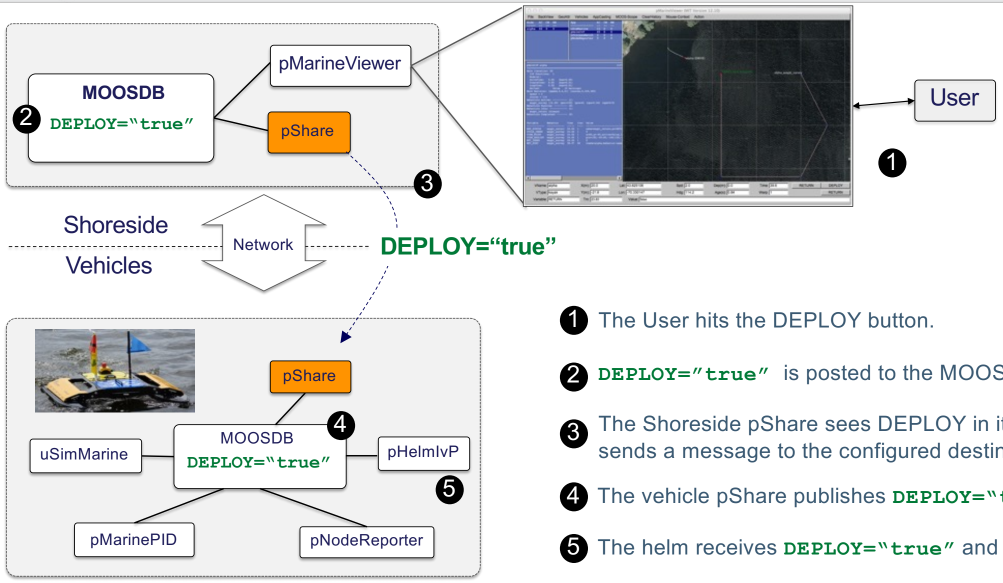


Revisiting the Alpha Mission

- Today’s lab, one of the first exercises is “Alpha pShare”.
- We split the Alpha mission onto two machines: A shoreside machine and robot machine.



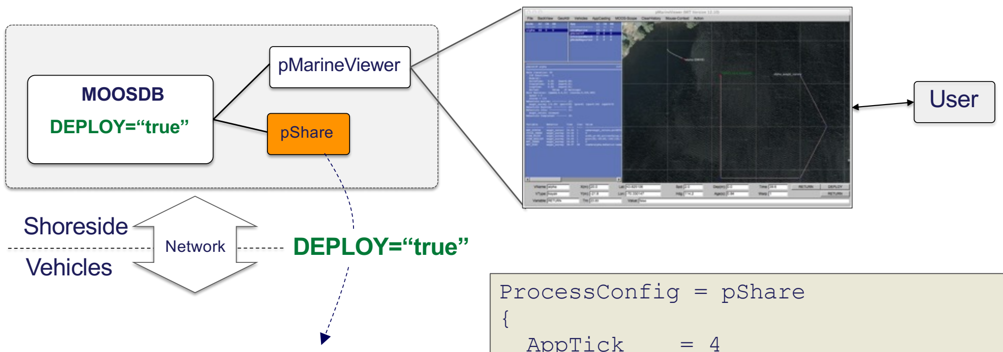
The uField Topology



1. The User hits the DEPLOY button.
2. `DEPLOY="true"` is posted to the MOOSDB by pMarineViewer
3. The Shoreside pShare sees DEPLOY in its in-box and sends a message to the configured destination
4. The vehicle pShare publishes `DEPLOY="true"` to the local MOOSDB
5. The helm receives `DEPLOY="true"` and starts the vehicle.

9

Configuring the *Sending* pShare



```
ProcessConfig = pShare
{
  AppTick    = 4
  CommsTick  = 4

  output = src_name=DEPLOY, route=192.1682.2.4:9202
}
```

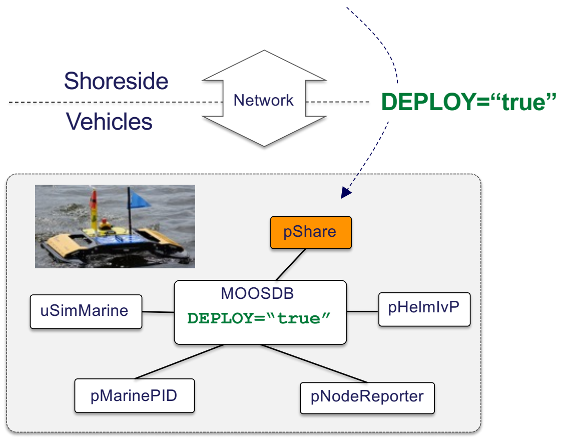
Name of variable to send

IP address of target machine

Port on target machine

10

Configuring the *Receiving* pShare



```
ProcessConfig = pShare
{
  AppTick    = 4
  CommsTick  = 4

  input = route=localhost:9202
}
```

The port we are listening on

Two-Way pShare

- pShare always names variables to send, and does not specify variables to receive,
- Communication is almost always flowing in *both* directions

ShoreSide

```
ProcessConfig = pShare
{
  AppTick    = 4
  CommsTick  = 4

  input = route=localhost:9200

  output = src_name=DEPLOY, route=192.168.1.2:9201
  output = src_name=RETURN, route=192.168.1.3:9201
}
```

Addresses of vehicles

- Command and control messages to vehicle
- E.g., `DEPLOY` and `RETURN`

Vehicle

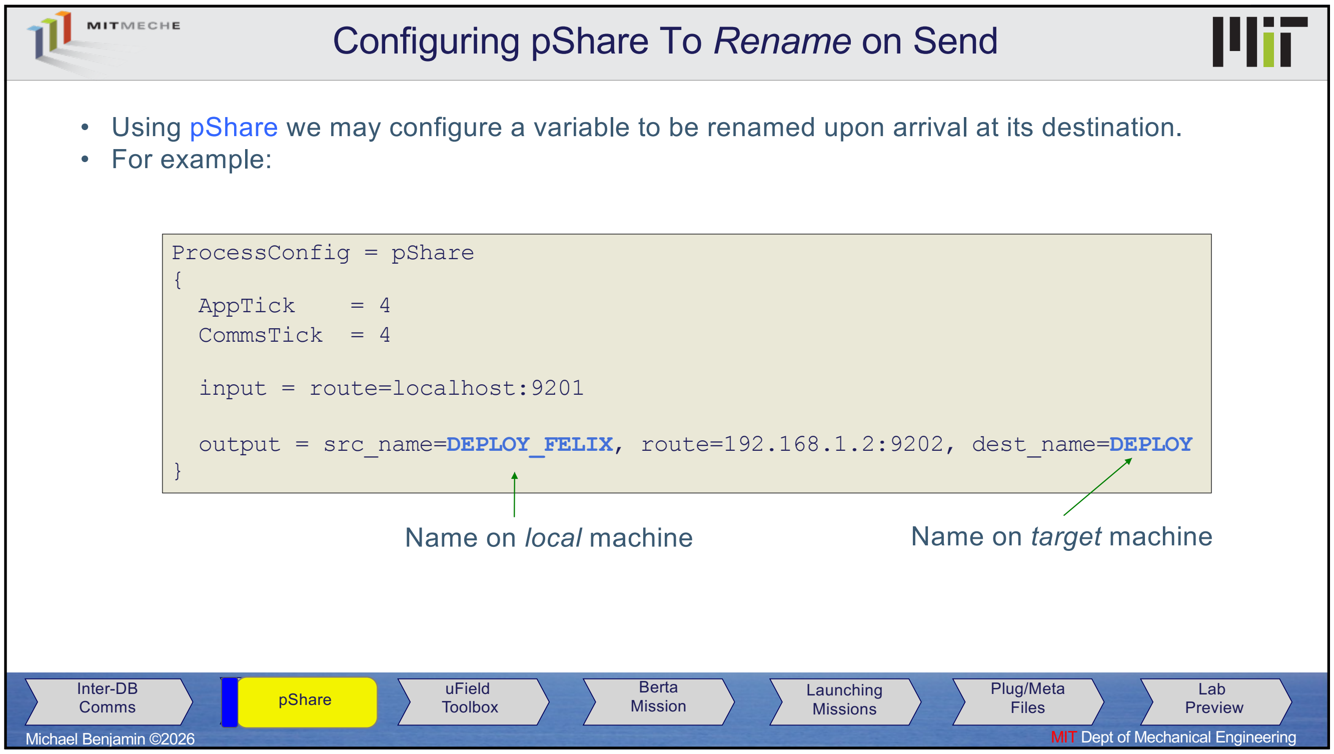
```
ProcessConfig = pShare
{
  AppTick    = 4
  CommsTick  = 4

  input = route=localhost:9201

  output = src_name=NODE_REPORT, route=192.168.1.1:9200
  output = src_name=VIEW_POINT,  route=192.168.1.1:9200
}
```

