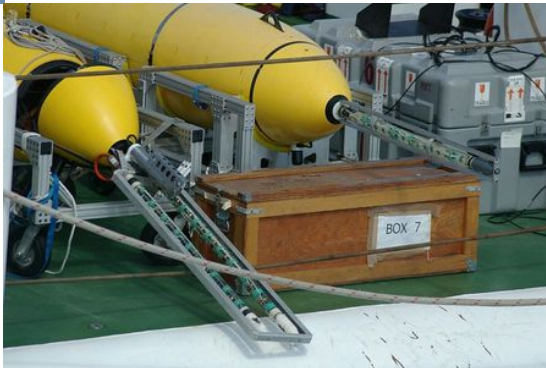




New Features and Applications in MOOS-IvP 13.5



Michael R. Benjamin
MIT Dept. of Mechanical Engineering
Computer Science and Artificial Intelligence Lab
Laboratory for Autonomous Marine Sensing Systems

mikerb@mit.edu



Acknowledgements

- **Office of Naval Research (ONR 311)** has sponsored MOOS-IvP for nearly 10 years. Dr. Don Wagner, Dr. Behzad Kamgar-Parsi, Dr. Wen Masters.
- **Battelle** has sponsored, beginning in 2010, MOOS-IvP, MOOS, the development of our autonomy course at MIT, the purchase of the Kingfisher platforms, and the build-out of our new Charles River facilities, and the development of our regression testing code. Dr. Rob Carnes

Faculty collaborators:



Prof. Henrik Schmidt
MIT MechE



Prof. John Leonard
MIT MechE/CSAIL



Prof. Paul Newman
Oxford

My group:



Alon Yaari
MIT Research Scientist



LT Arthur Anderson
MIT Navy PhD student



LT Kyle Woerner
MIT Navy MS student



Michael Novitzky
Georgia Tech Visiting PhD

Objectives and Motivations



- Algorithms / Software Functionality
- Documentation
- Verification and Validation
- Competitions
- Lab sequences
- Example Missions
- Online Tutorials / Lectures

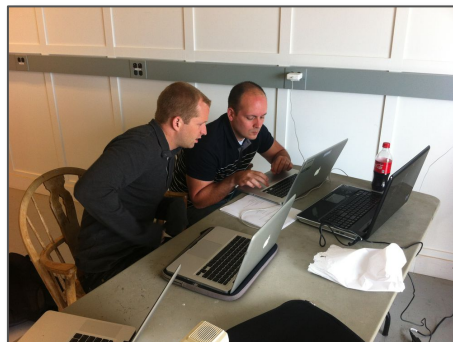
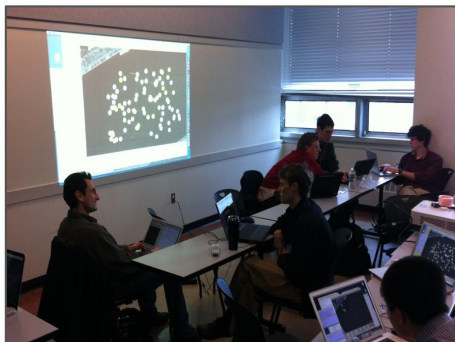


- Algorithms / Software Functionality (Stand-by Helm, uField Toolbox, AppCasting)
- Documentation (www.moos-ivp.org/download)
- Verification and Validation
- Competitions (Hazard Search Competition, Hunter-Prey Competition)
- Lab sequences (<http://oceanai.mit.edu/2680>)
- Example Missions (www.moos-ivp.org/download)
- Online Tutorials / Lectures (<http://oceanai.mit.edu/2680>)



- Effective **autonomy** can compensate for limits in **sensing** and **communications**.
- Effective **communications** can compensate for limits in **sensing** and **autonomy**.

- Understanding the ocean with marine robotic platforms.
- Educational focus is on autonomous decision making for marine robotic platforms.
- Students use an extensive MIT-developed autonomy and simulation codebase.
- On-board sensor-processing, inter-vehicle communications.



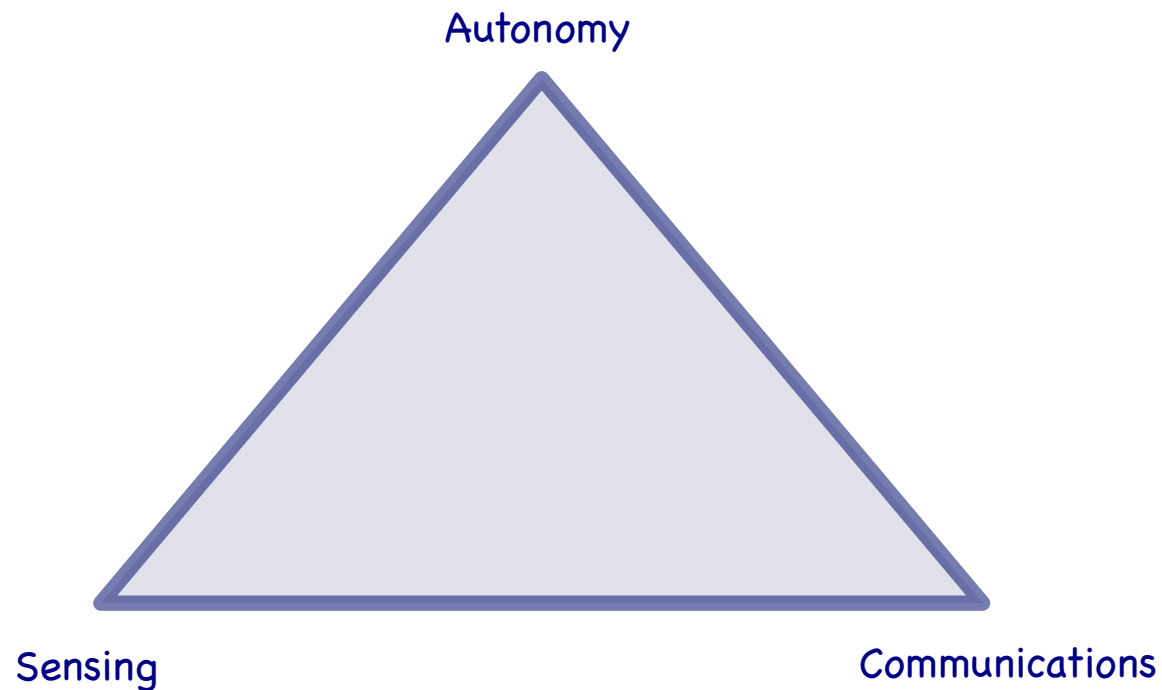


2.680 Marine Autonomy, Sensing and Communications (Funded by Battelle)



Class focus: **Autonomy**, **Sensing** and **Communications**

- Effective autonomy can compensate for limits in sensing and communications.
- Effective communications can compensate for limits in sensing and autonomy.



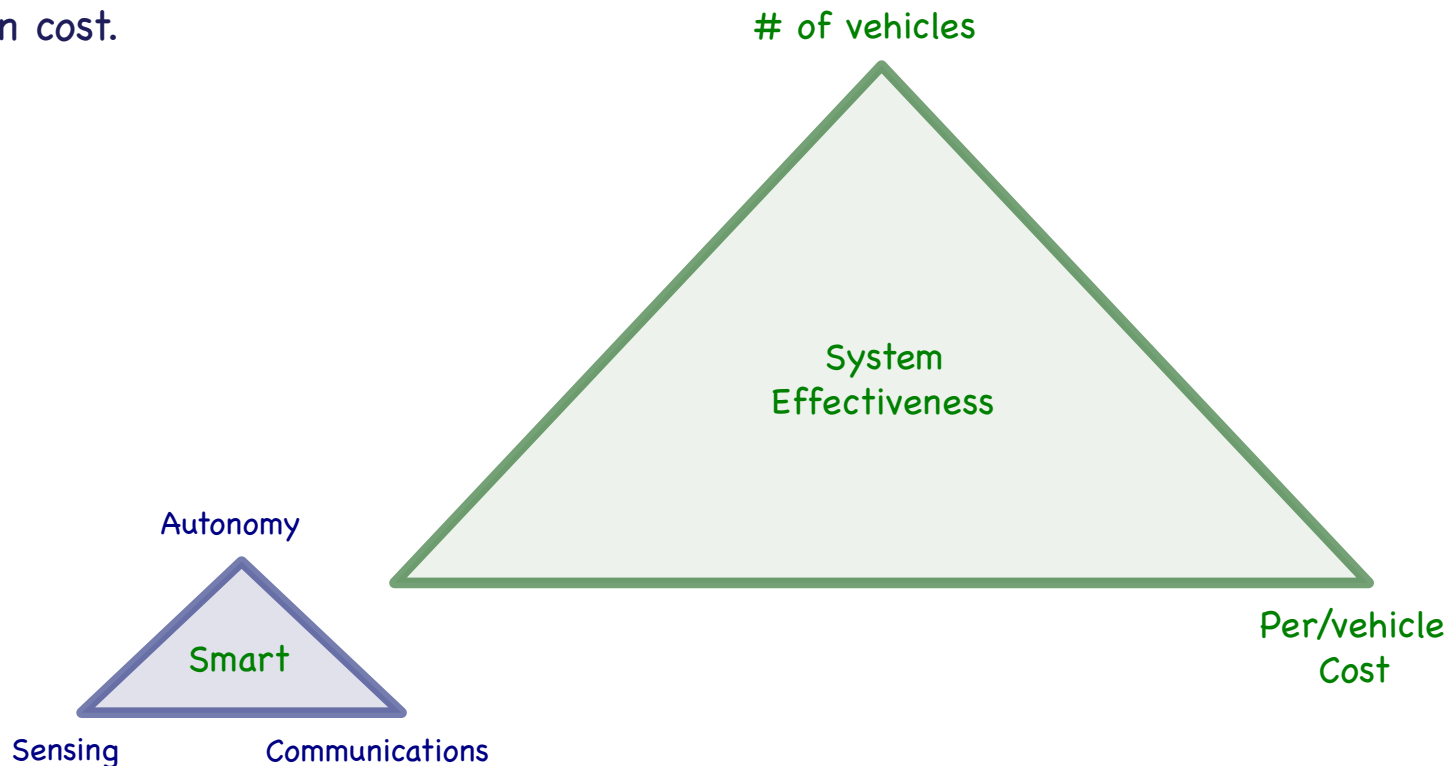


2.680 Marine Autonomy, Sensing and Communications (Funded by Battelle)



- A “smarter” vehicle means more can be done with fewer vehicles.
- A “smarter” vehicle means more can be done with a less capable (cheaper) vehicle.

Bottom line is system effectiveness
for a given cost.



In-Class Competitions

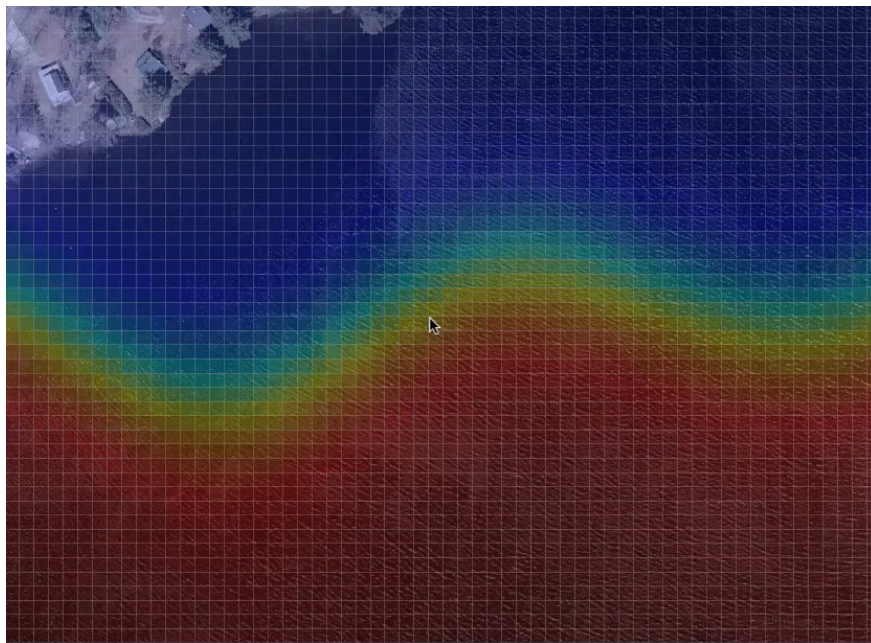
Autonomous Front Detection

Objective: Detect and characterize a moving temperature gradient

Output: Parameters of the gradient: **amplitude**, **period**, **angle**, **wavelength**, **offset**, **alpha**, **beta**, **temperature-north**, **temperature-south**.

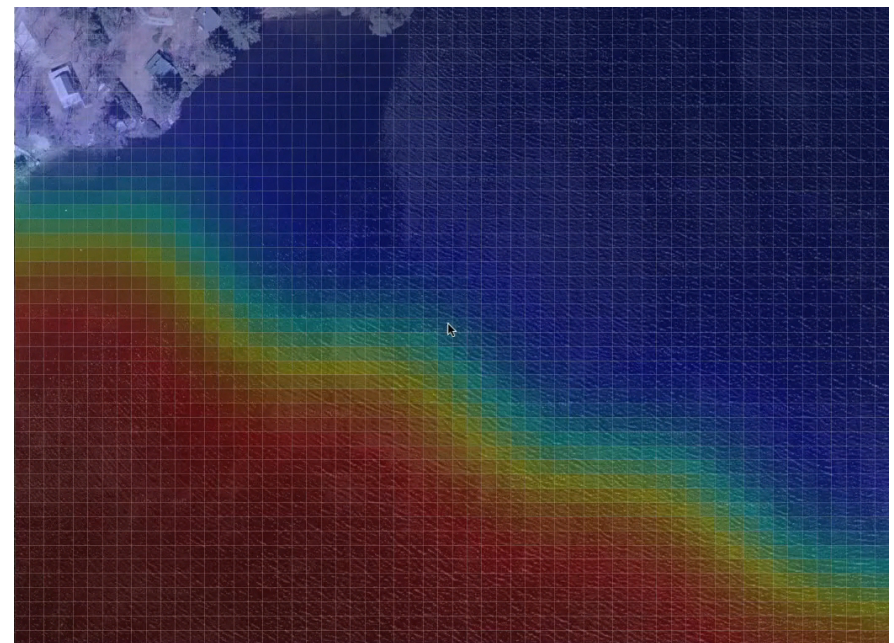
Given: A simulated CTD sensor and simulated annealer for parameter estimation.

Assignment: Build one or more vehicle behaviors for maneuvering the vehicle to collect CTD measurements.



```
offset = -90
angle = 5
amplitude = 20
period = 200
wavelength = 200
```

```
alpha = 400
beta = 20
temp_north = 20
temp_south = 25
```



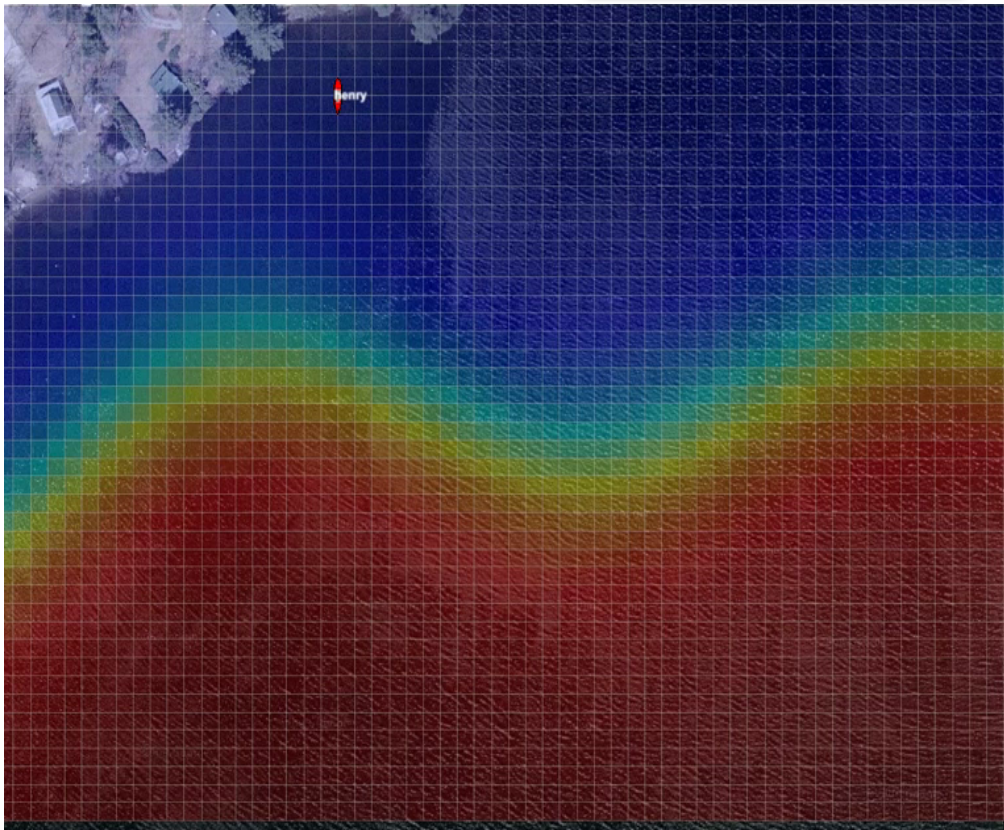
```
offset = -90
angle = -25
amplitude = 7
period = 75
wavelength = 100
```

```
alpha = 400
beta = 20
temp_north = 20
temp_south = 25
```


Autonomous Front Detection

Student Competition

Team Entry:
Rob Truax,
Isaac Evans



- Detection using three coordinated behaviors
- Combined using multi-objective optimization

Behavior #1: BHV_GoSideways

- Desired heading along decision line, whichever direction is closer.

Behavior #2: BHV_WaveFollow

- Constant desired speed
- Heading toward hot or cold with small offset
- Bang-bang setpoint controller to crisscross wavefront.

Behavior #3: BHV_RubberBand

- Holds vehicle inside the operation area

Autonomous Front Detection

Student Competition

Team Entry:

Rob Truax,
Isaac Evans



- Detection using three coordinated behaviors
- Combined using multi-objective optimization

Behavior #1: BHV_GoSideways

- Desired heading along decision line, whichever direction is closer.

Behavior #2: BHV_WaveFollow

- Constant desired speed
- Heading toward hot or cold with small offset
- Bang-bang setpoint controller to crisscross wavefront.

