一、实验目的：

1）以用户访问一个 Web 网站首页为例，深入理解 Web 服务系统的工作原理，从计算机网络体系结构的角度出发，分别从应用层、传输层、网络层以及数据链路层分析网络为 Web 服务提供的技术支持以及工作原理；

2）通过分析 Wireshark 抓包工具所抓的数据报，分析 DNS、WEB 协议的工作原理；

3）分析 TCP 协议通过三次握手建立连接的时序关系，以及通过四次挥手释放 TCP 连接的时序关系；

4）分析 ARP 协议工作原理；

5）分析数据链路层工业以太网工作原理（数据帧的语法）；

二、过程要求

实验内容 1：网络协议分析与验证

1）对 DNS 请求报文、HTTP 请求报文从应用层到数据链路层不同协议单元首部各个字段进行解释说明；

2）对 TCP 连接请求报文从传输层到数据链路层不同协议单元首部各个字段进行解释说明；

3）对 ARP 请求报文和应答报文从网络层到数据链路层不同协议单元首部各个字段进行解释说明；

4）通过对捕获的数据包进行分析，提供证据，说明如何获得以下信息：

a. www.baidu.com 对应的 IP 地址，有几种方法可以获得该 IP 地址；

b. 网关 IP 地址和 MAC 地址分别是多少，有几种方法可以获得；

c. 发送方和接收方 TCP 协议协商的初始序号？真真发送数据的实际起始序号是多少？为什么？

d. HTTP 协议协商的版本号是多少？该版本号 HTTP 协议工作特点是什么？提供证据说明。

e. 一个 TCP 连接从建立到释放，总共发送和接收了多少字节数据？为什么？

f. 针对一个 TCP 连接，提供该连接建立三次握手报文段和四次挥手报文段，为什么说该证据是针对一个 TCP 连接？

5）以 HTTP 请求报文为例子，当 WEB 服务器接收到该报文后，接收方从数据链路层到应用层如何知道不同层数据字段的长度，开始和起始位置。

6）从应用层到数据链路层有哪些校验字段，分别采用什么方法计算校验码，其校验范围分别是什么，不同层重复的校验是多余的吗？为什么？

7）如果在本次实验过程，对抓取的报文进行分析，发现 DNS 和 ARP 协议没有工作，为什么？如何解决该问题？在解决该问题过程中用到两个网络命令,分别是什么,写出这两个命令具体应用。

8）如果在本次实验过程，用户在客户端 DOS>ping www.baidu.com,连续发送了三次 ICMP ECHO 请求报文，但显示第一次接收 ICMP ECHO 应答报文超时，说明网络不同；但后面两次 ICMP ECHO 应答报文接收正常，又说明网络是连通的，为什么？

实验内容 2：网络广播报文发送编程

1）从抓取的报文中过滤出源 IP = 发送方 IP 地址的某一个报文（可以手动或者采用过滤器）；

2）分析广播报文传输层采用 UDP/TCP，理解广播或者组播为什么不是 TCP？

3）抓取发送的广播报文，找出通信的五元组信息和数据帧首部信息，分析目的 IP 地址、源 IP 地址、协议类型、目的 MAC 地址、源 MAC 地址等与单播 UDP 用户数据报的不同。

三、相关知识

Wireshark 是一款功能强大的网络分析调试和数据包协议分析软件，相对于同类型的产品来说

具有优秀的 GUI 和众多分类信息及过滤选项，用户通过 Wireshark 将网卡设置为混杂模式，可以查看网络中发送的所有通信流量。因此采用 Wireshark 捕获网卡上传输数据的信息非常高效，下面介绍 Wireshark 抓包工具的使用与主要方法

1）Wireshark 主界面

主界面由 7 个部分组成：

①菜单栏：实现各种命令操作。

②工具栏：提供快速访问菜单中常用命令功能。

③过滤工具栏：提供用户输入过滤数据包规则界面。

④分组显示面板：具体显示分组内容。

⑤分组字节面板：显示面板中选中的某个数据包二进制内容。

⑥状态栏：显示当前工具状态与捕获数据详情。

2）Packet Details Pane(数据包详细信息)，在数据包列表中选择指定数据包，在数据包详细信息中会显示数据包的所有详细信息内容。数据包详细信息面板是最重要的，用来查看协议中的每一个字段。各行信息分别为：

①Frame:物理层的数据帧概况。

②Ethernet II: 数据链路层以太网帧头部信息。

③Internet Protocol Version 6: 互联网层 IP 包头部信息。

④Transmission Control Protocol: 传输层 T 的数据段头部信息。

3）过滤器设置

①抓包过滤器

抓包过滤器的菜单栏路径为 Capture --> Capture Filters。用于在抓取数据包前设置。用户可以根据需求设置不同的过滤条件即可。

②显示过滤器

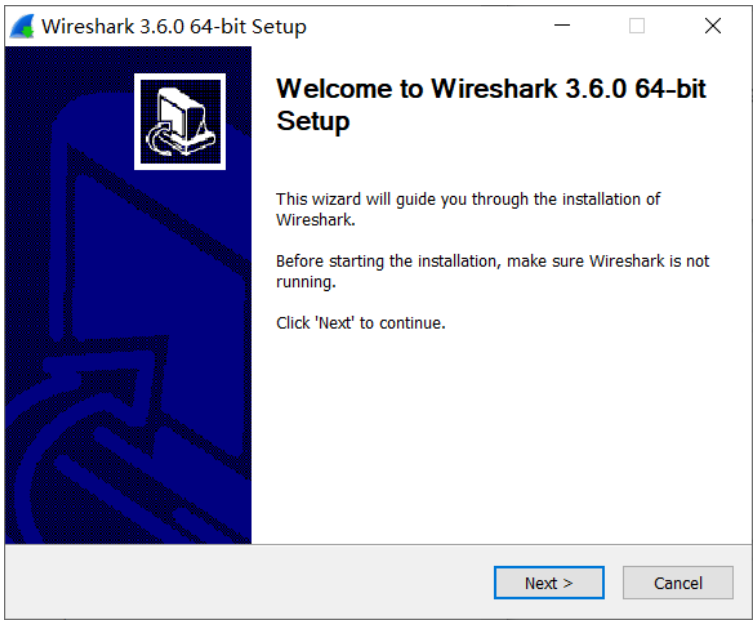
显示过滤器是用于在抓取数据包后设置过滤条件进行过滤数据包。通常是在抓取数据包时设置条件相对宽泛，抓取的数据包内容较多时使用显示过滤器设置条件顾虑以方便分析。同样上述场景，在捕获时未设置捕获规则直接通过网卡进行抓取所有数据包。

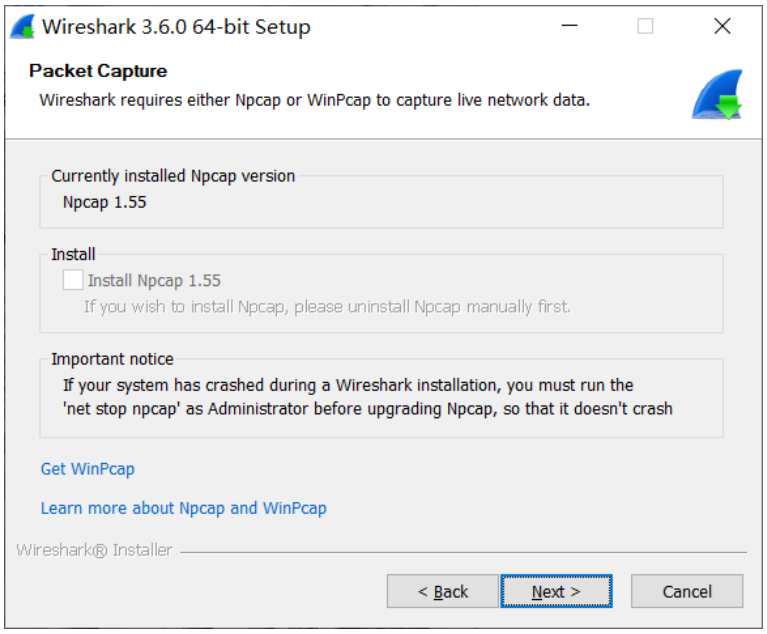
四、实现原理或流程

实验内容 1：网络协议分析与验证

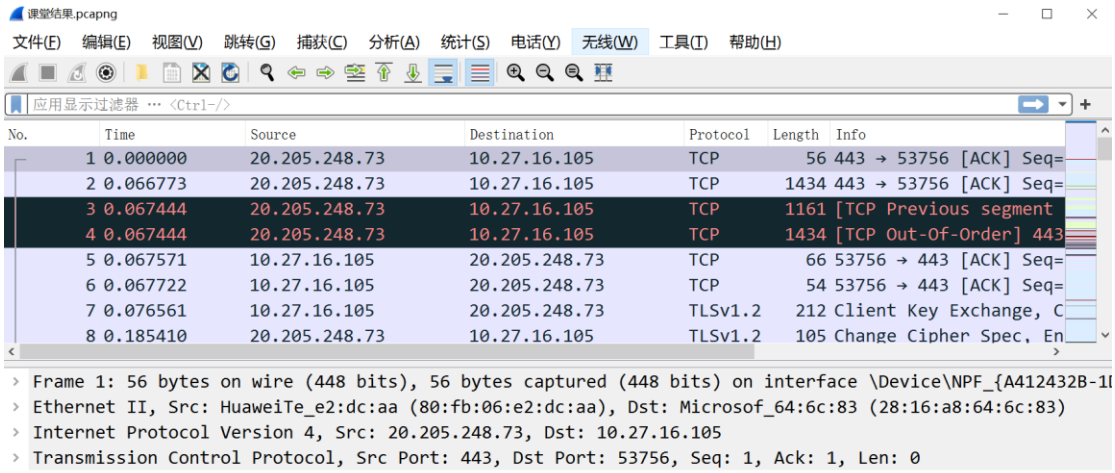
1）安装 WINPCAP 组件；

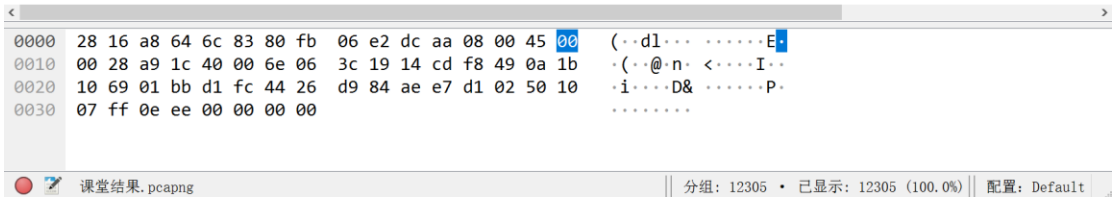
2）安装 wireshark 抓包工具及 Npcap 组件；





3）启动 wireshark 抓包工具，并激活的网络接口上开始抓包；





4）用户在浏览器地址栏输入：www.baidu.com 回车，直到百度首先在浏 览器上显示为止；

5）抓包结束，开始对报文分析，对实验要求内容找到报文证据。

实验内容 2：网络广播报文发送编程

1）编写程序，发送三层广播报文；

2）在另一台利用抓包工具，抓取广播报文，并对报文首部进行分析；

五、程序代码

```
#include <iostream>
#include "winsock.h"
#include "windows.h"
#include <stdio.h>
#pragma comment(lib,"wsock32.lib")
#define RECV_PORT 2222
#define SEND_PORT 3306
BOOL optReturn=TRUE;
SOCKET sock;
struct sockaddr_in sockAddrFrom,sockAddrTo,sockAddrIn;
```

