

**MIT 2.680**  
**UNMANNED MARINE VEHICLE AUTONOMY,**  
**SENSING, AND COMMUNICATIONS**

## Lecture 11- Inter-Vehicle Messaging

April 4th, 2023



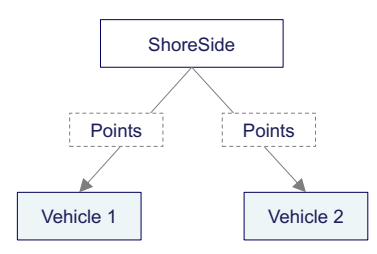
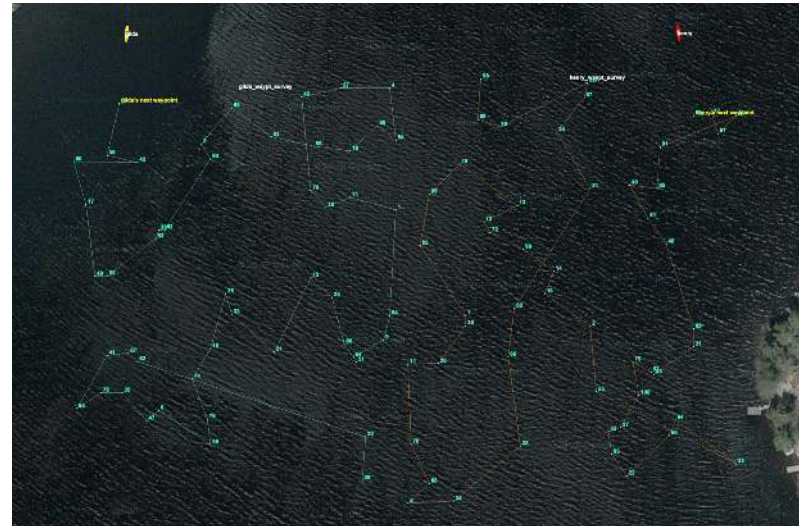
Web: <http://oceanai.mit.edu/2.680>  
 Email:  
 Mike Benjamin, [mikerb@mit.edu](mailto:mikerb@mit.edu)


MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”  Photo by Arjan Vermeij GLINT '09

1


**Are these vehicles collaborating?**

**Mission Type 1:**  
 Distributed Travelling Salesman





MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging” 

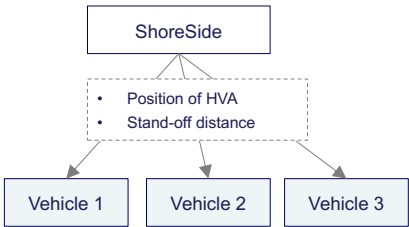
2



## Are these vehicles collaborating?




**Mission Type 2:**  
Protection of High Value Asset (HVA)



```

graph TD
    ShoreSide[ShoreSide] --> Info[Position of HVA  
Stand-off distance]
    Info --> V1[Vehicle 1]
    Info --> V2[Vehicle 2]
    Info --> V3[Vehicle 3]
        
```




Two modes shown:


- Sharing local position information for collision avoidance only.
- Inter-vehicle messaging for achieving spacing parity

MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”

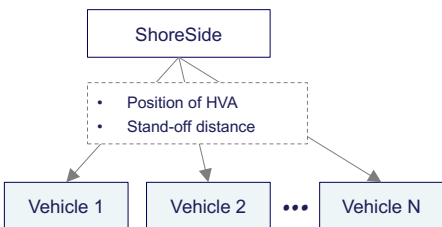
3



## Are these vehicles collaborating?




**Mission Type 3:**  
Protection of High Value Asset (HVA)  
Plus Decentralized Interdiction



```

graph TD
    ShoreSide[ShoreSide] --> Info[Position of HVA  
Stand-off distance]
    Info --> V1[Vehicle 1]
    Info --> V2[Vehicle 2]
    Info --> VN[Vehicle N]
        
```




Two types of inter-vehicle communication:


- Position information for collision avoidance and for achieving position parity (protocol based)
- Inter-vehicle auctions for decentralized decision-making for handling “intruders”

MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”

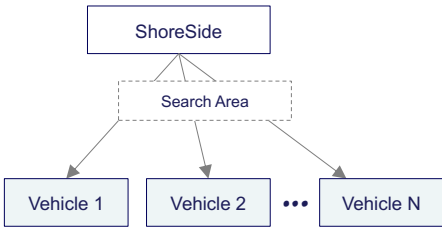
4



## Are these vehicles collaborating?



**Mission Type 4:**  
Distributed Area Coverage/Survey

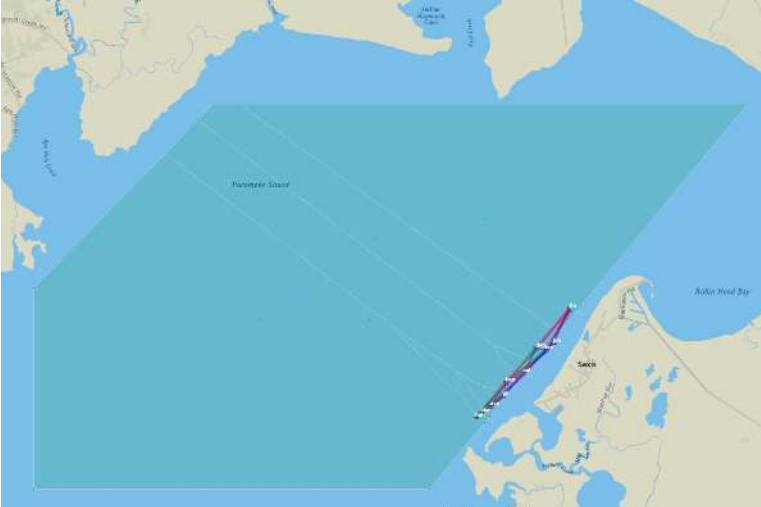


```

graph TD
    ShoreSide[ShoreSide] --> SearchArea[Search Area]
    SearchArea --> Vehicle1[Vehicle 1]
    SearchArea --> Vehicle2[Vehicle 2]
    SearchArea --> VehicleN[Vehicle N]


```

- Position information for collision avoidance and for achieving region parity (protocol based)
- Each vehicle drives toward the centroid of its own locally understood Voronoi region. The region evolves as information arrives.




MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”

5



## Inter-Vehicle Communications



- Methods of Inter-vehicle communications
- Limits on inter-vehicle communications
- Simulating inter-vehicle communications on a network
- uFieldNodeComms
- uFieldMessageHandler
- Indicators of successful messaging
- Debugging dropped messages
- Lab Preview

MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”

6

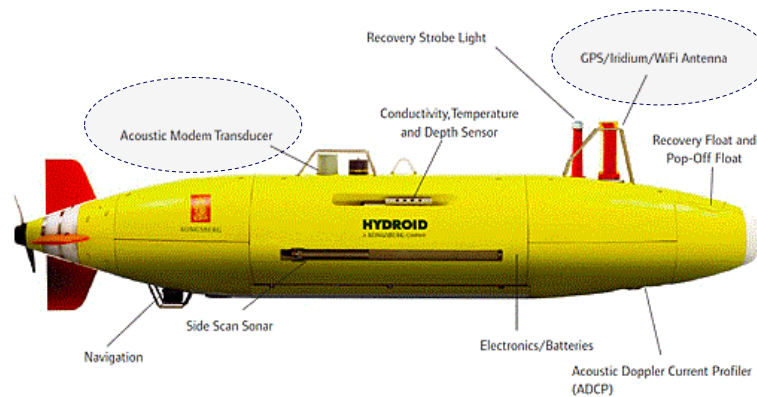


## Inter-Vehicle Communications



### How do two vehicles / robots / machines talk to each other?

- Depends on how far away they are from each other
- Depends on what is between them (air, water, or both)
- Messages may be sent directly between robots, or over a network



MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”



7

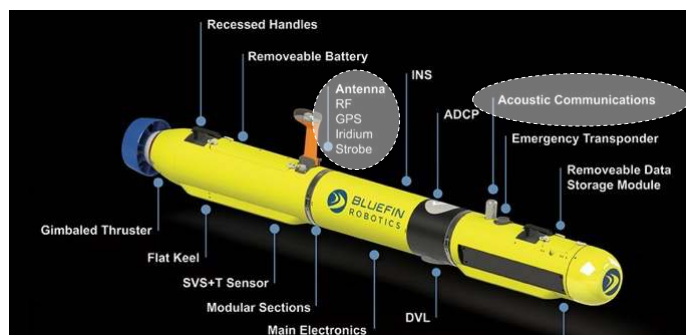


## Inter-Vehicle Communications



### How do two vehicles / robots / machines talk to each other?

- Depends on how far away they are from each other
- Depends on what is between them (air, water, or both)
- Messages may be sent directly between robots, or over a network




MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”




8





## Inter-Vehicle Communications




**How do two vehicles / robots / machines talk to each other?**


- Depends on how far away they are from each other
- Depends on what is between them (air, water, or both)
- Messages may be sent directly between robots, or over a network

**Assumptions for now:**


- All robots have a unique name with known network address
- Messages may be sent to an individual robot, all robots, or a group of robots
- A message may or may not be received by the target robot
- No acknowledgement is built into the messaging structure (although you can do this yourself)
- A message may be range-limited (the receiving robot is too far away)
- A message may be band-width limited (the message has a maximum length)
- A message may be frequency limited (there may be a minimum wait time between messages)
- A message may have latency (it's arrival time at the receiving robot is not guaranteed)

MIT 2.680 Spring 2023 – Marine Autonomy – "Lecture 11: Inter-Vehicle Messaging"


9



## Inter-Vehicle Communications



**How do two vehicles / robots / machines talk to each other?**


- Depends on how far away they are from each other
- Depends on what is between them (air, water, or both)
- Messages may be sent directly between robots, or over a network


**Assumptions for now:**

- All robots have a unique name with known network address
- Messages may be sent to an individual robot, all robots, or a group of robots
- A message may or may not be received by the target robot
- No acknowledgement is built into the messaging structure (although you can do this yourself)
- A message may be range-limited (the receiving robot is too far away)
- A message may be band-width limited (the message has a maximum length)
- A message may be frequency limited (there may be a minimum wait time between messages)
- A message may have latency (it's arrival time at the receiving robot is not guaranteed)

