

****

**J I A N G S U U N I V E R S I T Y**

**“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](#_Toc45715804)

[1.1 功能介绍 1](#_Toc45715805)

[1.2 流程图 1](#_Toc45715806)

[1.3 代码分析 1](#_Toc45715807)

[1.4 实验结果 8](#_Toc45715808)

[1.5 实验收获 11](#_Toc45715809)

# 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

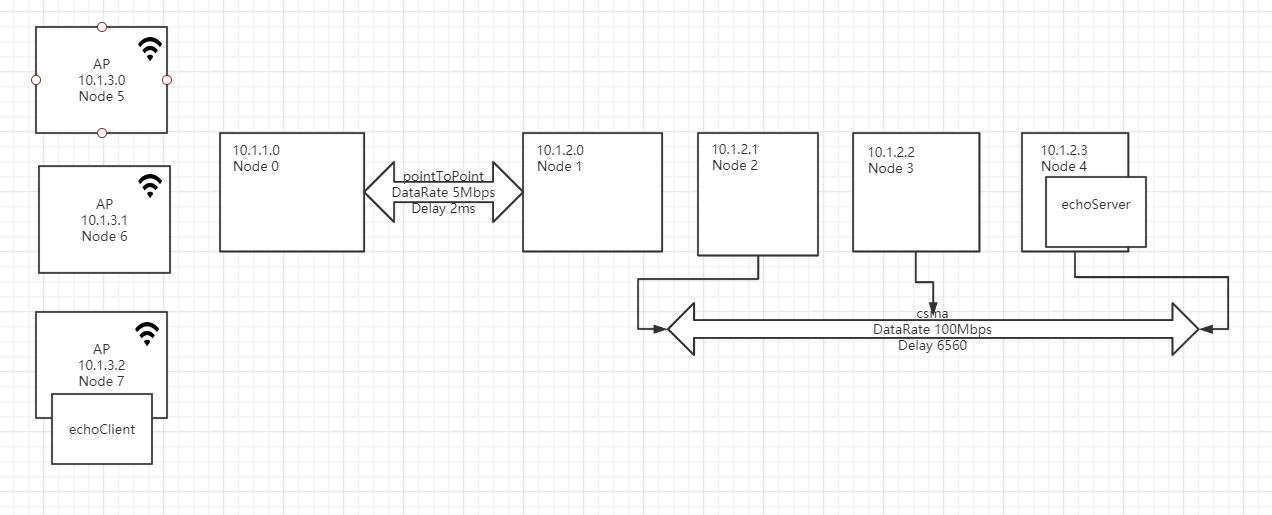


图1.1 流程图

