

链路层抓包及协议分析

第四章实验 1 报告

赖泽强组

1 实验目的

- 实现链路层抓包过程。
- 了解各个层的字段构成及报文格式。
- 理论结合实现，提高实践能力。

2 实验内容

利用 WinPcap 实现网络数据链路层帧捕获，显示分析帧和上层包结构。

首先屏幕显示当前配置的网络适配器，并要求选择捕获适配器编号，按照捕获帧的层次关系显示以下信息：

- 数据链路层（MAC 子层）层结构及各个字段的值
- 网络层分组的格式及各个字段的值
- 运输层报文段的格式及各个字段的值
- 应用层报文格式及各个字段的值

3 实验原理

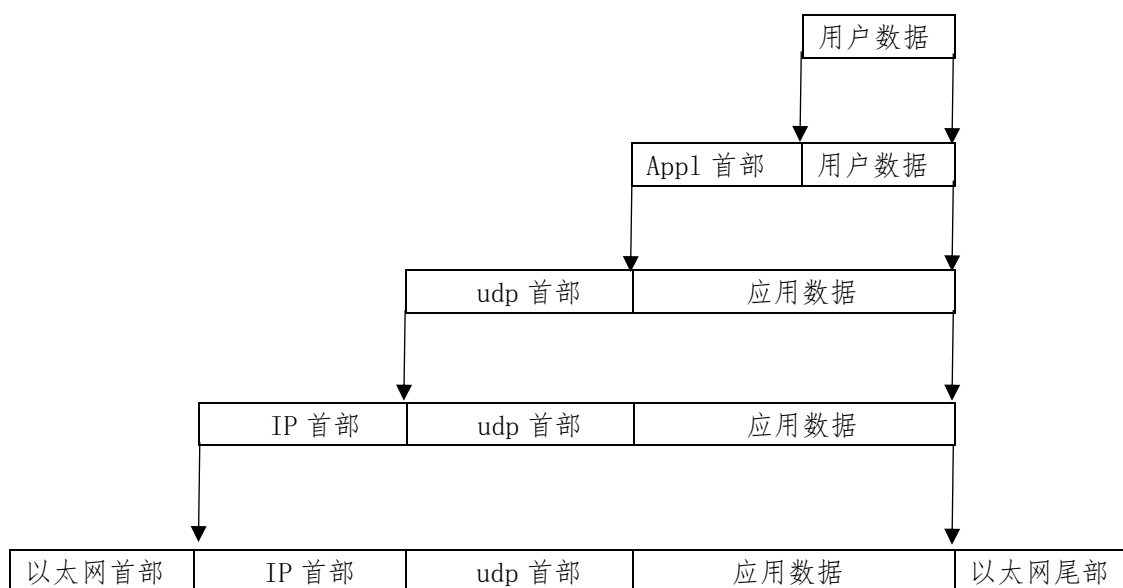
3.1 以太网帧

在以太网链路上的数据包称作以太网帧。以太网起始部分由前导码和帧开始符组成（实验中不考虑）。后面紧跟着一个以太网报头，以 MAC 地址说明目的地址和源地址。帧的中部是该帧负载的包含其他协议报头的数据包（如 IP 协议）。以太网由一个 32 位冗余校验码结尾。它用于检验数据传输是否出现损坏。

3.2 实验原理图

从链路层向上分别为网络层，传输层和应用层，本实验中网络层选择的是 IP 协议，传输层选择的是 UDP 协议。

不同的协议层对数据包有不同的称谓，在传输层叫做段（segment），在网络层叫做数据报（datagram），在链路层叫做帧（frame）。数据封装成帧后发到传输介质上，到达目的主机后每层协议再剥掉相应的首部，最后将应用层数据交给应用程序处理。通信过程中，每层协议都要加上一个数据首部，称为封装。如下图所示：



抓包所获得的数据即为以太网帧的数据，从下层到上层依次输出各个层的格式及字段值。（从下到上依次为数据链路层，网络层，传输层和应用层）

4 实验环境

4.1 开发环境

语言	环境
C++	Windows 10 1803 Visual Studio 2017 (15.6.6)
Java	Windows 10 1803 IntelliJ IDEA 2018.3.3
Python	Windows 10 1803 Visual Studio code Python 3.7