Address of the shoreside

- Status messages to the Shoreside
- E.g., `NODE_REPORT` and `VIEW_POINT`



Configuring pShare To *Rename* on Send

- Using pShare we may configure a variable to be renamed upon arrival at its destination.
- For example:

```
ProcessConfig = pShare
{
  AppTick    = 4
  CommsTick  = 4

  input = route=localhost:9201

  output = src_name=DEPLOY_FELIX, route=192.168.1.2:9202, dest_name=DEPLOY
}
```

Name on *local* machine

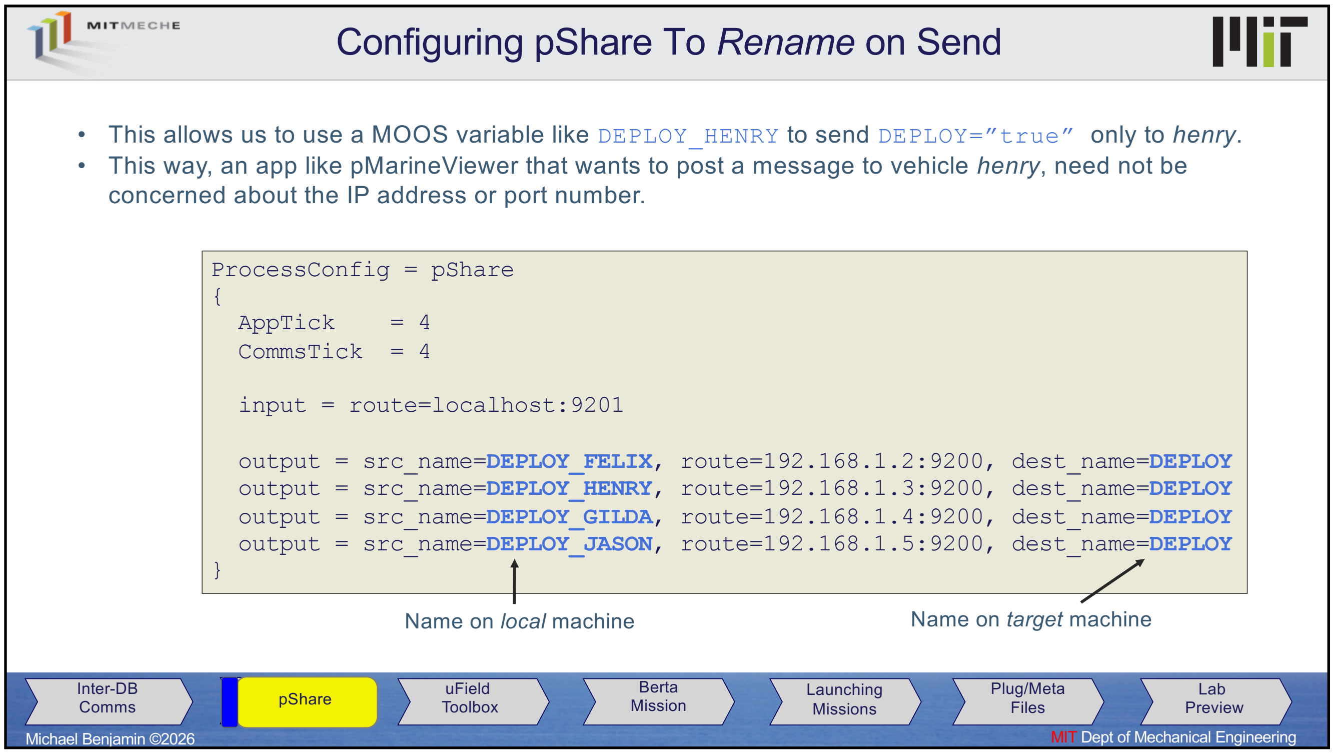
Name on *target* machine

Inter-DB Comms | pShare | uField Toolbox | Berta Mission | Launching Missions | Plug/Meta Files | Lab Preview

Michael Benjamin ©2026

MIT Dept of Mechanical Engineering

13



Configuring pShare To *Rename* on Send

- This allows us to use a MOOS variable like `DEPLOY_HENRY` to send `DEPLOY="true"` only to *henry*.
- This way, an app like pMarineViewer that wants to post a message to vehicle *henry*, need not be concerned about the IP address or port number.

```
ProcessConfig = pShare
{
  AppTick    = 4
  CommsTick  = 4

  input = route=localhost:9201

  output = src_name=DEPLOY_FELIX, route=192.168.1.2:9200, dest_name=DEPLOY
  output = src_name=DEPLOY_HENRY, route=192.168.1.3:9200, dest_name=DEPLOY
  output = src_name=DEPLOY_GILDA, route=192.168.1.4:9200, dest_name=DEPLOY
  output = src_name=DEPLOY_JASON, route=192.168.1.5:9200, dest_name=DEPLOY
}
```

Name on *local* machine

Name on *target* machine

Inter-DB Comms | pShare | uField Toolbox | Berta Mission | Launching Missions | Plug/Meta Files | Lab Preview

Michael Benjamin ©2026

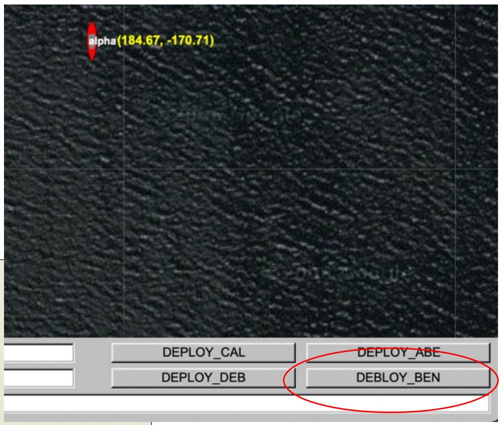
MIT Dept of Mechanical Engineering

14

Configuring pShare To *Rename* on Send

Example:

- The pMarineViewer GUI can be configured to have a dedicated **DEPLOY** button for four vehicles, **abe, ben, cal, deb**.
- Each button posts DEPLOY_VNAME.
- The DEPLOY_VNAME posting in the shoreside MOOSDB lands in the the vehicle MOOSDB as DEPLOY



```
ProcessConfig = pShare
{
  AppTick    = 4
  CommsTick  = 4

  input = route=localhost:9201

  output = src_name=DEPLOY_ABE, route=192.168.1.2:9200, dest_name=DEPLOY
  output = src_name=DEPLOY_BEN, route=192.168.1.3:9200, dest_name=DEPLOY
  output = src_name=DEPLOY_CAL, route=192.168.1.4:9200, dest_name=DEPLOY
  output = src_name=DEPLOY_DEB, route=192.168.1.5:9200, dest_name=DEPLOY
}
```