Behavior #3: BHV_RubberBand

- Holds vehicle inside the operation area



MIT Autonomy Lab

At the MIT Sailing Pavilion



On-campus, nearly year-round facility

- Used by MIT 2.680 students, and graduate research for several faculty
- Yearly fees support by MIT Mechanical Engineering and Battelle
- Lab Equipment and vehicles funded by Battelle





MIT Autonomy Lab

At the MIT Sailing Pavilion



On-campus, nearly year-round facility

- Used by MIT 2.680 students, and graduate research for several faculty
- Yearly fees support by MIT Mechanical Engineering and Battelle
- Lab Equipment and vehicles funded by Battelle





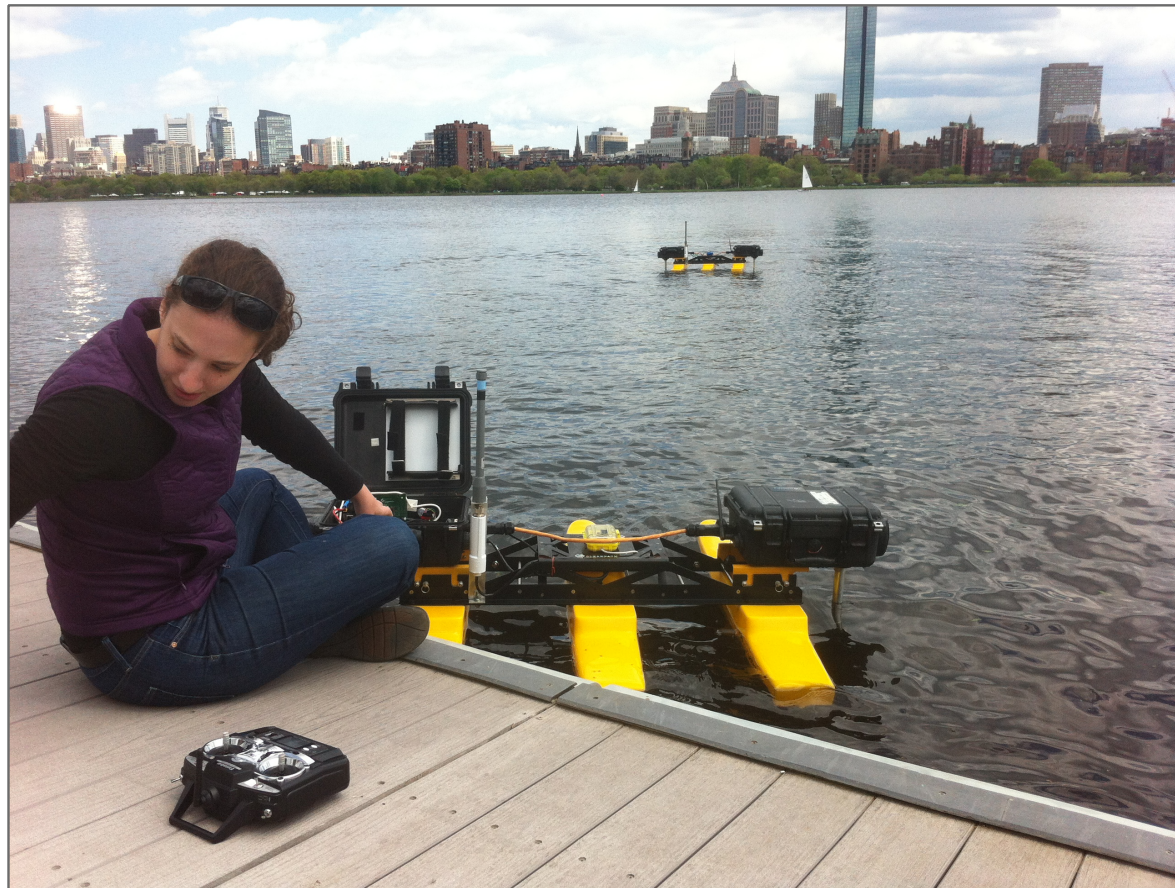
MIT Autonomy Lab

At the MIT Sailing Pavilion



On-campus, nearly year-round facility

- Used by MIT 2,680 students, and graduate research for several faculty
- Yearly fees support by MIT Mechanical Engineering and Battelle
- Lab Equipment and vehicles funded by Battelle





MIT Autonomy Lab

At the MIT Sailing Pavilion



On-campus, nearly year-round facility

- Used by MIT 2.680 students, and graduate research for several faculty
- Yearly fees support by MIT Mechanical Engineering and Battelle
- Lab Equipment and vehicles funded by Battelle





MIT Autonomy Lab

At the MIT Sailing Pavilion



On-campus, nearly year-round facility

- Used by MIT 2.680 students, and graduate research for several faculty
- Yearly fees support by MIT Mechanical Engineering and Battelle
- Lab Equipment and vehicles funded by Battelle

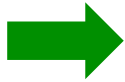




Outline



- Objectives and Motivations
- The Marine Autonomy Courseware
Lectures, Labs, Documentation at <http://oceanai.mit.edu/2680>
- The uField Toolbox
- AppCasting
- Changes / Additions to the Helm



The Shoreside/Vehicle Topology

"Shoreside" could be:



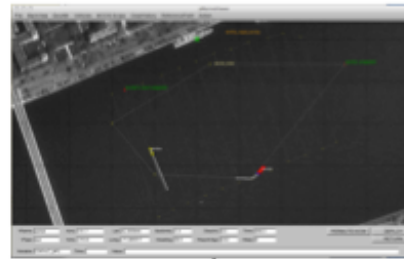
Topside on a ship



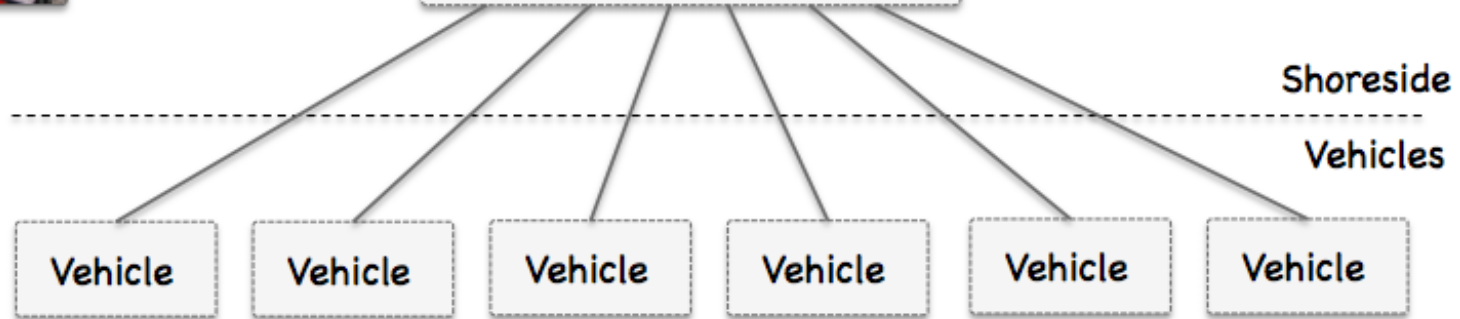
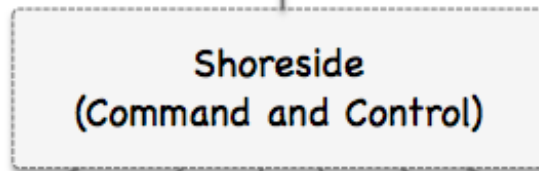
At a shoreside lab
MIT Pavilion



Instructor's
computer

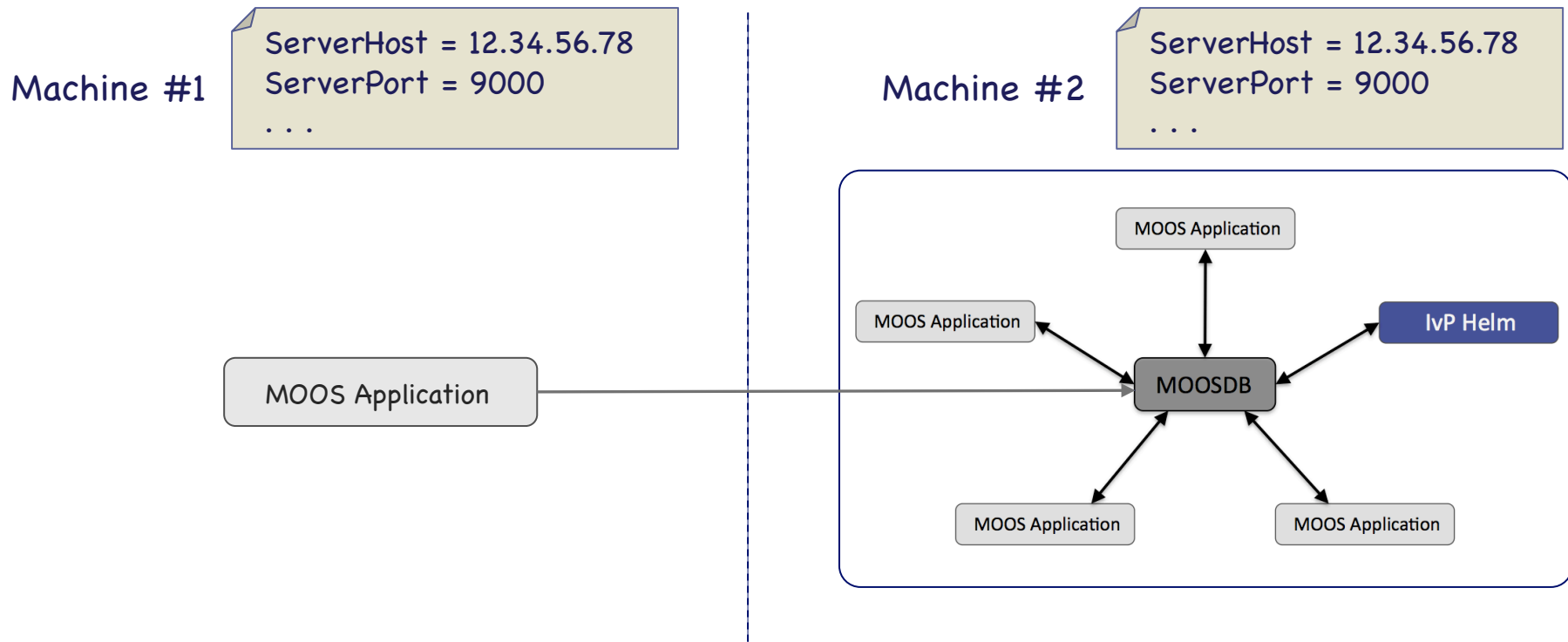


User(s)



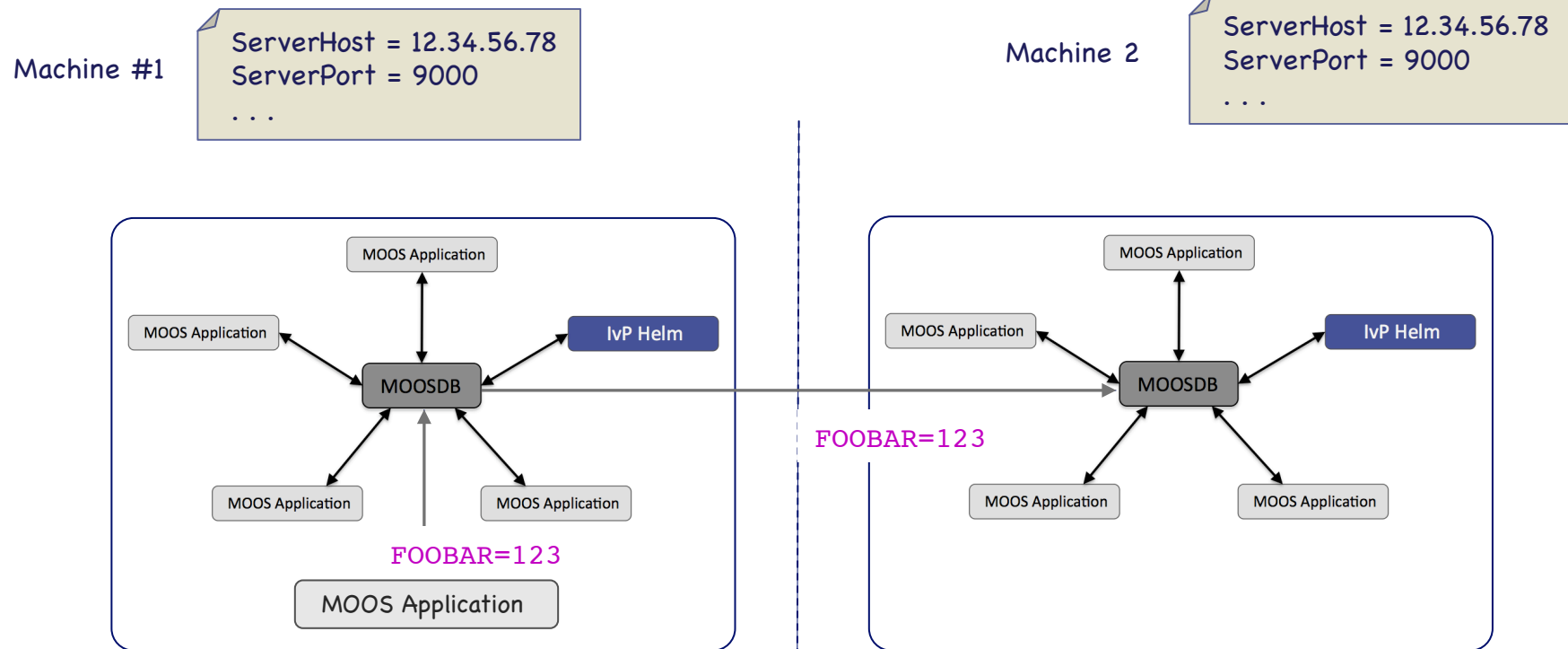
Communications Between Machines / Vehicles

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



Communications Between Machines / Vehicles

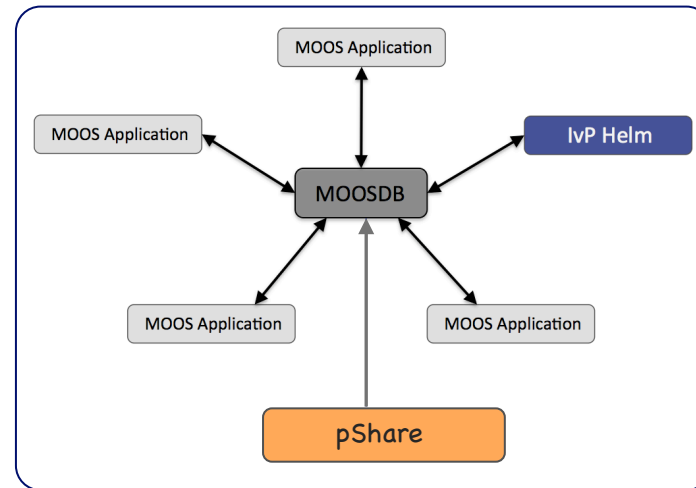
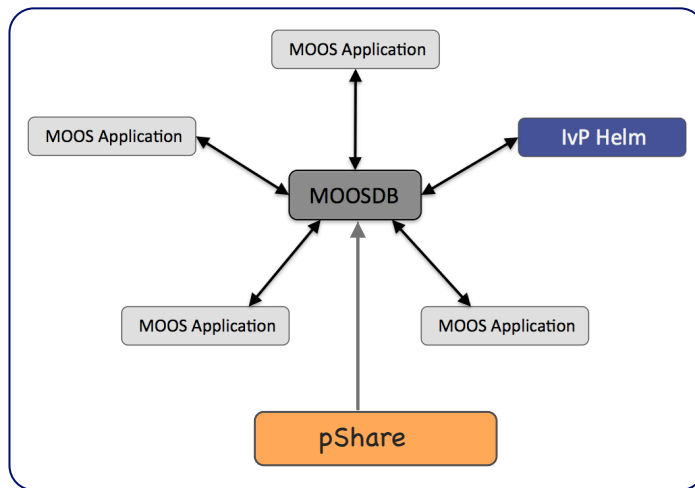
How do we get two MOOSDB's (communities) to talk to each other?



When the two machines are on the same network, we can use [pShare](#).

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.

pShare Configuration

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

```
ProcessConfig = pShare
{
  input  = route=localhost:9201

  output = src_name=VIEW_POLYGON, route=localhost:9202, dest_name=POLYGON
}
```

The port *we* are listening on

Name on target machine

Name of variable locally

IP address of target machine

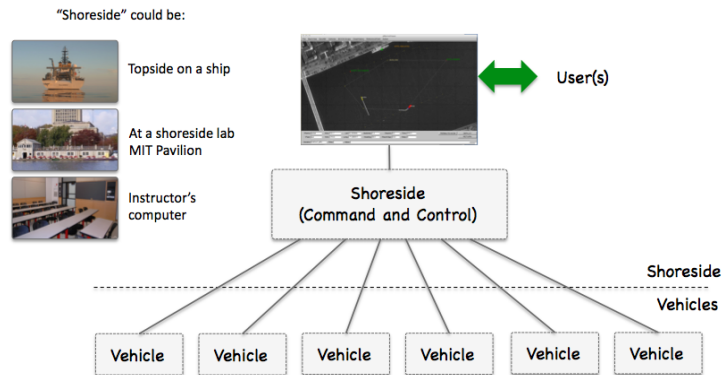
Port on which pShare is listening in the target community.

The uField Toolbox

(Overview)

The **uField Toolbox** is:

- A collection of about a dozen MOOS applications, each a **U**tility for **F**ielding multiple vehicles with a shoreside/topside command-and-control MOOS Community.



