Redis飞机组网实验

1. 实验目的

探索多无人机系统中数据通信的实现方式，通过比较基于 Redis 的三种通信架构，分析不同方法在实时性、扩展性和系统性能上的表现。重点是验证发布-订阅模式与键值存储模式的实际应用。

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\3.Redis

- ./e1.1-PubSubComm/RedisUtils.py：Redis初始化程序
- ./e1.1-PubSubComm/UAV1Ctrl.py：1号飞机控制程序
- ./e1.1-PubSubComm/UAV2Ctrl.py：2号飞机控制程序
- ./e1.1-PubSubComm/UAV3Ctrl.py：3号飞机控制程序
- ./e1.1-PubSubComm/UAV4Ctrl.py：4号飞机控制程序
- ./e1.1-PubSubComm/SITLRun4MavlinkFull.bat：仿真环境一键启动脚本
- ./e1.1-PubSubComm/Python38Run.bat：Python环境一键启动脚本
- ./e1.1-PubSubComm/UavPythonRunALL.bat：所有飞机控制程序一键启动脚本
- ./e1.2-KeyValueComm/RedisUtils.py：Redis初始化程序
- ./e1.2-KeyValueComm/UAV1Ctrl.py：1号飞机控制程序
- ./e1.2-KeyValueComm/UAV2Ctrl.py：2号飞机控制程序
- ./e1.2-KeyValueComm/UAV3Ctrl.py：3号飞机控制程序
- ./e1.2-KeyValueComm/UAV4Ctrl.py：4号飞机控制程序

- ./e1.2-KeyValueComm/SITLRun4MavlinkFull.bat：仿真环境一键启动脚本
- ./e1.2-KeyValueComm/Python38Run.bat：Python环境一键启动脚本
- ./e1.2-KeyValueComm/UavPythonRunALL.bat：所有飞机控制程序一键启动脚本
- ./e1.3-MultiChannelComm/RedisUtils.py：Redis初始化程序
- ./e1.3-MultiChannelComm/UAV1Ctrl.py：1号飞机控制程序
- ./e1.3-MultiChannelComm/UAV2Ctrl.py：2号飞机控制程序
- ./e1.3-MultiChannelComm/UAV3Ctrl.py：3号飞机控制程序
- ./e1.3-MultiChannelComm/UAV4Ctrl.py：4号飞机控制程序
- ./e1.3-MultiChannelComm/SITLRun4MavlinkFull.bat：仿真环境一键启动脚本
- ./e1.3-MultiChannelComm/Python38Run.bat：Python环境一键启动脚本
- ./e1.3-MultiChannelComm/UavPythonRunALL.bat：所有飞机控制程序一键启动脚本

