Wireshark

Dr. Xiqun Lu
College of Computer Science and Technology,
Zhejiang University, Hangzhou

实验目的

- 通过分析各种不同网络协议，加深理解第一堂课中重点：“Protocol layering”
- Network architecture: a set of layers and protocols
- **Protocol stack**: a list of the protocols used by a certain system, one protocol per layer
  - A **protocol** defines the *format* and the *order* of messages exchanged between two or more communication entities, as well as the *actions* taken on the transmission and/or receipt of a message or other event. [2]

The TCP/IP Reference Model (IV)

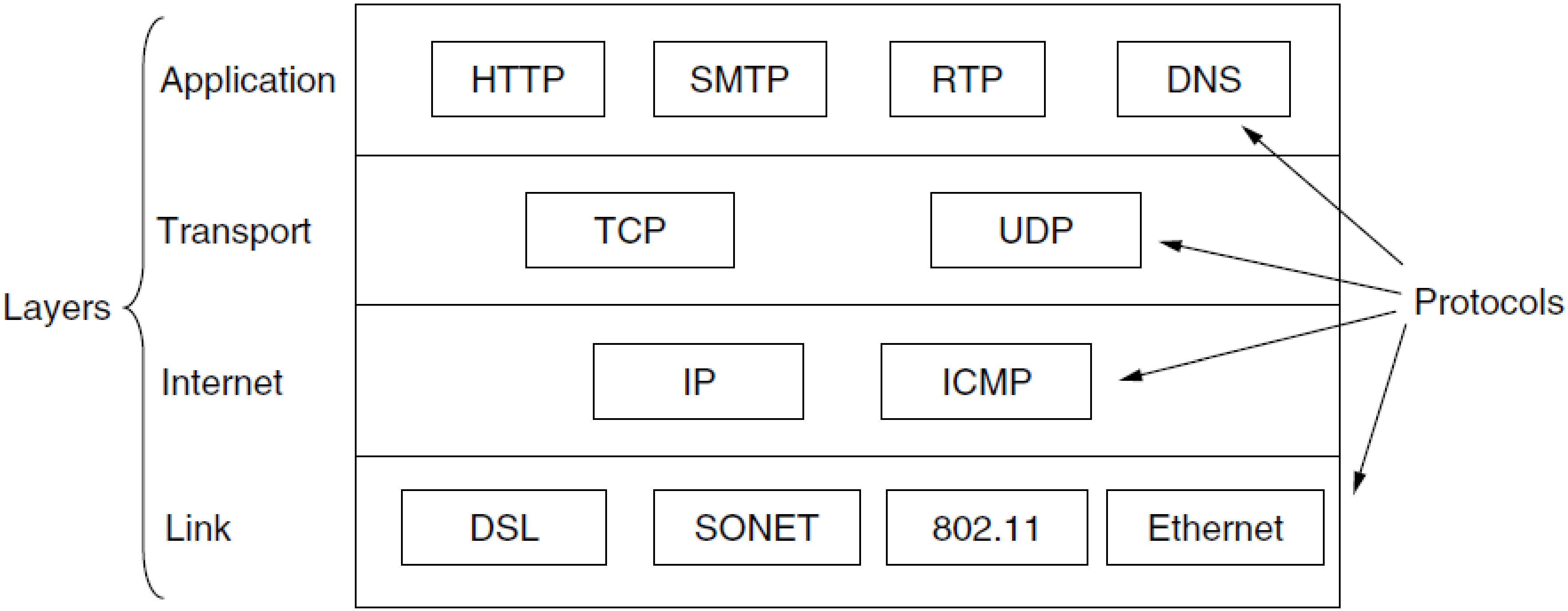
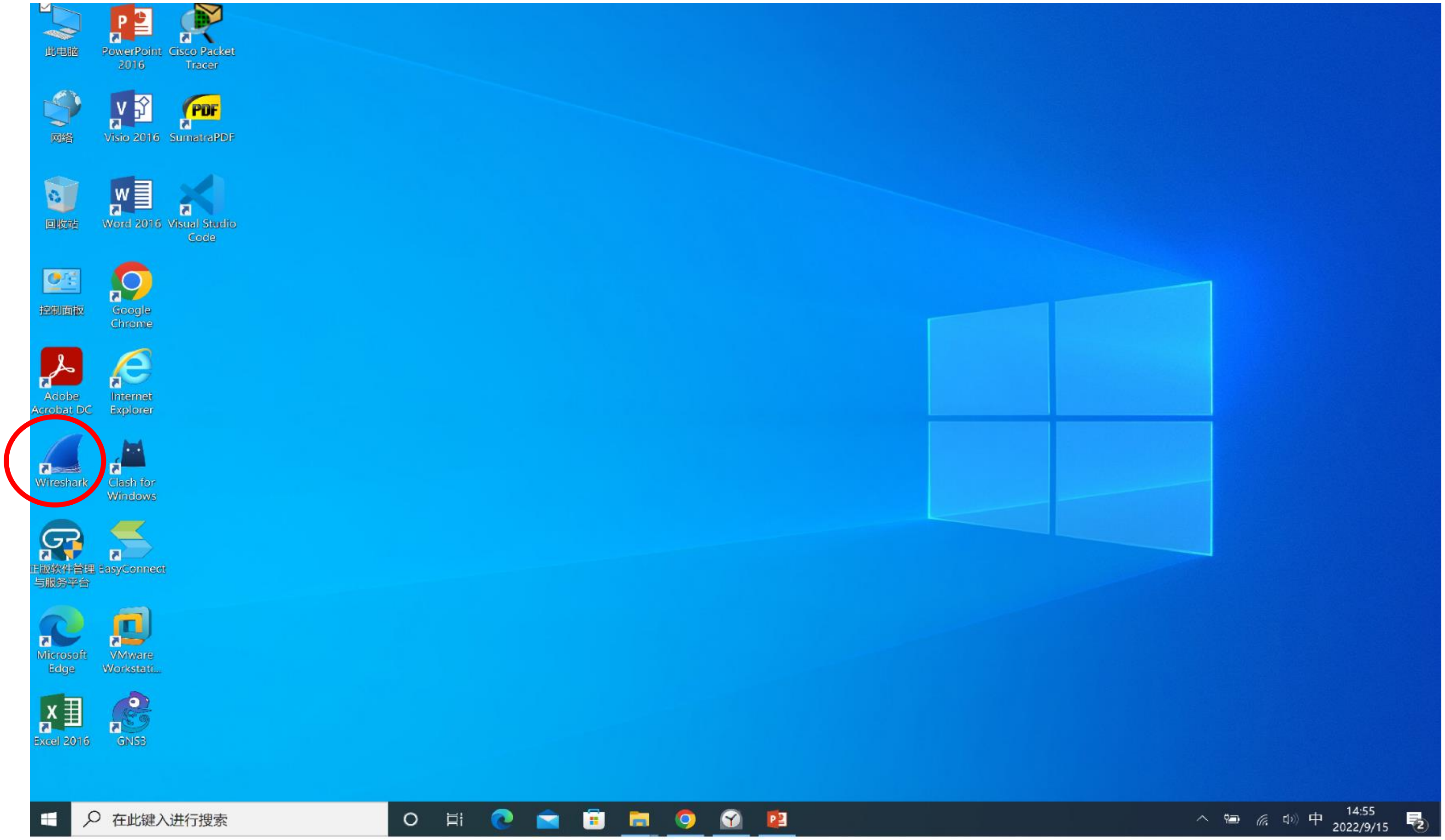