- All applications are documented in the MOOS-IvP Tools document, online. <http://oceanai.mit.edu/moos-ivp-pdf/moosivp-tools.pdf>

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

The uField Toolbox

(Overview)

All applications are documented in the MOOS-IvP Tools document, online.
<http://oceanai.mit.edu/moos-ivp-pdf/moosivp-tools.pdf>

MOOS-IvP Autonomy Tools Users Manual
Release 13.2

13.2

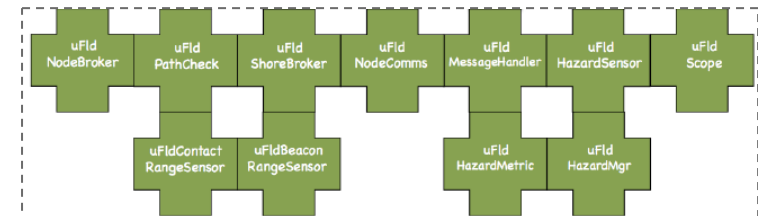
Michael R. Benjamin
 Department Mechanical Engineering
 Computer Science and Artificial Intelligence Laboratory
 Massachusetts Institute of Technology, Cambridge MA

February 14, 2013 - Release 13.2

Abstract

This document describes several MOOS-IvP autonomy tools. *uHelmScope* provides a run-time scoping window into the state of an active IvP Helm executing its mission. *pMarineViewer* is a geo-based GUI tool for rendering marine vehicles and geometric data in their operational area. *uXMS* is a terminal based tool for scoping on a MOOSDB process. *uTermCommand* is a terminal based tool for poking a MOOSDB with a set of MOOS file pre-defined variable-value pairs selectable with aliases from the command-line. *pEchoVar* provides a way of echoing a post to one MOOS variable with a new post having the same value to a different variable. *uProcessWatch* monitors the presence or absence of a set of MOOS processes and summarizes the collective status in a single MOOS variable. *uPokeDB* provides a way of poking the MOOSDB

(uField Toolbox Apps)





The uField Toolbox

(Overview)



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

1. Facilitation of Inter MOOSDB Share configuration

- pHostInfo
- uFldNodeBroker
- uFldShoreBroker

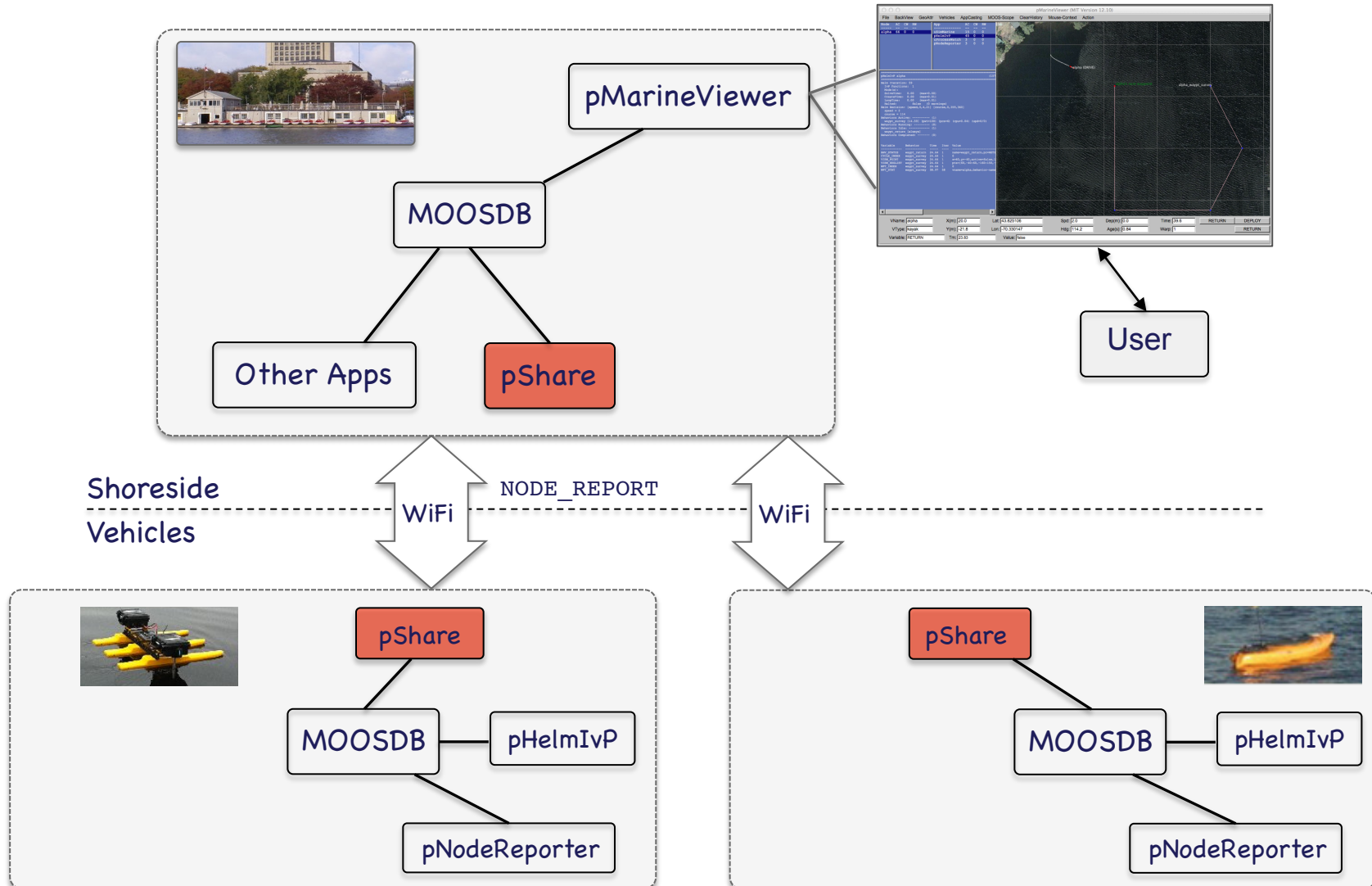
2. Simulation of Inter-Vehicle Messaging

- uFldNodeComms
- uFldMessageHandler

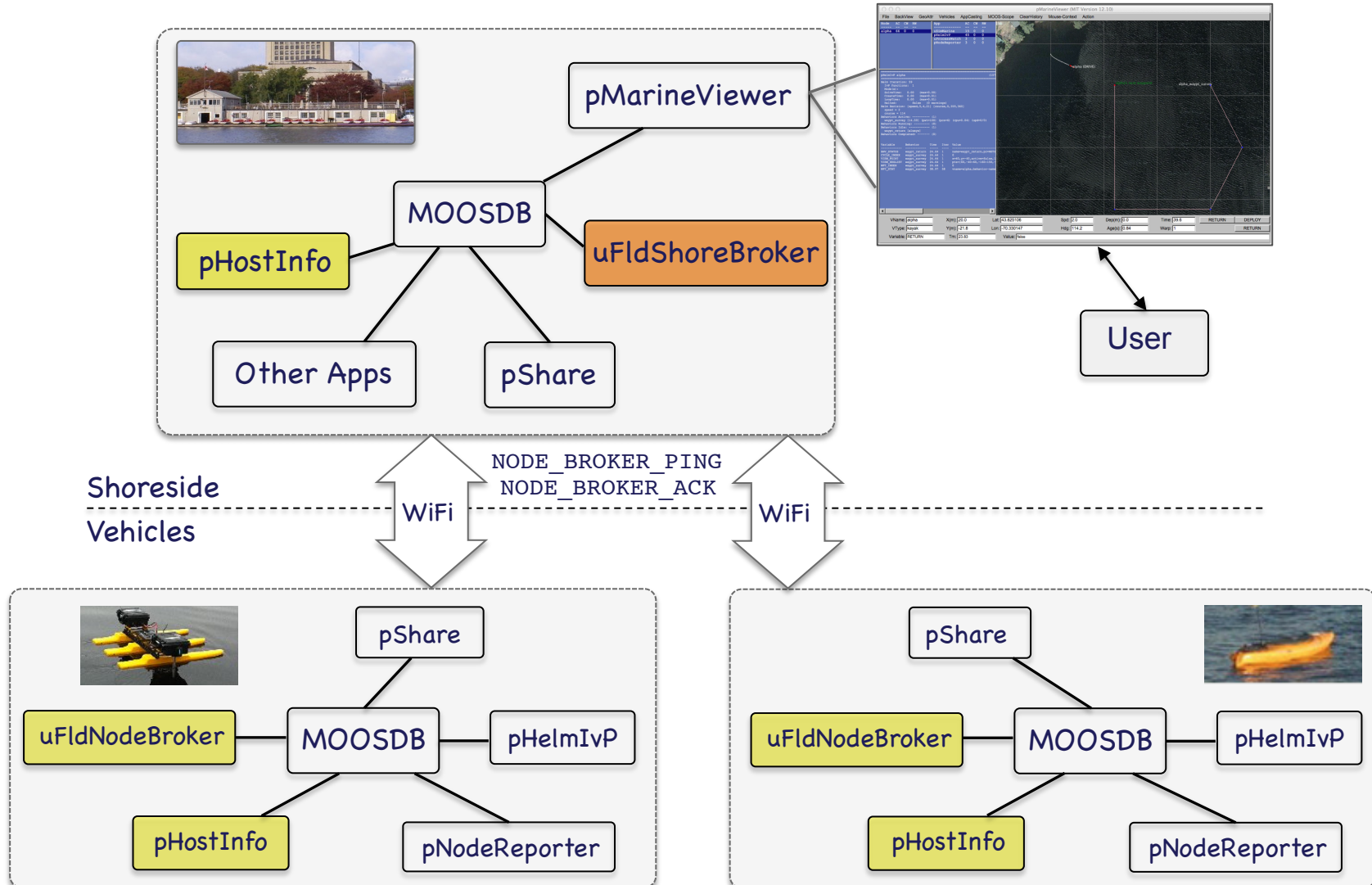
3. Sensor Simulation

- uFldHazardSensor
- uFldHazardMgr
- uFldHazardMetric
- uFldContactRangeSensor
- uFldBeaconRangeSensor

Inter-MOOSDB sharing needs to be configured:



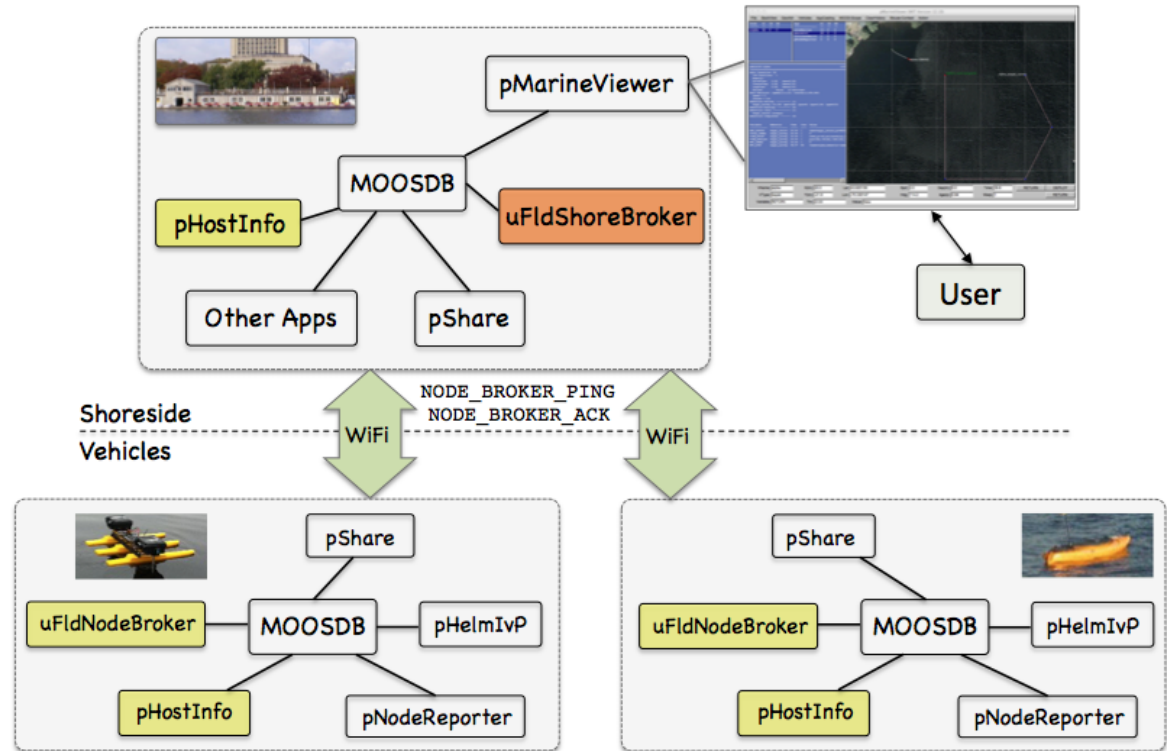
We want this to be as automatic as possible.



We want this to be as automatic as possible.

pHostInfo

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

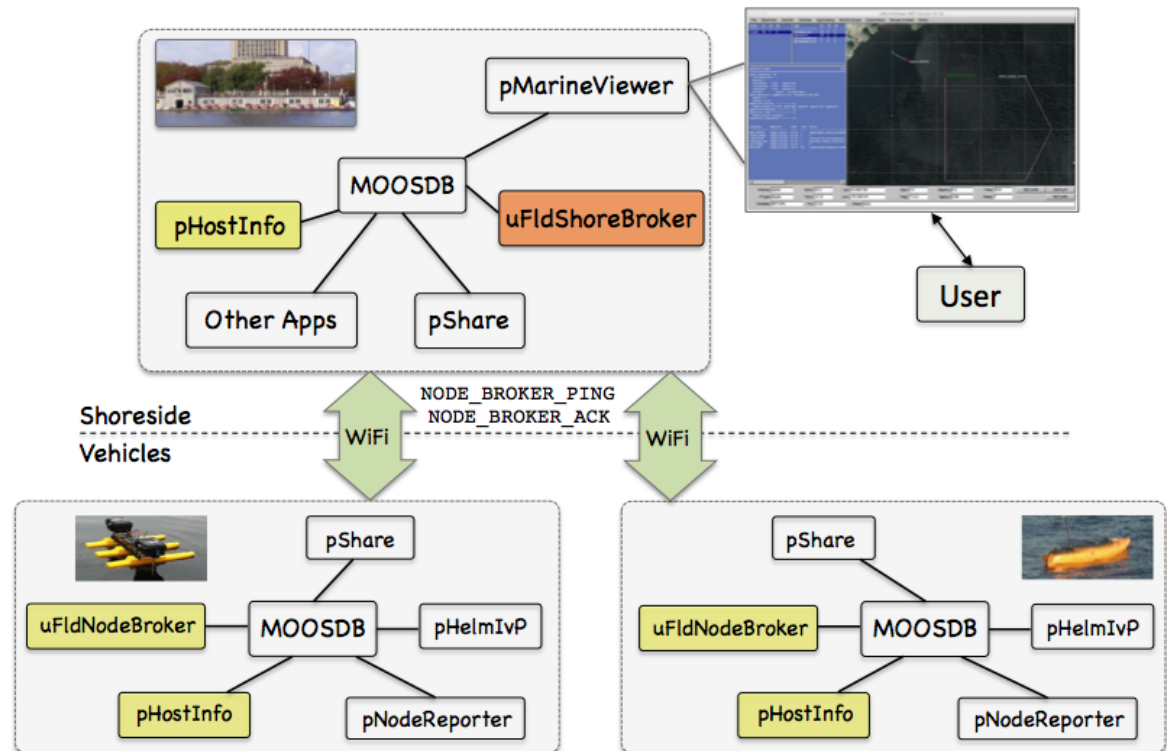


We want this to be as automatic as possible.

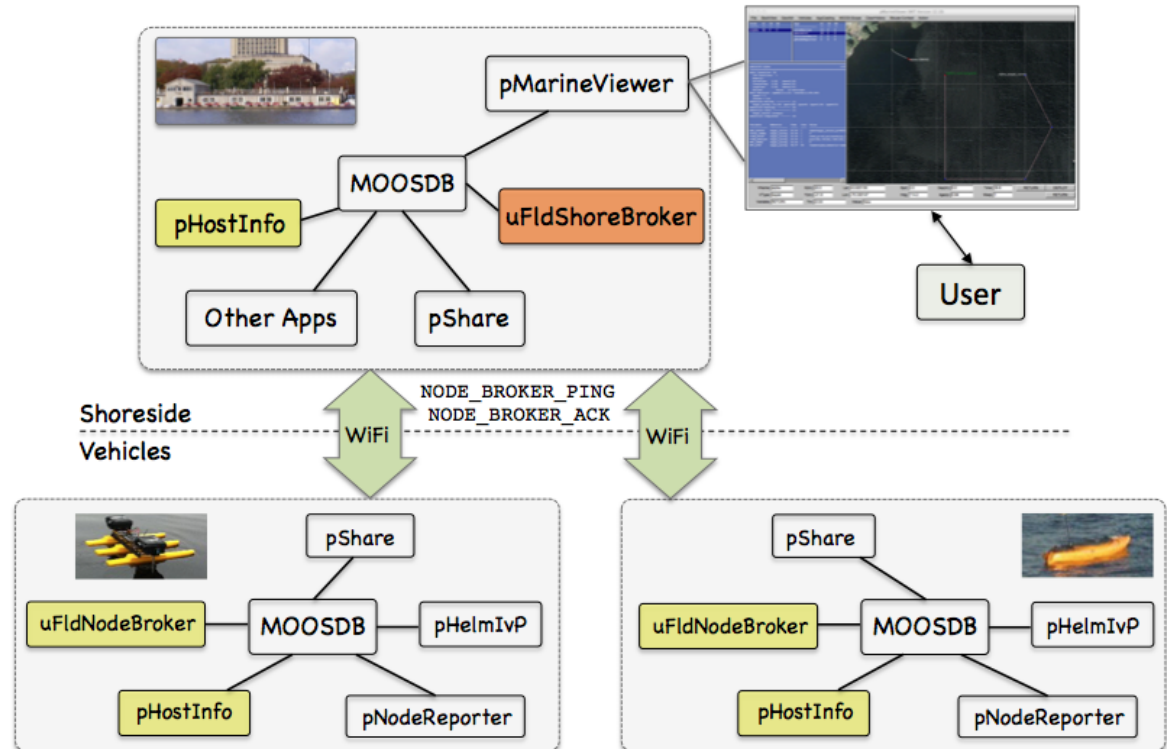
uFldNodeBroker

A MOOS app for

- finding a shore side,
- determining it's IP address and pShare input route,
- Auto-configuring its own local pShare outgoing route



We want this to be as automatic as possible.



uFldShoreBroker

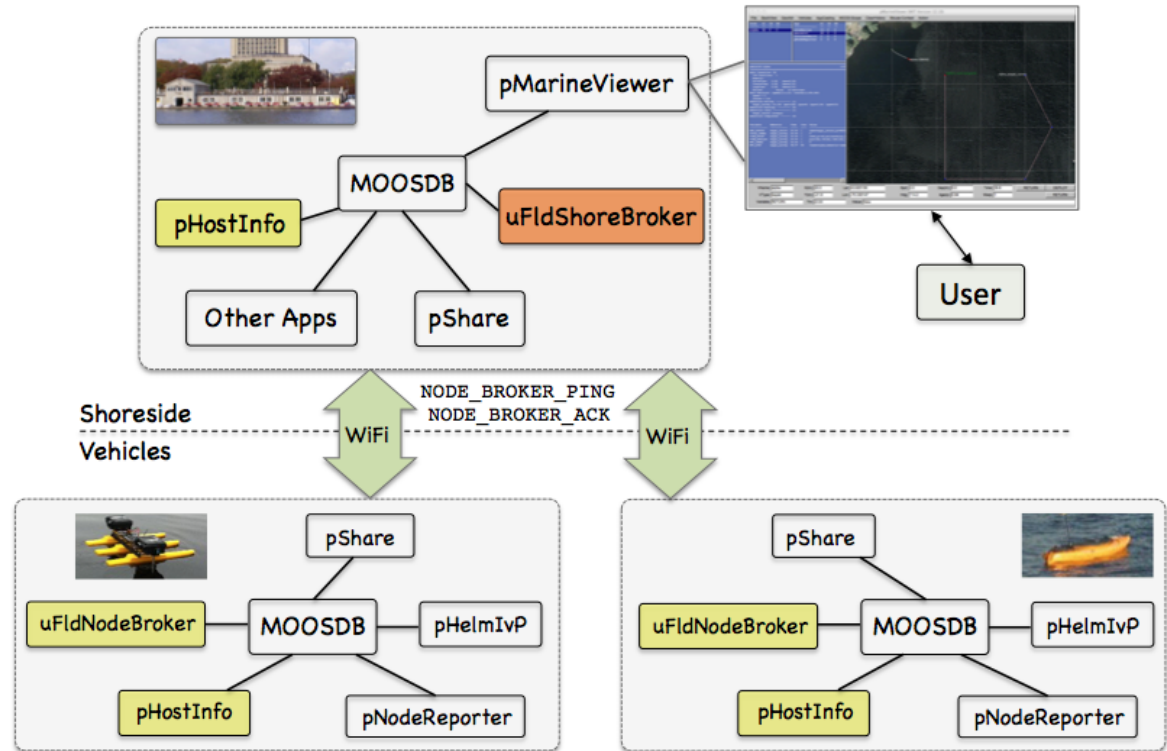
A MOOS app for

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

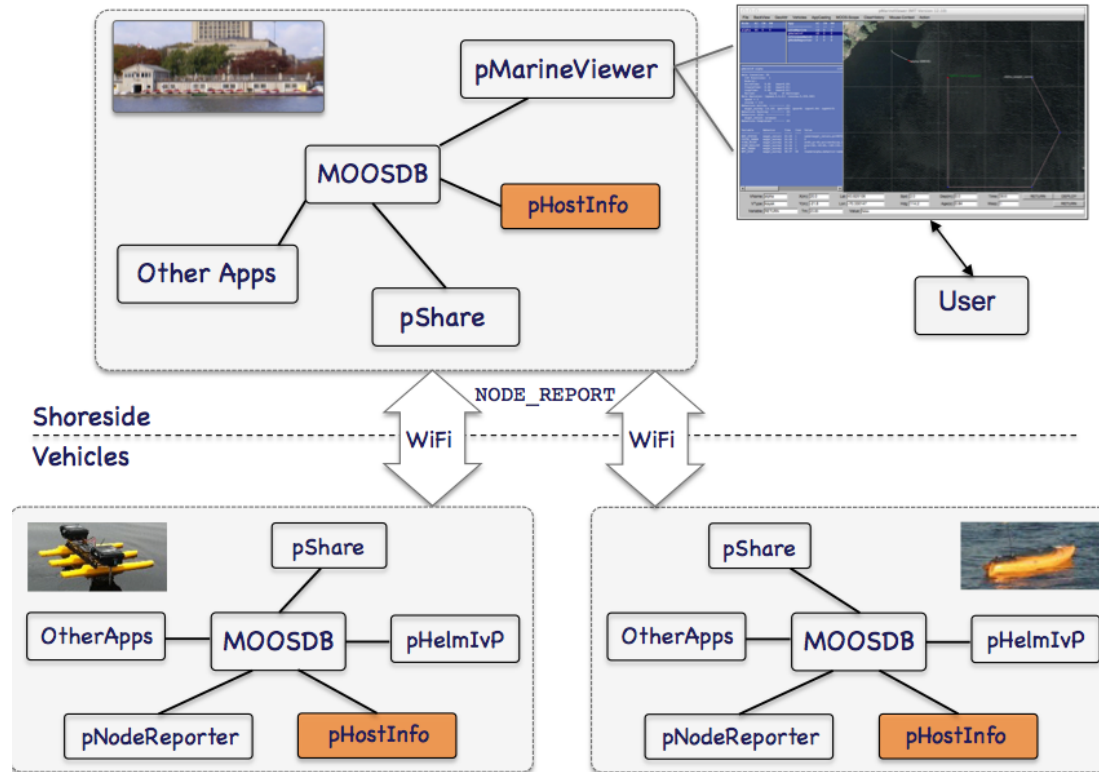
We want this to be as automatic as possible.

