

《计算机网络与通信》课程 实验二

2023年秋季学期

实验目的

- 学习捕获和分析网络数据包
- 掌握以太网MAC帧、802.11数据帧和IPv4/IPv6数据包的构成，了解各字段的含义
- 掌握ICMP协议，ping和tracert指令的工作原理
- 掌握ARP协议的请求/响应机理

实验内容

► 任务1：捕获和分析有线以太网数据包

准备步骤：学习Wireshark基本操作

1.1 分析MAC帧

1.2 分析IP数据报首部

1.3 观察IP分片

1.4 ICMP协议分析（以ping指令为例）

1.5 tracert工作原理分析

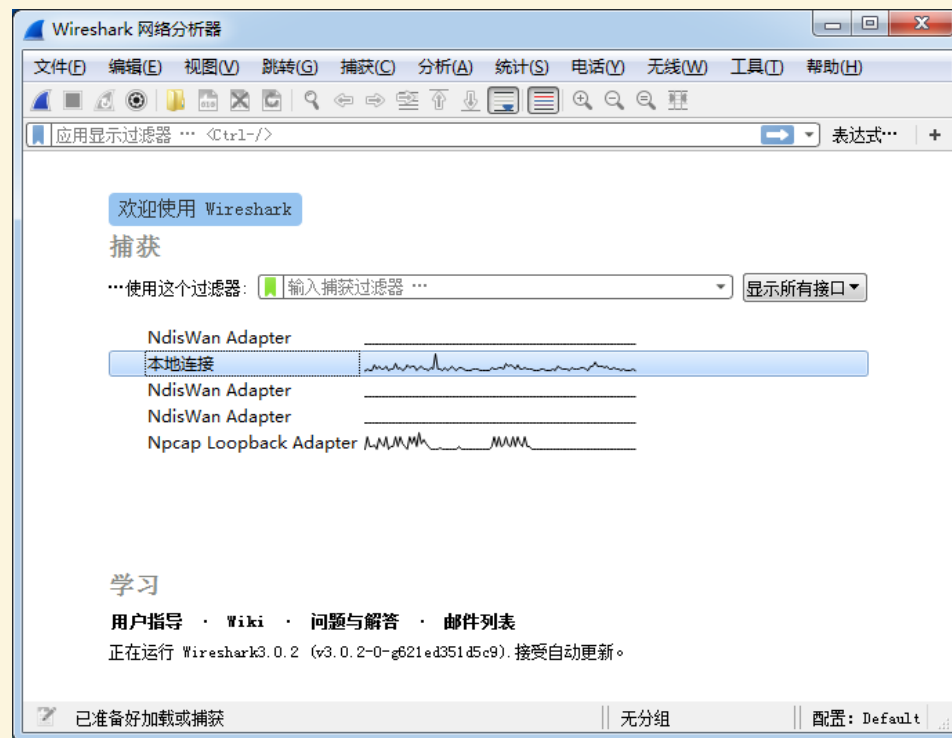
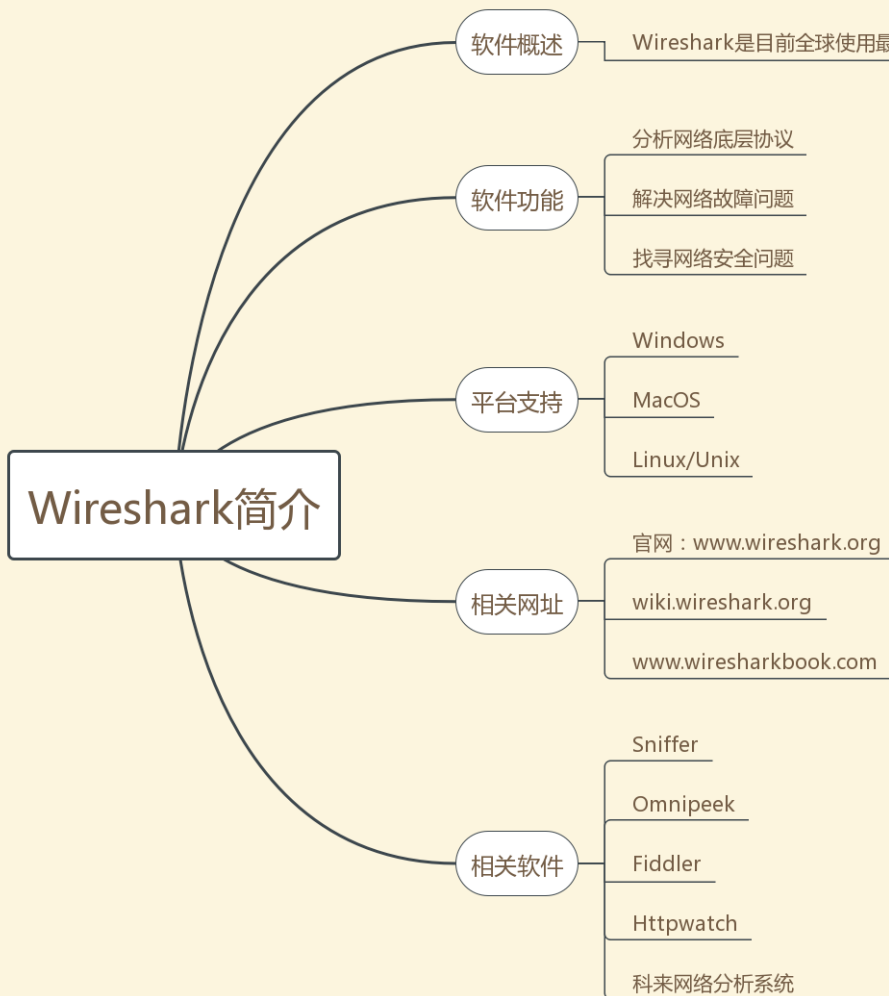
1.6 ARP协议分析

► 任务2：捕获和分析802.11数据

构建无线环境，捕获无线数据包、分析802.11数据

► 任务3：探索Wireshark更多功能和其它抓包工具(选做)

Wireshark 简介



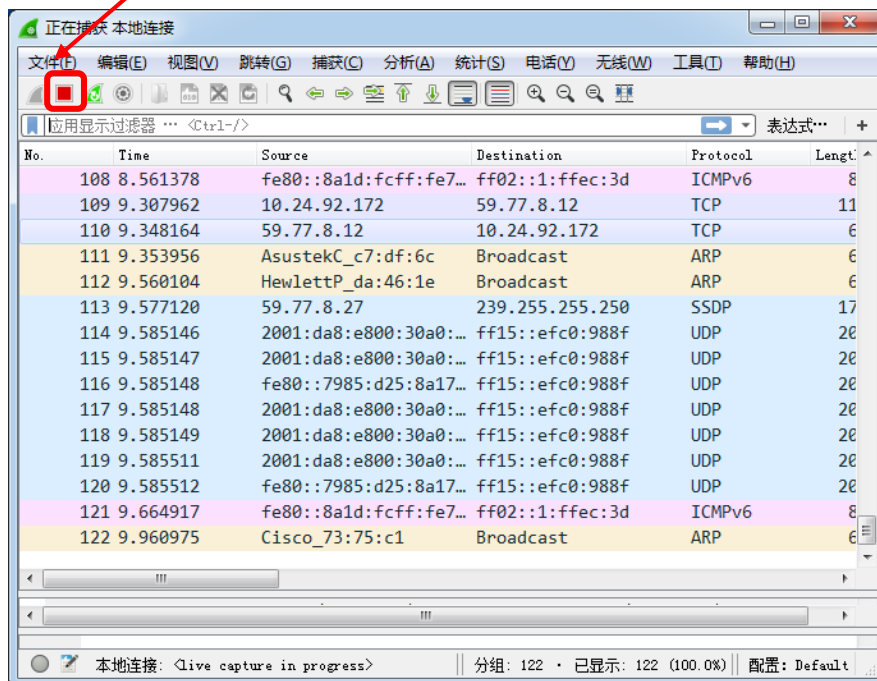
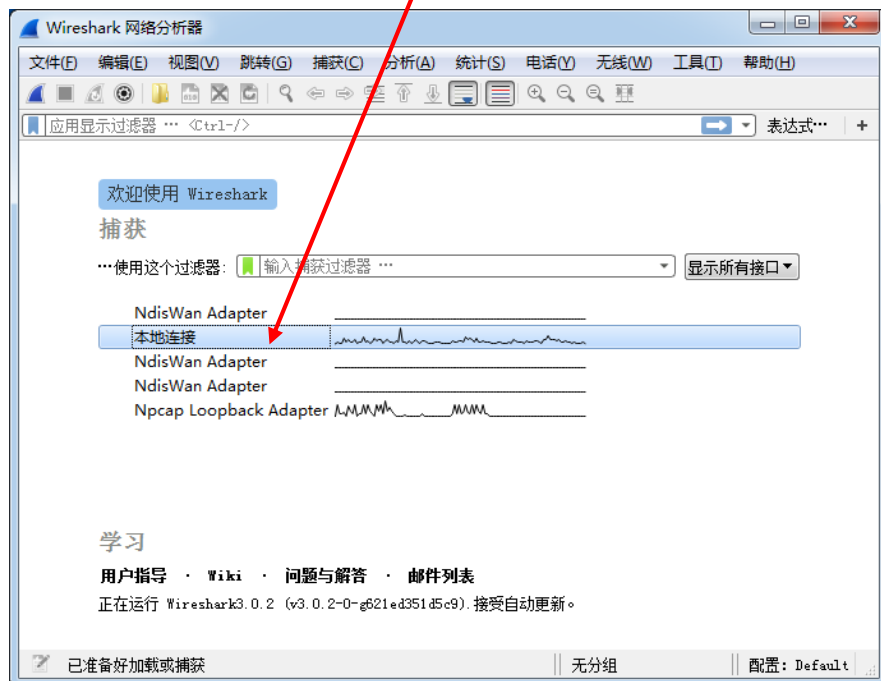
Wireshark 简介

1. 打开桌面上安装好的Wireshark



2. 程序界面显示了当前的网络接口列表，

选择要抓取的**网络接口**，双击开始抓包，点击**stop**按钮停止抓包。



主窗口信息

The image shows a Wireshark network traffic capture window titled "*Realtek PCIe GBE Family Controller: 本地连接". The interface includes a menu bar, a toolbar, and a packet list table. The selected packet (No. 6) is an ICMP Echo (ping) request. Below the packet list, the packet details pane shows the structure of the frame: Ethernet II, Internet Protocol Version 4, and Internet Control Message Protocol. At the bottom, the packet bytes pane displays the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	59.77.8.36	78.119.36.59.broad...	UDP	89	pxc-spvr(4006) → irdmi(8000)
2	0.030559	78.119.36.59.broad.dg.gd...	59.77.8.36	UDP	121	irdmi(8000) → pxc-spvr(4006)
3	0.079797	Benchmark_8c:57:f9	Broadcast	ARP	60	Who has 59.77.8.69? Tell 59.7
4	0.449833	192.168.32.209	255.255.255.255	DNS	87	Standard query 0xfe4a PTR 20
5	0.635711	NewH3CTe_1c:fc:9b	Spanning-tree-(for-...	STP	119	MST. Root = 32768/0/3c:f5:cc
6	1.045557	59.77.8.36	59.77.8.1	ICMP	74	Echo (ping) request id=0x010
7	1.047764	59.77.8.1	59.77.8.36	ICMP	74	Echo (ping) reply id=0x010
8	1.079701	Benchmark_8c:57:f9	Broadcast	ARP	60	Who has 59.77.8.69? Tell 59.7
9	1.197331	NewH3CTe_1d:4b:3b	Broadcast	ARP	60	Gratuitous ARP for 192.168.3
10	1.197594	HewlettP_01:4b:db	Broadcast	ARP	60	Who has 59.77.8.1? Tell 59.7
11	1.197595	HewlettP_01:4b:db	Broadcast	ARP	60	Who has 59.77.8.1? Tell 59.7
12	1.387523	59.77.8.36	dns1.xmu.edu.cn	DNS	83	Standard query 0x4e8c PTR 34
13	1.785287	2001:da8:e800::35	2001:da8:e800:30a0::...	TCP	74	http(80) → 50156 [FIN, ACK] !

