



江 蘇 大 學

JIANGSU UNIVERSITY

“无线传感网与识别技术”实验报告

学院名称: 计算机科学与通信工程学院

专业班级: 物联网工程 18 级

学生姓名: 张承楷

学生学号: 3180611023

指导教师: 熊书明

2020 年 7 月 14 日

目录

一、建立无线网络拓扑结构	1
1.1 功能介绍	1
1.2 流程图	1
1.3 代码分析	1
1.4 实验结果	8
1.5 实验收获	11

一、建立无线网络拓扑结构

1.1 功能介绍

在这个实例中，包含了一对 P2P 节点，包括一个以太网信道以及 WiFi 信道。P2P 节点间相互通信，在有线信道上，3 个结点通过 CSMA 协调，交流。无线信道上，AP 通过 WiFi 互相交流。

1.2 流程图

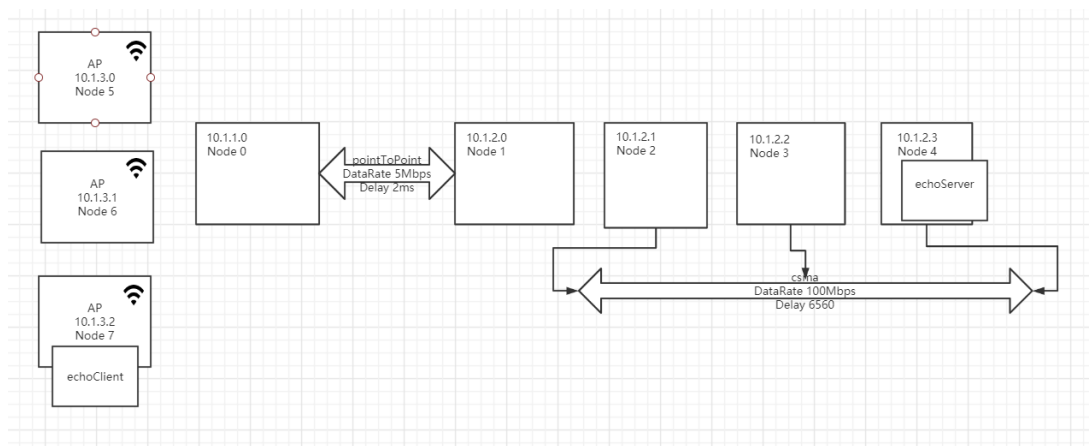


图 1.1 流程图

1.3 代码分析

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */  
/*  
 * This program is free software; you can redistribute it and/or modify  
 * it under the terms of the GNU General Public License version 2 as  
 * published by the Free Software Foundation;  
 *  
 * This program is distributed in the hope that it will be useful,  
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
```

```
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*/
```

```
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mobility-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"

// Default Network Topology
//
// Wifi 10.1.3.0
//      AP
// *   *   *   *
// |   |   |   | 10.1.1.0
// n5  n6  n7  n0 ----- n1  n2  n3  n4
//      point-to-point |   |   |
//                      =====
//                      LAN 10.1.2.0
```

```
using namespace ns3;
```

//声明了一个叫 SecondScriptExample 的日志构件,可以实现打开或者关闭控制台日志的输出

```
NS_LOG_COMPONENT_DEFINE ("ThirdScriptExample");
```

```
int
main (int argc, char *argv[])
{
    //决定是否开启两个 UdpApplication 的 Logging 组件
    bool verbose = true;
    uint32_t nCsma = 3;
    uint32_t nWifi = 3;
    bool tracing = false;
    //打印信息
    CommandLine cmd;

    cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
    cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);
    cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
    cmd.AddValue ("tracing", "Enable pcap tracing", tracing);

    cmd.Parse (argc,argv);