pHostInfo

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



Purpose: Determine the IP address of the machine.
 Publish the result in PHI_HOST_IP

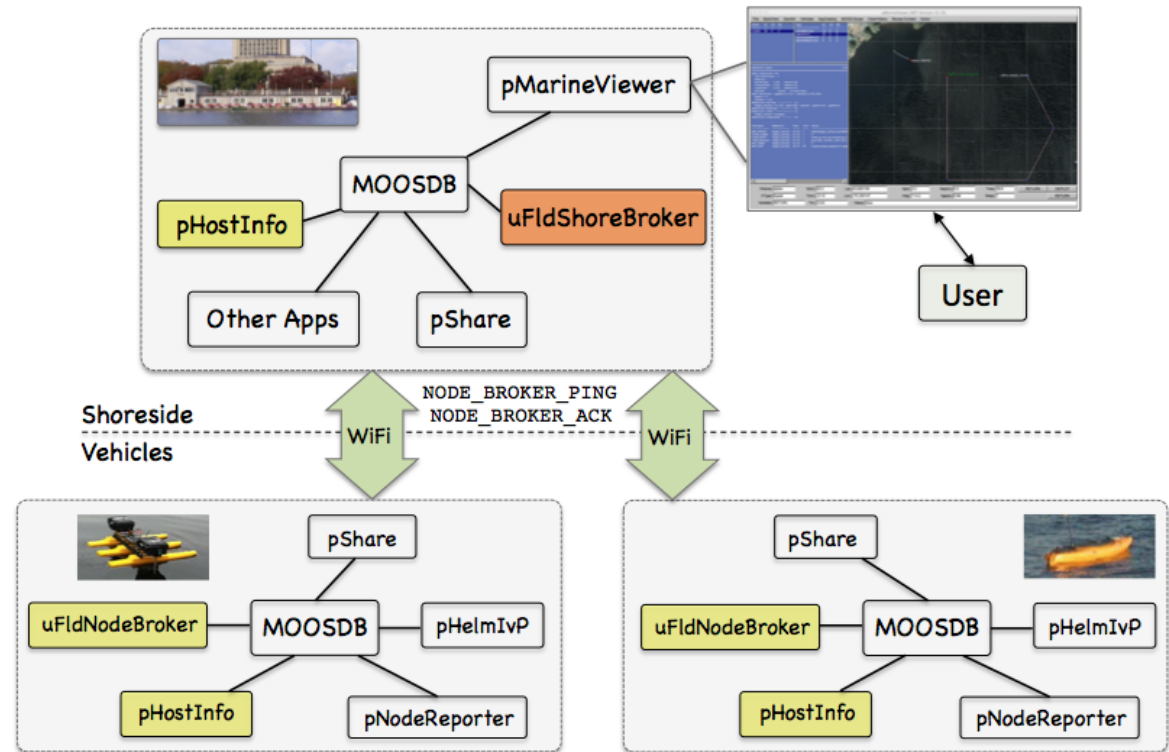


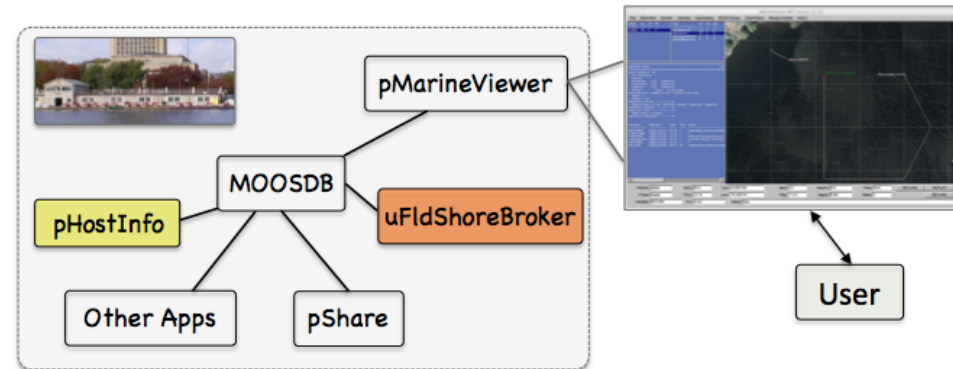
```

PHI_HOST_IP = 118.10.24.23
PHI_HOST_IP_ALL = 118.10.24.23,169.224.126.40
PHI_HOST_IP_VERBOSE = OSX_ETHERNET2=118.10.24.23,OSX_AIRPOT=169.224.126.40
  
```

uFldNodeBroker

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

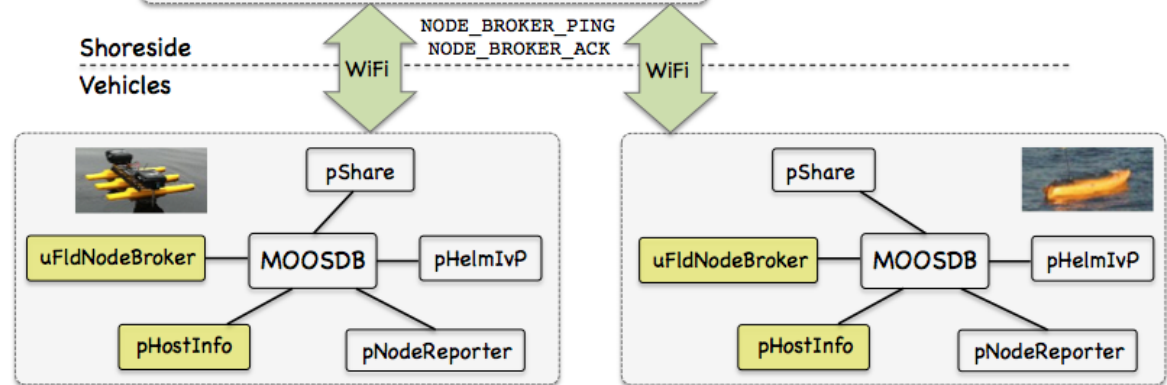




uFldNodeBroker

A MOOS app for

- finding a shore side,
- determining it's IP address and pShare input route,
- Auto-configuring its own local pShare outgoing route



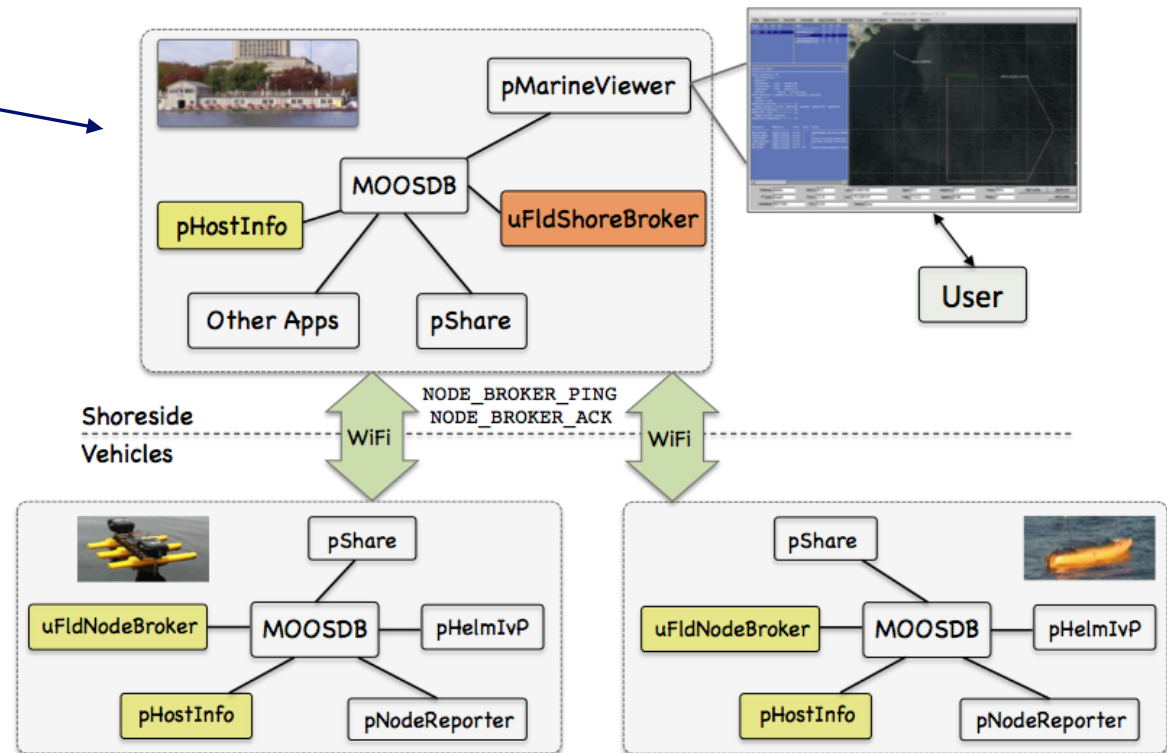
- Gets local host IP information from pHostInfo.
- Pings a candidate shoreside community with information about itself
`NODE_BROKER_PING = community=henry,hostip=192.168.1.1,port_db=9000,pshare_iroutes=192.168.1.1:9200,timewarp=8`
- Receives reply from shoreside with information about the shoreside community.
`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`

Runs in the **shoreside** community

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

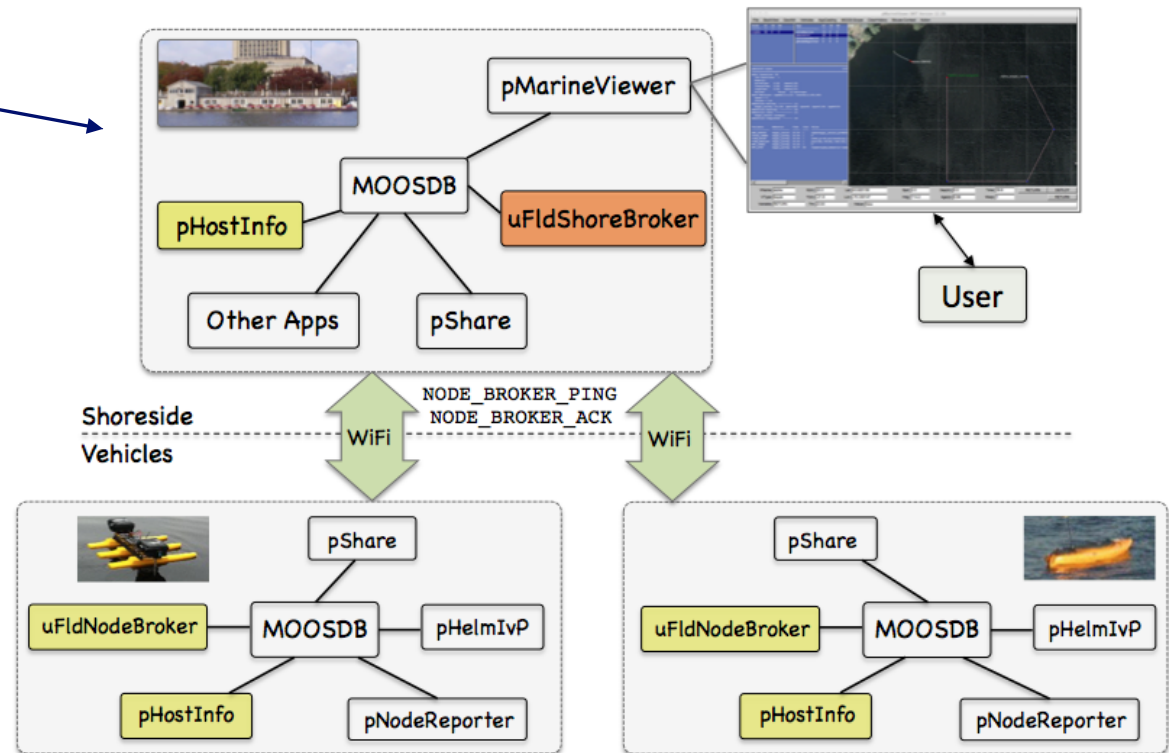


Runs in the **shoreside** community

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



- Gets local host IP information from pHostInfo.
- Receives a ping from a candidate shoreside community with information about a vehicle.

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

- Sends reply from shoreside to vehicle 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
```



The uField Toolbox

(Overview)



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

1. Facilitation of Inter MOOSDB Share configuration

- pHostInfo
- uFldNodeBroker
- uFldShoreBroker

2. Simulation of Inter-Vehicle Messaging

- uFldNodeComms
- uFldMessageHandler

3. Sensor Simulation

- uFldHazardSensor
- uFldHazardMgr
- uFldHazardMetric
- uFldContactRangeSensor
- uFldBeaconRangeSensor

The uFldMessageHandler App

Inter-vehicle messaging

Vehicle **alpha** (source vehicle)

(Some MOOS App)
Publishes:

```
NODE_MESSAGE_LOCAL =  
  "src_node=alpha,dest_node=bravo,  
  var_name=STATUS,string_var=searching"
```



Vehicle **bravo** (dest vehicle)

uFldMessageHandler
Subscribes/Handles:
Publishes:

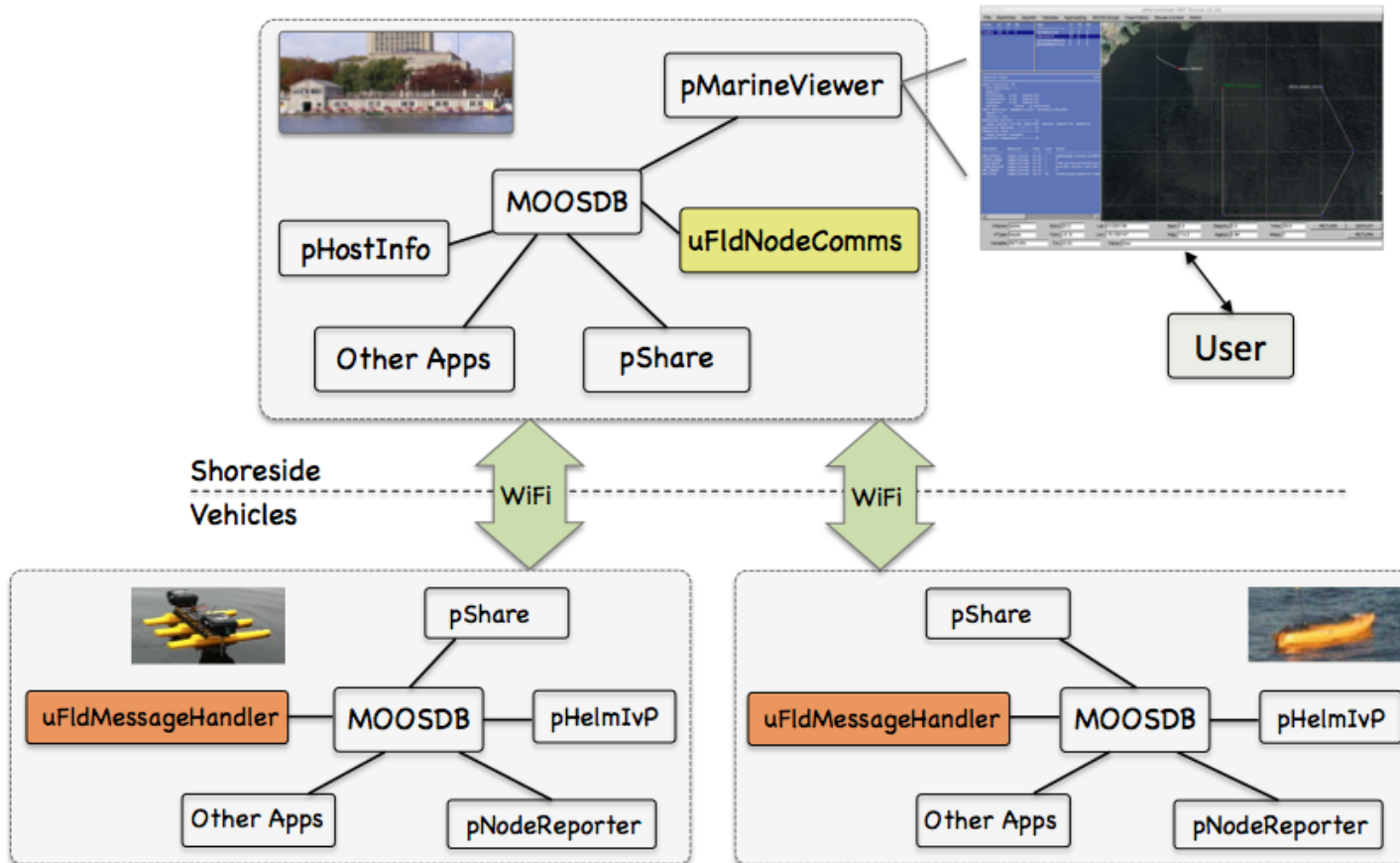
```
NODE_MESSAGE =  
  "src_node=alpha,dest_node=bravo,  
  var_name=STATUS,string_var=searching"
```

