MIT 2.680

UNMANNED MARINE VEHICLE AUTONOMY, SENSING, AND COMMUNICATIONS

Lecture 8 – Multi-Vehicle Mission Debugging

March 10th, 2026

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

Email:
Mike Benjamin, mikerb@mit.edu

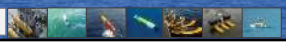
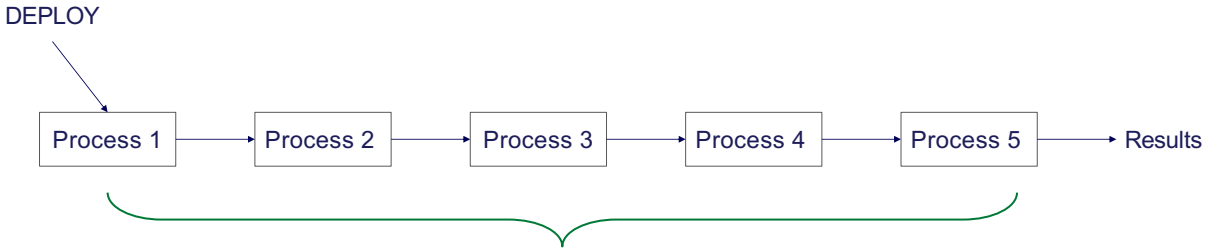


Photo by Arjan Vermeij
GLINT '09

1

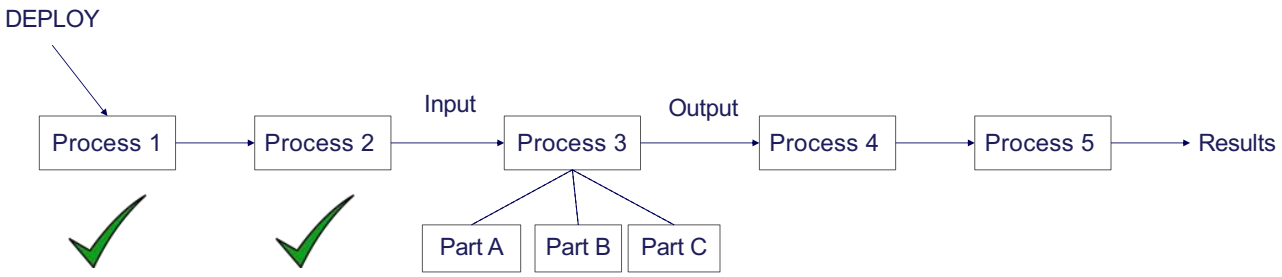
Know Your Pipeline



- A pipeline may involve several processes (e.g., MOOS Apps), handing a portion of the problem
- In development, and debugging, it is usually preferable to focus step by step.

2

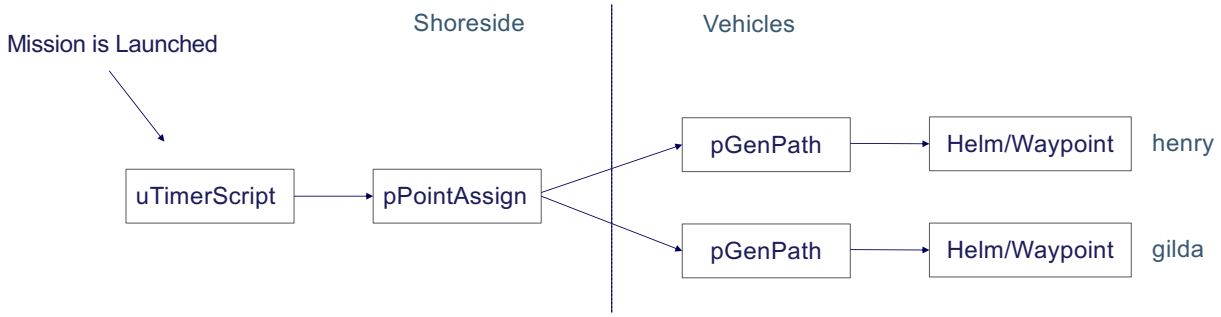
Know Your Pipeline



- Every stage typically has its own input and output
- If you can, build the components of the pipeline, step by step and verify