    // The underlying restriction of 18 is due to the grid position
    // allocator's configuration; the grid layout will exceed the
    // bounding box if more than 18 nodes are provided.
    if (nWifi > 18)
    {
        std::cout << "nWifi should be 18 or less; otherwise grid layout exceeds the bound
ing box" << std::endl;
        return 1;
    }

    if (verbose)
    {
        LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
        LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
    }
}
```

//创建使用 P2P 链路链接的 2 个节点

NodeContainer p2pNodes;

p2pNodes.Create (2);

//设置传送速率和信道延迟,传输速率 5Mbps,延迟 2ms

PointToPointHelper pointToPoint;

pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));

pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

//安装 P2P 网卡设备到 P2P 网络节点

NetDeviceContainer p2pDevices;

p2pDevices = pointToPoint.Install (p2pNodes);

//创建 NodeContainer 类对象，用于总线(CSMA)网络

NodeContainer csmaNodes;

//将第二个 P2P 节点添加到 CSMA 的 NodeContainer

csmaNodes.Add (p2pNodes.Get (1));

//创建 Bus network 上另外 3 个 node

csmaNodes.Create (nCsma);

//创建和设置 CSMA 设备及信道，通信速率是 100M，延迟 6560s

CsmaHelper csma;

csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));

csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

//安装网卡设备到 CSMA 信道的网络节点

NetDeviceContainer csmaDevices;

csmaDevices = csma.Install (csmaNodes);

//创建 NodeContainer 类对象，用于 WiFi 网络

NodeContainer wifiStaNodes;

wifiStaNodes.Create (nWifi);

```
//设置 WiFi 网络的第一个节点为 AP
NodeContainer wifiApNode = p2pNodes.Get (0);

//初始化物理信道,在物理部分设置虚拟信道部分
YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
//YansWifiPhyHelper 共享相同的底层信道,也就是说,它们共享相同的无线介质,
可以相互通信
YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
phy.SetChannel (channel.Create ());

//SetRemoteStationManager 的方法告诉助手使用何值速率控制算法
WifiHelper wifi;
wifi.SetRemoteStationManager ("ns3::AarfWifiManager");

//配置 MAC 类型和基础设施网络的 SSID。先创建 IEEE802.11 的 SSID 对
象,
//用来设置 MAC 层的“SSID”属性值。助手创建的特定种类 MAC 层被
“ns3::StaWifiMac”类型属性所指定。
WifiMacHelper mac;
Ssid ssid = Ssid ("ns-3-ssid");
mac.SetType ("ns3::StaWifiMac",
             "Ssid", SsidValue (ssid),
             "ActiveProbing", BooleanValue (false));

//安装网卡设备到 WiFi 信道的网络节点, 并配置参数, 在 MAC 层和 PHY 层
可以调用方法来安装这些站的无线设备
NetDeviceContainer staDevices;
staDevices = wifi.Install (phy, mac, wifiStaNodes);

//配置 AP 节点
mac.SetType ("ns3::ApWifiMac",
             "Ssid", SsidValue (ssid));
```

//创建单一 AP 共享相同的 PHY 层属性

```
NetDeviceContainer apDevices;  
apDevices = wifi.Install (phy, mac, wifiApNode);
```

//加入移动模型。希望 STA 节点能够移动，而使 AP 节点固定住

```
MobilityHelper mobility;
```

```
mobility.SetPositionAllocator ("ns3::GridPositionAllocator",  
                                "MinX", DoubleValue (0.0),  
                                "MinY", DoubleValue (0.0),  
                                "DeltaX", DoubleValue (5.0),  
                                "DeltaY", DoubleValue (10.0),  
                                "GridWidth", UIntegerValue (3),  
                                "LayoutType", StringValue ("RowFirst"));
```

//RandomWalk2dMobilityModel,节点以一个随机的速度在一个随机方向上移动

```
mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",  
                            "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));  
mobility.Install (wifiStaNodes);
```

```
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");  
mobility.Install (wifiApNode);
```

//安装协议栈

```
InternetStackHelper stack;  
stack.Install (csmaNodes);  
stack.Install (wifiApNode);  
stack.Install (wifiStaNodes);
```

//分配 IP 地址

//10.1.1.0 创建 2 个点到点设备需要的 2 个地址

//10.1.2.0 分配地址给 CSMA 网络

//10.1.3.0 分配地址给 STA 设备和无线网络的 AP