```
STATUS = "searching"
```

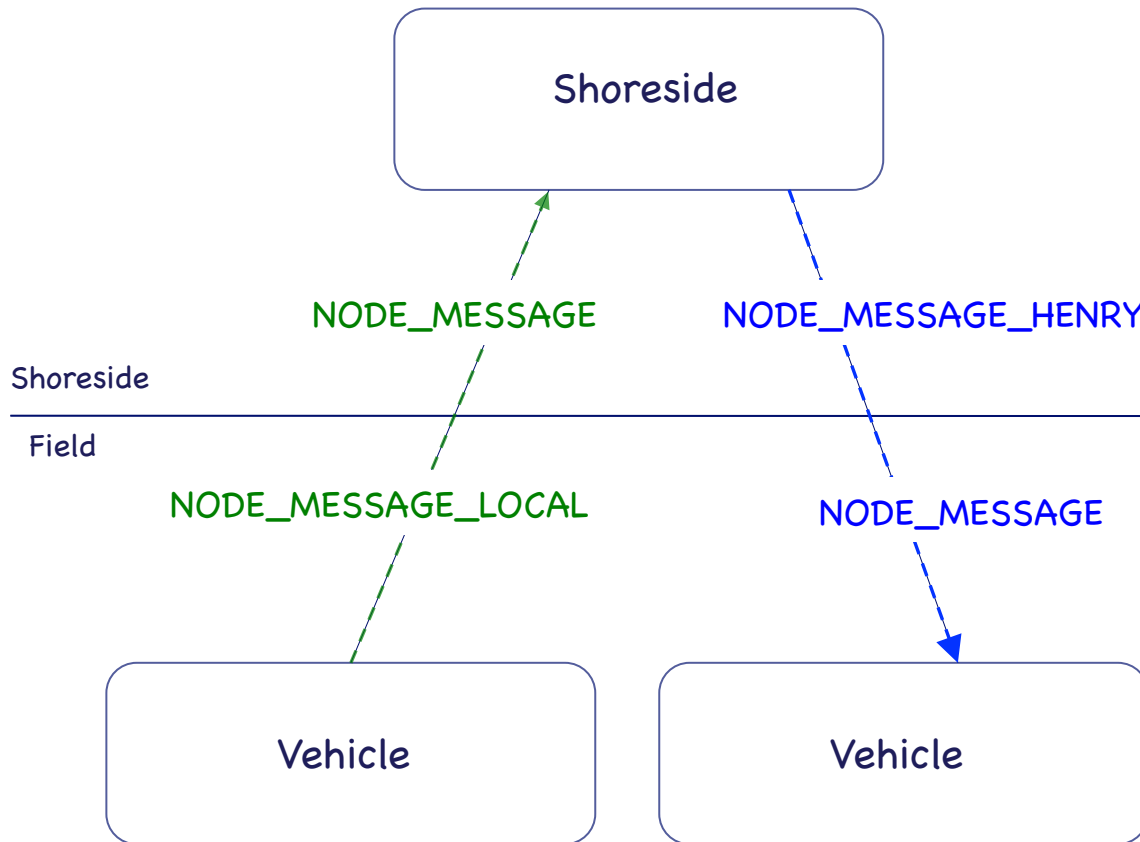
The uFldMessageHandler App

Typical Topology

The **uFldMessageHandler** app is running on all vehicles wishing to receive messages.



Message routing is handled on the shoreside



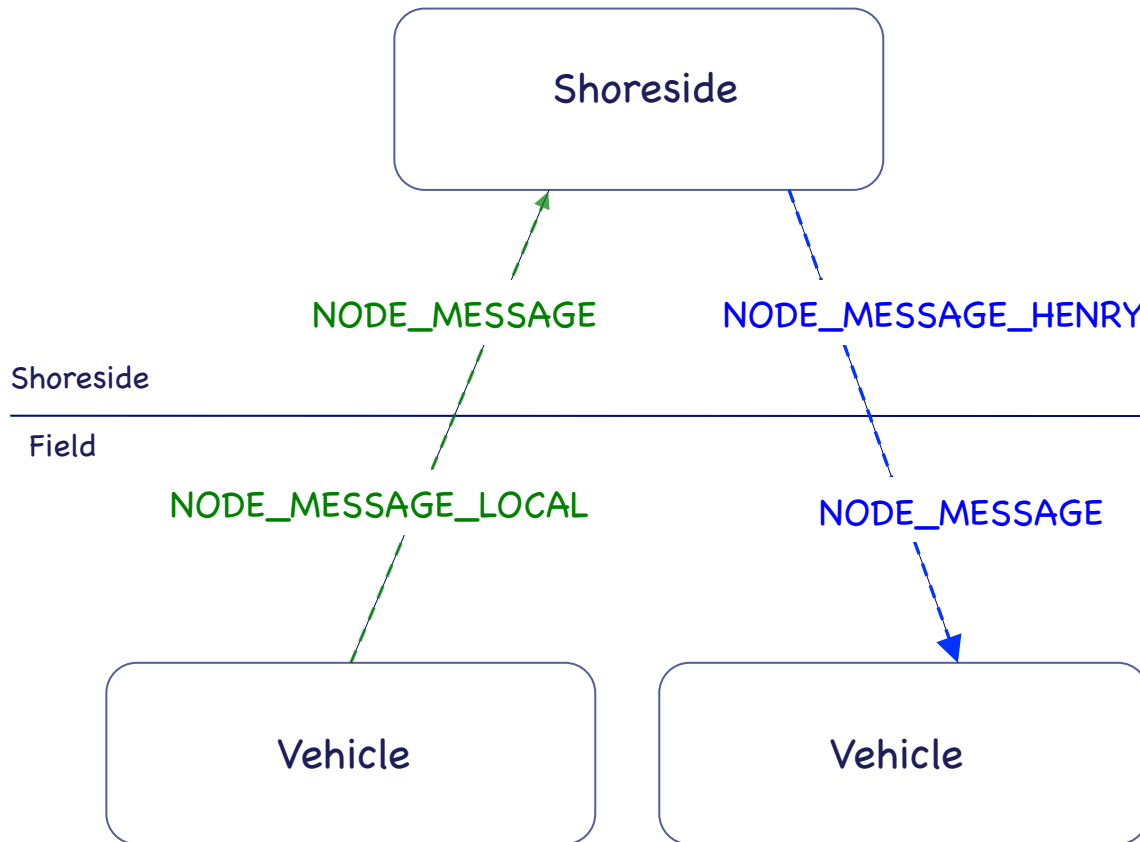
```

ProcessConfig = uFieldShoreBroker
{
  QBRIDGE = NODE_MESSAGE
}
  
```

```

ProcessConfig = uFieldNodeBroker
{
  BRIDGE = src=NODE_MESSAGE_LOCAL
           alias=NODE_MESSAGE
}
  
```

Message routing is handled on the shoreside
 But it's not the case that all messages make it through
 They are handled by uFldNodeComms.



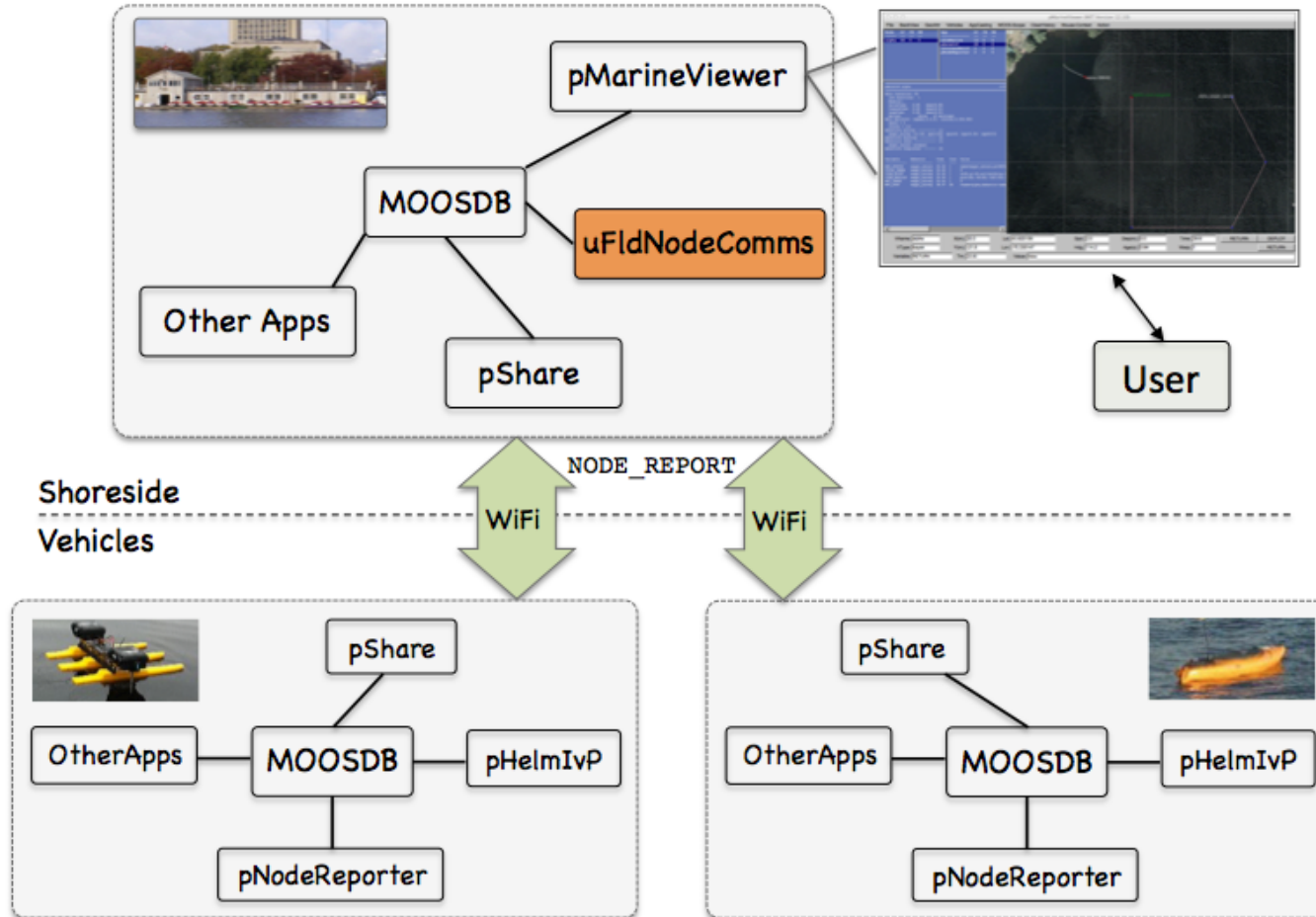
```
ProcessConfig = uFieldShoreBroker
{
    QBRIDGE = NODE_MESSAGE
}
```

```
ProcessConfig = uFieldNodeBroker
{
    BRIDGE = src=NODE_MESSAGE_LOCAL
            alias=NODE_MESSAGE
}
```


The uFldNodeComms App

Typical Topology

The **uFldNodeComms** app runs on the shoreside, limits intervehicle messaging.



The uFldNodeComms App

Typical Application Topology

The `uFldNodeComms` configuration parameters:

```
ProcessConfig = uFldNodeComms
{
  COMMS_RANGE      = 200
  MIN_MSG_INTERVAL = 60
  MAX_MSG_LENGTH   = 100
}
```

Distance in meters between vehicles

Min time between messages from a vehicle

Max chars in a string message



The uField Toolbox

(Sensor Simulation)



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

1. Facilitation of Inter MOOSDB Share configuration

- pHostInfo
- uFldNodeBroker
- uFldShoreBroker

2. Simulation of Inter-Vehicle Messaging

- uFldNodeComms
- uFldMessageHandler

3. Sensor Simulation

- uFldHazardSensor
- uFldHazardMgr
- uFldHazardMetric
- uFldContactRangeSensor
- uFldBeaconRangeSensor

The uField Toolbox

(Sensor Simulation)

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

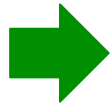
1. Facilitation of Inter MOOSDB Share configuration

- pHostInfo
- uFldNodeBroker
- uFldShoreBroker

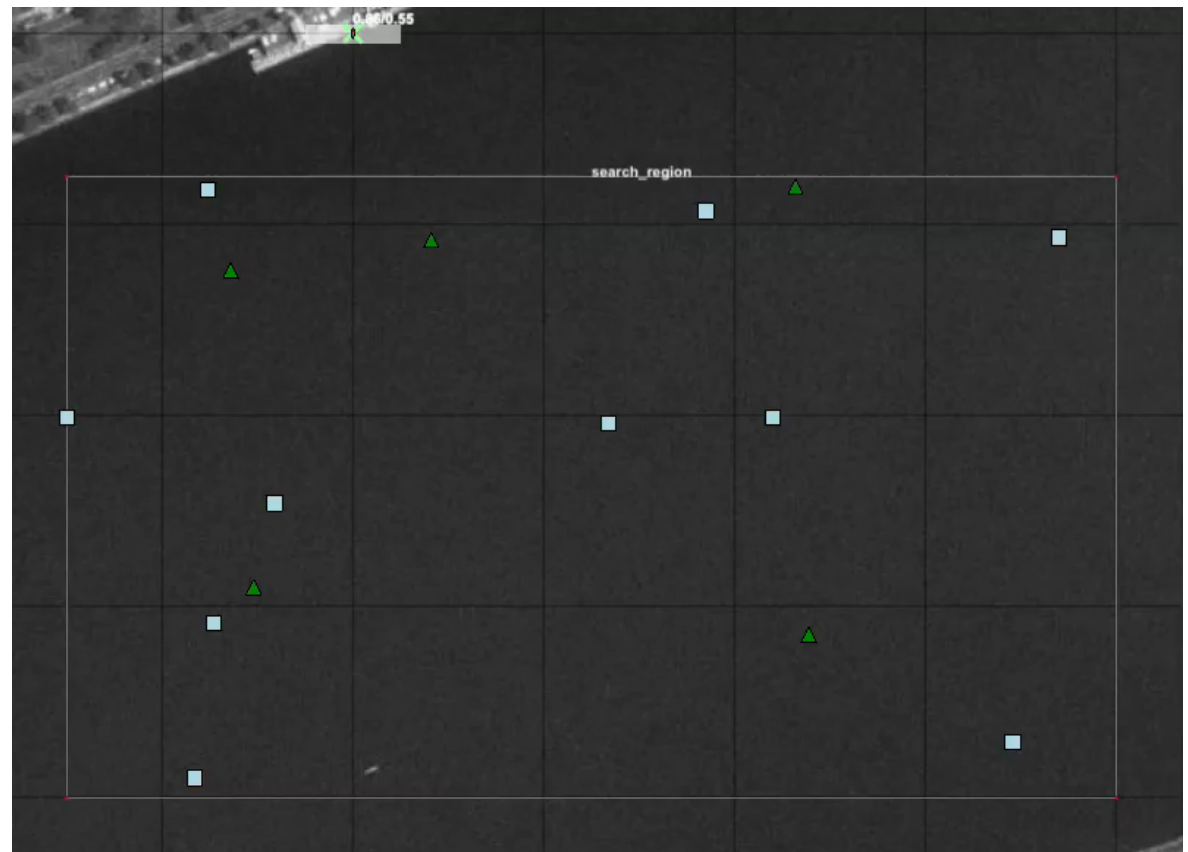
2. Simulation of Inter-Vehicle Messaging

- uFldNodeComms
- uFldMessageHandler

3. Sensor Simulation



- uFldHazardSensor
- uFldHazardMgr
- uFldHazardMetric
- uFldContactRangeSensor
- uFldBeaconRangeSensor



The uField Toolbox

(Sensor Simulation)

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

1. Facilitation of Inter MOOSDB Share configuration

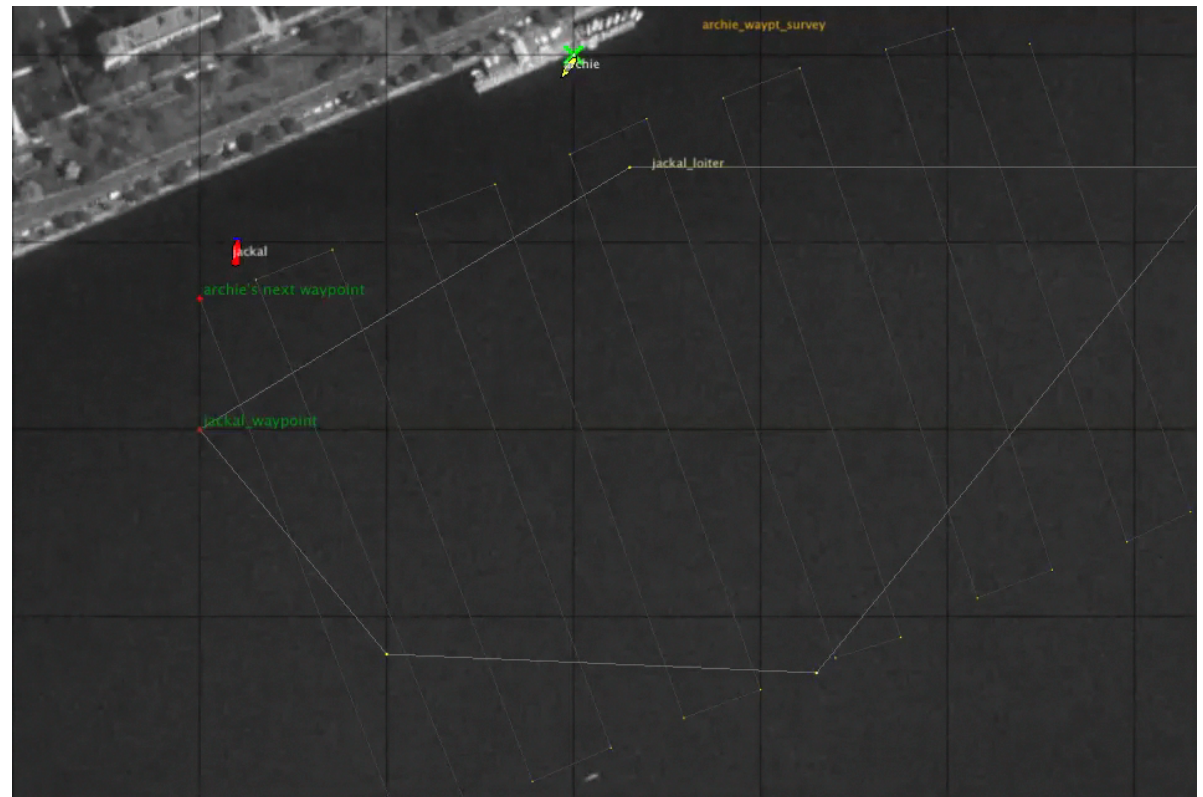
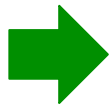
- pHostInfo
- uFldNodeBroker
- uFldShoreBroker

2. Simulation of Inter-Vehicle Messaging

- uFldNodeComms
- uFldMessageHandler

3. Sensor Simulation

- uFldHazardSensor
- uFldHazardMgr
- uFldHazardMetric
- uFldContactRangeSensor
- uFldBeaconRangeSensor



The uField Toolbox

(Sensor Simulation)

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

1. Facilitation of Inter MOOSDB Share configuration

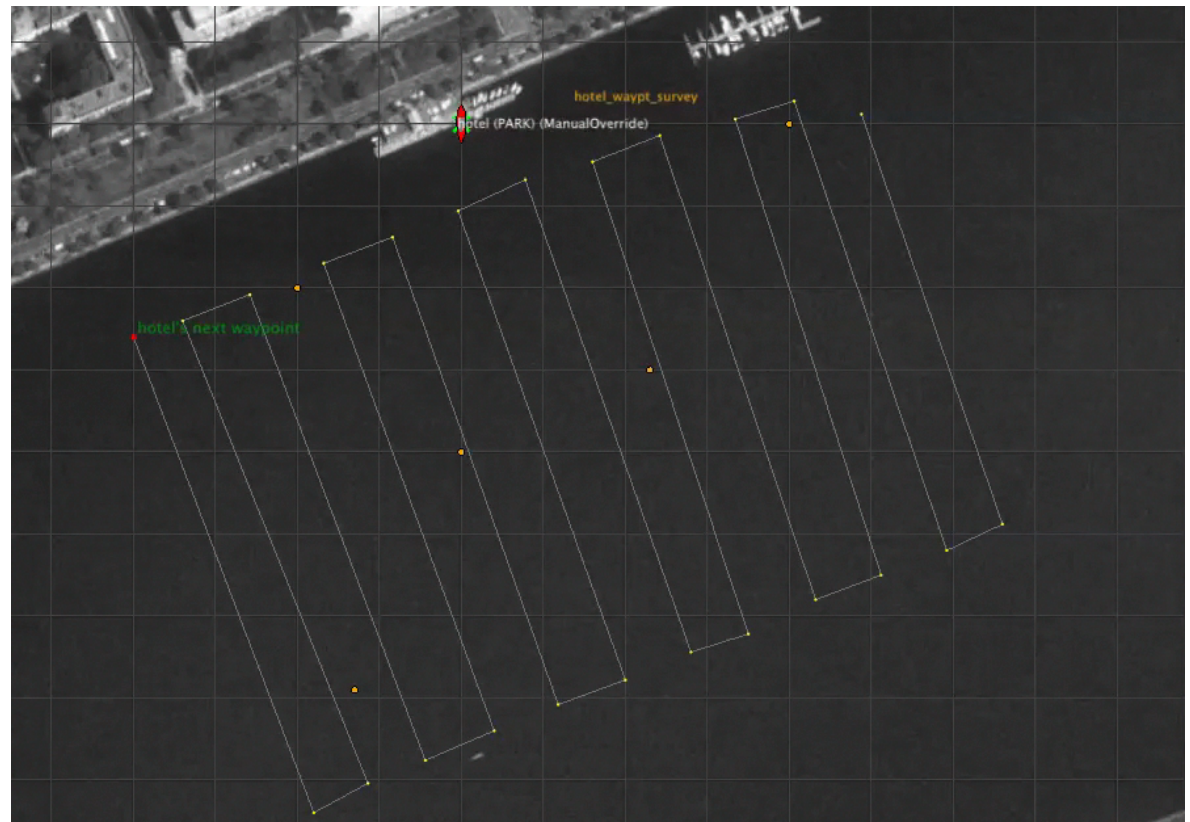
- pHostInfo
- uFldNodeBroker
- uFldShoreBroker

2. Simulation of Inter-Vehicle Messaging

- uFldNodeComms
- uFldMessageHandler

3. Sensor Simulation

- uFldHazardSensor
- uFldHazardMgr
- uFldHazardMetric
- uFldContactRangeSensor
- uFldBeaconRangeSensor



- Objectives and Motivations
- The Marine Autonomy Courseware
- The uField Toolbox
- AppCasting
- Changes / Additions to the Helm
- Ongoing / Future Efforts





AppCasting

in MOOS

AppCasting was motivated by a few observations:

- The biggest headache of users new to MOOS (students in MIT 2.680) was the derailment of a mission due to an unnoticed configuration or runtime error.
- Debugging typically involves re-launching with app terminal windows open and analyzing expected vs. observed output.
- Deploying multiple vehicles each with multiple MOOS Apps means a lot of terminal windows are open.
- On a remotely deployed vehicle, one cannot ssh in and see any application terminal output at all!
- Since terminal output is rarely viewable for the above practical reasons, apps are rarely designed with much thought put into their terminal output.

... Introducing AppCasting in MOOS



Without AppCasting



The screenshot displays the pMarineViewer (MIT Version 13.1) interface. The main window shows a satellite map of a coastal area with a grid overlay. A red line indicates a path or boundary, with a point labeled 'alpha (DRIVE)'. The interface includes a menu bar (File, BackView, GeoAttr, Vehicles, AppCasting, MOOS-Scope, Mouse-Context, Action) and a toolbar.

The left sidebar contains a list of MOOSDB entries and their configurations:

- MOOSDB as MOOSName "MOOSDB"**: Lists various logging tasks for "pLogger" such as "IVPHELM_CREATE_CPU", "IVPHELM_IPF_CNT", "IVPHELM_ITER", "IVPHELM_LOOP_CPU", "IVPHELM_STATEVARS", and "NAV_SPEED_ALT".
- pLogger as MOOSName "pLogger"**: Shows wildcard logging configurations for the tasks listed above.
- uSimMarine as MOOSName "uSimMarine"**: Displays thrust maps and acceleration/deceleration values.
- pMarinePID as MOOSName "pMarinePID"**: Shows PID control parameters like PID_REPORT, PID_COURSE, and PID_SPEED.
- pHelmivP as MOOSName "pHelmivP"**: Shows view point and segment information.
- pMarineViewer as MOOSName "pMarineViewer"**: Shows the current instance status.
- uProcessWatch as MOOSName "uProcessWatch"**: Shows event monitoring.
- pNodeReporter as MOOSName "pNodeReporter"**: Shows node report summary and posted reports.
- uFldScope as MOOSName "uFldScope"**: Shows field of view parameters.
- pEchoVar as MOOSName "pEchoVar"**: Shows echo variable settings.

The bottom control panel for the 'alpha' node includes the following fields:

- VName: alpha
- X(m): 60.1
- Lat: 43.824485
- Spd: 2.0
- Dep(m): 0.0
- Time: 64.8
- RETURN: F
- DEPLOY
- VType: kayak
- Y(m): -91.5
- Lon: -70.329635
- Hdg: 180.0
- Age(s): 0.03
- Warp: 1
- RETURN: T
- Variable: RETURN
- Tm: 2.00
- Value: false

The bottom-most section shows a table of flips:

Key	Hits	Source	Dest	Src Sep	Dest Sep	Filter	Old Field	New Field
1	128	NODE_REPORT_LOCAL	FOOBAR	,	#	type:kayak	X	xpos
1	128	NODE_REPORT_LOCAL	FOOBAR	,	#	type:kayak	Y	ypos



With AppCasting



pMarineViewer (MIT Version 13.1)

File BackView GeoAttr Vehicles AppCasting MOOS-Scope Mouse-Context Action

Node	AC	CW	RW	App	AC	CW	RW
alpha	85	0	0	uSimMarine	19	0	0
				pHelmIvP	50	0	0
				pMarineViewer	4	0	0
				uProcessWatch	3	0	0
				pNodeReporter	3	0	0
				uFldScope	3	0	0
				pEchoVar	3	0	0

