

Python网络仿真API实验

1. 实验目的

本实验旨在深入理解和掌握多无人机系统的网络仿真API应用。通过搭建SITL (Software In The Loop) 仿真环境，学习如何使用NetSimAPI实现多架无人机之间的协同控制与数据共享。实验将使学习者熟悉以下关键概念和技能：

- 理解多无人机系统中网络通信的基本原理和应用场景
- 掌握NetSimAPI网络仿真接口的设计与实现方法
- 学习多线程编程在无人机通信中的应用
- 掌握基于PX4MavCtrlV4的MAVLink通信机制
- 实现跨平台 (Windows/Linux) 的无人机协同通信机制
- 学习无人机编队控制和追踪算法的实现方法

2. 实验要求

此编写实验目的

- 软件要求：Windows 10及以上版本；RflySim工具链^[1]。
- 硬件要求：笔记本/台式电脑1台^[2]。

3. 实验地址

例程目录：

[\[安装目录\]\RflySimAPIs\9.RflySimComm\0.ApiExps\e1.RflyNetAPIExps\6.PythonNetSimAPI](#)

[AddCopterArrayTest.py](#)：添加无人机阵列测试脚本，演示如何向网络中添加无人机节点

[Python38Run.bat](#)：RflySim Python运行脚本，提供Python3.8环境及相关库支持

[SITLRun4MavlinkFull.bat](#)：启动4机SITL仿真环境脚本，启动QGC地面站及4个CopterSim和RflySim3D软件

[UAV1Ctrl.py](#)：1号无人机控制程序，作为主控无人机程序，负责协调其他无人机

[UAV2Ctrl.py](#)：2号无人机控制程序，接收其他无人机数据并追踪1号飞机

[UAV3Ctrl.py](#)：3号无人机控制程序，接收其他无人机数据并追踪2号飞机

[UAV4Ctrl.py](#)：4号无人机控制程序，接收其他无人机数据并追踪3号飞机，同时控制UE4视角

[UavPythonRunALL.bat](#)：批量启动所有无人机控制程序脚本，一键启动所有无人机控制脚本

4. 实验内容或步骤

4.1 仿真初始化

双击运行 [SITLRun4MavlinkFull.bat](#) 文件，等待仿真环境初始化完成。脚本将会启动 1 个 QGC 地面站，4 个 CopterSim、1 个 RflySim3D 软件，等待CopterSim软件下侧日志栏必须打印出 `GPS 3D fixed & EKF initialization finished` 字样代表初始化完成。如下图所示：



4.2 运行控制程序

双击运行Python38Run.bat，在其中输入"python UAV1Ctrl.py"。

```
C:\windows\system32\cmd.exe x + v
Python3.8 environment has been set with openCV+pymavlink+numpy+pyulog etc.
You can use pip or pip3 command to install other libraries
Put Python38Run.bat into your code folder
Use the command: 'python XXX.py' to run the script with Python

E:\summerwork\work1\9.RflySimComm\0.ApiExps\6.PythonNetSimAPI>python UAV1Ctrl.py|
```

双击运行Python38Run.bat，在其中输入"python UAV2Ctrl.py"。

```
C:\windows\system32\cmd.exe x + v
Python3.8 environment has been set with openCV+pymavlink+numpy+pyulog etc.
You can use pip or pip3 command to install other libraries
Put Python38Run.bat into your code folder
Use the command: 'python XXX.py' to run the script with Python

E:\summerwork\work1\9.RflySimComm\0.ApiExps\6.PythonNetSimAPI>python UAV2Ctrl.py|
```

双击运行Python38Run.bat，在其中输入"python UAV3Ctrl.py"。

```
C:\windows\system32\cmd.exe x + v
Python3.8 environment has been set with openCV+pymavlink+numpy+pyulog etc.
You can use pip or pip3 command to install other libraries
Put Python38Run.bat into your code folder
Use the command: 'python XXX.py' to run the script with Python

E:\summerwork\work1\9.RflySimComm\0.ApiExps\6.PythonNetSimAPI>python UAV3Ctrl.py|
```

双击运行Python38Run.bat，在其中输入"python UAV4Ctrl.py"。

```
C:\windows\system32\cmd.exe x + v
Python3.8 environment has been set with openCV+pymavlink+numpy+pyulog etc.
You can use pip or pip3 command to install other libraries
Put Python38Run.bat into your code folder
Use the command: 'python XXX.py' to run the script with Python

E:\summerwork\work1\9.RflySimComm\0.ApiExps\6.PythonNetSimAPI>python UAV4Ctrl.py|
```

