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 7 日

目录

一、NS3 内容介绍

1.1 简介

NS-3 是一款离散事件驱动的网络仿真器，主要应用于研究和教育领域，不仅提供了模型的分组数据网络和执行工作,并提供一个模拟引擎为用户进行仿真实验。

NS-3 具有以下特点：

1.NS-3 开源且向后不兼容 NS-2，两者是不同的模拟器，NS-2 使用 OTcl 语言，NS-3 全部采用 C++语言编写，并且可选择 Python 语言编写。

2.程序主要在控制台上使用与运行，支持平台只有 Linux 与 macOS。

1.2 编译器

NS3 的编译系统采用了 Waf。它是用 Python 开发的新一代编译管理系统。使用 waf 对 NS3 源代码进行编译时，可以分为优化编译和调试编译两种情况。默认情况将进行调试编译。

编译命令如下：

./waf clean

./waf -d optimized --enable-examples --enable-testsconfigure

1.3 协调工具 bake

Bake 是一个官方提供的下载最新版本的 ns-3 的协调工具，通过 git 下载之后，可以选择下载不同的版本：

- ns - 3.31 :相对应的模块释放; 它会下载 组件类似于释放 tarball。
- ns-3-dev :一个类似的模块,但使用开发代码树
- ns - allinone 3.31 :模块,包括其他可选特性。如点击路由、Openflow ns-3 和网络仿真摇篮

- ns-3-allinone :类似于 allinone 的发布版本模块,但是对于开发代码。

1.4 程序参数

```
$ ./waf --run <ns3-program> --command-template="%s <args>"
```

<ns3-program>：程序名

command-template：可以为赋值的参数

%s <args>：可以为特别命令，如%s –help

1.5 重点概念

1.5.1 节点 Node

在计算机网络术语中，任何一台连接到网络的计算设备被称为主机，被称为终端。而 NS3 是一个网络模拟器,选用了一个来源于图论,在其他网络模拟器中亦广泛使用的术语：节点，作为终端的代名词。

NS3 中基本计算设备被抽象为节点。节点由用 C++编写的 Node 类来描述。Node 类提供了用于管理计算设备的各种方法。也可以将节点设想为一台可以添加各种功能的计算机，因此可以给它添加应用程序，协议栈，外设卡及驱动程序等。

1.5.2 信道

在 NS-3 中,信道可以把节点连接到代表数据交换信道的对象上,用 C++编写的 Channel 类来实现。它提供了管理通信子网对象和把节点连接至信道的各种方法。信道类同样可以由开发者以面向对象的方法自定义。一个信道实例可以模拟一条简单的线缆，也可以模拟一个复杂的巨型以太网交换机，甚至无线网络中充满障碍物的三维空间。

1.5.3 网络设置

在 NS3 中，网络设备这一抽象概念相当于硬件设备和软件驱动的总和。网络设备由用 C++编写的 NetDevice 类来描述。NetDevice 类提供了管理连接其他节点和信道对象的各种方法，并且允许开发者以面向对象的方法来自定义。

NS3 仿真环境中,网络设备相当于安装在节点上,使得节点通过信道和其他节点通信。

像真实的计算机一样，一个节点可以通过多个网络设备同时连接到多条信道上。

在其中有几个特定的网络设备的实例，它们分别是CsmaNetDevice,PointToPointNetDevice, 和 WifiNetDevice。其中 CsmaNetDevice 被设计成在 csma 信道中工作，而 PointToPointNetDevice 在 PointToPoint 信道中工作，WifiNetNevice 在 wifi 信道中工作。

1.5.4 应用程序

在 NS3 中，需要被仿真的用户程序被抽象为应用。用 Application 类来描述。这个类提供了管理仿真过程中用户层应用的各种方法。开发者应当用面向对象的方法自定义和创建新的应用。

在这里使用 Application 类的两个实例：UdpEchoClientApplication 和 UdpEchoServerApplication。这些应用程序包含了一个 client 应用和一个 server 应用来发送和回应仿真网络中的数据包。