```

=====
pHelmIvP alpha                                0/0 (179)
=====
Helm Iteration: 122
IvP Functions: 1
Mode(s):
SolveTime: 0.00 (max=0.00)
CreateTime: 0.01 (max=0.01)
LoopTime: 0.01 (max=0.01)
Halted: false (0 warnings)
Helm Decision: [speed,0,4,21] [course,0,359,360]
speed = 2
course = 113
Behaviors Active: ----- (1)
waypt_survey [30.41] (pwt=100) (pcs=6) (cpu=0.14) (upd=0/0)
Behaviors Running: ----- (0)
Behaviors Idle: ----- (1)
waypt_return[always]
Behaviors Completed: ----- (0)

Variable      Behavior      Time  Iter  Value
-----
BHV_STATUS    waypt_return  14.59  1     name=waypt_return,pc=RETURN..rue,stat
CYCLE_INDEX   waypt_survey  14.59  1     0
VIEW_POINT    waypt_survey  14.59  1     x=60,y=-40,active=false,lab.ex_color
VIEW_SEGLIST  waypt_survey  14.59  1     pts={60,-40:60,-160:150,-16..,vertex_
WPT_INDEX     waypt_survey  14.59  1     0
WPT_STAT      waypt_survey  45.00  122   vname=alpha,behavior-name=w..0/0,cycl

=====
Most Recent Events (1):
=====
[14.59]: var=MOOS_MANUAL_OVERRIDE:matter=true, skew=0.22
=====

```

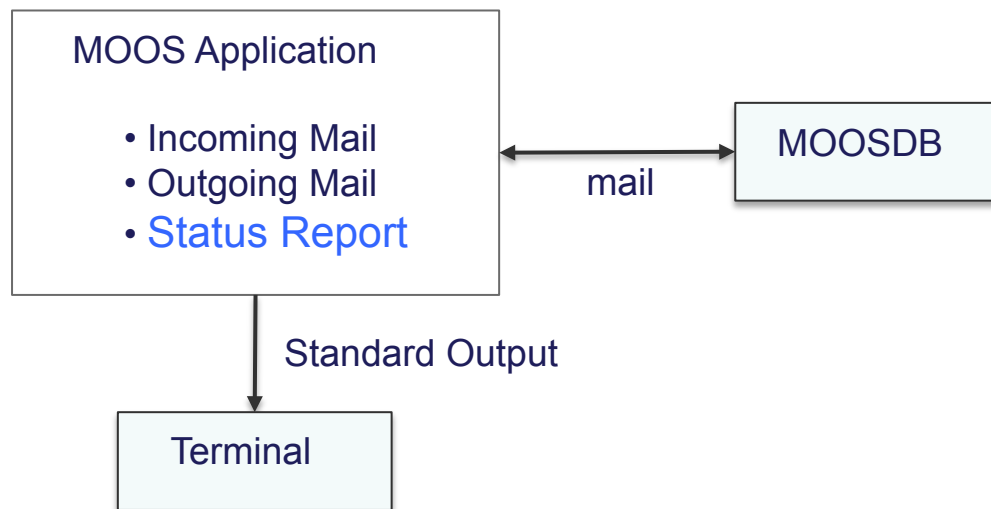
VName: X(m): Lat: Spd: Dep(m): Time: RETURN:F

VType: Y(m): Lon: Hdg: Age(s): Warp:

Variable: Tm: Value:

MOOS Application I/O

- A typical MOOS application interacts by way of mail and the MOOSDB.



- Most applications also produce debugging/status info to the terminal.
- Often this format is an afterthought.
- Often this content is out of sight, if a terminal is not open.

Typical Terminal Output

Typical terminal output of a MOOSApp will show:

- Startup summary and health status,
- A simple heart beat character or other simple health indicator.

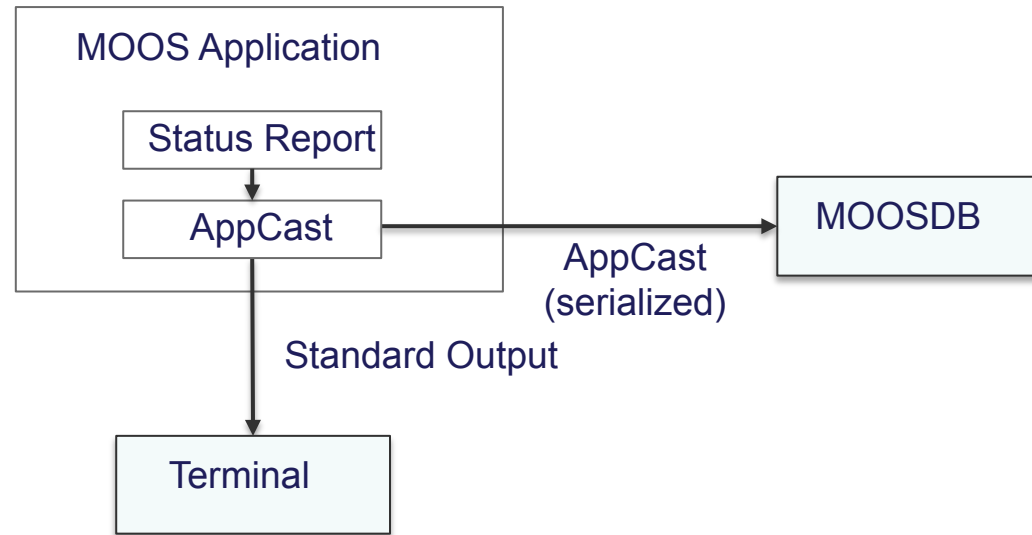
pLogger

```
Terminal — pLogger — 87x22 — 84
pLogger
*****
*                                     *
*      This is MOOS Client            *
*      c. P Newman 2001              *
*                                     *
*****

-----MOOS CONNECT-----
contacting a MOOS server localhost:9000 - try 00001
Contact Made
Handshaking as "pLogger"
Handshaking Complete
Invoking User OnConnect() callback...ok

Warning:
  neither "::GlobalLogPath" or "Path" are specified
  Will Log to ./
pLogger is Running:
  AppTick   @ 4.0 Hz
  CommsTick @ 4 Hz
```

Introducing AppCasting



An AppCast-Enabled MOOS App:

- Generates an AppCast representing its status report.
- The AppCast is sent to the terminal standard output.
(From the user's perspective it looks like any other MOOS application.)
- The AppCast is also serialized and sent to the MOOSDB.

An Example AppCast

From the uProcessWatch MOOSApp

```

=====
uProcessWatch henry (160)
=====
Summary: All Present

Antler List: pBasicContactMgr, pHelmIvP, pHostInfo, pLogger, pMarinePID
             pNodeReporter, pShare, uFldMessageHandler, uFldNodeBroker
             uSimMarine, uXMS

ProcName      Watch Reason  Status
-----
pBasicContactMgr  ANT      DB  OK
pHelmIvP          ANT WATCH DB  OK
pHostInfo         ANT      DB  OK
pLogger          ANT      DB  OK
pMarinePID       ANT WATCH DB  OK
pNodeReporter    ANT WATCH DB  OK
pShare           ANT      DB  OK
uFldMessageHandler ANT      DB  OK
uFldNodeBroker   ANT      DB  OK
uSimMarine       ANT WATCH DB  OK