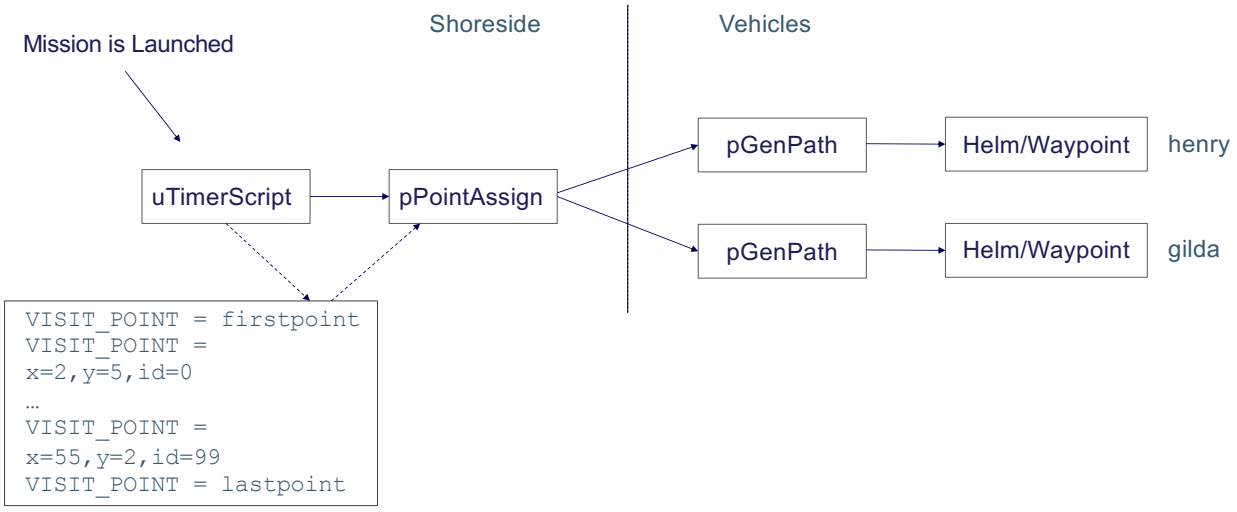
The Distributed TSP Pipeline

The Distributed TSP Pipeline



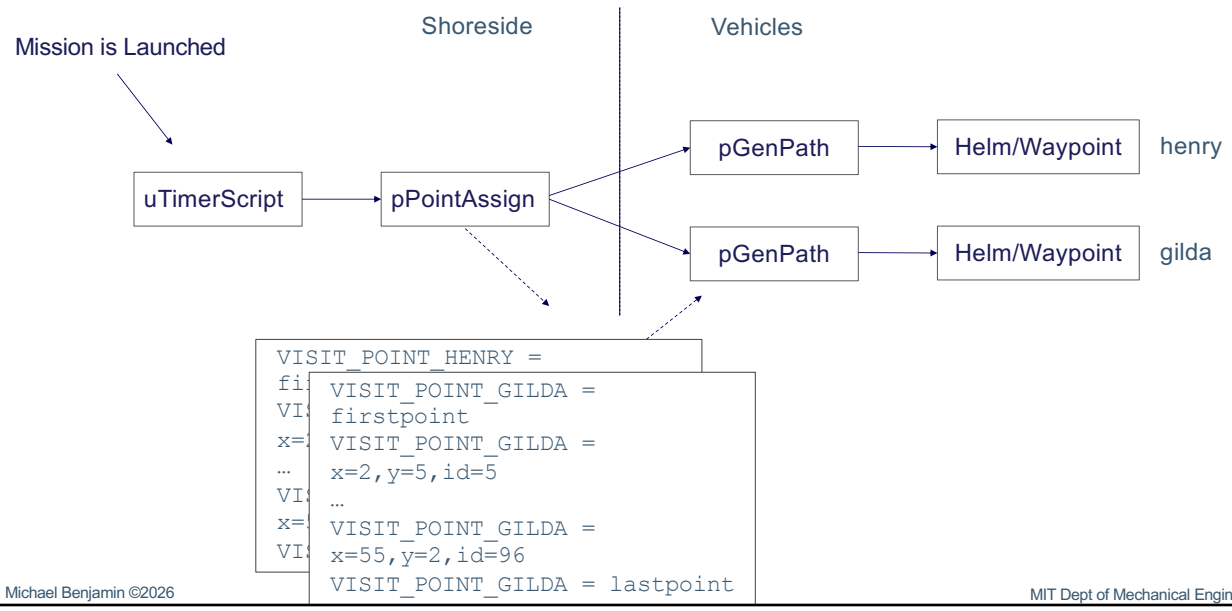
5

The Distributed TSP Pipeline

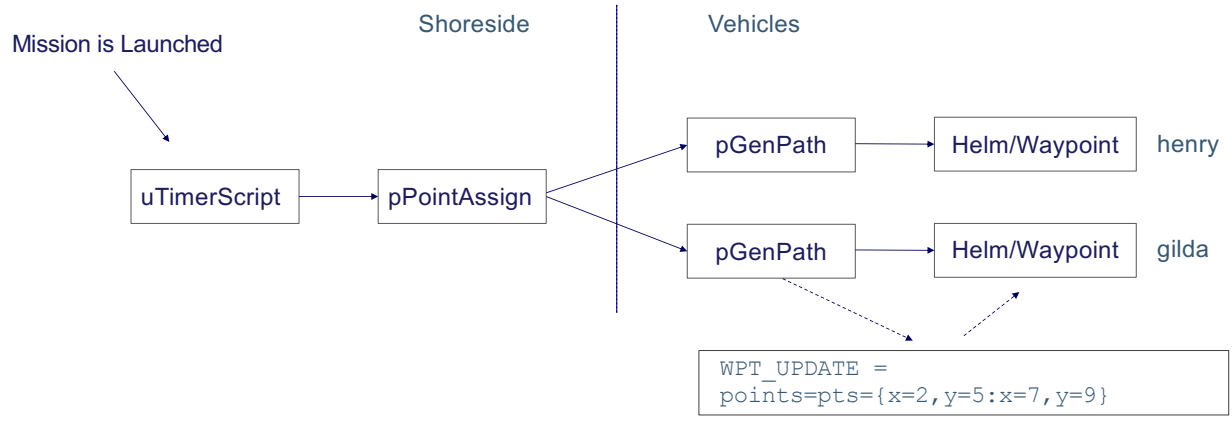


6

The Distributed TSP Pipeline



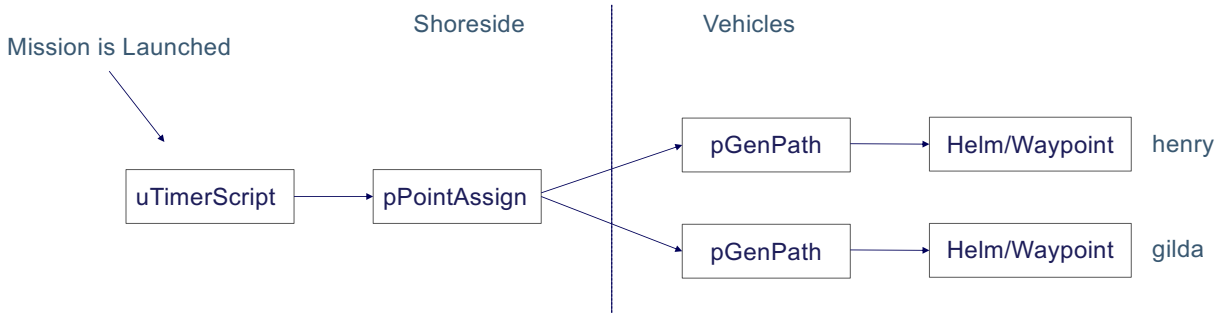
The Distributed TSP Pipeline



Implementing the Distributed TSP Pipeline

9

The Distributed TSP Pipeline Posting Points




```
VISIT_POINT = firstpoint
VISIT_POINT =
x=2,y=5,id=0
…
VISIT_POINT =
x=55,y=2,id=99
VISIT_POINT = lastpoint
```

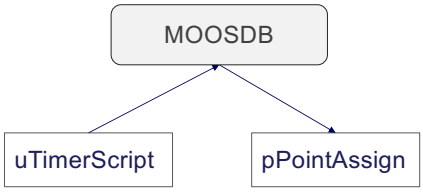
- A timer script is added to the shoreside MOOS community
- See the uTimerScript documentation (Section 9.1) for a similar example
- Use the block_on parameter to ensure that pPointAssign is listening before posting

```
block_on = pPointAssign
```


10

The Distributed TSP Pipeline Posting Points

What happens if uTimerScript starts well before pPointAssign?



shoreside.moos

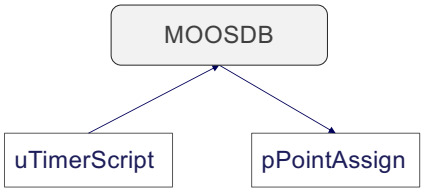
```
Processconfig = ANTLER
{
  MSBetweenLaunches = 200

  Run = MOOSDB           @ NewConsole = false
  Run = uTimerScript     @ NewConsole = false
  Run = pShare           @ NewConsole = false
  Run = pMarineViewer    @ NewConsole = false
  Run = pLogger          @ NewConsole = false
  Run = pHostInfo        @ NewConsole = false
  Run = uFldShoreBroker  @ NewConsole = false
  Run = pPointAssign     @ NewConsole = false
}
```

