粗粒度组网模拟实Redis 网络组网信号质量检测实验

1. 实验目的

通过无人机集群组网发送的数据都会发送到redis服务器中，然后根据组网的规则判断能否到达目的无人机并计算丢包。

2. 实验要求

- 软件要求：Windows 10及以上版本；RflySim工具链[1]；Redis。
- 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\2.AdvExps\e1.CoaGraNetSimExps\2.NetSimMini_redis_nomat

- ./SITLRun4MavlinkFull.bat：仿真环境一键启动脚本。
- ./Python38Run.bat：Python环境一键启动脚本。
- ./RedisUtils.py：Redis初始化程序。
- ./UAV1Ctrl.py：1号飞机控制程序。
- ./UAV2Ctrl.py：2号飞机控制程序。
- ./UAV3Ctrl.py：3号飞机控制程序。
- ./UAV4Ctrl.py：4号飞机控制程序。
- ./UavPythonRunALL.bat：所有飞机控制程序一键启动脚本。
- ./NetworkSimulationNoRflysim.py：信息转发服务程序。
- ./PyViaGui.py：通信质量可视化程序。

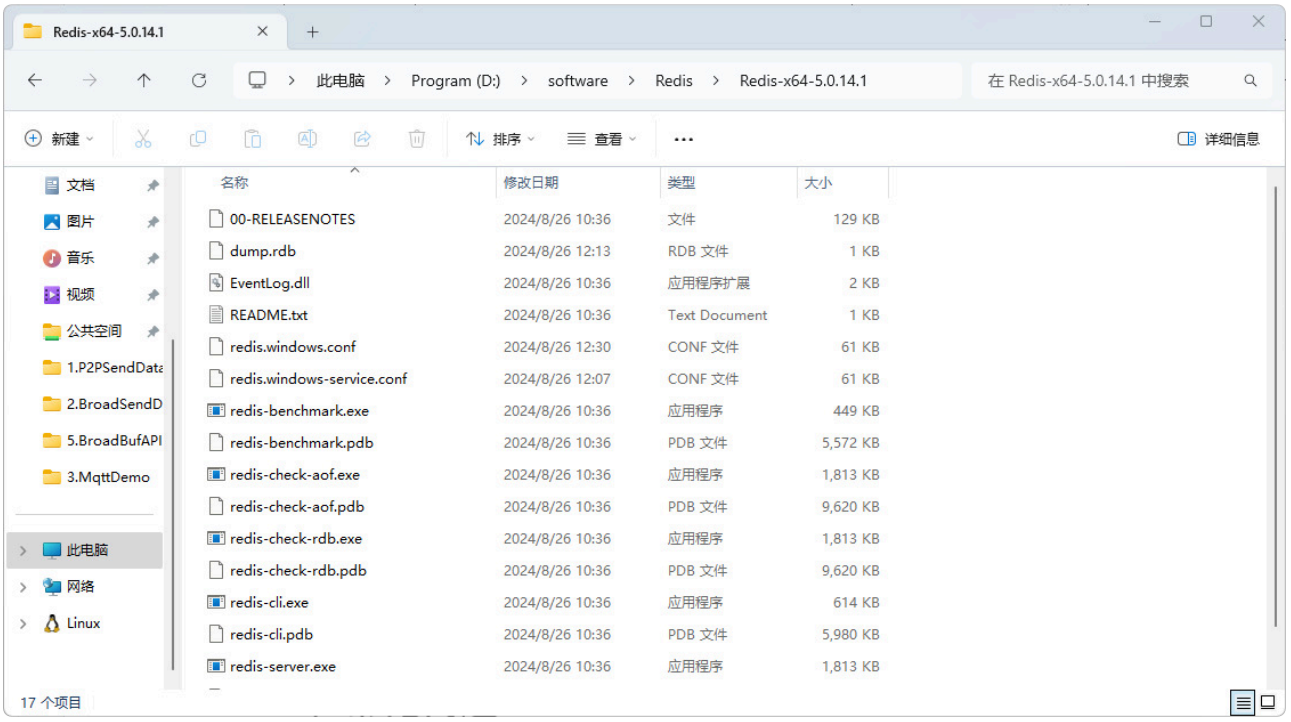
4. 实验内容或步骤

4.1 步骤1：Redis安装

如果没有安装过Redis，或者不知道Redis相关知识的，请查看文档：
*:\PX4PSP\RflySimAPIs\9.RflySimComm\0.ApiExps\5.RedisDemo\readme.pdf