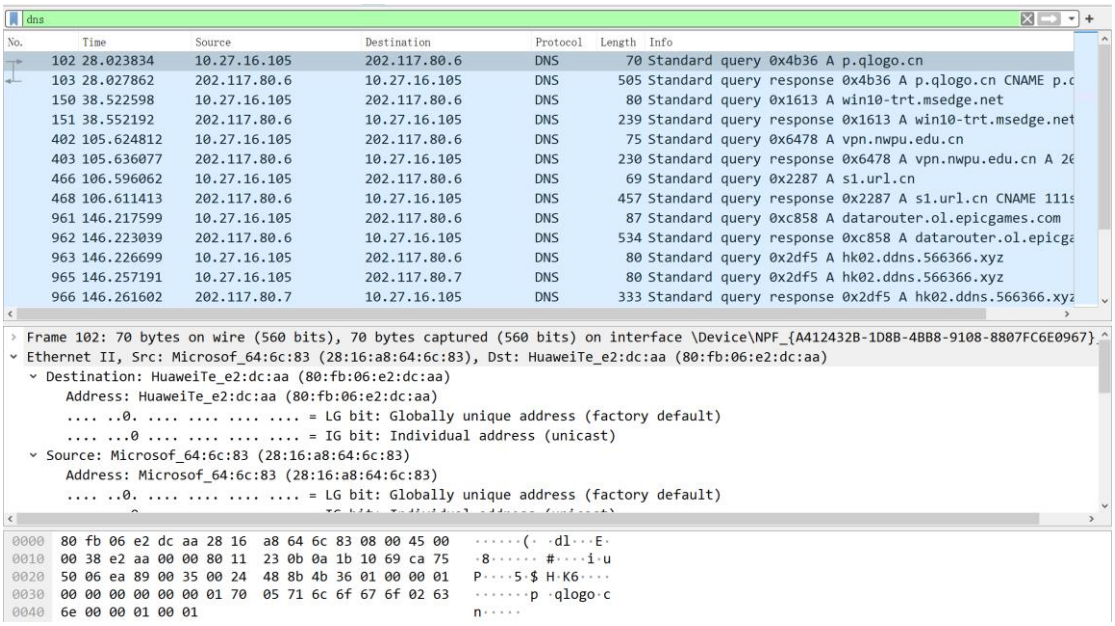
```
DWORD CreateSocket()
{
    WSADATA WSAData;
    if(WSAStartup(MAKEWORD(2,2),&WSAData)!=0)
    {
        printf("socket lib load error!");
        return false;
    }
    if((sock=socket(AF_INET,SOCK_DGRAM,0))==INVALID_SOCKET)
    {
        printf("create socket failed!\n");
        WSACleanup();
        return false;
    }
    sockAddrIn.sin_family=AF_INET;
    sockAddrIn.sin_addr.s_addr=INADDR_ANY;
    sockAddrIn.sin_port=htons(SEND_PORT);
    if(bind(sock,(LPSOCKADDR)&sockAddrIn,sizeof(sockAddrIn)))
    {
        closesocket(sock);
        WSACleanup();
        return false;
    }
    if(setsockopt(sock,SOL_SOCKET,SO_BROADCAST,(char
*)&optReturn,sizeof(optReturn)==SOCKET_ERROR))
    {
        closesocket(sock);
        WSACleanup();
        return false;
    }
    return true;
}

DWORD BroadDataSend(char lpBuffer[])
{
    int lengthSend=0;
    sockAddrTo.sin_family=AF_INET;
```

```
    sockAddrTo.sin_addr.s_addr=INADDR_BROADCAST;
    sockAddrTo.sin_port=htons(RECV_PORT);
    if((lengthSend =
sendto(sock,lpBuffer,strlen(lpBuffer),MSG_DONTROUTE,(struct
sockaddr*)&sockAddrTo,
                            sizeof(sockaddr)))==SOCKET_ERROR)
    {
        closesocket(sock);
        WSACleanup();
        return FALSE;
    }
    return true;
}
int main()
{
    char buffer[100];
    int i;
    CreateSocket();
    printf("press any key to continue!\n");
    getchar();
    for(i=0;i<100;i++)
    {
        sprintf(buffer,"data %d",i);
        BroadDataSend(buffer);
        Sleep(50);
    }
    getchar();
    return true;
}
```

六、运行结果与分析

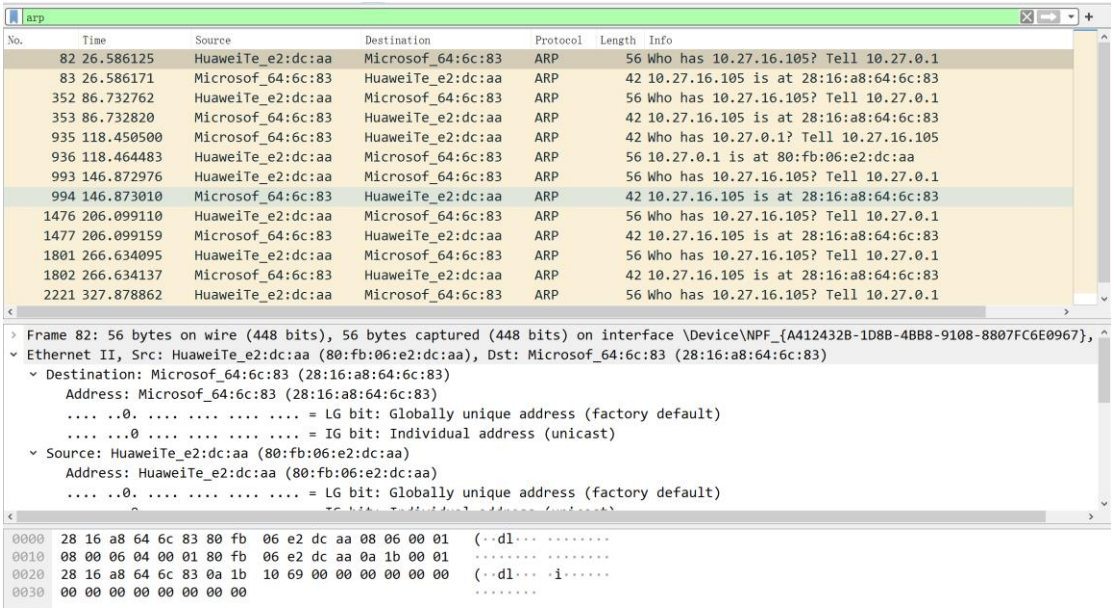
实验内容 1：

①抓取的 DNS 报文以及对应的 DNS 应答报文：



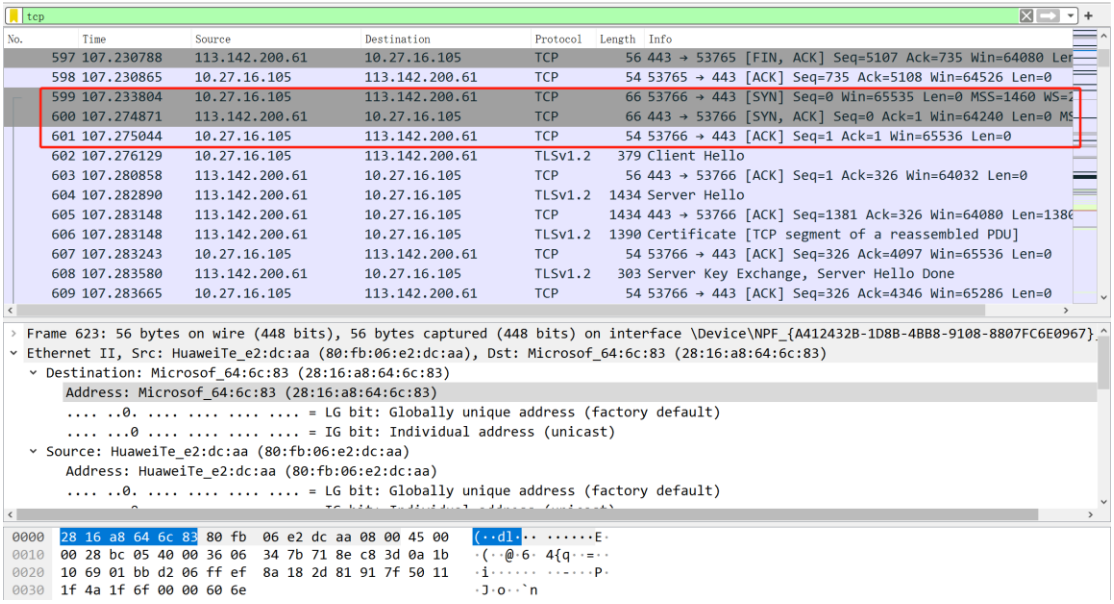
可以看到，第 102 行是 10.27.16.105（本机）对 202.117.80.6 的 DNS 请求报文，第 103 行是 202.117.80.6 对 10.27.16.105 的应答 DNS 报文，抓取成功。