Recall that when an app connects to the MOOSDB, it only gets the most recent mail for any registered variables.

11

The Distributed TSP Pipeline Posting Points

We can “fix” this problem by moving uTimerScript to be the last app launched.



shoreside.moos

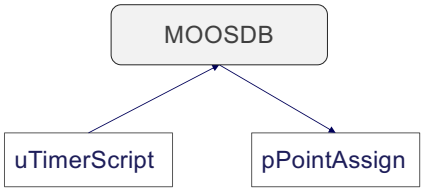
```
Processconfig = ANTLER
{
  MSBetweenLaunches = 200

  Run = MOOSDB           @ NewConsole = false
  Run = pPointAssign     @ NewConsole = false
  Run = pShare           @ NewConsole = false
  Run = pMarineViewer    @ NewConsole = false
  Run = pLogger          @ NewConsole = false
  Run = pHostInfo        @ NewConsole = false
  Run = uFldShoreBroker  @ NewConsole = false
  Run = uTimerScript     @ NewConsole = false
}
```

This is a brittle solution. Even if the person editing the .moos file were to put a comment in the file explaining why uTimerScript must be launched last.

12

The Distributed TSP Pipeline Posting Points

Another arguably better fix is to delay the timer script for some number of seconds.



shoreside.moos

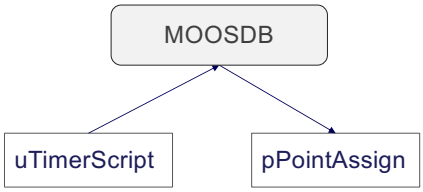
```
ProcessConfig = uTimerScript
{
  AppTick   = 3
  CommsTick = 3

  paused     = false
  reset_time = all-posted
  reset_max  = 0
  delay_start = 10

  rand_var   = …
  rand_var   = …

  event = …
}
```

This is still feels brittle. What's the right amount of time?

13

The Distributed TSP Pipeline Posting Points

A more durable fix is to ask uTimerScript to wait until pPointAssign is connected to the MOOSDB.