`DEPLOY_BEN = true`

Configuring pShare To *Broadcast*

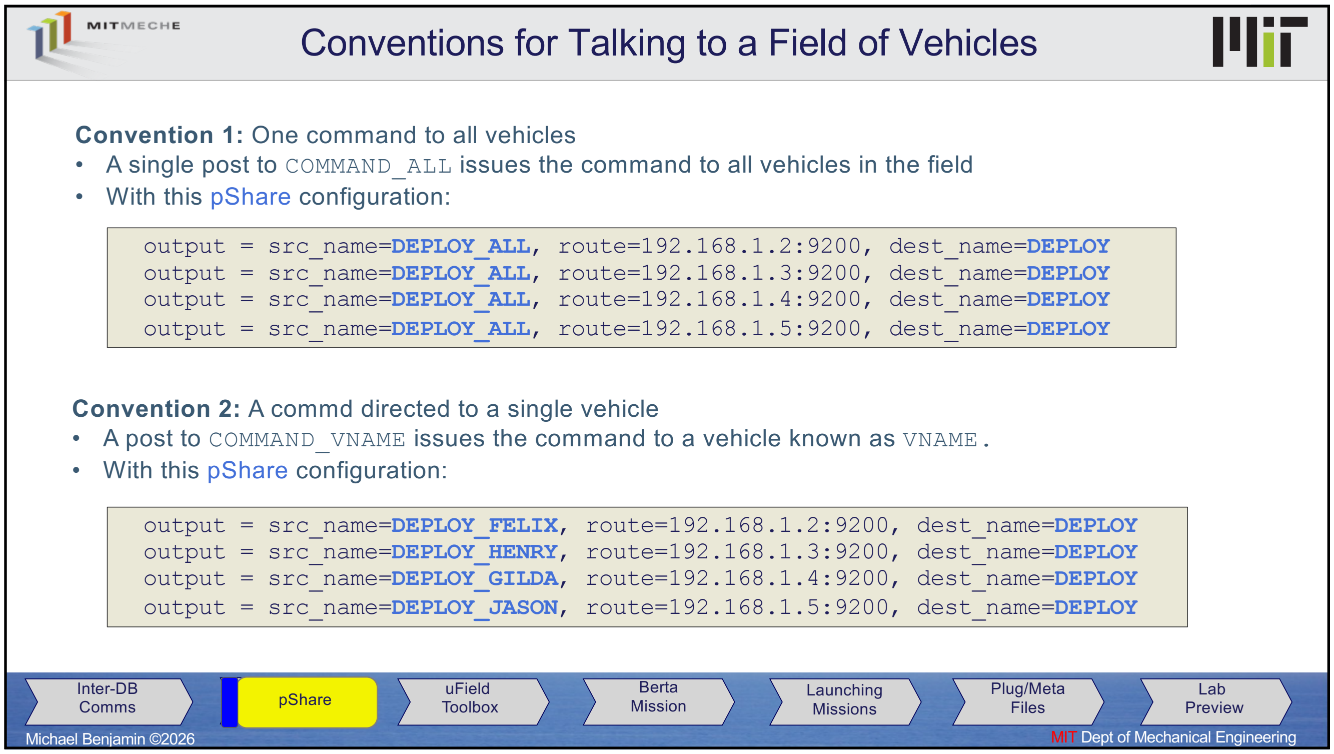
- The renaming feature can also be used to essentially broadcast a message.
- A single post can be shared out to many vehicles.

```
ProcessConfig = pShare
{
  AppTick    = 4
  CommsTick  = 4

  input = route=localhost:9201

  output = src_name=DEPLOY_ALL, route=192.168.1.2:9200, dest_name=DEPLOY
  output = src_name=DEPLOY_ALL, route=192.168.1.3:9200, dest_name=DEPLOY
  output = src_name=DEPLOY_ALL, route=192.168.1.4:9200, dest_name=DEPLOY
  output = src_name=DEPLOY_ALL, route=192.168.1.5:9200, dest_name=DEPLOY
}
```

Name on *local* machine   Name on *target* machine



MITMECHE

Conventions for Talking to a Field of Vehicles

Convention 1: One command to all vehicles

- A single post to `COMMAND_ALL` issues the command to all vehicles in the field
- With this pShare configuration:

```
output = src_name=DEPLOY_ALL, route=192.168.1.2:9200, dest_name=DEPLOY
output = src_name=DEPLOY_ALL, route=192.168.1.3:9200, dest_name=DEPLOY
output = src_name=DEPLOY_ALL, route=192.168.1.4:9200, dest_name=DEPLOY
output = src_name=DEPLOY_ALL, route=192.168.1.5:9200, dest_name=DEPLOY
```

Convention 2: A commd directed to a single vehicle

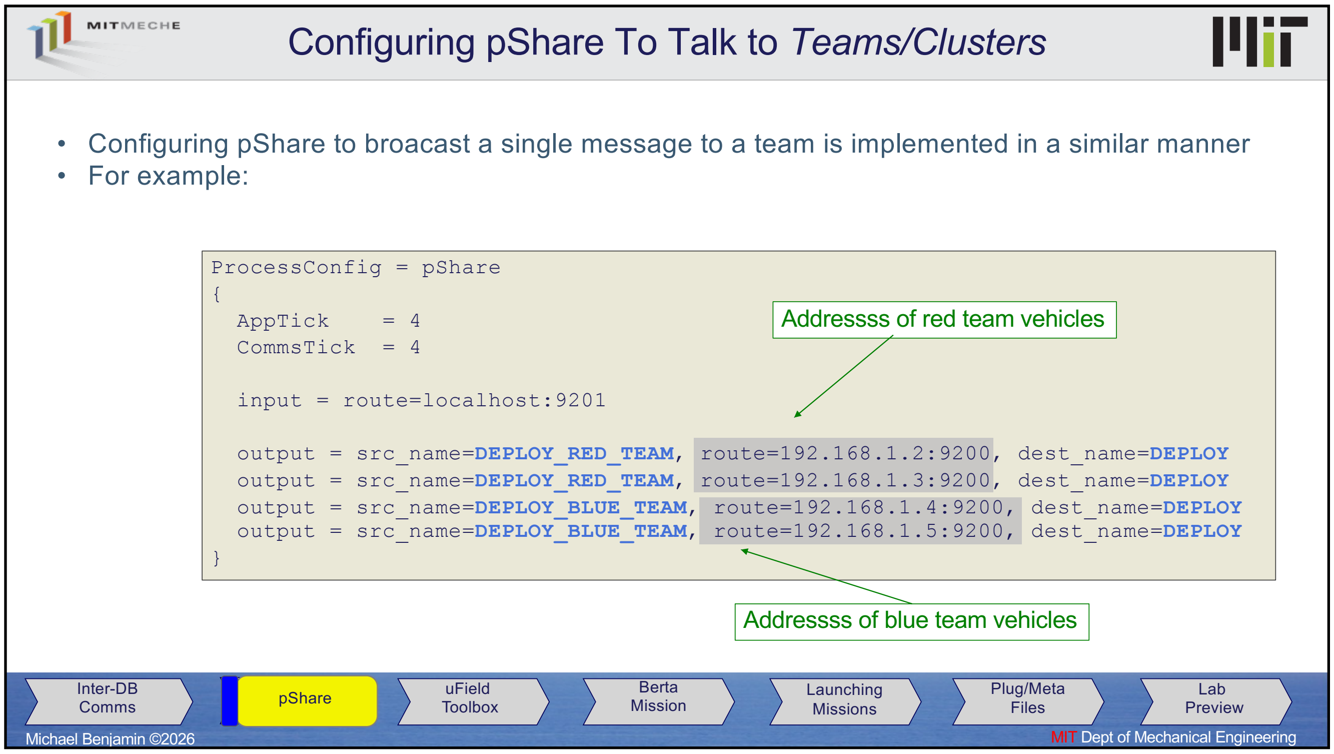
- A post to `COMMAND_VNAME` issues the command to a vehicle known as `VNAME`.
- With this pShare configuration:

```
output = src_name=DEPLOY_FELIX, route=192.168.1.2:9200, dest_name=DEPLOY
output = src_name=DEPLOY_HENRY, route=192.168.1.3:9200, dest_name=DEPLOY
output = src_name=DEPLOY_GILDA, route=192.168.1.4:9200, dest_name=DEPLOY
output = src_name=DEPLOY_JASON, route=192.168.1.5:9200, dest_name=DEPLOY
```

Inter-DB Comms | pShare | uField Toolbox | Berta Mission | Launching Missions | Plug/Meta Files | Lab Preview

Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

17



MITMECHE

Configuring pShare To Talk to *Teams/Clusters*

- Configuring pShare to broacast a single message to a team is implemented in a similar manner
- For example:

```
ProcessConfig = pShare
{
  AppTick    = 4
  CommsTick  = 4

  input = route=localhost:9201

  output = src_name=DEPLOY_RED_TEAM, route=192.168.1.2:9200, dest_name=DEPLOY
  output = src_name=DEPLOY_RED_TEAM, route=192.168.1.3:9200, dest_name=DEPLOY
  output = src_name=DEPLOY_BLUE_TEAM, route=192.168.1.4:9200, dest_name=DEPLOY
  output = src_name=DEPLOY_BLUE_TEAM, route=192.168.1.5:9200, dest_name=DEPLOY
}
```