Frame 6: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0

- Ethernet II, Src: Dell_85:da:d0 (b8:ca:3a:85:da:d0), Dst: Cisco_73:75:c1 (88:1d:fc:73:75:c1)
- Internet Protocol Version 4, Src: 59.77.8.36 (59.77.8.36), Dst: 59.77.8.1 (59.77.8.1)
- Internet Control Message Protocol

0000 88 1d fc 73 75 c1 b8 ca 3a 85 da d0 08 00 45 00 ...su...:....E.
0010 00 3c 43 50 00 00 80 01 00 00 3b 4d 08 24 3b 4d <CP....:;M;\$;M
0020 08 01 08 00 44 9f 01 00 07 bd 61 62 63 64 65 66D...:abcdef
0030 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 ghijklmn opqrstuv
0040 77 61 62 63 64 65 66 67 68 69 wabcdegh hi

Internet Control Message Protocol (icmp), 40 字节 | 分组: 42 · 已显示: 42 (100.0%) · 已丢弃: 0 (0.0%) | 配置: Default

分层解读

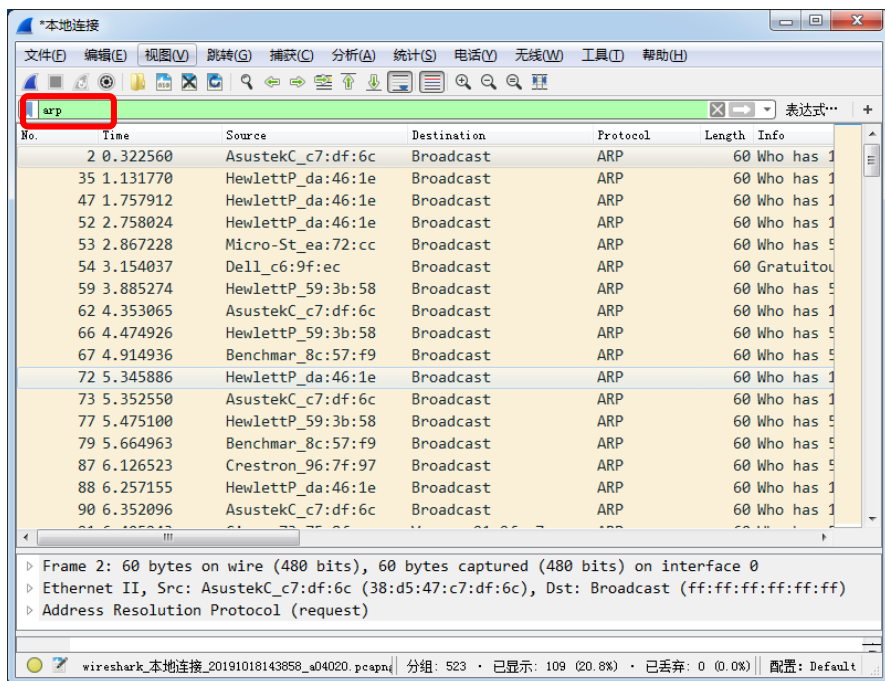
原始数据

数据包过滤

过滤抓取到的数据包，在显示过滤框中输入过滤器表达式：

- arp //显示arp协议报文

- ip.src == a.b.c.d && icmp //显示源地址为a.b.c.d的icmp报文



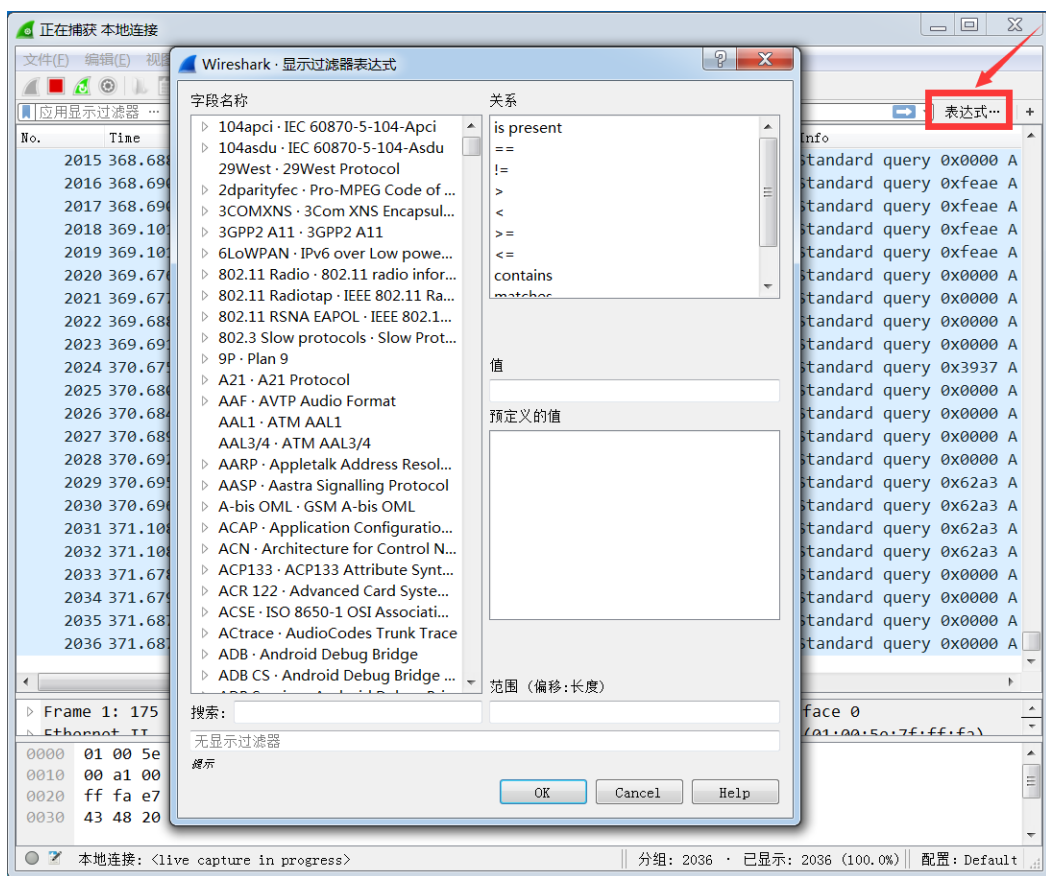
► 7 更多的内容请参阅[WireShark帮助文档](#)

显示过滤器

作用：针对已捕获的报文，过滤出符合过滤规则的报文

Wireshark官网简单示例 >>[点这里](#)

全面详细的语法解释 >>[点这里](#)



所有过滤规则都可通过点击右上角的表达式查看

显示过滤器的示例

① 过滤特定端口

tcp.port eq 80 : 显示源端口或目的端口为80的报文

tcp.srcport eq 80: 显示源端口为80的报文

tcp.dstport eq 55332: 显示目的端口为55332的报文

② 按长度进行过滤

tcp.len >0: 过滤具有有效载荷的tcp报文

udp.length == 26 : 过滤长度为26字节的udp包

ip.len eq 1500 : 过滤长度为1500字节的IP数据包。

frame.len le 128: 过滤数据帧总体长度小于128B的记录

③ 过滤IP地址

ip.src eq 192.168.1.107 : 按源IP地址过滤的数据包

ip.dst eq 192.168.1.107 : 按目的IP地址过滤的数据包

ip.addr eq 192.168.1.107: 不区分源和目的IP地址过滤的数据包

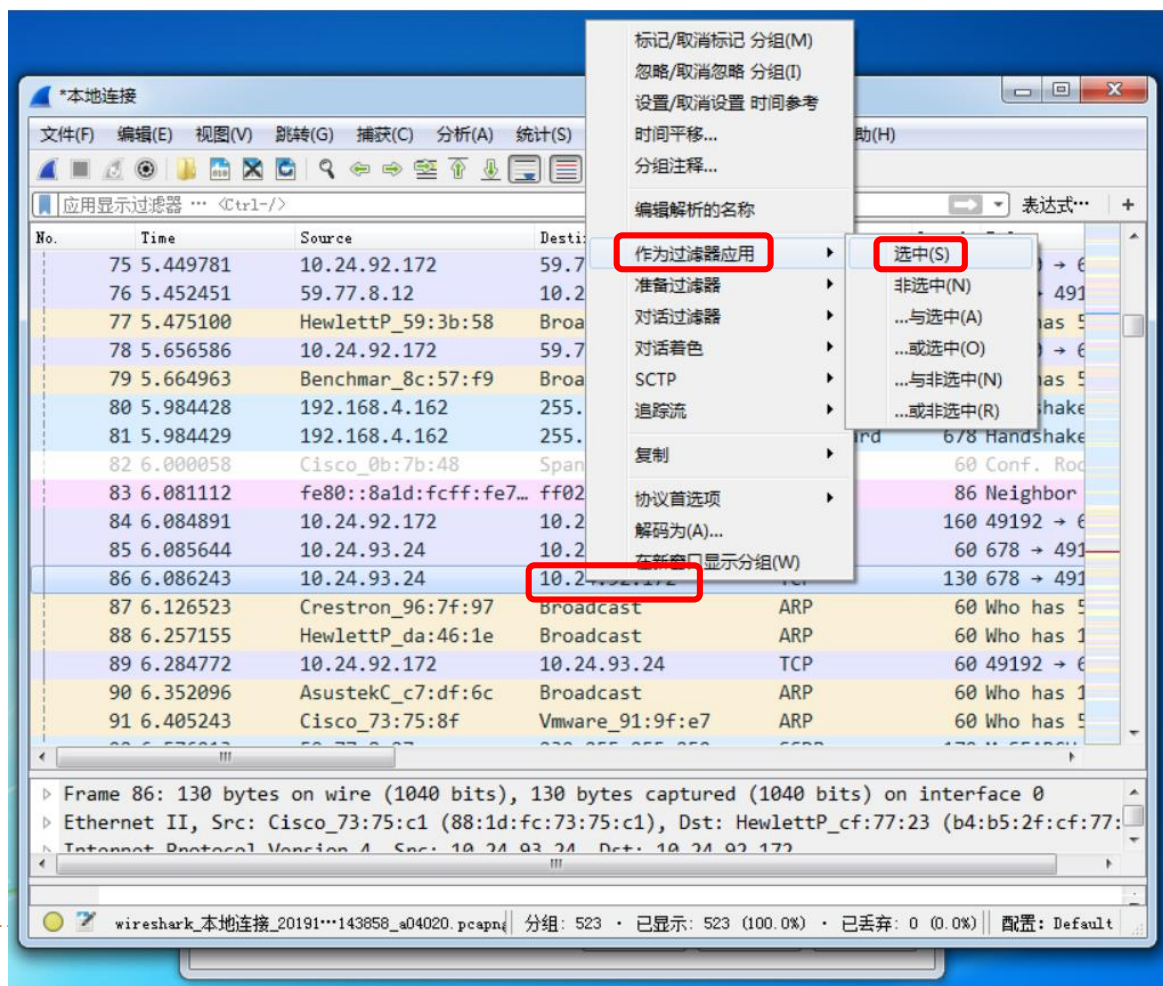
④ 过滤包含特定字符串域名的报文，如：http.host contains “xmu”

⑤ 多个过滤条件，例如满足http以及且端口为80的过滤规则：http && tcp.port == 80

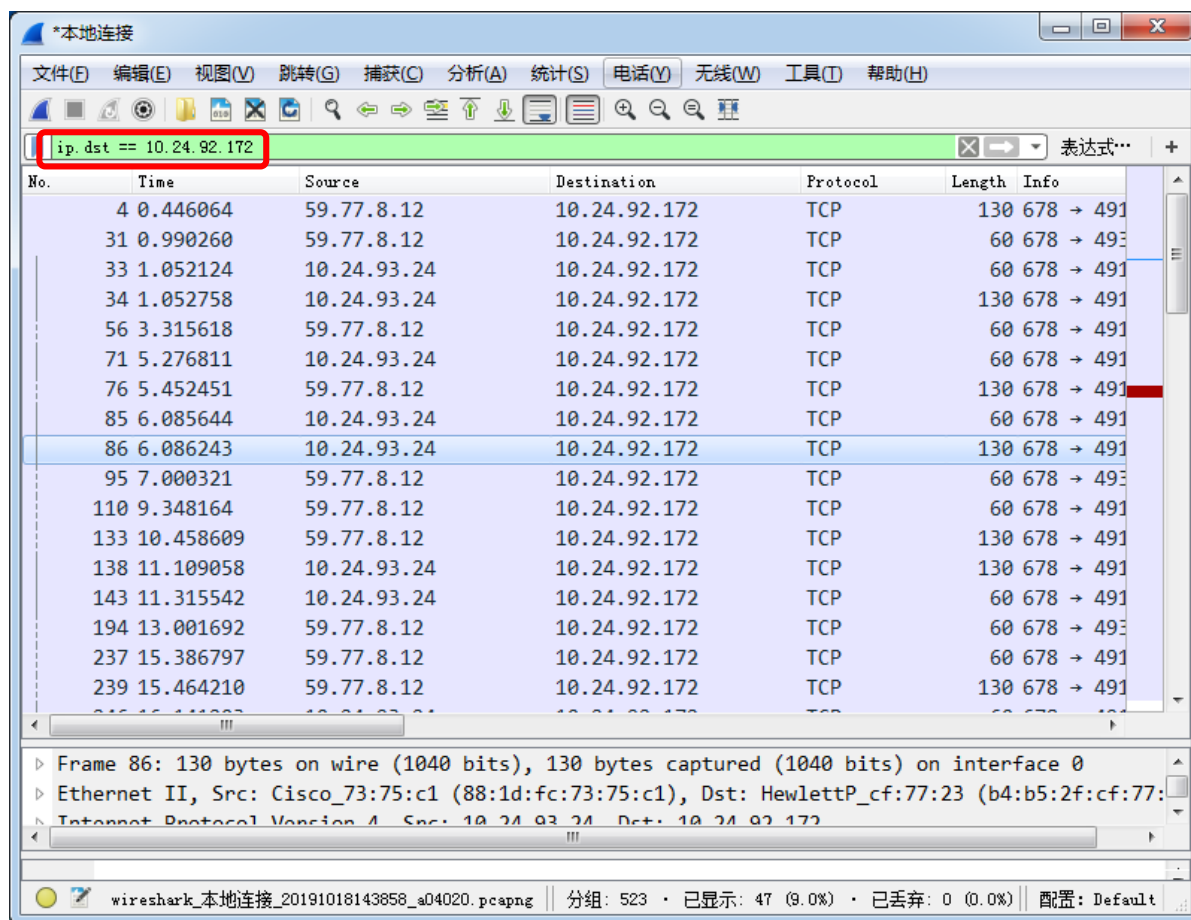
组合符号有“且” and (&&), “或” or (||), “取反” not (!) 这几种

