智能无人集群系统开发与实践

基于RflySim平台的全栈开发案例

第9讲　通信协议与集群组网

大纲

1. 实验平台配置
2. 关键接口介绍
3. 基础实验案例（免费版）
4. 进阶接口实验（个人版）
5. 进阶案例实验（集合版）
6. 扩展案例（完整版）
7. 小结

1.安装方法

- 1.1 需要安装的组件
- • Visual Studio 2017（体验版和完整版都需要安装）
- • 为MATLAB配置C++ 编译器（体验版和完整版都需要安装）
- • Matlab 2023a*（高级完整版安装）

下面介绍Visual Studio 2017的安装方法（需要联网）： 在本平台中，已经放置了Visual Studio 2017的安装包

1.安装方法

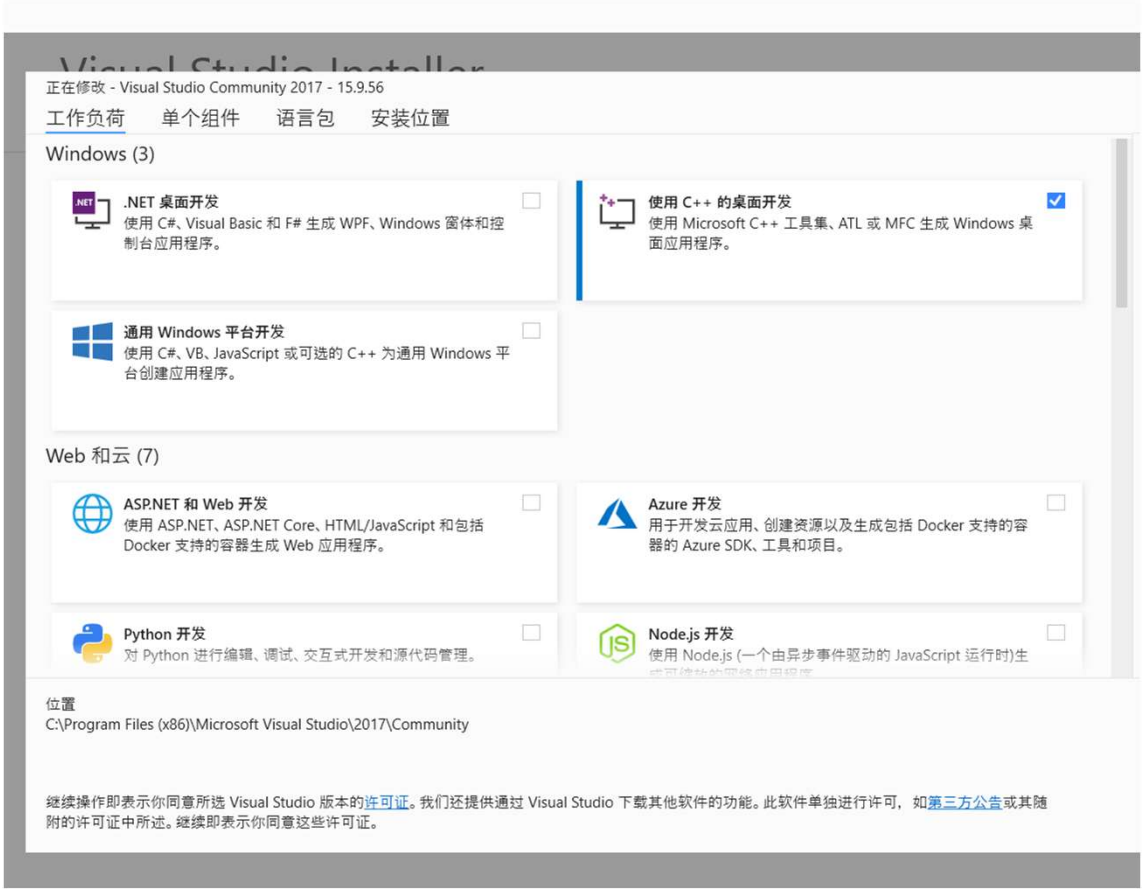
- 1.2 Visual Studio 2017的安装方法
- • 首先，我们可以打开平台安装的位置，找到*:\PX4PSP\RflySimAPIs此处位置，此处放置的是平台中的一些例程以及软件的安装包
- • 之后，我们可以打开第四章的内容，找到基础版的例程，4.RflySimModel\1.BasicExps，我们可以在其中找到名为VS2017Installer的文件夹，其中便是Visual Studio 2017的安装包。

在线安装步骤（需联网）如下：双击“RflySimAPIs\SimulinkControlAPI\VS2017Installer\vs_community2017.exe”

1.安装方法

1.2 Visual Studio 2017的安装方法

- 安装Visual Studio 2017 (也可以用其他版本，MATLAB能识别即可)。
- 后续课程很多地方都需要用到Visual Studio编译器，例如MATLAB S-Function Builder模块的使用、Simulink自动生成C/C++模型代码等
- 本课程内容只需勾选右图的“C++的桌面开发”即可。



1.安装方法

- 1.2 Visual Studio 2017的安装方法

- 注意：高版本MATLAB也可安 装VS2019，但是MATLAB只能识别到低于自己版本的Visual Studio，因此MATLAB 2017b无 法识别VS 2019。
- 注意：请不要更改VS默认安装 目录（例如装到D盘），会导致MATLAB无法识别。
- 不能使用Mingw编译器，需VS

1.安装方法

- **1.3 为MATLAB配置C++ 编译器**

• 在MATLAB的命令行窗口中输入指令“mex - setup”

• 一般来说会自动识别并安装上VS 2017编译器，如右图所示显示“MEX 配置使用‘Microsoft Visual C++ 2017’以进行编译”说明安装正确

• 若有其他编译器，本页 面还可以切换选择 VS 2013/2015等其他编译器

1.安装方法

- 1.4 Matlab 2023a的安装方法
- MATLAB 安装包下载路径：
- https://ww2.mathworks.cn/products/matlab.html