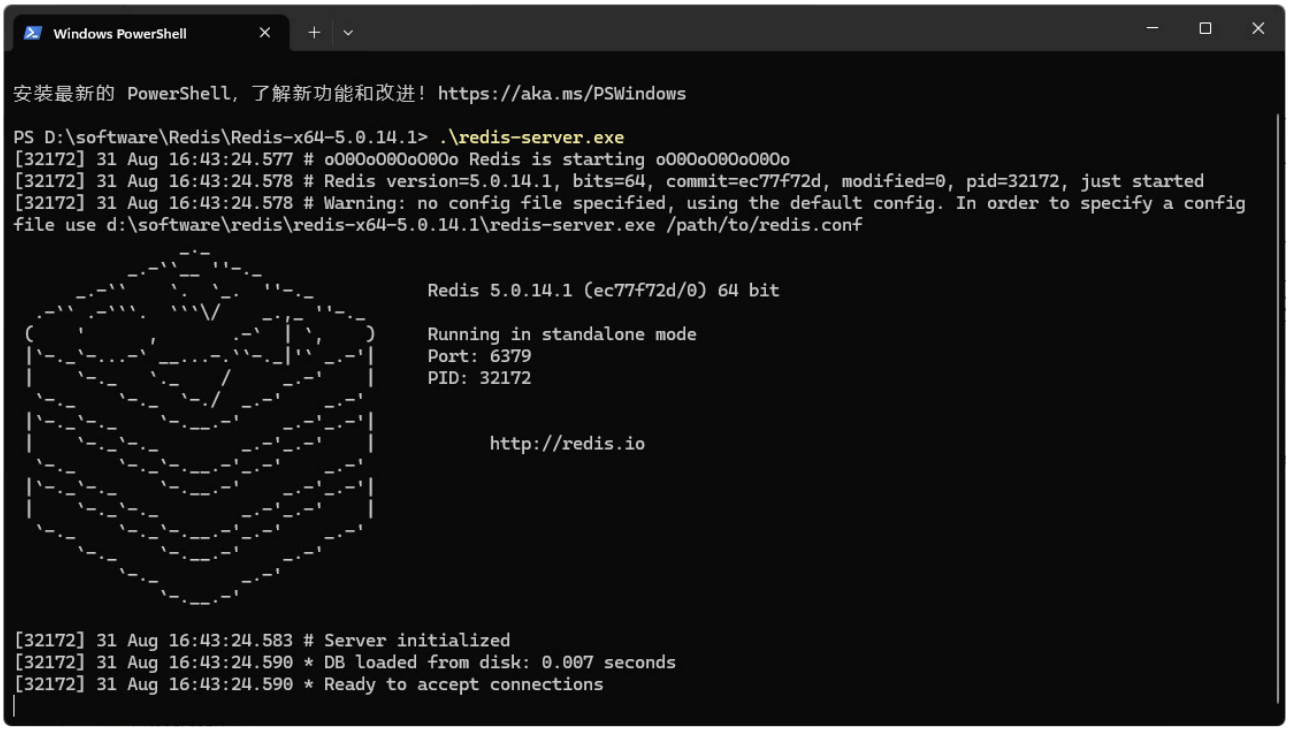
4.2 步骤2：本机实验

进入Redis的安装目录中。



在该目录中打开cmd，输入如下的命令，来开启Redis服务器。

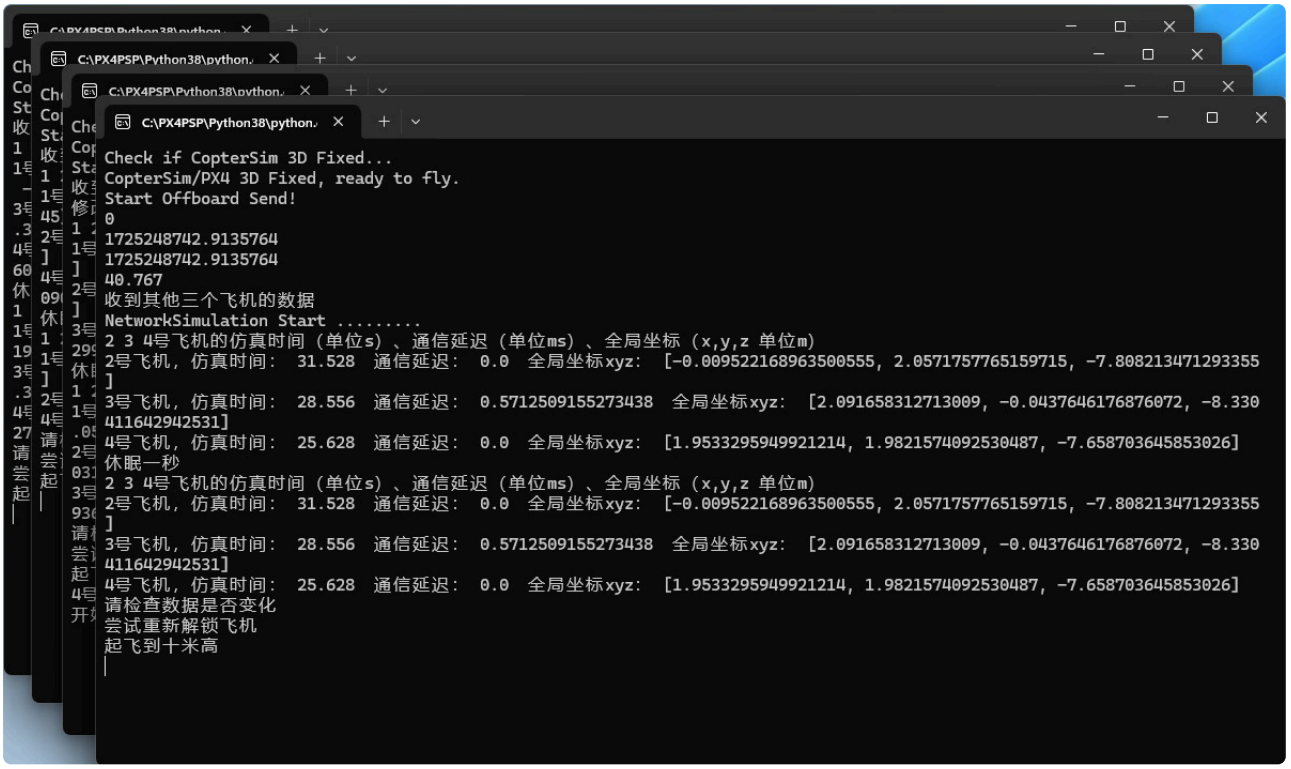
.\redis-server.exe