4. 实验内容或步骤

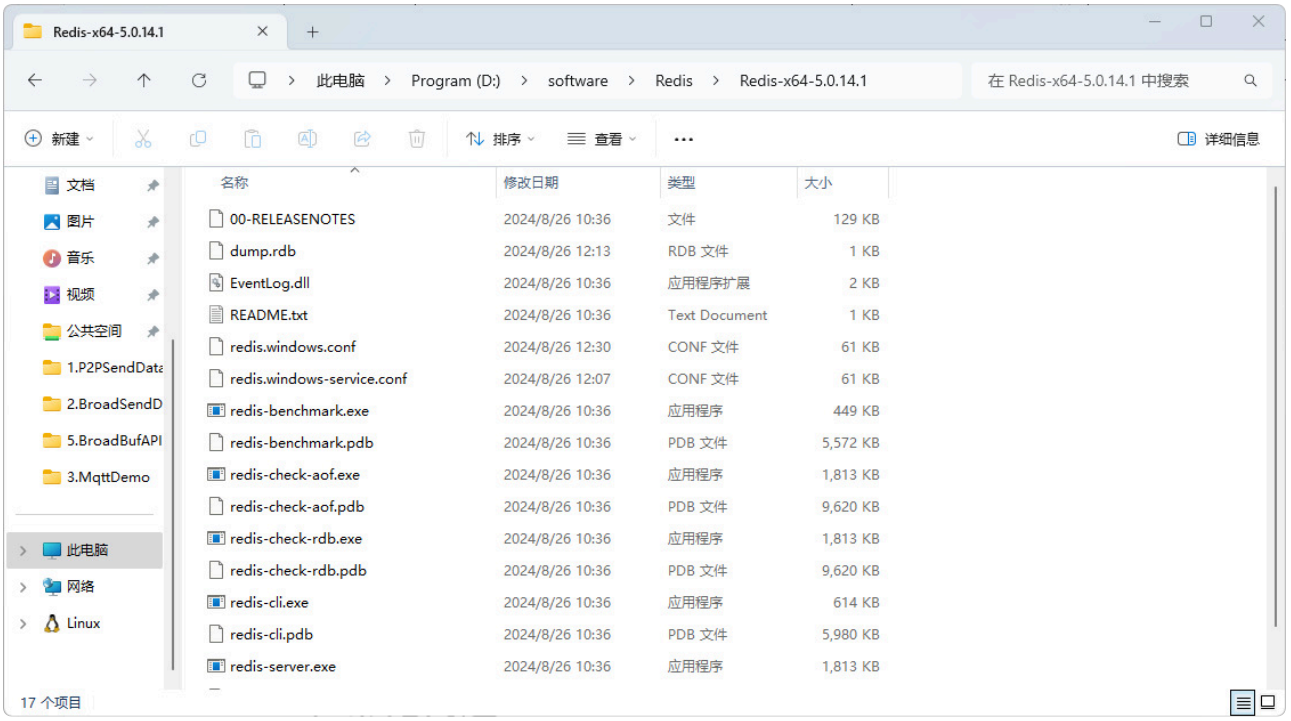
本实验分为两个小实验。

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

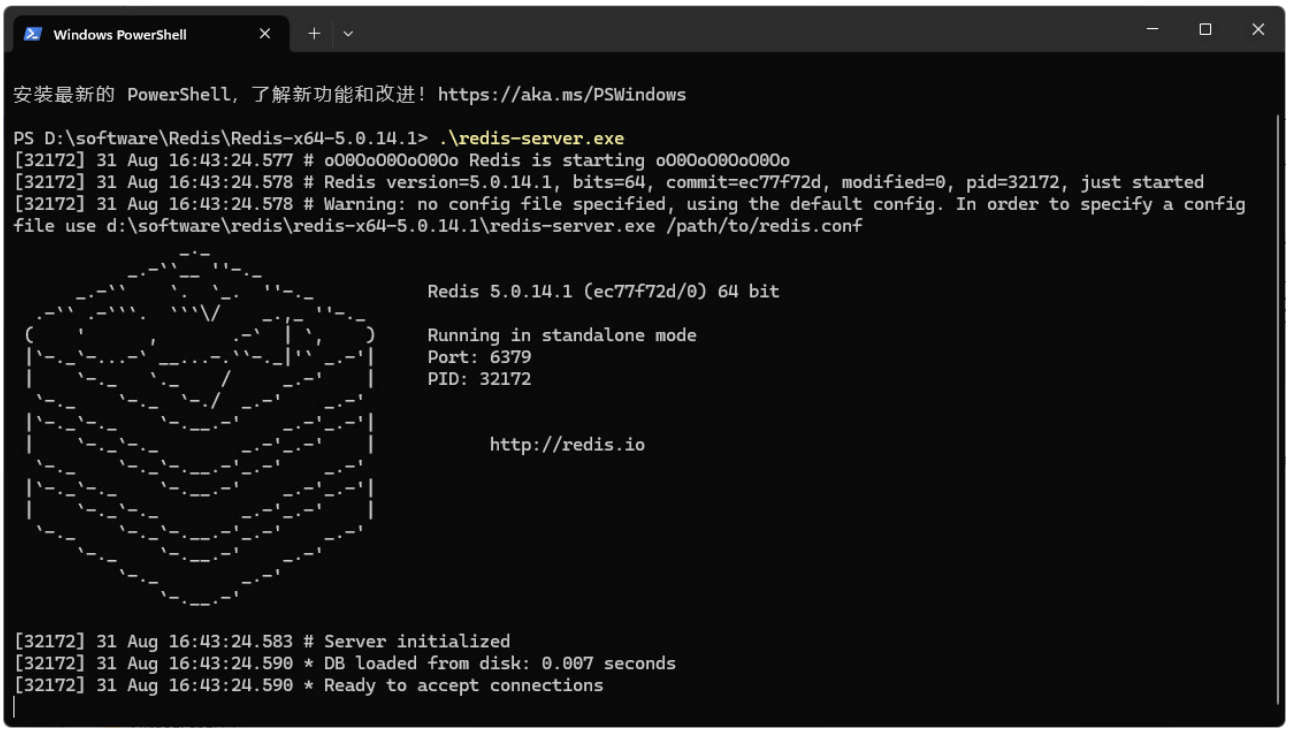
4.2 步骤2：本机实验

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。

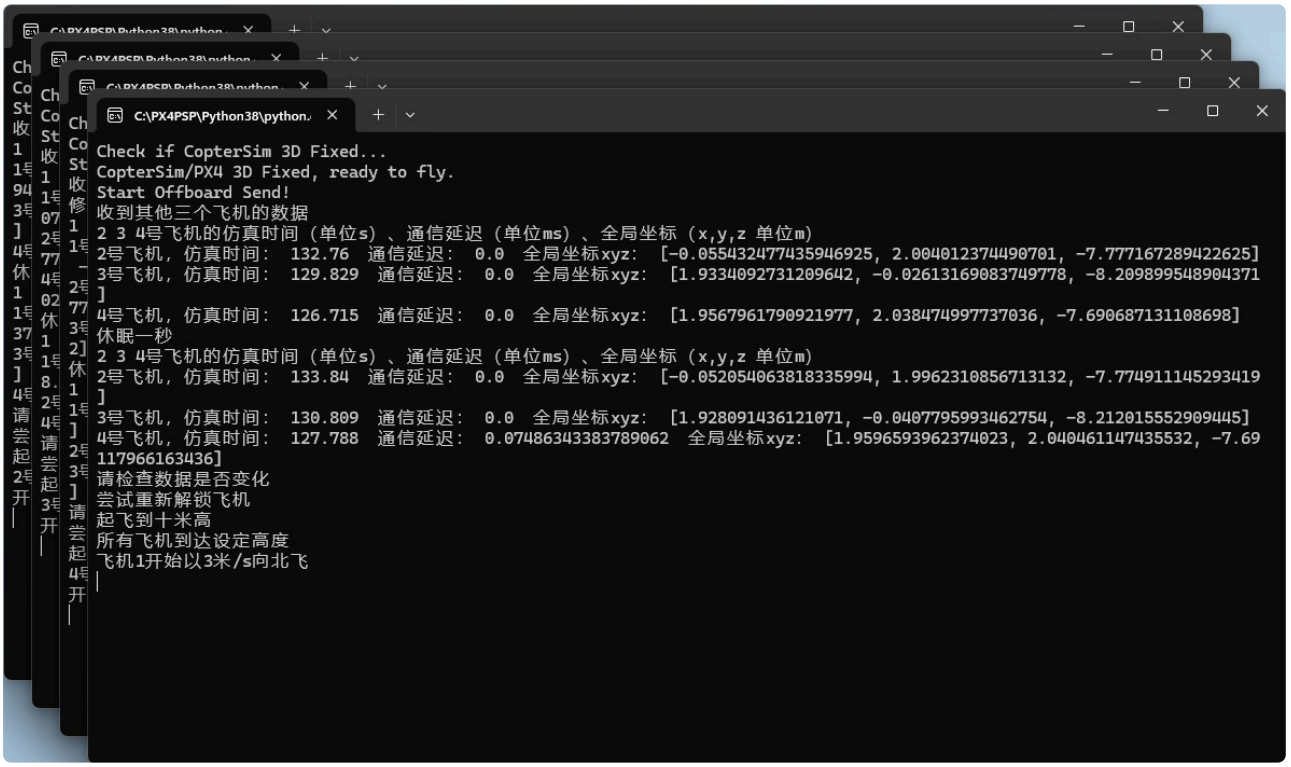
.\redis-server.exe



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。



打开RflySim3D，可以看到飞机正在编队飞行。

5. 关键知识点

通过将不同的飞机绑定到各自的数据发送端口，并订阅相应的端口数据，将通信数据发送到Redis，从而实现飞机间的数据交互。三个实验在发布、订阅和数据存储方式上略有差异，以下将分别进行对比说明。

关键知识点1：Redis多无人机发布-订阅通信实验(e1.1-PubSubComm)

sub_callback 实现了接收到订阅频道数据后的处理逻辑。该函数通过 Redis 的回调机制处理接收到的数据，并根据 CopterID 判断数据的来源，将其存储在 UavData 列表中。