大纲

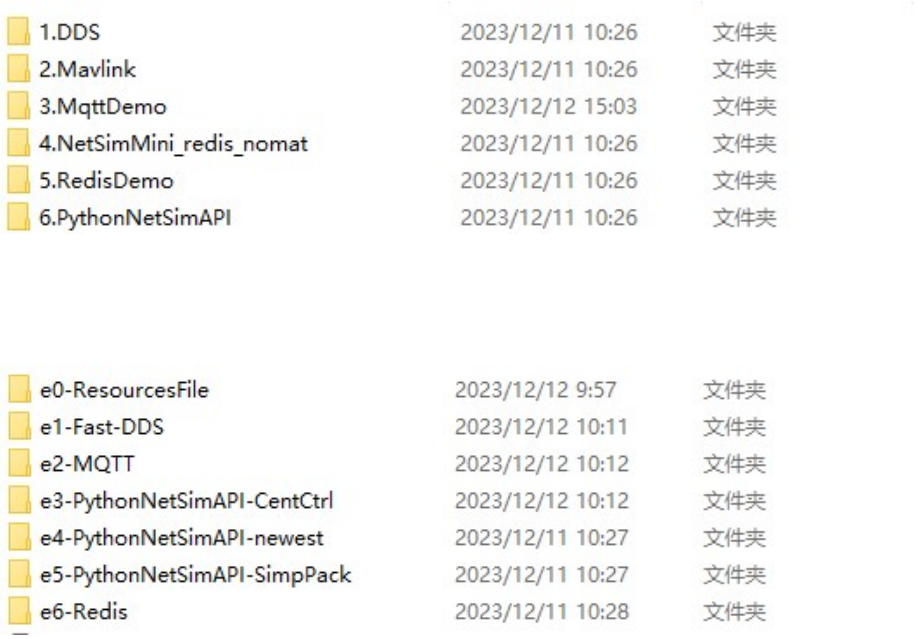
1. 实验平台配置
2. 关键接口介绍
3. 基础实验案例（免费版）
4. 进阶接口实验（个人版）
5. 进阶案例实验（集合版）
6. 扩展案例（完整版）
7. 小结

2.关键接口介绍

- 2.0 基础实验总览

包括基础功能接口“RflySimAPIs/9.RflySimComm\0.ApiExps”以及基础例程“RflySimAPIs\9.RflySimComm\1.BasicExps”

详细参见API.pdf以及Readme.pdf



2.关键接口介绍

- **2.1 DDS组网通信实验**
- 配置DDS组网需要的环境。自行创建DDS协议和收发端口实现DDS通信。

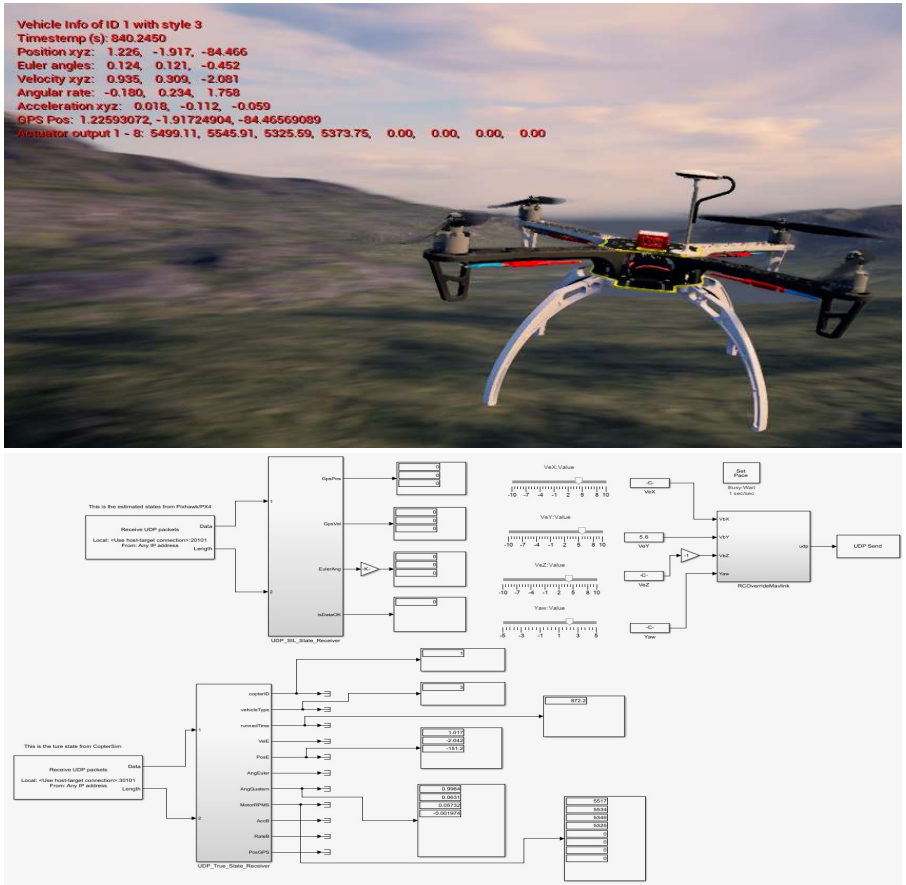
- 详细操作及实验效果见 0.ApiExps\1.DDS\readme.pdf

```
F:\work\Git\9.RflySimComm\SourceCode\e3\DDS组网通信例程\windows\Fast-DDS-python-1.2.0\Demo>python HelloWorldExample.py -
p publisher
Received {message}
Creating Start.
Creating publisher.
Writer is waiting discovery...
Publisher matched subscriber 1.f.a7.60.60.69.24.f7.0.0.0.0.0.0.1.4
Writer discovery finished...
Sending Hello World : 0
Sending Hello World : 1
Sending Hello World : 2
Sending Hello World : 3
Sending Hello World : 4
Sending Hello World : 5
Sending Hello World : 6
Sending Hello World : 7
Sending Hello World : 8
Sending Hello World : 9
```

```
F:\work\Git\9.RflySimComm\SourceCode\e3\DDS组网通信例程\windows\Fast-DDS-python-1.2.0\Demo>python HelloWorldExample.py -
p subscriber
Received {message}
Creating Start.
Creating subscriber.
Press any key to stopSubscriber matched publisher 1.f.a7.60.44.27.94.4b.0.0.0.0.0.0.1.3
Received Hello World : 0
Received Hello World : 1
Received Hello World : 2
Received Hello World : 3
Received Hello World : 4
Received Hello World : 5
Received Hello World : 6
Received Hello World : 7
Received Hello World : 8
Received Hello World : 9
Subscriber unmatched publisher 1.f.a7.60.44.27.94.4b.0.0.0.0.0.0.1.3
```