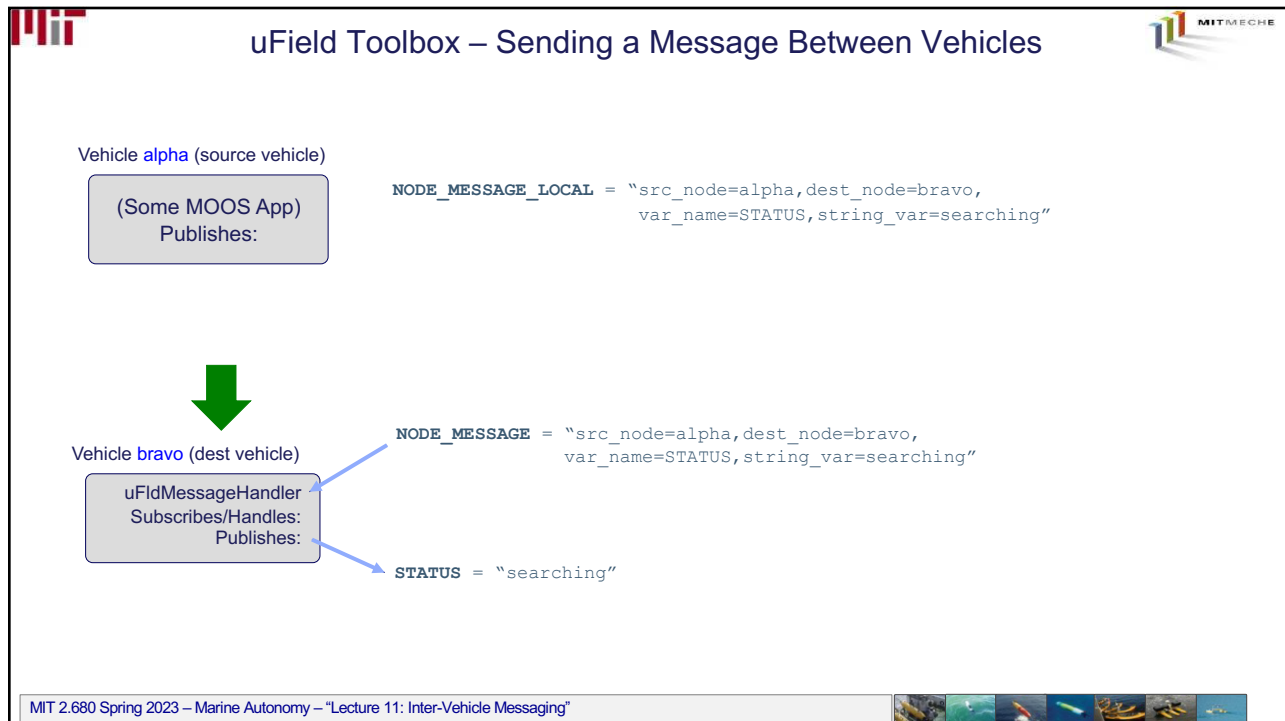
**Focus of 2.680**

How can we use robot mobility and autonomy to overcome these limitations?

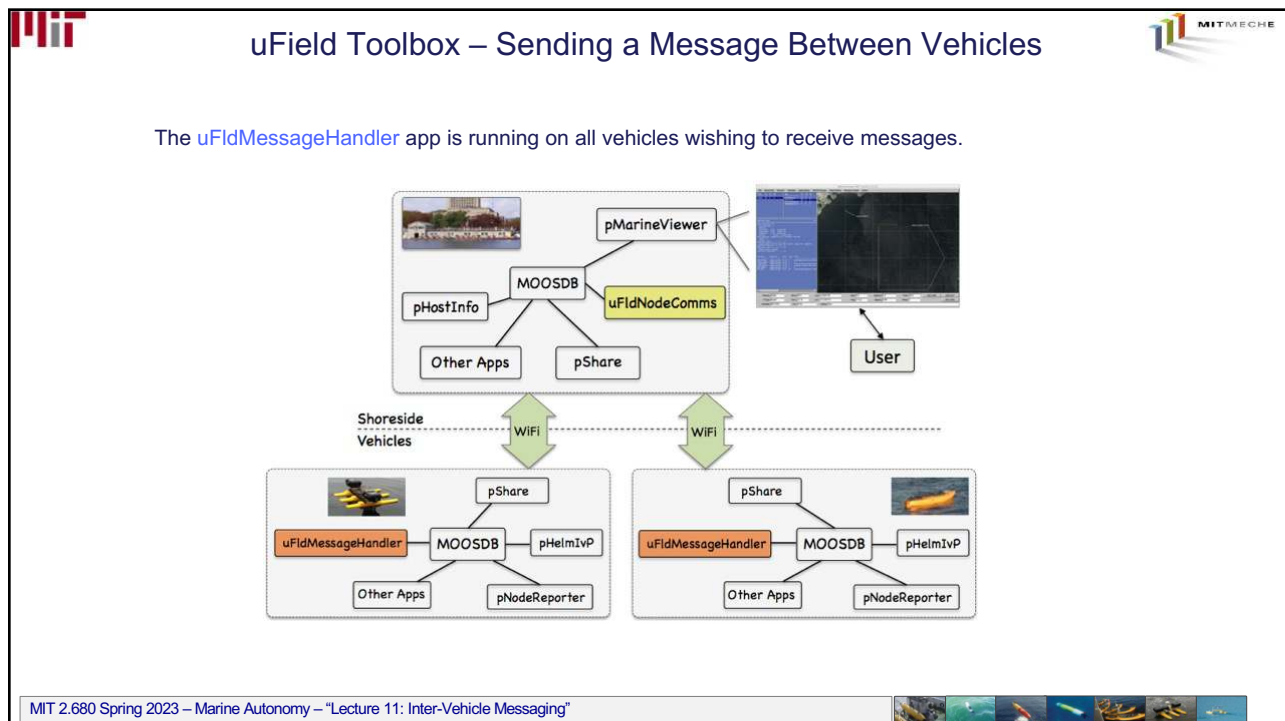


MIT 2.680 Spring 2023 – Marine Autonomy – "Lecture 11: Inter-Vehicle Messaging"