上述Step2-step5，可以通过一键启动脚本 `UavPythonRunALL.bat` 进行运行。

4.3 运行日志查看

观察命令行窗口，可以看到飞机的信息被打印。

```

C:\Python\Python\Python x + -
Check if CopterSia 3D Fixed...
CopterSia/P3M 3D Fixed, ready to fly.
Start Offboard Send!
收到其他三个飞机的数据
新位置数据接收飞机4
1 2号飞机的仿真时间(单位s)、通信延迟(单位ms)、全局坐标(x,y,z 单位m)
1号飞机, 仿真时间: 44.453 通信延迟: 0.0 全局坐标xyz: (0.85215841856928547, 0.47828492512186649, -0.155448232428557)
2号飞机, 仿真时间: 41.433 通信延迟: 0.0 全局坐标xyz: (-0.0024933114099981247, 2.429669861405283, -7.92566180096621)
3号飞机, 仿真时间: 38.659 通信延迟: 0.0 全局坐标xyz: (2.831186138668882, -0.047556482211575965, -8.172871921948955)
休眠一秒
1 2号飞机的仿真时间(单位s)、通信延迟(单位ms)、全局坐标(x,y,z 单位m)
1号飞机, 仿真时间: 45.656 通信延迟: 0.0 全局坐标xyz: (0.94681785899722484, 0.86418915065818511, -8.16194838626595)
2号飞机, 仿真时间: 42.676 通信延迟: 1.0042195551757812 全局坐标xyz: (0.004195825151802784, 2.835813131158064, -7.93316829039555)
3号飞机, 仿真时间: 39.659 通信延迟: 0.0 全局坐标xyz: (2.8453994153468396, -0.83963835624485848, -8.1767481672861734)
清除传感器是否变化
尝试重新接收飞机
起飞到十米高
1号飞机到达设定高度, 且3号飞机已经启动
开始接收3号飞机

C:\Python\Python\Python x + -
Check if CopterSia 3D Fixed...
CopterSia/P3M 3D Fixed, ready to fly.
Start Offboard Send!
收到其他三个飞机的数据
1 2号飞机的仿真时间(单位s)、通信延迟(单位ms)、全局坐标(x,y,z 单位m)
1号飞机, 仿真时间: 41.696 通信延迟: 0.0 全局坐标xyz: (-0.8022804814318286626, 2.8295489139571687, -7.928569889890438)
2号飞机, 仿真时间: 38.677 通信延迟: 0.0 全局坐标xyz: (2.8338258862728812, -0.8476548494818183, -8.17289383533319)
3号飞机, 仿真时间: 35.648 通信延迟: 0.0 全局坐标xyz: (2.8485599766247967, 1.99848857373987, -7.929161941672831)
休眠一秒
1 2号飞机的仿真时间(单位s)、通信延迟(单位ms)、全局坐标(x,y,z 单位m)
1号飞机, 仿真时间: 42.696 通信延迟: 0.0 全局坐标xyz: (0.80421614545878332279, 2.835895248278532, -7.932239528928898)
2号飞机, 仿真时间: 39.696 通信延迟: 0.0 全局坐标xyz: (2.844584393336450969, -0.8398868772812474, -8.1748381378897257)
3号飞机, 仿真时间: 36.668 通信延迟: 0.0 全局坐标xyz: (2.84243711345446097, 2.88889739664976376, -7.93437922229611)
清除传感器是否变化
尝试重新接收飞机
起飞到十米高
所有飞机到达设定高度
飞机1开始以3米/s向北飞
飞机2开始以3米/s向东飞
飞机3开始以3米/s向北飞

```



5. 关键知识点

本实验的整体思路是构建一个多无人机系统间的网络仿真框架，通过NetSimAPI实现无人机间的数据交换与协同控制。整个系统采用分布式控制模式，每架无人机都有独立的控制程序，通过UDP网络协议进行数据通信。

I AddCopterArrayTest.py 解析:

该文件是一个简单的测试脚本，用于演示如何向网络中添加无人机节点，主要包括：

```
1 import time
2 import math
3 import sys
4
5 import PX4MavCtrlV4
6 import NetSimAPIV4
7
8 #Create a new MAVLink communication instance, UDP sending port (CopterSim's
9 receiving port) is 20100
10 # 创建#1号飞机的通信实例，和CopterSim#1号相连
11 mav = PX4MavCtrlV4.PX4MavCtrlV4(1)
12 net = NetSimAPIV4.NetSimAPI(mav)
13
14 net.netAddUavSendList([2,100,900,901])
15 net.netAddUavSendList([6,107,909,1000])
16 print(net.netSendIDList)
17 print(net.netSendStarList)
18 print(net.netSendMaskList)
```

