

MIT 2.680
UNMANNED MARINE VEHICLE AUTONOMY,
SENSING, AND COMMUNICATIONS

Lecture 10 – Multi-Vehicle Mission Debugging



March 14th, 2024



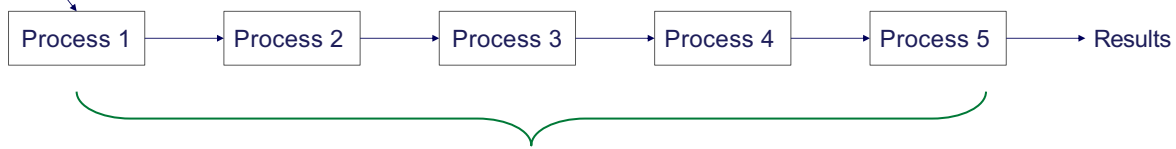
Web: <http://oceanai.mit.edu/2.680>
Email: Mike Benjamin, mikerb@mit.edu

MIT 2.680 Spring 2024 – Marine Autonomy – “Multi-Vehicle Missions”  Photo by Arjan Vermeij GLINT '09

1

 **Know Your Pipeline** 

DEPLOY



```
graph LR; P1[Process 1] --> P2[Process 2]; P2 --> P3[Process 3]; P3 --> P4[Process 4]; P4 --> P5[Process 5]; P5 --> R[Results];
```

- 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.

Michael Benjamin ©2024 MIT Dept of Mechanical Engineering

2

MITMECHE MIT

Know Your Pipeline

```
graph LR; DEPLOY --> P1[Process 1]; P1 --> P2[Process 2]; P2 -- Input --> P3[Process 3]; P3 -- Output --> P4[Process 4]; P4 --> P5[Process 5]; P5 --> Results; P3 --- PA[Part A]; P3 --- PB[Part B]; P3 --- PC[Part C];
```