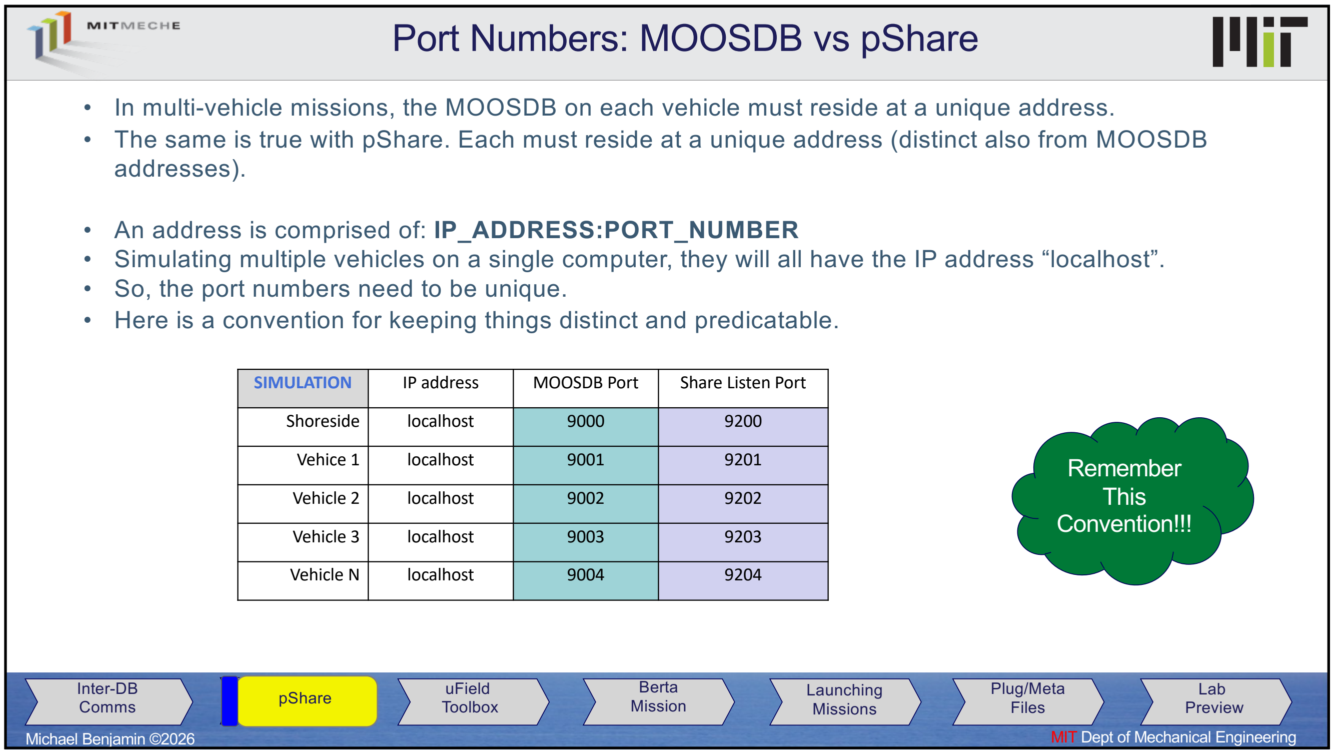
Addressss of red team vehicles

Addressss of blue team vehicles

Inter-DB Comms | pShare | uField Toolbox | Berta Mission | Launching Missions | Plug/Meta Files | Lab Preview

Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

18



Port Numbers: MOOSDB vs pShare

- In multi-vehicle missions, the MOOSDB on each vehicle must reside at a unique address.
- The same is true with pShare. Each must reside at a unique address (distinct also from MOOSDB addresses).

- An address is comprised of: **IP_ADDRESS:PORT_NUMBER**
- Simulating multiple vehicles on a single computer, they will all have the IP address "localhost".
- So, the port numbers need to be unique.
- Here is a convention for keeping things distinct and predicatable.

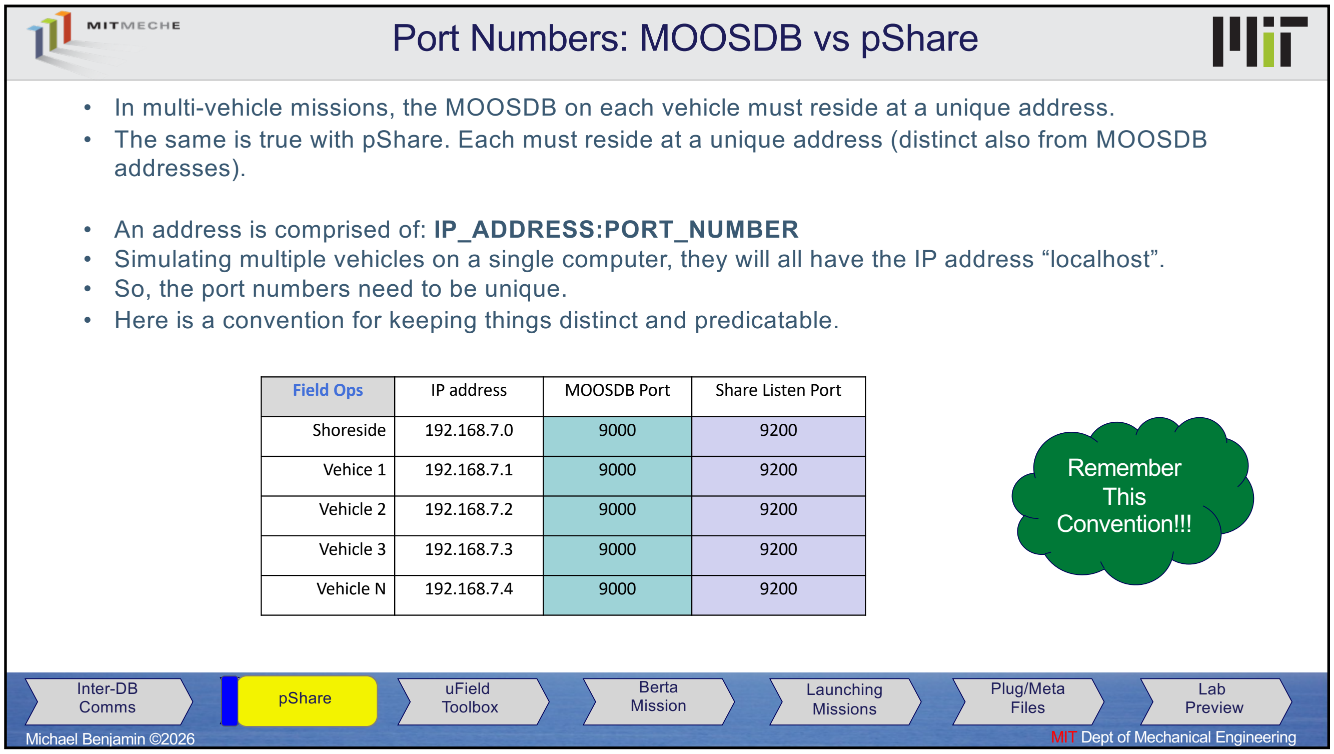
SIMULATION	IP address	MOOSDB Port	Share Listen Port
Shoreside	localhost	9000	9200
Vehice 1	localhost	9001	9201
Vehicle 2	localhost	9002	9202
Vehicle 3	localhost	9003	9203
Vehicle N	localhost	9004	9204

Remember This Convention!!!

Inter-DB Comms | pShare | uField Toolbox | Berta Mission | Launching Missions | Plug/Meta Files | Lab Preview

Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

19



Port Numbers: MOOSDB vs pShare

- In multi-vehicle missions, the MOOSDB on each vehicle must reside at a unique address.
- The same is true with pShare. Each must reside at a unique address (distinct also from MOOSDB addresses).

- An address is comprised of: **IP_ADDRESS:PORT_NUMBER**
- Simulating multiple vehicles on a single computer, they will all have the IP address "localhost".
- So, the port numbers need to be unique.
- Here is a convention for keeping things distinct and predicatable.

Field Ops	IP address	MOOSDB Port	Share Listen Port
Shoreside	192.168.7.0	9000	9200
Vehice 1	192.168.7.1	9000	9200
Vehicle 2	192.168.7.2	9000	9200
Vehicle 3	192.168.7.3	9000	9200
Vehicle N	192.168.7.4	9000	9200

Remember This Convention!!!

Inter-DB Comms | pShare | uField Toolbox | Berta Mission | Launching Missions | Plug/Meta Files | Lab Preview

Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

20

MITMECHE

Dynamic Configuration of pShare

- Sometimes the choice of vehicles, their IP addresses may change just prior to deployment.
- Sometimes even the Shoreside computer (and thus its IP address) may change just prior to deployment.

- We want some of the configuration to be *automatic*, otherwise things don't scale well.

21

MITMECHE

Dynamic Configuration of pShare

- While pShare is a rather special MOOS app (communication over multiple MOOS communities),
- It is still a MOOS app that reads mail and takes action.

pShare may launch with this initial configuration:

```
ProcessConfig = pShare
{
  input = route=localhost:9200

  output = src_name=DEPLOY_FELIX, route=192.168.1.2:9200, dest_name=DEPLOY
  output = src_name=DEPLOY_HENRY, route=192.168.1.3:9200, dest_name=DEPLOY
  output = src_name=DEPLOY_GILDA, route=192.168.1.4:9200, dest_name=DEPLOY
  output = src_name=DEPLOY_JASON, route=192.168.1.5:9200, dest_name=DEPLOY
}
```

Upon receipt of incoming MOOS mail:

```
PSHARE_CMD="cmd=output,src_name=DEPLOY_KEVIN,dest_name=DEPLOY,route=192.168.1.6:9200"
```

pShare will augment its configuration. As if the below had been part of the original configuration:

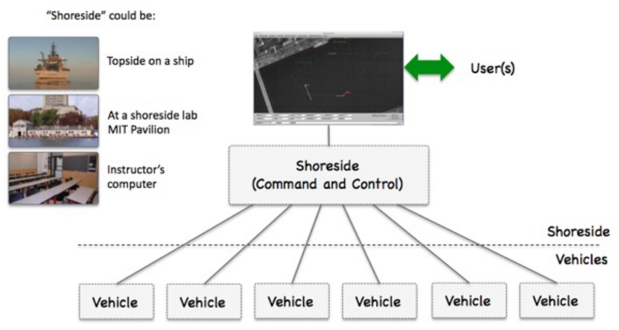
```
output = src_name=DEPLOY_KEVIN, route=192.168.1.6:9200, dest_name=DEPLOY
```

22

The uField Toolbox Overview

The uField Toolbox is:

- A collection of about a dozen MOOS applications, each a Utility for Fielding multiple vehicles with a shoreside/topside command-and-control MOOS Community.