Figure 1-22. The TCP/IP model with some protocols we will study.

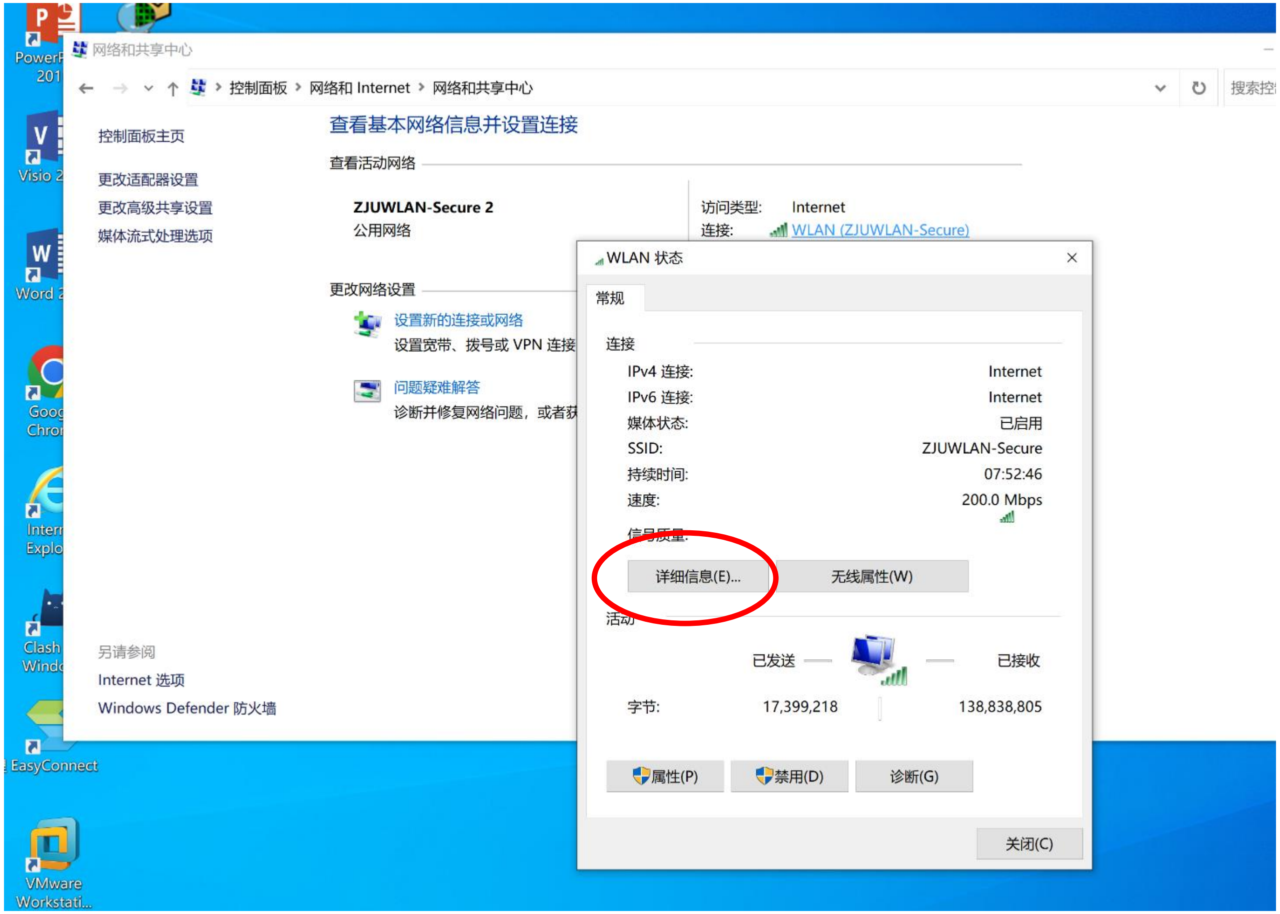
Step 1: WireShark

- 安装WireShark
  - 下载地址： https://www.wireshark.org/
  - 安装成功，桌面上会出现一个“鲨鱼鳍”的图标，如下图红圆圈中图标。



Step 2:了解自己电脑的一些网络一些设置

- 先在“控制面板” 中打开网络中心，然后详细信息里有你电脑当前所使用的IP地址，你也可以看一下你电脑的物理地址（MAC地址）