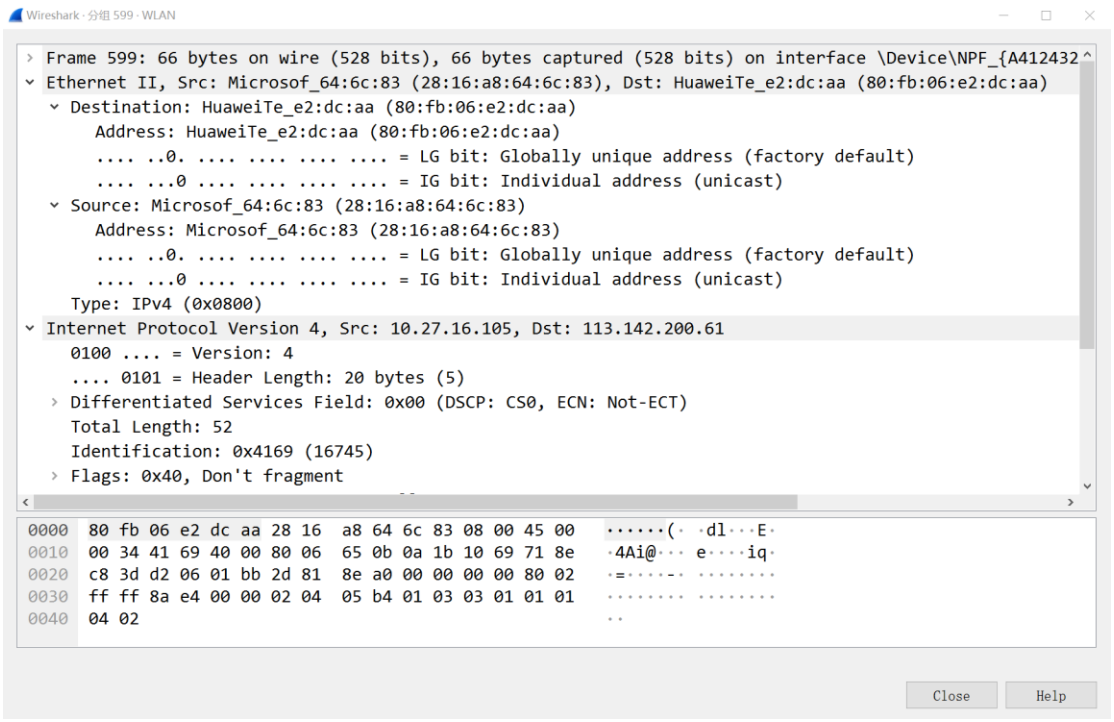
②抓取到 ARP 请求报文和对应的 ARP 应答报文：

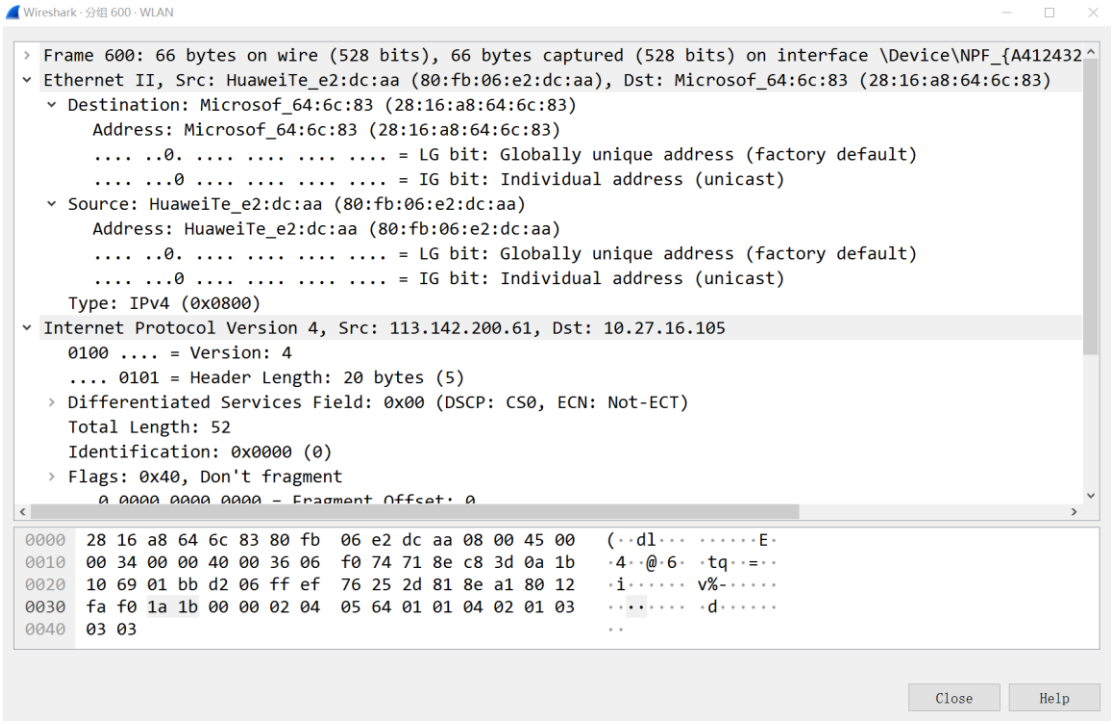


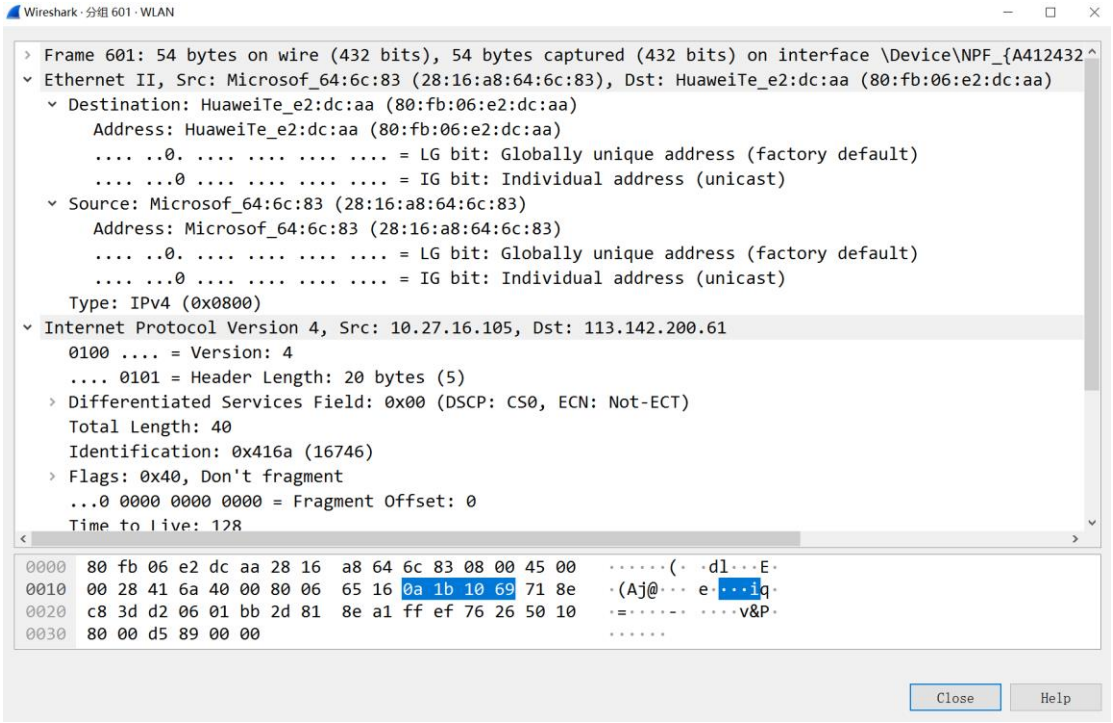
可以看到，第一条信息 10.27.0.1 发送的 ARP 请求报文，第二条信息是 ARP 的应答报文，即由 Microsof_64:6c:83 回答给 10.27.0.1 的 ARP 应答报文。

③抓取到一个 TCP 连接建立三次握手的报文段：


tcp
No. Time Source Destination Protocol Length Info
597 107.230788 113.142.200.61 10.27.16.105 TCP 56 443 → 53765 [FIN, ACK] Seq=5107 Ack=735 Win=64080 Len
598 107.230865 10.27.16.105 113.142.200.61 TCP 54 53765 → 443 [ACK] Seq=735 Ack=5108 Win=64526 Len=0
599 107.233804 10.27.16.105 113.142.200.61 TCP 66 53766 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=
600 107.274871 113.142.200.61 10.27.16.105 TCP 66 443 → 53766 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MS
601 107.275044 10.27.16.105 113.142.200.61 TCP 54 53766 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0
602 107.276129 10.27.16.105 113.142.200.61 TLSv1.2 379 Client Hello
603 107.280858 113.142.200.61 10.27.16.105 TCP 56 443 → 53766 [ACK] Seq=1 Ack=326 Win=64032 Len=0
604 107.282890 113.142.200.61 10.27.16.105 TLSv1.2 1434 Server Hello
605 107.283148 113.142.200.61 10.27.16.105 TCP 1434 443 → 53766 [ACK] Seq=1381 Ack=326 Win=64080 Len=138
606 107.283148 113.142.200.61 10.27.16.105 TLSv1.2 1390 Certificate [TCP segment of a reassembled PDU]
607 107.283243 10.27.16.105 113.142.200.61 TCP 54 53766 → 443 [ACK] Seq=326 Ack=4097 Win=65536 Len=0
608 107.283580 113.142.200.61 10.27.16.105 TLSv1.2 303 Server Key Exchange, Server Hello Done
609 107.283665 10.27.16.105 113.142.200.61 TCP 54 53766 → 443 [ACK] Seq=326 Ack=4346 Win=65286 Len=0
Frame 623: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF_{A412432B-1D8B-4BB8-9108-8807FC6E0967}
Ethernet II, Src: HuaweiTe_e2:dc:aa (80:fb:06:e2:dc:aa), Dst: Microsof_64:6c:83 (28:16:a8:64:6c:83)
Destination: Microsof_64:6c:83 (28:16:a8:64:6c:83)
Address: Microsof_64:6c:83 (28:16:a8:64:6c:83)
.... ..0. .... = LG bit: Globally unique address (factory default)
....0 .... = IG bit: Individual address (unicast)
Source: HuaweiTe_e2:dc:aa (80:fb:06:e2:dc:aa)
Address: HuaweiTe_e2:dc:aa (80:fb:06:e2:dc:aa)
.... ..0. .... = LG bit: Globally unique address (factory default)
0000 28 16 a8 64 6c 83 80 fb 06 e2 dc aa 08 00 45 00
0010 00 28 bc 05 40 00 36 06 34 7b 71 8e c8 3d 0a 1b
0020 10 69 01 bb d2 06 ff ef 8a 18 2d 81 91 7f 50 11
0030 1f 4a 1f 6f 00 00 60 6e