```
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign (p2pDevices);

address.SetBase ("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces;
csmaInterfaces = address.Assign (csmaDevices);

address.SetBase ("10.1.3.0", "255.255.255.0");
address.Assign (staDevices);
address.Assign (apDevices);

//最右端的节点放置 echo 服务端程序。
UdpEchoServerHelper echoServer (9);

ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));

//将回显客户端放在最后创建的 STA 节点上，指向 CSMA 网络的服务器
UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);
echoClient.SetAttribute ("MaxPackets", UIntegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UIntegerValue (1024));

ApplicationContainer clientApps =
    echoClient.Install (wifiStaNodes.Get (nWifi - 1));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));

//启用路由
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
```

```

//设置终止时间
Simulator::Stop (Seconds (10.0));

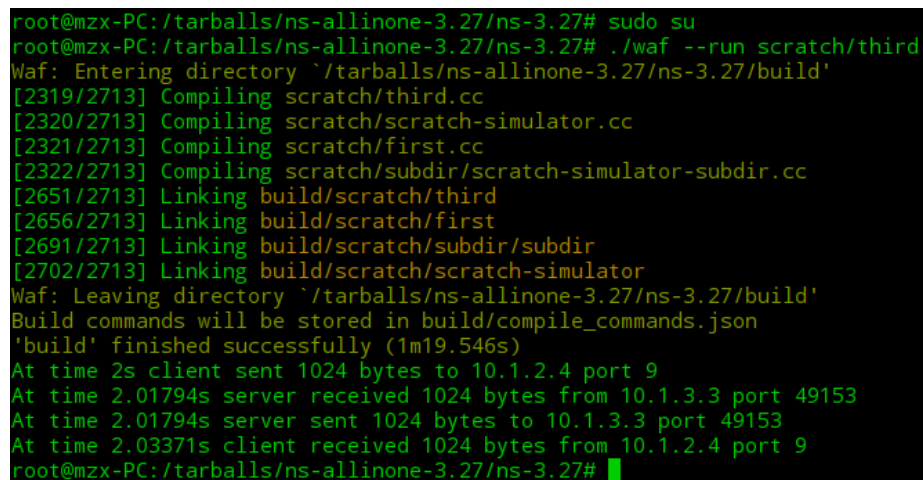
//将 pcap 数据打印出来
if (tracing == true)
{
    pointToPoint.EnablePcapAll ("third");
    phy.EnablePcap ("third", apDevices.Get (0));
    csma.EnablePcap ("third", csmaDevices.Get (0), true);
}

//运行，结束
Simulator::Run ();
Simulator::Destroy ();