The uField Toolbox is comprised of three general capabilities:

1. Facilitation of Inter MOOSDB Share configuration
2. Simulation of Inter-Vehicle Messaging
3. Sensor Simulation

- All applications are documented in the MOOS-IvP Tools document, online.
  http://oceanai.mit.edu/ivpman/ufield

Inter-DB Comms | pShare | uField Toolbox | Berta Mission | Launching Missions | Plug/Meta Files | Lab Preview

The uField Toolbox

The *uField Toolbox* is comprised of three general capabilities:

1. Facilitation of Inter MOOSDB Share configuration
   - pHostInfo
   - uFldNodeBroker
   - uFldShoreBroker

   Lab 6: Multi-Vehicle Operations

   Lab 7: Distributed Traveling Salesman

2. Simulation of Inter-Vehicle Messaging
   - uFldNodeComms
   - uFldMessageHandler

   Lab 8: Constrained Inter-vehicle Messaging

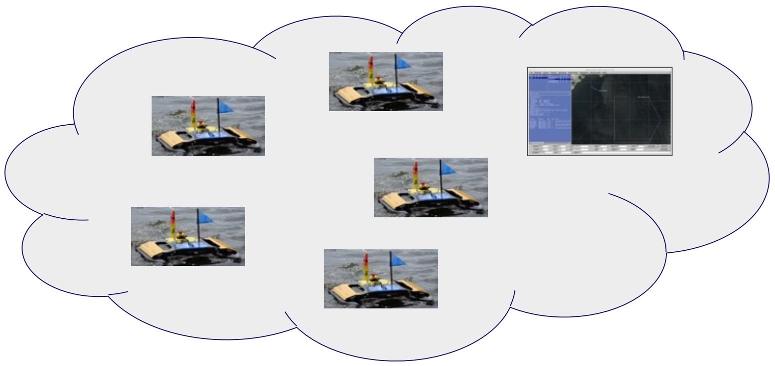
3. Sensor Simulation
   - uFldSwimSensor
   - uFldManOverboard

   Lab 9: Man Overboard Lab

Inter-DB Comms | pShare | uField Toolbox | Berta Mission | Launching Missions | Plug/Meta Files | Lab Preview

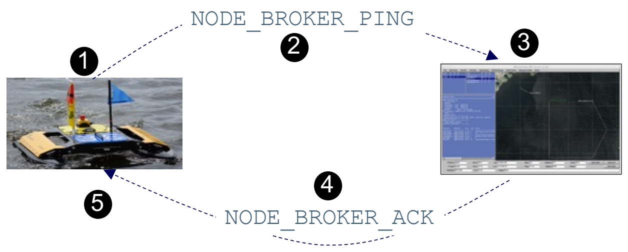
Facilitation of pShare Configuration

Starting Conditions:
- All vehicles are on the Internet, perhaps only a local network
- Shoreside has no idea what vehicles are out there and where
- Vehicles have some idea of the Shoreside IP address (a short list of places to try)



The Configuration Sequence

- The vehicles use their short list of possible locations for the Shoreside IP address
- They initiate comms to the the shoreside, in the form a "ping". A test-balloon of sorts.



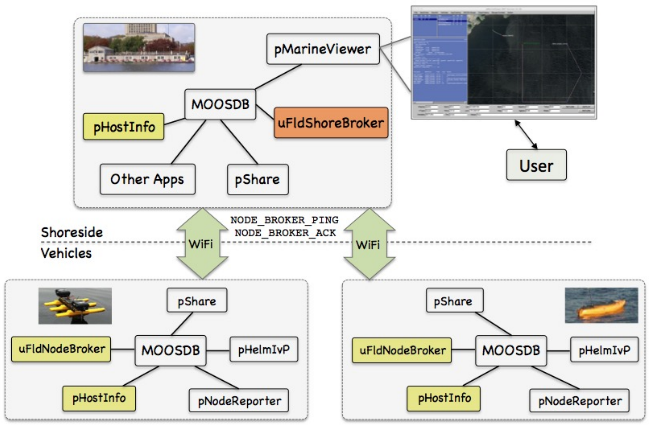
1. Robot auto-discovers its own IP address
2. Robot pings the shoreside, announcing the robot's name, IP address and port number it is listening on
3. Shoreside updates its pShare configuration for the new robot
4. Shoreside sends an acknowlegement to the robot, confirming to the robot the receipt of the robot ping
5. Robot receives acknowledgement from Shoreside and updates its pShare configuration to allow comms to the Shoreside

The uField Toolbox
Three Tools for Auto pShare Configuration

pHostInfo — A MOOS app for automatically determining the local machines IP address, and publishing it to the MOOSDB

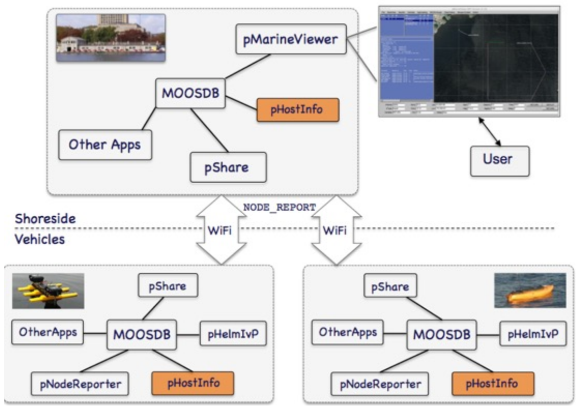
uFldNodeBroker — A MOOS app for
- finding a shoreside,
- determining it's IP address and pShare input route,
- Auto-configuring its own local pShare outgoing route

uFldShoreBroker — A MOOS app for
- Listening for incoming nodes
- Notifying the nodes of the shoreside IP address and pShare input route,
- Auto-configuring its own local pShare outgoing route



The pHostInfo Utility

Upon launch, pHostInfo will:
- Determine the IP address of the machine.
- Publish the result in PHI_HOST_IP
- It will use whatever OS utility is available to discover its own IP address.
- Sometimes a robot may have serveral network interfaces.
- Sometimes pHostInfo will guess wrong.
- This information is used by uFldNodeBroker to initiate a connection to the Shoreside.



pHostInfo will publish to the local MOOSDB:

```
        PHI_HOST_IP = 118.10.24.23
    PHI_HOST_IP_ALL = 118.10.24.23,169.224.126.40
   PHI_HOST_PORT_DB = 9000
PHI_HOST_IP_VERBOSE = OSX_ETHERNET2=118.10.24.23,OSX_AIRPORT=169.224.126.40
```

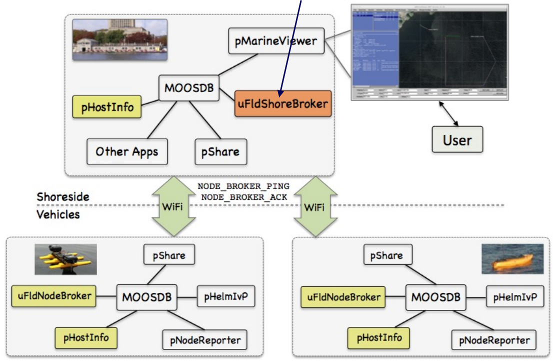
The uField Toolbox - uFldShoreBroker

Automating Shoreside/Vehicle Connections

Runs in the shoreside community

Upon launch, uFldShoreBroker will:

- Wait for pHostInfo to post our own IP address
- Listen for `NODE_BROKER_PING` messages coming from vehicles in the field.
- Process each ping sending a message back to the robot in the form of `NODE_BROKER_ACK`, acknowledging the Shoreside IP address.
- Adjust its own pShare configuration accordingly.