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

Michael Benjamin ©2024 MIT Dept of Mechanical Engineering

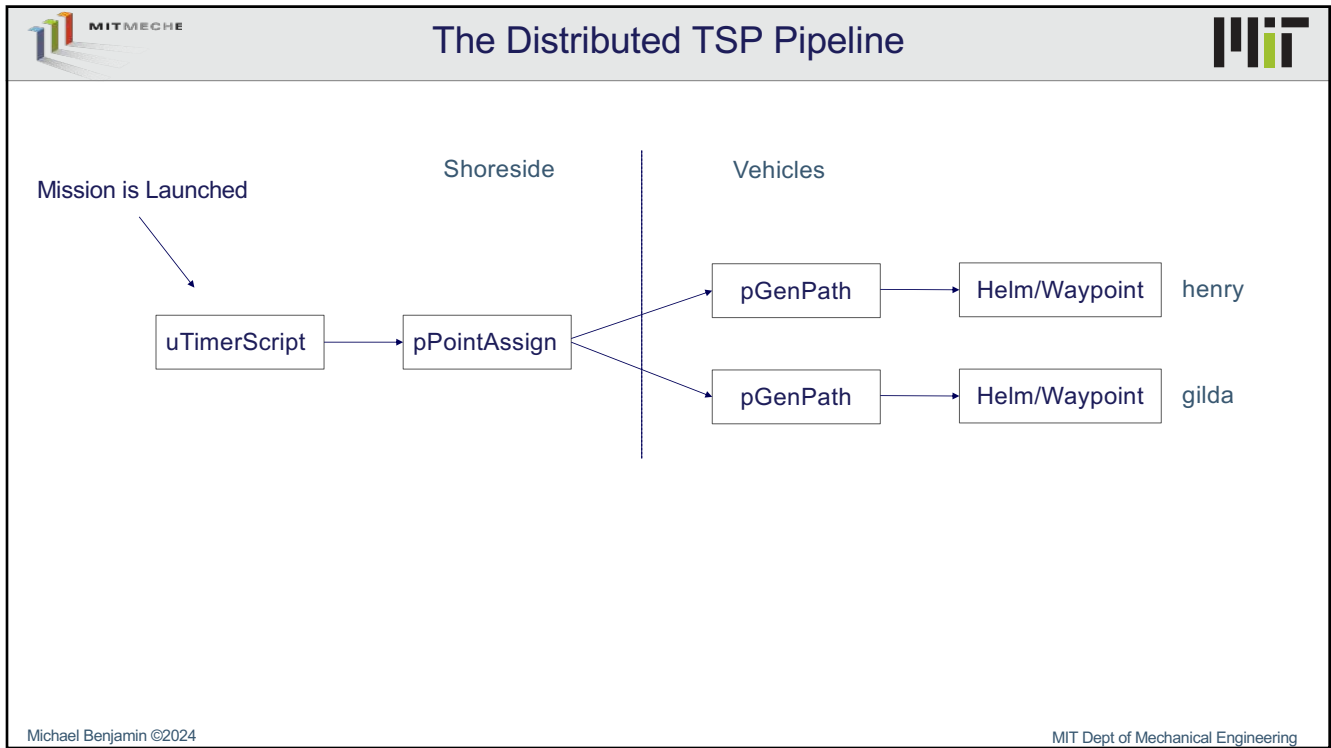
3

MITMECHE MIT

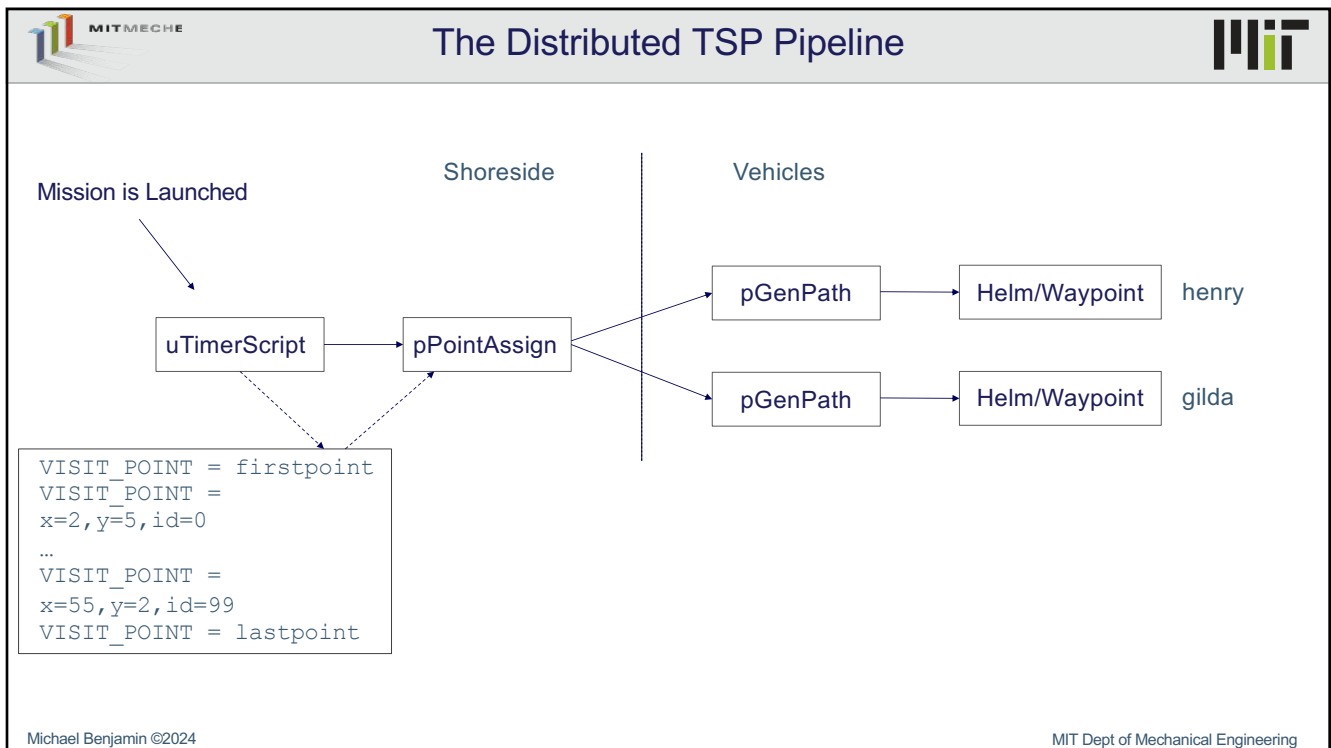
The Distributed TSP Pipeline

Michael Benjamin ©2024 MIT Dept of Mechanical Engineering

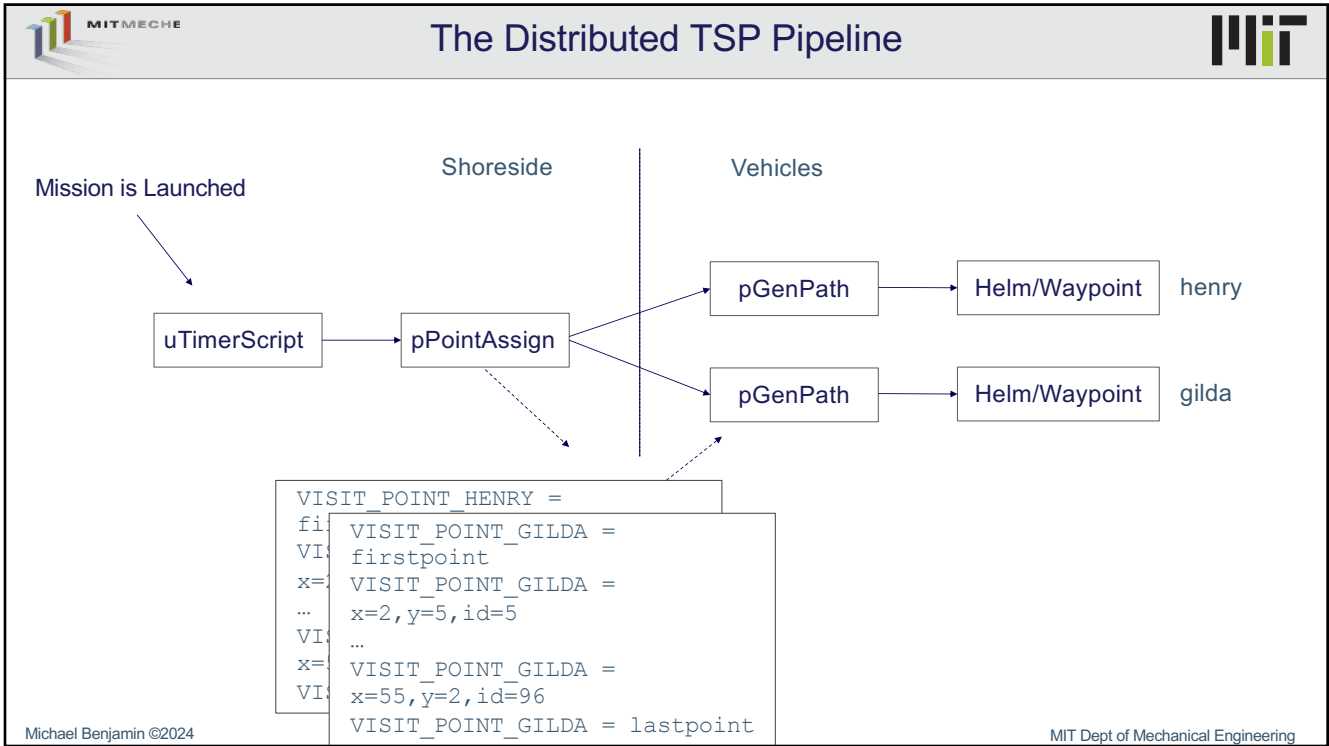
4



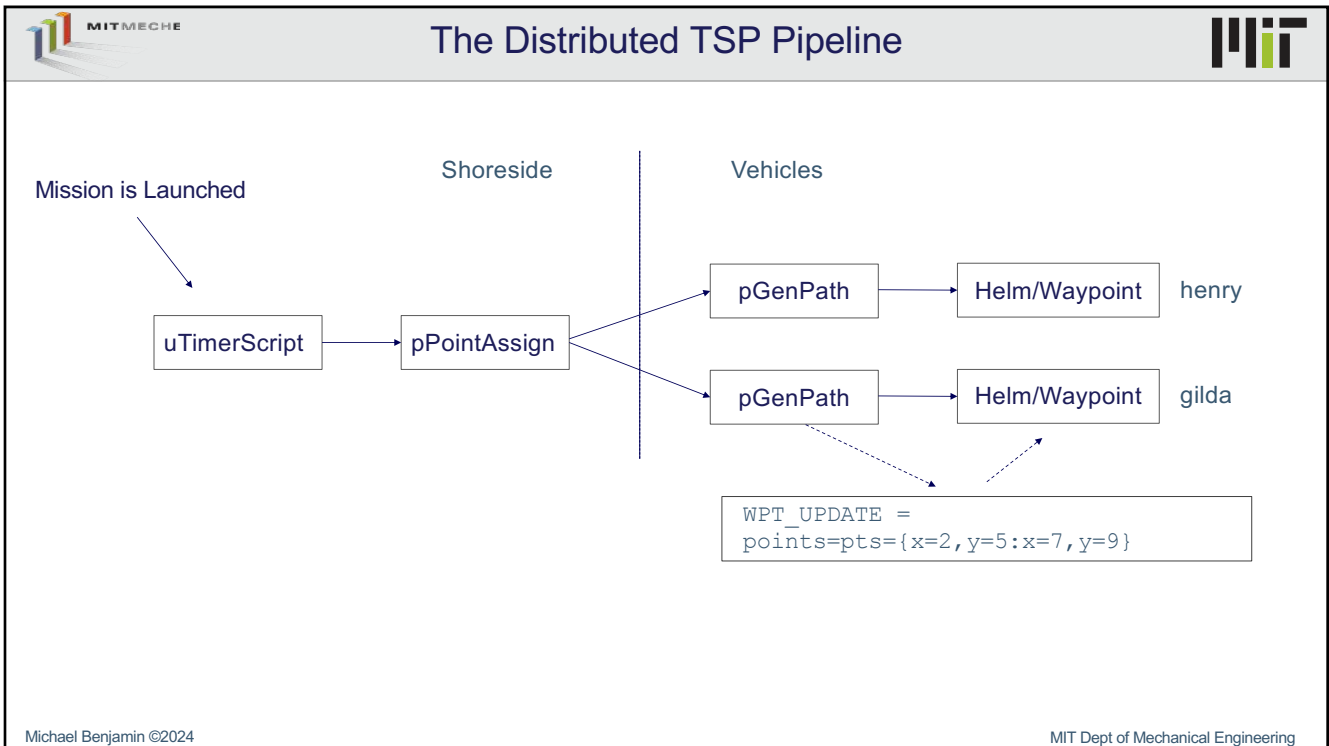
5





6



7



8





Implementing the Distributed TSP Pipeline


Michael Benjamin ©2024

MIT Dept of Mechanical Engineering

9



The Distributed TSP Pipeline Posting Points



Mission is Launched

```

uTimerScript
            
```

Shoreside

```

pPointAssign
            
```

Vehicles

```
pGenPath
```

→

```
Helm/Waypoint
```

henry

```
pGenPath
```

→

```
Helm/Waypoint
```

gilda

```

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
```

Michael Benjamin ©2024


MIT Dept of Mechanical Engineering