生成显示过滤器表达式 (I)

- ▶ 选中想要过滤的报文属性->右击->作为过滤器应用->选中
- ▶ 过滤器显示栏就会出现想要的过滤规则模板，在此基础上进行修改



生成显示过滤器表达式 (II)



捕获过滤器的使用

捕获过滤器 提前设置好过滤规则，只捕获符合过滤规则的报文

Wireshark官网简单示例 [>>点这里](#)

全面详细的语法解释 [>>点这里](#)

示例：

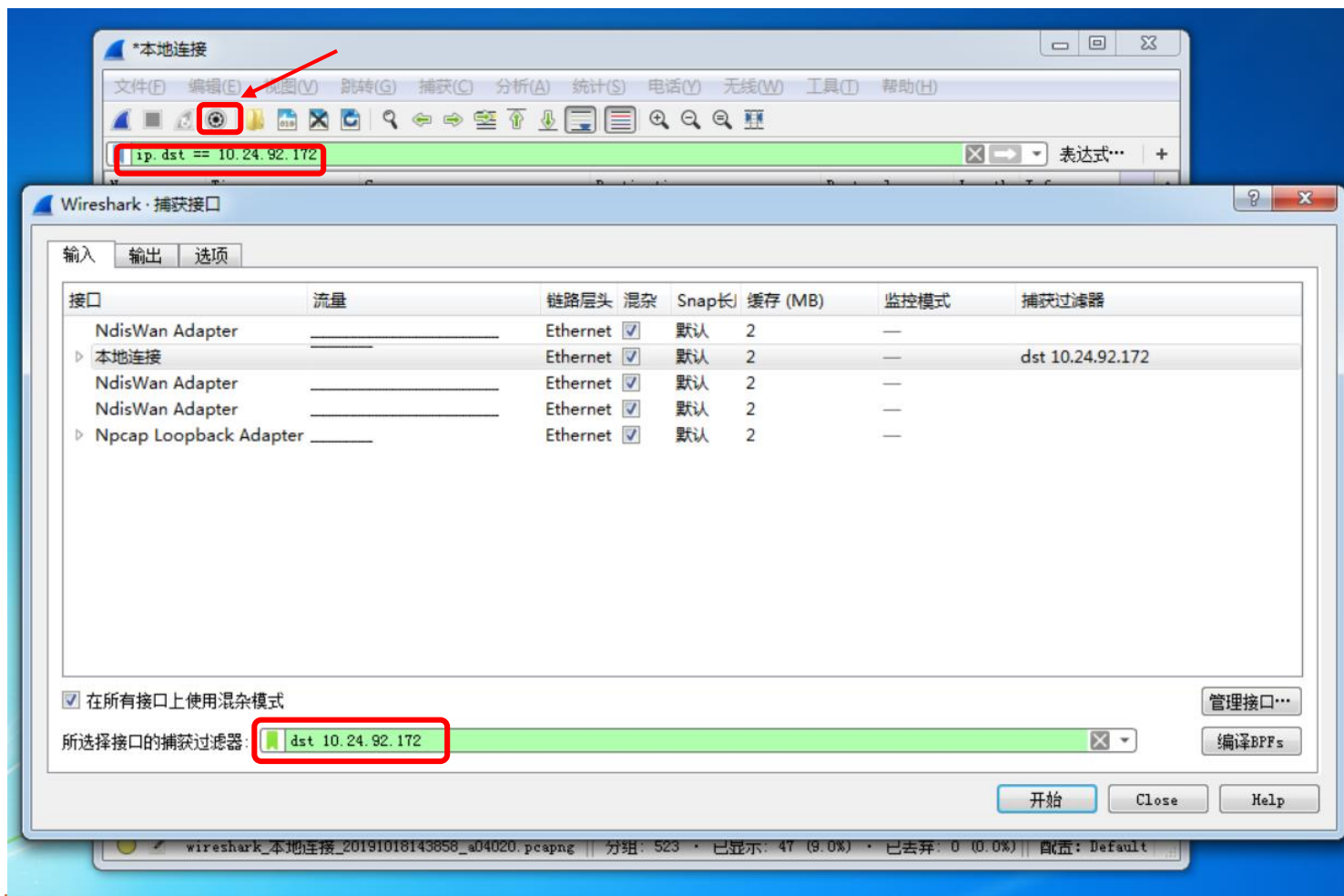
①过滤指定端口规则：port 80

②过滤指定IP规则：host 10.24.90.XXX

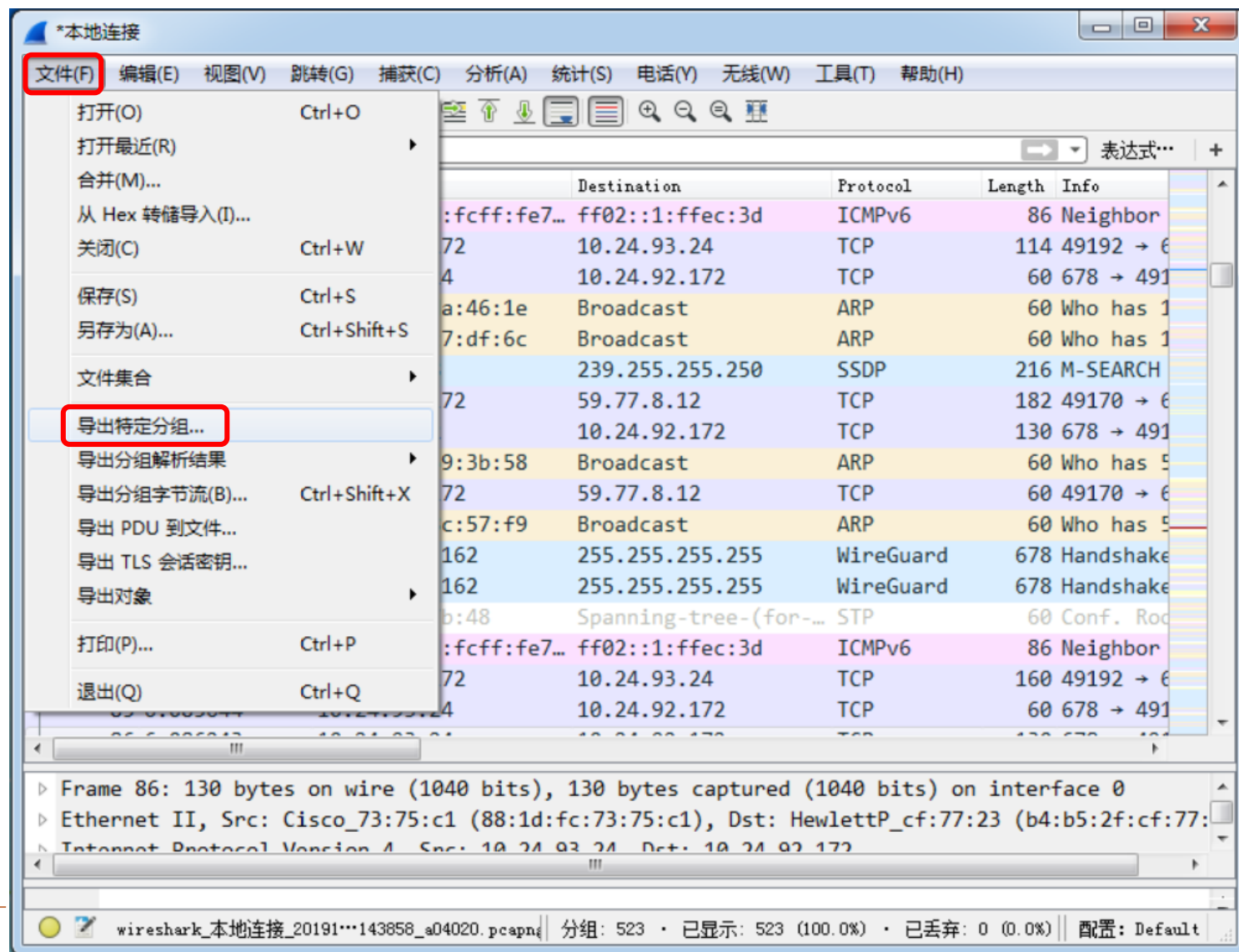
③过滤指定域名规则：host www.example.com

捕获过滤器

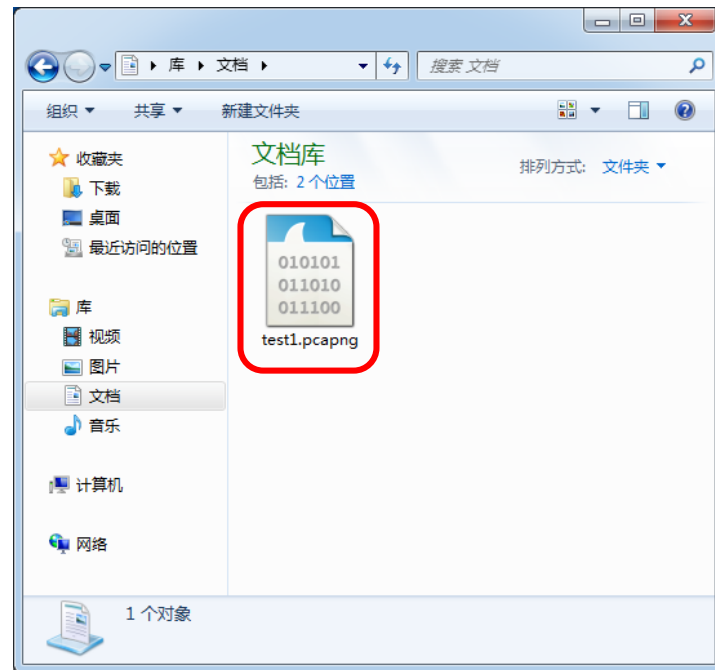
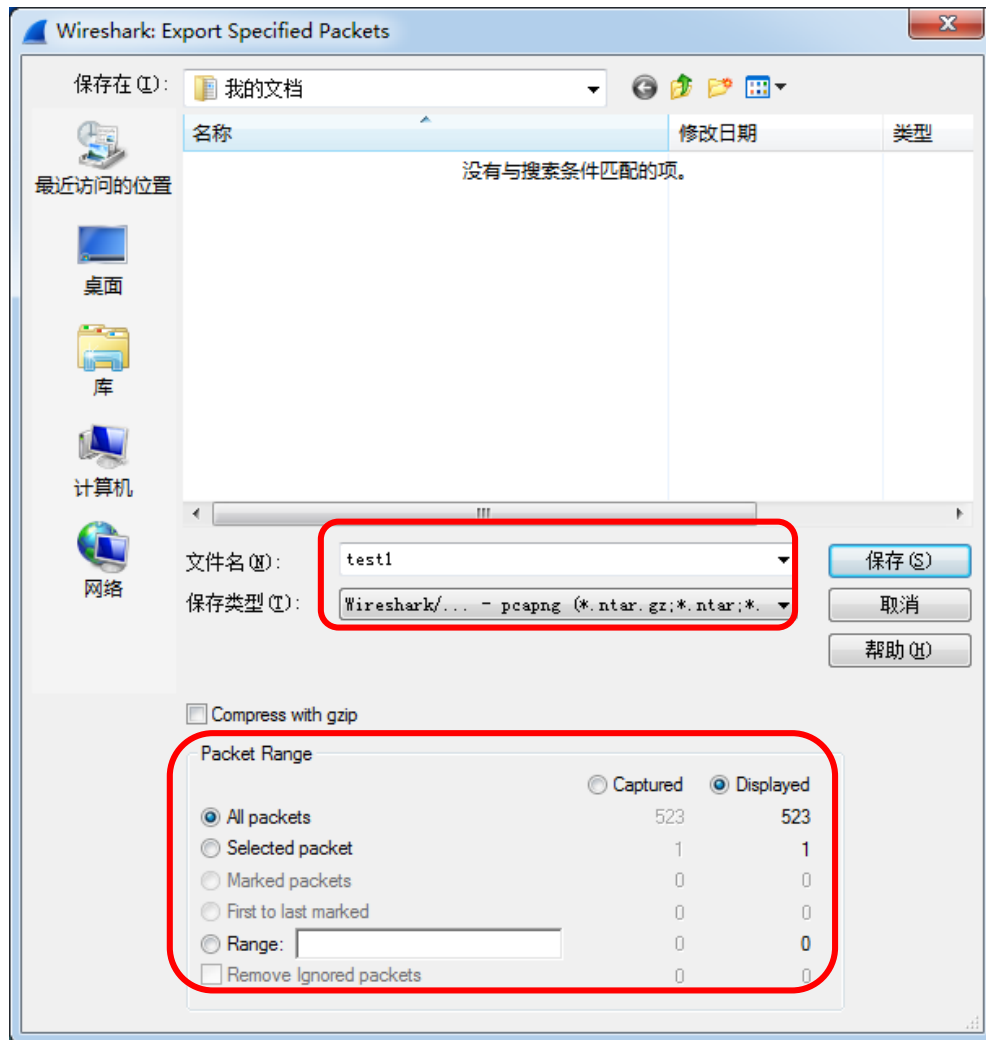
► 使用方法