return 0;
}

```

1.4 实验结果



```

root@mzx-PC:/tarballs/ns-allinone-3.27/ns-3.27# sudo su
root@mzx-PC:/tarballs/ns-allinone-3.27/ns-3.27# ./waf --run scratch/third
Waf: Entering directory `/tarballs/ns-allinone-3.27/ns-3.27/build'
[2319/2713] Compiling scratch/third.cc
[2320/2713] Compiling scratch/scratch-simulator.cc
[2321/2713] Compiling scratch/first.cc
[2322/2713] Compiling scratch/subdir/scratch-simulator-subdir.cc
[2651/2713] Linking build/scratch/third
[2656/2713] Linking build/scratch/first
[2691/2713] Linking build/scratch/subdir/subdir
[2702/2713] Linking build/scratch/scratch-simulator
Waf: Leaving directory `/tarballs/ns-allinone-3.27/ns-3.27/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1m19.546s)
At time 2s client sent 1024 bytes to 10.1.2.4 port 9
At time 2.01794s server received 1024 bytes from 10.1.3.3 port 49153
At time 2.01794s server sent 1024 bytes to 10.1.3.3 port 49153
At time 2.03371s client received 1024 bytes from 10.1.2.4 port 9
root@mzx-PC:/tarballs/ns-allinone-3.27/ns-3.27#

```

图 1.2 运行程序

在这个实例中，可以看到与 first.cc 和 second.cc 类似的信息，客户端与服务器相互通信，通过某个端口相互通信。

在此开启 UDPClient 的日志打印，这里将信息保存在一个文件中，可以看到 UDPClient 服务器更多输出的信息。

现在查看 third.cc 打印的 pcap 信息：

Third0-0.pcap

No.	Time	Source	Destination	Protoc	Lengt	Info
1	0.0000...	10.1.3.3	10.1.2.4	UDP	1054	49153 → 9 Len=1024
2	0.0186...	10.1.2.4	10.1.3.3	UDP	1054	9 → 49153 Len=1024

> Frame 1: 1054 bytes on wire (8432 bits), 1054 bytes captured (8432 bits)

> Point-to-Point Protocol

> Internet Protocol Version 4, Src: 10.1.3.3, Dst: 10.1.2.4

> User Datagram Protocol, Src Port: 49153, Dst Port: 9

> Data (1024 bytes)

图 1.3 Third0-0.pcap

可以看到，客户端向服务器发送消息，长度是 1054 字节。

Third0-1.pcap

No.	Time	Source	Destination	Protoc	Lengt	Info
1	0.0000...	00:00:00_00:0...	Broadcast	802.	61	Beacon frame, SN=0, FN=0, Flags=....., BI=100, SSID=ns-3-ssid
2	0.0002...	00:00:00_00:0...	00:00:00_00:0...	802.	53	Association Request, SN=0, FN=0, Flags=....., SSID=ns-3-ssid
3	0.0002...	00:00:00_00:0...	00:00:00_00:0...	802.	14	Acknowledgement, Flags=0.....
4	0.0004...	00:00:00_00:0...	00:00:00_00:0...	802.	53	Association Request, SN=0, FN=0, Flags=....., SSID=ns-3-ssid
5	0.0004...	00:00:00_00:0...	00:00:00_00:0...	802.	14	Acknowledgement, Flags=0.....
6	0.0005...	00:00:00_00:0...	00:00:00_00:0...	802.	44	Association Response, SN=1, FN=0, Flags=....., SSID=Wildcard (Broadcast)
7	0.0007...	00:00:00_00:0...	00:00:00_00:0...	802.	14	Acknowledgement, Flags=0.....
8	0.0008...	00:00:00_00:0...	00:00:00_00:0...	802.	53	Association Request, SN=0, FN=0, Flags=....., SSID=ns-3-ssid
9	0.0009...	00:00:00_00:0...	00:00:00_00:0...	802.	14	Acknowledgement, Flags=0.....
10	0.0010...	00:00:00_00:0...	00:00:00_00:0...	802.	44	Association Response, SN=2, FN=0, Flags=....., SSID=Wildcard (Broadcast)
11	0.0011...	00:00:00_00:0...	00:00:00_00:0...	802.	14	Acknowledgement, Flags=0.....
12	0.0013...	00:00:00_00:0...	00:00:00_00:0...	802.	44	Association Response, SN=3, FN=0, Flags=....., SSID=Wildcard (Broadcast)
13	0.0014...	00:00:00_00:0...	00:00:00_00:0...	802.	14	Acknowledgement, Flags=0.....