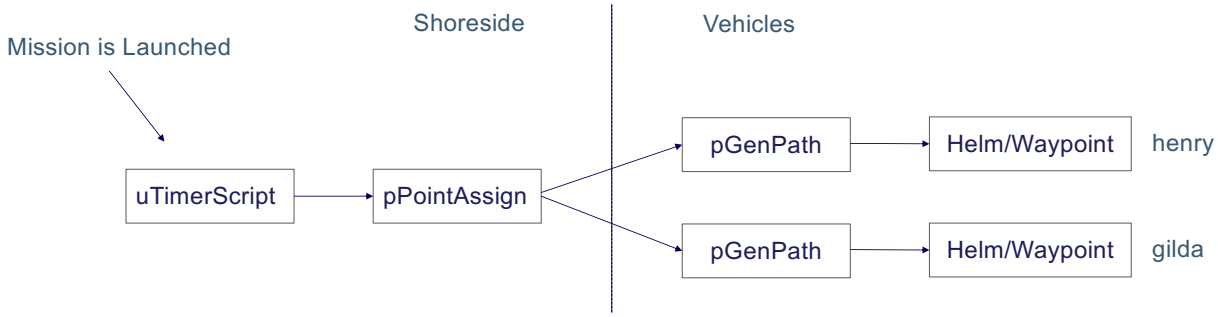


shoreside.moos

```
ProcessConfig = uTimerScript
{
  AppTick   = 3
  CommsTick = 3

  paused     = false
  reset_time = all-posted
  reset_max  = 0
  block_on   = pPointAssign

  rand_var   = …
  rand_var   = …

  event = …
}
```


14

The Distributed TSP Pipeline Posting Points



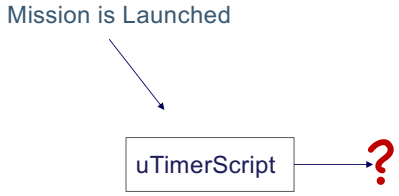

```
VISIT_POINT = firstpoint
VISIT_POINT =
x=2,y=5,id=0
…
VISIT_POINT =
x=55,y=2,id=99
VISIT_POINT = lastpoint
```

- A timer script is added to the shoreside MOOS community
- See the uTimerScript documentation (Section 9.1) for a similar example
- Use the block_on parameter to ensure that pPointAssign is listening before posting

```
block_on = pPointAssign
```


15

The Distributed TSP Pipeline Posting Points



How do we know that uTimerScript has done its job?

```
VISIT_POINT = firstpoint
VISIT_POINT =
x=2,y=5,id=0
…
VISIT_POINT =
x=55,y=2,id=99
VISIT_POINT = lastpoint
```


16

The Distributed TSP Pipeline Posting Points

Mission is Launched

uTimerScript → ?

```
VISIT_POINT = firstpoint
VISIT_POINT =
x=2,y=5,id=0
…
VISIT_POINT =
x=55,y=2,id=99
VISIT_POINT = lastpoint
```

How do we know that uTimerScript has done its job?

uTimerScript is an AppCasting MOOS App, and it will publish output indicating the number of postings.

```
uTimerScript shoreside                                    0/0(500)
Current Script Information:
     Elements: 102(102)
      Reinits: 0
    Time Warp: 1
 Time Zero is: At uTimerScript launch time (not MOOSDB start time)
  Delay Start: 0
  Delay Reset: 0
    Reset Max: 0

Run criteria:
   Block Apps: (ok) no blocking apps
       Paused: (ok) Not paused
 ConditionsOK: (ok) No un-met conditions

RandomVar  Type     Min   Max  Parameters
---------  -------  ----  ---  ----------
X          uniform  -25   200
Y          uniform  -175  -25

Total Random Pairs: 0

Var1  Var2  Type  Parameters
----  ----  ----  ----------

#/Tot  #/Loc  T/Total  T/Local  Variable/Var
-----  -----  -------  -------  ------------
94     94     2.77     0.27     VISIT_POINT = x=174.62,y=-68.44,id=94
95     95     2.77     0.27     VISIT_POINT = x=-22.975,y=-164.77,id=95
96     96     2.77     0.27     VISIT_POINT = x=42.972,y=-71.575,id=96
97     97     2.77     0.27     VISIT_POINT = x=76.768,y=-105.55,id=97
98     98     2.77     0.27     VISIT_POINT = x=59.578,y=-103.735,id=98
99     99     2.77     0.27     VISIT_POINT = x=57.62,y=-25.57,id=99
100    100    2.77     0.27     VISIT_POINT = x=189.718,y=-161.35,id=100
101    101    2.77     0.27     VISIT_POINT = lastpoint

Most Recent Events (1):
[0.26]: Script Start. Warp=1, DelayStart=0.0, DelayReset=0.0
```


The Distributed TSP Pipeline Posting Points

Mission is Launched

uTimerScript → ?

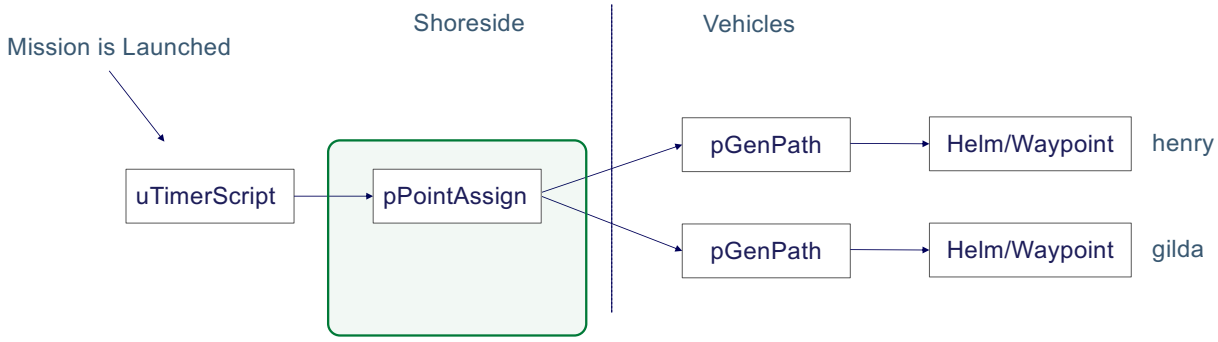
```
VISIT_POINT = firstpoint
VISIT_POINT =
x=2,y=5,id=0
…
VISIT_POINT =
x=55,y=2,id=99
VISIT_POINT = lastpoint
```