二、NS3 实例运行分析

2.1 简单的点对点传输数据包实例

2.1.1 功能介绍

这一个实例会在两个节点间创建一个简单的点到点的连接，并且在这两个节点之间传送一个数据包，作为一个入门的实例，格式比较普遍，也比较简洁。

2.1.2 流程图

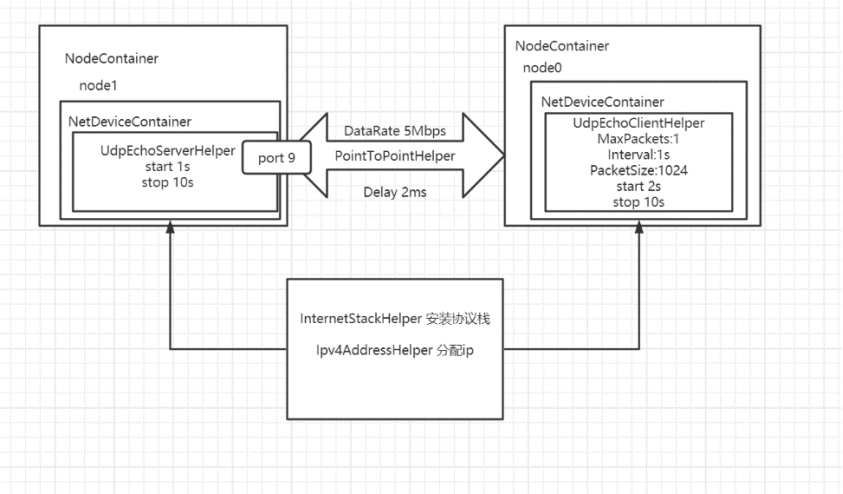

图 2.1 first.cc 流程图

2.1.3 代码分析

```
1.  /* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
2.  /*
3.   * This program is free software; you can redistribute it and/or modify
4.   * it under the terms of the GNU General Public License version 2 as
5.   * published by the Free Software Foundation;
6.   *
7.   * This program is distributed in the hope that it will be useful,
8.   * but WITHOUT ANY WARRANTY; without even the implied warranty of
9.   * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
10.  * GNU General Public License for more details.
11.  *
12.  * You should have received a copy of the GNU General Public License
13.  * along with this program; if not, write to the Free Software
14.  * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
15.  */
16.
17. //递归加载库
18. #include "ns3/core-module.h"
19. #include "ns3/network-module.h"
20. #include "ns3/internet-module.h"
21. #include "ns3/point-to-point-module.h"
22. #include "ns3/applications-module.h"
23.
```

```
24. //声明命名空间
25. using namespace ns3;
26.
27. //日志记录模块
28. NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");
29.
30. int
31. main (int argc, char *argv[])    //main 函数
32. {
33.   CommandLine cmd;
34.   cmd.Parse (argc, argv);
35.
36.   Time::SetResolution (Time::NS);    //最小时间间隔，默认为 1ns
37.   //启用两个日志记录组件 构建客户机和回声回声服务器应用程序
38.   LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
39.   LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
40.
41.   NodeContainer nodes;    //模拟电脑对象结点，是一个拓扑辅助，因为互联网中的常常是点对点式
    的，所以用 node
42.   nodes.Create (2);    //新建立 2 个节点指针
43.
44.   /*
45.    *  此处的 pointToPoint 为点对点设备的连接，在现实世界中常为网线（双绞线等等设备），或者
    WiFi
46.    *  我们可以通过一个 PointToPointHelper 配置和连接 ns-
    3 PointToPointNetDevice 和 PointToPointChannel
47.    */
48.   PointToPointHelper pointToPoint;
49.   //PointToPointHelper 对象使用值 5 mbps DataRate 创建一个 PointToPointNetDevice 对象
50.   pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
51.   //设置传播延迟为 2ms
52.   pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
53.
54.   /*
55.    * 使用 NetDeviceContainer 来保存创建的节点
56.    * 为每个节点 NodeContainer 创建 PointToPointNetDevice 并保存在设备容器
57.    */
58.   NetDeviceContainer devices;
59.   devices = pointToPoint.Install (nodes);
60.
61.   /*
62.    * 安装拓扑辅助网络栈