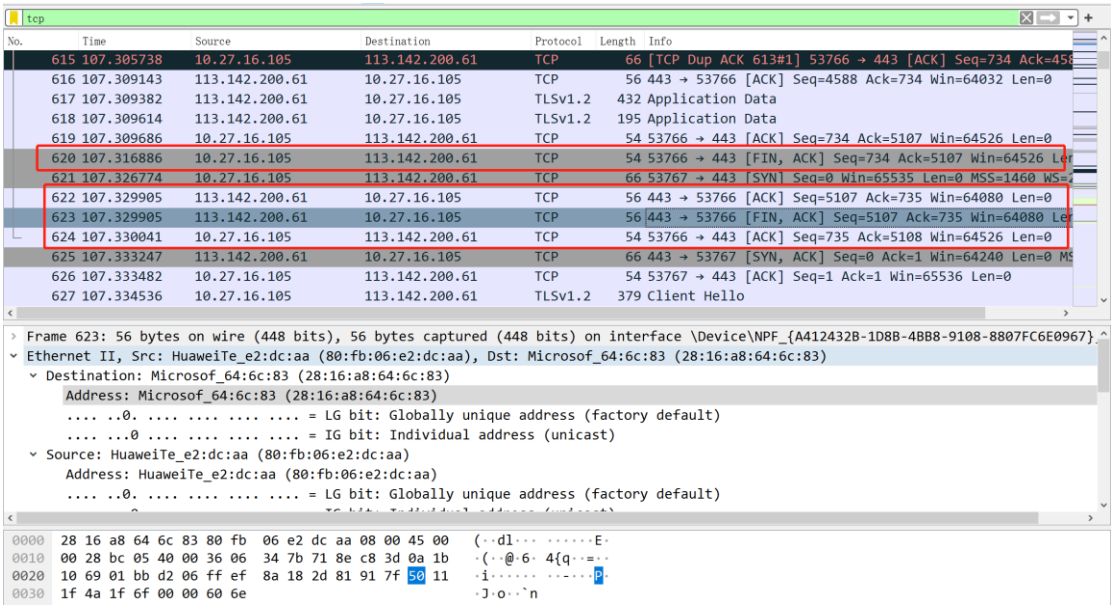

Wireshark · 分组 599 · WLAN
Frame 599: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{A412432
Ethernet II, Src: Microsof_64:6c:83 (28:16:a8:64:6c:83), Dst: HuaweiTe_e2:dc:aa (80:fb:06:e2:dc:aa)
Destination: HuaweiTe_e2:dc:aa (80:fb:06:e2:dc:aa)
Address: HuaweiTe_e2:dc:aa (80:fb:06:e2:dc:aa)
.... ..0. .... = LG bit: Globally unique address (factory default)
....0 .... = IG bit: Individual address (unicast)
Source: Microsof_64:6c:83 (28:16:a8:64:6c:83)
Address: Microsof_64:6c:83 (28:16:a8:64:6c:83)
.... ..0. .... = LG bit: Globally unique address (factory default)
....0 .... = IG bit: Individual address (unicast)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 10.27.16.105, Dst: 113.142.200.61
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 52
Identification: 0x4169 (16745)
Flags: 0x40, Don't fragment
0000 80 fb 06 e2 dc aa 28 16 a8 64 6c 83 08 00 45 00
0010 00 34 41 69 40 00 80 06 65 0b 0a 1b 10 69 71 8e
0020 c8 3d d2 06 01 bb 2d 81 8e a0 00 00 00 00 80 02
0030 ff ff 8a e4 00 00 02 04 05 b4 01 03 03 01 01 01
0040 04 02
Close
Help

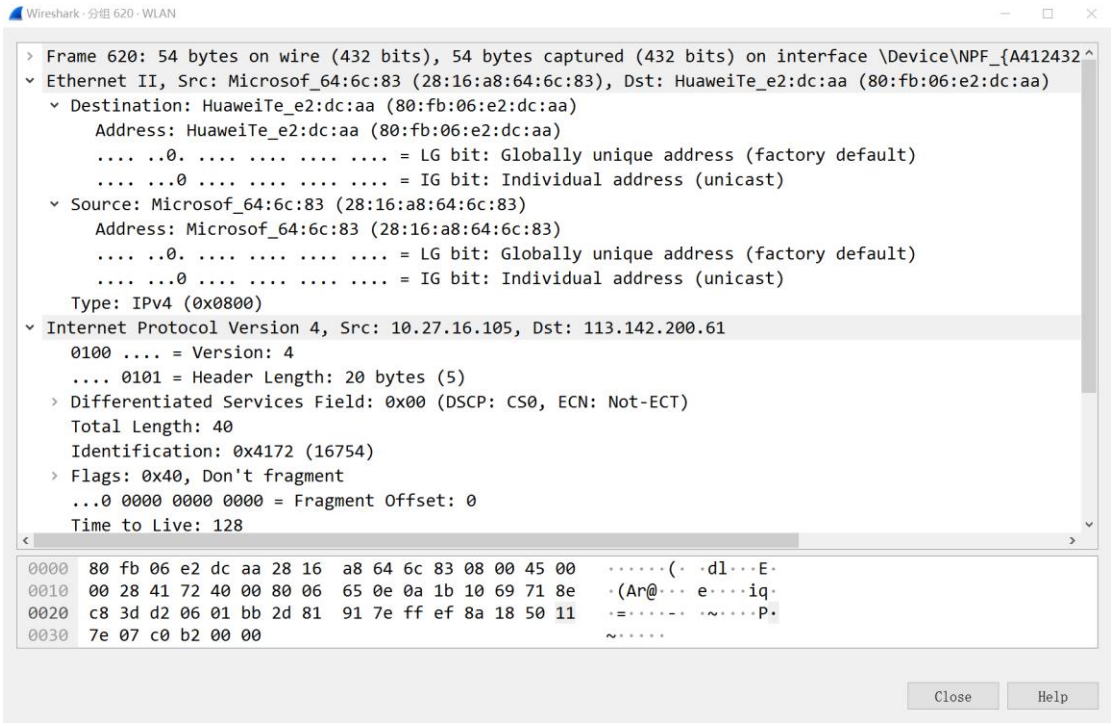


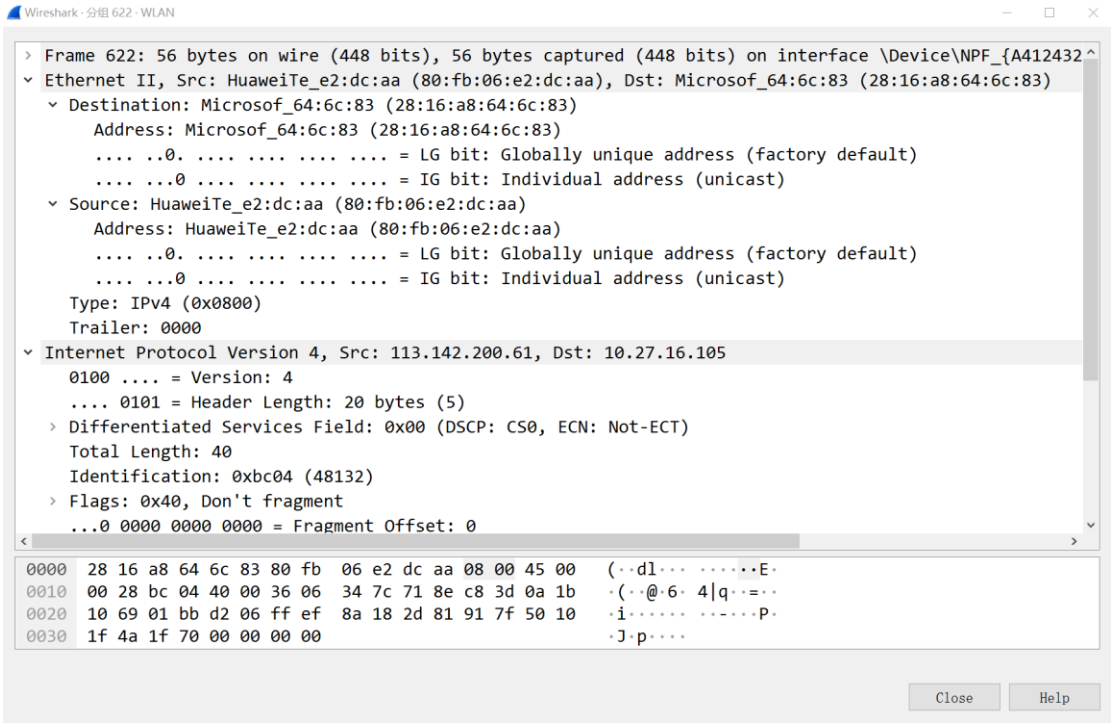


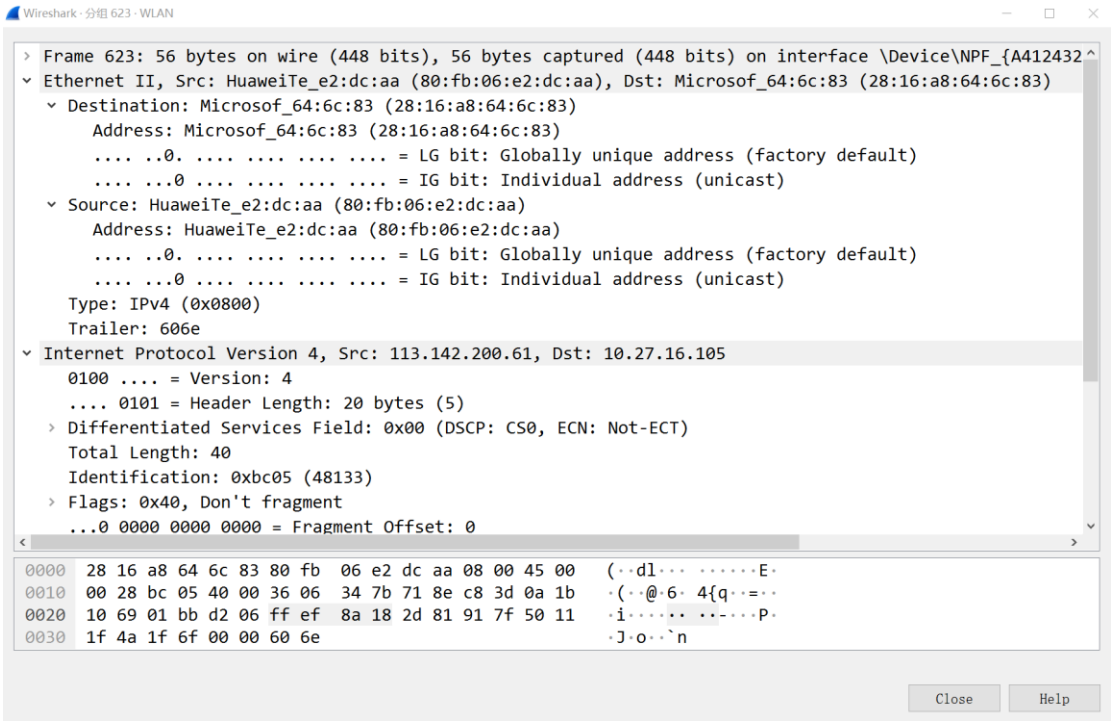
可以看到该图的第 599、600、601 三条消息，就是 TCP 连接建立三次握手的过程。

