

****

**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7日**

目录

[一、NS3内容介绍 1](#_Toc45372274)

[1.1 简介 1](#_Toc45372275)

[1.2 编译器 1](#_Toc45372276)

[1.3 协调工具bake 1](#_Toc45372277)

[1.4 程序参数 2](#_Toc45372278)

[1.5 重点概念 2](#_Toc45372279)

[1.5.1 节点Node 2](#_Toc45372280)

[1.5.2 信道 2](#_Toc45372281)

[1.5.3 网络设置 2](#_Toc45372282)

[1.5.4 应用程序 3](#_Toc45372283)

[二、NS3实例运行分析 3](#_Toc45372284)

[2.1 简单的点对点传输数据包实例 3](#_Toc45372285)

[2.1.1 功能介绍 3](#_Toc45372286)

[2.1.2 流程图 4](#_Toc45372287)

[2.1.3 代码分析 4](#_Toc45372288)

[2.1.4 运行结果 7](#_Toc45372289)

[2.1.5 实验收获 8](#_Toc45372290)

[2.2实现CSMA的以太信道 8](#_Toc45372291)

[2.2.1 功能介绍 8](#_Toc45372292)

[2.2.2 流程图 8](#_Toc45372293)

[2.2.3 代码分析 8](#_Toc45372294)

[2.2.4 运行结果 12](#_Toc45372295)

[2.2.5 实验收获 13](#_Toc45372296)

# 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

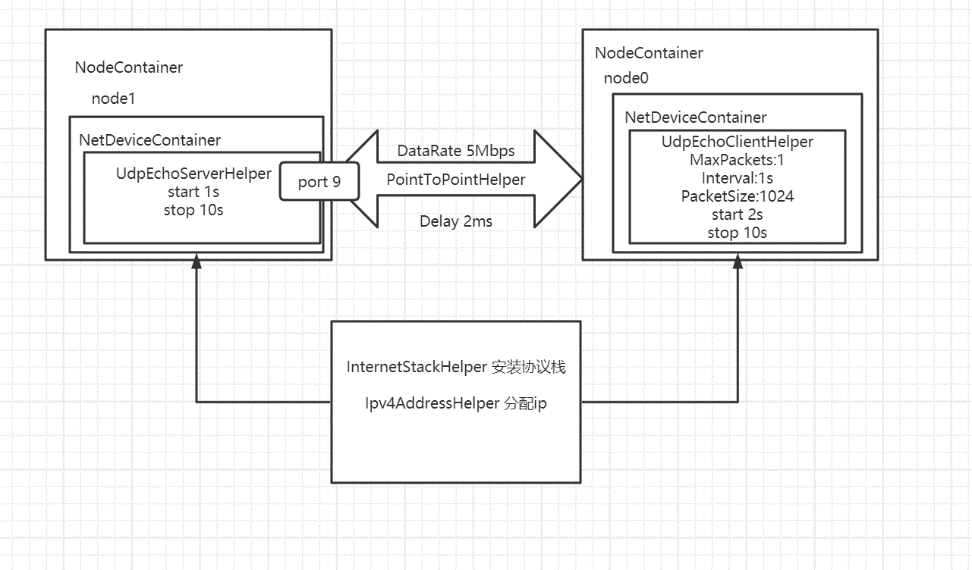


图2.1 first.cc 流程图

### 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. \*/
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"
24. //声明命名空间
25. **using** **namespace** ns3;
27. //日志记录模块
28. NS\_LOG\_COMPONENT\_DEFINE ("FirstScriptExample");
30. **int**
31. main (**int** argc, **char** \*argv[])    //main函数
32. {
33. CommandLine cmd;
34. cmd.Parse (argc, argv);
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);
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
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"));
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);
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);
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);
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);
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
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
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