```

```
63.    * 给上面创建的 nodeContainer 的每个节点安装包括 TCP,UDP,IP 的协议栈
64.    */
65.   InternetStackHelper stack;
66.   stack.Install (nodes);
67.
68.   /* 声明可以分配的 ipv4 地址，是一个帮助器，地址由 10.1.1.0 开始递增（不能重复），子网掩码设置为 255.255.255.0*/
69.   Ipv4AddressHelper address;
70.   address.SetBase ("10.1.1.0", "255.255.255.0");
71.   //实际分配给设备 ipv4 地址，使用一个接口，同时记录的设备与 ip 的信息，像是路由表
72.   Ipv4InterfaceContainer interfaces = address.Assign (devices);
73.
74.   //创建一个服务器应用程序，等待输入的 UDP 数据包，并将其发送回原始发件人
75.   UdpEchoServerHelper echoServer (9);
76.
77.   //Install()方法执行，初始化回显服务器的应用，并将应用连接到一个节点上去;包含一个隐式转换；安装一个 UdpEchoServerApplication
78.   ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
79.   serverApps.Start (Seconds (1.0));    //应用在 1s 时生效
80.   serverApps.Stop (Seconds (10.0));    //10s 时停止
81.
82.   //创建一个 UdpEchoClientHelper 的对象，令其设置客户端的远端地址为服务器节点的 IP 地址。
83.   UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
84.   //告诉客户我们允许它发送的最大数目的包
85.   echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
86.   //客户端数据包之间要等多长时间
87.   echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
88.   //客户端大数据包的有效载荷
89.   echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
90.   //如 echo 服务端一样，我们告诉客户端开始和停止时间
91.   ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
92.   clientApps.Start (Seconds (2.0));    //时间为 2s 的时候开始
93.   clientApps.Stop (Seconds (10.0));    //10s 时结束
94.
95.   /*
96.    * 当 Simulator::Run 被调用时，系统会开始遍历预设事件的列表并执行。
97.    * 在 1s 时会使 echo 服务端应用生效
98.    * 在 2s 时让 echo 客户端应用开始
99.    */
100.   Simulator::Run ();
101.   //因为这个程序只发送一个数据包，所以执行完毕后会全局调用模拟器销毁
102.   Simulator::Destroy ();
103.   return 0;
```

```
104.   }
```

2.1.4 运行结果

```
root@mzx-PC:/tarballs/ns-allinone-3.27/ns-3.27# sudo ./waf --run scratch/f
irst
Waf: Entering directory `/tarballs/ns-allinone-3.27/ns-3.27/build'
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 (2.411s)
At time 2s client sent 1024 bytes to 10.1.1.2 port 9
At time 2.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time 2.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time 2.00737s client received 1024 bytes from 10.1.1.2 port 9
root@mzx-PC:/tarballs/ns-allinone-3.27/ns-3.27#
```

图 2.2 运行结果

在上图日志组件中显示,在 2s 时，客户端将一个 1024 字节的数据包通过端口 9 发送到回显服务器 10.1.1.2。在 2.00369s 时，服务器从 10.1.1.1 通过 49153 端口收到了 1024 字节。在像上面一样，回声服务器又重新发包，客户端接收。

打开更详细的日志记录设置

```
root@mzx-PC:/tarballs/ns-allinone-3.27/ns-3.27# export NS_LOG=UdpEchoClien
tApplication=level_all
```