10

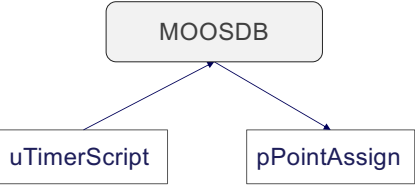
5



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.

Michael Benjamin ©2024
MIT Dept of Mechanical Engineering

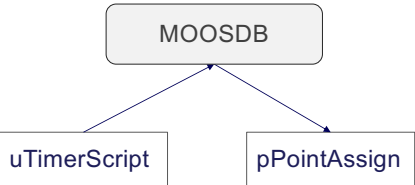
11



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.

Michael Benjamin ©2024
MIT Dept of Mechanical Engineering

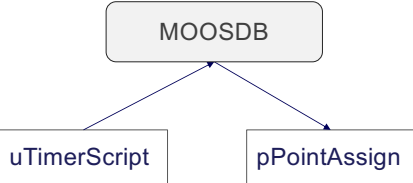
12



The Distributed TSP Pipeline Posting Points



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



```

graph TD
    MOOSDB[MOOSDB] --> uTimerScript[uTimerScript]
    MOOSDB --> pPointAssign[pPointAssign]
            
```

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

Michael Benjamin ©2024
MIT Dept of Mechanical Engineering

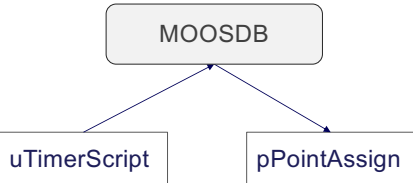
13



The Distributed TSP Pipeline Posting Points



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



```

graph TD
    MOOSDB[MOOSDB] --> uTimerScript[uTimerScript]
    MOOSDB --> pPointAssign[pPointAssign]
            
```

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 = ...
}
            
```

Michael Benjamin ©2024
MIT Dept of Mechanical Engineering

14



The Distributed TSP Pipeline Posting Points



Mission is Launched

↓

uTimerScript

→

pPointAssign

Shoreside

Vehicles

pGenPath

Helm/Waypoint

henry

pGenPath

Helm/Waypoint

gilda

```

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


Michael Benjamin ©2024

MIT Dept of Mechanical Engineering

15



The Distributed TSP Pipeline Posting Points



Mission is Launched

↓

uTimerScript

→ ?

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
    
```


Michael Benjamin ©2024

MIT Dept of Mechanical Engineering

16



The Distributed TSP Pipeline Posting Points



How do we know that uTimerScript has done its job?

Mission is Launched

↓

uTimerScript → ?