数据包保存



数据包保存



任务1：捕获和分析有线以太网数据包

- ping 命令，是用来测试两台主机是否连通的有效工具。

```
ping [-t] [-a] [-n count] [-l length] [-f] [-i ttl] [-v tos] [-r count] [-s count]  
      [-j computer-list] | [-k computer-list]  
      [-w timeout] destination-list
```

-a: 将地址解析为计算机名。

-n count: 发送count指定的ECHO数据包数。默认值为4。


-l length: 发送包含由 length指定的数据量的ECHO数据包。默认为32字节；最大值是 65,527。

-f: 在数据包中发送“不要分片”标志，避免数据包被路由上的网关分片。

-i ttl: 将“生存时间”字段设置为ttl指定的值。

捕获ping数据 (IPv4)

先启动Wireshark抓包，然后在终端窗口发起命令：*ping IP/域名*
停止抓包后，先保存数据、再开始分析



```
管理员: C:\Windows\System32\cmd.exe
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Windows\system32>ping 10.24.90.1

正在 Ping 10.24.90.1 具有 32 字节的数据:
来自 10.24.90.1 的回复: 字节=32 时间=7ms TTL=255
来自 10.24.90.1 的回复: 字节=32 时间=2ms TTL=255
来自 10.24.90.1 的回复: 字节=32 时间=3ms TTL=255
来自 10.24.90.1 的回复: 字节=32 时间=2ms TTL=255

10.24.90.1 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间<以毫秒为单位>:
    最短 = 2ms, 最长 = 7ms, 平均 = 3ms

C:\Windows\system32>
```

捕获ping数据 (IPv6)

启动Wireshark，并在终端窗口发起命令：`ping -6 IPv6地址/域名`
停止抓包后，先保存数据、再开始分析。

示例：`ping -6 www.xmu.edu.cn`

```
C:\Users\admin>ping -6 2001:da8:e800:68:a5de:e411:395f:c152

正在 Ping 2001:da8:e800:68:a5de:e411:395f:c152 具有 32 字节的数据:
来自 2001:da8:e800:68:a5de:e411:395f:c152 的回复: 时间<1ms
来自 2001:da8:e800:68:a5de:e411:395f:c152 的回复: 时间<1ms
来自 2001:da8:e800:68:a5de:e411:395f:c152 的回复: 时间<1ms
来自 2001:da8:e800:68:a5de:e411:395f:c152 的回复: 时间<1ms

2001:da8:e800:68:a5de:e411:395f:c152 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 0ms, 最长 = 0ms, 平均 = 0ms

C:\Users\admin>_
```

过滤出与本机ip相关的数据

- ▶ 在Wireshark输入过滤条件，显示与本机ip相关的数据，选择其中一个分组进行分析。
- ▶ 本机ip，也可以在终端窗口输入**ipconfig**查看：

```
以太网适配器 本地连接:

    连接特定的 DNS 后缀 . . . . . : 
    IPv6 地址 . . . . . : 2001:da8:e8
    本地连接 IPv6 地址. . . . . : fe80::98e6=
    IPv4 地址 . . . . . : 59.77.8.36
    子网掩码 . . . . . : 255.255.255
    默认网关. . . . . : fe80::8a1d=
                        fe80::1x13
                        59.77.8.1
```

- ▶ **ipconfig**的命令学习

显示过滤器的示例

① 过滤特定端口

tcp.port eq 80 : 显示源端口或目的端口为80的报文

tcp.srcport eq 80: 显示源端口为80的报文

tcp.dstport eq 55332: 显示目的端口为55332的报文

② 按长度进行过滤

tcp.len >0: 过滤具有有效载荷的tcp报文

udp.length == 26 : 过滤长度为26字节的udp包

ip.len eq 1500 : 过滤长度为1500字节的IP数据包。

frame.len le 128: 过滤数据帧总体长度小于128B的记录

③ 过滤IP地址

ip.src eq 192.168.1.107 : 按源IP地址过滤的数据包

ip.dst eq 192.168.1.107 : 按目的IP地址过滤的数据包

ip.addr eq 192.168.1.107: 不区分源和目的IP地址过滤的数据包

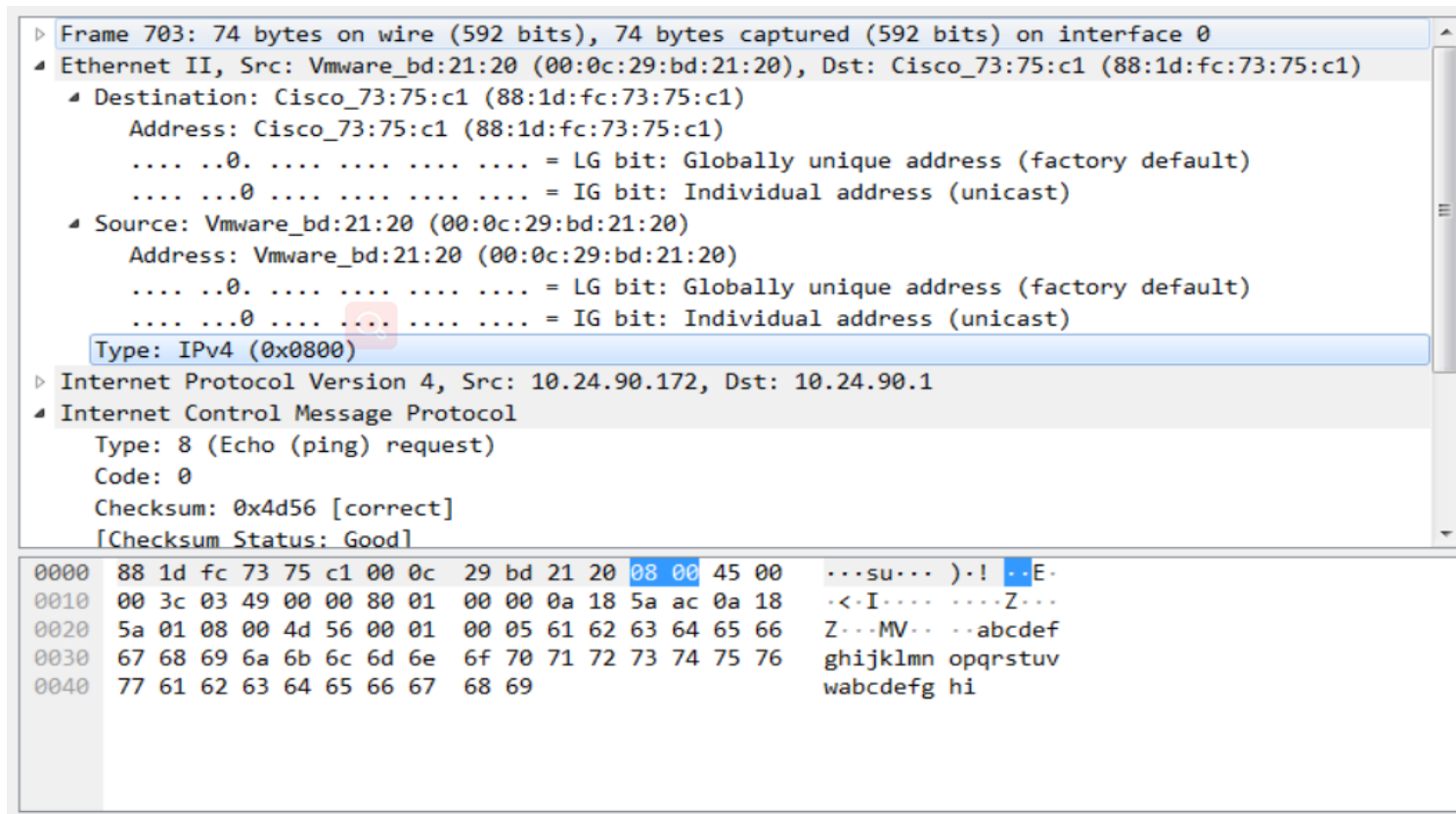
④ 过滤包含特定字符串域名的报文，如：http.host contains “xmu”

⑤ 多个过滤条件，例如满足http以及且端口为80的过滤规则：http && tcp.port == 80

组合符号有“且” and (&&), “或” or (||), “取反” not (!) 这几种

1.1 观察MAC帧格式

点击Ethernet II展开，查看MAC帧（图中标示的为“类型”字段）



The image shows a Wireshark packet capture window. The top pane displays the packet details for Frame 703. The Ethernet II section is expanded, showing the destination and source MAC addresses, their bit fields (LG and IG bits), and the type field, which is highlighted with a red box and labeled 'Type: IPv4 (0x0800)'. The bottom pane shows the raw packet data in hexadecimal and ASCII format. The first 14 bytes of the packet are highlighted in blue, corresponding to the destination MAC address.

```
▶ Frame 703: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
└─ Ethernet II, Src: Vmware_bd:21:20 (00:0c:29:bd:21:20), Dst: Cisco_73:75:c1 (88:1d:fc:73:75:c1)
  └─ Destination: Cisco_73:75:c1 (88:1d:fc:73:75:c1)
    Address: Cisco_73:75:c1 (88:1d:fc:73:75:c1)
    ....0. .... = LG bit: Globally unique address (factory default)
    ....0. .... = IG bit: Individual address (unicast)
  └─ Source: Vmware_bd:21:20 (00:0c:29:bd:21:20)
    Address: Vmware_bd:21:20 (00:0c:29:bd:21:20)
    ....0. .... = LG bit: Globally unique address (factory default)
    ....0. .... = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800)
▶ Internet Protocol Version 4, Src: 10.24.90.172, Dst: 10.24.90.1
└─ Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0x4d56 [correct]
  [Checksum Status: Good]

0000  88 1d fc 73 75 c1 00 0c 29 bd 21 20 08 00 45 00  ...su... )!.E.
0010  00 3c 03 49 00 00 80 01 00 00 0a 18 5a ac 0a 18  -<.I....Z...
0020  5a 01 08 00 4d 56 00 01 00 05 61 62 63 64 65 66  Z...MV..abcdef
0030  67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76  ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67 68 69                    wabcdefg hi
```

1.1实验报告：图文结合，说明MAC帧各个字段的含义、解读EUI-48信息。

1.2 观察IP数据报的首部结构

点击Internet Protocol Version 4展开，查看IP数据报结构

```

▶ Frame 703: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▶ Ethernet II, Src: Vmware_bd:21:20 (00:0c:29:bd:21:20), Dst: Cisco_73:75:c1 (88:1d:fc:73:75:c1)
▶ Internet Protocol Version 4, Src: 10.24.90.172, Dst: 10.24.90.1
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 60
  Identification: 0x0349 (841)
  ▶ Flags: 0x0000
  Time to live: 128
  Protocol: ICMP (1)
  Header checksum: 0x0000 [validation disabled]
  [Header checksum status: Unverified]
  Source: 10.24.90.172
  Destination: 10.24.90.1
▶ Internet Control Message Protocol