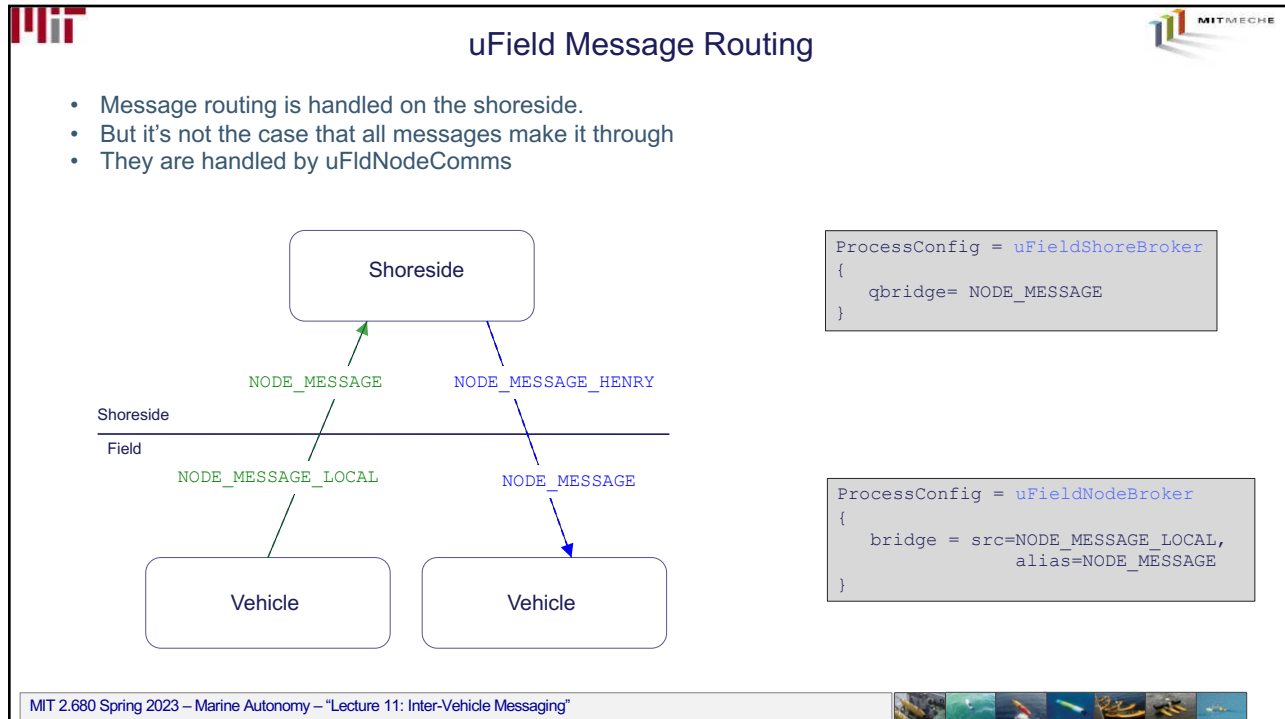
10



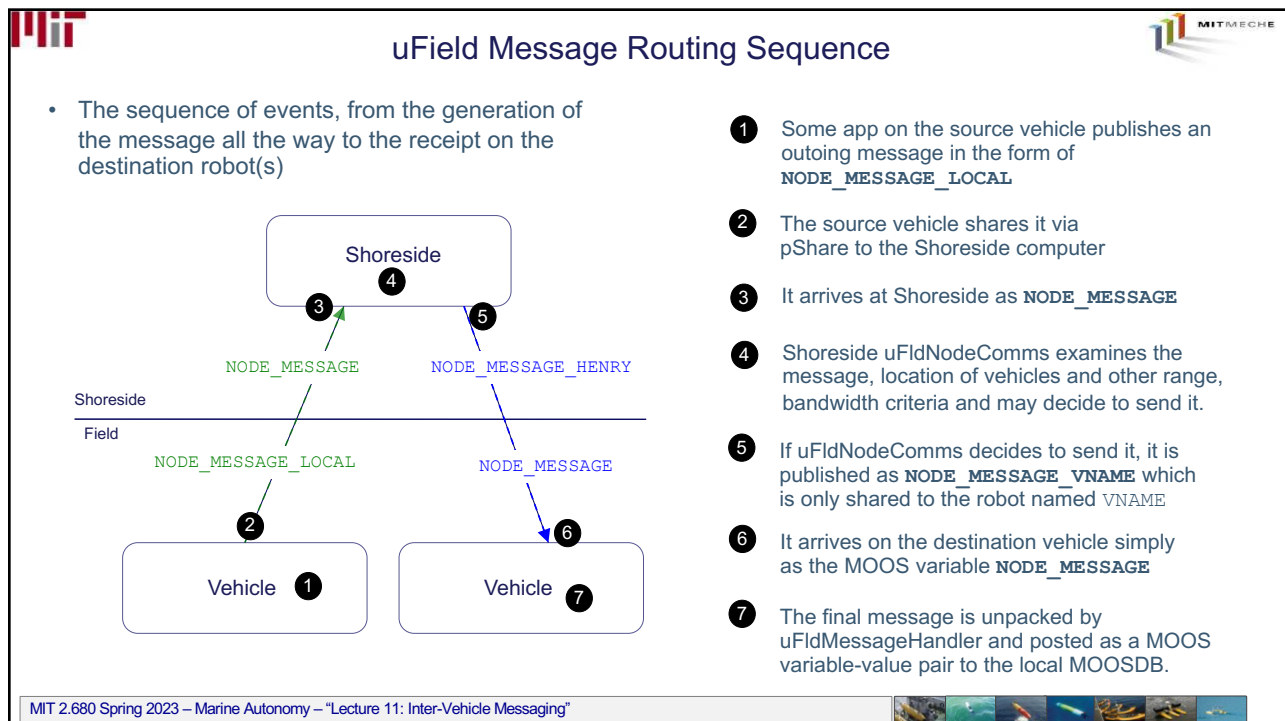
12




13



14




15



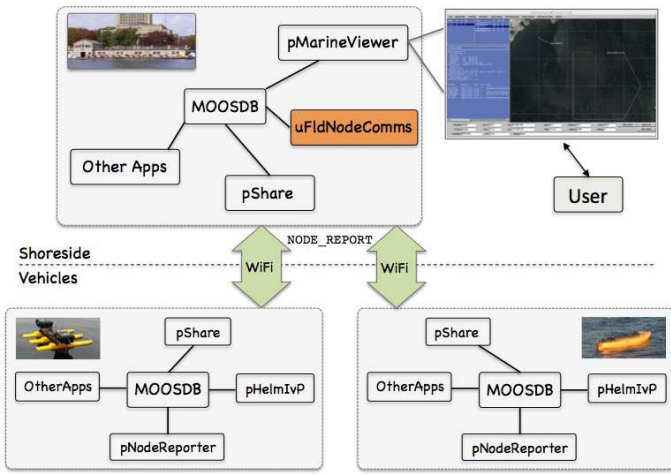
## The uFldNodeComms App

Typical Application Topology



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


- It subscribes for the the **NODE\_REPORT** messages arriving from all vehicles.
- It knows the position of all vehicles.
- It shares vehicle position information to all other vehicles. To support collision avoidance. Separate from message passing.
- It knows the range between any pair of vehicles and may use that to block a message.
- It keeps track of when each vehicle sent its previous message, to perhaps limit message frequency.
- It publishes visual objects for pMarineViewer to indicate comms status.
- It keeps track of all sent and dropped messages for viewing and debugging in its AppCast output, viewable in pMarineViewer.