4.2 部署环境

可在 Windows, Linux, macOS 三平台正确编译运行。

5 实验步骤

由金靖轩组编写出 C++版本的链路层抓包及协议分析程序。李安腾组和刘文卓组分别写出 Java 和 Python 版本。

5.1 代码简单流程

1. 定义各个层的协议头。
2. 获取所有可用的适配器，并打印出来。
3. 打开程序中输入所选择的适配器。
4. 利用 pcap_loop 函数进行抓包。
5. 每抓到一个数据包，调用函数 packet_handler 进行接收。
6. 从链路层到应用层依次输出协议头的数据。

5.2 协议头的定义

(1) 链路层协议头 (14 字节)

```
typedef struct mac_header {  
    u_char dest_addr 6 ;  
    u_char src_addr 6 ;  
    u_char type 2 ;  
} mac_header;
```

(2) 网络层协议头 (20 字节)

```
typedef struct ip_header {  
    u_char ver_ihl;           //Version (4 bits) + Internet header length  
    u_char tos;               //Type of service  
    u_short tlen;             //Total length  
    u_short identification;   //Identification  
    u_short flags_fo;         //Flags (3 bits) + Fragment offset (13 bits)  
    u_char ttl;               //Time to live  
    u_char proto;             //Protocol  
    u_short crc;              // eader chec sum  
    u_char saddr 4 ;          //Source address  
    u_char daddr 4 ;          //Destination address  
} ip_header;
```

(3) 传输层协议头 (8 字节)

```
typedef struct udp_header{  
    u_short sport;           // 源端口 (Source port)  
    u_short dport;           // 目的端口 (Destination port)  
    u_short len;              // UDP 数据包长度 (Datagram length)  
    u_short crc;              // 校验和 (Checksum)  
    //u_int applied_data;     // 应用数据部分  
}udp_header;
```

5.3 抓包实现

```
//这里 pcap_loop 的作用是抓包，每抓到一个包之后就调用 packet_handler 函数来处理
pcap_loop(adhandle, 0, packet_handler, NULL);

//每一个传入的数据包调用 packet_handler 函数进行接收
void packet_handler{
    //接收数据（具体看源代码）
    //按照协议头的格式输出各个层的字段
}
```

6 实验总结

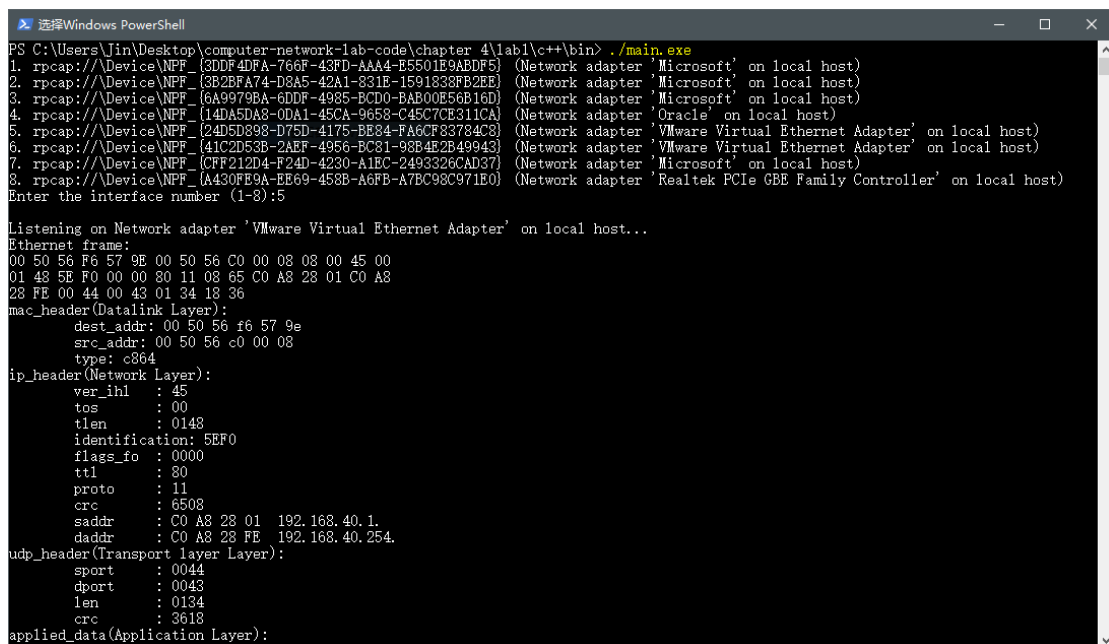
这通过这次试验，培养了自己动手的能力，另外，通过对抓包过程的实现以及对以太网帧格式的分析，加深了对以太网的了解。