查看基本网络信息并设置连接

查看活动网络

ZJUWLAN-Secure 2
公用网络

访问类型: Internet
连接: WLAN (ZJUWLAN-Secure)

更改网络设置

设置新的连接或网络
设置宽带、拨号或 VPN 连接

问题疑难解答
诊断并修复网络问题，或者获

WLAN 状态

常规

连接

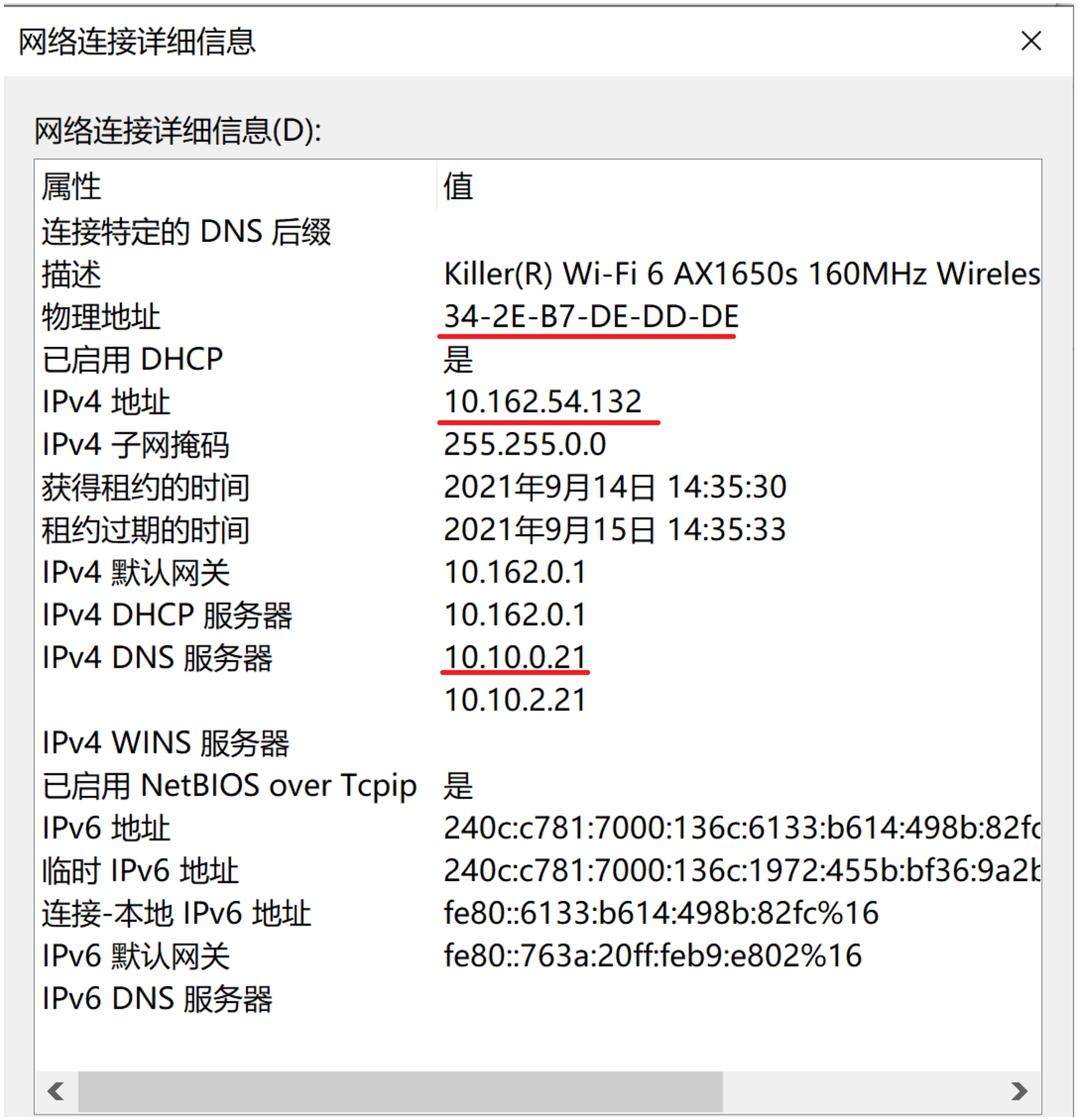
IPv4 连接:	Internet
IPv6 连接:	Internet
媒体状态:	已启用
SSID:	ZJUWLAN-Secure
持续时间:	07:52:46
速度:	200.0 Mbps
信号质量:	