```

uTimerScript shoreside 0/0(500)
-----
Current Script Information:
  Elements: 102(102)
  Reinitss: 0
  Time Warp: 1
Time Warp All At: uTimerscript launch time (not MOOSEB 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 -25 250
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=-103.55,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
                
```


```

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


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

Michael Benjamin ©2024
MIT Dept of Mechanical Engineering

17



The Distributed TSP Pipeline Posting Points



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:

Mission is Launched

↓

uTimerScript → ?

```

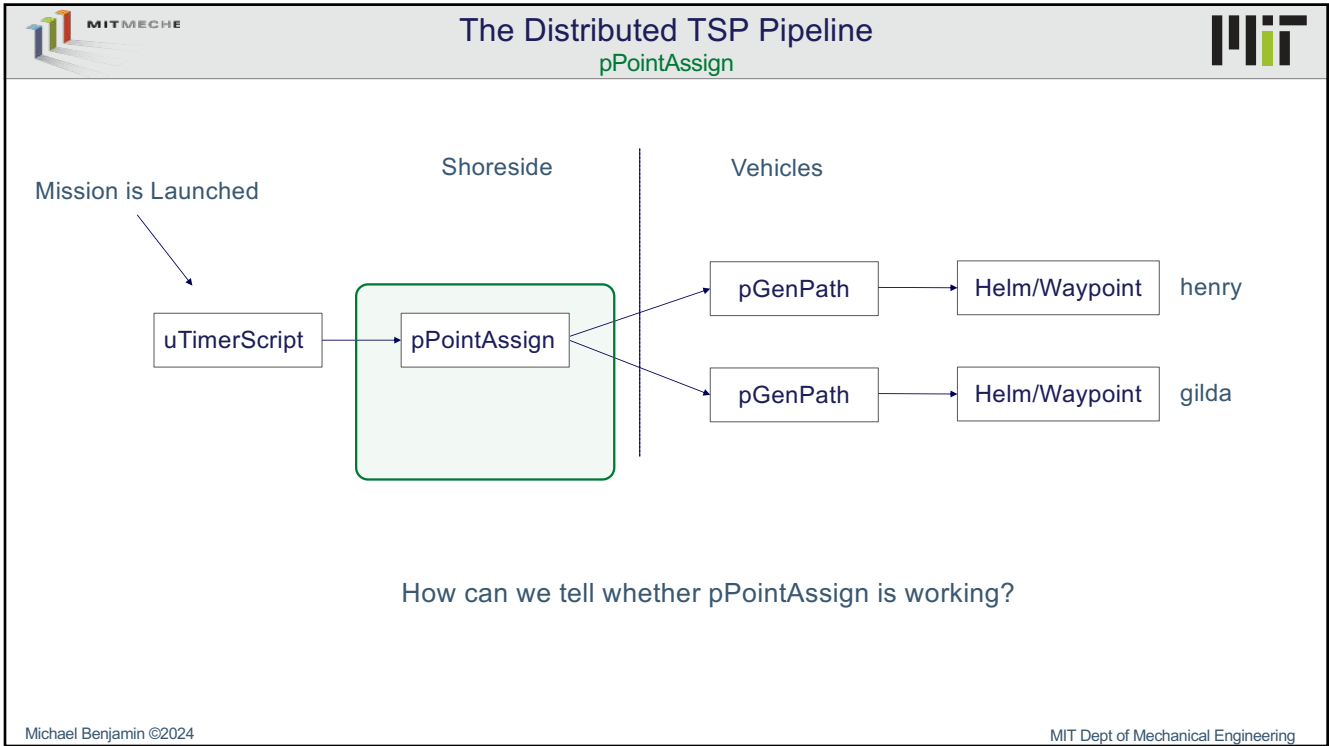
$ aloggrep SHORESIDE.alog VISIT_POINT
#####
%% LOG FILE: SHORESIDE.alog
%% FILE OPENED ON Wed Dec 31 19:00:00 1969
%% LOGSTART 6470836882.04
2.747 VISIT_POINT uTimerScript firstpoint
2.747 VISIT_POINT uTimerScript x=174.935,y=-171.595,id=1
2.747 VISIT_POINT uTimerScript x=-18.61,y=-119.68,id=2
2.747 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=-103.55,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
                
```

```

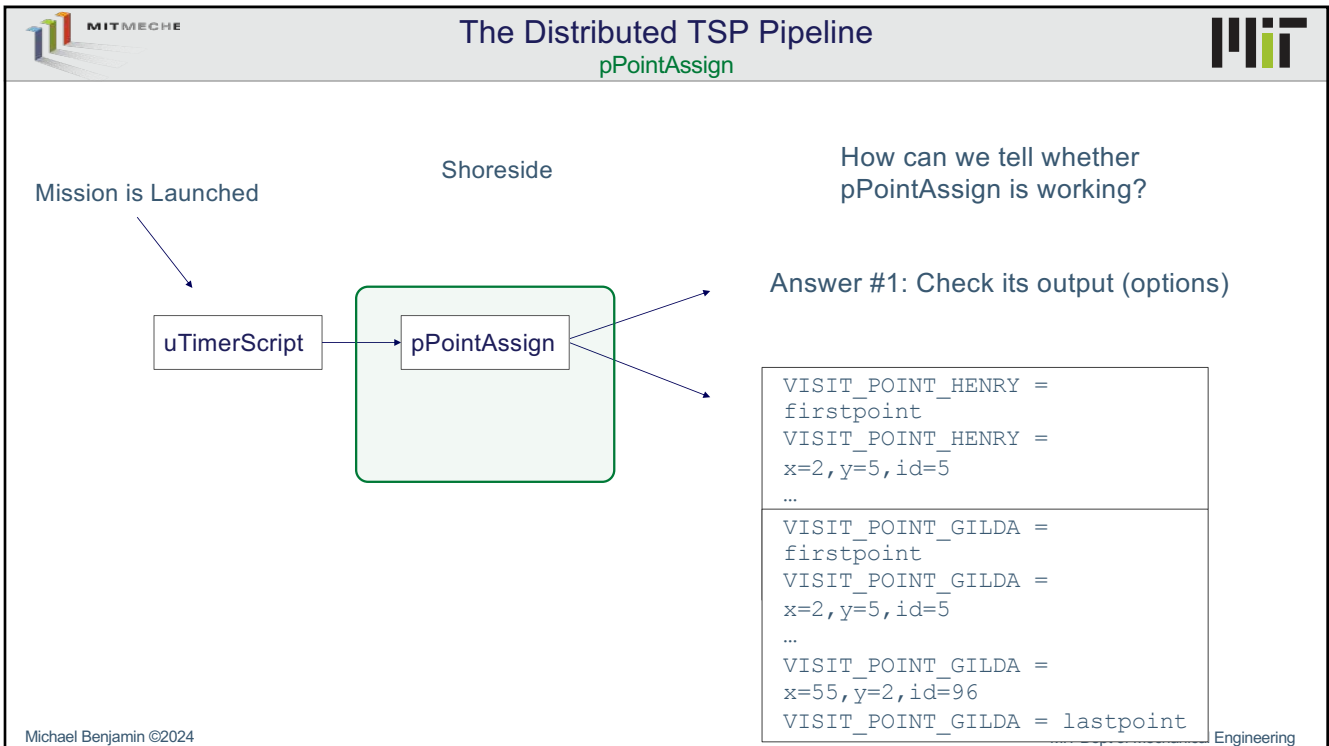
VISIT_POINT = firstpoint
VISIT_POINT =
x=2,y=5,id=0
...
VISIT_POINT =
x=55,y=2,id=99
VISIT_POINT = lastpoint
                
```