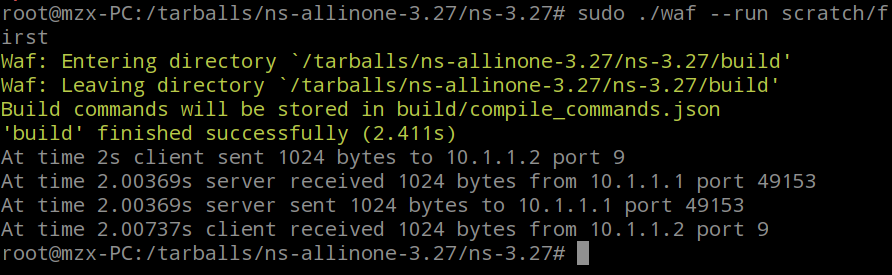


图2.2 运行结果

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



图2.3 打开日志记录等级

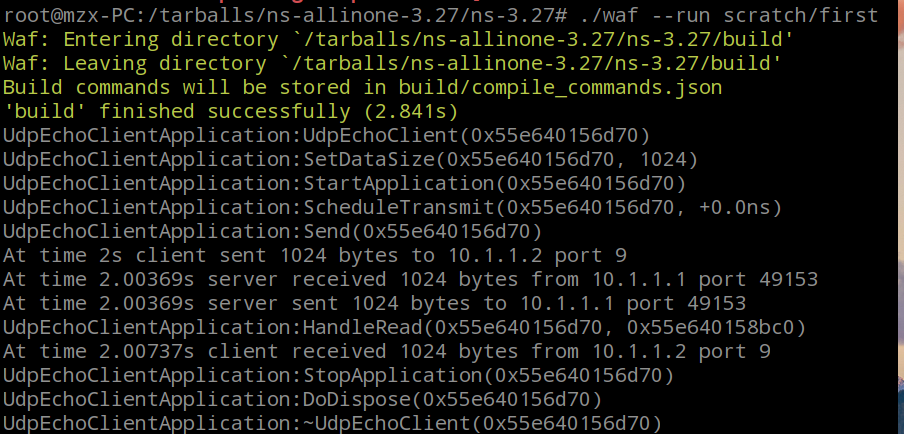


图2.4 日志

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

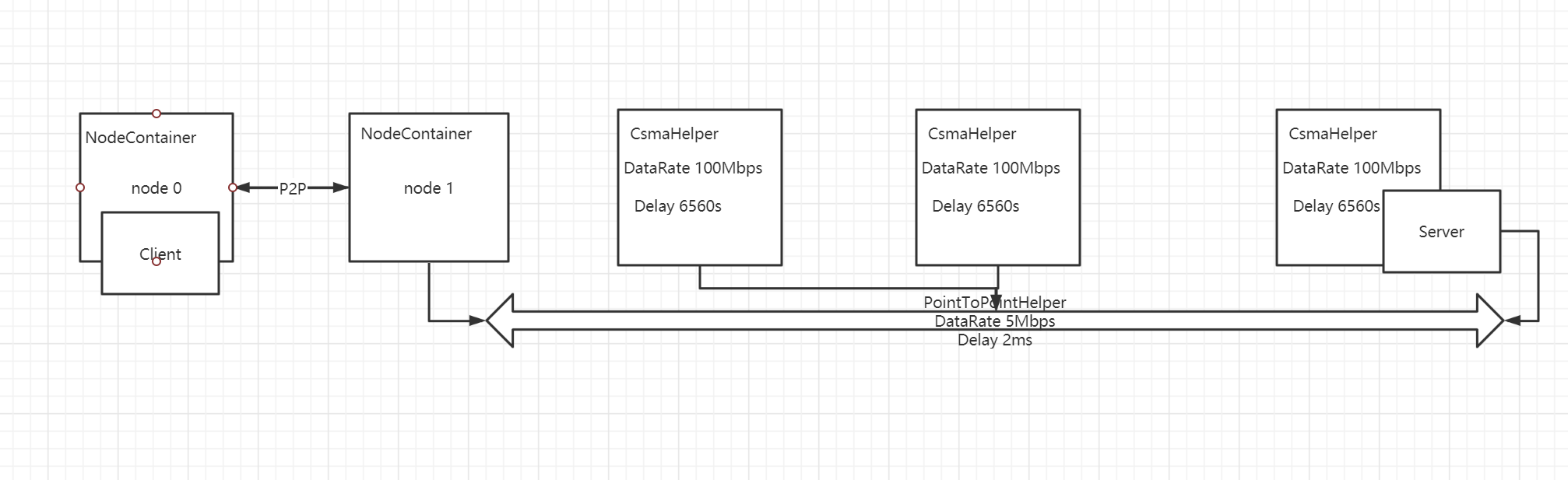


图2.5 second.cc流程图

### 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. \*
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. \*/
17. //导入所需的包
18. #include "ns3/core-module.h"
19. #include "ns3/network-module.h"
20. #include "ns3/csma-module.h"
21. #include "ns3/internet-module.h"
22. #include "ns3/point-to-point-module.h"
23. #include "ns3/applications-module.h"
24. #include "ns3/ipv4-global-routing-helper.h"
26. // Default Network Topology
27. //
28. //       10.1.1.0
29. // n0 -------------- n1   n2   n3   n4
30. //    point-to-point  |    |    |    |
31. //                    ================
32. //                      LAN 10.1.2.0
34. //声明空间和设置日志组件
35. **using** **namespace** ns3;
37. NS\_LOG\_COMPONENT\_DEFINE ("SecondScriptExample");