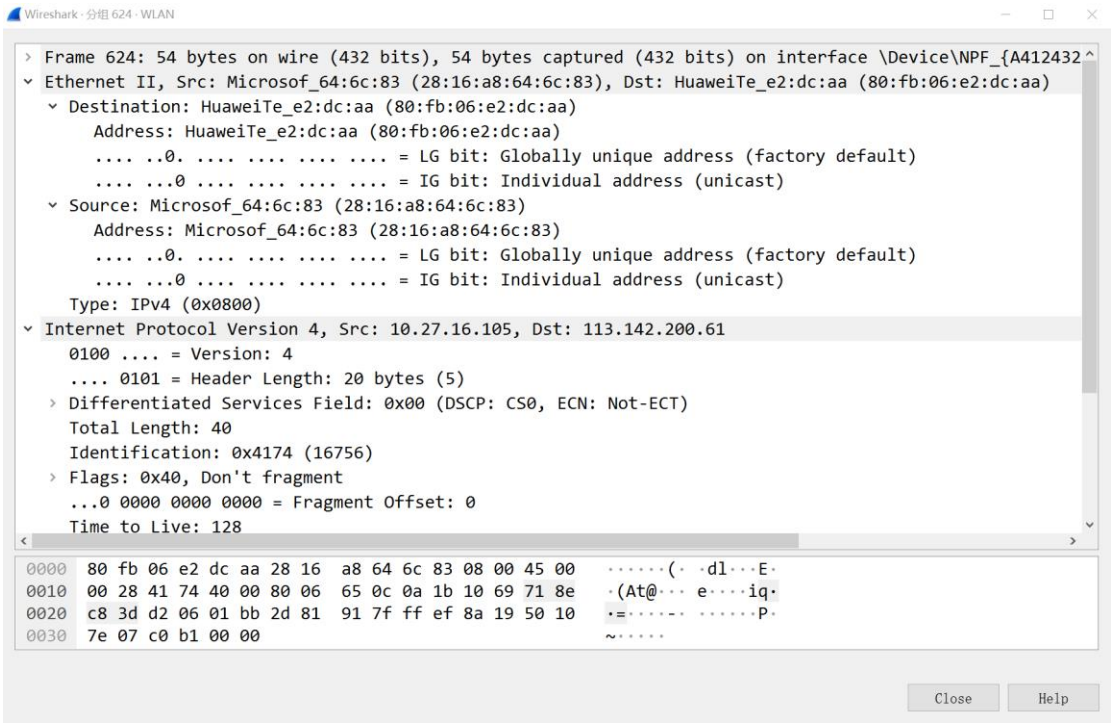
④抓取到③中对应的 TCP 连接四次握手释放的报文：


tcp
No. Time Source Destination Protocol Length Info
615 107.305738 10.27.16.105 113.142.200.61 TCP 66 [TCP Dup ACK 613#1] 53766 → 443 [ACK] Seq=734 Ack=458
616 107.309143 113.142.200.61 10.27.16.105 TCP 56 443 → 53766 [ACK] Seq=4588 Ack=734 Win=64032 Len=0
617 107.309382 113.142.200.61 10.27.16.105 TLSv1.2 432 Application Data
618 107.309614 113.142.200.61 10.27.16.105 TLSv1.2 195 Application Data
619 107.309686 10.27.16.105 113.142.200.61 TCP 54 53766 → 443 [ACK] Seq=734 Ack=5107 Win=64526 Len=0
620 107.316886 10.27.16.105 113.142.200.61 TCP 54 53766 → 443 [FIN, ACK] Seq=734 Ack=5107 Win=64526 Len
621 107.326774 10.27.16.105 113.142.200.61 TCP 66 53767 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=
622 107.329905 113.142.200.61 10.27.16.105 TCP 56 443 → 53766 [ACK] Seq=5107 Ack=735 Win=64080 Len=0
623 107.329905 113.142.200.61 10.27.16.105 TCP 56 443 → 53766 [FIN, ACK] Seq=5107 Ack=735 Win=64080 Len
624 107.330041 10.27.16.105 113.142.200.61 TCP 54 53766 → 443 [ACK] Seq=735 Ack=5108 Win=64526 Len=0
625 107.333247 113.142.200.61 10.27.16.105 TCP 66 443 → 53767 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MS
626 107.333482 10.27.16.105 113.142.200.61 TCP 54 53767 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0
627 107.334536 10.27.16.105 113.142.200.61 TLSv1.2 379 Client Hello
Frame 623: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF_{A412432B-1D8B-4BB8-9108-8807FC6E0967}
Ethernet II, Src: HuaweiTe_e2:dc:aa (80:fb:06:e2:dc:aa), Dst: Microsof_64:6c:83 (28:16:a8:64:6c:83)
Destination: Microsof_64:6c:83 (28:16:a8:64:6c:83)
Address: Microsof_64:6c:83 (28:16:a8:64:6c:83)
.... ..0. .... = LG bit: Globally unique address (factory default)
....0 .... = IG bit: Individual address (unicast)
Source: HuaweiTe_e2:dc:aa (80:fb:06:e2:dc:aa)
Address: HuaweiTe_e2:dc:aa (80:fb:06:e2:dc:aa)
.... ..0. .... = LG bit: Globally unique address (factory default)
0000 28 16 a8 64 6c 83 80 fb 06 e2 dc aa 08 00 45 00
0010 00 28 bc 05 40 00 36 06 34 7b 71 8e c8 3d 0a 1b
0020 10 69 01 bb d2 06 ff ef 8a 18 2d 81 91 7f 50 11
0030 1f 4a 1f 6f 00 00 60 6e


Wireshark · 分组 620 · WLAN
Frame 620: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{A412432
Ethernet II, Src: Microsof_64:6c:83 (28:16:a8:64:6c:83), Dst: HuaweiTe_e2:dc:aa (80:fb:06:e2:dc:aa)
Destination: HuaweiTe_e2:dc:aa (80:fb:06:e2:dc:aa)
Address: HuaweiTe_e2:dc:aa (80:fb:06:e2:dc:aa)
.... ..0. .... = LG bit: Globally unique address (factory default)
....0 .... = IG bit: Individual address (unicast)
Source: Microsof_64:6c:83 (28:16:a8:64:6c:83)
Address: Microsof_64:6c:83 (28:16:a8:64:6c:83)
.... ..0. .... = LG bit: Globally unique address (factory default)
....0 .... = IG bit: Individual address (unicast)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 10.27.16.105, Dst: 113.142.200.61
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 40
Identification: 0x4172 (16754)
Flags: 0x40, Don't fragment
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 128
0000 80 fb 06 e2 dc aa 28 16 a8 64 6c 83 08 00 45 00
0010 00 28 41 72 40 00 80 06 65 0e 0a 1b 10 69 71 8e
0020 c8 3d d2 06 01 bb 2d 81 91 7e ff ef 8a 18 50 11
0030 7e 07 c0 b2 00 00
Close
Help


Wireshark · 分组 622 · WLAN
Frame 622: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF_{A412432
Ethernet II, Src: HuaweiTe_e2:dc:aa (80:fb:06:e2:dc:aa), Dst: Microsof_64:6c:83 (28:16:a8:64:6c:83)
Destination: Microsof_64:6c:83 (28:16:a8:64:6c:83)
Address: Microsof_64:6c:83 (28:16:a8:64:6c:83)
.... ..0. .... = LG bit: Globally unique address (factory default)
....0 .... = IG bit: Individual address (unicast)
Source: HuaweiTe_e2:dc:aa (80:fb:06:e2:dc:aa)
Address: HuaweiTe_e2:dc:aa (80:fb:06:e2:dc:aa)
.... ..0. .... = LG bit: Globally unique address (factory default)
....0 .... = IG bit: Individual address (unicast)
Type: IPv4 (0x0800)
Trailer: 0000
Internet Protocol Version 4, Src: 113.142.200.61, Dst: 10.27.16.105
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 40
Identification: 0xbc04 (48132)
Flags: 0x40, Don't fragment
...0 0000 0000 0000 = Fragment Offset: 0
0000 28 16 a8 64 6c 83 80 fb 06 e2 dc aa 08 00 45 00
0010 00 28 bc 04 40 00 36 06 34 7c 71 8e c8 3d 0a 1b
0020 10 69 01 bb d2 06 ff ef 8a 18 2d 81 91 7f 50 10
0030 1f 4a 1f 70 00 00 00 00
Close
Help


Wireshark · 分组 623 · WLAN
Frame 623: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF_{A412432
Ethernet II, Src: HuaweiTe_e2:dc:aa (80:fb:06:e2:dc:aa), Dst: Microsof_64:6c:83 (28:16:a8:64:6c:83)
Destination: Microsof_64:6c:83 (28:16:a8:64:6c:83)
Address: Microsof_64:6c:83 (28:16:a8:64:6c:83)
.... ..0. .... = LG bit: Globally unique address (factory default)
....0 .... = IG bit: Individual address (unicast)
Source: HuaweiTe_e2:dc:aa (80:fb:06:e2:dc:aa)
Address: HuaweiTe_e2:dc:aa (80:fb:06:e2:dc:aa)
.... ..0. .... = LG bit: Globally unique address (factory default)
....0 .... = IG bit: Individual address (unicast)
Type: IPv4 (0x0800)
Trailer: 606e
Internet Protocol Version 4, Src: 113.142.200.61, Dst: 10.27.16.105
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 40
Identification: 0xbc05 (48133)
Flags: 0x40, Don't fragment
...0 0000 0000 0000 = Fragment Offset: 0
0000 28 16 a8 64 6c 83 80 fb 06 e2 dc aa 08 00 45 00
0010 00 28 bc 05 40 00 36 06 34 7b 71 8e c8 3d 0a 1b
0020 10 69 01 bb d2 06 ff ef 8a 18 2d 81 91 7f 50 11
0030 1f 4a 1f 6f 00 00 60 6e
Close
Help



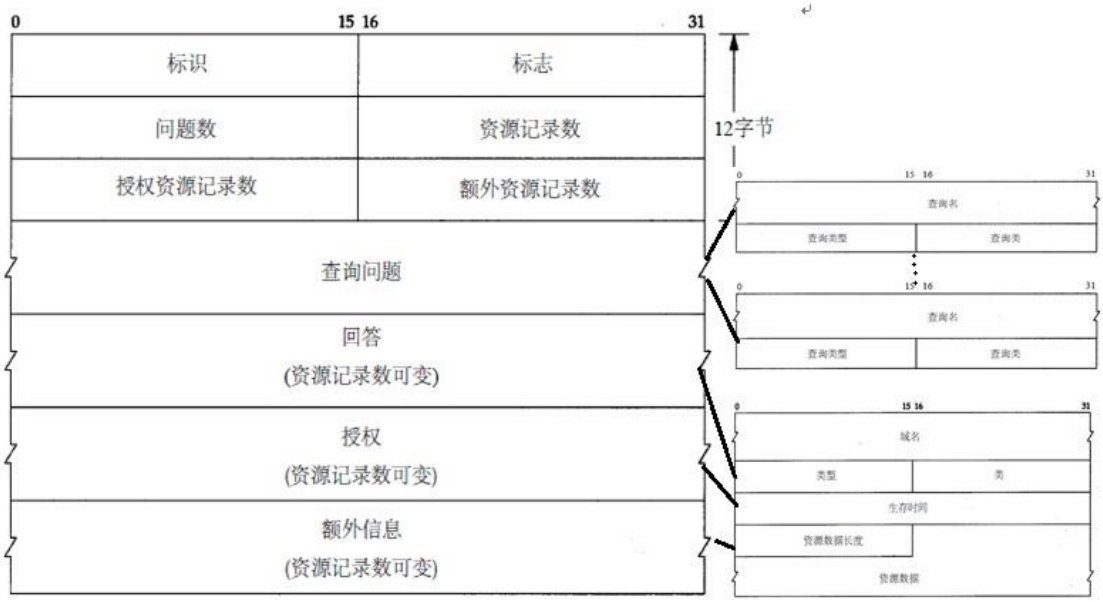
可以看到该图的第 620、622、623、624 就是 TCP 连接四次握手释放过程。

结果分析：

DNS 负责将域名转换成 IP 地址，TCP 负责通信之间的连接建立，，ARP 协议用于将目的 IP 转换为对应的 MAC 地址，ICMP 提供网络传输中的差错检测，UDP 为 DNS 提供 UDP 传输，也就是说 DNS 采用 UDP 而不是 TCP 传输，数据链路层工业以太网协议负责数据链路层数据帧的传输。

