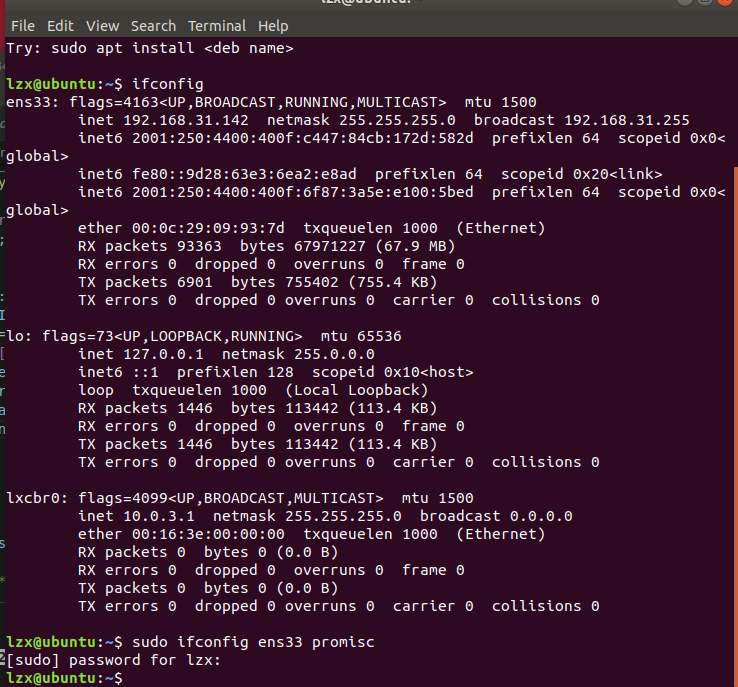
**网卡设置：**

由于demo是基于fdnetdevice，因此需要绑定物理网卡，假设你的Ubuntu系统物理网卡名为ens33，需要设置为promiscuous模式，直接在终端输入sudo ifcongfig ens33 promisc。



**DDS设置：**

由于DDS底层已经被修改过，因此需要安装我们已经修改的DDS，目前gitee不支持上传过大的文件。我们指修改了\src\fastrtps\src\cpp\rtps\transportUDPTransportInterface.cc文件，将DDS源码-修改底层发包机制中的文件替换DDS源代码中的文件，然后DDS安装就按照压缩包中的redme文件的说明即可，安装完成后跑一个DDSUDP的通信例程，即可开始实验

**DDSUDP例程：切记不要在同一台主机上运行订阅者和发布者，这样不会用UDP而是用共享内存，NS3无法获取数据包，无法达到实验效果**



将dataPro1.idl放置到任意目录，假设你放到~/ddaudp目录，

执行cd ~/ddsudp进入该目录下，

运行fastddsgen -example CMake dataPro1.idl，生成DDS的C++文件，

执行mkdir build

cd ./build

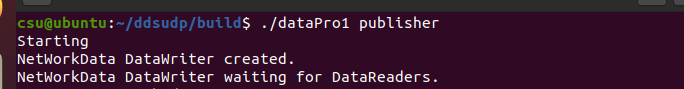
执行 sudo cmake ..