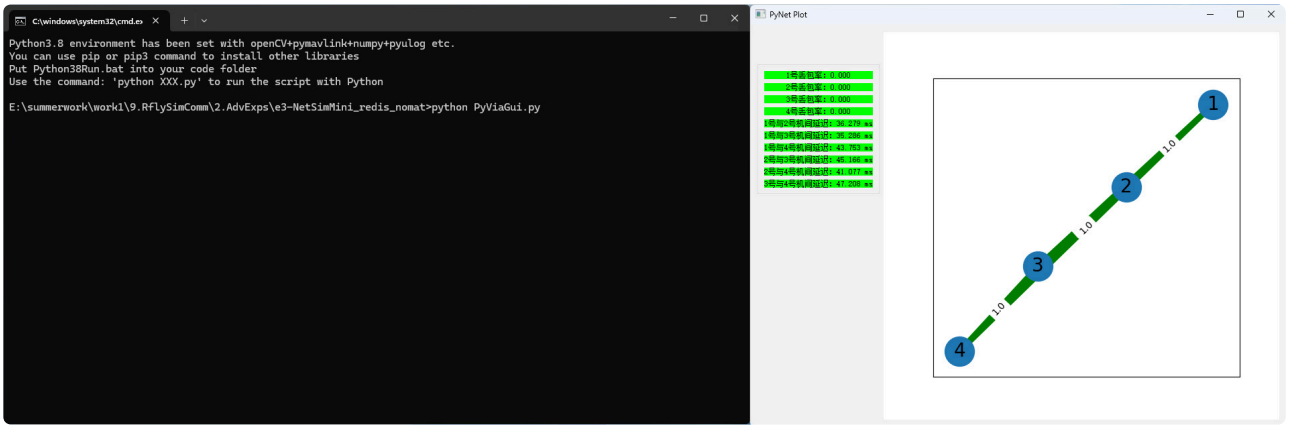
注意，默认情况下该服务是前台运行的服务，如果关闭该命令框，则Redis服务器会被关闭。

双击运行SITLRun4MavlinkFull.bat来自动创建四架飞机。等待所有CopterSim的左下角打印"PX4: GPS 3D fixed & EKF initialization finished."。