①DNS 报文

应用层首部：



该部分中每个字段含义如下。

标识：DNS 报文的 ID 标识。对于请求报文和其对应的应答报文，该字段的值是相同的。通过它可以区分 DNS 应答报文是对哪个请求进行响应的。

标志：DNS 报文中的标志字段。

QR	opcode	AA	TC	RD	RA	(zero)	rcode
1	4	1	1	1	1	3	4

问题数 QDCOUNT：无符号 16 位整数表示报文请求段中的问题记录数。

资源记录数 ANCOUNT：无符号 16 位整数表示报文回答段中的回答记录数。

授权资源记录数 NSCOUNT：无符号 16 位整数表示报文授权段中的授权记录数。

额外资源记录数 ARCOUNT：无符号 16 位整数表示报文附加段中的附加记录数。

传输层首部（UDP 协议）：

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
源端口																目标端口															
长度																校验															

各个字段含义：

源端口和目的端口，端口号理论上可以有 2^16 这么多。因为它的长度是 16 个 bit。端口的详细见下一章节。

Length 占用 2 个字节，标识 UDP 头的长度，包括首部长度和数据长度。可以有 65535 字节那么长。但是一般网络在传送的时候，一次一般传送不了那么长的协议（涉及到 MTU 的问题），就只好对数据分片。

Checksum ：校验和,包含 UDP 头和数据部分。这是一个可选的选项，并不是所有的系统都对 UDP 数据包加以检验和数据(相对 TCP 协议的必须来说)，但是 RFC 中标准要求，发送端应该计算检验和。

网络层首部：

各个字段含义：

版本：IPv4、IPv6

首部长度：十进制数 5~15，32 位（4 字节）为一个单位长度，首部固定部分有 20 个字节，所以至少首部长度至少是 5 个单位的长度（二进制位 0101），而其最大值为 1111（十进制为 15），最大为 15 个 32 位的字长，即 60 字节。

区分服务：一般很少用，用来获得更好的服务

总长度：首部和数据的长度和，总长度字段为 16 位，所以可以数据报的最大长度为 2 的 16 次方减去 1，为 65535 字节。一般都不会到这么大，因为其下一层数据链路层的协议单元帧的最大传送单元 MTU，也是有规定的。比如以太网的规定 MTU 值是 1500 字节。若超过，需要分片处理。

标识：占 16 位。将 IP 软件在存储器中维持一个计数器，每产生一个数据报，计数器加一，并将此值赋值给标识字段。

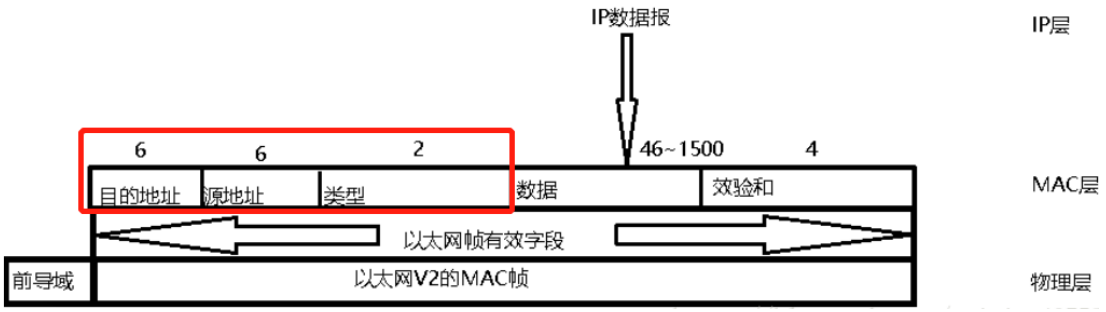
标志：占 3 位，MF，MF=1 表示还有分片，MF=0 表示最后一个分片。DF=0 允许分片，DF!=0 不能分片。

片偏移：占 13 位。以 8 字节为一个片偏移单位。表示某片在原分组中的相对位置。

生存时间（TTL）：占 8 位，数据报在网络中的寿命，以前以秒为单位，超过时间，或者 TTL 小于 1 秒，就丢弃数据报。由于路由器处理数据报速度加快，现在不用秒，而用跳数限制，TTL 的单位变成跳数。8 位最大为 255，所以数据报能在网络中，最多跳转 255 个路由。

协议：携带的协议，比如 ICMP、IGMP、IP、TCP、UDP、ESP、OSPF 等
首部检验和，占 16 位，只检查数据报首部，不检查数据不符。
源地址：占 32 位
目的地址：占 32 位

数据链路层首部（MAC 帧）：



HTTP 报文

应用层首部：

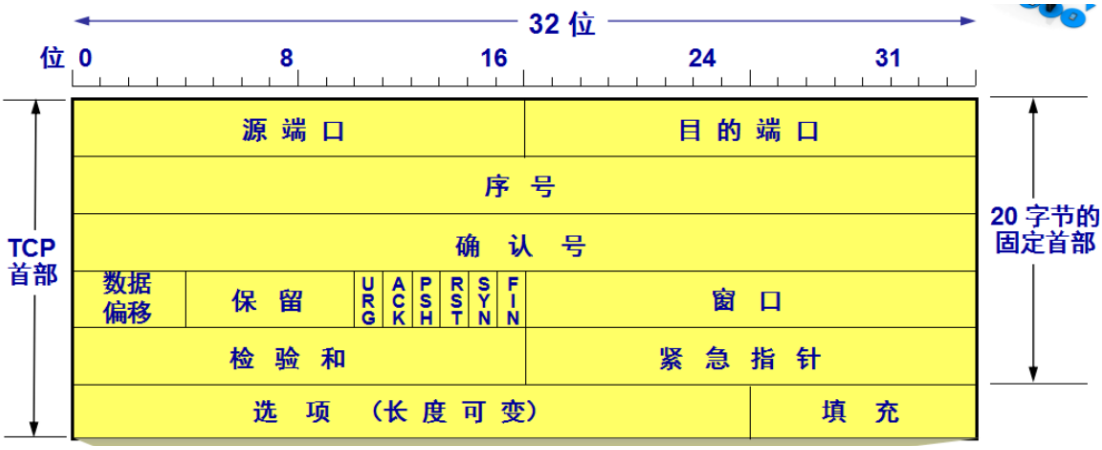
通用首部字段：请求报文和响应报文双方都会使用的字段。

请求首部字段：从客户端向服务器端发送请求报文时使用的首部。补充了请求的附加内容、客户端信息、响应内容相关优先级等信息。

响应首部字段：从服务器端向客户端返回响应报文时使用的首部。补充了相应的附加内容，也会要求客户端附加额外的内容信息。

实体首部信息：针对请求报文和响应报文的实体部分使用的首部。补充了资源内容更新时间等与实体有关的信息。

传输层首部（TCP 协议）：



源端口和目的端口各占 2 个字节，分别写入源端口和目的端口。序号占 4 字节。序号范围是[0，2^32－1]，共 2^32（即 4294967296）个序号。确认号占 4 字节，是期望收到对方下一个报文段的第一个数据字节的序号。数据偏移占 4 位，它指出 TCP 报文段的数据起始处距离 TCP 报文段的起始处有多远。保留占 6 位，保留为今后使用，但目前应置为 0。

紧急 URG 当 URG=1 时，表明紧急指针字段有效。它告诉系统此报文段中有紧急数据，应尽快发送（相当于高优先级的数据），而不要按原来的排队顺序来传送。确认 ACK 仅当 ACK = 1 时确认号字段才有效，当 ACK=0 时确认号无效。TCP 规定，在连接建立后所有的传送的报文段都必须把 ACK 置为 1。

推送 PSH 当两个应用进程进行交互式的通信时，有时在一端的应用进程希望在键入一个命令后立即就能收到对方的响应。在这种情况下，TCP 就可以使用推送（push）操作。这时，发送方 TCP 把 PSH 置为 1，并立即创建一个报文段发送出去。接收方 TCP 收到 PSH=1 的报文段，就尽快地（即“推送”向前）交付接收应用进程。而不用再等到整个缓存都填满了后再向上交付。

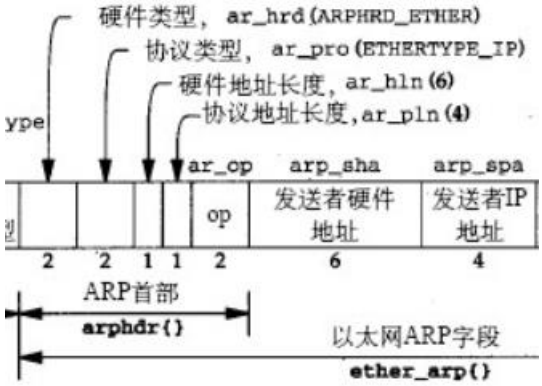
复位 RST 当 RST=1 时，表名 TCP 连接中出现了严重错误（如由于主机崩溃或其他原因），必须释放连接，然后再重新建立传输连接。RST 置为 1 还用来拒绝一个非法的报文段或拒绝打开一个连接。同步 SYN 在连接建立时用来同步序号。当 SYN=1 而 ACK=0 时，表明这是一个连接请求报文段。对方若同意建立连接，则应在响应的报文段中使 SYN=1 和 ACK=1，因此 SYN 置为 1 就表示这是一个连接请求或连接接受报文。

终止 FIN 用来释放一个连接。当 FIN=1 时，表明此报文段的发送发的数据已发送完毕，并要求释放运输连接。窗口占 2 字节。窗口值是[0，2^16-1]之间的整数。检验和占 2 字节。紧急指针占 2 字节。紧急指针仅在 URG=1 时才有意义，它指出本报文段中的紧急数据的字节数（紧急数据结束后就是普通数据。选项长度可变，最长可达 4 字节。当没有使用“选项”时，TCP 的首部长度是 20 字节。