Assuming you are running pLogger, after the mission has been stopped, the aloggrep tool can be used to see all postings to a particular variable, or all postings made by a particular app:

```
$ aloggrep SHORESIDE.alog VISIT_POINT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% LOG FILE:       SHORESIDE.alog
%% FILE OPENED ON  Wed Dec 31 19:00:00 1969
%% LOGSTART        6470836882.04
2.747        VISIT_POINT          uTimerScript    firstpoint
2.747        VISIT_POINT          uTimerScript    x=174.935,y=-171.595,id=1
2.747        VISIT_POINT          uTimerScript    x=-18.61,y=-119.68,id=2
2.747        VISIT_POINT          uTimerScript    x=137.428,y=-76.165,id=3
2.747        VISIT_POINT          uTimerScript    x=195.275,y=-128.785,id=4
2.747        VISIT_POINT          uTimerScript    x=179.435,y=-141.25,id=5
2.747        VISIT_POINT          uTimerScript    x=160.535,y=-94.015,id=6
2.747        VISIT_POINT          uTimerScript    x=184.137,y=-36.055,id=7
     o o o
2.753        VISIT_POINT          uTimerScript    x=174.62,y=-68.44,id=94
2.753        VISIT_POINT          uTimerScript    x=-22.975,y=-164.77,id=95
2.753        VISIT_POINT          uTimerScript    x=42.972,y=-71.575,id=96
2.753        VISIT_POINT          uTimerScript    x=76.768,y=-105.55,id=97
2.753        VISIT_POINT          uTimerScript    x=59.578,y=-103.735,id=98
2.753        VISIT_POINT          uTimerScript    x=57.62,y=-25.57,id=99
2.753        VISIT_POINT          uTimerScript    x=189.718,y=-161.35,id=100
2.753        VISIT_POINT          uTimerScript    lastpoint
```

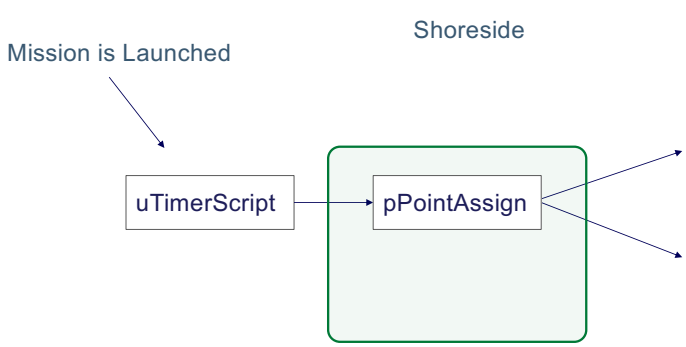
Michael Benjamin ©2026 MIT Dept of Mechanical Engineering

The Distributed TSP Pipeline
pPointAssign



How can we tell whether pPointAssign is working?

The Distributed TSP Pipeline
pPointAssign



How can we tell whether pPointAssign is working?

Answer #1: Check its output (options)

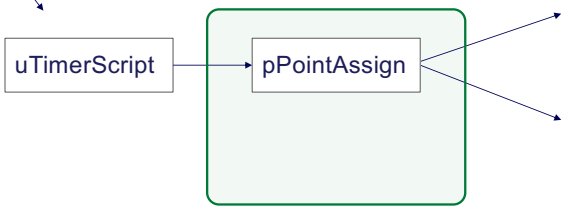
```
VISIT_POINT_HENRY =
firstpoint
VISIT_POINT_HENRY =
x=2,y=5,id=5
…
```

```
VISIT_POINT_GILDA =
firstpoint
VISIT_POINT_GILDA =
x=2,y=5,id=5
…
VISIT_POINT_GILDA =
x=55,y=2,id=96
VISIT_POINT_GILDA = lastpoint
```

The Distributed TSP Pipeline
pPointAssign

Mission is Launched

Shoreside



How can we tell whether pPointAssign is working?

Answer #1: Check its output (options)

- The uXMS utility
- Realmcasting Output
- aloggrep (post-mission)

Using RealmCasting to Scope output of pPointAssign

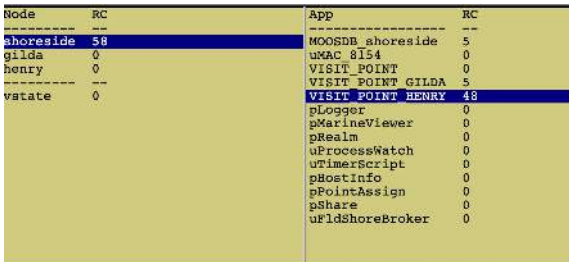
Step 1: Make sure pRealm is running in the shoreside MOOS community (we will want it on the vehicles too)

Step 2: Add a Config block in the shoreside.moos file for pRealm with:

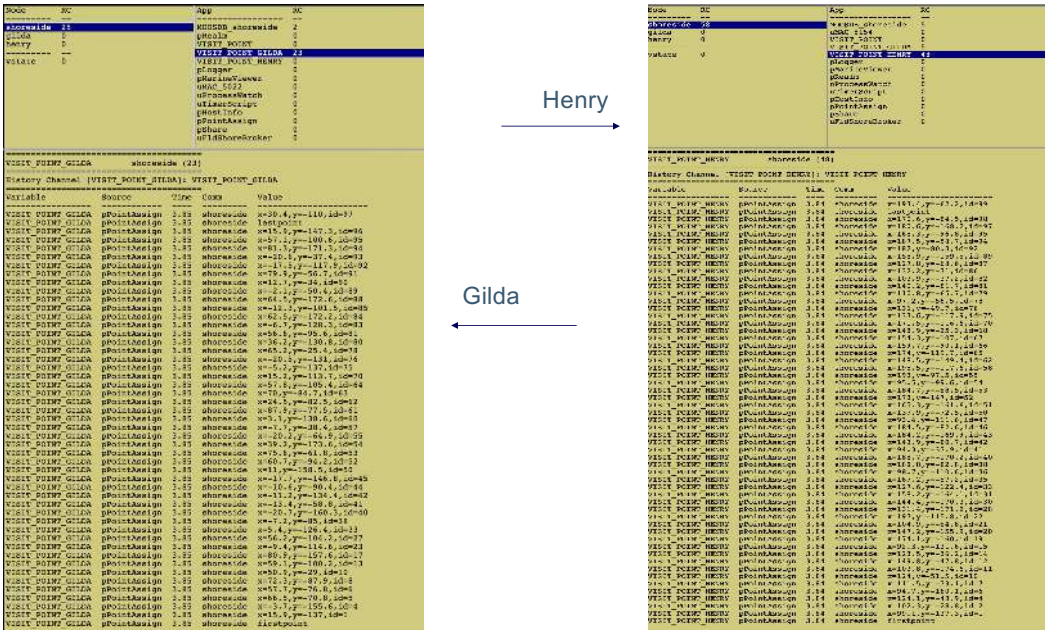
```
ProcessConfig = pRealm
{
  AppTick    = 2
  CommsTick  = 2

  hist_var = VISIT_POINT
  hist_var = VISIT_POINT_HENRY
  hist_var = VISIT_POINT_GILDA

  msg_max_hist = 120

}
```

Step 3: Re-Launch your mission, and toggle to RealmCasting from AppCasting mode with the ‘a’ key

Step 4: Select the “shoreside” in the Nodes panel, and select VISIT_POINT_HENRY in the Apps panel:



VISIT_POINT Output



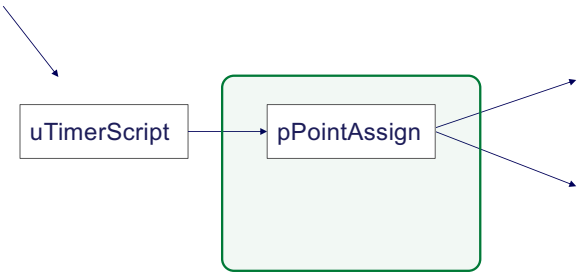
23

The Distributed TSP Pipeline

pPointAssign

Shoreside

Mission is Launched



Monitoring the output is great for ***confirming*** that things work, but what if things are not working and there is no output?

How can we tell whether pPointAssign is working?

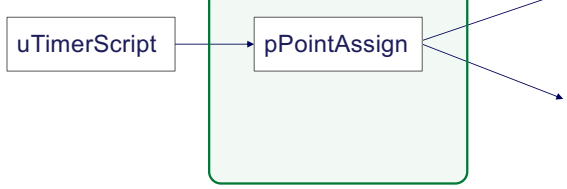
Answer #2: Monitor its internal state using AppCasting

24

The Distributed TSP Pipeline
pPointAssign

Shoreside

Mission is Launched



Monitoring the output is great for ***confirming*** that things work, but what if things are not working and there is no output?

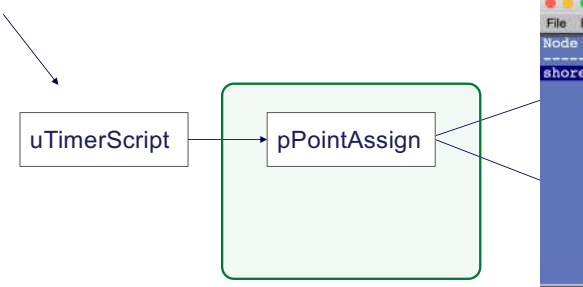
How can we tell whether pPointAssign is working?

Answer #2: Monitor its internal state using AppCasting

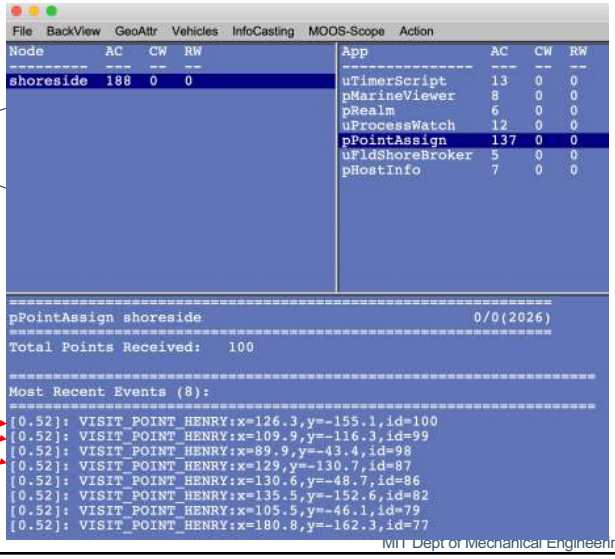
The Distributed TSP Pipeline
pPointAssign