这段代码演示了如何创建MAVLink通信实例，并向网络中添加无人机节点。通过netAddUavSendList方法可以添加无人机的ID列表、星型拓扑列表和掩码列表。

I UAV控制程序解析:

1. [UAV1Ctrl.py](#): 主控无人机程序，负责协调其他无人机。主要功能包括：

- 初始化MAVLink通信实例，连接到指定编号的无人机
- 配置网络转发，将本机数据转发到其他无人机端口
- 监听其他无人机数据并解析
- 执行基本的飞行控制任务，如起飞、移动等

```
1 # 将本机数据转发给 '224.0.0.10' 下的 60002,60003,60004系列端口
2 # 这几个端口目前会被#2 #3 #4号飞机订阅，因此实际上是发给了#2 #3 #4号飞机
3 # 如果是组网仿真的话，先要发给组网程序的指定IP和端口，再转发到各飞机监听端口
4 net.enNetForward([60002,60003,60004], '224.0.0.10')
5
6 # 开始监听所有发给60001端口（目前协议里面对应#1号飞机）的数据
7 net.StartNetRec(60001, '224.0.0.10')
```

2. [UAV2Ctrl.py](#)/[UAV3Ctrl.py](#)/[UAV4Ctrl.py](#): 追踪无人机程序，实现对其他无人机的追踪功能。主要功能包括：

- 初始化通信实例并设置网络转发

- 接收并解析其他无人机的状态数据
- 实现追踪算法，保持与其他无人机的相对位置关系

```
1 print("开始追踪1号飞机")
2 while True:
3     targPos=uav1.uavGlobalPos
4     Curpos = mav.uavGlobalPos
5
6     # 比例控制，增益为1
7     dVx = 1*(targPos[0] - 1 - Curpos[0]) # 始终在1号飞机后1米位置
8     dVy = 1*(targPos[1] - Curpos[1])
9     dVz = 1*(targPos[2] - Curpos[2])
10
11     if abs(dVx)<0.2:
12         dVx=0
13
14     if abs(dVy)<0.2:
15         dVy=0
16
17     if abs(dVz)<0.2:
18         dVz=0
19
20
21     mav.SendVelNED(dVx, dVy, dVz, 0)
22
23     # 睡眠一下，使得更新率接近30Hz
24     time.sleep(0.03)
```

这些程序共同构成了一个完整的多无人机协同控制系统，实现了数据的接收、解析和协同控制功能。

6. 参考资料

1. RflySim官方文档：<https://rflysim.com/doc/zh/>
2. PX4 官方文档：<https://docs.px4.io/>
3. MAVLink 协议规范：<https://mavlink.io/en/>

7. 常见问题

Q1: 运行程序时出现"无法找到NetSimAPIV4模块"错误?

A1: 确保将必要的API文件（[NetSimAPIV4.py](#)、[PX4MavCtrlV4.py](#)等）文件放在与UAV控制程序相同的目录下，或将其放置在Python解释器能够找到的路径中。如果是在分布式环境中运行，需要确保目标机器上也存在这些API文件。

Q2: 无人机之间无法接收到彼此的数据?

A2: 首先确认SITL仿真环境已正确启动，各CopterSim实例正常运行。检查网络连接是否正常，确保UDP端口和多播地址设置正确。另外，检查网络转发设置是否正确（[enNetForward](#)），确保数据发送端口与接收端口匹配。

Q3: 在Linux/Ubuntu环境下运行出现问题?

A3: 确保Python环境配置正确，相关依赖库已安装。在Linux环境下可能需要调整某些路径设置，同时确保[NetSimAPIV4.py](#)和其他必要的API文件都已复制到Linux系统中。检查权限设置，确保脚本具有执行权限。

Q4: 无人机追踪效果不稳定或振荡严重?

A4: 这通常是由于控制参数设置不当造成的。可以尝试调整比例控制器的增益参数，减小增益值以降低系统响应速度，从而减少振荡。同时，检查时间步长设置是否合适，确保控制频率稳定。

1. <https://rflysim.com/> ↩

2. 推荐配置请见: <https://rflysim.com/doc/zh/HowToInstall.pdf> ↩