活动

	已发送	已接收
字节:	17,399,218	138,838,805

Step 2:了解自己电脑的一些网络设置

- 一些协议名称
  - DNS
  - DHCP
- 一些地址信息与概念:
  - 子网掩码：255.255.0.0
  - IPv4地址(32bits): 10.162.54.132
  - IPv6地址(128bits): 240c:c781:7000:2d93:6133:b614:498b:82fc
    - IP地址随着使用环境变化而变化
  - 物理地址(MAC, 48bits): 34-2E-B7-DE-DD-DE
    - 如同人的身份证号

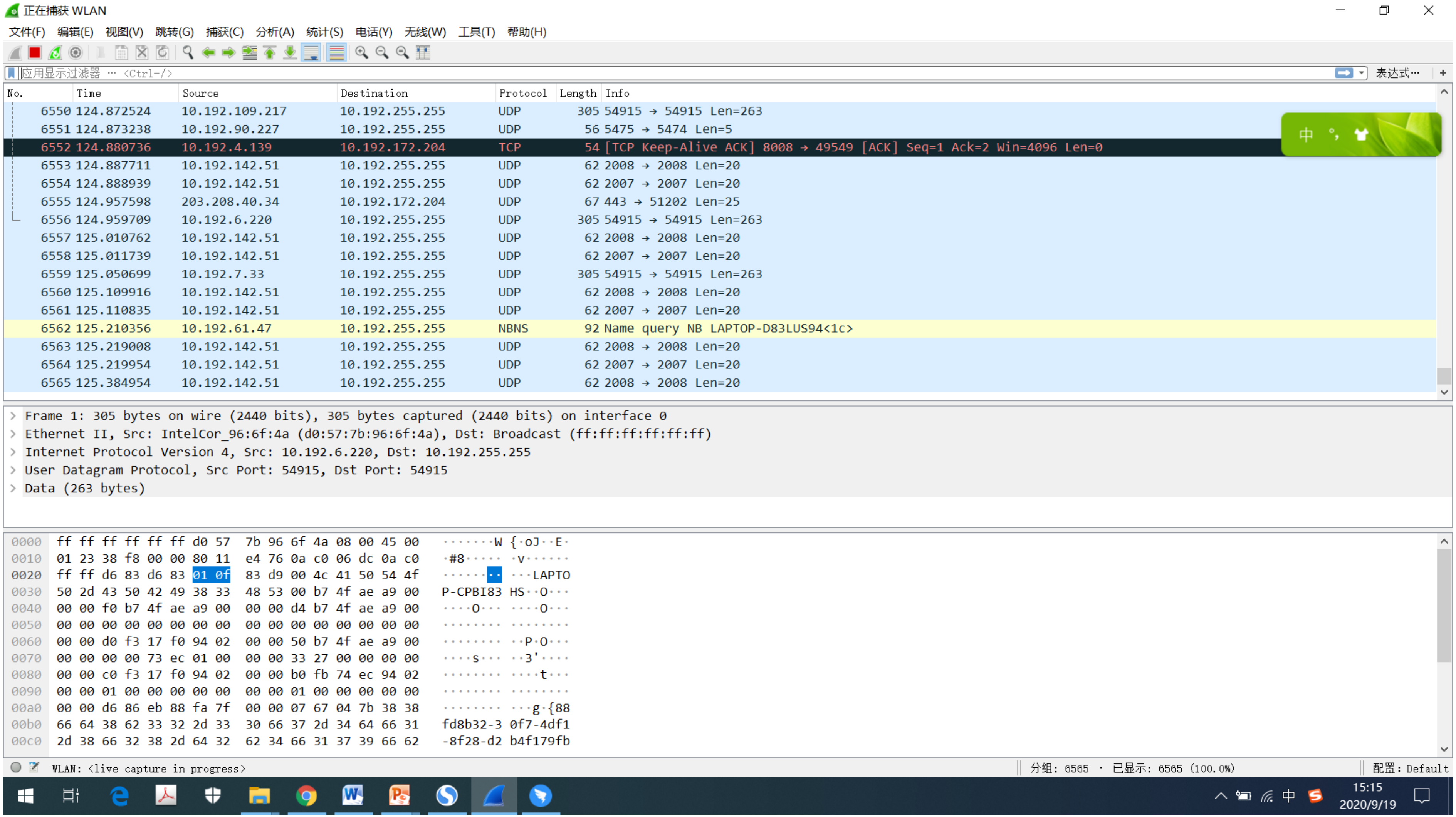


网络连接详细信息

网络连接详细信息(D):

属性	值
连接特定的 DNS 后缀	
描述	Killer(R) Wi-Fi 6 AX1650s 160MHz Wireles
物理地址	34-2E-B7-DE-DD-DE
已启用 DHCP	是
IPv4 地址	10.162.54.132
IPv4 子网掩码	255.255.0.0
获得租约的时间	2021年9月14日 14:35:30
租约过期的时间	2021年9月15日 14:35:33
IPv4 默认网关	10.162.0.1
IPv4 DHCP 服务器	10.162.0.1
IPv4 DNS 服务器	10.10.0.21 10.10.2.21
IPv4 WINS 服务器	
已启用 NetBIOS over Tcpip	是
IPv6 地址	240c:c781:7000:136c:6133:b614:498b:82f
临时 IPv6 地址	240c:c781:7000:136c:1972:455b:bf36:9a2
连接-本地 IPv6 地址	fe80::6133:b614:498b:82fc%16
IPv6 默认网关	fe80::763a:20ff:feb9:e802%16
IPv6 DNS 服务器	

Interface of WireShark


正在捕获 WLAN
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
应用显示过滤器 … <Ctrl-/>
表达式…
+
中 °, 👕

```
No.    Time        Source           Destination      Protocol  Length Info
  6550 124.872524  10.192.109.217   10.192.255.255   UDP        305 54915 → 54915 Len=263
  6551 124.873238  10.192.90.227    10.192.255.255   UDP         56 5475 → 5474 Len=5
  6552 124.880736  10.192.4.139     10.192.172.204   TCP         54 [TCP Keep-Alive ACK] 8008 → 49549 [ACK] Seq=1 Ack=2 Win=4096 Len=0
  6553 124.887711  10.192.142.51    10.192.255.255   UDP         62 2008 → 2008 Len=20
  6554 124.888939  10.192.142.51    10.192.255.255   UDP         62 2007 → 2007 Len=20
  6555 124.957598  203.208.40.34    10.192.172.204   UDP         67 443 → 51202 Len=25
  6556 124.959709  10.192.6.220     10.192.255.255   UDP        305 54915 → 54915 Len=263
  6557 125.010762  10.192.142.51    10.192.255.255   UDP         62 2008 → 2008 Len=20
  6558 125.011739  10.192.142.51    10.192.255.255   UDP         62 2007 → 2007 Len=20