Shoreside

Mission is Launched

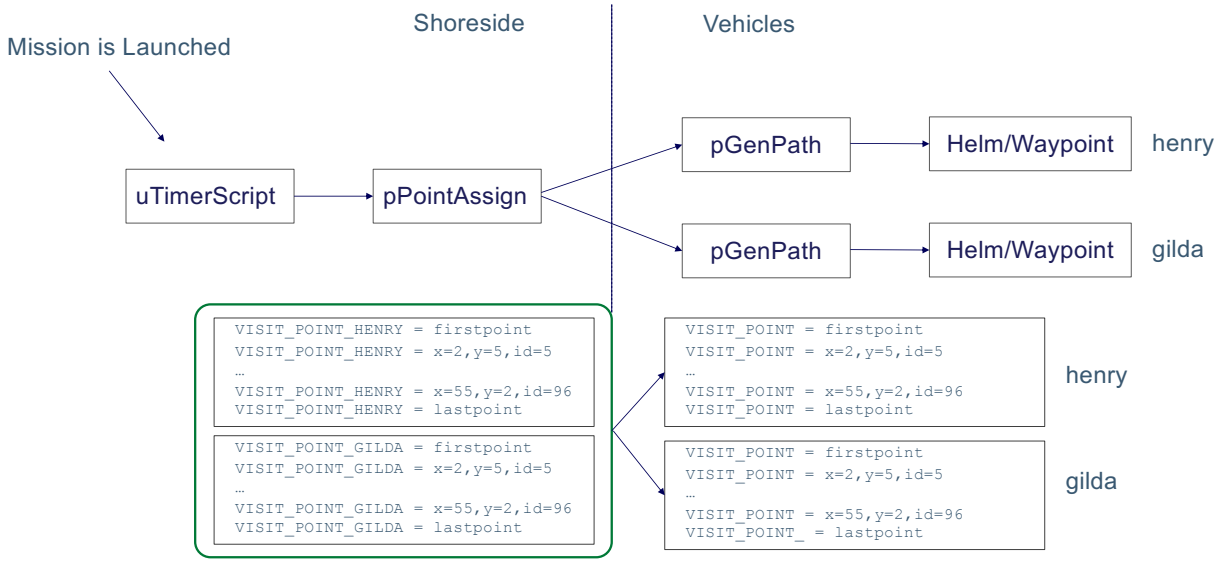


Example AppCasting output for pPointAssign



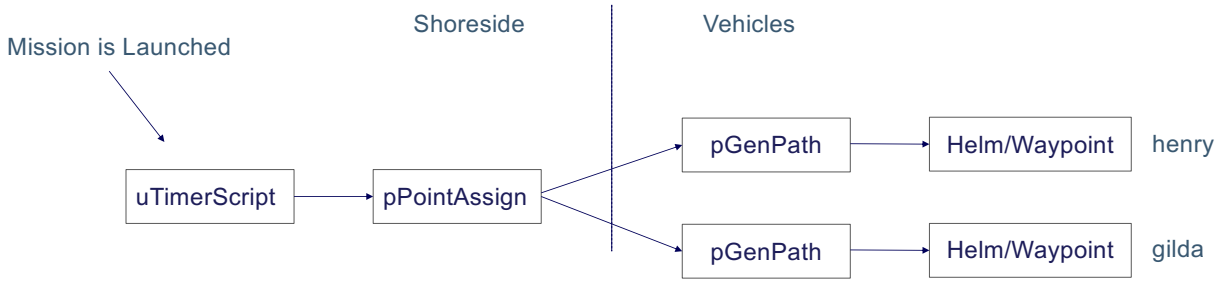
- Confirm points received
- Confirm postings
- You can add whatever helps

The Distributed TSP Pipeline
Sending Points to the Vehicles



27

The Distributed TSP Pipeline
Sending Points to the Vehicles

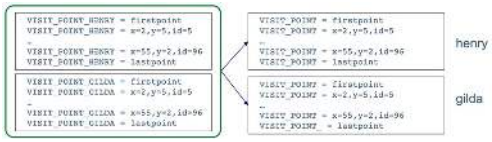


Step 1: Make sure the variable is shared. In uFldShoreBroker:

```
ProcessConfig = uFldShoreBroker
{
  AppTick    = 2
  CommsTick  = 2

  qbridge = VISIT_POINT
}
```

Step 2: Make sure pRealm is running, and add the below config parameters.