```

0000	88 1d fc 73 75 c1 00 0c 29 bd 21 20 08 00 45 00	...su...)! ..E.
0010	00 3c 03 49 00 00 80 01 00 00 0a 18 5a ac 0a 18	<.I....Z...
0020	5a 01 08 00 4d 56 00 01 00 05 61 62 63 64 65 66	Z...MV... ..abcdef
0030	67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76	ghijklmn opqrstuv
0040	77 61 62 63 64 65 66 67 68 69	wabcdefg hi

观察IP数据报的首部结构

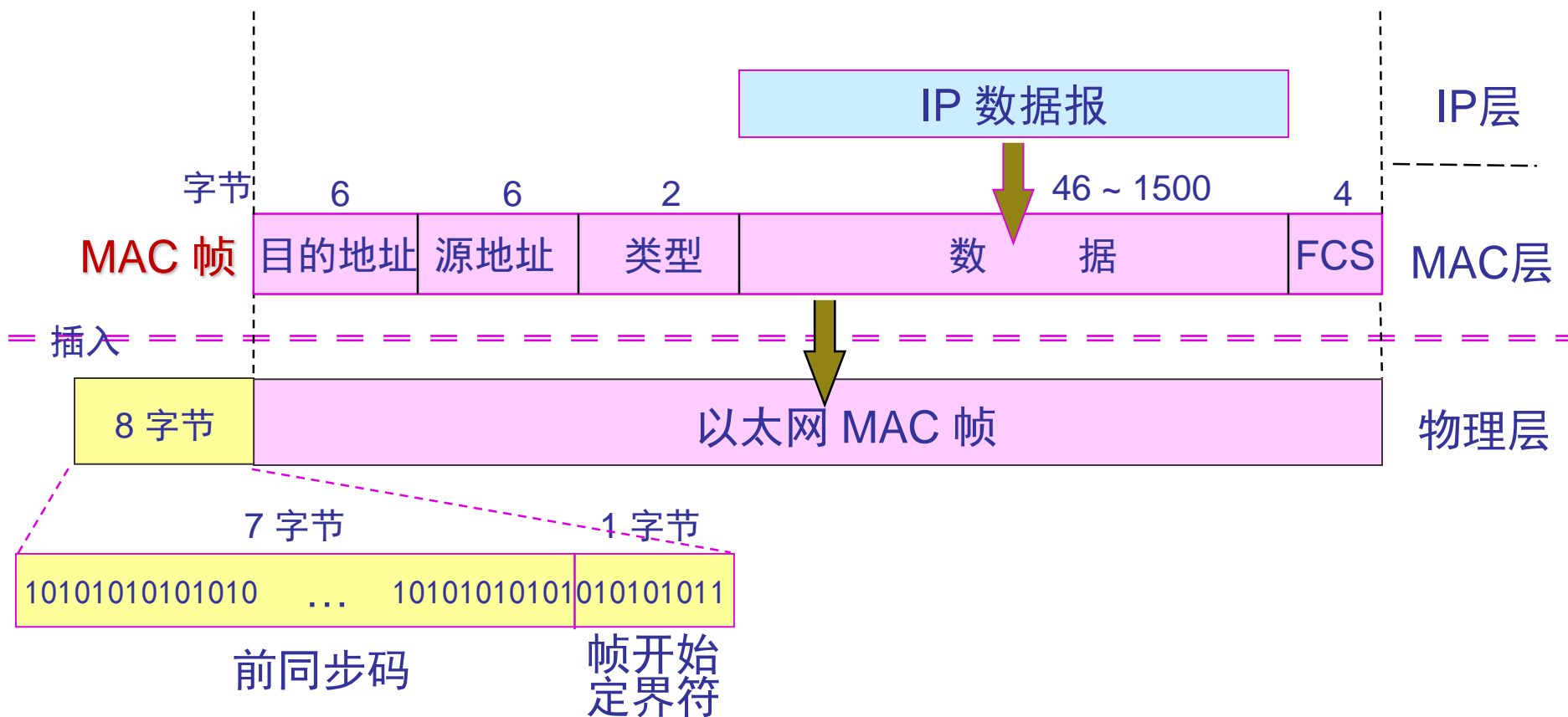
点击Internet Protocol Version 6展开，查看IP数据报结构

```
▶ Frame 88: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
▶ Ethernet II, Src: Vmware_f0:18:7f (00:0c:29:f0:18:7f), Dst: IPv6mcast_ff:ef:72:89 (33:33:ff:ef:72:89)
▼ Internet Protocol Version 6, Src: fe80::ec6d:251e:1932:e4d5 (fe80::ec6d:251e:1932:e4d5), Dst: ff02::1:ffef:72:89 (ff02::1:ffef:72:89)
  ▶ 0110 .... = Version: 6
  ▶ .... 0000 0000 .... = Traffic class: 0x00000000
    .... 0000 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 32
    Next header: ICMPv6 (58)
    Hop limit: 255
    Source: fe80::ec6d:251e:1932:e4d5 (fe80::ec6d:251e:1932:e4d5)
    Destination: ff02::1:ffef:7289 (ff02::1:ffef:7289)
▶ Internet Control Message Protocol v6
```

0000	33 33 ff ef 72 89 00 0c 29 f0 18 7f 86 dd 60 00	33..r...).....
0010	00 00 00 20 3a ff fe 80 00 00 00 00 00 00 ec 6d	... :...m
0020	25 1e 19 32 e4 d5 ff 02 00 00 00 00 00 00 00 00	%..2.....
0030	00 01 ff ef 72 89 87 00 6c d5 00 00 00 00 fe 80r.. l.....
0040	00 00 00 00 00 00 88 75 87 51 c7 ef 72 89 01 01u .Q..r...
0050	00 0c 29 f0 18 7f	..)...

1.2实验报告：图文结合，说明IPv4数据报首部各个字段的含义、帧长与数据报长的关系；并且与IPv6数据报作简单对比。

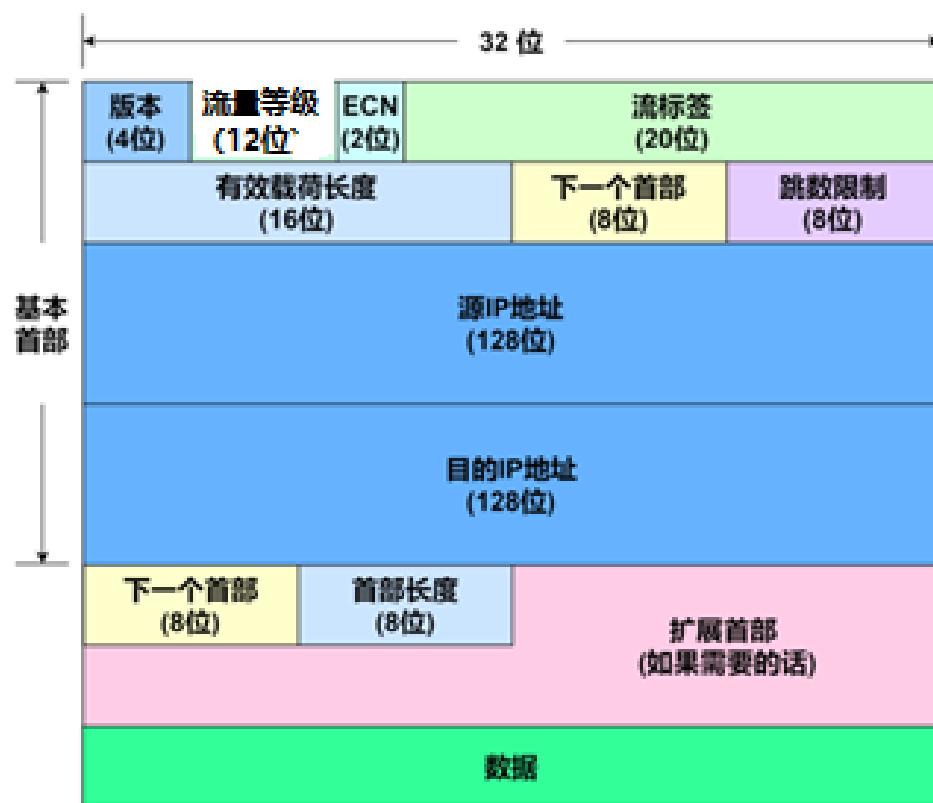
附录1：以太网MAC帧格式



附录2：IPv4数据报的格式



附录3：IPv6数据报的格式



1.3 IP数据报分片

给 `www.xmu.edu.cn` 发送以下 `ping` 命令，请比较命令的结果，并用 Wireshark 软件进行观察分析。

(a) `ping -4 www.xmu.edu.cn`

(b) `ping www.xmu.edu.cn -l 1472 -f -n 1`

(c) `ping www.xmu.edu.cn -l 1473 -f -n 1`

(d) `ping www.xmu.edu.cn -l 1473 -n 1`

1.3实验报告：图文结合，比较终端窗口与抓包数据，验证发送的字节数量。比较个命令的结果。

示例

启动抓包，执行 `ping 210.34.0.2 -l 1472 -f -n 1`，停止抓包，保存

+

Frame 277: 1414 bytes on wire (11312 bits), 1414 bytes captured (11312 bits)

[-]

Ethernet II, Src: 70:1c:e7:74:52:e8 (70:1c:e7:74:52:e8), Dst: b8:8e:82:df:87:be (b8:8e:82:df:87:be)

[-]

Destination: b8:8e:82:df:87:be (b8:8e:82:df:87:be)
Address: b8:8e:82:df:87:be (b8:8e:82:df:87:be)
.....0..... = IG bit: Individual address (unicast)
.....0..... = LG bit: Globally unique address (factory default)

[-]

Source: 70:1c:e7:74:52:e8 (70:1c:e7:74:52:e8)
Address: 70:1c:e7:74:52:e8 (70:1c:e7:74:52:e8)
.....0..... = IG bit: Individual address (unicast)
.....0..... = LG bit: Globally unique address (factory default)

Type: IP (0x0800)

[-]

Internet Protocol, Src: 192.168.3.32 (192.168.3.32), Dst: 210.34.0.2 (210.34.0.2)
Version: 4
Header length: 20 bytes
+ Differentiated Services Field: 0x00 (DSCP
Total Length: 1400
Identification: 0x76f7 (30455)
+ Flags: 0x02 (Don't Fragment)
Fragment offset: 0
Time to live: 64
Protocol: ICMP (1)

C:\Users\William>ping 210.34.0.2 -l 1372 -f -n 1

正在 Ping 210.34.0.2 具有 1372 字节的数据:
来自 210.34.0.2 的回复: 字节=1372 时间=6ms TTL=57

210.34.0.2 的 Ping 统计信息:
数据包: 已发送 = 1, 已接收 = 1, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
最短 = 6ms, 最长 = 6ms, 平均 = 6ms

0020 00 02 08 00 89 76 00 01 00 1e 61 62 63 64 65 66V.. ..abcdef

0030 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 ghijklmn opqrstuv

0040 77 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f wabcdefgh hijklmno

0050 70 71 72 73 74 75 76 77 61 62 63 64 65 66 67 68 pqrstuvwxyz abcdefgh

0060 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77 61 ijklmnop qrstuvw

0070 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 bcdefghi jklmnop

Data (data.data), 1372 bytes

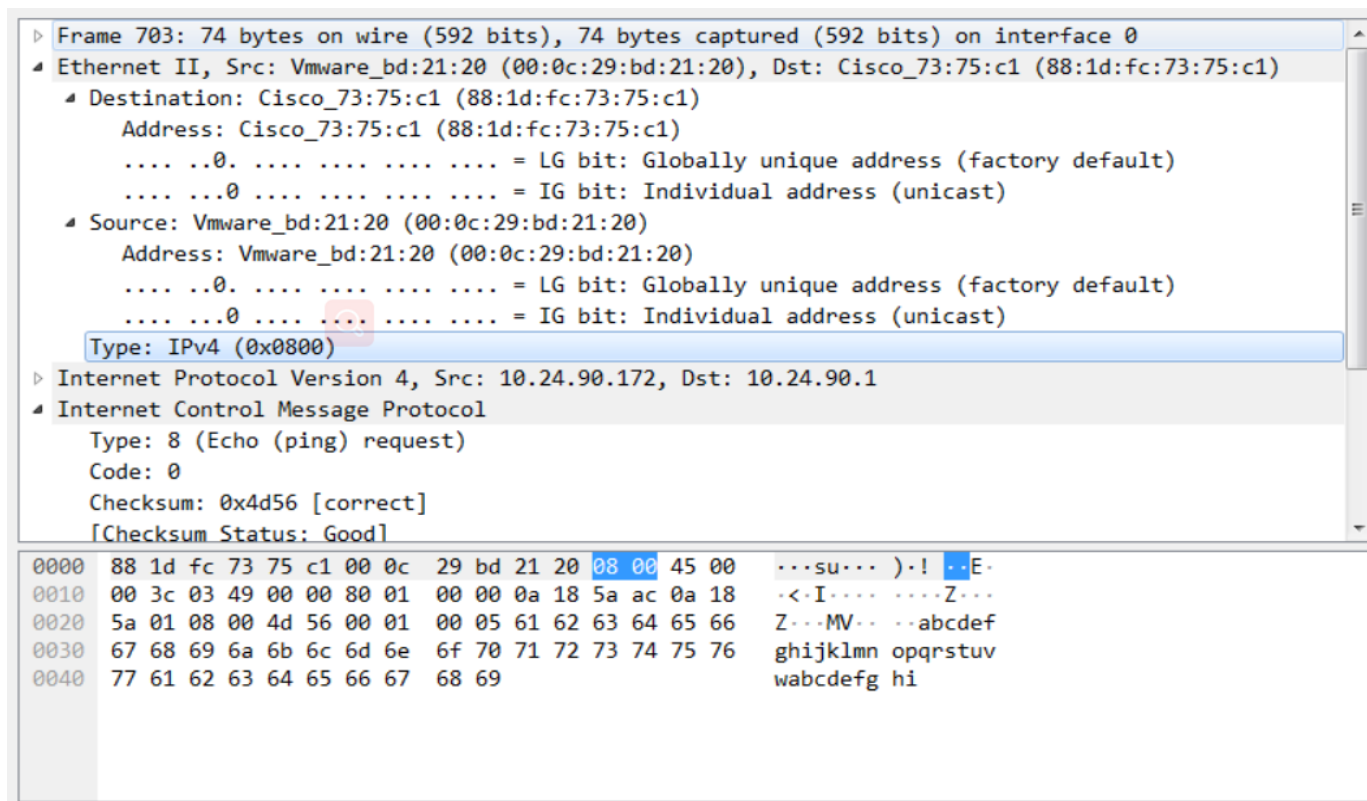
Packets: 131586 Displayed: 8 ...