Michael Benjamin ©2024
MIT Dept of Mechanical Engineering

18



19



20

MITMECHE

The Distributed TSP Pipeline

pPointAssign

MIT

Mission is Launched

Shoreside

uTimerScript

pPointAssign

How can we tell whether pPointAssign is working?

Answer #1: Check its output (options)

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

Michael Benjamin ©2024

MIT Dept of Mechanical Engineering

21

MITMECHE

The Distributed TSP Pipeline

pPointAssign

MIT

Mission is Launched

Shoreside

uTimerScript

pPointAssign


How can we tell whether pPointAssign is working?

Answer #2: Monitor its internal state using AppCasting

Michael Benjamin ©2024


MIT Dept of Mechanical Engineering

22

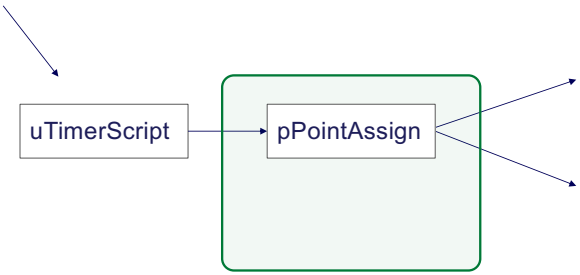


The Distributed TSP Pipeline

pPointAssign



Mission is Launched



Shoreside


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

Michael Benjamin ©2024


MIT Dept of Mechanical Engineering

25

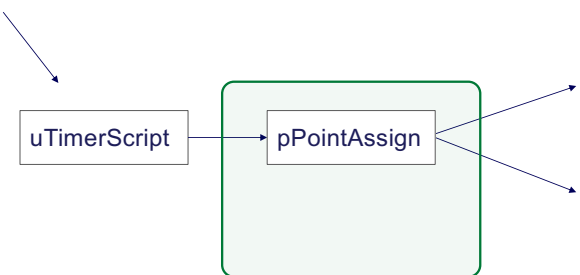


The Distributed TSP Pipeline

pPointAssign



Mission is Launched



Shoreside


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


Michael Benjamin ©2024

MIT Dept of Mechanical Engineering

26



The Distributed TSP Pipeline



pPointAssign

Mission is Launched

```

graph LR
    A[uTimerScript] --> B[pPointAssign]
    subgraph Shoreside
        B
    end
        
```

