多无人机广播缓冲区数据传输实验

1. 实验目的

本实验旨在演示如何通过RflySim平台实现多无人机间的广播数据传输。具体目标包括：

1. 掌握NetSimAPIV4网络仿真接口的基本使用方法；
2. 学习如何利用struct模块打包和解包二进制数据；
3. 理解多无人机系统中数据广播的机制和实现方式；
4. 实现无人机间缓冲区数据的发送与接收；
5. 验证分布式仿真环境中多机协同的数据通信能力。

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\4.BroadSendbuf

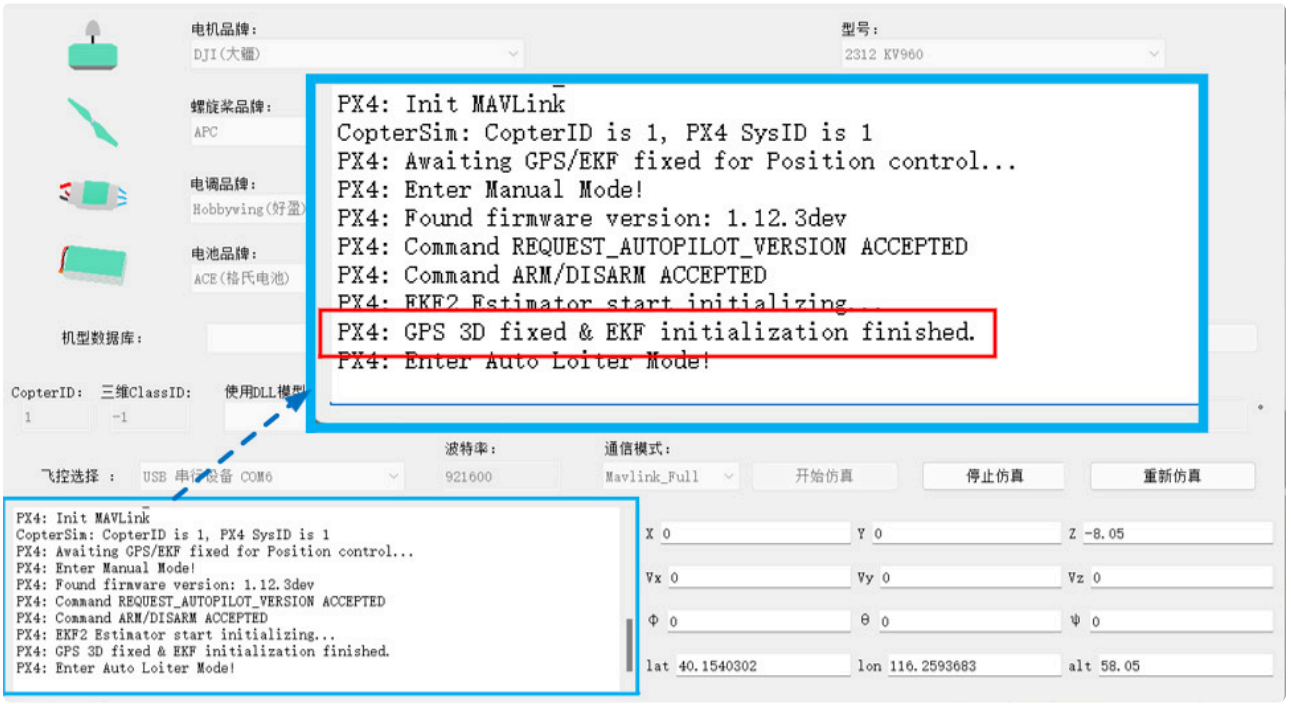
Python38Run.bat：RflySim Python运行脚本
SITLRun3MavlinkFull.bat：启动3机SITL仿真环境脚本
UAV1Ctrl.py：1号无人机控制脚本(发送方) UAV2Ctrl.py：2号无人机控制脚本(发送方)
UAV3Ctrl.py：3号无人机控制脚本(接收方) UavPythonRunALL.bat：批量启动Python控制脚本