图 2.3 打开日志记录等级

```
root@mzx-PC:/tarballs/ns-allinone-3.27/ns-3.27# ./waf --run scratch/first
Waf: Entering directory `/tarballs/ns-allinone-3.27/ns-3.27/build'
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 (2.841s)
UdpEchoClientApplication:UdpEchoClient(0x55e640156d70)
UdpEchoClientApplication:SetDataSize(0x55e640156d70, 1024)
UdpEchoClientApplication:StartApplication(0x55e640156d70)
UdpEchoClientApplication:ScheduleTransmit(0x55e640156d70, +0.0ns)
UdpEchoClientApplication:Send(0x55e640156d70)
At time 2s client sent 1024 bytes to 10.1.1.2 port 9
At time 2.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time 2.00369s server sent 1024 bytes to 10.1.1.1 port 49153
UdpEchoClientApplication:HandleRead(0x55e640156d70, 0x55e640158bc0)
At time 2.00737s client received 1024 bytes from 10.1.1.2 port 9
UdpEchoClientApplication:StopApplication(0x55e640156d70)
UdpEchoClientApplication:DoDispose(0x55e640156d70)
UdpEchoClientApplication:~UdpEchoClient(0x55e640156d70)
```

图 2.4 日志

在这个日志中，信息显示的更加详细，可以看到 udpclient 设置自身的属性及发送的过程，其余与上文中的过程类似，就不再赘述。

2.1.5 实验收获

作为 ns-3 最简单的入门实验，这个实验让我了解了基本构造实验中节点，以及模拟通信的过程，并且了解如何使用日志组件。

2.2 实现 CSMA 的以太信道

2.2.1 功能介绍

本实例构造了一个 p2p 链路及 3 个其余节点在 CSMA 信道中的表现，模拟了网络节点在 CSMA 信道中安排路由表，网络信息的过程。

2.2.2 流程图

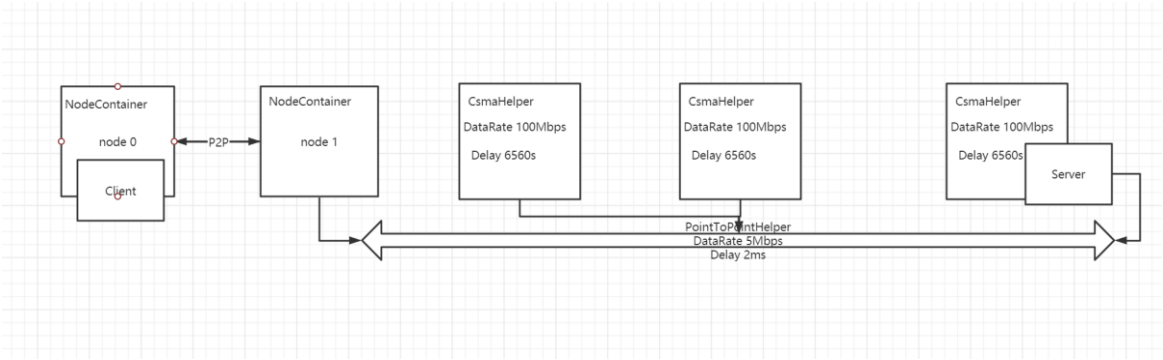

图 2.5 second.cc 流程图

2.2.3 代码分析

```
1.  /* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
2.  /*
3.   * This program is free software; you can redistribute it and/or modify
4.   * it under the terms of the GNU General Public License version 2 as
5.   * published by the Free Software Foundation;
6.   *
7.   * This program is distributed in the hope that it will be useful,
8.   * but WITHOUT ANY WARRANTY; without even the implied warranty of
9.   * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
10.  * GNU General Public License for more details.
11.  *
```

```
 * 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
 */

//导入所需的包
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/ipv4-global-routing-helper.h"

// Default Network Topology
//
//       10.1.1.0
// n0 -------------- n1   n2   n3   n4
//    point-to-point  |    |    |    |
//                    ================
//                      LAN 10.1.2.0

//声明空间和设置日志组件
using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("SecondScriptExample");


int
main (int argc, char *argv[])
{
  //通过 verbose 这个 flag 控制 UdpEchoClientApplication 和 UdpEchoServerApplication 启用日志记录组件，默认是开
  bool verbose = true;
  uint32_t nCsma = 3;
  //输出信息
  CommandLine cmd;
  cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
  cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);

  cmd.Parse (argc,argv);
  //如果 verbose 为 false，就不输出 log 了
  if (verbose)
```

```
54.     {
55.       LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
56.       LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
57.     }
58.