F

▶ 28

1.4 解释ICMP报文

点击Internet Control Message Protocol展开，查看ICMP报文



1.4实验报告：图文结合，说明执行一次ping命令，共有几个ICMP请求帧和回应帧，比较请求帧和回应帧的差别及其对应IP头部的变化。

1.5 tracer命令

**tracert [-d] [-h maximum_hops] [-j host-list] [-w timeout] [-R]
[-S srcaddr] [-4] [-6] target_name**

-d : 将地址解析成主机名

-h maximum_hops: 搜索目标的最大跃点数

-j host-list: 与主机列表一起的松散源路由，用于 IPv4

-w timeout: 等待每个回复的超时时间（单位：毫秒）

-R : 跟踪往返行程路径，用于 IPv6

-S srcaddr: 要使用的源地址，用于 IPv6

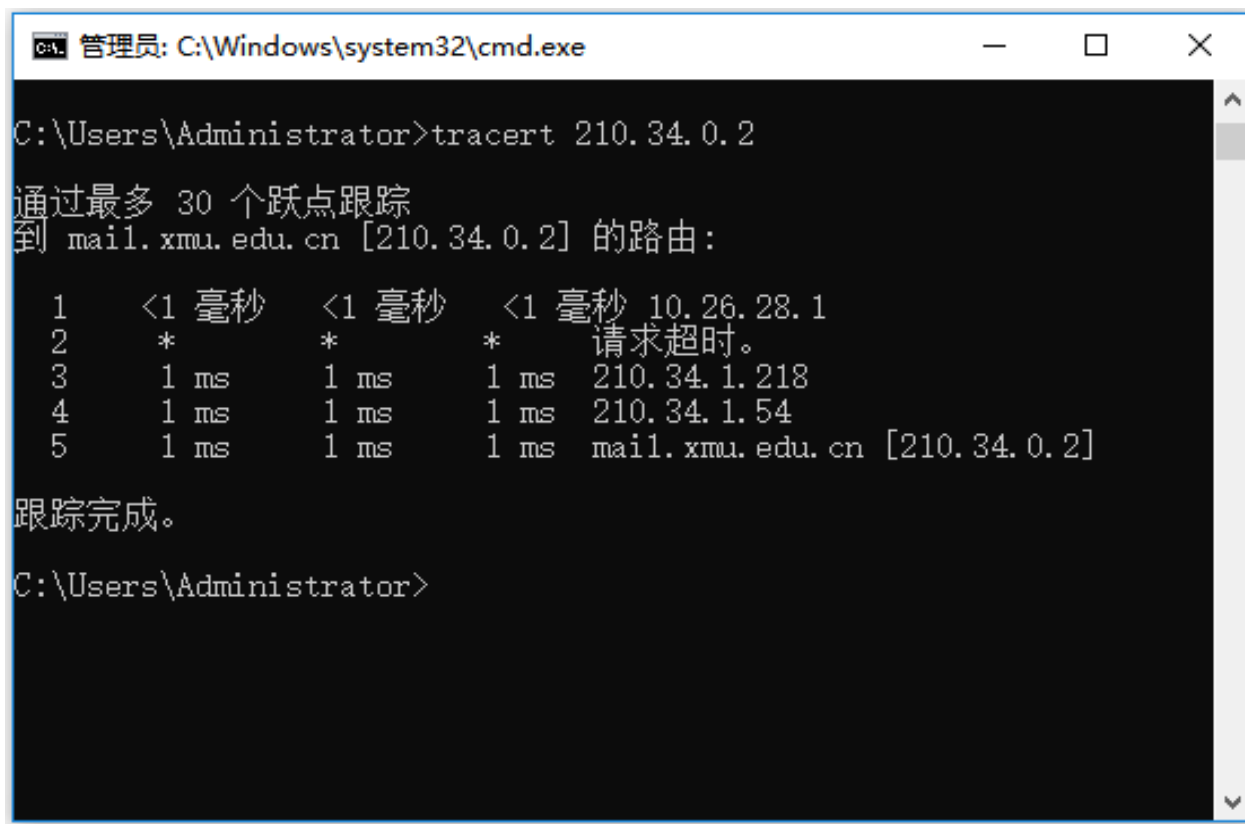
-4 : 强制使用 IPv4

-6 : 强制使用 IPv6

target_name: 指定目标，可以是 IP 地址或主机名

tracert 示例

抓包，在终端发起网络命令：`tracert` IP/域名，停止抓包，保存



```
C:\Windows\system32\cmd.exe

C:\Users\Administrator>tracert 210.34.0.2

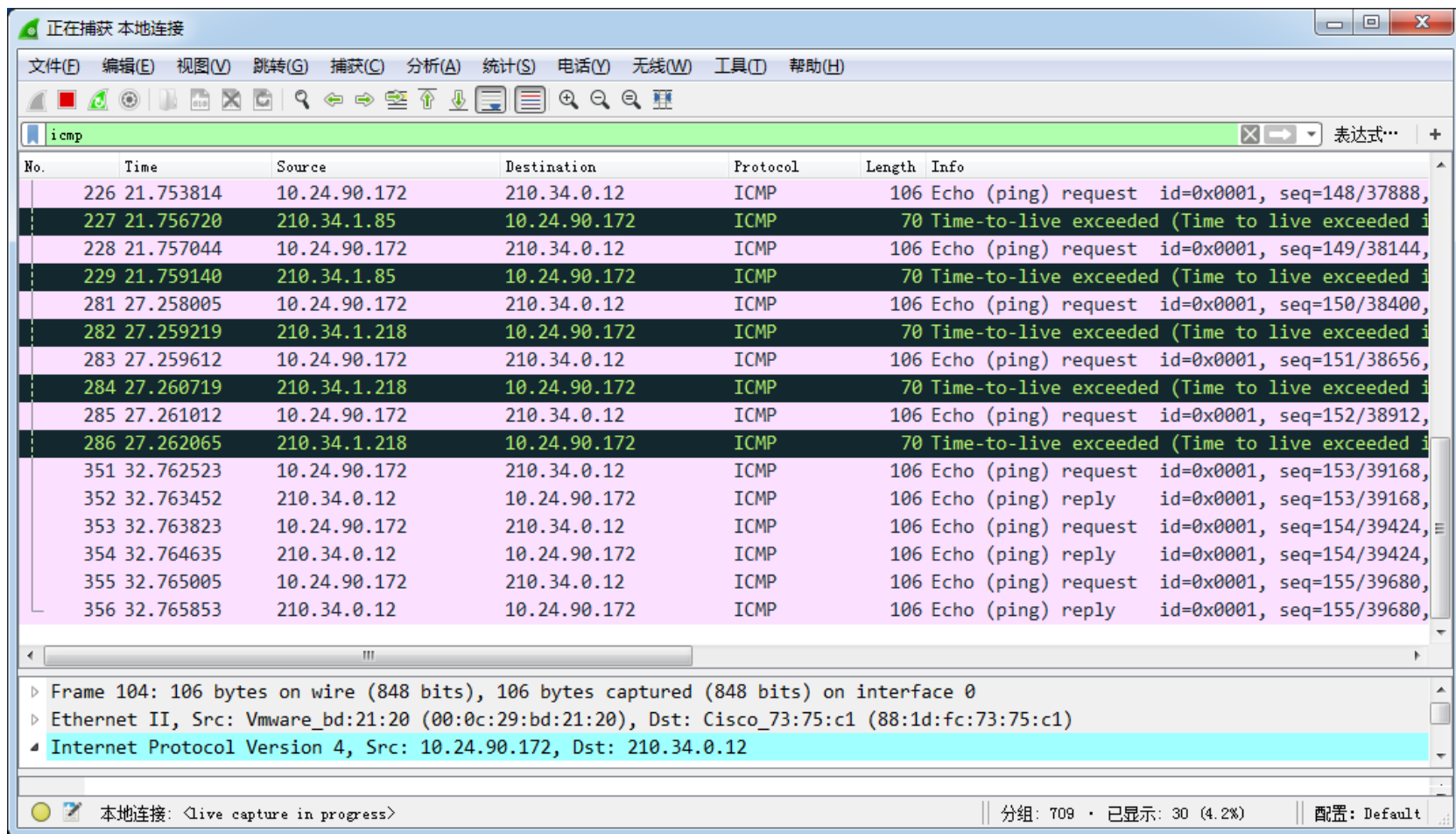
通过最多 30 个跃点跟踪
到 mail.xmu.edu.cn [210.34.0.2] 的路由:

  1    <1 毫秒    <1 毫秒    <1 毫秒  10.26.28.1
  2    *          *          *          请求超时。
  3    1 ms       1 ms       1 ms     210.34.1.218
  4    1 ms       1 ms       1 ms     210.34.1.54
  5    1 ms       1 ms       1 ms     mail.xmu.edu.cn [210.34.0.2]

跟踪完成。

C:\Users\Administrator>
```

设置过滤条件icmp，显示所捕获的ICMP数据包：



1.5实验报告：图文结合，描述tracert工作原理，画出示意图。



1.6 ARP协议分析

实验环境：

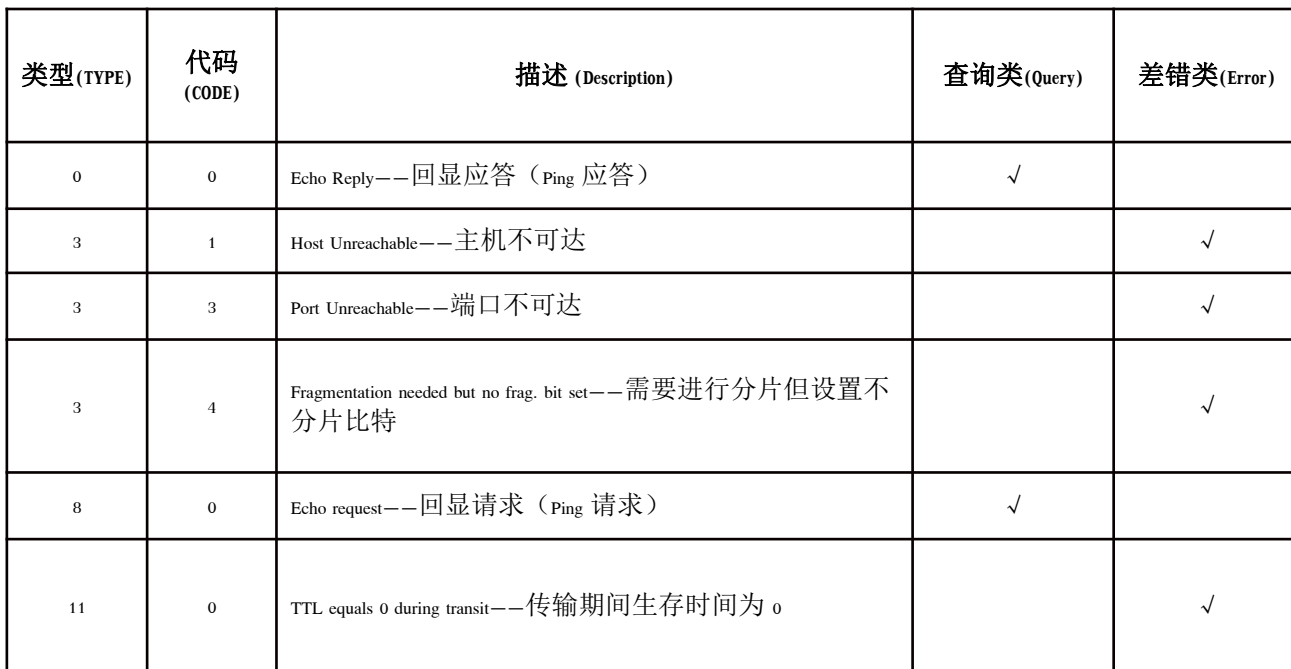
两个以上同学使用电脑连接同一个手机热点或宿舍wifi路由器，保证处在同一个局域网内，记录下各自的ip地址