4. 实验内容或步骤

4.1 本机实验

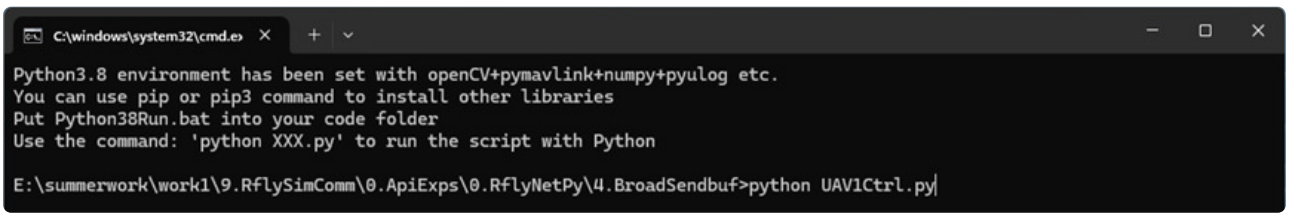
4.1.1 仿真环境初始化

双击运行SITLRun3MavlinkFull.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， 3 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：

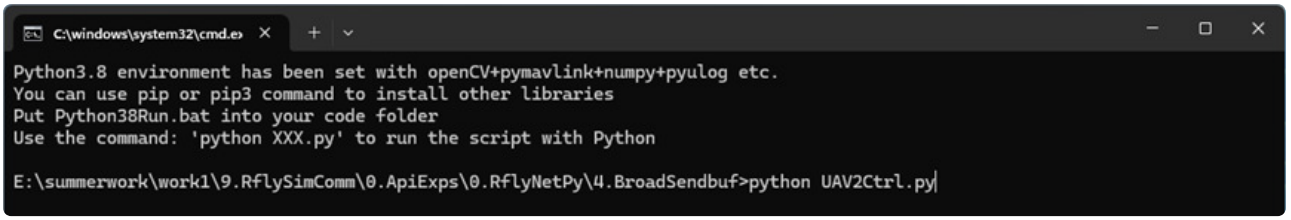


4.1.2 运行控制程序

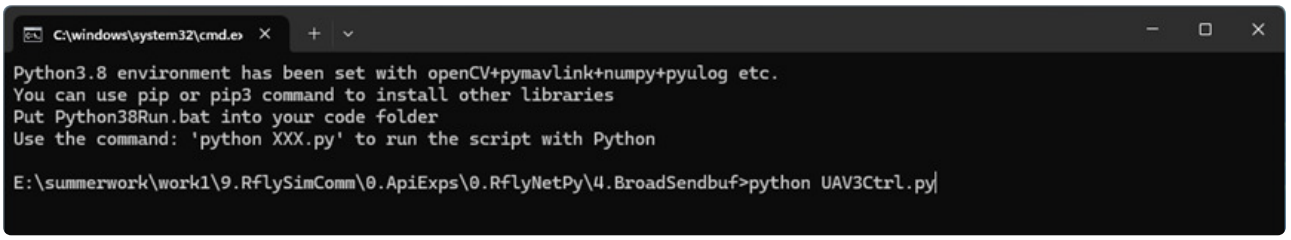
双击运行"Python38Run.bat"，在其中输入"python UAV1Ctrl.py"，启动1号飞机广播发送程序。