59.   nCsma = nCsma == 0 ? 1 : nCsma;
60.
61.   //创建使用 P2P 链路链接的 2 个 node
62.   NodeContainer p2pNodes;
63.   p2pNodes.Create (2);
64.
65.   //声明 csma 的节点，也是总线的一部分
66.   NodeContainer csmaNodes;
67.   //将之前 P2P 的 NodeContianer 的第二个节点添加到 CSMA 的 NodeContainer，
68.   //以获得 CSMA device;这个 node 将会有两个 device
69.   csmaNodes.Add (p2pNodes.Get (1));
70.   //创建一个额外节点组成的其余部分 CSMA 网络，包含 3 个 node
71.   csmaNodes.Create (nCsma);
72.
73.   //帮助创建一套 PointToPointNetDevice 对象，设置传送速率和信道延迟
74.   PointToPointHelper pointToPoint;
75.   //设置通道传输率为 5M
76.   pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
77.   //设置传输延迟为 2ms
78.   pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
79.
80.   //安装 P2P 网卡设备到 P2P 网络节点
81.   NetDeviceContainer p2pDevices;
82.   p2pDevices = pointToPoint.Install (p2pNodes);
83.
84.   //CsmaHelper 工作只是就像一个 PointToPointHelper,但它创建和 CSMA 设备和连接频道
85.   CsmaHelper csma;
86.   //数据速率为每秒 100 比特
87.   csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
88.   csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));
89.
90.   //使用 NetDeviceContainer 来保存创建的节点
91.   NetDeviceContainer csmaDevices;
92.   csmaDevices = csma.Install (csmaNodes);
93.
94.   //安装协议栈
95.   InternetStackHelper stack;
96.   stack.Install (p2pNodes.Get (0));
```

```
97.   stack.Install (csmaNodes);
98.
99.   //分配 ipv4 地址，从 101.1.1.0 开始分配，子网掩码为 255.255.255.0
100.   Ipv4AddressHelper address;
101.   address.SetBase ("10.1.1.0", "255.255.255.0");
102.   Ipv4InterfaceContainer p2pInterfaces;
103.   p2pInterfaces = address.Assign (p2pDevices);
104.   //将 IP 地址分配给 CSMA 设备接口
105.   address.SetBase ("10.1.2.0", "255.255.255.0");
106.   Ipv4InterfaceContainer csmaInterfaces;
107.   csmaInterfaces = address.Assign (csmaDevices);
108.
109.   //建立一个服务器
110.   UdpEchoServerHelper echoServer (9);
111.   //安装节点，在 1s 时启动，10s 时结束，安装在 CSMA 网段的最后一个节点上
112.   ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma));
113.   serverApps.Start (Seconds (1.0));
114.   serverApps.Stop (Seconds (10.0));
115.
116.   //通过 helper 拿到节点地址，设置最大包的值为 1，时间间隔为 1s,包大小 1024
117.   UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);
118.   echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
119.   echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
120.   echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
121.
122.   //安装客户端服务器，2s 时开始，10s 时结束，安装在 P2P 网段的第一个节点上
123.   ApplicationContainer clientApps = echoClient.Install (p2pNodes.Get (0));
124.   clientApps.Start (Seconds (2.0));
125.   clientApps.Stop (Seconds (10.0));
126.
127.   //如 OSPF 一样建立路由表
128.   Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
129.
130.   //启用 pcap 跟踪
131.   pointToPoint.EnablePcapAll ("second");
132.   //设置额外的参数
133.   csma.EnablePcap ("second", csmaDevices.Get (1), true);
134.
135.   //运行，清理，结束
136.   Simulator::Run ();
137.   Simulator::Destroy ();
138.   return 0;
139.   }
```