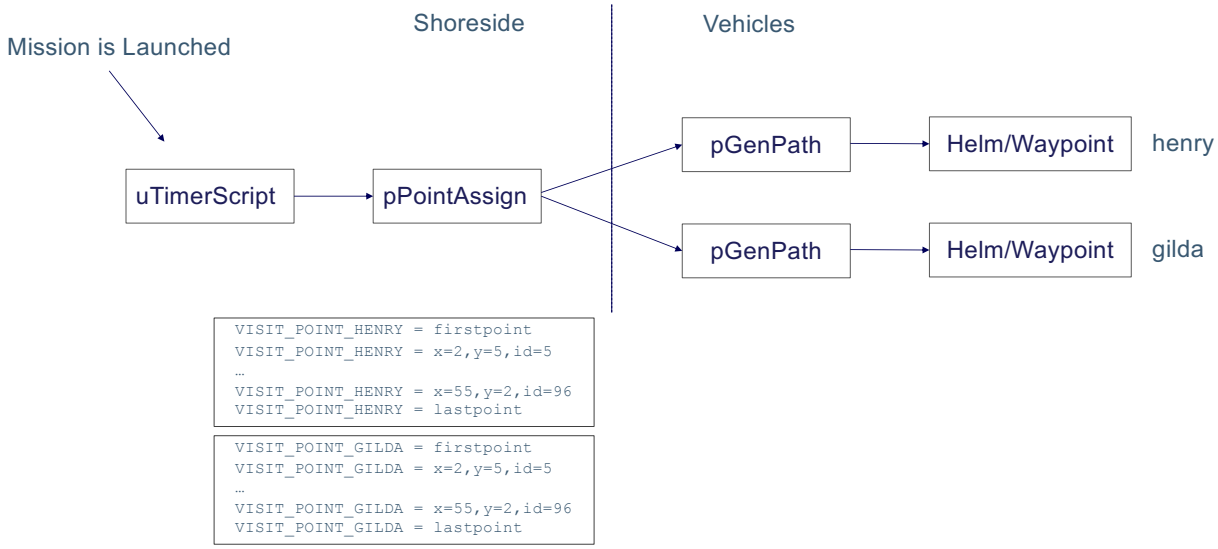
```
ProcessConfig = pRealm
{
  hist_var = VISIT_POINT
  msg_max_hist = 100
}
```



28

The Distributed TSP Pipeline

Sending Points to the Vehicles



29

Points

Storing, Serializing and De-Serializing

Creating a point

```
#include ”XYPoint.h”

XYPoint my_point;
my_point.set_vx(24);
my_point.set_vy(68);
```

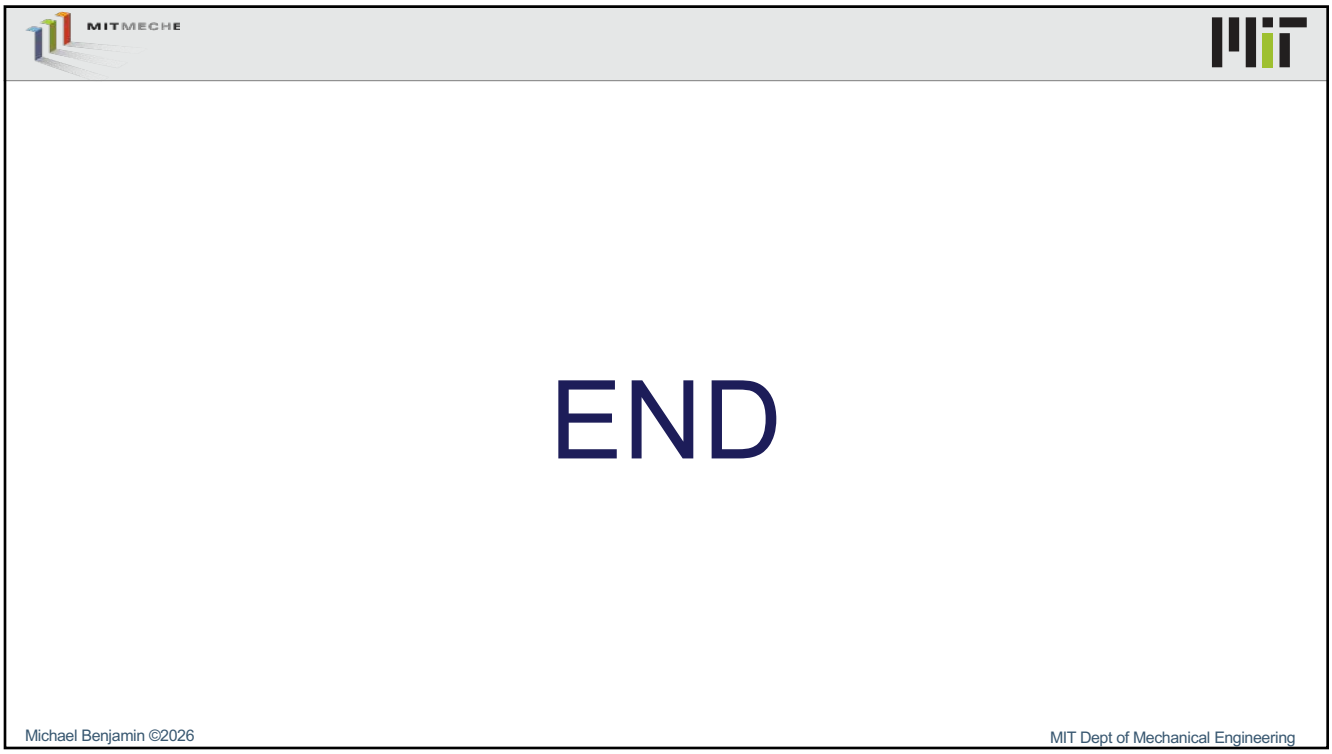
Serializing a point for posting

```
XYPoint my_point;
my_point.set_vx(24);
my_point.set_vy(68);
string str = my_point.get_spec();
Notify(“VISIT_POINT”, str);
```

DeSerializing (receiving) a point from a posting

```
#include ”XYFormatUtilsPoint.h”

string str = “x=24, y=68”;
XYPoint my_point = string2Point(str);
```


30


MITMECHE
END
Michael Benjamin ©2026
MIT Dept of Mechanical Engineering

31