2.关键接口介绍

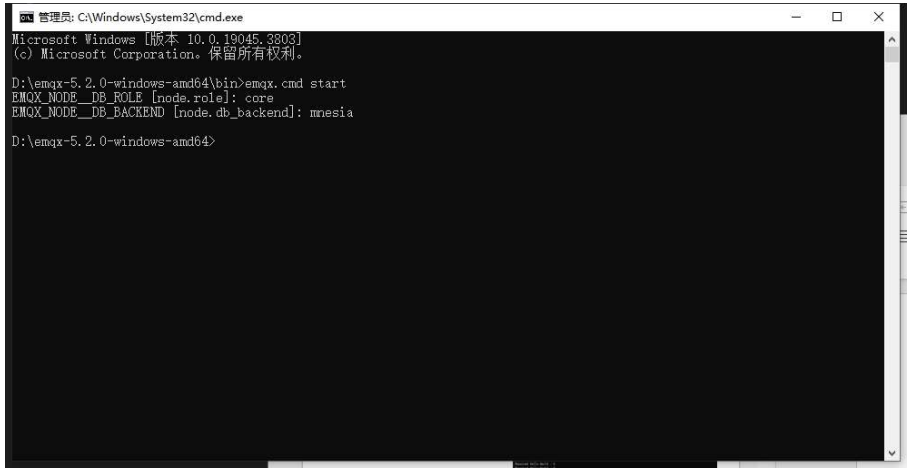
- 2.2 MAVlink通信实验
- 使用 MAVlink 通信使用不同通信模式实现飞机控制和对飞机飞行状态信息的获取。
- 详细操作及实验效果见 0.ApiExps\2.Mavlink\readme.pdf

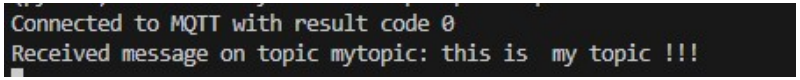


2.关键接口介绍

- **2.3 Mqtt通信实验**
- 使用mqtt开启服务器，将发布器和订阅器连接到服务器，并通过话题收发实现通信。

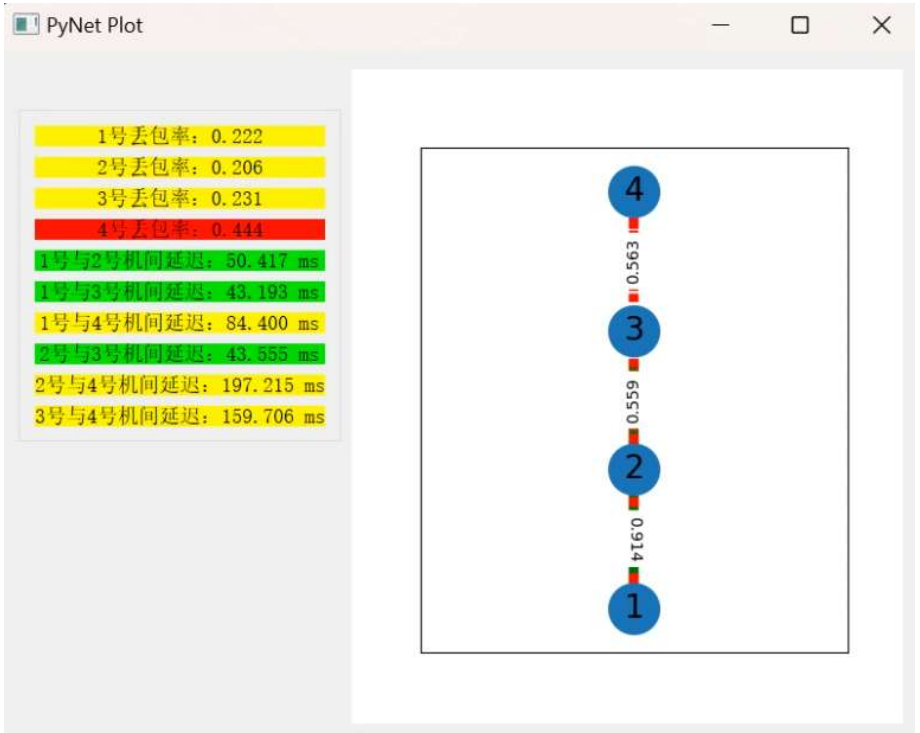
- 详细操作及实验效果见 0.ApiExps\3.MqttDemo\readme.pdf





2.关键接口介绍

- 2.4网络仿真模拟实验
- 通过创建Redis通信，模拟地形对信号传输的影响。获取无人机间的通信质量。
- 关于Redis安装及使用见 1.BasicExps\e0-ResourcesFile\Windows下Redis组网通信例程.pdf
- 详细操作及实验效果见 0.ApiExps\4.NetSimMini_redis_nomat\readme.pdf



2.关键接口介绍

- **2.5 Redis 通信模拟实验**
- 开启Redis服务器，创建 通信链接，模拟 **Redis** 通信。

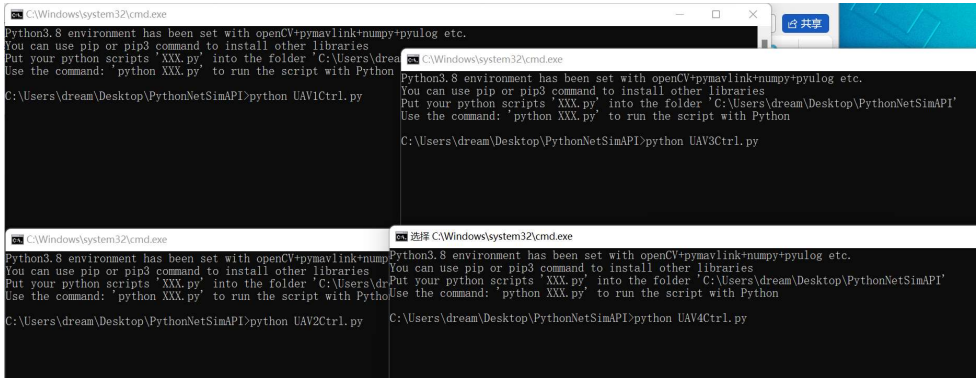
- 详细操作及实验效果见 0.ApiExps\5.RedisDemo\readme.pdf