存储逻辑按照固定索引与无人机 ID 绑定，以1号无人机为例， 2 号无人机的数据存储在 UavData[0]，3 号存储在 UavData[1]，以此类推。这种存储方式简单直观，但缺乏灵活性，当无人机数量或 ID 发生变化时，需要手动调整代码结构。

PublicUavData 通过遍历目标频道列表（如 UAV2、UAV3、UAV4），将当前无人机状态数据发布到这些频道中，从而实现无人机间的数据共享。每次发布的数据包含时间戳、无人机 ID、位置、速度、姿态等关键信息。其特点是基于 Redis 的发布-订阅机制，具有较高的实时性，但发布的目标频道是静态的（由 enNetForwardList 定义），如果需要新增或移除频道，必须手动修改代码，扩展性较为有限。

关键知识点2：Redis多无人机键值存储通信实验(e1.2-KeyValueComm)

GetCallback 函数负责从 Redis 中获取其他无人机的数据，依赖键值存储的方式来实现订阅功能。通过轮询指定的 UavList，函数持续调用 redisConnect.get_data(uav) 从 Redis 获取对应无人机的数据，并存储到 UavData 字典中，同时计算数据的通信延迟并更新字典中的 timeDelay 字段。与程序一的回调订阅机制不同，这里没有直接使用 Redis 的发布-订阅模型，而是通过键值存储来模拟订阅效果。这种方式不依赖 Redis 的实时事件触发，但需要定期主动轮询 Redis，因此在实时性上略逊一筹，但在架构上更灵活，可以动态调整要获取的键值。