双击运行"Python38Run.bat"，在其中输入"python UAV2Ctrl.py"，启动2号飞机广播发送程序。



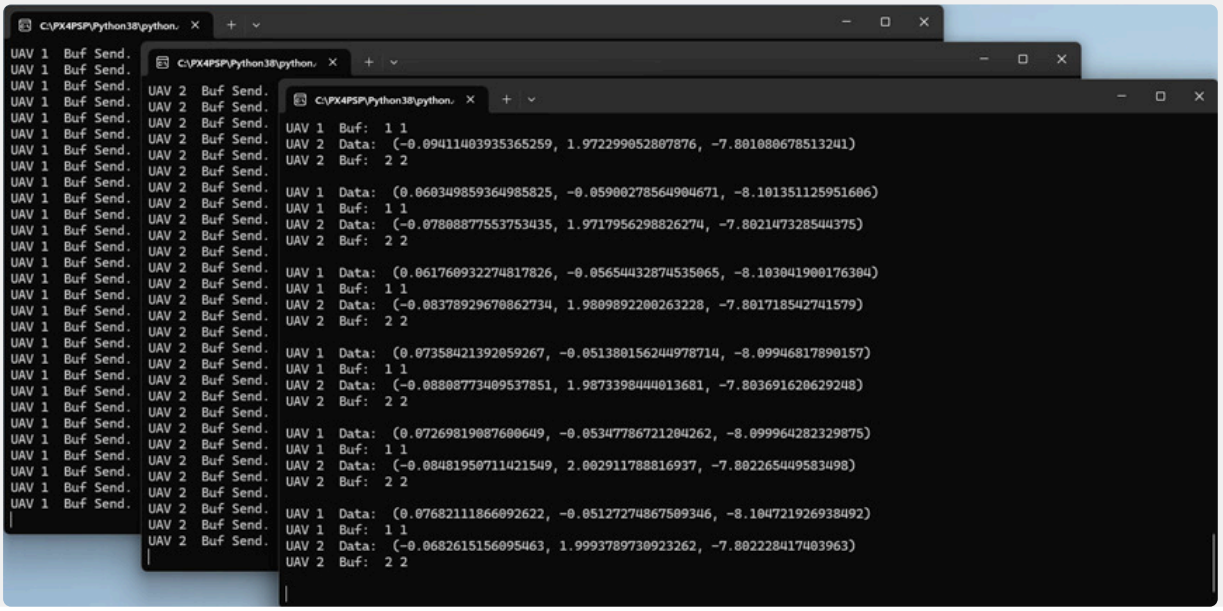
双击运行"Python38Run.bat"，在其中输入"python UAV3Ctrl.py"，启动3号飞机广播接收程序。