- Posts (received on the Shoreside)

```
NODE_BROKER_PING = community=henry,hostip=192.168.1.1,port_db=9000,
                   pshare_iroutes=192.168.1.1:9200,timewarp=8
```

- Replies sent from shoreside with information about the shoreside community.

```
NODE_BROKER_ACK = community=shoreside,hostip=192.168.1.199,port_db=9000,
                  pshare_iroutes=192.168.1.199:9300,timewarp=8,status=ok
```

- Augments the local pShare configuration

```
PSHARE_CMD = src_name=DEPLOY_HENRY,dest_name=DEPLOY,route=192.68.1.199:9300
```

Inter-DB Comms | pShare | uField Toolbox | Berta Mission | Launching Missions | Plug/Meta Files | Lab Preview

Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

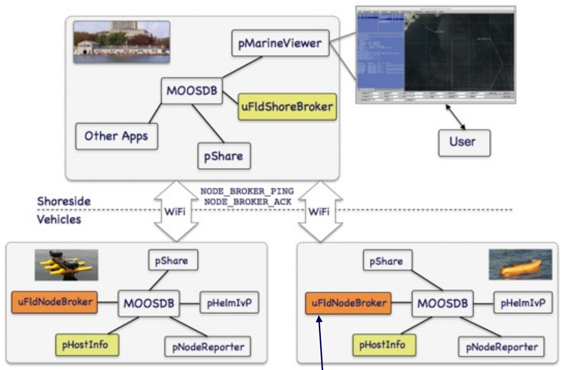
29

The uField Toolbox – uFldNodeBroker

Automating Shoreside/Vehicle Connections

Upon launch, uFldNodeBroker will:

- Wait for pHostInfo to post our own IP address
- Publish a `NODE_BROKER_PING` to the Shoreside, guessing the Shoreside IP from a list.
- Then wait for an acknowledgement from the Shoreside in the form of `NODE_BROKER_ACK`
- Adjust its own pShare configuration accordingly.



Runs on the vehicle community

(one on each vehicle)

- Posts (to be sent to Shoreside)

```
NODE_BROKER_PING = community=henry,hostip=192.168.1.1,port_db=9000,
                   pshare_iroutes=192.168.1.1:9200,timewarp=8
```

- Receives reply from shoreside with information about the shoreside community.

```
NODE_BROKER_ACK = community=shoreside,hostip=192.168.1.199,port_db=9000,
                  pshare_iroutes=192.168.1.199:9300,timewarp=8,status=ok
```

- Augments the local pShare configuration

```
PSHARE_CMD = src_name=NODE_REPORT_LOCAL,dest_name=NODE_REPORT,route=192.68.1.199:9300
```

Inter-DB Comms | pShare | uField Toolbox | Berta Mission | Launching Missions | Plug/Meta Files | Lab Preview

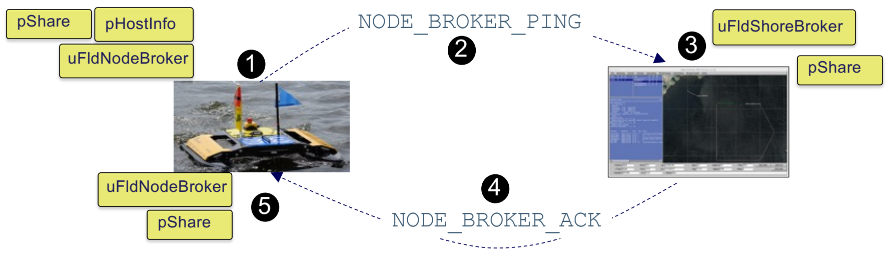
Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

30

MITMECHE

The Configuration Sequence

- The vehicles use their short list of possible locations for the Shoreside IP address
- They initiate comms to the the shoreside, in the form a "ping". A test-balloon of sorts.



1. Robot auto-discovers its own IP address (pHostInfo), modifies outgoing connections (pShare) to send a message (uFldNodeBroker) to the Shoreside
2. Robot pings the shoreside, announcing the robot's name, IP address and port number it is listening on
3. Shoreside processes message from robot (uFldShoreBroker) and updates its comms configuration (pShare) for the new robot
4. Shoreside sends an acknowlegement to the robot, confirming to the robot the receipt of the robot ping
5. Robot receives acknowledgement from Shoreside and updates its configuration (pShare) to allow commes to the Shoreside

MITMECHE

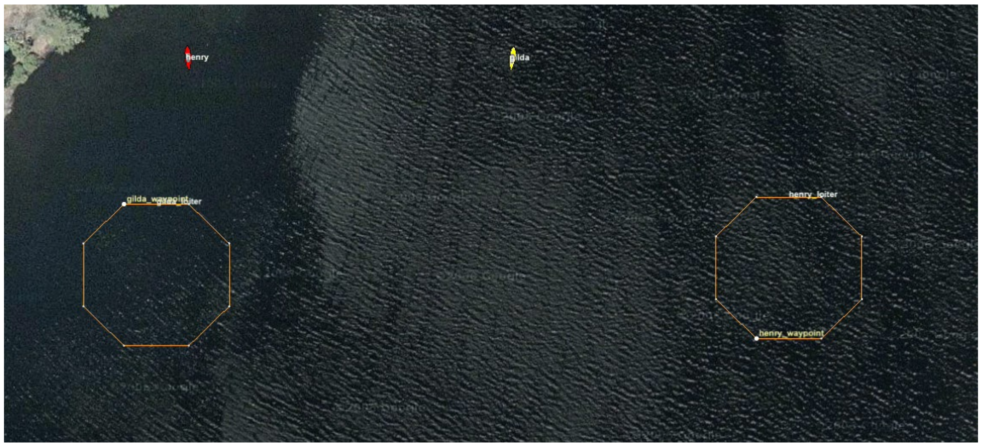
The Berta Mission

MITMECHE

The Berta Mission

The m2_berta mission is the “alpha mission” for multiple vehicles

```
$ cd ivp/missions/m2_berta
$ ./launch.sh 10
```



Inter-DB Comms | pShare | uField Toolbox | Berta Mission | Launching Missions | Plug/Meta Files | Lab Preview

Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

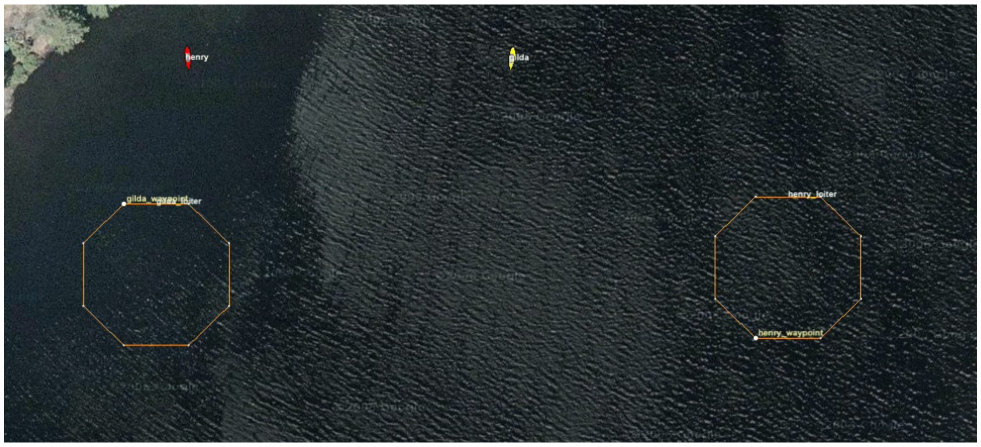
33

MITMECHE

The Berta Mission

The m2_berta mission involves:

- one shoreside MOOS community
- one MOOS community for each vehicle



Inter-DB Comms | pShare | uField Toolbox | Berta Mission | Launching Missions | Plug/Meta Files | Lab Preview

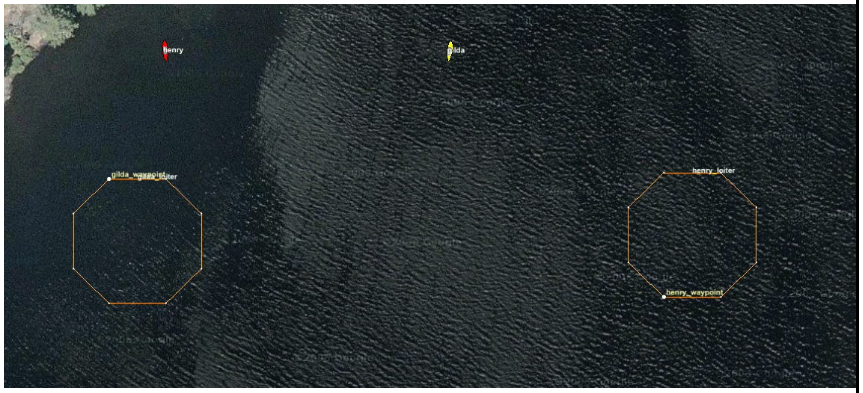
Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