40. **int**
41. main (**int** argc, **char** \*argv[])
42. {
43. //通过verbose这个flag控制UdpEchoClientApplication 和 UdpEchoServerApplication 启用日志记录组件，默认是开
44. **bool** verbose = **true**;
45. uint32\_t nCsma = 3;
46. //输出信息
47. CommandLine cmd;
48. cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
49. cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
51. cmd.Parse (argc,argv);
52. //如果verbose为false，就不输出log了
53. **if** (verbose)
54. {
55. LogComponentEnable ("UdpEchoClientApplication", LOG\_LEVEL\_INFO);
56. LogComponentEnable ("UdpEchoServerApplication", LOG\_LEVEL\_INFO);
57. }
59. nCsma = nCsma == 0 ? 1 : nCsma;
61. //创建使用P2P链路链接的2个node
62. NodeContainer p2pNodes;
63. p2pNodes.Create (2);
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);
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"));
80. //安装P2P网卡设备到P2P网络节点
81. NetDeviceContainer p2pDevices;
82. p2pDevices = pointToPoint.Install (p2pNodes);
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)));
90. //使用NetDeviceContainer来保存创建的节点
91. NetDeviceContainer csmaDevices;
92. csmaDevices = csma.Install (csmaNodes);
94. //安装协议栈
95. InternetStackHelper stack;
96. stack.Install (p2pNodes.Get (0));
97. stack.Install (csmaNodes);
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);
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));
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));
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));
127. //如OSPF一样建立路由表
128. Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
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**);
135. //运行，清理，结束
136. Simulator::Run ();
137. Simulator::Destroy ();
138. **return** 0;
139. }