The diagram illustrates the typical application topology. On the shoreside, a central box contains **MOOSDB**, **uFldNodeComms**, **pShare**, **Other Apps**, and **pMarineViewer**. **uFldNodeComms** is connected to **MOOSDB** and **pShare**. **pMarineViewer** is connected to **MOOSDB** and **uFldNodeComms**. A **User** is shown interacting with **pMarineViewer**. Below the shoreside, two **Shoreside Vehicles** are shown. Each vehicle box contains **Other Apps**, **MOOSDB**, **pShare**, **pHelmIVP**, and **pNodeReporter**. **MOOSDB** is connected to **Other Apps**, **pShare**, and **pHelmIVP**. **pNodeReporter** is connected to **MOOSDB**. **pShare** is connected to **MOOSDB**. The shoreside and vehicles are connected via **WiFi** links. A **NODE\_REPORT** message is shown being sent from a vehicle to the shoreside.

MIT 2.680 Spring 2023 – Marine Autonomy – "Lecture 11: Inter-Vehicle Messaging"

16



## The uFldNodeComms Configuration



The **uFldNodeComms** configuration parameters:

```


ProcessConfig = uFldNodeComms
{
  comms_range      = 200
  min_msg_interval = 60
  max_msg_length   = 100
  view_node_report_pulses = true
  stale_time       = 5
  groups           = true
  critical_range    = 1000
}
  
```

- Distance in meters between vehicles (default is 100m)
- Min time in seconds between messages from a vehicle (default is 30 sec)
- Max chars in a string message (default is 1,000 characters)
- Boolean indicating whether visual artifacts are to be generated indicating that node reports are being shared between vehicles


MIT 2.680 Spring 2023 – Marine Autonomy – "Lecture 11: Inter-Vehicle Messaging"

17





## The uFldNodeComms Basic Configuration



The `uFldNodeComms` configuration parameters:

```

ProcessConfig = uFldNodeComms
{
  comms_range      = 200
  min_msg_interval = 60
  max_msg_length   = 100

  view_node_report_pulses = true

  stale_time       = 5
  groups           = true

  critical_range   = 1000
}


```


Distance in meters between vehicles (default is 100m)

Min time in seconds between messages from a vehicle (default is 30 sec)


Max chars in a string message (default is 1,000 characters)

Boolean indicating whether visual artifacts are to be generated indicating that node reports are being shared between vehicles




MIT 2.680 Spring 2023 – Marine Autonomy – "Lecture 11: Inter-Vehicle Messaging"


18



## The uFldNodeComms Handling Stale Vehicles



The `uFldNodeComms` configuration parameters:

```

ProcessConfig = uFldNodeComms
{
  comms_range      = 200
  min_msg_interval = 60
  max_msg_length   = 100

  view_node_report_pulses = true


  stale_time       = 5
  groups           = true

  critical_range   = 1000
}


```

**stale\_time:** Time in seconds after which a vehicle will not receive node reports or messages unless a node report has been received by that vehicle. The default is 5 seconds.


- Since up-to-date inter-vehicle range information is used as part of the criteria in determining whether a vehicle receives a new node report from another, the position of the candidate recipient vehicle needs to reasonably up-to-date.
- If a recipient vehicle becomes stale, it will not receive **NODE\_MESSAGE** messages.

MIT 2.680 Spring 2023 – Marine Autonomy – "Lecture 11: Inter-Vehicle Messaging"


19



## The uFldNodeComms Support for Groups



The **uFldNodeComms** configuration parameters:

```

ProcessConfig = uFldNodeComms
{
  comms_range      = 200
  min_msg_interval = 60
  max_msg_length   = 100

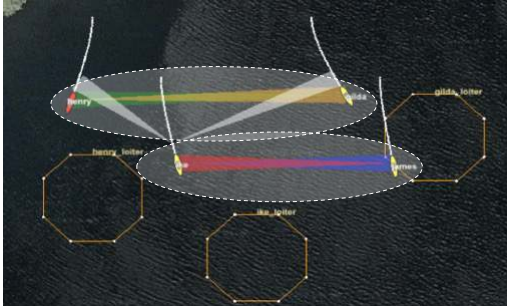
  view_node_report_pulses = true

  stale_time       = 5
  groups           = true
  critical_range    = 1000
}

```

**groups:** If true, inter-vehicle node reports are shared only if two vehicles are in the same group. Default is false.


- The group name is a field contained in the node report itself, so the onus is on the vehicle to include this information as part of its report.
- pNodeReporter can be configured with **group=<group-name>** where the group information is declared for inclusion in all node reports.




- Motivation for groups: to support multi-vehicle competitions where some vehicles want to convey positions to teammates, but not adversaries.

MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”

20



## The uFldNodeComms Critical Range



The **uFldNodeComms** configuration parameters:

```

ProcessConfig = uFldNodeComms
{
  comms_range      = 200
  min_msg_interval = 60
  max_msg_length   = 100

  view_node_report_pulses = true

  stale_time       = 5
  groups           = true
  critical_range    = 1000
}



```

**critical range:** Range in meters within which inter-vehicle node reports will be shared even if group membership would otherwise disallow. The default is 30 meters.

- When the two vehicles are within a range deemed critical, as set by the **critical\_range** configuration parameter, node reports are shared between vehicles regardless of the **comms\_range** parameter and the **groups** parameter.
- The default for this parameter is 30 meters.
- The thought behind this feature is that, while it may be advantageous to not broadcast your own vehicle position to non group members for the purposes of a competition, it may be a good idea to share this information for the sake of collision avoidance.

MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”

21


## The uFldMessageHandler Configuration

The `uFldMessageHandler` configuration parameters:

```
ProcessConfig = uFldMessageHandler
{
    strict_addressing = false
    appcast_trunc_msg = 60
}
```

**strict\_addressing:** If true, only messages with a destination specified by `dest_node`, matching the local community name are processed. Other messages with a destination specified by a group designation are ignored. The default is false.

**appcast\_trunc\_msg:** Number of characters allowed in the appcast report for each line reporting a successful message. The default is 75. Setting it to zero means no truncating will be applied.

MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”


22




## Signs of Healthy Messaging

MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”


23



## Visual Signs of Healthy Messaging

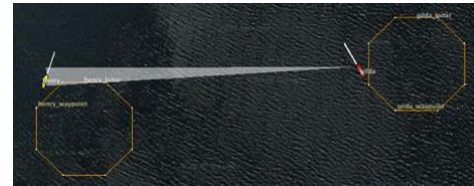
From the charlie\_dana\_messaging mission in Lab 9



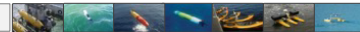
NODE\_REPORT messages are being shared



NODE\_MESSAGE messages are being shared



MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”



24



## AppCasting Signs of Healthy Messaging

Status of `uFidNodeComms` is contained in its AppCast output.

There are several fields confirming healthy messaging.

```

1  =====
2  uFidNodeComms shoreside                                0/0(339)
3  =====
4  Node Report Summary
5  =====
6      Total Received: 3101
7      GILDA: 1552      (0.0)
8      HENRY: 1549      (0.0)
9  =====
10     Total Sent: 628
11     GILDA: 315
12     HENRY: 313
13  =====
14  Node Message Summary
15  =====
16     Total Msgs Received: 4
17     HENRY: 4      (24.1)
18  =====
19     Total Sent: 4
20     GILDA: 4
21  =====
22     Total Blocked Msgs: 0
23     Invalid: 0
24     Stale Receiver: 0
25     Too Recent: 0
26     Msg Too Long: 0
27     Range Too Far: 0
28  =====
29
30  Most Recent Events (4):
31  =====
32  [96.22]: Msg rec'd: src_node=henry,dest_node=gilda,var_name=UPDATE_LOITER,string_val=speed=2.4
33  [71.10]: Msg rec'd: src_node=henry,dest_node=gilda,var_name=UPDATE_LOITER,string_val=speed=2.4
34  [56.46]: Msg rec'd: src_node=henry,dest_node=gilda,var_name=UPDATE_LOITER,string_val=speed=2.4
35  [38.32]: Msg rec'd: src_node=henry,dest_node=gilda,var_name=UPDATE_LOITER,string_val=speed=0.4
  
```


Lots of messages received and sent – that's a good sign!

No blocked messages. Also a good sign!


MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”



25



## AppCasting Signs of Healthy Messaging



Status of `uFldMessageHandler` is contained in its AppCast output


- Totals valid messages
- Invalid messages are ill-formed
- Rejected messages failed one of the range, bandwidth etc. criteria
- Per source summary, one line per other robot.
- Finite list of most recent messages.
- Automatically truncated in number.
- Truncated in length as per set by the user with the `appcast_trunc_msg` parameter.

```


1 =====
2 uFldMessageHandler gilda 0/0(841)
3 =====
4 Overall Totals Summary
5 =====
6 Total Received Valid: 3
7 Invalid: 0
8 Rejected: 0
9 Time since last Msg: 101.3
10
11 Per Source Node Summary
12 =====
13 Source Total Elapsed Variable Value
14 -----
15 henry 3 101.3 RETURN true
16
17 Last Few Messages: (oldest to newest)
18 =====
19 Valid Mgs:
19 src_node=henry,dest_node=gilda,var_name=UPDATE_LOITER,string_val=speed
20 src_node=henry,dest_node=gilda,var_name=UPDATE_LOITER,string_val=speed
21 src_node=henry,dest_node=gilda,var_name=RETURN,string_val=true
22 Invalid Mgs:
23 NONE
24 Rejected Mgs:
25 NONE
  
```

MIT 2.680 Spring 2023 – Marine Autonomy – "Lecture 11: Inter-Vehicle Messaging"

26



## ALog Signs of Healthy Messaging



We can check the log files *on the Shoreside*:

```

$ cd LOG_SHORESIDE_18_3_2018_16_36_17
$ aloggrep *.alog NODE_MESSAGE_CHARLIE NODE_MESSAGE_DANA
  
```

```

#####
%% LOG FILE:      ./LOG_SHORESIDE_18_3_2018_16_36_18/LOG_SHORESIDE_18_3_2018_16_36_18.alog
%% FILE OPENED ON Wed Dec 31 19:00:00 1969
%% LOGSTART      22821080675.4
#####
120.863  NODE_MESSAGE_CHARLIE uFldNodeComms src_node=dana,dest_node=all,var_name=UP_LOITER,string_val=ycenter_assign=-43.075
127.423  NODE_MESSAGE_DANA   uFldNodeComms src_node=charlie,dest_node=all,var_name=UP_LOITER,string_val=ycenter_assign=-19.65
553.160  NODE_MESSAGE_CHARLIE uFldNodeComms src_node=dana,dest_node=all,var_name=UP_LOITER,string_val=ycenter_assign=-24.35
572.299  NODE_MESSAGE_DANA   uFldNodeComms src_node=charlie,dest_node=all,var_name=UP_LOITER,string_val=ycenter_assign=-69.675
Total lines retained: 9 (0.01%)
Total lines excluded: 60558 (99.99%)
Total chars retained: 839 (0.01%)
Total chars excluded: 9972138 (99.99%)
Variables retained: (2) NODE_MESSAGE_CHARLIE, NODE_MESSAGE_DANA
  
```

**NOTE:** `NODE_MESSAGE_CHARLIE` entries indicate outgoing messages to `charlie`. In this case we can also see they are coming from `dana`.