34

The Berta Mission Behaviors

Each vehicle has 4 behaviors:

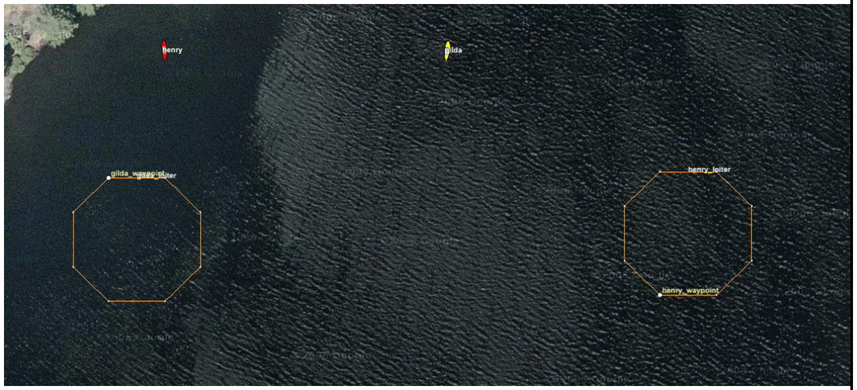
- a Loiter Behavior
- a Collision Avoidance Behavior
- a StationKeep Behavior
- a return-home Waypoint Behavior



The Berta Mission East-West Transitions

The Shoreside MOOS Community:

- Runs uTimerScript to periodically switch the East/West Regions
- The Region assignments are sent out to each vehicle.

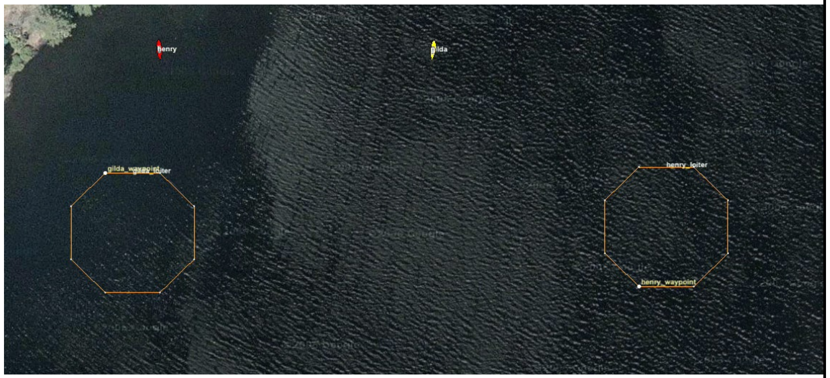


MITMECHE

The Berta Mission Node Reports

Each Vehicle is generating **NODE_REPORT** messages

- Node reports are sent to the shoreside to pMarineViewer can render the vehicles
- Node reports are also shared out to all other vehicles, so the CollisionAvoidance behavior has the info it needs



Inter-DB Comms | pShare | uField Toolbox | Berta Mission | Launching Missions | Plug/Meta Files | Lab Preview

Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

37

MITMECHE

Launching Missions

Inter-DB Comms | pShare | uField Toolbox | Berta Mission | Launching Missions | Plug/Meta Files | Lab Preview

Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

38

Launching Missions

- Launching a multi-vehicle mission involves launching several MOOS communities
- It could be done one at a time from the Terminal:

```
$ cd ivp/missions/m2_berta
$ pAntler shoreside.moos --MOOSTimeWarp 10
$ pAntler henry.moos --MOOSTimeWarp 10
$ pAntler gilda.moos --MOOSTimeWarp 10
```

- A preferable method is to launch with a single launch script

```
$ cd ivp/missions/m2_berta
$ ./launch.sh 10
```

Bash Scripts

- Our launch scripts are Bash Scripts
- Bash scripting is its own programming language
- Bash scripting has been around since the early days of Unix
- Bash scripts typically end in the suffix "`.sh`"
- They can take arguments on the command line:

```
$ ./launch.sh 10
```

- The best way to learn about Bash scripting is to Google the topic – there is a ton of learning material online
- Bash scripting can be enormously useful for many other tasks outside MOOS-IvP
- In our lab, knowledge of C++ and Bash opens almost every door

Inside Our Bash Scripts

- Our launch scripts automate many of the tedious configuration tasks
- They all typically will interpret a single numerical argument as the TimeWarp
- For example, the following launches a mission with TimeWarp 10

```
$ ./launch.sh 10
```

- At some point in the above script, pAntler will be invoked for each MOOS community:

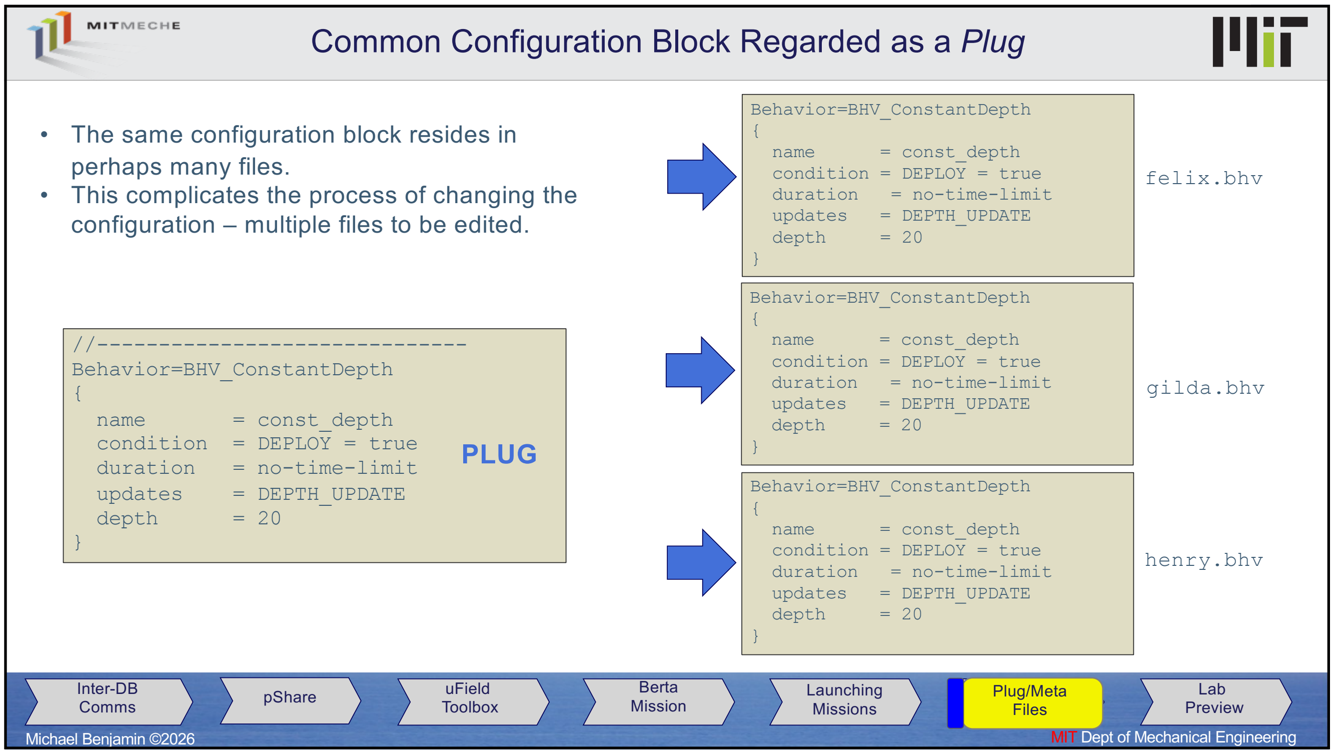
```
pAntler targ_shoreside.moos
pAntler targ_henry.moos
pAntler targ_gild.moos
```

Maintaining Multiple Mission Files

- With multiple vehicles, come multiple configuration files
- Between vehicles they are very similar, perhaps 98% the same
- A configuration edit on one vehicle requires edits on all other vehicle files
- This process is error-prone and tedious

- Consider a mission where each vehicle is configured with a ConstantDepth behavior

```
//-----------------------------
Behavior=BHV_ConstantDepth
{
  name         = const_depth
  condition    = DEPLOY = true
  duration     = no-time-limit
  updates      = DEPTH_UPDATE
  depth        = 20
}
```



Common Configuration Block Regarded as a *Plug*

- The same configuration block resides in perhaps many files.
- This complicates the process of changing the configuration – multiple files to be edited.

```
//--------------------------------
Behavior=BHV_ConstantDepth
{
  name        = const_depth
  condition   = DEPLOY = true
  duration    = no-time-limit
  updates     = DEPTH_UPDATE
  depth       = 20
}
```