=====
Most Recent Events (8):
=====
[4.01]: Resurrected: [uFldMessageHandler]
[2.01]: PROC_WATCH_EVENT: Process [uFldMessageHandler] is missing.
[0.00]: Noted to be present: [pShare]
[0.00]: Noted to be present: [pLogger]
[0.00]: Noted to be present: [pBasicContactMgr]
[0.00]: Noted to be present: [pHostInfo]
[0.00]: Noted to be present: [uFldNodeBroker]
[0.00]: Noted to be present: [uFldMessageHandler]

```

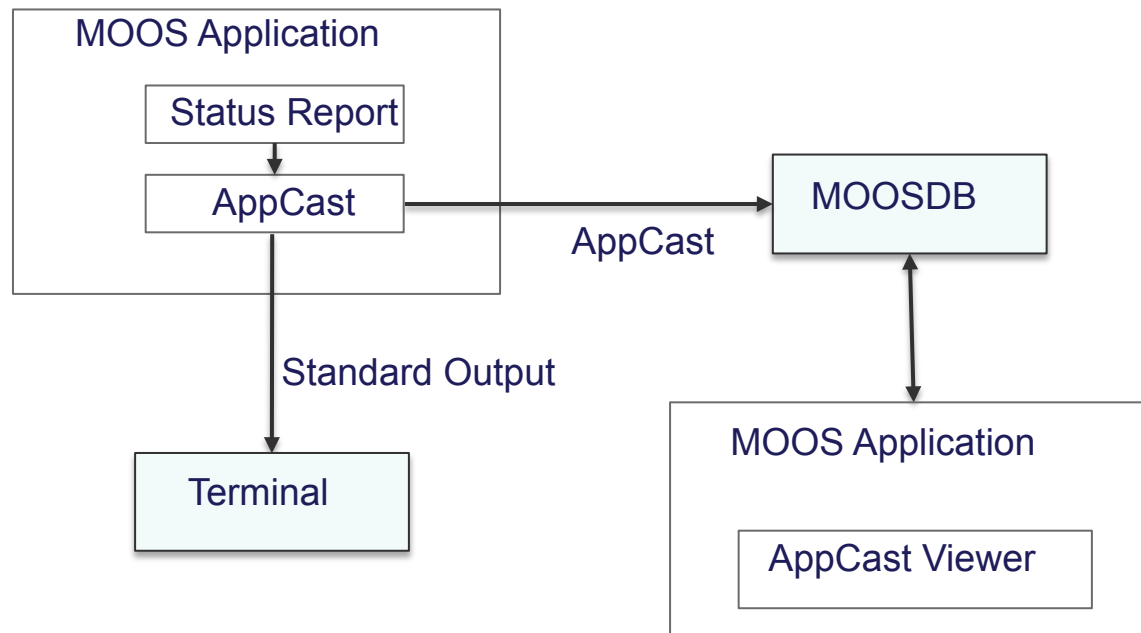
List of
Strings

List of
Events
(Limited)

Application
Iteration
Counter

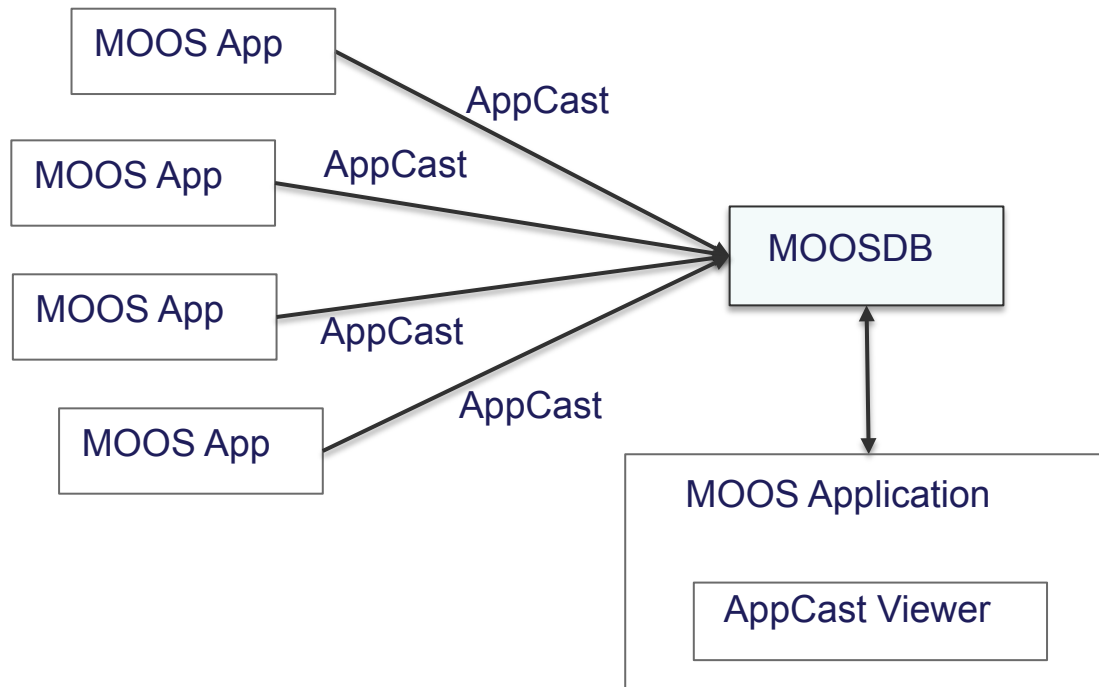
AppCast Viewing

- A separate MOOS utility application may be run to view AppCasts from any AppCast-enabled application.



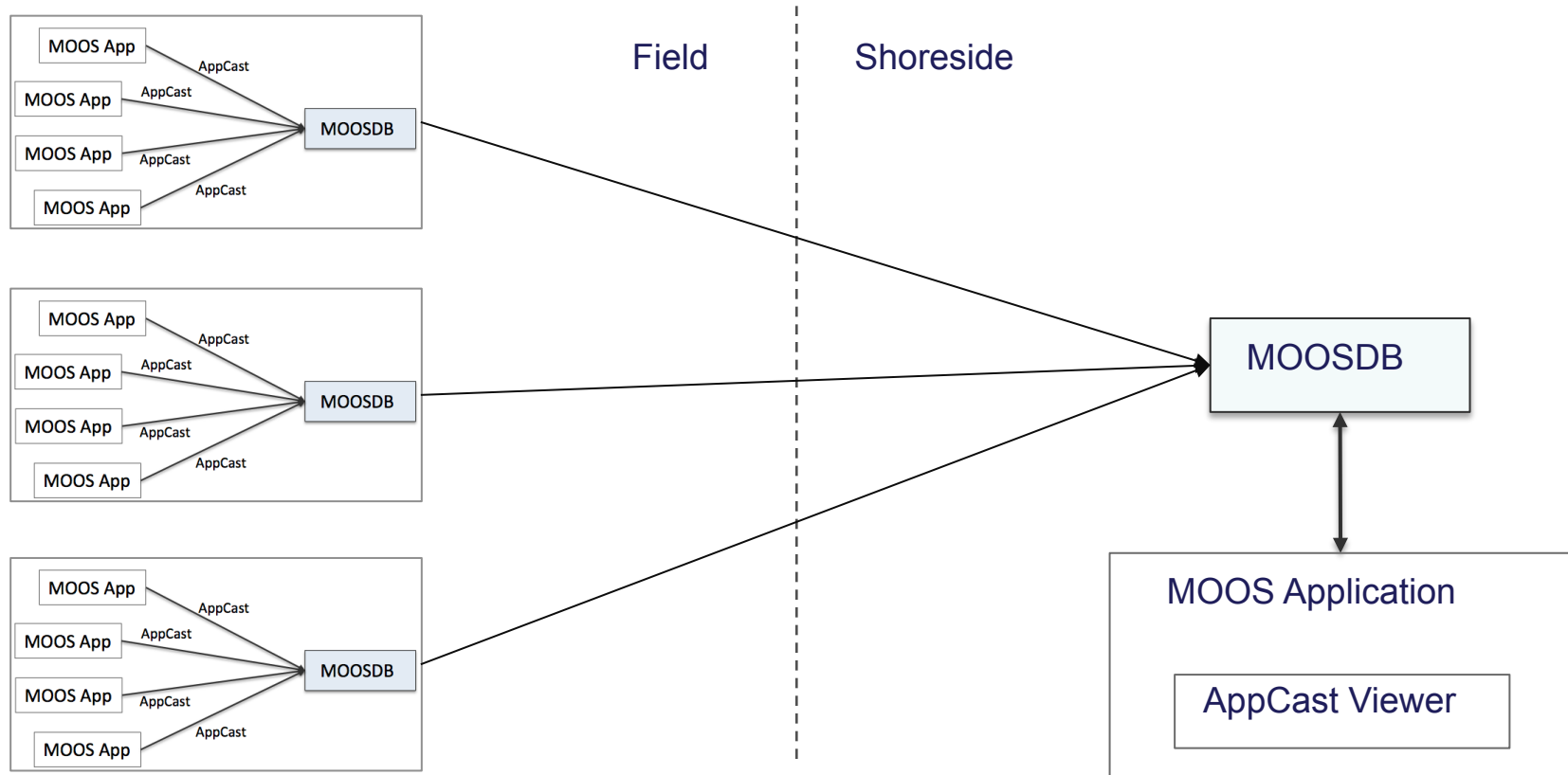
- Now a user can see application output even if an app initially was sending terminal output to `/dev/null`.

AppCast Viewing



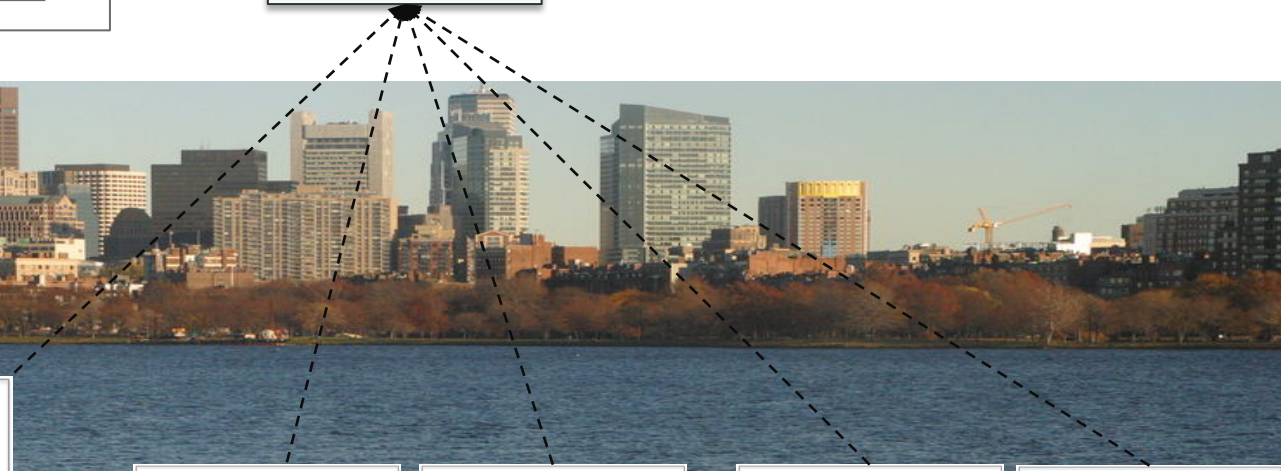
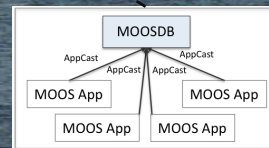
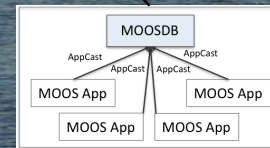
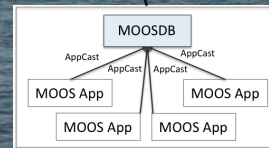
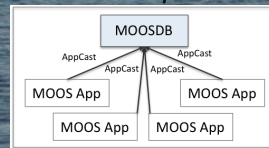
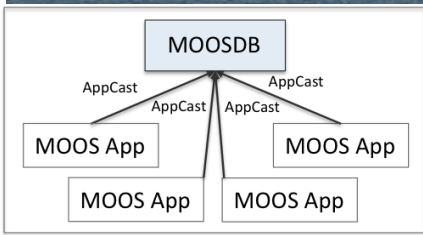
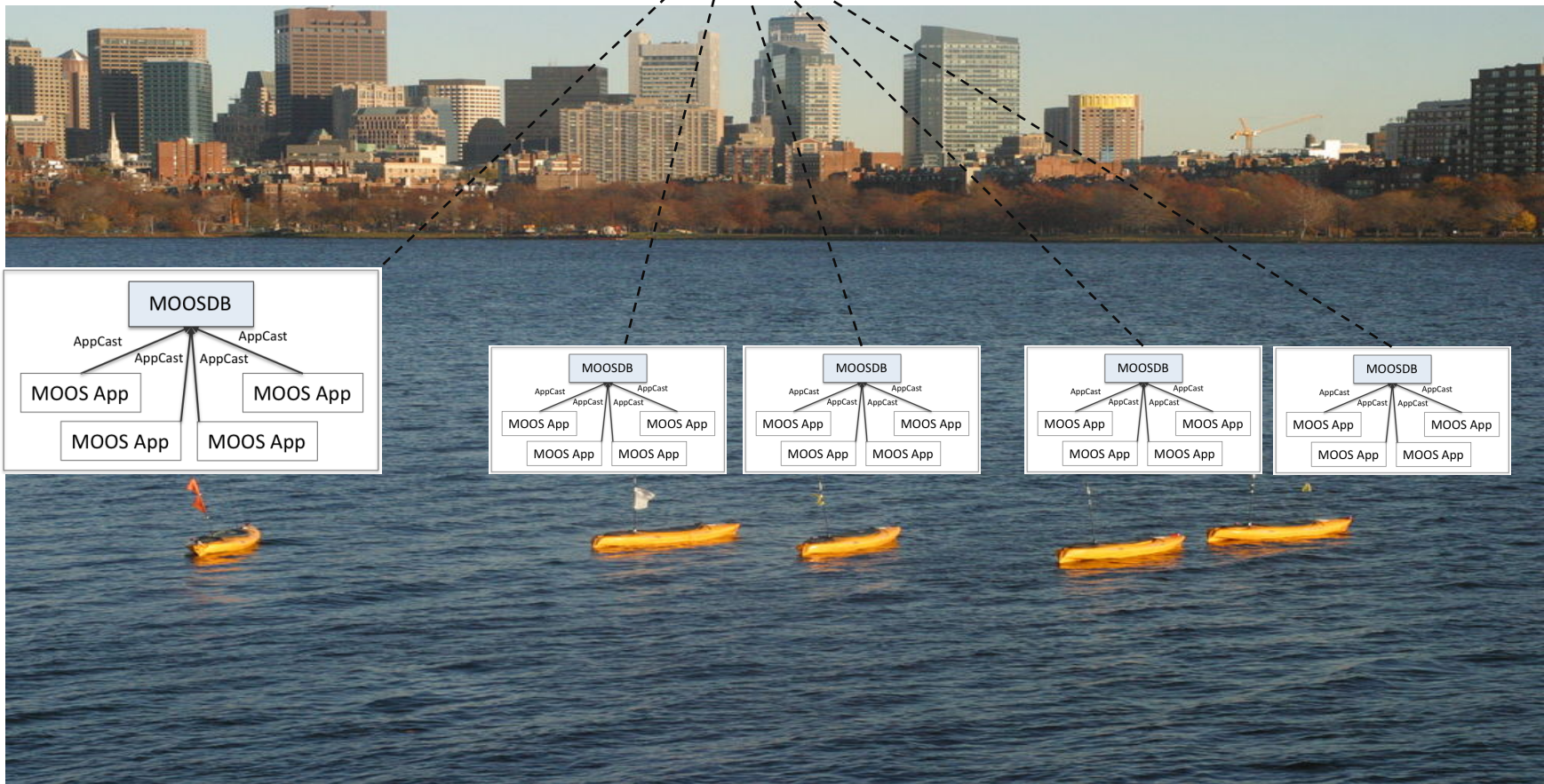
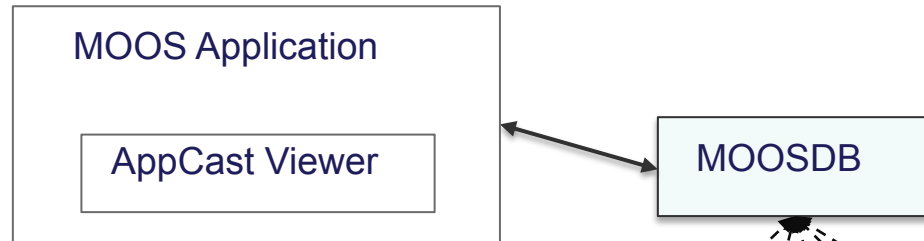
- The AppCast viewer may “connect” to multiple applications.
- The AppCast viewer can switch between “channels”.
- The AppCast viewer brings Config and RunTime alerts to the user’s attention even when not monitoring that channel.

AppCast Viewing



- The AppCast viewer may connect to multiple vehicles, diving down to the vehicle and application it selects.

AppCast Viewing



AppCast Viewers

What does an AppCast Viewer do?

- Sends AppCast requests to clients.
- Renders received AppCasts.
- Allows the user to select/switch between different MOOSApps and vehicles

Currently there are three AppCast Viewer applications:

(1) uMAC

```

Terminal — uMAC — 72x35 — #2
pAntier uMAC
-----
uMAC 9842: Nodes (7) (5)
uProcessWatch archie (6627)
Summary: All Present
Antier List: pBasicContactMgr, pHelmIvP, pHostInfo, pLogger, pMarinePID
             pNodeReporter, pShare, uFlidMessageHandler, uFlidNodeBroker
             uSimMarine
-----
ProcName      Watch Reason  Status
-----
pBasicContactMgr ANT WATCH DB OK
pHelmIvP      ANT WATCH DB OK
pHostInfo     ANT WATCH DB OK
pLogger       ANT WATCH DB OK
pMarinePID    ANT WATCH DB OK
pNodeReporter ANT WATCH DB OK
pShare        ANT WATCH DB OK
uFlidMessageHandler ANT WATCH DB OK
uFlidNodeBroker ANT WATCH DB OK
uSimMarine    ANT WATCH DB OK
-----
Most Recent Events (8):
[3.02]: Resurrected: [pHostInfo]
[3.02]: Resurrected: [uFlidNodeBroker]
[3.02]: Resurrected: [uFlidMessageHandler]
[1.01]: Resurrected: [pBasicContactMgr]
[1.01]: Resurrected: [pShare]
[0.01]: PROC_WATCH_EVENT: Process [pBasicContactMgr] is missing.
[0.01]: PROC_WATCH_EVENT: Process [pShare] is missing.
[0.01]: PROC_WATCH_EVENT: Process [pHostInfo] is missing.
    
```

Terminal
(good for ssh'ing into a remote vehicle)

(2) uMACView

```

uMACView
-----
Node AppCasting
-----
shoreside 79 0 0 uFlidMessageHandler 4 0 0
archie 16 0 0 uSimMarine 2 0 0
david 44 0 0 pHelmIvP 2 0 0
archie 16 0 0 pNodeReporter 2 0 0
charlie 16 0 0 uProcessWatch 28 0 0
betty 16 0 0 pBasicContactMgr 2 0 0
prey 12 0 0 uFlidNodeBroker 2 0 0
prey 12 0 0 pHostInfo 2 0 0
-----
uProcessWatch david (506)
Summary: All Present
Antier List: pBasicContactMgr, pHelmIvP, pHostInfo, pLogger, pMarinePID
             pNodeReporter, pShare, uFlidMessageHandler, uFlidNodeBroker
             uSimMarine
-----
ProcName      Watch Reason  Status
-----
pBasicContactMgr ANT WATCH DB OK
pHelmIvP      ANT WATCH DB OK
pHostInfo     ANT WATCH DB OK
pLogger       ANT WATCH DB OK
pMarinePID    ANT WATCH DB OK
pNodeReporter ANT WATCH DB OK
pShare        ANT WATCH DB OK
uFlidMessageHandler ANT WATCH DB OK
uFlidNodeBroker ANT WATCH DB OK
uSimMarine    ANT WATCH DB OK
-----
Most Recent Events (8):
[3.02]: Resurrected: [pHostInfo]
[3.02]: Resurrected: [uFlidNodeBroker]
[3.02]: Resurrected: [uFlidMessageHandler]
[1.01]: Resurrected: [pBasicContactMgr]
[1.01]: Resurrected: [pShare]
[0.01]: PROC_WATCH_EVENT: Process [pShare] is missing.
[0.01]: PROC_WATCH_EVENT: Process [pHostInfo] is missing.
[0.01]: PROC_WATCH_EVENT: Process [uFlidNodeBroker] is missing.
[0.01]: PROC_WATCH_EVENT: Process [uFlidMessageHandler] is missing.
    
```

GUI (f+k)

(3) pMarineViewer

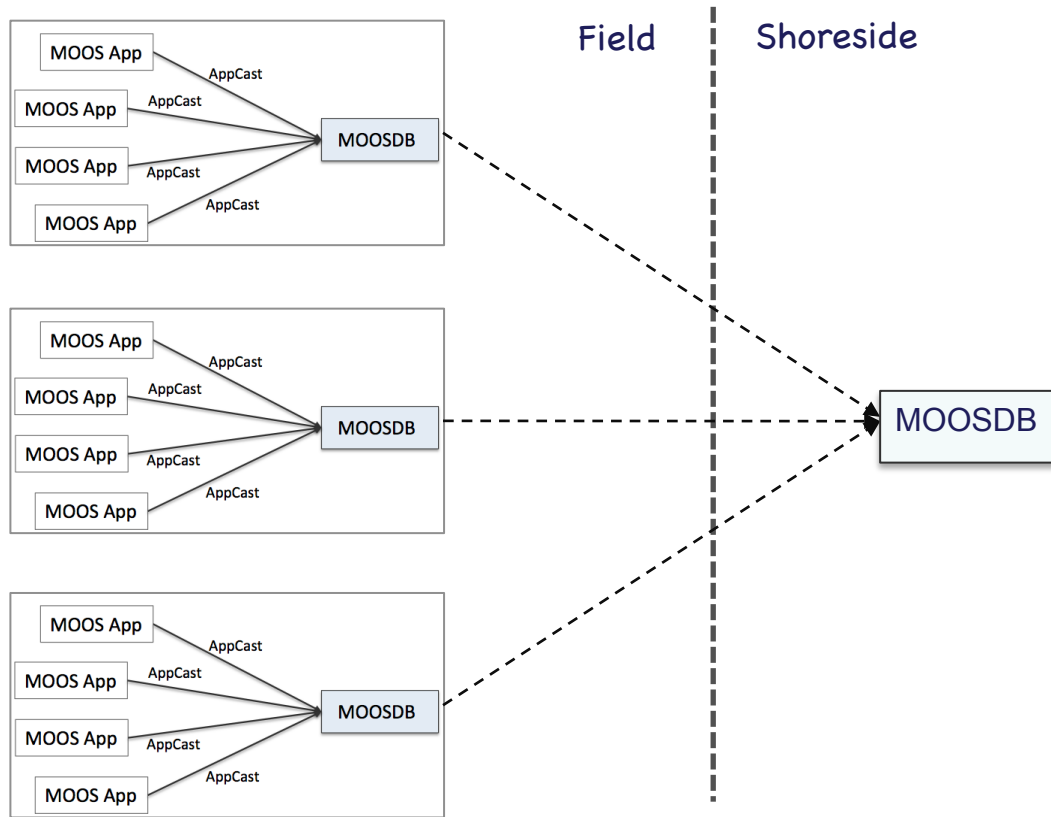
```

pMarineViewer (MIT Version 12.10)
-----
Node AC CW RW App AC CW RW
shoreside 1539 0 0 uTimerScript 1494 0 0
archie 67 0 0 pMarineViewer 11 0 0
prey 50 0 0 uFlidNodeBroker 10 0 0
betty 68 0 0 uFlidNodeComms 11 0 0
charlie 68 0 0 pHostInfo 10 0 0
david 1229 0 0 uMACView 3 0 0
ernie 62 0 0
-----
uTimerScript shoreside (1539)
Current Script Information:
Element: 0(16)
MinIn: 0
Delay Start: 1
Delay Reset: 0
Search: false
ConditionOf: true
-----
RandomVar Type Min Max Parameters
-----
01 uniform -200 0
02 uniform -200 100
03 uniform -100 100
04 uniform -400 -200
05 uniform 100 300
06 uniform -400 -200
07 uniform 300 500
08 uniform -200 100
09 uniform 100 300
10 uniform -150 300
11 uniform -150 0
-----
#?not #?loc #?total #?local Variable/Var
-----
07 7 3601.84 400.14 UP_LOITER_3 = center_essip=244,-41
08 8 3601.84 400.14 UP_LOITER_4 = center_essip=287,-384
09 9 3601.84 400.14 UP_LOITER_5 = center_essip=34,-141
70 10 3801.79 700.10 UP_LOITER_1 = center_essip=104,-236
71 11 3801.79 700.10 UP_LOITER_2 = center_essip=205,-166
72 12 3801.79 700.10 UP_LOITER_3 = center_essip=105,-279
73 13 3801.79 700.10 UP_LOITER_4 = center_essip=39,-185
74 14 3801.79 700.10 UP_LOITER_5 = center_essip=344,-121
75 uniform -150 0
-----
Most Recent Events (3):
[3200.75]: Script [NoInit_Marv1, DelayStart=0.0, DelayReset=0.0
[3400.37]: Script [NoInit_Marv1, DelayStart=0.0, DelayReset=0.0
[0.01]: Script [NoInit_Marv1, DelayStart=0.0, DelayReset=0.0
    