14	0.1024...	00:00:00_00:0...	Broadcast	802.	61	Beacon frame, SN=4, FN=0, Flags=....., BI=100, SSID=ns-3-ssid
15	0.2048...	00:00:00_00:0...	Broadcast	802.	61	Beacon frame, SN=5, FN=0, Flags=....., BI=100, SSID=ns-3-ssid
16	0.3072...	00:00:00_00:0...	Broadcast	802.	61	Beacon frame, SN=6, FN=0, Flags=....., BI=100, SSID=ns-3-ssid
17	0.4096...	00:00:00_00:0...	Broadcast	802.	61	Beacon frame, SN=7, FN=0, Flags=....., BI=100, SSID=ns-3-ssid
18	0.5120...	00:00:00_00:0...	Broadcast	802.	61	Beacon frame, SN=8, FN=0, Flags=....., BI=100, SSID=ns-3-ssid
19	0.6144...	00:00:00_00:0...	Broadcast	802.	61	Beacon frame, SN=9, FN=0, Flags=....., BI=100, SSID=ns-3-ssid
20	0.7168...	00:00:00_00:0...	Broadcast	802.	61	Beacon frame, SN=10, FN=0, Flags=....., BI=100, SSID=ns-3-ssid
21	0.8192...	00:00:00_00:0...	Broadcast	802.	61	Beacon frame, SN=11, FN=0, Flags=....., BI=100, SSID=ns-3-ssid
22	0.9216...	00:00:00_00:0...	Broadcast	802.	61	Beacon frame, SN=12, FN=0, Flags=....., BI=100, SSID=ns-3-ssid
23	1.0240...	00:00:00_00:0...	Broadcast	802.	61	Beacon frame, SN=13, FN=0, Flags=....., BI=100, SSID=ns-3-ssid
24	1.1264...	00:00:00_00:0...	Broadcast	802.	61	Beacon frame, SN=14, FN=0, Flags=....., BI=100, SSID=ns-3-ssid
25	1.2288...	00:00:00_00:0...	Broadcast	802.	61	Beacon frame, SN=15, FN=0, Flags=....., BI=100, SSID=ns-3-ssid
26	1.3312...	00:00:00_00:0...	Broadcast	802.	61	Beacon frame, SN=16, FN=0, Flags=....., BI=100, SSID=ns-3-ssid
27	1.4336...	00:00:00_00:0...	Broadcast	802.	61	Beacon frame, SN=17, FN=0, Flags=....., BI=100, SSID=ns-3-ssid
28	1.5360...	00:00:00_00:0...	Broadcast	802.	61	Beacon frame, SN=18, FN=0, Flags=....., BI=100, SSID=ns-3-ssid
29	1.6384...	00:00:00_00:0...	Broadcast	802.	61	Beacon frame, SN=19, FN=0, Flags=....., BI=100, SSID=ns-3-ssid
30	1.7408...	00:00:00_00:0...	Broadcast	802.	61	Beacon frame, SN=20, FN=0, Flags=....., BI=100, SSID=ns-3-ssid
31	1.8432...	00:00:00_00:0...	Broadcast	802.	61	Beacon frame, SN=21, FN=0, Flags=....., BI=100, SSID=ns-3-ssid
32	1.9456...	00:00:00_00:0...	Broadcast	802.	61	Beacon frame, SN=22, FN=0, Flags=....., BI=100, SSID=ns-3-ssid
33	1.9740...	0001	0604	SNA	64	SNA device <-> Non-SNA Device
34	1.9740...	00:00:00_00:0...	00:00:00_00:0...	802.	14	Acknowledgement, Flags=0.....
35	1.9741...	0001	0604	SNA	64	SNA device <-> Non-SNA Device
36	1.9743...	0002	0604	SNA	64	SNA device <-> Non-SNA Device
37	1.9745...	00:00:00_00:0...	00:00:00_00:0...	802.	14	Acknowledgement, Flags=0.....
38	1.9760...	00:00:00_00:0...	00:00:00_00:0...	LLC	1088	S, func=RN, N(R)=0; DSAP SNA Individual, SSAP NULL LSAP Command
39	1.9760...	00:00:00_00:0...	00:00:00_00:0...	802.	14	Acknowledgement, Flags=0.....
40	1.9996...	0001	0604	SNA	64	SNA device <-> Non-SNA Device
41	1.9999...	0002	0604	SNA	64	SNA device <-> Non-SNA Device
42	2.0000...	00:00:00_00:0...	00:00:00_00:0...	802.	14	Acknowledgement, Flags=0.....

图 1.4 Third0-1.pcap

在这个捕捉文件中，可以看到 CSMA/CA 侦听，发送广播帧，发送 RTS/CTS 帧，等等报文。

ASCII 格式：

添加代码改动，并且添加 C++ 的标准输入输出流 `#include<iostream>`。