  6559 125.050699  10.192.7.33      10.192.255.255   UDP        305 54915 → 54915 Len=263
  6560 125.109916  10.192.142.51    10.192.255.255   UDP         62 2008 → 2008 Len=20
  6561 125.110835  10.192.142.51    10.192.255.255   UDP         62 2007 → 2007 Len=20
  6562 125.210356  10.192.61.47     10.192.255.255   NBNS        92 Name query NB LAPTOP-D83LUS94<1c>
  6563 125.219008  10.192.142.51    10.192.255.255   UDP         62 2008 → 2008 Len=20
  6564 125.219954  10.192.142.51    10.192.255.255   UDP         62 2007 → 2007 Len=20
  6565 125.384954  10.192.142.51    10.192.255.255   UDP         62 2008 → 2008 Len=20
```

```
> Frame 1: 305 bytes on wire (2440 bits), 305 bytes captured (2440 bits) on interface 0
> Ethernet II, Src: IntelCor_96:6f:4a (d0:57:7b:96:6f:4a), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Internet Protocol Version 4, Src: 10.192.6.220, Dst: 10.192.255.255
> User Datagram Protocol, Src Port: 54915, Dst Port: 54915
> Data (263 bytes)
```

```
0000  ff ff ff ff ff ff d0 57  7b 96 6f 4a 08 00 45 00   ·······W {·oJ··E·
0010  01 23 38 f8 00 00 80 11  e4 76 0a c0 06 dc 0a c0   ·#8····· ·v······
0020  ff ff d6 83 d6 83 01 0f  83 d9 00 4c 41 50 54 4f   ········ ···LAPTO
0030  50 2d 43 50 42 49 38 33  48 53 00 b7 4f ae a9 00   P-CPBI83 HS··O···
0040  00 00 f0 b7 4f ae a9 00  00 00 d4 b7 4f ae a9 00   ····O··· ····O···
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0060  00 00 d0 f3 17 f0 94 02  00 00 50 b7 4f ae a9 00   ········ ··P·O···
0070  00 00 00 00 73 ec 01 00  00 00 33 27 00 00 00 00   ····s··· ··3'····
0080  00 00 c0 f3 17 f0 94 02  00 00 b0 fb 74 ec 94 02   ········ ····t···
0090  00 00 01 00 00 00 00 00  00 00 01 00 00 00 00 00   ········ ········
00a0  00 00 d6 86 eb 88 fa 7f  00 00 07 67 04 7b 38 38   ········ ···g·{88
00b0  66 64 38 62 33 32 2d 33  30 66 37 2d 34 64 66 31   fd8b32-3 0f7-4df1
00c0  2d 38 66 32 38 2d 64 32  62 34 66 31 37 39 66 62   -8f28-d2 b4f179fb
```