4.1.3 运行数据接收程序效果

观察每个窗口，每个程序都打印自己收到的数据，并输出。1-2号飞机，将结构体转为buf，并发出，在命令提示符窗口打印。3号飞机，收到包后，解析并打印。

> 注：步骤2-4可以用直接双击运行"UavPythonRunALL.bat"来替代，会自动完成上述操作。



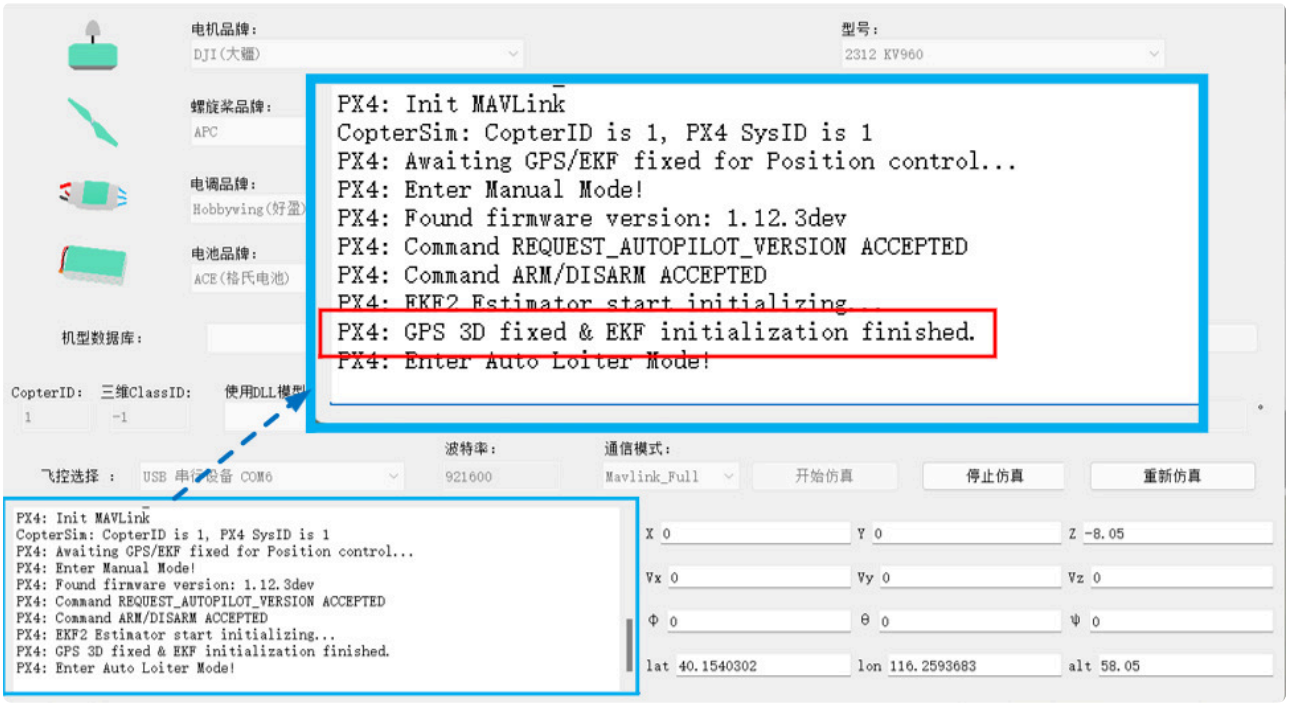
4.2 联机实验

4.2.1 准备工作

1）准备3台任意系统主机（可以是NX主机、Ubuntu虚拟机、Windows电脑）。确保两台机器是在同一个局域网内。
2）按照[RflySim按安装理解]\RflySimAPIs\1.RflySimIntro\2.AdvExps\e3.PythonConfig实验将[RflySim安装路径]\RflySimAPIs\RflySimSDK\RflySimSDK库复制到每台主机上，并确保RflySimSDK已经导入当前主机的Python环境中。