2.2.4 运行结果

```
root@mzx-PC:/tarballs/ns-allinone-3.27/ns-3.27# sudo ./waf --run scratch/s
econd
Waf: Entering directory `/tarballs/ns-allinone-3.27/ns-3.27/build'
[ 956/2711] Compiling scratch/subdir/scratch-simulator-subdir.cc
[ 957/2711] Compiling scratch/scratch-simulator.cc
[2320/2711] Compiling scratch/second.cc
[2321/2711] Compiling scratch/first.cc
[2646/2711] Linking build/scratch/second
[2685/2711] Linking build/scratch/first
[2686/2711] Linking build/scratch/scratch-simulator
[2687/2711] Linking build/scratch/subdir/subdir
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 (1m21.065s)
At time 2s client sent 1024 bytes to 10.1.2.4 port 9
At time 2.0078s server received 1024 bytes from 10.1.1.1 port 49153
At time 2.0078s server sent 1024 bytes to 10.1.1.1 port 49153
At time 2.01761s client received 1024 bytes from 10.1.2.4 port 9
root@mzx-PC:/tarballs/ns-allinone-3.27/ns-3.27#
```

图 2.6 运行结果

这个实验中，CsmaChannel 信道模拟了用于一个可以实现载波侦听多路访问通信子网中的媒介。在这个实例中，看起来与 first.cc 毫无差异，实际上经过了多道转发的过程。

在实例运行的过程中，使用 pcap 将过程记录下来，可以进一步观察运行的过程。

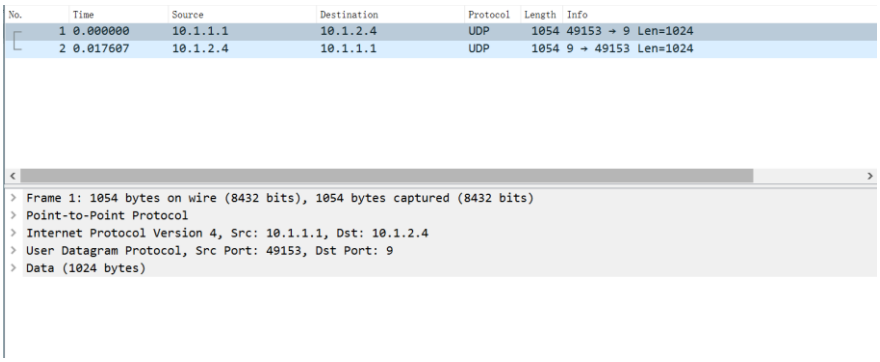

图 2.7 pcap 捕捉

在这里 P2P 的 0 节点将长度 1024 的包发给了 CSMA 最后一个节点，而服务器同样返回节点。

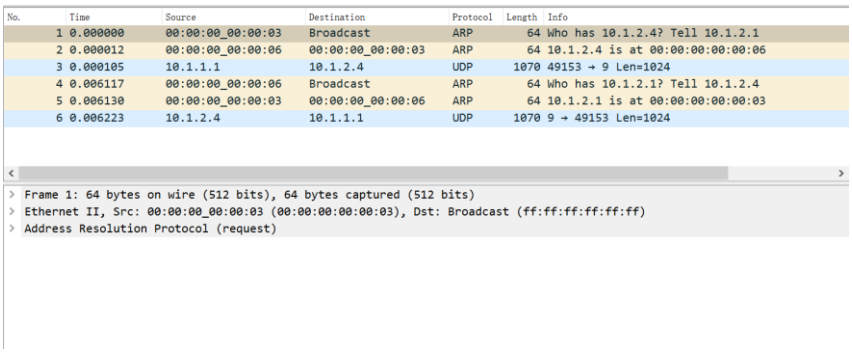

图 2.8 pcap 捕捉

首先，可以观察到客户端发出 ARP 请求，并接收到 ARP 响应。收到后发送数据包，服务器收到后也通过 ARP 请求与响应返回一个包。

2.2.5 实验收获

通过这个实例，我进一步了解了 csma 模拟时的运行状态，以及如何获取并分析 pcap 文件，来进一步了解网络节点的通信过程。