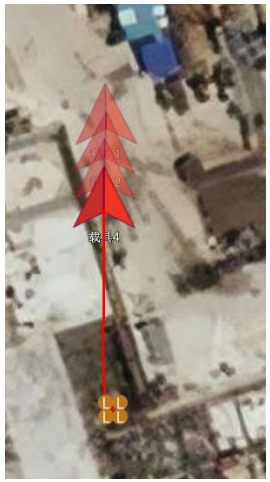
```
channel: This is a example channel
0.0001287460327148437 5
channel: This is a example channel
0.0001530647277832031 2
channel: This is a example channel
0.0001635551452636718 8
channel: This is a example channel
0.0001513957977294922
channel: This is a example channel
0.0001938343048095703
channel: This is a example channel
0.0001826286315917968 8
channel: This is a example channel
0.0001699924468994140 6
channel: This is a example channel
0.0001423358917236328
channel: This is a example channel
0.0001418590545654297
channel: This is a example channel
0.0001714229583740234 4
channel: This is a example channel
0.0001523494720458984 4
channel: This is a example channel
0.0001685619354248047
channel: This is a example channel
0.0002028942108154297
channel: This is a example channel
0.0001993179321289062 5
```

2.关键接口介绍

- **2.6 net组网实验**
- 通过向同一IP下的不同端口发送数据实现数据共享，理解通信原理。

- 详细操作及实验效果见 [0.ApiExps\6.PythonNetSimAPI\readme.pdf]





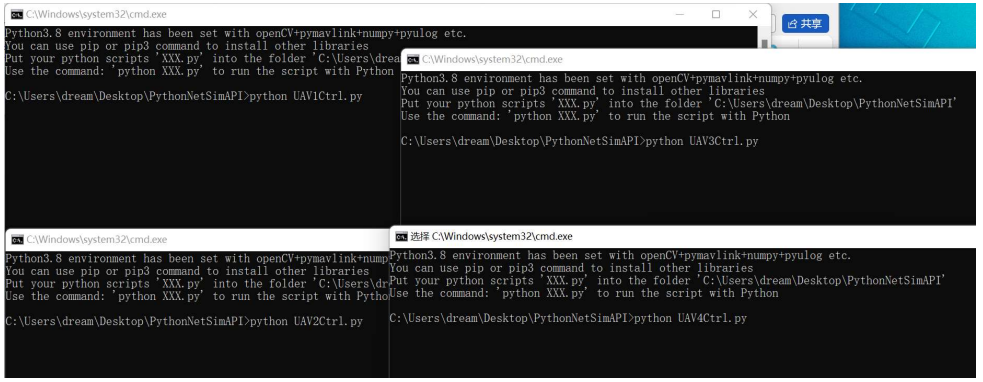
大纲

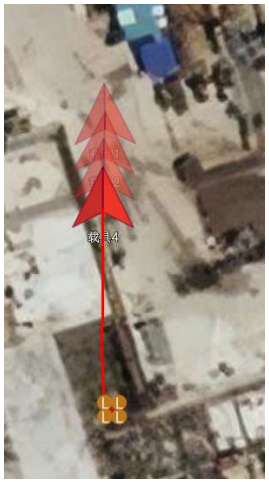
1. 实验平台配置
2. 关键接口介绍
3. 基础实验案例（免费版）
4. 进阶接口实验（个人版）
5. 进阶案例实验（集合版）
6. 扩展案例（完整版）
7. 小结

3.基础实验案例

- **3.1 fast-DDS 通信组网实验**
- 使用**fast-DDS**实现飞机间的信息交互。

- 详细操作及实验效果见 [1.BasicExps\e1-Fast-DDS\readme.pdf]