MIT 2.680 Spring 2023 – Marine Autonomy – "Lecture 11: Inter-Vehicle Messaging"

27





# When Things Go Wrong

## (How to Debug)

MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”



28



# When Things Go Wrong

You were expecting:



But you're seeing this instead (no comms pulses)





MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”

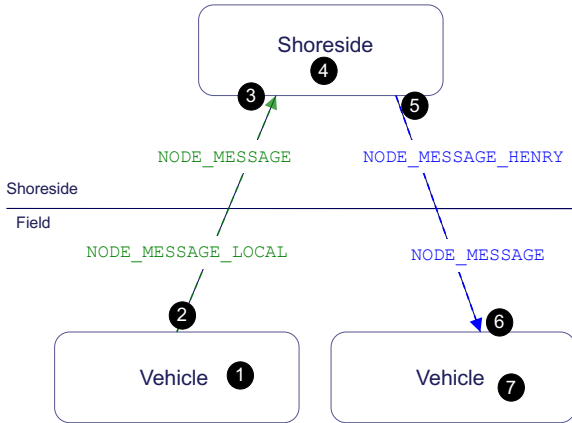


29

## Debugging Broken Messaging

- Re-visiting the message passing “pipeline”:





- 1 Some app on the source vehicle publishes an outgoing message in the form of **NODE\_MESSAGE\_LOCAL**
- 2 The source vehicle shares it via pShare to the Shoreside computer
- 3 It arrives at Shoreside as **NODE\_MESSAGE**
- 4 Shoreside uFldNodeComms examines the message, location of vehicles and other range, bandwidth criteria and may decide to send it.
- 5 If uFldNodeComms decides to send it, it is published as **NODE\_MESSAGE\_VNAME** which is only shared to the robot named **VNAME**
- 6 It arrives on the destination vehicle simply as the MOOS variable **NODE\_MESSAGE**
- 7 The final message is unpacked by uFldMessageHandler and posted as a MOOS variable-value pair to the local MOOSDB.

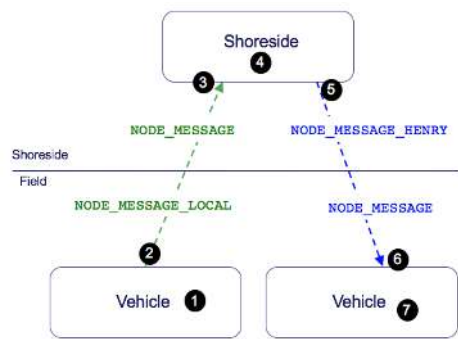
MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”

30

## Debugging Broken Messaging (Stage 1)

- Re-visiting the message passing “pipeline”:



- 1 Some app on the source vehicle publishes an outgoing message in the form of **NODE\_MESSAGE\_LOCAL**

DEBUGGING STEPS


- Was **NODE\_MESSAGE\_LOCAL** ever actually posted to the MOOSDB on the vehicle?
- You can check while running the mission by running a scope:

```
$ uXMS mission.moos NODE_MESSAGE_LOCAL
```
- You can check after running the mission by examining the alog file:


```
$ aloggrep file.alog NODE_MESSAGE_LOCAL
```
- If it was never posted, re-examine what was supposed to generate this posting.

MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”

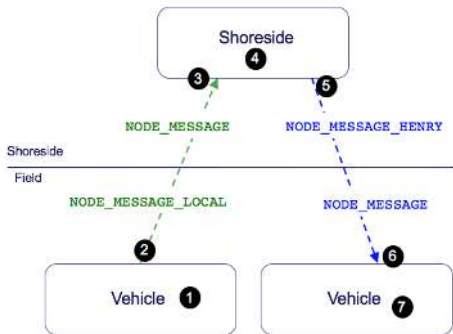
31



## Debugging Broken Messaging (Stage 2/3)



- Re-visiting the message passing “pipeline”:



- 2 The source vehicle shares it via pShare to the Shoreside computer
- 3 It arrives at Shoreside as `NODE_MESSAGE`


**DEBUGGING STEPS**

- Did `NODE_MESSAGE` arrive in the Shoreside?
- You can check after running the mission by examining the *Shoreside* `alog` file:
 


```
$ aloggrep shoreside.alog NODE_MESSAGE
```
- If it was never posted, things to check:
- Was pShare running on vehicle? Shoreside?
- Did the vehicle `uFldNodeBroker` config block include sharing for `NODE_MESSAGE_LOCAL`?

MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”

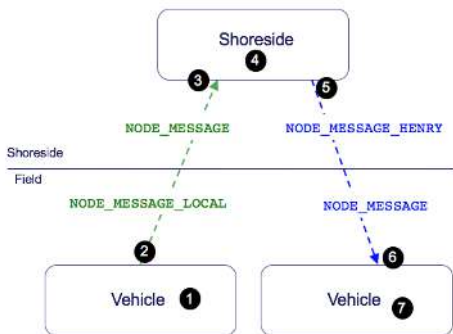
32



## Debugging Broken Messaging (Stage 4)



- Re-visiting the message passing “pipeline”:



- 4 Shoreside `uFldNodeComms` examines the message, location of vehicles and other range, bandwidth criteria and may decide to send it.

**DEBUGGING STEPS**


- In this stage `uFldNodeComms` will ingest a `NODE_MESSAGE` and post a `NODE_MESSAGE_VNAME` if all goes well. Was `NODE_MESSAGE_VNAME` posted?
- You can check after running the mission by examining the *Shoreside* `alog` file:
 

```
$ aloggrep shoreside.alog NODE_MESSAGE_HENRY
```
- If it was never posted, things to check:
- Was the message blocked because it was ill-formed?
- Was the message blocked due to range between vehicles?
- Was the message blocked due to message length?
- Was the message blocked due to a stale receiving vehicle?
- Was the message blocked due to frequency constraints?


**For debugging blocked messages, the AppCasting output of `uFldNodeComms` is your most powerful debugging tool.**

MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”

33



## Debugging Blocked Messages at the Shoreside



A *blocked message* at the Shoreside is a one where uFldNodeComms has ingested a **NODE\_MESSAGE** but has not made a corresponding **NODE\_MESSAGE\_VNAME** post.