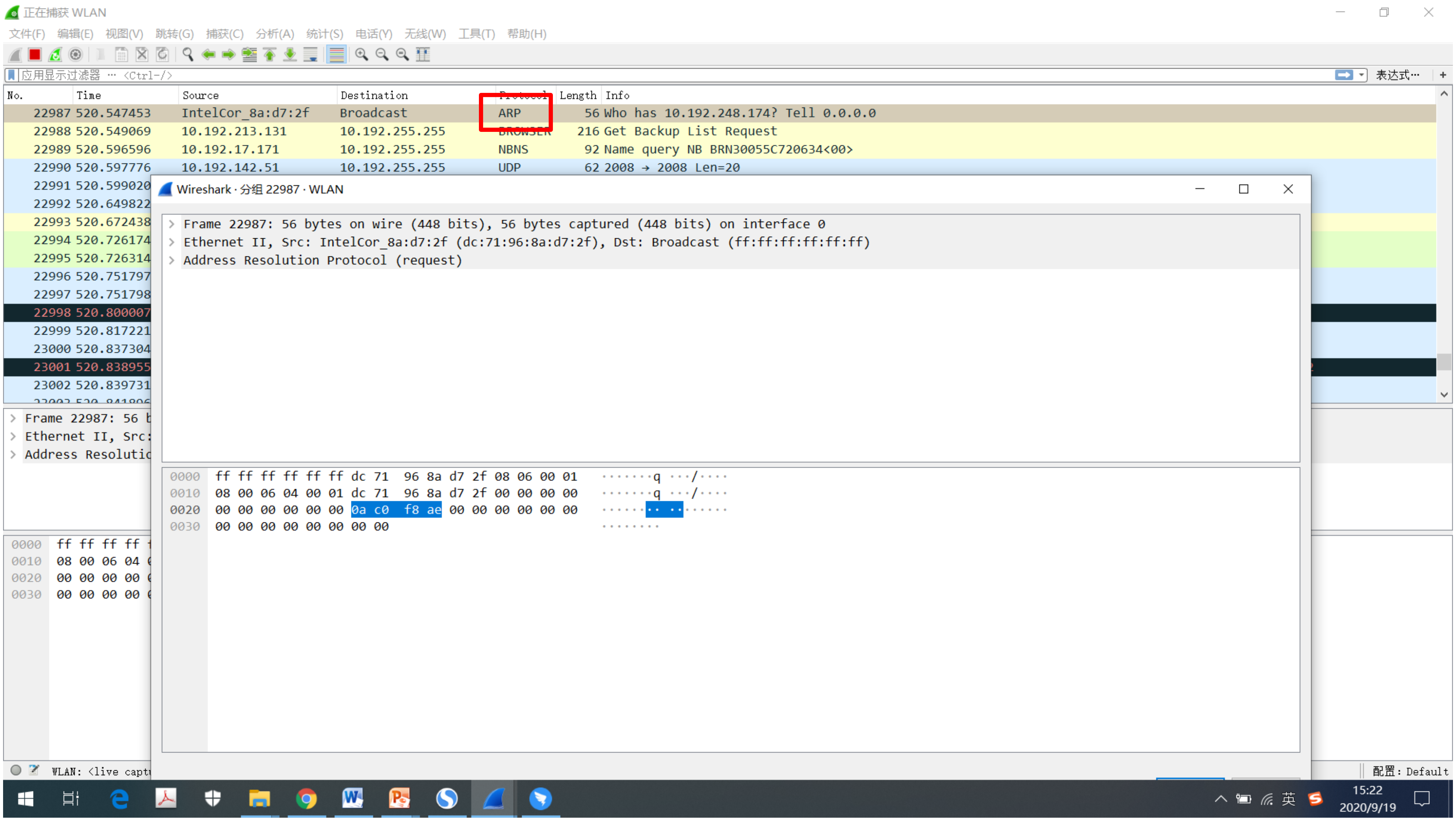
WLAN: <live capture in progress>　　分组: 6565 · 已显示: 6565 (100.0%)　　配置: Default