步骤：

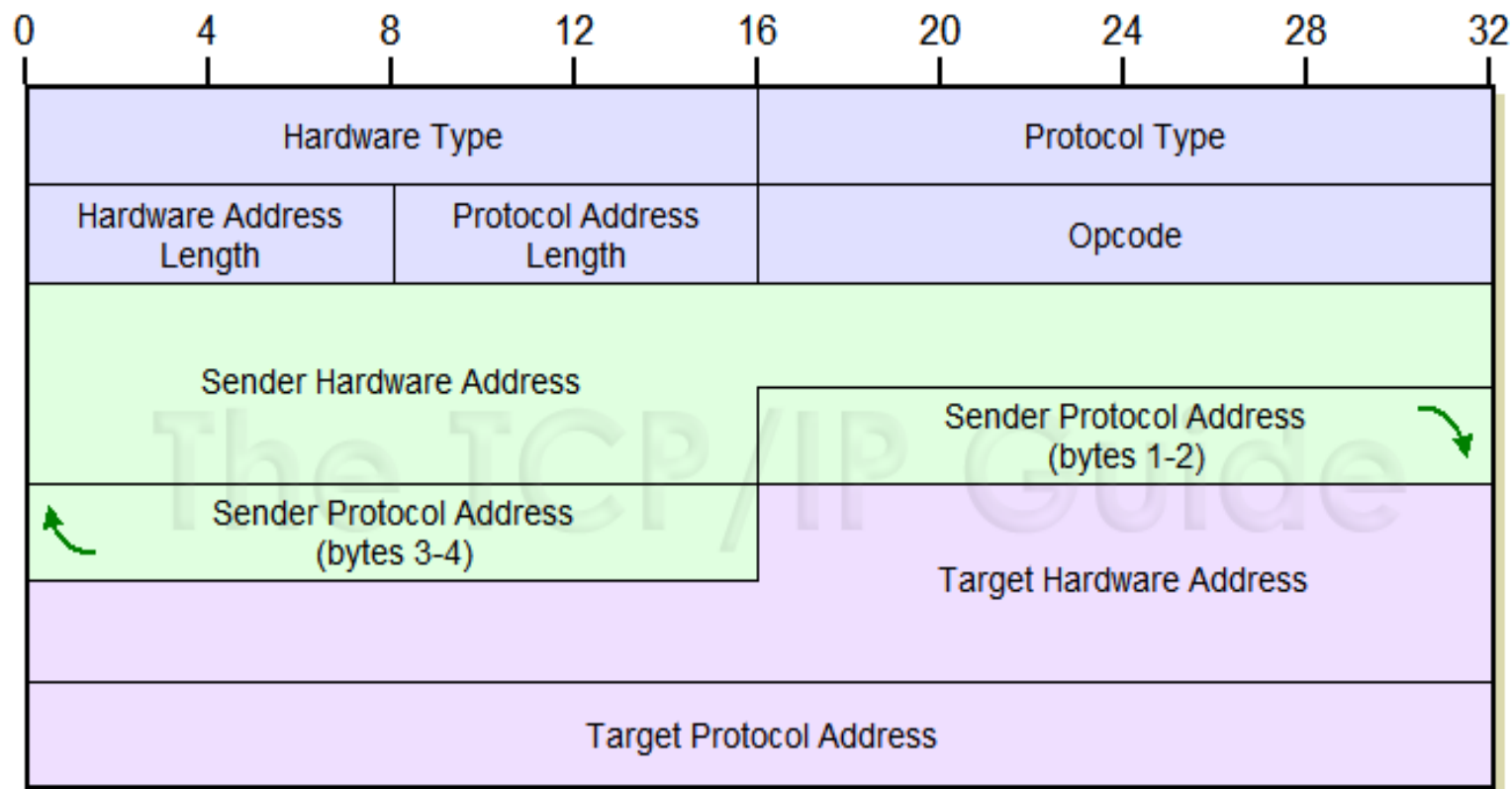
1. 以管理员身份运行终端窗口，运行`arp -d`命令，清空本机已有的ARP缓存；
2. 启动wireshark抓包，`ping`旁边同学的ip；
3. 重复`arp -d`命令，`ping www.baidu.com`；
4. 停止抓包、保存数据，同时过滤出arp和icmp报文。

1.6实验报告：图文结合，解释ARP报文（请求/响应）每个字段的含义；
`ping`局域网内的主机和局域网外的主机，产生的ARP请求有何不同？结合课本上的工作原理进行说明。





附录4：ARP报文的格式



任务2：捕获和分析802.11数据

➤ 实验室环境：

USB无线网卡 + Ubuntu22.04系统，捕获无线数据包

1. 在D盘或E盘，找到Ubuntu22.04.rar并将其解压缩，
若不存在则从网盘下载：

box.xmu.edu.cn/share/b8b13a69f27b9996843eb7b527

2. 在解压后的文件夹，找到Ubuntu22.04.vmx 双击打开

也可以尝试其他系统：

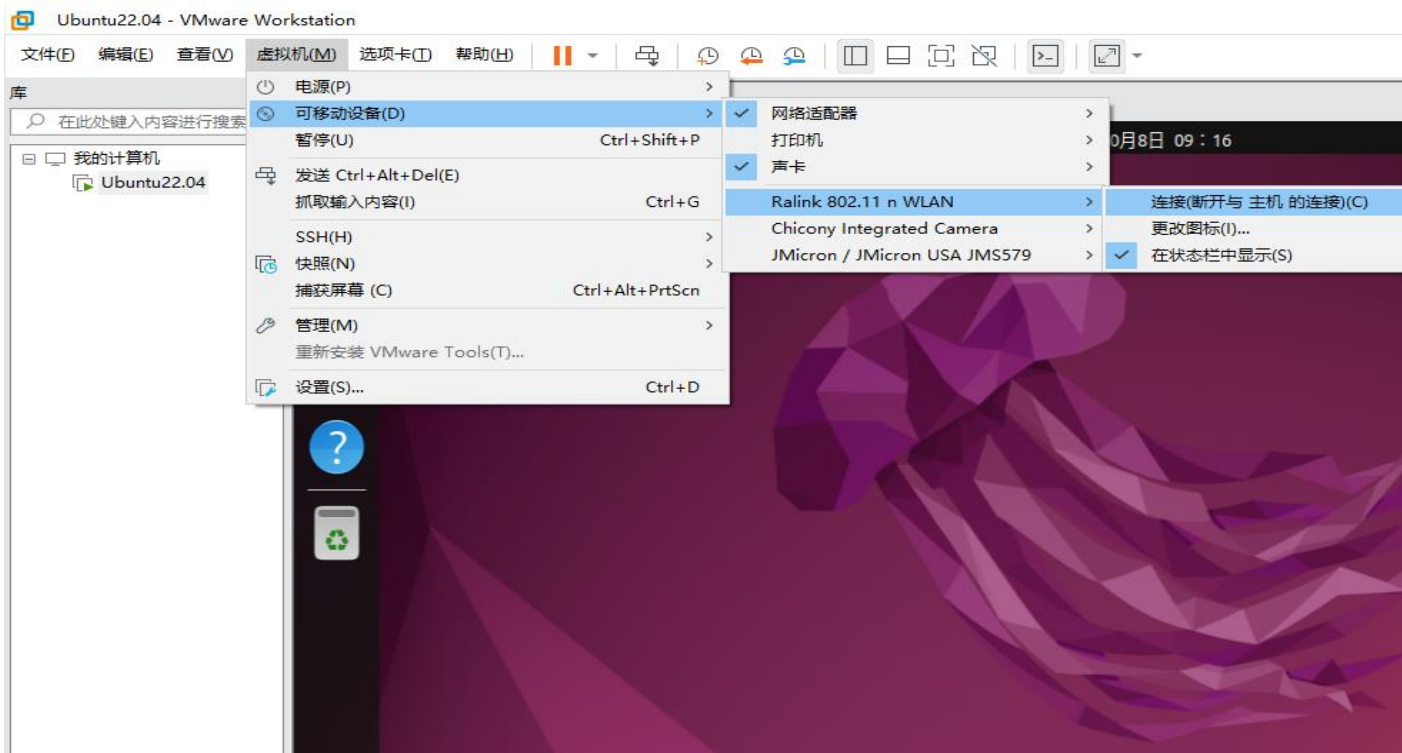
- Mac系统
- 其他版本Linux

➤ 实验报告

过滤出802.11控制帧、数据帧、管理帧，并进行分析。

构建无线环境(实验室)

- 启动VMware里的Linux系统，将USB无线网卡连接到电脑主机
- 更改菜单设置，使网卡连接到虚拟机上



打开终端，输入`ifconfig`或`ip addr`，
会看到多了一个`wlx0c826831a179`

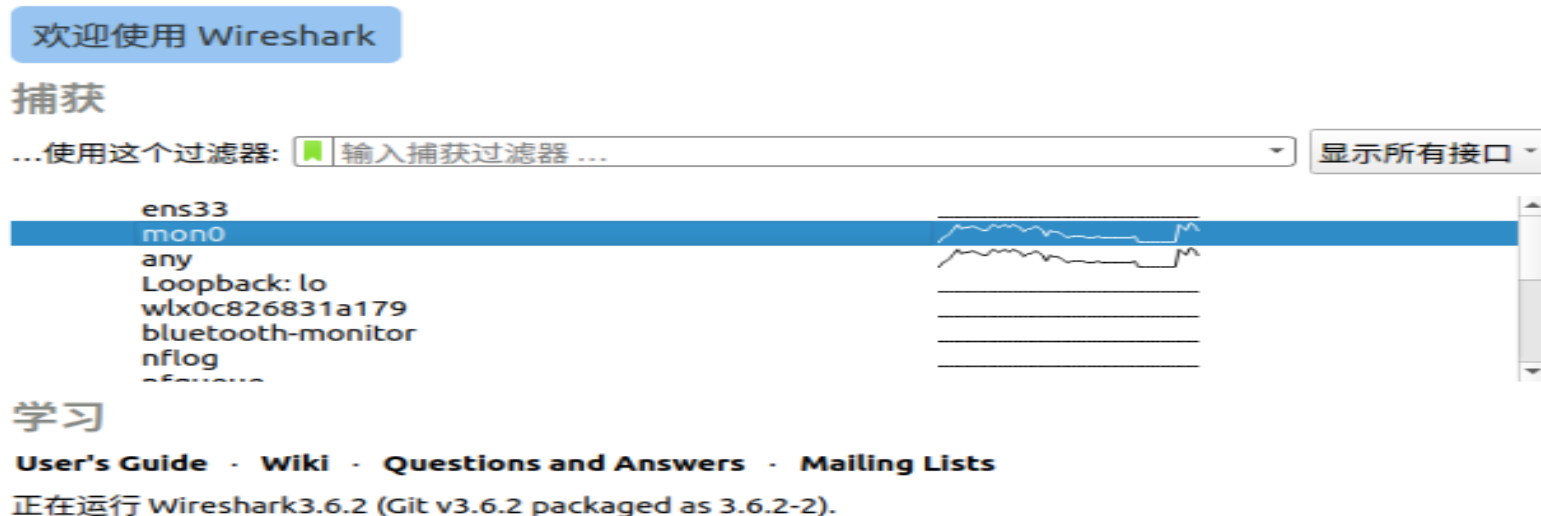
```
linux@Ubuntu22:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 1000
    link/ether 00:0c:29:29:3e:fc brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.237.128/24 brd 192.168.237.255 scope global dynamic noprefixroute ens33
        valid_lft 1594sec preferred_lft 1594sec
    inet6 fe80::73bc:4f66:f883:1aed/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
5: wlx0c826831a179: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 0c:82:68:31:a1:79 brd ff:ff:ff:ff:ff:ff
linux@Ubuntu22:~$
```

无线网卡配置

- 安装iw命令: `sudo apt install iw`
- 新建一个虚拟网卡mon0, 并将其修改为监听模式

```
linux@Ubuntu22:~$ sudo iw wlx0c826831a179 interface add mon0 type monitor
linux@Ubuntu22:~$ sudo ifconfig mon0 up
```

- 安装wireshark: `sudo apt install wireshark`
- 打开Wireshark软件: `sudo wireshark`



捕获和分析802.11数据

➤ 双击mon0，即可捕获到802.11数据

The screenshot shows the Wireshark network protocol analyzer interface. The top menu bar includes File (F), Edit (E), View (V), Jump (G), Capture (C), Analyze (A), Statistics (S), Phone (Y), Wireless (W), Tools (T), and Help (H). Below the menu is a toolbar with various icons for file operations, capture control, and analysis. The main display area is divided into three panes. The top pane shows the packet list with columns for No., Time, Source, Destination, Protocol, Length, and Info. The middle pane shows the packet details for the selected packet (No. 17). The bottom pane shows the packet bytes.