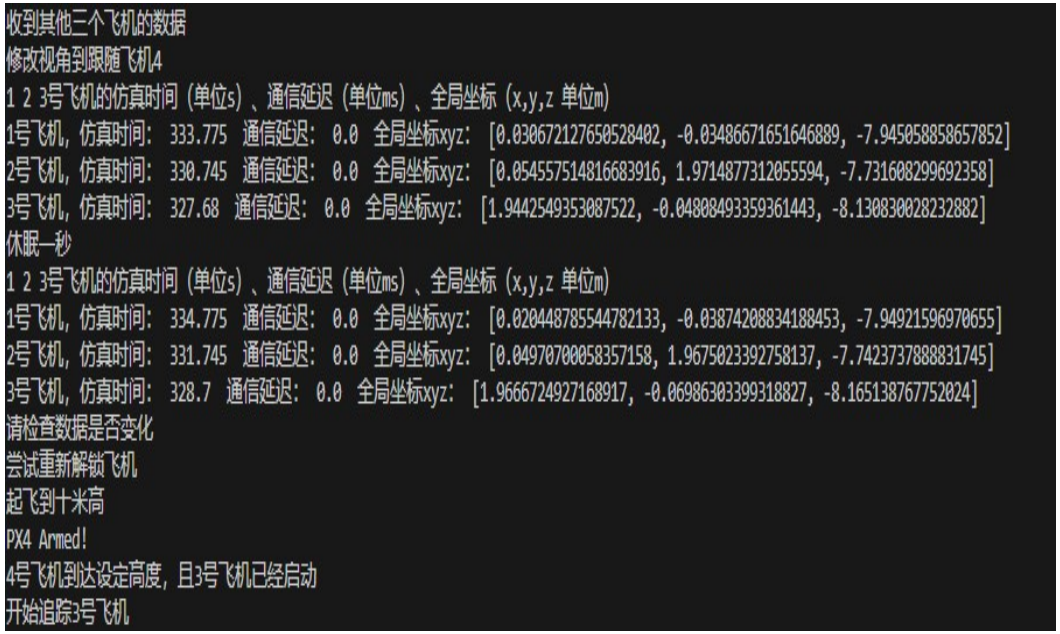




3.基础实验案例

- 3.2 MQTT多无人机控制实验
- 使用mqtt实现飞机间的信息交互，使用Mavlink实现飞机自身的控制。

- 详细操作及实验效果见 1.BasicExps\e2-MQTT\readme.pdf



3.基础实验案例

- 3.3 Net-CentCtrl飞机通信实验
- 使用Net完成飞机间的信息交互。

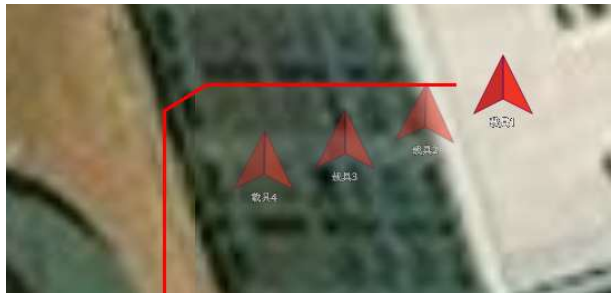
- 详细操作及实验效果见 1.BasicExps\e3-PythonNetSimAPI-CentCtrl\readme.pdf

```
收到4号飞机的数据
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）
4号飞机，仿真时间： 42.51 通信延迟： 0.0 全局坐标xyz： (0.0, 0.0, 0.0)
休眠一秒
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）
4号飞机，仿真时间： 43.53 通信延迟： 0.0 全局坐标xyz： (0.0, 0.0, 0.0)
请检查数据是否变化
尝试重新解锁飞机
起飞到十米高
PX4 Armed!
1号飞机到达设定高度，且4号飞机已经启动
飞机1开始以3米/s向北飞
飞机1停止飞行
飞机1开始以3米/s向东飞
```

```
收到1号飞机的数据
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： 46.005 通信延迟： 0.0 全局坐标xyz： (-0.017929215153247746, -0.076881639750
73925, -7.9771995483187705)
休眠一秒
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： 47.025 通信延迟： 0.0 全局坐标xyz： (-0.007656581868917378, -0.081644065769
45415, -7.994564840033439)
请检查数据是否变化
尝试重新解锁飞机
起飞到十米高
2号飞机到达设定高度，且1号飞机已经启动
开始追踪1号飞机
```

```
收到2号飞机的数据
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）
2号飞机，仿真时间： 48.43 通信延迟： 0.0 全局坐标xyz： (0.0009192207744517233, 1.9831188
8, -12.115752877215677)
休眠一秒
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）
2号飞机，仿真时间： 49.39 通信延迟： 0.0 全局坐标xyz： (0.02201995792893552, 1.977986789
-14.662516774157815)
请检查数据是否变化
尝试重新解锁飞机
起飞到十米高
3号飞机到达设定高度，且2号飞机已经启动
开始追踪2号飞机
```

```
收到3号飞机的数据
修改视角到跟随飞机4
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）
3号飞机，仿真时间： 49.515 通信延迟： 0.0 全局坐标xyz： (1.978295037388547, 0.004737986183
64, -9.33546677295139)
休眠一秒
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）
3号飞机，仿真时间： 50.54 通信延迟： 0.0 全局坐标xyz： (1.9741302652803498, 0.004282861883
7, -11.89173308807822)
请检查数据是否变化
尝试重新解锁飞机
起飞到十米高
4号飞机到达设定高度，且3号飞机已经启动
开始追踪3号飞机
```



3.基础实验案例

- 3.4 Net 飞机通信实验
- 使用Net完成飞机间的数据交互。这个例程飞机将自身信息分享到组网程序后由组网程序分发给各飞机。
- 详细操作及实验效果见 1.BasicExps\e4-PythonNetSimAPI-newest\readme.pdf

```
net.enNetForward([61000],'224.0.0.10')

print('Check if CopterSim 3D Fixed...')
while True:
    if mav.isPX4Ekf3DFixed:
        print('CopterSim/PX4 3D Fixed, ready to fly.')
        break
    time.sleep(0.5)

time.sleep(1)
print('Start Offboard Send!')
# 启用Offboard控制
mav.initOffboard()
time.sleep(1)


# 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
net.StartNetRec(60001,'224.0.0.10')
```

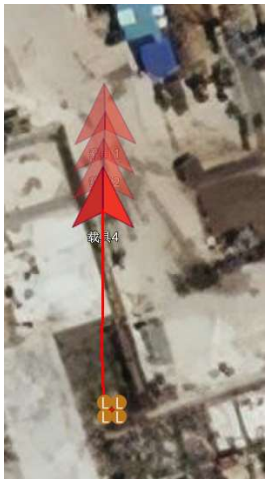
```
def receiveFromUavCommunicationDatas(self,  port:int = 61000):
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
    sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    sock.bind(('0.0.0.0', port))
    # time.sleep(3)
    mreq = struct.pack("=4sl", socket.inet_aton("224.0.0.10"), socket.INADDR_ANY)
    sock.setsockopt(socket.IPPROTO_IP,socket.IP_ADD_MEMBERSHIP,mreq)

    while True:
        buf, addr= sock.recvfrom(1024)
        cksum,cpID,sendMode,StartIdx,SendMask,TimeUnix=struct.unpack('iiiiQd',buf[0:32])
        destIds = self.getDests(SendMask, StartIdx)
        delaysandnewDestIds =[]
        curTime = time.time_ns()/1e9
        targetTimePortList=[]
        for tgID in destIds:
            targetTimePortList.append((TimeUnix+0.005,60000+tgID))# 统一添加3毫秒的延迟
        # 使用互斥锁
        self.uavsSendmutex.acquire()
        self.uavsPacketBufList.append([buf,targetTimePortList])
        self.uavsSendmutex.release()
```

3.基础实验案例

- 3.5 Net 飞机通信实验
- 使用Net完成飞机间的信息交互。
- 详细操作及实验效果见1.BasicExps\e5-PythonNetSimAPI-SimpPack\readme.pdf

```
收到4号飞机的数据
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）
4号飞机，仿真时间： 42.51 通信延迟： 0.0 全局坐标xyz： (0.0, 0.0, 0.0)
休眠一秒
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）
4号飞机，仿真时间： 43.53 通信延迟： 0.0 全局坐标xyz： (0.0, 0.0, 0.0)
请检查数据是否变化
尝试重新解锁飞机
起飞到十米高
PX4 Armed!
1号飞机到达设定高度，且4号飞机已经启动
飞机1开始以3米/s向北飞
飞机1停止飞行
飞机1开始以3米/s向东飞
```