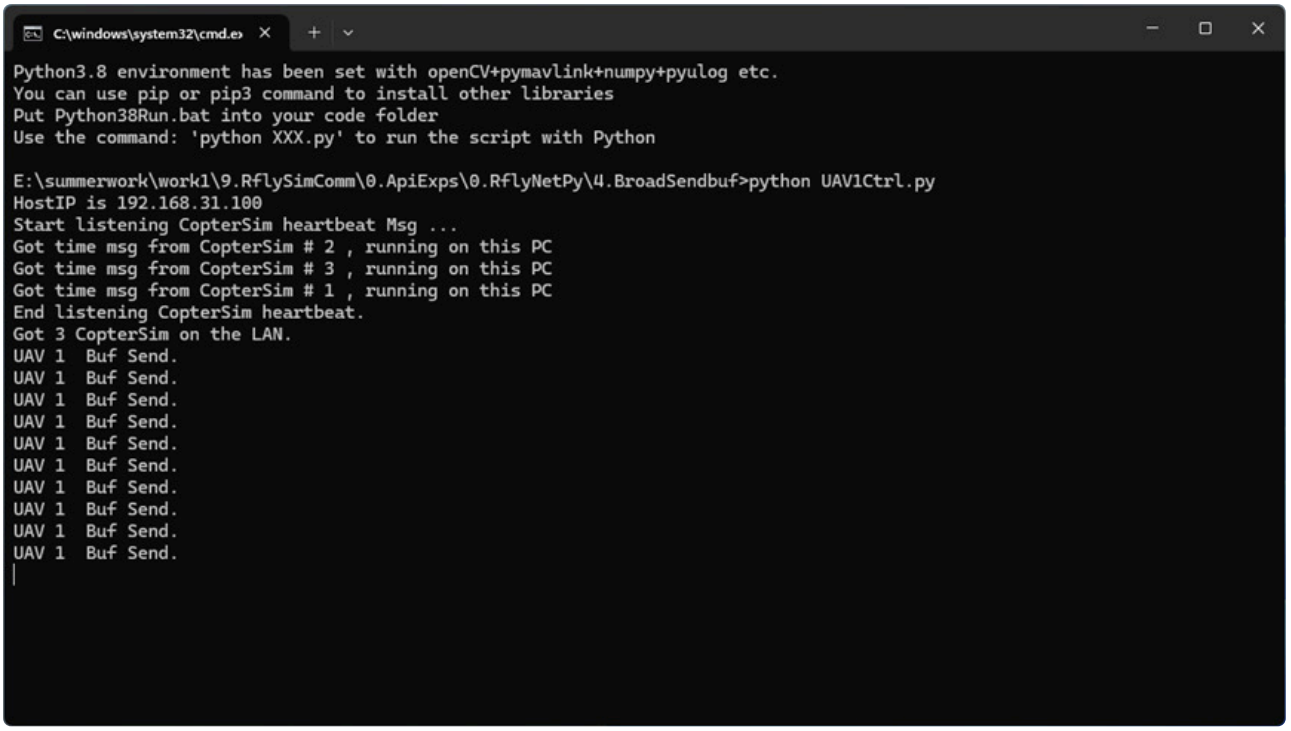
4.2.2 仿真环境初始化

双击运行SITLRun3MavlinkFull.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， 3 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：