```
AsciiTraceHelper ascii;          //创建一个ASCII trace对象

pointToPoint.EnableAsciiAll (ascii.CreateFileStream("third.tr"));    //包含两个方法调用。
```

图 1.5 添加代码

会将 ASCII 格式的信息输出到 `third.tr` 里面。

```
+ 2.00813 /NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Enqueue ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header (tos 0x0 DSCP Default ECN Not-ECT tt
- 2.00813 /NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Dequeue ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header (tos 0x0 DSCP Default ECN Not-ECT tt
r 2.01182 /NodeList/1/DeviceList/0/$ns3::PointToPointNetDevice/MacRx ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header (tos 0x0 DSCP Default ECN Not-ECT ttl 63 id 0
+ 2.02385 /NodeList/1/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Enqueue ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header (tos 0x0 DSCP Default ECN Not-ECT tt
- 2.02385 /NodeList/1/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Dequeue ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header (tos 0x0 DSCP Default ECN Not-ECT tt
r 2.02674 /NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/MacRx ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header (tos 0x0 DSCP Default ECN Not-ECT ttl 63 id 0
```

图 1.6 输出信息

这里选择一条信息来加以分析：

```
+ 2.00813 /NodeList/0/DeviceList/0/$ns3::PointToPointNetDevice/TxQueue/Enqueue
ns3::PppHeader (Point-to-Point Protocol: IP (0x0021)) ns3::Ipv4Header (tos 0x0 DSCP
Default ECN Not-ECT ttl 63 id 0 protocol 17 offset (bytes) 0 flags [none] length: 1052
10.1.3.3 > 10.1.2.4) ns3::UdpHeader (length: 1032 49153 > 9) Payload (size=1024)
```

1. +：传输队列入队操作。
2. 2.00813：仿真时间，以 s 为单位。
3. /NodeList/0/DeviceList/0/\$ns3::PointToPointNetDevice/TxQueue/Enqueue：确定哪个 trace 发送端发起这个事件，\$ns3::PointToPointNetDevice 告诉我们第 0 个节点的设备列表的第 0 个位置的设备类型。入队操作在最后部分的“trace path”TxQueue/Enqueue 中体现。
4. ns3::PppHeader：表明数据包封装成点到点协议
5. Point-to-Point Protocol: IP(0x0021))
6. ns3::UdpHeader:显示数据包的 UDP 头
11. Payload (size=1024)：表明数据包数据量为 1024bytes

对于 CSMA/CA 的模拟可以使用 vis 模拟多端点的随机运动，可以可视化的查看更多的信息。

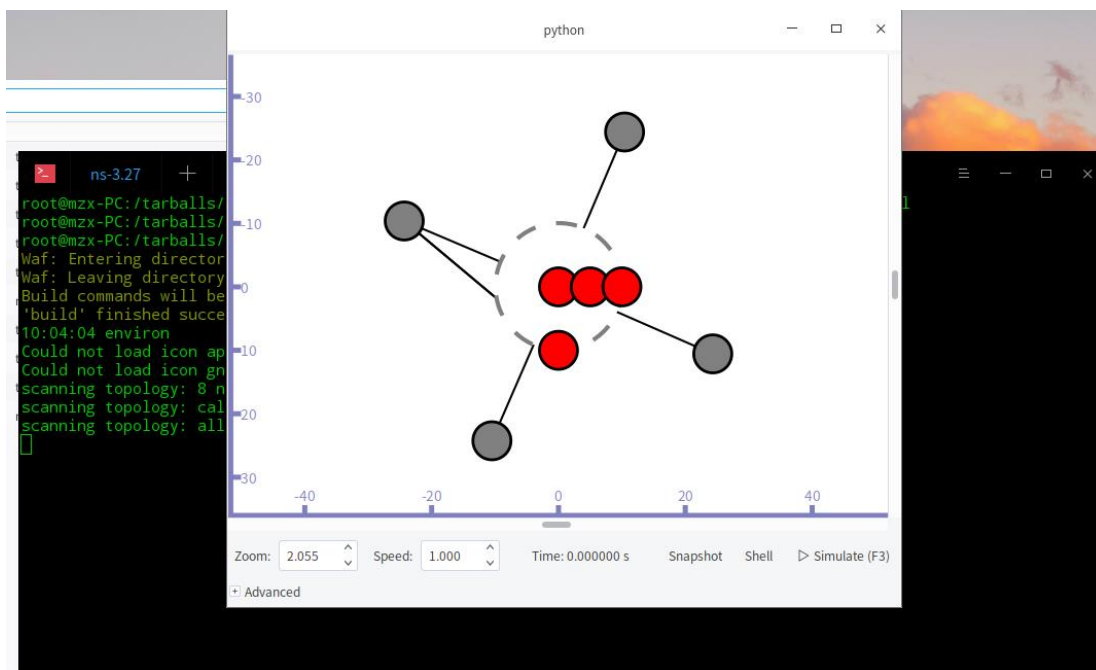


图 1.7 可视化模拟

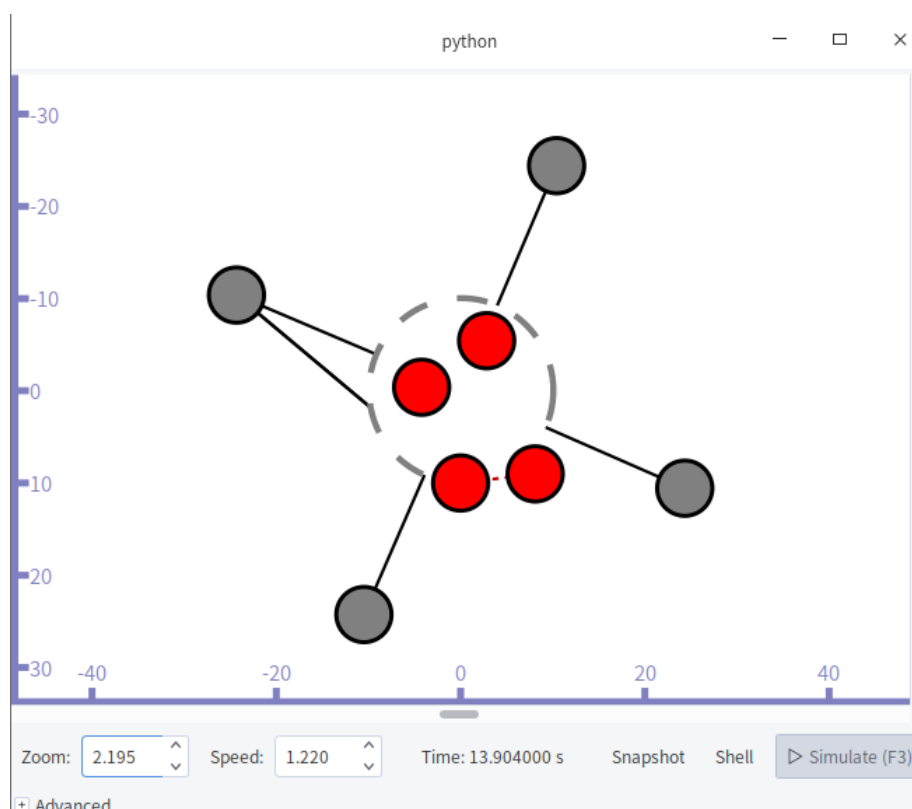


图 1.8 运行模拟

1.5 实验收获

这个实例模拟了 WiFi 建立信道时经历的过程，发出的不同帧来建立信道，让

我进一步熟悉了 NS-3 的使用，也对 CSMA/CA 的建立过程更加熟悉，同时也了解了 NS-3 的可视化模拟的使用与方法。