- 至上而下Wireshark三个面板：“Packet List”（分组列表），“Packet Detail”(分组详情)，“Packet Byte”(分组字节流)

Interface of Wireshark

- Wireshark三个面板：
  - “Packet List”（分组列表）
  - “Packet Detail”(分组详情)
  - “Packet Byte”(分组字节流)
- 列表中的每行显示捕捉文件的一个包。如果你的鼠标移到其中一行上，该包的更多详细信息会显示在“Packet Detail/分组详情”和"Packet Byte/分组字节流"面板。
- 在分析(解剖)分组时，Wireshark会将协议信息放到各个列。因为高层协议通常会覆盖底层协议，您通常在分组列表面板看到的都是每个包的最高层协议描述。（在Wireshark中最高层是应用层，底层是数据链路层）

Example I: ARP



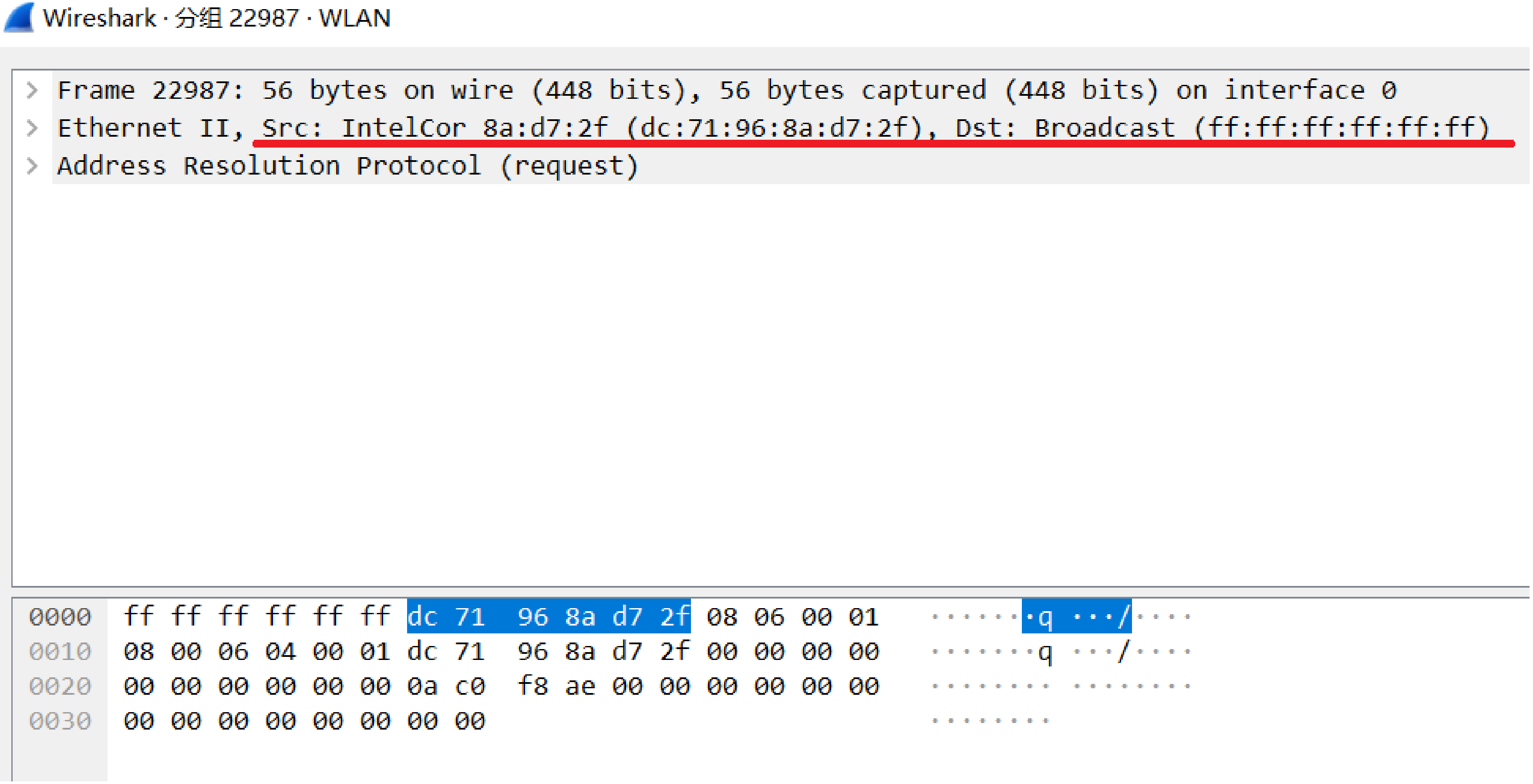
```
No.    Time        Source              Destination      Protocol  Length Info
 22987 520.547453  IntelCor_8a:d7:2f   Broadcast        ARP           56 Who has 10.192.248.174? Tell 0.0.0.0
 22988 520.549069  10.192.213.131      10.192.255.255   BROWSER      216 Get Backup List Request
 22989 520.596596  10.192.17.171       10.192.255.255   NBNS          92 Name query NB BRN30055C720634<00>
 22990 520.597776  10.192.142.51       10.192.255.255   UDP           62 2008 → 2008 Len=20
 22991 520.599020
 22992 520.649822
 22993 520.672438
 22994 520.726174
 22995 520.726314
 22996 520.751797
 22997 520.751798
 22998 520.800007
 22999 520.817221
 23000 520.837304
 23001 520.838955
 23002 520.839731
```

```
> Frame 22987: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface 0
> Ethernet II, Src: IntelCor_8a:d7:2f (dc:71:96:8a:d7:2f), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Address Resolution Protocol (request)
```

```
0000  ff ff ff ff ff ff dc 71  96 8a d7 2f 08 06 00 01   ·······q ···/····
0010  08 00 06 04 00 01 dc 71  96 8a d7 2f 00 00 00 00   ·······q ···/····
0020  00 00 00 00 00 00 0a c0  f8 ae 00 00 00 00 00 00   ········ ········
0030  00 00 00 00 00 00 00 00                            ········
```