SetUavData 函数实现了当前无人机数据的发布操作，采用 Redis 的键值存储方式，通过 redisConnect.set_data("UAV1", UAV1) 将当前无人机的状态数据存储到 Redis 中的 UAV1 键下。与程序一不同的是，程序二并没有直接向其他无人机的订阅频道发送消息，而是将数据存储在 Redis 键值对中。其他无人机通过轮询获取这些键值，从而实现数据共享。这种发布方式不依赖特定的频道或实时事件，适合需要动态管理和访问多个数据源的场景，同时也简化了订阅端逻辑。

关键知识点3：Redis多无人机多频道订阅通信实验(e1.3-MultiChannelComm)

SubUavNet 函数通过 Redis 的多频道订阅功能同时监听多个无人机的频道。当订阅的频道接收到消息时，回调函数 sub_callback 被触发。回调函数会根据频道名称识别数据来源，将接收到的数据动态存储到字典 UavData 中，并计算通信延迟。字典的键为频道名称，值为对应的无人机状态数据。多频道的订阅设计能够灵活扩展新的频道，动态管理多个无人机的数据，是一种高效且扩展性良好的实现方式。

实验三的发布功能延续了实验一的实时性特点，但在订阅功能上进行了增强。通过多频道订阅功能，可以同时监听多个无人机的频道，并将数据动态存储在字典中。与实验一不同，实验三的订阅方式更灵活，支持动态扩展新的频道，而不需要手动调整代码逻辑。
Redis多无人机多频道订阅通信实验(e1.3-MultiChannelComm)：

SubUavNet 函数通过 Redis 的多频道订阅功能同时监听多个无人机的频道。当订阅的频道接收到消息时，回调函数 sub_callback 被触发。回调函数会根据频道名称识别数据来源，将接收到的数据动态存储到字典 UavData 中，并计算通信延迟。字典的键为频道名称，值为对应的无人机状态数据。多频道的订阅设计能够灵活扩展新的频道，动态管理多个无人机的数据，是一种高效且扩展性良好的实现方式。

```
def sub_callback(channel,data):
```

"""

订阅回调函数：处理从 Redis 多频道订阅接收到的无人机数据。

动态存储数据：

- 使用 UavData 字典动态存储接收到的无人机数据。

- 字典的键是频道名称，值是数据内容。

3. 特点：

- 多频道支持：支持同时从多个频道（如 UAV2、UAV3、UAV4）接收数据。

- 扩展性强：无需提前定义具体频道，添加新频道时只需更新订阅列表即可。

- 实时性：基于 Redis 的发布-订阅机制，消息实时性较高。

"""

UavID = channel

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 SubUavNet():

"""

数据订阅函数：订阅多个 Redis 频道，接收其他无人机发布的数据。

多频道订阅：

- 通过 `redisConnect.sub_data_multiple_channels` 同时订阅多个频道（如 `UAV2`、`UAV3`、`UAV4`）。

- 适用于多无人机系统中需要接收多个数据源的场景。

异步订阅：

- 使用多线程执行订阅任务，避免阻塞主线程，提升程序响应速度。

"""

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()
```

实验三的发布功能延续了实验一的实时性特点，但在订阅功能上进行了增强。通过多频道订阅功能，可以同时监听多个无人机的频道，并将数据动态存储在字典中。与实验一不同，实验三的订阅方式更灵活，支持动态扩展新的频道，而不需要手动调整代码逻辑。

```
def PublicUavData():

"""

数据发布函数：通过 Redis 的发布-订阅机制向特定频道发布当前无人机状态数据。

实时发布：

- 每当 `mav.netEvent` 检测到新消息时，获取当前无人机的状态数据（如时间戳、位置、速度等）。

- 数据被发布到 Redis 的 `UAV1` 频道，供其他订阅者接收。

"""

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)
```

6.参考资料 {#6参考资料 }

1. RflySim官方文档
2. Redis官方文档
3. ../0.ApiExps/5.RedisDemo/readme.pdf

7.常见问题

Q1：Redis服务无法启动怎么办？

A1：检查Redis安装是否完整，确认端口是否被占用，可以尝试更换端口号重新启动Redis服务。

Q2：多无人机通信延迟过高如何解决?

A2：针对发布-订阅模式，检查网络状况；对于键值存储模式，可适当调整轮询频率；确保Redis服务器性能满足需求。

Q3：如何增加更多无人机到通信网络中?

A3：发布-订阅模式需要修改代码结构，重新定义索引与无人机ID的绑定关系；键值存储模式具有较好的扩展性，可通过动态调整键值实现；多频道订阅模式支持动态扩展新频道，只需更新订阅列表即可。

---

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