No.	Time	Source	Destination	Protocol	Length	Info
17	10.408571240	1e:bb:b1:a5:fb:f0	Broadcast	802.11	86	Probe Request, SN=149, FN=0, Flags=....., SSID=Wildcard (Broadcast)
18	10.827886710	1e:bb:b1:a5:fb:f0	Broadcast	802.11	86	Probe Request, SN=150, FN=0, Flags=....., SSID=Wildcard (Broadcast)
19	11.250469047	1e:bb:b1:a5:fb:f0	Broadcast	802.11	86	Probe Request, SN=151, FN=0, Flags=....., SSID=Wildcard (Broadcast)
20	11.670043841	1e:bb:b1:a5:fb:f0	Broadcast	802.11	86	Probe Request, SN=152, FN=0, Flags=....., SSID=Wildcard (Broadcast)
21	12.092591217	1e:bb:b1:a5:fb:f0	Broadcast	802.11	86	Probe Request, SN=153, FN=0, Flags=....., SSID=Wildcard (Broadcast)
22	12.515415242	1e:bb:b1:a5:fb:f0	Broadcast	802.11	86	Probe Request, SN=154, FN=0, Flags=....., SSID=Wildcard (Broadcast)
23	14.693809715	1e:bb:b1:a5:fb:f0	Broadcast	802.11	86	Probe Request, SN=155, FN=0, Flags=....., SSID=Wildcard (Broadcast)
24	15.098920905	1e:bb:b1:a5:fb:f0	Broadcast	802.11	86	Probe Request, SN=156, FN=0, Flags=....., SSID=Wildcard (Broadcast)
25	15.503549453	1e:bb:b1:a5:fb:f0	Broadcast	802.11	86	Probe Request, SN=157, FN=0, Flags=....., SSID=Wildcard (Broadcast)
26	15.914178497	1e:bb:b1:a5:fb:f0	Broadcast	802.11	86	Probe Request, SN=158, FN=0, Flags=....., SSID=Wildcard (Broadcast)
27	16.290331682	1e:bb:b1:a5:fb:f0	Broadcast	802.11	86	Probe Request, SN=159, FN=0, Flags=....., SSID=Wildcard (Broadcast)
28	16.654501985	1e:bb:b1:a5:fb:f0	Broadcast	802.11	86	Probe Request, SN=160, FN=0, Flags=....., SSID=Wildcard (Broadcast)
29	17.016040824	1e:bb:b1:a5:fb:f0	Broadcast	802.11	86	Probe Request, SN=161, FN=0, Flags=....., SSID=Wildcard (Broadcast)
30	17.388729142	1e:bb:b1:a5:fb:f0	Broadcast	802.11	86	Probe Request, SN=162, FN=0, Flags=....., SSID=Wildcard (Broadcast)
31	17.753498594	1e:bb:b1:a5:fb:f0	Broadcast	802.11	86	Probe Request, SN=163, FN=0, Flags=....., SSID=Wildcard (Broadcast)
32	18.117674698	1e:bb:b1:a5:fb:f0	Broadcast	802.11	86	Probe Request, SN=164, FN=0, Flags=....., SSID=Wildcard (Broadcast)
33	18.480406888	1e:bb:b1:a5:fb:f0	Broadcast	802.11	86	Probe Request, SN=165, FN=0, Flags=....., SSID=Wildcard (Broadcast)

Frame 1: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface mon0, id 0
Radiotap Header v0, Length 13
802.11 radio information
IEEE 802.11 Probe Request, Flags:
IEEE 802.11 Wireless Management

附录5：Wireshark中的无线帧类型和过滤规则对照表

帧类型	过滤器语法
Management frame	wlan.fc.type == 0
Control frame	wlan.fc.type == 1
Data frame	wlan.fc.type == 2
Association request	wlan.fc.type_subtype == 0x00
Association response	wlan.fc.type_subtype == 0x01
Reassociation request	wlan.fc.type_subtype == 0x02
Reassociation response	wlan.fc.type_subtype == 0x03
Probe request	wlan.fc.type_subtype == 0x04
Probe response	wlan.fc.type_subtype == 0x05
Beacon	wlan.fc.type_subtype == 0x08
Disassociate	wlan.fc.type_subtype == 0x0A
Authentication	wlan.fc.type_subtype == 0x0B
Deauthentication	wlan.fc.type_subtype == 0x0C
Action frame	wlan.fc.type_subtype == 0x0D
Block ACK requests	wlan.fc.type_subtype == 0x18
Block ACK	wlan.fc.type_subtype == 0x19
Power save poll	wlan.fc.type_subtype == 0x1A
Request to send	wlan.fc.type_subtype == 0x1B
Clear to send	wlan.fc.type_subtype == 0x1C
ACK	wlan.fc.type_subtype == 0x1D
Contention free period end	wlan.fc.type_subtype == 0x1E
NULL data	wlan.fc.type_subtype == 0x24
QoS data	wlan.fc.type_subtype == 0x28
QoS Null data	wlan.fc.type_subtype == 0x2C

任务3.1：探索Wireshark更丰富的功能（选做）

- 数据流追踪
- 协议分层统计
- 网络节点和会话统计功能
-
- **实验报告建议**
 - 探索Wireshark的更多高级功能，并以截图演示操作过程
 - 对这些功能的演示结果进行分析说明

数据流追踪

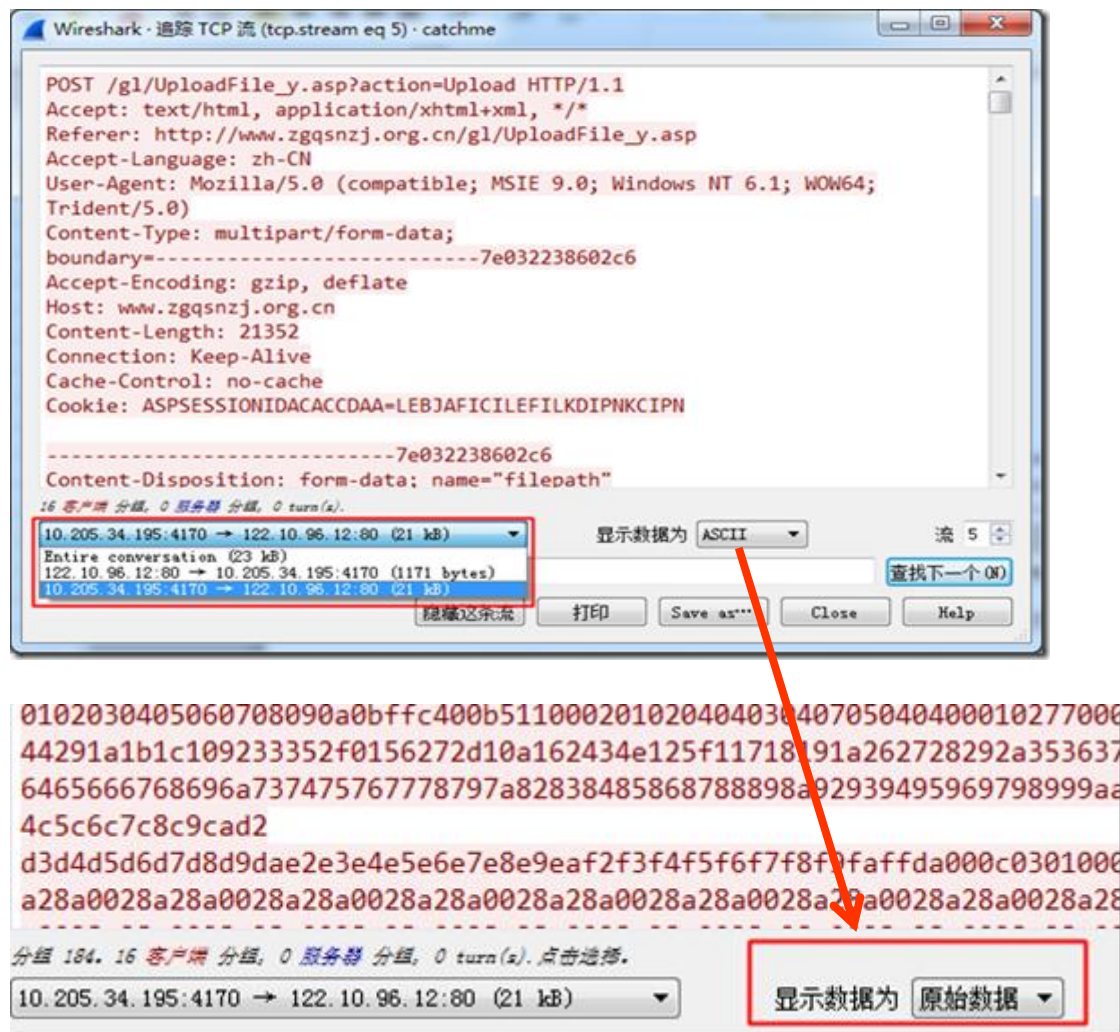
- 可以自己准备一张图片，并随便找一个允许上传的网站，然后用 Wireshark 将上传的过程抓包

```
* [16 Reassembled TCP Segments (21886 bytes): #183(534),  
  [Frame: 183, payload: 0-533 (534 bytes)]  
  [Frame: 184, payload: 534-1993 (1460 bytes)]  
  [Frame: 185, payload: 1994-3453 (1460 bytes)]  
  [Frame: 186, payload: 3454-4913 (1460 bytes)]  
  [Frame: 187, payload: 4914-6373 (1460 bytes)]  
  [Frame: 188, payload: 6374-7833 (1460 bytes)]  
  [Frame: 196, payload: 7834-9293 (1460 bytes)]  
  [Frame: 197, payload: 9294-10753 (1460 bytes)]  
  [Frame: 198, payload: 10754-12213 (1460 bytes)]  
  [Frame: 203, payload: 12214-13673 (1460 bytes)]  
  [Frame: 204, payload: 13674-15133 (1460 bytes)]  
  [Frame: 205, payload: 15134-16593 (1460 bytes)]  
  [Frame: 206, payload: 16594-18053 (1460 bytes)]  
  [Frame: 207, payload: 18054-19513 (1460 bytes)]  
  [Frame: 208, payload: 19514-20973 (1460 bytes)]  
  [Frame: 209, payload: 20974-21885 (912 bytes)]  
  [Segment count: 16]
```

文件较大时，TCP协议会将其分成若干个数据段（Segment），每个数据段都是一个独立的IP数据包。

请看看“数据流追踪”功能可以做什么？

数据流追踪



协议分层统计

- 有时你需要分析捕获文件中协议的分布情况，比如获知TCP协议的百分比，UDP的百分比，使用 Wireshark 的“协议分级”就可以实现。

协议	按分组百分比	分组	按字节百分比	字节	比特/秒	结束 分组	结束 字节
▼ Frame	100.0	702	100.0	741772	585 k	0	0
▼ Ethernet	100.0	702	1.3	9828	7763	0	0
▼ Internet Protocol Version 4	99.7	700	1.9	14000	11 k	0	0
▼ User Datagram Protocol	20.8	146	0.2	1168	922	0	0
Session Traversal Utilities for NAT	2.3	16	0.2	1296	1023	16	1296
OICQ - IM software, popular in China	1.0	7	0.1	1057	834	7	1057
Datagram Transport Layer Security	16.7	117	12.2	90541	71 k	117	90541
Data	0.9	6	0.1	611	482	6	611
▼ Transmission Control Protocol	78.9	554	84.0	623215	492 k	484	555908
Transport Layer Security	12.0	84	82.8	614051	485 k	68	395941
▼ Hypertext Transfer Protocol	0.3	2	1.0	7634	6030	1	7295
JavaScript Object Notation	0.1	1	0.0	99	78	1	99
Address Resolution Protocol	0.3	2	0.0	56	44	2	56

任务3.2：探索其它抓包工具（选做）

➤ 探索其它抓包工具

➤ 实验报告建议

- 提供抓包截图、简单分析
- 比较和Wireshark的使用区别