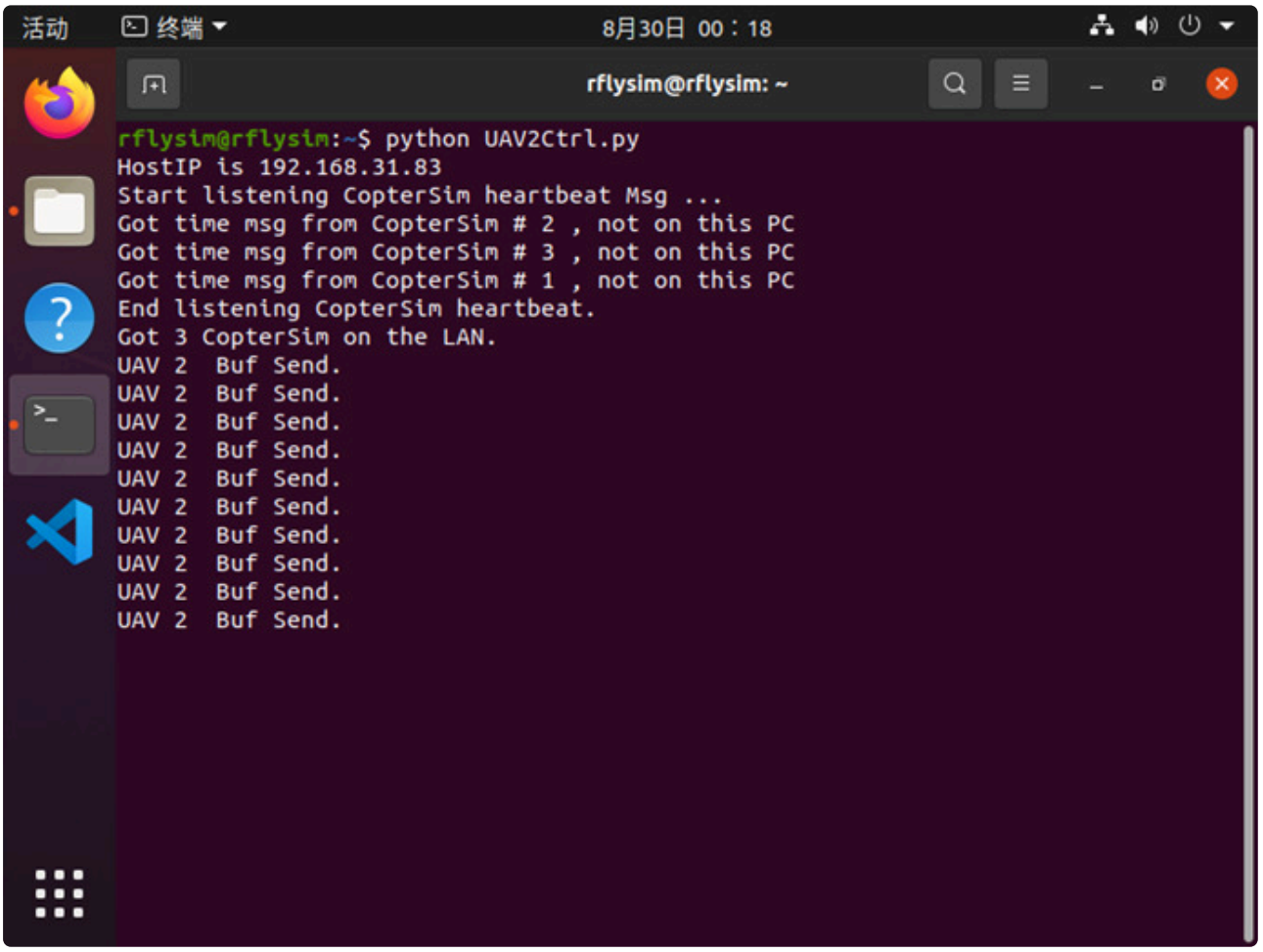


4.2.3 运行控制程序

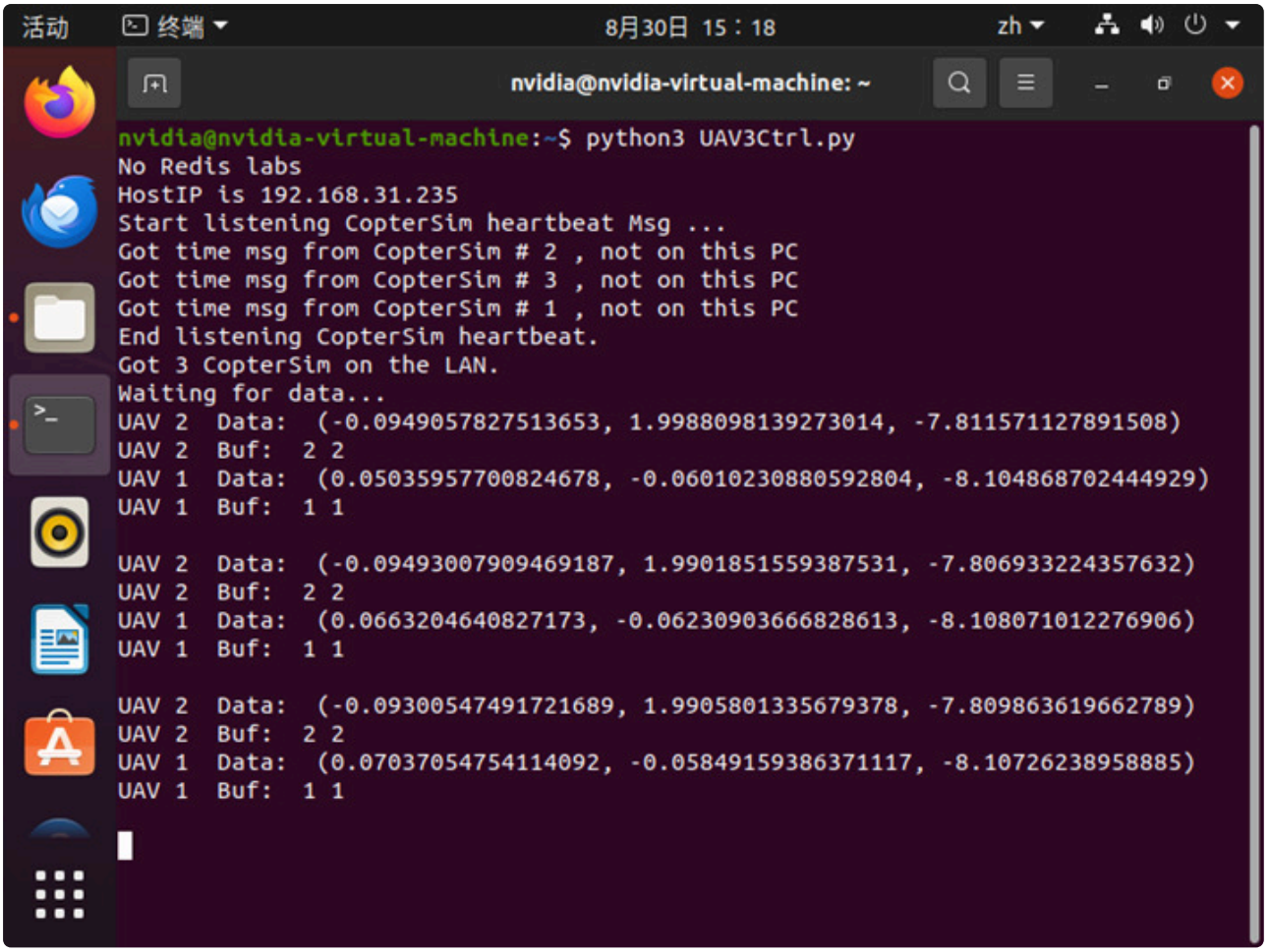
在Windows主机中，运行UAV1Ctrl.py，启动1号飞机广播发送程序。



在一台Ubuntu虚拟机中，运行UAV2Ctrl.py，启动2号飞机广播发送程序。



在另一台Ubuntu虚拟机中，运行UAV3Ctrl.py，启动3号飞机广播接收程序。



能看到和本地测试一样的效果，说明测试正确。

5. 关键知识点

本实验的整体思路是通过RflySim平台构建一个多无人机仿真环境，使用网络广播技术实现无人机之间的数据传输。整体框架分为发送端和接收端，发送端负责打包数据并通过网络广播发送，接收端负责接收并解析数据。

1. UAV1Ctrl.py 和 UAV2Ctrl.py 核心逻辑

这两个脚本的功能基本相同，都是作为数据发送端，实现广播数据的发送。

```python
import time
import math
import sys
import ReqCopterSim
import PX4MavCtrlV4 as PX4MavCtrl
import NetSimAPIV4
import struct

CopterID=1  # 对于UAV2Ctrl.py则是CopterID=2

## 分布式仿真中请求指定飞机返回数据到本电脑
req = ReqCopterSim.ReqCopterSim()

## 获取目标电脑IP并配置CopterSim回传数据到本电脑
TargetIP = req.getSimIpID(CopterID)
req.sendReSimIP(CopterID)

## 创建通信实例
mav = PX4MavCtrl.PX4MavCtrler(CopterID,TargetIP)
net = NetSimAPIV4.NetSimAPI(mav)

mav.InitMavLoop() # 连接飞控/CopterSim并准备数据
time.sleep(2)
net.enNetForward() # 启用网络转发，默认发给60000端口，即所有飞机

# 循环发送数据包
while True:
    buf = struct.pack('iiii',123456789,CopterID,CopterID,CopterID) # 打包数据
    net.netForwardBuf(buf) # 发送buf包
    print('UAV',CopterID,' Buf Send.')
    time.sleep(1)
```