```

GUI (f+k)

AppCast Viewing

with the uMACView Tool



Select the Vehicle/Node

Select the MOOSApp

Node	AC	CW	RW	App	AC	CW	RW
shoreside	79	0	0	uFldMessageHandler	4	0	0
ernie	16	0	0	uSimMarine	2	0	0
david	44	0	0	pHelmIvP	2	0	0
archie	16	0	0	pNodeReporter	2	0	0
charlie	16	0	0	uProcessWatch	28	0	0
betty	16	0	0	pBasicContactMgr	2	0	0
prey	12	0	0	uFldNodeBroker	2	0	0
				pHostInfo	2	0	0

```

uProcessWatch david (506)
Summary: All Present
Antler List: pBasicContactMgr, pHelmIvP, pHostInfo, pLogger, pMarinePID,
             pNodeReporter, pShare, uFldMessageHandler, uFldNodeBroker,
             uSimMarine

ProcName      Watch Reason  Status
-----
pBasicContactMgr  ANT          DB    OK
pHelmIvP         ANT WATCH    DB    OK
pHostInfo        ANT          DB    OK
pLogger          ANT WATCH    DB    OK
pMarinePID       ANT WATCH    DB    OK
pNodeReporter    ANT WATCH    DB    OK
pShare           ANT          DB    OK
uFldMessageHandler ANT          DB    OK
uFldNodeBroker   ANT          DB    OK
uSimMarine       ANT WATCH    DB    OK
  
```

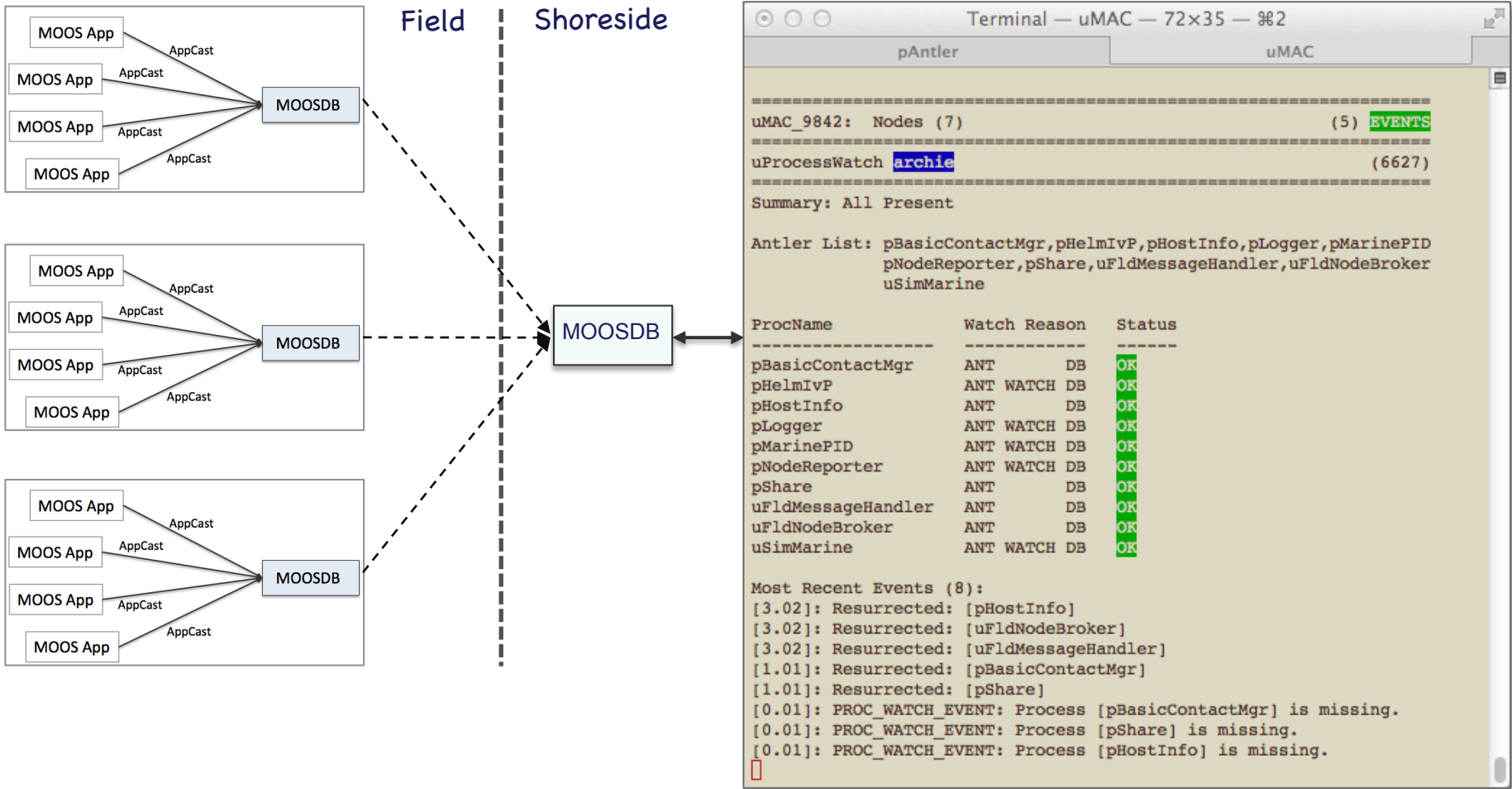
```

Most Recent Events (8):
-----
[3.02]: Resurrected: [pHostInfo]
[3.02]: Resurrected: [uFldNodeBroker]
[3.02]: Resurrected: [uFldMessageHandler]
[1.01]: Resurrected: [pShare]
[0.01]: PROC_WATCH_EVENT: Process [pShare] is missing.
[0.01]: PROC_WATCH_EVENT: Process [pHostInfo] is missing.
[0.01]: PROC_WATCH_EVENT: Process [uFldNodeBroker] is missing.
[0.01]: PROC_WATCH_EVENT: Process [uFldMessageHandler] is missing.
  
```

View the AppCast

- uMACView is a stand-alone, GUI-Based Viewer
- Launch from the command-line or w/ pAntler.

AppCast Viewing with uMAC



- Terminal interface provides most of what the GUI tools provide.
- Primary advantage: When a remote vehicle is not sending AppCasts to a shoreside, user can ssh into the vehicle and launch uMAC to debug.

The AppCast Structure

AppCast Config Warnings

- An AppCast is an instance of the AppCast C++ class.
- It contains:

Config warnings:

- Usually created at App startup time.
- Unlimited in quantity.

```

=====
uProcessWatch gilda 1/1 (150)
=====
Configuration Warnings: 1
[1 of 1]: Unhandled config line: foobar=abracadabra

Runtime Warnings: 1
[1]: Process [pNodeReporter] is missing.

Summary: AWOL: pNodeReporter

Antler List: pBasicContactMgr, pHelmIvP, pHostInfo, pLogger, pMarinePID
             pNodeReporter, pShare, uFldMessageHandler, uFldNodeBroker
             uSimMarine, uXMS

ProcName          Watch Reason  Status
-----
pBasicContactMgr  ANT   DB   OK
pHelmIvP          ANT WATCH DB   OK
pHostInfo         ANT   DB   OK
pLogger           ANT   DB   OK
pMarinePID        ANT WATCH DB   OK
pNodeReporter     ANT WATCH DB  MISSING
pShare            ANT   DB   OK
uFldMessageHandler ANT   DB   OK
uFldNodeBroker    ANT   DB   OK
uSimMarine        ANT WATCH DB   OK

=====
Most Recent Events (8):
=====
[120.15]: PROC_WATCH_EVENT: Process [pNodeReporter] is missing.
[0.00]: Noted to be present: [pShare]
[0.00]: Noted to be present: [pLogger]
[0.00]: Noted to be present: [pBasicContactMgr]
[0.00]: Noted to be present: [pHostInfo]
[0.00]: Noted to be present: [uFldNodeBroker]
[0.00]: Noted to be present: [uFldMessageHandler]
[0.00]: Noted to be present: [uSimMarine]

```

Config Warnings

Run Warnings

General messages

Events



The AppCast Structure

AppCast Run Warnings

- An AppCast is an instance of the AppCast C++ class.
- It contains:

Run warnings:

- Created typically well after launch time when something goes wrong.
- Limited in quantity, (don't want the size of an appcast to grow unbounded)
- Provisions are made in AppCast Viewers to ensure RunWarnings come to the user's attention.

```

=====
uProcessWatch gilda 1/1 (150)
=====
Config Warnings: 1
[1 of 1]: Unhandled config line: foobar=abracadabra

Run Warnings: 1
[1]: Process [pNodeReporter] is missing.

Summary: AWOL: pNodeReporter

Antler List: pBasicContactMgr, pHelmIvP, pHostInfo, pLogger, pMarinePID
             pNodeReporter, pShare, uFldMessageHandler, uFldNodeBroker
             uSimMarine, uXMS

ProcName          Watch Reason      Status
-----
pBasicContactMgr  ANT      DB      OK
pHelmIvP          ANT WATCH DB      OK
pHostInfo         ANT      DB      OK
pLogger          ANT      DB      OK
pMarinePID       ANT WATCH DB      OK
pNodeReporter    ANT WATCH DB      MISSING
pShare           ANT      DB      OK
uFldMessageHandler ANT      DB      OK
uFldNodeBroker   ANT      DB      OK
uSimMarine       ANT WATCH DB      OK

=====
Most Recent Events (8):
=====
[120.15]: PROC_WATCH_EVENT: Process [pNodeReporter] is missing.
[0.00]: Noted to be present: [pShare]
[0.00]: Noted to be present: [pLogger]
[0.00]: Noted to be present: [pBasicContactMgr]
[0.00]: Noted to be present: [pHostInfo]
[0.00]: Noted to be present: [uFldNodeBroker]
[0.00]: Noted to be present: [uFldMessageHandler]
[0.00]: Noted to be present: [uSimMarine]

```

Config Warnings

Run Warnings

General messages

Events



The AppCast Structure

AppCast Messages

- An AppCast is an instance of the AppCast C++ class.
- It contains:

Messages:

- Free in format. Up to the user to pick the information and layout.
- Typically “cleared” on each generation of an appcast.
- Just a list of strings.
- Tables, columns etc. done by the user.

```

=====
uProcessWatch gilda 1/1 (150)
=====
Configuration Warnings: 1
[1 of 1]: Unhandled config line: foobar=abracadabra

Runtime Warnings: 1
[1]: Process [pNodeReporter] is missing.

Summary: AWOL: pNodeReporter

Antler List: pBasicContactMgr, pHelmIvP, pHostInfo, pLogger, pMarinePID
             pNodeReporter, pShare, uFldMessageHandler, uFldNodeBroker
             uSimMarine, uXMS

ProcName          Watch Reason      Status
-----
pBasicContactMgr  ANT      DB      OK
pHelmIvP          ANT WATCH DB      OK
pHostInfo         ANT      DB      OK
pLogger          ANT      DB      OK
pMarinePID       ANT WATCH DB      OK
pNodeReporter    ANT WATCH DB      MISSING
pShare           ANT      DB      OK
uFldMessageHandler ANT      DB      OK
uFldNodeBroker   ANT      DB      OK
uSimMarine       ANT WATCH DB      OK

=====
Most Recent Events (8):
=====
[120.15]: PROC_WATCH_EVENT: Process [pNodeReporter] is missing.
[0.00]: Noted to be present: [pShare]
[0.00]: Noted to be present: [pLogger]
[0.00]: Noted to be present: [pBasicContactMgr]
[0.00]: Noted to be present: [pHostInfo]
[0.00]: Noted to be present: [uFldNodeBroker]
[0.00]: Noted to be present: [uFldMessageHandler]
[0.00]: Noted to be present: [uSimMarine]
    
```


The AppCast Structure

AppCast Events

- An AppCast is an instance of the AppCast C++ class.
- It contains:

Events:

- Created typically after launch time when something "notable" happens.
- Limited in quantity. (don't want the size of an appcast to grow unbounded)
- Each event is just a string.

```

=====
uProcessWatch gilda 1/1 (150)
=====
Config Warnings: 1
[1 of 1]: Unhandled config line: foobar=abracadabra

Run Warnings: 1
[1]: Process [pNodeReporter] is missing.

Summary: AWOL: pNodeReporter

Antler List: pBasicContactMgr, pHelmIvP, pHostInfo, pLogger, pMarinePID
             pNodeReporter, pShare, uFldMessageHandler, uFldNodeBroker
             uSimMarine, uXMS

ProcName          Watch Reason  Status
-----
pBasicContactMgr  ANT   DB   OK
pHelmIvP          ANT WATCH DB   OK
pHostInfo         ANT   DB   OK
pLogger          ANT   DB   OK
pMarinePID       ANT WATCH DB   OK
pNodeReporter    ANT WATCH DB  MISSING
pShare           ANT   DB   OK
uFldMessageHandler ANT   DB   OK
uFldNodeBroker   ANT   DB   OK
uSimMarine       ANT WATCH DB   OK

=====
Most Recent Events (8):
=====
[120.15]: PROC_WATCH_EVENT: Process [pNodeReporter] is missing.
[0.00]: Noted to be present: [pShare]
[0.00]: Noted to be present: [pLogger]
[0.00]: Noted to be present: [pBasicContactMgr]
[0.00]: Noted to be present: [pHostInfo]
[0.00]: Noted to be present: [uFldNodeBroker]
[0.00]: Noted to be present: [uFldMessageHandler]
[0.00]: Noted to be present: [uSimMarine]

```

Config Warnings

Run Warnings

General messages

Events





How do you make an
“AppCast-Enabled”
MOOS application?

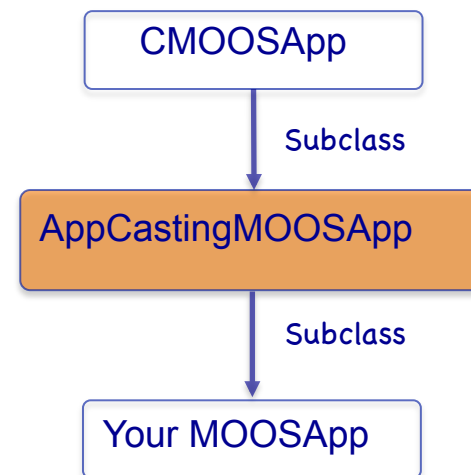
On-Demand AppCasting

To implement on-demand appcasting, a few things need to be done in each application.

- Apps must register for `APPCAST_REQ` mail.
An AppCast request will renew a token for some number of seconds
Until the token expires, the app generates an appcast repeatedly.
- Even while appcasting, the app only generates an AppCast every N secs.
The app keeps track of the last real-time appcast generation.
- Each app handles a config setting indicating whether an xterm is open.
This setting is a global variable in the `.moos` config file.

So a new generic
"AppCastingMOOSApp"
class is used:

Minimizes boilerplate in
individual apps.





Using the AppCastingMOOSApp Superclass

Six Steps



Step 1: Subclass the AppCastingMOOSApp Superclass

Step 2: Invoke two superclass methods in your Iterate()

Step 3: Invoke a superclass method when you register variables.

Step 4: Invoke a superclass method during OnNewMail().

Step 5: Invoke a superclass method during OnStartUp()

Trivial, 1-2 line changes
in each case

Step 6: Implement your buildReport() function.

This is where you get
to be creative about
what your app reports.



Using the AppCastingMOOSApp Superclass

Your Class Definition



Step 1: Subclass the AppCastingMOOSApp Superclass

```
0 #include "MOOS/libMOOS/Thirdparty/AppCasting/AppCastingMOOSApp.h"
1 class YourMOOSApp : public AppCastingMOOSApp
2 {
3     // All your normal class declaration stuff
4
5     bool buildReport();
6 };
```

The `buildReport()` function is a virtual function in the superclass. It is where you can do the work of constructing an AppCast.



Using the AppCastingMOOSApp Superclass

Modifying Your Iterate() and Registrations



Step 2: Invoke two superclass methods in your Iterate()

```
0 bool YourMOOSApp::Iterate()  
1 {  
2     AppCastingMOOSApp::Iterate();  
3  
4     // Do all your normal Iterate stuff  
5  
6     AppCastingMOOSApp::PostReport();  
7     return(true);  
8 };
```

Updates the current MOOSTime, and # of iterations.

Determines if an AppCast is warranted, and invokes `buildReport()` if so.

Step 3: Invoke a superclass method when you register variables.

```
0 void YourMOOSApp::registerVariables()  
1 {  
2     AppCastingMOOSApp::RegisterVariables();  
3  
4     // Do all your other registrations  
5 }
```

The superclass will register for `APPCAST_REQ`, indicating another app, like `uMAC`, is interested in appcasts from this app.



Using the AppCastingMOOSApp Superclass

Modifying Your OnNewMail() and OnStartup()



Step 4: Invoke a superclass method when you handle mail.

```
0 bool YourMOOSApp::OnNewMail(MOOSMSG_LIST &NewMail)
1 {
2     AppCastingMOOSApp::OnNewMail(NewMail);
3
4     // Do all your other normal mail handling.
5 }
```

The superclass will handle the `APPCAST_REQ` mail.

Step 5: Invoke a superclass method during OnStartup()

```
0 void YourMOOSApp::OnStartup()
1 {
2     AppCastingMOOSApp::OnStartup();
3
4     // Do all your other startup stuff
5 }
```

The superclass will register for `APPCAST_REQ`, indicating another app, like uMAC, is interested in appcasts from this app.



END