双击运行UavPythonRunALL.bat，来一键启动所有飞机的控制程序。也可以使用"Python38Run.bat"来手动运行四个控制程序。



双击运行Python38Run.bat，输入"python PyViaGui.py"来运行可视化成勋，查看实验效果。



5. 关键知识点

关键知识点1：无人机集群组网

发布无人机数据到Redis。mav.netEvent.wait()：等待MAVLink数据更新。redisConnect.pub_data("UAV1",UAV1)：将无人机的数据发布到Redis数据库中的UAV1频道。

```python
def PublicUavData():
    while True:
        mav.netEvent.wait()
        TimeUnix = time.time_ns()/1e9
        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)
```

订阅其他无人机的数据

```python
def sub_callback(channel, data):
    # Process the data received from Redis
    UavID = channel
    TimeCur = time.time_ns()/1e9
    PacketPing = redisConnect.get_data("PacketPing")
    if len(UavData) >= 3:
        UavGpsPos = mav.uavPosNED + UavData["UAV2"]["uavPosNED"] + UavData["UAV3"]
["uavPosNED"] + UavData["UAV4"]["uavPosNED"]
        redisConnect.set_data("UavGpsPos", UavGpsPos)
    if len(UavData) < 3 or PacketPing == False:
        UavData[UavID] = data
        TimeUnix = time.time_ns()/1e9
        UavData[UavID]["timeDelay"] = TimeUnix - data["timeDelay"]
    elif UavID == "UAV2":
        UAV2ping = PacketPing[0*4 + 1]
        if TimeCur - LastTime[0] > UAV2ping:
            UavData[UavID] = data
            LastTime[0] = TimeCur
            UavData[UavID]["timeDelay"] = TimeUnix - data["timeDelay"]
    elif UavID == "UAV3":
        UAV3ping = PacketPing[0*4 + 2]
        if TimeCur - LastTime[1] > UAV3ping:
            UavData[UavID] = data
            LastTime[1] = TimeCur
            UavData[UavID]["timeDelay"] = TimeUnix - data["timeDelay"]
    elif UavID == "UAV4":
        UAV4ping = PacketPing[0*4 + 3]
        if TimeCur - LastTime[2] > UAV4ping:
            UavData[UavID] = data
            LastTime[2] = TimeCur
            UavData[UavID]["timeDelay"] = TimeUnix - data["timeDelay"]
```

sub_callback(channel, data)：处理从Redis中接收到的数据，并根据无人机ID更新UavData字典。redisConnect.get_data("PacketPing")：从Redis获取包的延迟信息，用于确定是否需要更新数据。redisConnect.set_data("UavGpsPos", UavGpsPos)：将合成的GPS位置数据存储到Redis中。

发布和订阅线程启动。PubUavNet()：启动一个线程来持续发布无人机数据到Redis。SubUavNet()：启动一个线程来订阅UAV2, UAV3, UAV4的频道，并处理接收到的数据。

```
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()
```

关键知识点2：组网信号质量检测

NetworkSimulationNoRflysimg.py文件实现了一个无人机网络仿真系统，主要用于模拟和分析多个无人机节点之间的通信质量与网络性能。它通过 UDP 协议从外部仿真系统（CopterSim）接收数据包，解析并更新每个节点的实时状态信息，使用 CopterData 类管理每个无人机的状态，包括位置、速度、姿态、电机转速、加速度等信息，整个系统能够计算出每对节点之间的路由表、丢包率、路径损失、吞吐量等关键指标。并通过 Redis 将这些数据存储和共享。

sendRoutingTable 方法周期性地更新并计算网络中各个节点之间的路由表、丢包率、路径丢失率、吞吐量、延迟等网络质量指标。每次迭代中，它会首先通过调用 calculateRoutingTable 来更新路由表，确保每个节点之间的最佳路径得到计算。接着，方法调用 calculatePacketLossRate 来计算每个节点的丢包率，以反映节点之间的通信可靠性。随后，它会调用 calculatePathLose 来计算路径丢失率、吞吐量和延迟等其他网络性能指标。