2. UAV3Ctrl.py 核心逻辑

这个脚本作为数据接收端，负责接收和解析来自其他无人机的数据包。

```python
import time
import math
import sys
import ReqCopterSim
import PX4MavCtrlV4 as PX4MavCtrl
import NetSimAPIV4
import struct
import threading
import copy

# 定义UAVData类用于存储无人机数据
class UAVData:
    def __init__(self):
        self.hasUpdate=False
        self.CopterID=False
        self.Data1=False
        self.Data2=False

    def update(self,Data):
        self.hasUpdate=True
        self.CopterID=Data[1]
        self.Data1=Data[2]
        self.Data2=Data[3]  # 注意这里应该是Data[3]而不是Data[2]

# 全局字典存储接收到的无人机数据
UAVDataDict={}

# 接收和解析数据的函数
def getMavEvent():
    global UAVDataDict
    print('Waiting for data...')
    while True:
        net.bufEvent.wait() # 阻塞等待新数据包
        bufData = copy.deepcopy(net.bufData) # 拷贝数据
        bufHead = copy.deepcopy(net.bufHead) # 拷贝包头
        net.bufEvent.clear()

        if len(net.bufData) == 16:  # 16字节是发送端打包的数据格式
            Data = struct.unpack('iiii',net.bufData) # 解包数据
            checksum = Data[0]
            if checksum==123456789: # 验证校验码
                CopterID = Data[1]

                # 将解析后的数据存入字典
                if UAVDataDict.__contains__(str(CopterID)):
                    UAVDataDict[str(CopterID)].update(Data)
                else:
                    uData = UAVData()
                    uData.update(Data)
                    UAVDataDict[str(CopterID)] = copy.deepcopy(uData)


```

```python
# 初始化通信
CopterID=3
req = ReqCopterSim.ReqCopterSim()
TargetIP = req.getSimIpID(CopterID)
req.sendReSimIP(CopterID)
mav = PX4MavCtrl.PX4MavCtrler(CopterID,TargetIP)
net = NetSimAPIV4.NetSimAPI(mav)

net.StartNetRec() # 开始监听60000端口

# 启动线程接收数据
t1 = threading.Thread(target=getMavEvent, args=())
t1.start()

# 主循环显示数据
while True:
    for uav in net.UavData:
        print('UAV',uav.CopterID,' Data: ',uav.uavGlobalPos)
        if UAVDataDict.__contains__(str(uav.CopterID)):
            print('UAV',uav.CopterID,' Buf:
',UAVDataDict[str(uav.CopterID)].Data1,UAVDataDict[str(uav.CopterID)].Data2)
    if len(net.UavData)>0:
        print('')
    time.sleep(1)
```

3. 数据打包与解析机制

- 打包格式: `buf = struct.pack('iiii',checksum,CopterID,Data1,Data2)`，使用'iiii'格式表示4个整型数据，总长度为16字节
- 校验机制: 使用预设的校验码123456789验证包的正确性
- 广播机制: 通过`netForwardBuf()`方法将数据包广播到网络中
- 接收机制: 通过`StartNetRec()`方法启动网络接收，并使用线程异步处理数据

6.参考资料

1. RflySim官方文档： https://rflysim.com/doc/zh/
2. MAVLink协议详解： https://mavlink.io/en/
3. 无人机集群控制技术原理： https://rflysim.com/doc/zh/SwarmControl.pdf

7.常见问题

Q1：运行UAV3Ctrl.py时无法接收到其他无人机发送的数据包。

A1：首先检查网络连接是否正常，确保各计算机在同一局域网内。其次确认防火墙设置是否阻止了Python进程的网络访问权限。最后检查SITL仿真环境是否正常启动，各CopterSim实例是否成功连接。

Q2：数据包解析失败或解析出错的数据。

A2：检查打包和解包的数据格式是否一致，特别是struct.pack/unpack的格式字符串。确保发送端和接收端使用的校验码相同，同时确认数据长度是否符合预期（16字节）。

Q3：多线程处理数据时出现数据竞争或丢失。

A3：在UAV3Ctrl.py中使用了copy.deepcopy()来避免数据竞争，确保在net.bufEvent.clear()之前完成数据拷贝。此外，线程同步机制使用了event.wait()和event.clear()来确保数据处理的原子性。

---

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