3.基础实验案例

- **3.6.1 Redis 通信实验**

- 多次发送，多次订阅。基于Redis实现飞机间通信。

- 详细操作及实验效果见 1.BasicExps\e6-Redis\e6.1\readme.pdf

```
Check if CopterSim 3D Fixed...
CopterSim/PX4 3D Fixed, ready to fly.
Start Offboard Send!
Failsafe mode deactivated
PX4 Armed!
收到其他三个飞机的数据
2 3 4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）
2号飞机，仿真时间:  213.635  通信延迟:  0.0  全局坐标xyz:  [69.08198264406333, -0.35688943640049975, -7.7525857701989 3]
3号飞机，仿真时间:  210.63  通信延迟:  0.0  全局坐标xyz:  [68.25676685900378, -0.4029859306441932, -7.908592274856632]
4号飞机，仿真时间:  207.56  通信延迟:  0.0  全局坐标xyz:  [67.29705473668884, -0.492072955089202, -8.089055331127543]
休眠一秒
2 3 4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）
2号飞机，仿真时间:  214.63  通信延迟:  0.0  全局坐标xyz:  [69.08603385255942, -0.3520948387747673, -7.758044683942426]
3号飞机，仿真时间:  211.65  通信延迟:  0.0  全局坐标xyz:  [68.2642665538 28, -0.3942648472415595, -7.898917785358494]
4号飞机，仿真时间:  208.56  通信延迟:  0.0  全局坐标xyz:  [67.34635588414977, -0.4868220243030692, -8.09900900878181]
请检查数据是否变化
尝试重新解锁飞机
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1停止飞行
飞机1开始以3米/s向东飞
```

```python
def PublicUavData():
    while True:
        # 如果mav收到了消息
        mav.netEvent.wait()
        # double uavTimeStmp 时间戳
        # float uavAngEular[3] 欧拉角 弧度
        # float uavVelNED[3] 速度 米
        # double uavPosGPSHome[3] GPS 维度（度）、经度（度
        # double uavPosNED[3] 本地位置 米 （相对起飞点）
        # double uavGlobalPos[3] 全局位置 （相对与所有飞机
        # d6f9d = 长度104
        TimeUnix = time.time_ns()/1e9 # 使用time_ns能获取
        UAV1 = {
            "timeDelay":TimeUnix,
            "CopterID":mav.CopterID,
            "uavTimeStmp":mav.uavTimeStmp,
            "uavAngEular":mav.uavAngEular,
            "uavVelNED":mav.uavVelNED,
            "uavPosGPSHome":mav.uavPosGPSHome,
            "uavPosNED":mav.uavPosNED,
            "uavGlobalPos":mav.uavGlobalPos}
        mav.netEvent.clear()
        for uav in enNetForwardList:
            redisConnect.pub_data(uav,UAV1)
        time.sleep(0.1)

def sub_callback(channel,data):
        TimeUnix = time.time_ns()/1e9 # 使用time_ns能获取
        data["timeDelay"] = TimeUnix - data["timeDelay"]
        if data["CopterID"] == 2: # 如果收到#2号飞机数据
            if  len(UavData) < 1:
                UavData.append(data)
            else :
                UavData[0] = data
        elif data["CopterID"] == 3: # 如果收到#2号飞机数据
            if  len(UavData) < 2:
                UavData.append(data)
            else :
                UavData[1] = data
        elif data["CopterID"] == 4: # 如果收到#2号飞机数据
            if  len(UavData) < 3:
                UavData.append(data)
            else :
                UavData[2] = data
```

3.基础实验案例

- **3.6.2 Redis 通信实验**

- 使用set，get进行飞机间的数据交互。

- 详细操作及实验效果见1.BasicExps\e6-Redis\e6.2\readme.pdf

```python
def SetUavData():
    while True:
        # 如果mav收到了消息
        mav.netEvent.wait()
        # double uavTimeStmp 时间戳
        # float uavAngEular[3] 欧拉角 弧度
        # float uavVelNED[3] 速度 米
        # double uavPosGPSHome[3] GPS 维度（度）、经度（度）、高度（米）
        # double uavPosNED[3] 本地位置 米 （相对起飞点）
        # double uavGlobalPos[3] 全局位置 （相对与所有飞机的地图中心）
        # d6f9d = 长度104
        TimeUnix = time.time_ns()/1e9 # 使用time_ns能获取比time更高的精度
        UAV1 = {
            "timeDelay":TimeUnix,
            "CopterID":mav.CopterID,
            "uavTimeStmp":mav.uavTimeStmp,
            "uavAngEular":mav.uavAngEular,
            "uavVelNED":mav.uavVelNED,
            "uavPosGPSHome":mav.uavPosGPSHome,
            "uavPosNED":mav.uavPosNED,
            "uavGlobalPos":mav.uavGlobalPos}
        mav.netEvent.clear()
        redisConnect.set_data("UAV1",UAV1)
        time.sleep(0.1)

def GetCallback(UavList):
    while True:
        for uav in UavList:
            data = redisConnect.get_data(uav)
            if data != False :
                UavData[uav] = data
                TimeUnix = time.time_ns()/1e9 # 使用time_ns能获取比time更高的精度
                UavData[uav]["timeDelay"] =TimeUnix - UavData[uav]["timeDelay"]
        time.sleep(0.01)

def SetUavNet():
    thread = threading.Thread(target=SetUavData,)
    thread.start()


def GetUavNet():
    GetUavList =["UAV2","UAV3","UAV4"]
    thread = threading.Thread(target=GetCallback, args=(GetUavList,))
    thread.start()
```

3.基础实验案例

- **3.6.3 Redis 通信实验**

- 单次发送，多次订阅。基于Redis实现飞机间数据交互。

- 详细操作及实验效果见1.BasicExps\e6-Redis\e6.3\readme.pdf

```python
def PublicUavData():
    while True:
        # 如果mav收到了消息
        mav.netEvent.wait()
        # double uavTimeStmp 时间戳
        # float uavAngEular[3] 欧拉角 弧度
        # float uavVelNED[3] 速度 米
        # double uavPosGPSHome[3] GPS 维度（度）、经度（度）、高度（米）
        # double uavPosNED[3] 本地位置 米 （相对起飞点）
        # double uavGlobalPos[3] 全局位置 （相对与所有飞机的地图中心）
        # d6f9d = 长度104
        TimeUnix = time.time_ns()/1e9 # 使用time_ns能获取比time更高的精度
        UAV1 = {
            "timeDelay":TimeUnix,
            "CopterID":mav.CopterID,
            "uavTimeStmp":mav.uavTimeStmp,
            "uavAngEular":mav.uavAngEular,
            "uavVelNED":mav.uavVelNED,
            "uavPosGPSHome":mav.uavPosGPSHome,
            "uavPosNED":mav.uavPosNED,
            "uavGlobalPos":mav.uavGlobalPos}
        mav.netEvent.clear()
        redisConnect.pub_data("UAV1",UAV1)
        time.sleep(0.1)

def sub_callback(channel,data):
    UavID = channel.decode('utf-8')
    UavData[UavID] = data
    TimeUnix = time.time_ns()/1e9 # 使用time_ns能获取比time更高的精度
    UavData[UavID]["timeDelay"] =TimeUnix - data["timeDelay"]

def PubUavNet():
    thread = threading.Thread(target=PublicUavData,)
    thread.start()


def SubUavNet():
    message_type = 'message'
    channels_to_subscribe = ["UAV2", "UAV3", "UAV4"]
    thread = threading.Thread(target=redisConnect.sub_data_multiple_channels, args=(message_type,channels_to_subscribe,sub_callback,))
    thread.start()
```