Example AppCasting output for pPointAssign

Node	AC	CW	RW	App	AC	CW	RW
shoreside	188	0	0	uTimerScript	13	0	0
				pMarineViewer	8	0	0
				pRealm	6	0	0
				uProcessWatch	12	0	0
				pPointAssign	137	0	0
				uFldShoreBroker	5	0	0
				pHostInfo	7	0	0


```

-----
pPointAssign shoreside                               0/0(2026)
-----
Total Points Received: 100
-----
Most Recent Events (8):
-----
[0.52]: VISIT_POINT_HENRY:x=126.3,y=-155.1,id=100
[0.52]: VISIT_POINT_HENRY:x=109.9,y=-116.3,id=99
[0.52]: VISIT_POINT_HENRY:x=89.9,y=-43.4,id=98
[0.52]: VISIT_POINT_HENRY:x=129,y=-130.7,id=87
[0.52]: VISIT_POINT_HENRY:x=130.6,y=-48.7,id=86
[0.52]: VISIT_POINT_HENRY:x=135.5,y=-152.6,id=82
[0.52]: VISIT_POINT_HENRY:x=105.5,y=-46.1,id=79
[0.52]: VISIT_POINT_HENRY:x=180.8,y=-162.3,id=77
        
```


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

Michael Benjamin ©2024
MIT Dept of Mechanical Engineering

27



The Distributed TSP Pipeline



Sending Points to the Vehicles

Mission is Launched

```

graph LR
    A[uTimerScript] --> B[pPointAssign]
    subgraph Shoreside
        B
    end
        
```

```

graph LR
    B --> C1[pGenPath]
    B --> C2[pGenPath]
    C1 --> D1[Helm/Waypoint]
    C2 --> D2[Helm/Waypoint]
    D1 --- E1[henry]
    D2 --- E2[gilda]
        
```

```

VISIT_POINT_HENRY = firstpoint
VISIT_POINT_HENRY = x=2,y=5,id=5
...
VISIT_POINT_HENRY = x=55,y=2,id=96
VISIT_POINT_HENRY = lastpoint

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
        
```