通过对报文的分析，对地址解析协议有了进一步的了解。并且把课本上多学的理论知识与实践结合起来，对以前的知识得到深化和巩固，为以后学习新的知识打下基础，也提高了学习的兴趣，收获很大。

并且通过这个实验使我们更加了解了数据链路层数据的传输过程，加深了理解。

附录

A 效果截图



```
PS C:\Users\Jin\Desktop\computer-network-lab-code\chapter 4\lab1\c++\bin> ./main.exe
1. rccap://Device\NPF_{3DDF4DFA-766F-43FD-AAA4-E5501E0ABDF5} (Network adapter 'Microsoft' on local host)
2. rccap://Device\NPF_{3B2BFA74-D8A5-42A1-831E-1501832FB2EE} (Network adapter 'Microsoft' on local host)
3. rccap://Device\NPF_{6A9979BA-6DDF-4985-BCD0-EAB00E56B16D} (Network adapter 'Microsoft' on local host)
4. rccap://Device\NPF_{14DA5DA3-0DA1-45CA-9658-C45C7CE311CA} (Network adapter 'Oracle' on local host)
5. rccap://Device\NPF_{24D5D898-D75D-4175-BE84-FA6CF83784C3} (Network adapter 'VMware Virtual Ethernet Adapter' on local host)
6. rccap://Device\NPF_{41C2D53B-2AEF-4956-EC81-9384E2B49943} (Network adapter 'VMware Virtual Ethernet Adapter' on local host)
7. rccap://Device\NPF_{CFF212D4-F24D-4230-A1EC-2493326CAD37} (Network adapter 'Microsoft' on local host)
8. rccap://Device\NPF_{A430FE9A-EE69-458B-A6FB-A7BC98C971E0} (Network adapter 'Realtek PCIe GBE Family Controller' on local host)
Enter the interface number (1-8):5
Listening on Network adapter 'VMware Virtual Ethernet Adapter' on local host...
Ethernet frame:
00 50 56 F6 57 9E 00 50 56 C0 00 08 00 45 00
01 48 5E F0 00 00 80 11 08 65 C0 A8 28 01 C0 A8
28 FE 00 44 00 43 01 34 18 36
mac_header(DataLink Layer):
    dest_addr: 00 50 56 f6 57 9e
    src_addr: 00 50 56 c0 00 08
    type: c864
ip_header(Network Layer):
    ver_ihl : 45
    tos : 00
    tlen : 0148
    identification: 5EF0
    flags_fo : 0000
    ttl : 80
    proto : 11
    crc : 6508
    saddr : C0 A8 28 01 192.168.40.1.
    daddr : C0 A8 28 FE 192.168.40.254.
udp_header(Transport Layer Layer):
    sport : 0044
    dport : 0043
    len : 0134
    crc : 3618
applied_data(Application Layer):
```