**Possible reasons for blocking:**

- The message was ill-formed.
- The message was blocked due to range between vehicles.
- The message was blocked due to message length.
- The message was blocked due to frequency constraints. (too soon since the previous successful message)
- The message was blocked due to a stale receiver vehicle, or the receiver vehicle is not known to uFldNodeComms.
- Re-run the mission and check the AppCast output of uFldNodeComms (see right).
- As of now, uFldNodeComms does not produce similar output to debugging MOOS variables for logging.


```

1 =====
2 uFldNodeComms shoreside                                0/0(339)
3 =====
4 Node Report Summary
5 =====
6         Total Received: 3101
7             GILDA: 1552             (0.0)
8             HENRY: 1549             (0.0)
9         -----
10        Total Sent: 628
11            GILDA: 315
12            HENRY: 313
13
14 Node Message Summary
15 =====
16 Total Msgs Received: 4
17             HENRY: 4             (24.1)
18         -----
19        Total Sent: 4
20            GILDA: 4
21
22 Total Blocked Msgs: 0
23     Invalid: 0
24 Stale Receiver: 0
25   Too Recent: 0
26   Msg Too Long: 0
27   Range Too Far: 0
28
29 =====
30 Most Recent Events (4):
31 =====
32 [96.22]: Msg rec'd: src_node=henry,dest_node=gilda,var_name=UPDATE_LOI?
33 [71.10]: Msg rec'd: src_node=henry,dest_node=gilda,var_name=UPDATE_LOI?
34 [56.46]: Msg rec'd: src_node=henry,dest_node=gilda,var_name=UPDATE_LOI?
35 [38.32]: Msg rec'd: src_node=henry,dest_node=gilda,var_name=UPDATE_LOI?


```

If there are blocked messages, they would be reported here


MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”



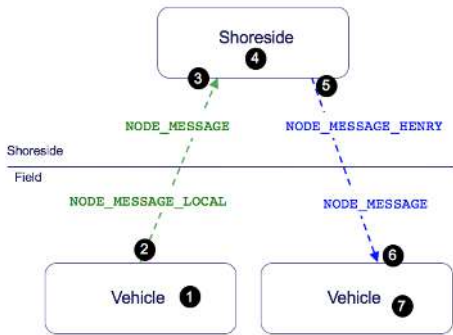
34



## Debugging Broken Messaging (Stage 5/6)



• Re-visiting the message passing “pipeline”:



5 If uFldNodeComms decides to send it, it is published as **NODE\_MESSAGE\_VNAME** which is only shared to the robot named **VNAME**

6 It arrives on the destination vehicle simply as the MOOS variable **NODE\_MESSAGE**

DEBUGGING STEPS

• uFldNodeComms has published a **NODE\_MESSAGE\_VNAME** and it should have resulted in **NODE\_MESSAGE** on the vehicle.

• You can the *vehicle* .alog file:


```
$ aloggrep vehicle.alog NODE_MESSAGE
```

**If it was never posted, things to check:**


- Was pShare running on the Shoreside
- Was pShare running on the vehicle?
- If you were able to deploy the vehicles and see their positions updated on pMarineViewer, then very likely pShare was running on both vehicles.
- Was the Shoreside pShare configured to share **NODE\_MESSAGE\_VNAME** and to **NODE\_MESSAGE**? Check the configuration block for uFldShoreBroker and look for a configuration like like:

qbridge = NODE\_MESSAGE


MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”



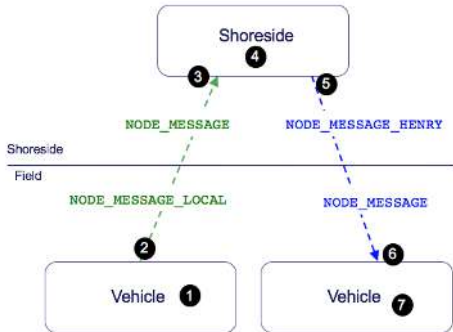
35



## Debugging Broken Messaging (Stage 7)



- Re-visiting the message passing “pipeline”:



7 The final message is unpacked by uFldMessageHandler and posted as a MOOS variable-value pair to the local MOOSDB.


**DEBUGGING STEPS**

- A **NODE\_MESSAGE** has arrived on the vehicle, but the contents of the message have not been posted.
- Again, you can verify that **NODE\_MESSAGE** has been received on the vehicle by checking the *vehicle* `alog` file:


```
$ aloggrep vehicle.alog NODE_MESSAGE
```

**If the contents of the message was not posted, things to check:**


- The message was invalid (ill-formed syntactically)
- The message was rejected, perhaps because the “addressee” was set to “all”, and message handler was configured to require strict matching of vehicle name.
- For debugging blocked messages, the AppCasting output of uFldMessageHandler is your most powerful debugging tool.**

MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”


36



## Debugging Broken Messaging (Stage 7)



**Possible reasons for unposted messages from an incoming **NODE\_MESSAGE** on a vehicle:**

- The message was invalid (ill-formed syntactically)
- The message was rejected, perhaps because the “addressee” was set to “all”, and message handler was configured to require strict matching of vehicle name.


```

1 =====
2 uFldMessageHandler gus 0/0 (841)
3 =====
4 Overall Totals Summary
5 =====
6 Total Received Valid: 3
7 Invalid: 0
8 Rejected: 0
9 Time since last Msg: 101.3
10
11 Per Source Node Summary
12 =====
13 Source Total Elapsed Variable Value
14 -----
15 hal 3 101.3 RETURN true
16
17 Last Few Messages: (oldest to newest)
18 =====
19 Valid Msgs:
20   src_node=hal,dest_node=gus,var_name=UP_LOITER,string_val=speed
21   src_node=hal,dest_node=gus,var_name=UP_LOITER,string_val=speed
22   src_node=hal,dest_node=gus,var_name=RETURN,string_val=true
23 Invalid Msgs:
24   NONE
25 Rejected Msgs:
26   NONE

```


If there are invalid or rejected messages, they would be reported here

Contents of recent invalid or rejected messages, are shown here

MIT 2.680 Spring 2023 – Marine Autonomy – “Lecture 11: Inter-Vehicle Messaging”



37






## Lab 9 Preview


### Inter-vehicle Messaging




In today's lab, we will build a simple two-vehicle mission:

- Each vehicle is loitering in its half of an east-west op-area.
- Each vehicle periodically sends a message to the other vehicle to switch its region





MIT 2.680 Spring 2023 – Marine Autonomy – "Lecture 11: Inter-Vehicle Messaging"


38



# END



MIT 2.680 Spring 2023 – Marine Autonomy – "Lecture 11: Inter-Vehicle Messaging"


39