## 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 bounding 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

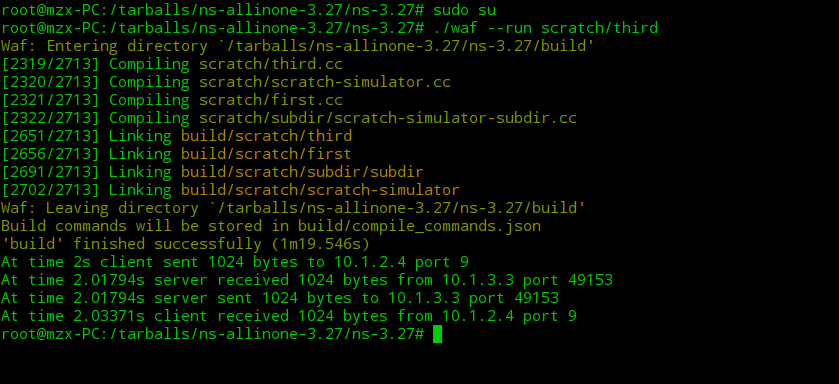


图1.2 运行程序

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

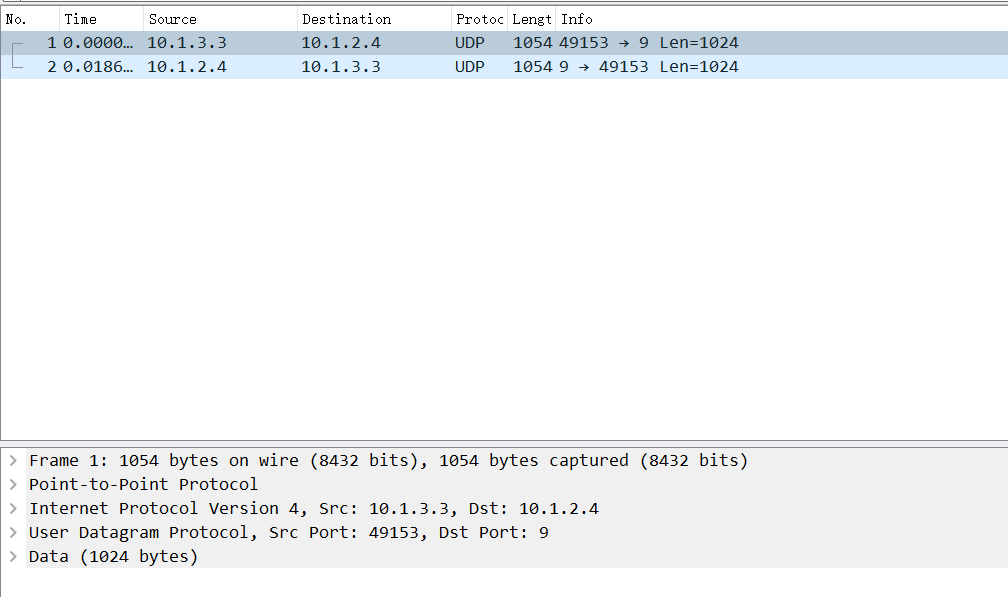


图1.3 Third0-0.pcap

可以看到，客户端向服务器发送消息，长度是1054字节。

Third0-1.pcap

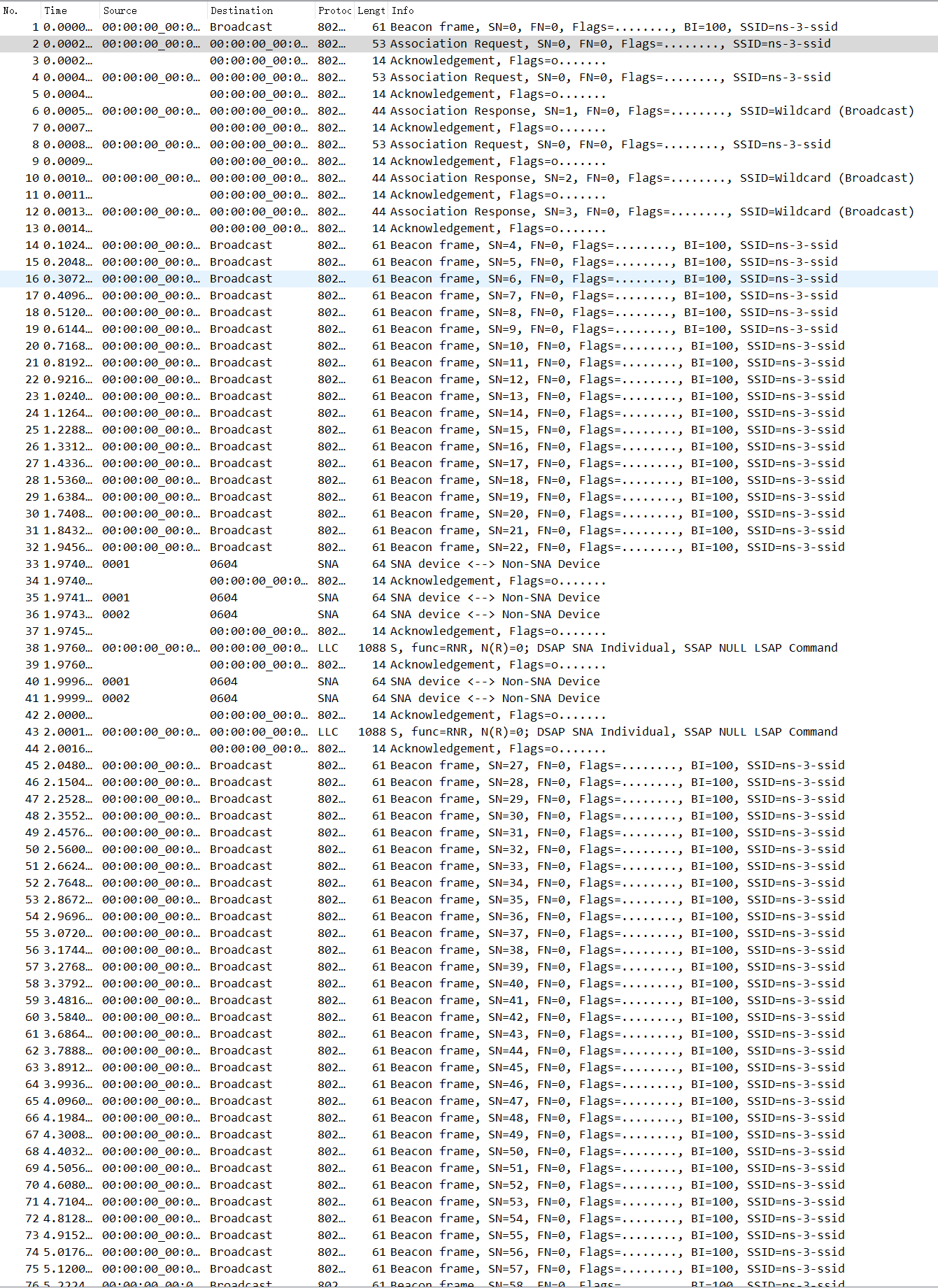


图1.4 Third0-1.pcap