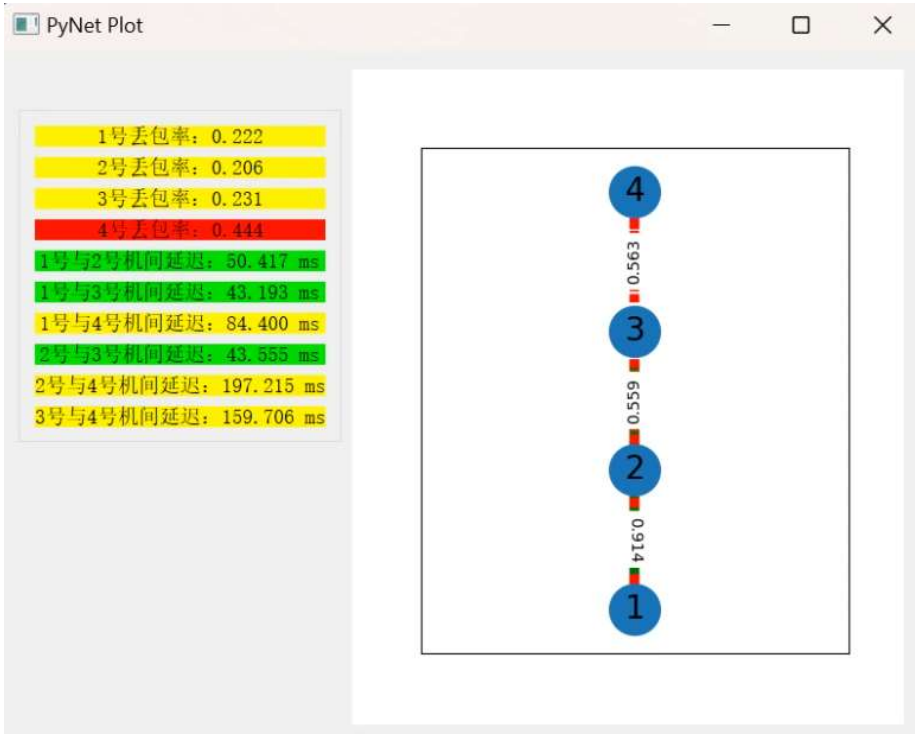
大纲

1. 实验平台配置
2. 关键接口介绍
3. 基础实验案例（免费版）
4. 进阶接口实验（个人版）
5. 进阶案例实验（集合版）
6. 扩展案例（完整版）
7. 小结

4.进阶案例实验

- **4.1 Redis 网络组网信号质量检测实验**
- 在进行仿真时，创建多节点间的数据交互，检测并返回节点间的通信质量。加深对**Redis**通信的理解。

- 详细操作及实验效果见 2.AdvExps\e0_AdvApiExps\1.NetSimMini_redis_nomat\readme.pdf



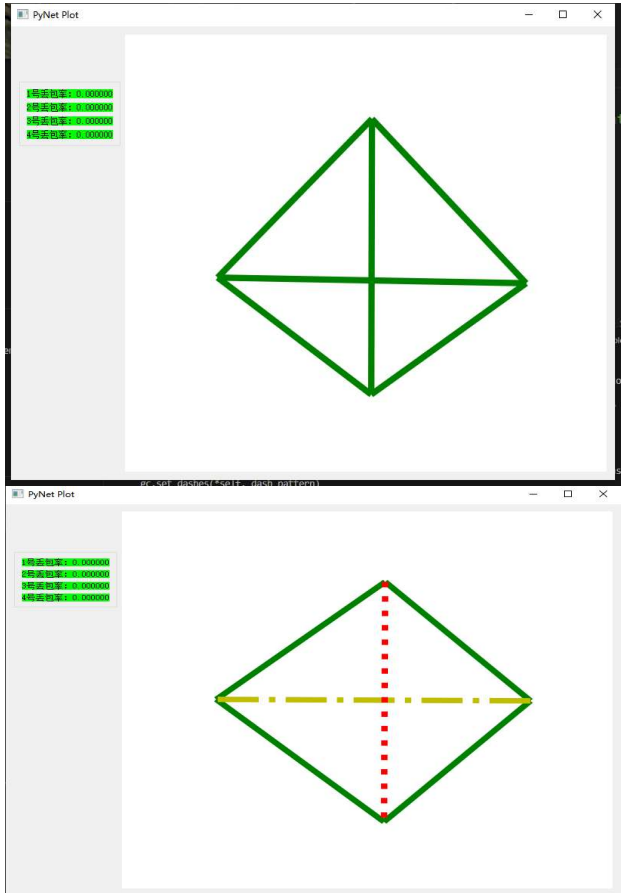
大纲

1. 实验平台配置
2. 关键接口介绍
3. 基础实验案例（免费版）
4. 进阶接口实验（个人版）
5. 进阶案例实验（集合版）
6. 扩展案例（完整版）
7. 小结

5. 进阶案例实验

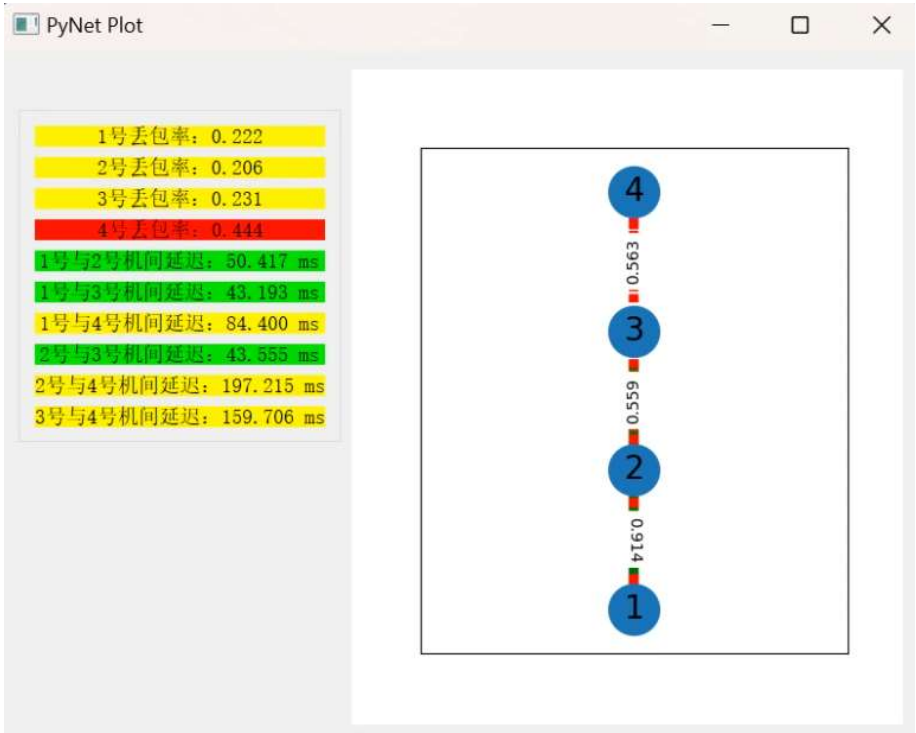
- 5.1 Net组网实验
- 本程序会将数据发送到(netSimPort,netSimIP)，经过网络仿真器中转，再根据目标ID，发往对应飞机的IP和端口。并事实检测飞机间的通信质量。