网络层首部：和 DNS 报文的网络层首部一致。

数据链路层首部：和 DNS 报文的网络层首部一致。

ARP 网络层首部：



硬件类型：指链路层网络类型。1 为以太网

协议类型：指要转换的地址类型。0x0800 为 IP 地址

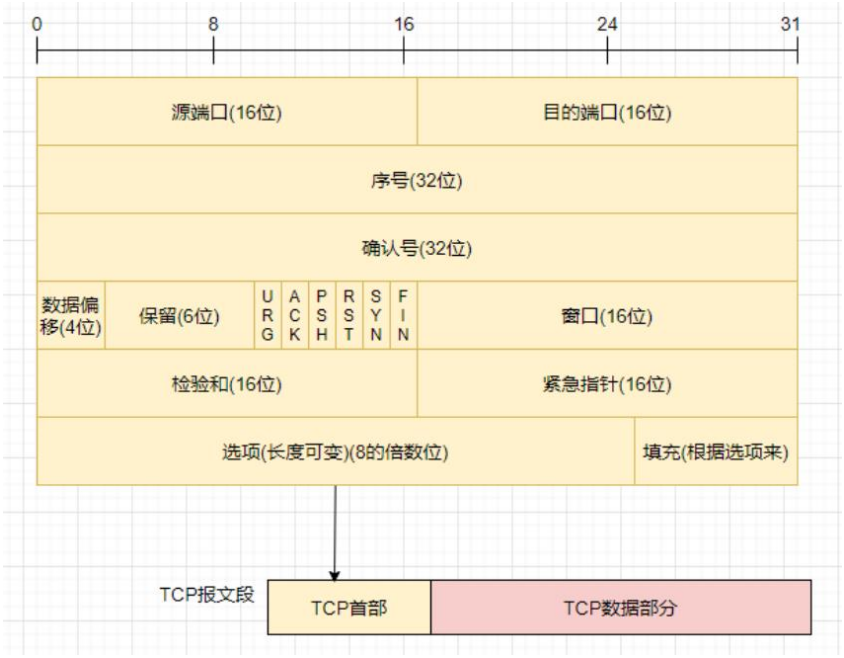
硬件地址长度：以太网地址长度（一般为 6 字节）

协议地址长度：IP 地址占 4 字节。

OP：是操作类型字段，值为 1，表示进行 ARP 请求；值为 2，表示进行 ARP 应答；值为 3，表示进行 RARP 请求；值为 4，表示进行 RARP 应答。

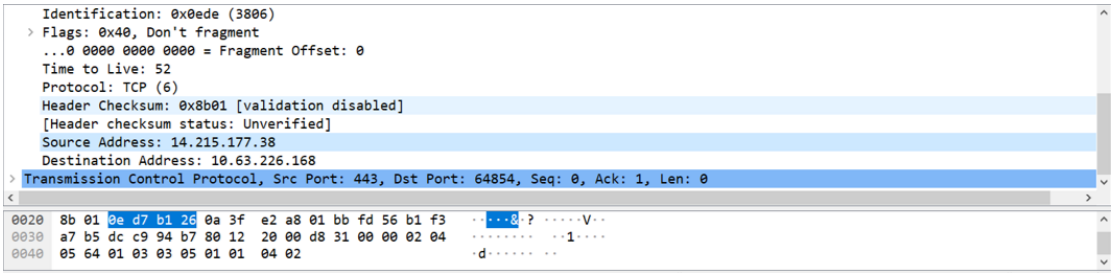
②ARP 报文格式分析：

③TCP 报文格式分析：



④通过对捕获的数据包进行分析，提供证据，说明如何获得以下信息：

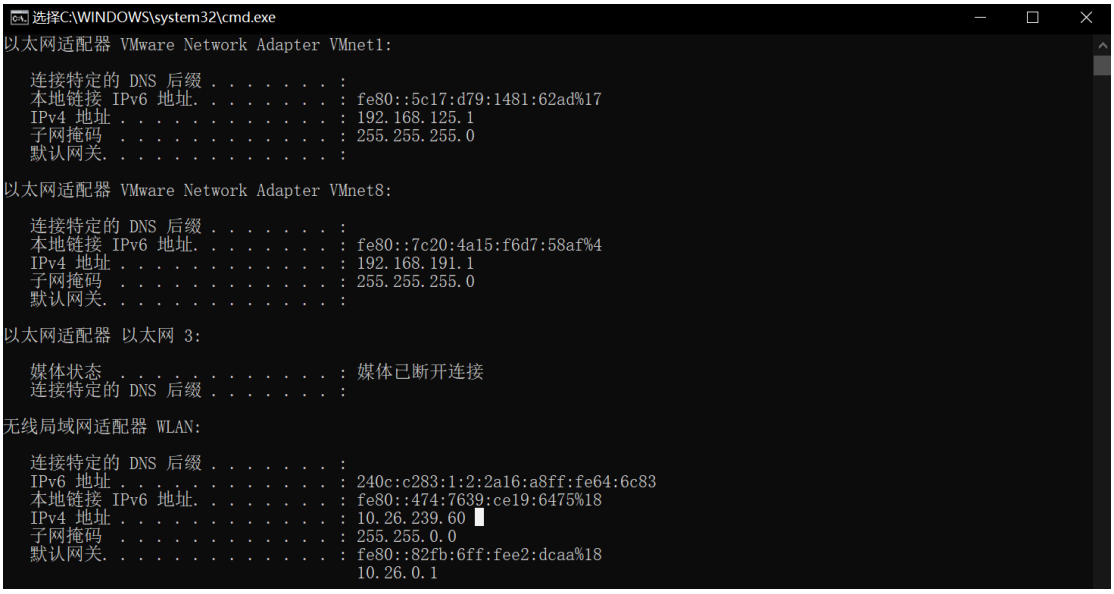
a. 百度的 IP 地址：

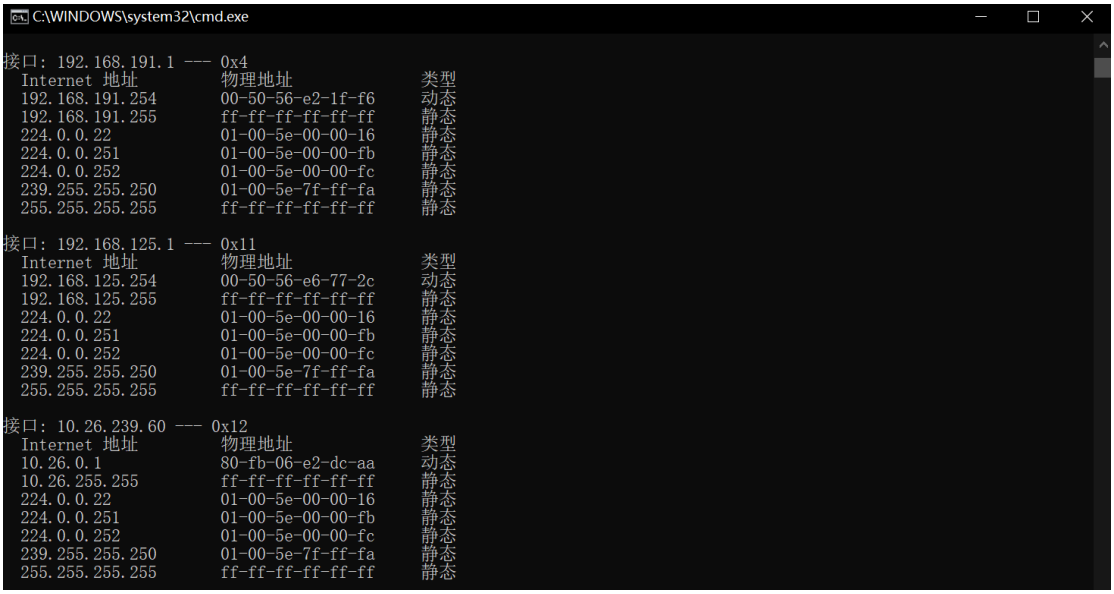


由于之前 DNS 服务器与百度建立了联系，因此此时应该是本机与 www.baidu.com 对应的 IP 地址通过 TCP 建立连接。

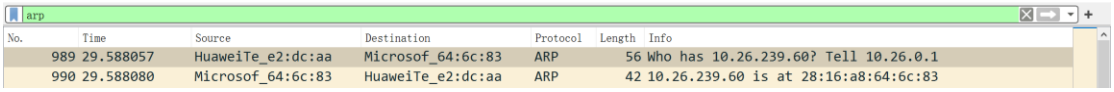
b. 以下介绍两种方法来获得网关 IP 地址和 MAC 地址：

法一：在 dos 命令框中通过 ipconfig 获得对应的默认网关，使用 arp -a 指令就可以获得默认网关所对应的 mac 地址：

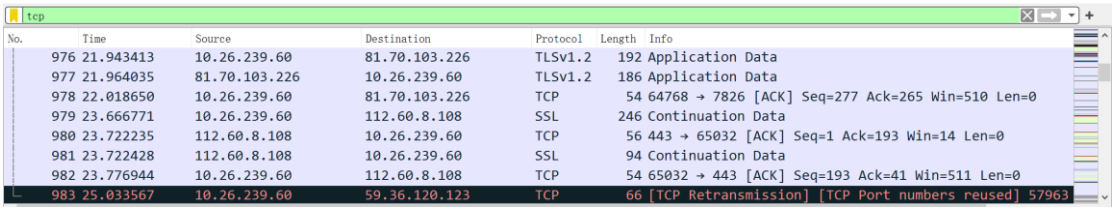




法二：通过在 wireshark 中用 arp 过滤看到对应的 mac 地址。

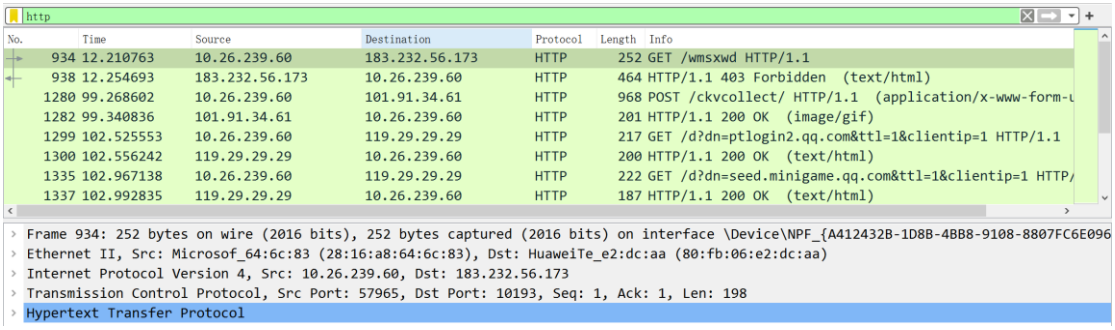


c.发送方和接收方 TCP 协议协商的初始序号？真真发送数据的实际起始序号是多少？为什么？



由以上报文可以看出：初始序号为 0，而真正开始发送数据的起始序号为 518，原因是 TLS 协议是夹在 tcp 与应用层之间的协议，在 tcp 建立连接的过程中也进行了 tls 协议的数据传输，从而导致序列号后移。

d. HTTP 协议协商的版本号是多少？该版本号 HTTP 协议工作特点是什么？提供证据说明。



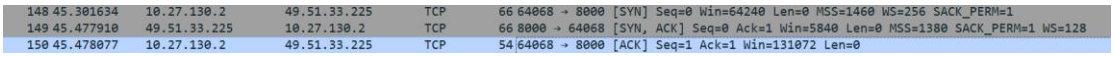
如上图所示，HTTP 协商的版本号应该是 1.1。

e. 一个 TCP 连接从建立到释放，总共发送和接收了多少字节数据？为什么？

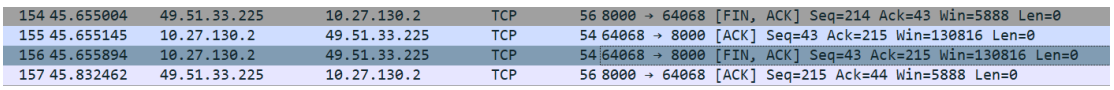
TCP 连接的三次握手的前两次是不可以携带数据的，逻辑上看，连接还没建立，携带数据也不合适。而建立连接的三次握手过程中的第三次握手允许携带数据；在释放连接之前两方都会确认已经将数据发送完毕，因此在释放连接过程中应该不会发送和接收数据。

f. 针对一个 TCP 连接，提供该连接建立三次握手报文段和四次挥手报文段，为什么说该证据是针对一个 TCP 连接？

三次握手：



四次挥手：



可以看到，tcp 建立连接时与释放连接时的 IP 地址是相互对应的，因此这针对的是一个 tcp 连接。

⑤以 HTTP 请求报文为例子，当 WEB 服务器接收到该报文后，接收方从数据链路层到应用层如何知道不同层数据字段的长度，开始和起始位置。

应用层：

浏览器封装 HTTP 请求报文，然后创建一个 TCP 套接字，采用四元组标识，具体为「源 IP 地址:192.168.43.138」+「源端口号:随机的,这里假设为 1234」+「目的 IP 地址:172.194.72.105」+「目的端口号：80」。