PLUG

felix.bhv

```
Behavior=BHV_ConstantDepth
{
  name       = const_depth
  condition = DEPLOY = true
  duration    = no-time-limit
  updates    = DEPTH_UPDATE
  depth      = 20
}
```

gilda.bhv

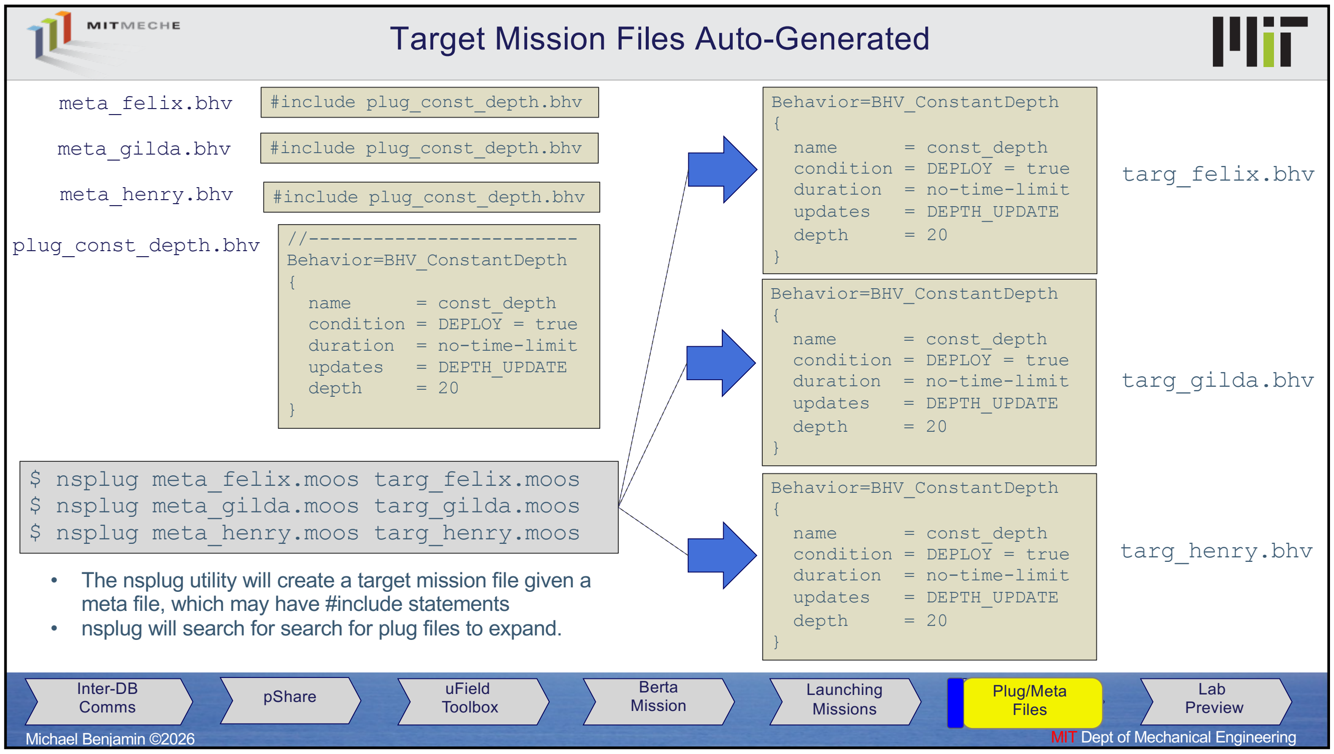
```
Behavior=BHV_ConstantDepth
{
  name       = const_depth
  condition = DEPLOY = true
  duration    = no-time-limit
  updates    = DEPTH_UPDATE
  depth      = 20
}
```

henry.bhv

```
Behavior=BHV_ConstantDepth
{
  name       = const_depth
  condition = DEPLOY = true
  duration    = no-time-limit
  updates    = DEPTH_UPDATE
  depth      = 20
}
```

Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

43



Target Mission Files Auto-Generated

meta_felix.bhv

```
#include plug_const_depth.bhv
```

meta_gilda.bhv

```
#include plug_const_depth.bhv
```

meta_henry.bhv

```
#include plug_const_depth.bhv
```

plug_const_depth.bhv

```
//--------------------------
Behavior=BHV_ConstantDepth
{
  name      = const_depth
  condition = DEPLOY = true
  duration  = no-time-limit
  updates   = DEPTH_UPDATE
  depth     = 20
}
```

```
$ nsplug meta_felix.moos targ_felix.moos
$ nsplug meta_gilda.moos targ_gilda.moos
$ nsplug meta_henry.moos targ_henry.moos
```

- The nsplug utility will create a target mission file given a meta file, which may have #include statements
- nsplug will search for search for plug files to expand.

targ_felix.bhv

```
Behavior=BHV_ConstantDepth
{
  name      = const_depth
  condition = DEPLOY = true
  duration  = no-time-limit
  updates   = DEPTH_UPDATE
  depth     = 20
}
```

targ_gilda.bhv

```
Behavior=BHV_ConstantDepth
{
  name      = const_depth
  condition = DEPLOY = true
  duration  = no-time-limit
  updates   = DEPTH_UPDATE
  depth     = 20
}
```

targ_henry.bhv

```
Behavior=BHV_ConstantDepth
{
  name      = const_depth
  condition = DEPLOY = true
  duration  = no-time-limit
  updates   = DEPTH_UPDATE
  depth     = 20
}
```

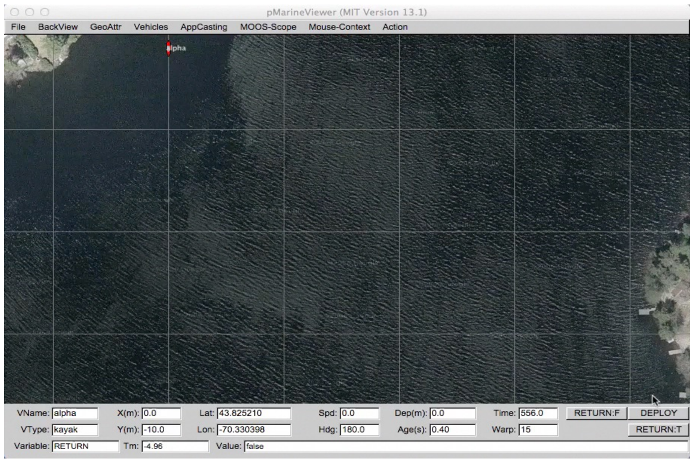
Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

44

Lab Preview

The Alpha pShare Mission

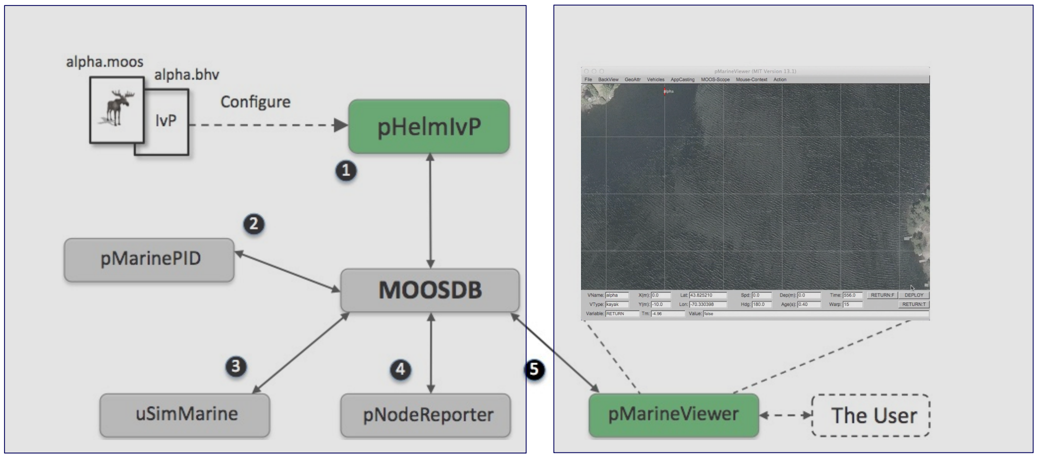
- Reconfigure the Alpha Mission to have a Shoreside and Vehicle community. Two *separate* MOOS communities
- It will look very familiar, but will be different "under the hood"



VName: alpha X(m): 0.0 Lat: 43.825210 Spd: 0.0 Dep(m): 0.0 Time: 556.0
VType: kayak Y(m): -10.0 Lon: -70.330398 Hdg: 180.0 Age(s): 0.40 Warp: 15
Variable: RETURN Tm: -4.96 Value: false

Revisiting the Alpha Mission

- Today's lab, one of the first exercises is "Alpha pShare".
- We split the Alpha mission onto two machines: A shoreside machine and robot machine.



END