执行 sudo make

如果上面的命令都没有问题，执行 ./dataPro1 publisher 添加一个发布者

执行 ./dataPro1 subscriber 添加一个订阅者

如果都没有问题，那么效果应该如下：



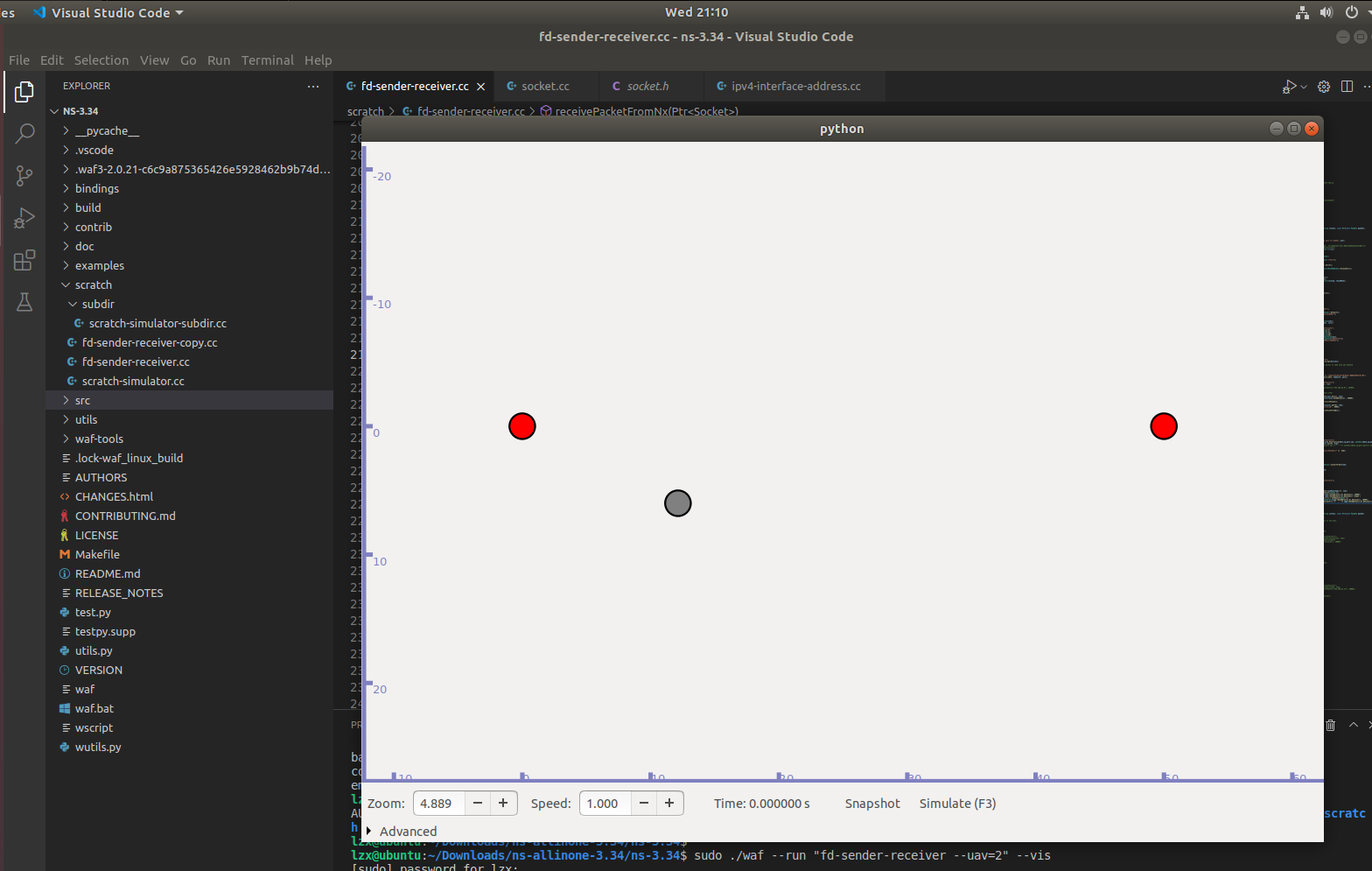
****

由于我们修改了DDS底层，因此datawriter和datareader不能匹配，他们的包都被转发到NS3所在的主机（其实是一个组播IP）

**NS3运行：**

C++源代码中的网卡名称需要修改为当前运行NS3主机的网卡名，相应的IP和路由也需要修改为当前主机网络的具体信息。

将fd-sender-receiver.cc放到NS3的scratch目录下，在终端输入 sudo ./waf “fd-sender-receiver –uav=1” –vis；此处—uav可以设置模拟的无人机数量，会根据输入的数量生成相应的节点。



如果一切都没有问题，那么DDS端应该可以接收到双方发送的数据：