在这些计算完成后，方法会通过循环遍历所有节点，并调用 Dijkstra 来计算每个节点到其他所有节点的最短路径，进一步更新网络的最优路由信息。更新完成后，方法将信号质量表（SignalQualityTable）、节点数量（NumberOfNodes）和最小距离（minDistance）网络数据存储到 Redis 数据库中。

```python
def sendRoutingTable(self):
        before_time = datetime.datetime.now()
        while True:
            # 更新路由表，计算节点之间的最佳路由
            self.calculateRoutingTable()
            # 计算每个节点的丢包率
            self.calculatePacketLossRate()
            # 计算路径丢失率、吞吐量、延迟等网络质量指标
            self.calculatePathLose()
            for i in range(self.numberOfNodes):
                self.Dijkstra(i)
            self.redisUtils.set_data("SignalQualityTable",self.SignalQualityTable)
            # self.redisUtils.set_data("PacketLossRate",self.packetLossRate)
            self.redisUtils.set_data("NumberOfNodes",self.numberOfNodes)
            self.redisUtils.set_data("minDistance",self.minDistance)
            time.sleep(0.4)
```

receiveFromCopterSimUdp 方法接收来自外部仿真系统的数据包，解析其中的无人机状态信息，并将这些信息更新到本地的仿真系统中。它通过 UDP 协议与仿真系统进行通信，接收包括无人机位置、速度、姿态、传感器数据等在内的实时信息。这些数据会被解析后，更新到仿真系统中对应的节点对象，确保仿真系统中的无人机状态与外部仿真环境保持同步。最终，这些更新后的无人机 GPS 数据会存储到 Redis 数据库中，便于其他模块或系统进行进一步的数据处理和分析。

```python
def receiveFromCopterSimUdp(self):
        """
        接收来自 CopterSim 的 UDP 数据包，解析数据包中的各种 UAV 状态信息（如位置、速度、姿态等）
        然后将这些信息更新到仿真节点中，并最终将所有 UAV 的 GPS 位置信息存储到 Redis 数据库中
        """
        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
        sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        # 绑定20008接口
        sock.bind(('0.0.0.0', 20008))
        mreq = struct.pack("=4sl", socket.inet_aton("224.0.0.11"), socket.INADDR_ANY)
        sock.setsockopt(socket.IPPROTO_IP,socket.IP_ADD_MEMBERSHIP,mreq)
        while True:
            message, addr = sock.recvfrom(1024)
            # 对数据包中的校验和进行验证
            checksum = struct.unpack("I",message[0:4])[0]
            copterID = struct.unpack("I",message[4:8])[0] - 1
            if checksum == 123456789 and self.numberOfNodes > copterID and len(message)==168:
                try:
                    # 提取出数据包中的各项信息，包括无人机的 ID、车辆类型、速度、姿态、加速度、电机转速、位置信息等。
                    # UavData = struct.unpack("4i24f7d",message[0:168])
                    self.nodes[copterID].vehicleType = struct.unpack("I",message[8:12])[0]
                    self.nodes[copterID].velocity = struct.unpack("3f",message[16:28])
                    self.nodes[copterID].angEuler = struct.unpack("3f",message[28:40])
                    self.nodes[copterID].angQuatern = struct.unpack("4f",message[40:56])
                    self.nodes[copterID].moterRPMS = struct.unpack("8f",message[56:88])
                    self.nodes[copterID].accB = struct.unpack("3f",message[88:100])
                    self.nodes[copterID].rateV = struct.unpack("3f",message[100:112])
                    self.nodes[copterID].runnedTime = struct.unpack("d",message[112:120])[0]
                    self.nodes[copterID].position = struct.unpack("3d",message[120:144])
                    self.nodes[copterID].posGPS = struct.unpack("3d",message[144:168])
                    # 将所有无人机的 GPS 位置信息汇总，存储在 Redis 数据库中的一个名为 UavGpsPos 的键中，以供后续使用。
                    all_uav_pos = []
                    for node in self.nodes:
                        for pos in node.position:
                            all_uav_pos.append(pos)
                    self.redisUtils.set_data("UavGpsPos",all_uav_pos)
                except:
```

```
                pass
```

6.参考资料

1. RflySim官方文档
2. Redis官方网站
3. Redis中文文档

7.常见问题

Q1：Redis服务器无法启动怎么办?

A1：检查是否有其他Redis进程正在运行，如果有，请先关闭它们。确认端口6379没有被其他程序占用。如果问题仍然存在，请尝试以管理员身份运行命令提示符。

Q2：程序连接Redis失败是什么原因?

A2：请确保Redis服务器已正确启动并且程序中的连接参数与Redis服务器配置一致。检查IP地址和端口号是否正确配置。

Q3：如何验证无人机集群组网是否正常工作?

A3：可以通过观察PyViaGui.py提供的图形用户界面来查看通信质量可视化结果，确认各个无人机之间的数据传输是否正常。同时也可以监控Redis数据库中各UAV频道的数据更新情况。

---

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