图 1 C++运行截图

```
E:\A-PPTandLab\计算机网络>java -jar PacketCapture.jar
■ 开始时间: Tue May 14 21:39:17 CST 2019
=====网卡信息=====
网卡1: \Device\NPF_{1DD06851-19C9-42FD-9B99-09FFC8FFE7FB} (Oracle)
数据链路层信息: EN10MB(Ethernet)
网络接口地址: /fe80:0:0:0:8ef:880d:c396:9738 子网掩码: null 广播地址: null
网络接口地址: /192.168.56.1 子网掩码: /255.255.255.0 广播地址: /255.255.255.255
MAC地址: a:0:27:0:0:5
=====
网卡2: \Device\NPF_{42D7E0B1-76CB-4484-B81A-8E1F391004F9} (Microsoft)
数据链路层信息: EN10MB(Ethernet)
网络接口地址: /fe80:0:0:0:903:1310:589:6cb7 子网掩码: null 广播地址: null
网络接口地址: /2001:da8:204:2512:0:0:3:b716 子网掩码: null 广播地址: null
网络接口地址: /10.63.186.86 子网掩码: /255.255.128.0 广播地址: /0.0.0.0
MAC地址: 68:7:15:4d:2a:97
=====
网卡3: \Device\NPF_{43AF5CB3-875E-4775-938E-921CEF12FDFB} (VMware Virtual Ethernet Adapter)
数据链路层信息: EN10MB(Ethernet)
网络接口地址: /fe80:0:0:0:f002:4e6d:be71:cfac 子网掩码: null 广播地址: null
网络接口地址: /192.168.192.1 子网掩码: /255.255.255.0 广播地址: /255.255.255.255
MAC地址: 0:50:56:c0:0:8
=====
网卡4: \Device\NPF_{ABF96C24-7E1C-431E-818C-68EEEFB9D1C3} (VMware Virtual Ethernet Adapter)
数据链路层信息: EN10MB(Ethernet)
网络接口地址: /fe80:0:0:0:a1e5:6ab0:3a87:5c9b 子网掩码: null 广播地址: null
网络接口地址: /192.168.190.1 子网掩码: /255.255.255.0 广播地址: /255.255.255.255
MAC地址: 0:50:56:c0:0:1
=====
网卡5: \Device\NPF_{847ACB1B-DF5F-4AAA-8C11-C9E0C2589F81} (Sangfor SSL VPN CS Support System VNIC)
数据链路层信息: EN10MB(Ethernet)
网络接口地址: /fe80:0:0:0:b136:f2df:27f:620e 子网掩码: null 广播地址: null
网络接口地址: /fe80:0:0:0:b136:f2df:27f:620e 子网掩码: null 广播地址: null
MAC地址: 0:ff:84:7a:cb:1b
=====
网卡6: \Device\NPF_{3026FD6B-AF45-47D4-8B7F-9DED5763947C} (Microsoft)
数据链路层信息: EN10MB(Ethernet)
网络接口地址: /fe80:0:0:0:f8a6:19f7:b874:b92a 子网掩码: null 广播地址: null
网络接口地址: /fe80:0:0:0:f8a6:19f7:b874:b92a 子网掩码: null 广播地址: null
MAC地址: 6a:7:15:4d:2a:97
=====
网卡7: \Device\NPF_{028157CF-A6E5-4FC9-B558-BB835161AD45} (TAP-Windows Adapter V9)
数据链路层信息: EN10MB(Ethernet)
网络接口地址: /fe80:0:0:0:98d3:c85c:6086:f52e 子网掩码: null 广播地址: null
网络接口地址: /fe80:0:0:0:98d3:c85c:6086:f52e 子网掩码: null 广播地址: null
MAC地址: 0:ff:2:81:57:cf
=====
网卡8: \Device\NPF_{67654DE1-3437-4E77-A060-1E4690C838DA} (Realtek PCIe GBE Family Controller)
数据链路层信息: EN10MB(Ethernet)
```

图 2 java 运行截图


```

waiting for windows to listen to WLAN
<Sniffed: TCP:92 UDP:8 ICMP:0 Other:0>
###[ Ethernet ]###
  dst      = 10:51:72:22:1f:7d
  src      = 28:e3:47:89:01:44
  type     = 0x800
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 77
  id       = 11700
  flags    = 0
  frag     = 0
  ttl      = 128
  proto    = udp
  checksum = 0xc2db
  src      = 10.62.133.68
  dst      = 183.202.2.196
  \options \
###[ UDP ]###
  sport    = 19145
  dport    = 1879
  len      = 57
  checksum = 0x1279
###[ Raw ]###
  load     = '6\x80cuu2\xc1\x11m\xe5#R544JH\x82\xc8\xcb\xcb\xcb\xcb\xcd\xcb\xcb\xcb\xcb\xcd9\xdb\x8c+++++#+)6666666a\x07'

```

图 5 python 运行截图