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>。

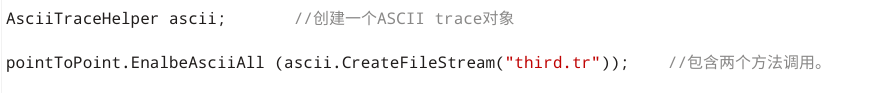


图1.5 添加代码

会将ASCII格式的信息输出到third.tr里面。

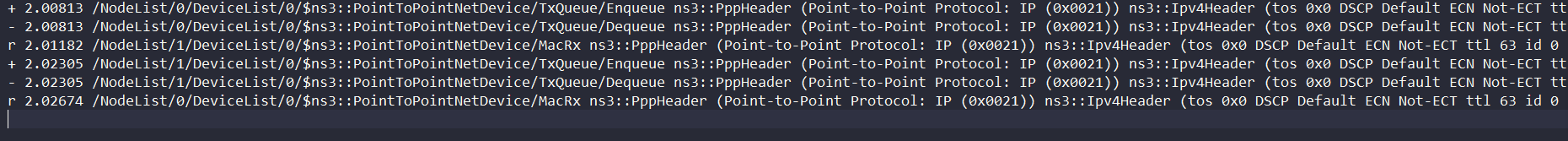


图1.6 输出信息

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。

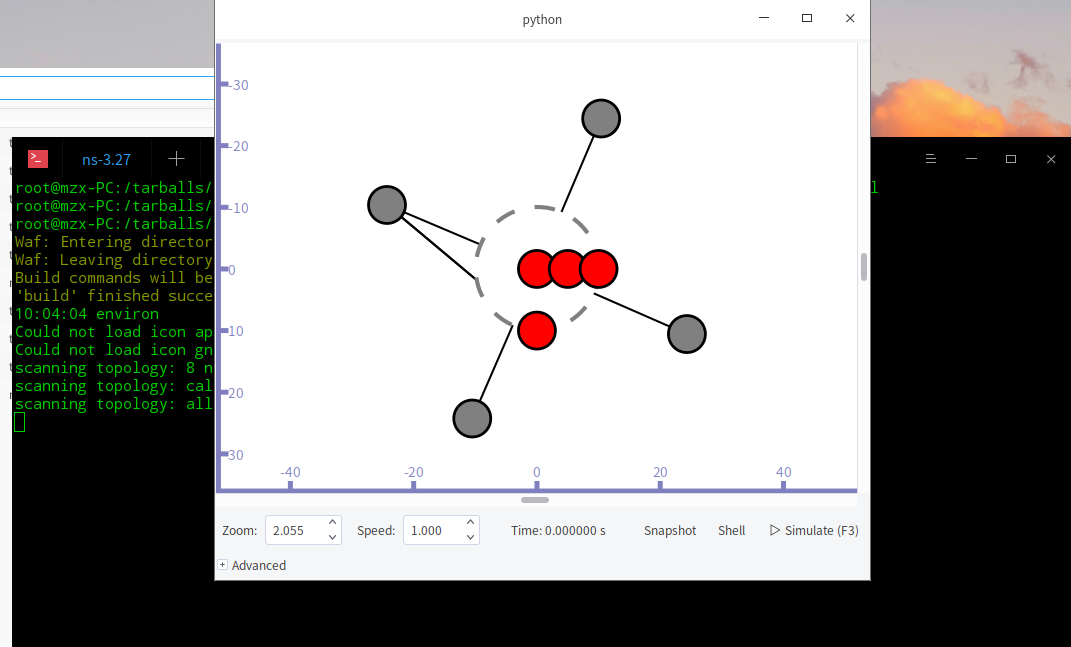


图1.7 可视化模拟

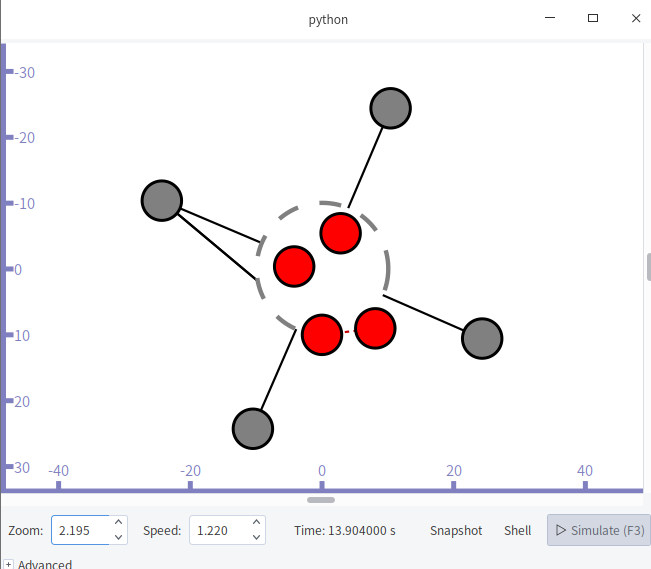


图1.8 运行模拟

## 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。