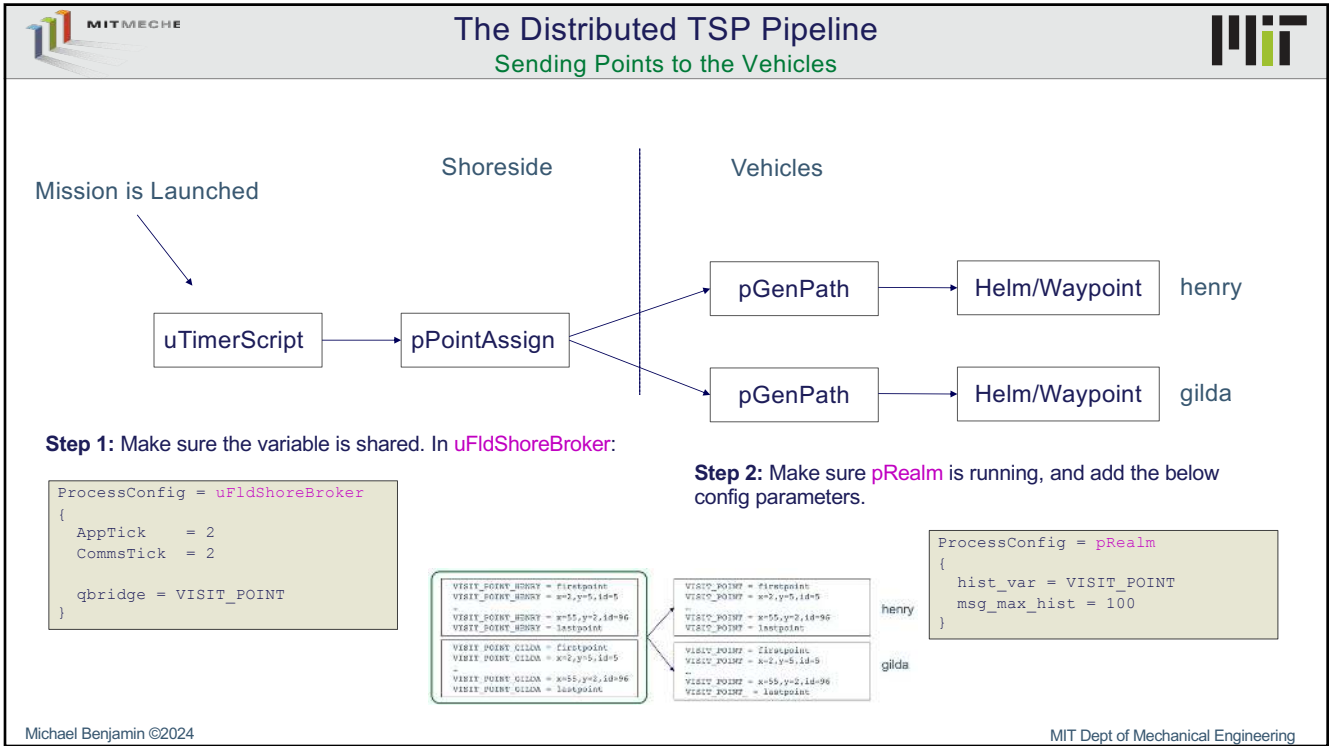
```

VISIT_POINT = firstpoint
VISIT_POINT = x=2,y=5,id=5
...
VISIT_POINT = x=55,y=2,id=96
VISIT_POINT = lastpoint

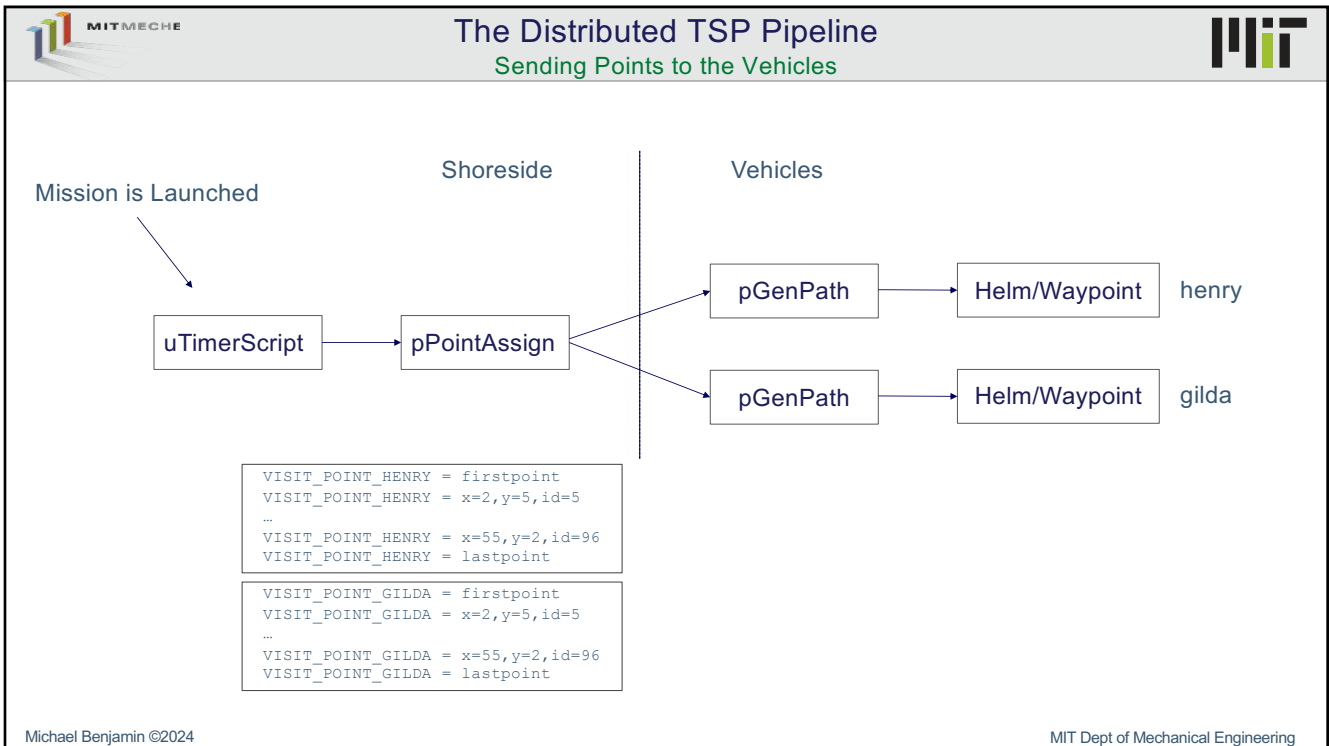
VISIT_POINT = firstpoint
VISIT_POINT = x=2,y=5,id=5
...
VISIT_POINT = x=55,y=2,id=96
VISIT_POINT = lastpoint
        
```

Michael Benjamin ©2024
MIT Dept of Mechanical Engineering


28




29



30



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);
```

Michael Benjamin ©2024MIT Dept of Mechanical Engineering

31



END

Michael Benjamin ©2024MIT Dept of Mechanical Engineering

32