- 详细操作及实验效果见 2.AdvExps\e1-NetSim4Demo\readme.pdf



5. 进阶案例实验

- **5.2 Redis 组网信号质量检测实验**
- **程序通过Redis的set，get进行通信，**
- **详细操作及实验效果见 2.AdvExps\e2-NetSimMini_redis_mat\readme.pdf**



5. 进阶案例实验

- 5.3单机控制实验与在线检测实验
- 增加心跳检测功能，检测自身通信状态。
- 详细操作及实验效果见 2.AdvExps\e3-Python\readme.pdf

```
PX4 Armed!
port 22001
Start network serve.
[-1326379.6885100934, -278943.02584314503, -14
Send to takeoff.
1 号机延迟为: 1055ms
1 号机已经连接时间为: 2s
1 号机延迟为: 1041ms
1 号机已经连接时间为: 3s
1 号机延迟为: 1085ms
1 号机已经连接时间为: 4s
1 号机延迟为: 1167ms
1 号机已经连接时间为: 5s
1 号机延迟为: 1175ms
1 号机已经连接时间为: 6s
1 号机延迟为: 1233ms
1 号机已经连接时间为: 7s
1 号机延迟为: 1303ms
1 号机已经连接时间为: 8s
1 号机延迟为: 1350ms
1 号机已经连接时间为: 9s
1 号机延迟为: 1435ms
1 号机已经连接时间为: 10s
1 号机延迟为: 1457ms
1 号机已经连接时间为: 11s
1 号机延迟为: 1595ms
1 号机已经连接时间为: 12s
1 号机延迟为: 1672ms
1 号机已经连接时间为: 14s
1 号机延迟为: 1738ms
1 号机已经连接时间为: 15s
```

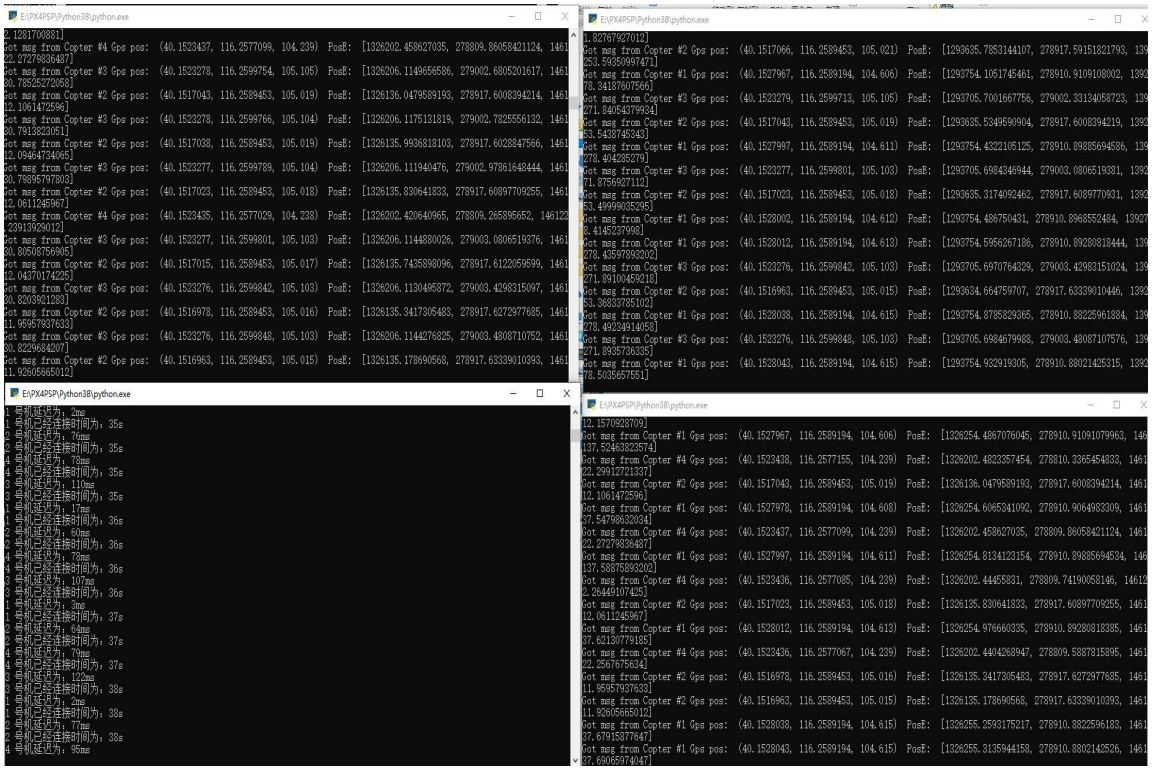
大纲

1. 实验平台配置
2. 关键接口介绍
3. 基础实验案例（免费版）
4. 进阶接口实验（个人版）
5. 进阶案例实验（集合版）
6. 扩展案例（完整版）
7. 小结

6.扩展案例

- 6.1粗粒度集群组网实验
- 通过无人机集群组网发送的数据都会发送到粗粒度组网程序监听的30000端口，然后根据粗粒度组网的规则判断能否到达目的无人机并计算丢包。

- 详细操作及实验效果见 3.CustExps\e0-NetSimMini_redis_mat\readme.pdf



大纲

1. 实验平台配置
2. 关键接口介绍
3. 基础实验案例（免费版）
4. 进阶接口实验（个人版）
5. 进阶案例实验（集合版）
6. 扩展案例（完整版）
7. 小结

7. 小结

- 本讲主要对无人机的通信创建以及飞机间的组网进行讲解，分为基础实验、进阶实验和扩展案例三部分，可以实现局域组网，无人机集群通信，不同通信架构。

如有疑问，请到https://doc.rflysim.com/查询更多信息。

RflySim更多教程

扫码咨询与交流

飞思RflySim技术交流群

谢谢！