### 2.2.4 运行结果

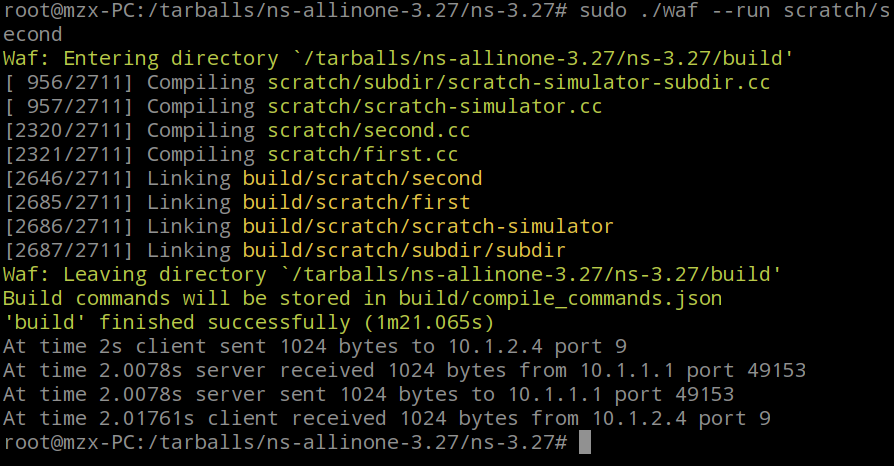


图2.6 运行结果

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。

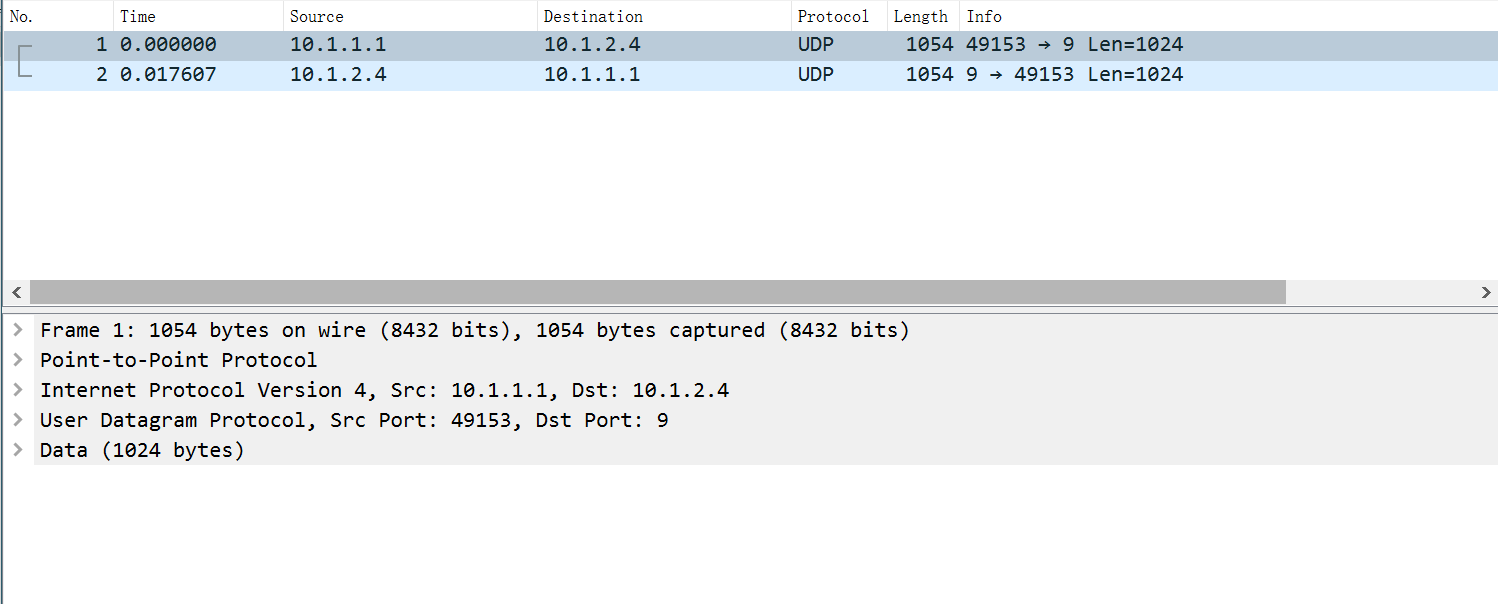


图2.7 pcap捕捉

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。

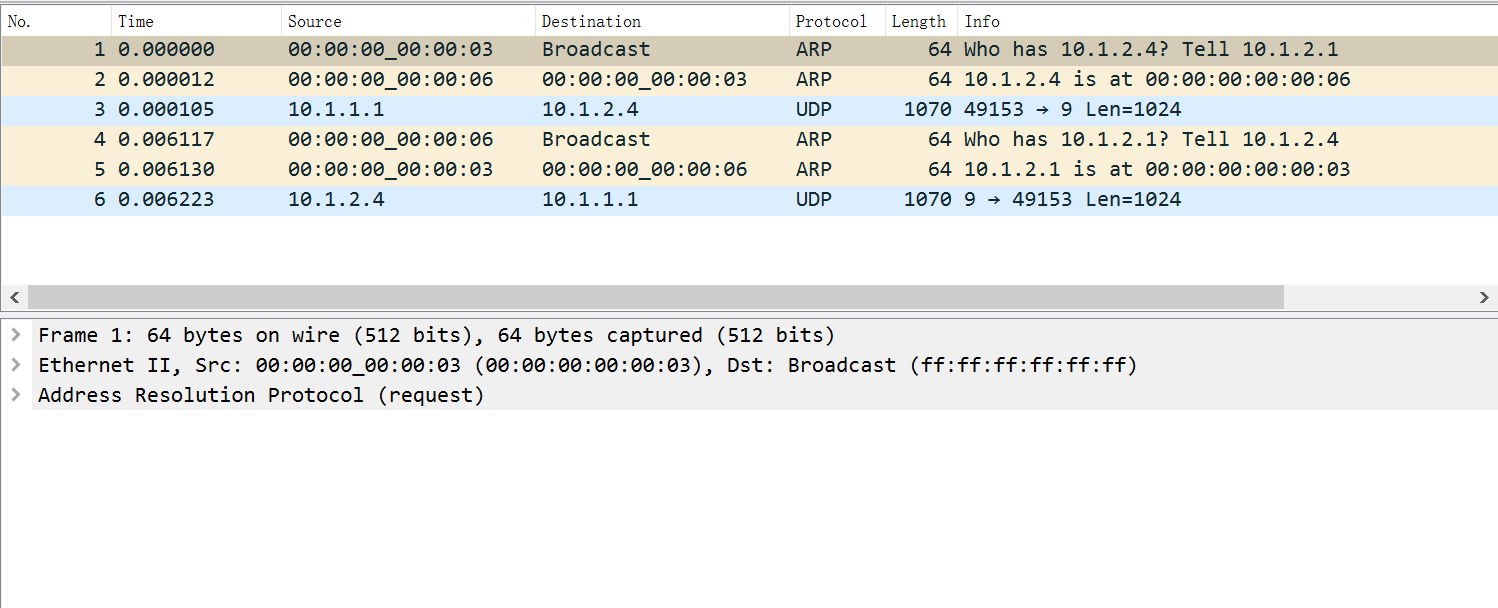


图2.8 pcap捕捉

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。