传输层：

运输层收取了报文，并判断与目的主机是否建立了 TCP 连接，这里假设没有。

那么，运输层将不会先发送应用层数据，得先判断与目的主机之间能够正常通讯，也就是需要『握手』打招呼。 『三次握手』的相关细节，这里也不再赘述了，上篇文章描述的很详细了，通过『三次握手』，发送端和接收端确认过发送与确认序号，分配了相应的缓存资源等。

一切准备就绪之后，运输层将应用层发过来的数据报又一层封装，添加进『源端口号』和『目的端口号』以及相关差错检验字段。

最后将 TCP 数据报向下传递到网络层。

网络层：

网络层其实很简单，拿到数据报并封装成 IP 数据报，即在原 TCP 报文的前提之上添加『源 IP 地址』和『目的 IP 地址』等字段信息。

链路层：

数据链路层拿到 IP 数据报，它需要封装成以太网帧才能在网络中传输，也就是它需要目的主机的 Mac 地址，然而我们只知道目的主机的 IP 地址。所以，链路层有一个 ARP 协议，直接或间接的能够根据目的 IP 地址获得使用该 IP 地址的主机 Mac 地址。

当然，ARP 协议运行的前提是，目的 IP 地址和当前发送方主机处于同一子网络中。如果不然，发送方将目的 Mac 地址填自己网关路由的 Mac 地址，然后通过物理层发送出去。

网关路由由于具有转发表和路由选择算法，所以它知道目的网络该怎么到达，所以一路转发，最终会发送到目的网络的网关路由上。最后，目的网络的网关路由同样会经由 ARP 协议，取得目的主机的 Mac 地址，然后广播发送，最后被目的主机接受。 而这些无论是报文还是分组还是数据帧中则会包含相应的数据信息。

⑥从应用层到数据链路层有哪些校验字段，分别采用什么方法计算校验码，其校验范围分别是什么，不同层重复的校验是多余的吗？为什么？

有哪些校验字段已在上述报文格式中进行呈现完成了。数据链路层中会对每一帧数据进行 CRC 校验，传输层 udp 和 tcp 也有校验字段，网络层的 ip 报头部也有校验字段，应用层可能用户程序也有一定的校验手段，层层校验的目的就是保证数据传输的无误，没有发错给其他人，没有中途发生数据异常变化。

⑦如果在本次实验过程，对抓取的报文进行分析，发现 DNS 和 ARP 协议没有工作，为什么？如何解决该问题？在解决该问题过程中用到两个网络命令，分别是什么，写出这两个命令具体应用。

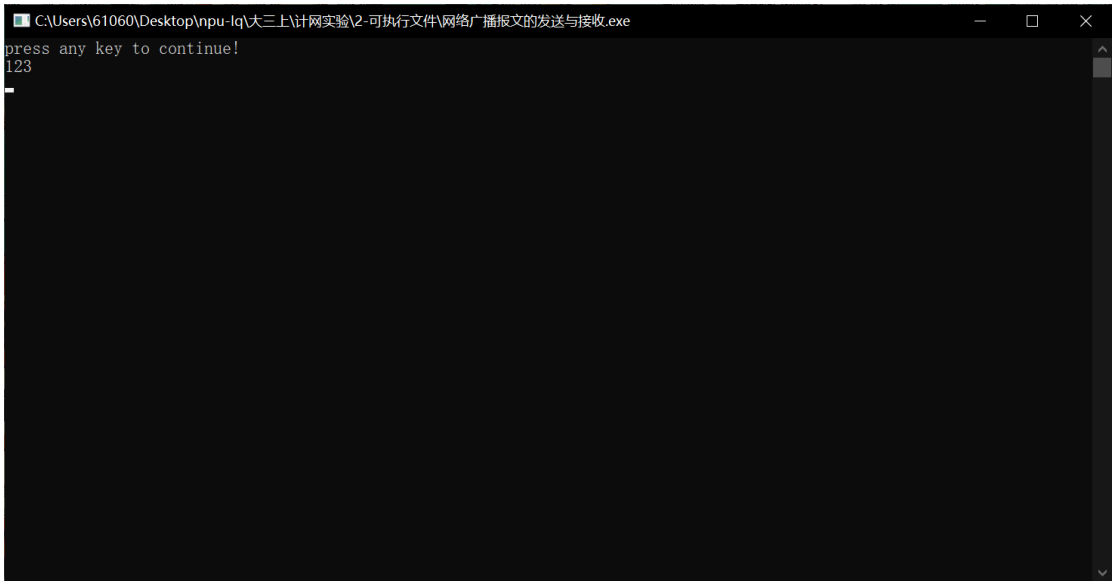
有可能是 dns 服务器的设置出现问题，导致不能找到对应的服务器，或是端口禁止访问了，对应的命令是 ping 和 netstat。

⑧如果在本次实验过程，用户在客户端 DOS>ping www.baidu.com,连续发送了三次 ICMP ECHO 请求报文，但显示第一次接收 ICMP ECHO 应答报文超时，说明网络不同；但后面两次 ICMP ECHO 应答报文接收正常，又说明网络是连通的，为什么？

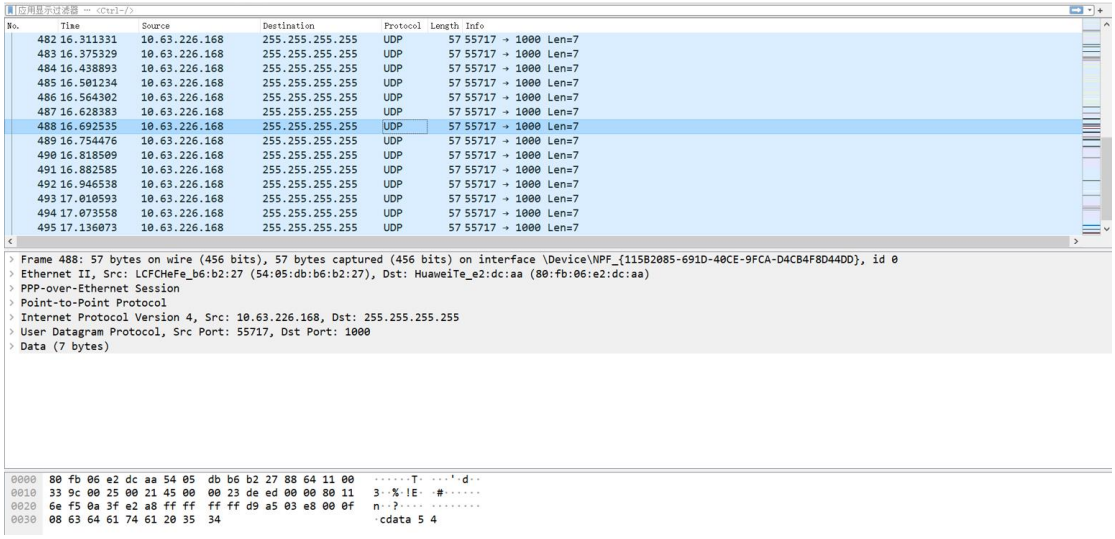
于 ICMP 报文，目标不可到达、源抑制和超时报文三种报文的格式是一样的，都会在连接无法建立时返回，因此应答报文超时不一定是网络不可到达，有可能是在第一次发送报文时目的端口被占用或未被寻找到，但之后的报文可以正常被目的端接收并发出应答报文。

实验内容二：

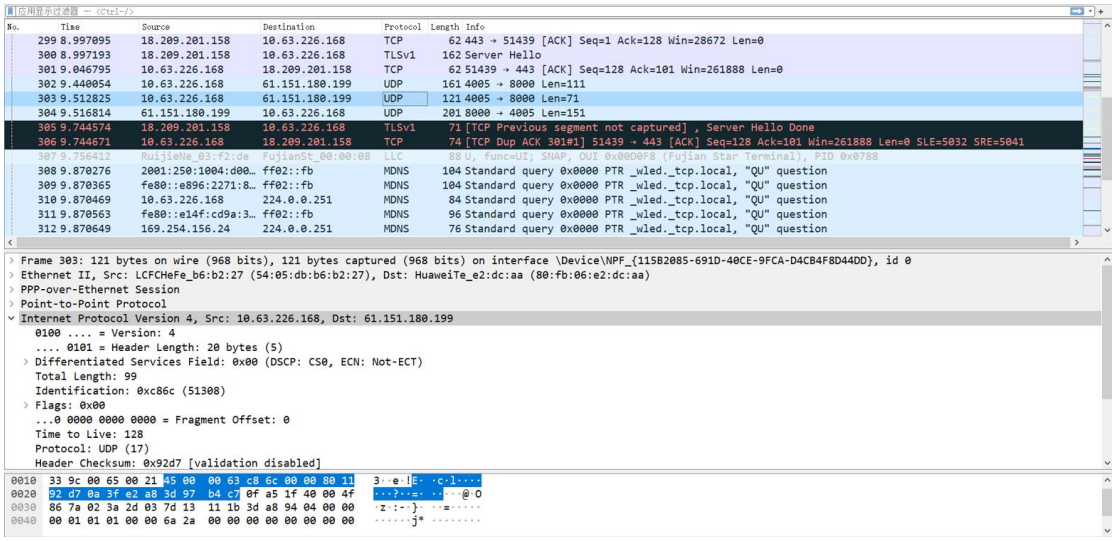
先打开抓包工具，随后运行程序：



之后可以在抓包工具中观察到：



此时的 UDP 帧中目的地址全部为 255.255.255.255，这是发送出的广播地址，与其他 UDP 帧进行对比可以很清楚的发现，目的 IP 地址与目的 MAC 地址有很明显的区别：



与单播 UDP 用户数据报的区别在于，目的 MAC 地址和目的 IP 地址不会是 ff:ff:ff:ff 和 255.255.255.255，而是目的用户的地址，ff:ff:ff:ff 和 255.255.255.255 是广播报文的专属地址。

广播或者组播因为要对局域网中的所有主机 IP 都发送，而 TCP 为了保证传输的可靠性一般不会建立多路的连接，因此使用 UDP 这种无连接就可以直接发送的协议会更符合广播的需要。