ARP: Address Resolution Protocol [2]

- Because there are both *network-layer addresses* (for example, Internet **IP addresses**) and *link-layer addresses* (that is, **MAC addresses**), there is a need to translate between them. For the Internet, this is the job of the **Address Resolution Protocol** (**ARP**) [RFC826]



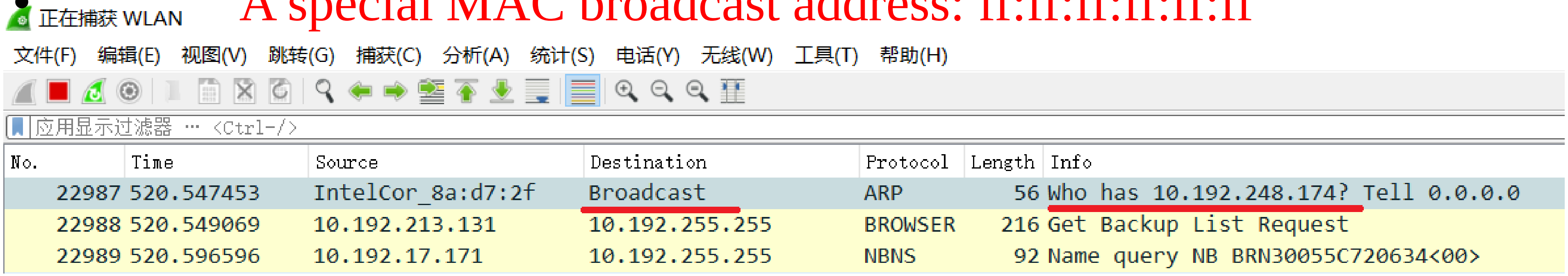
Wireshark · 分组 22987 · WLAN

```
> Frame 22987: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface 0
> Ethernet II, Src: IntelCor_8a:d7:2f (dc:71:96:8a:d7:2f), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Address Resolution Protocol (request)
```

```
0000  ff ff ff ff ff ff dc 71  96 8a d7 2f 08 06 00 01   ·······q ···/····
0010  08 00 06 04 00 01 dc 71  96 8a d7 2f 00 00 00 00   ·······q ···/····
0020  00 00 00 00 00 00 0a c0  f8 ae 00 00 00 00 00 00   ········ ········
0030  00 00 00 00 00 00 00 00                            ········
```

ARP: Address Resolution Protocol [2]

- The purpose of the ARP query packet is to query all the other nodes on the subnet to determine the MAC address corresponding to the IP address that is being resolved.
- An ARP query packet (In this example: it give the IP address: 10.192.248.174, want to know the MAC address of the IP address. A IPv4 address has 32 bits and expressed in decimals (十进制) ×××.×××.×××.×××)
- The MAC addresses are 6 bytes long, giving 2^{48} possible MAC addresses, and are expressed in hexadecimal (十六进制). (In this example: the MAC address of the source is dc:71:96:8a:d7:2f)
- A special MAC broadcast address: ff:ff:ff:ff:ff:ff

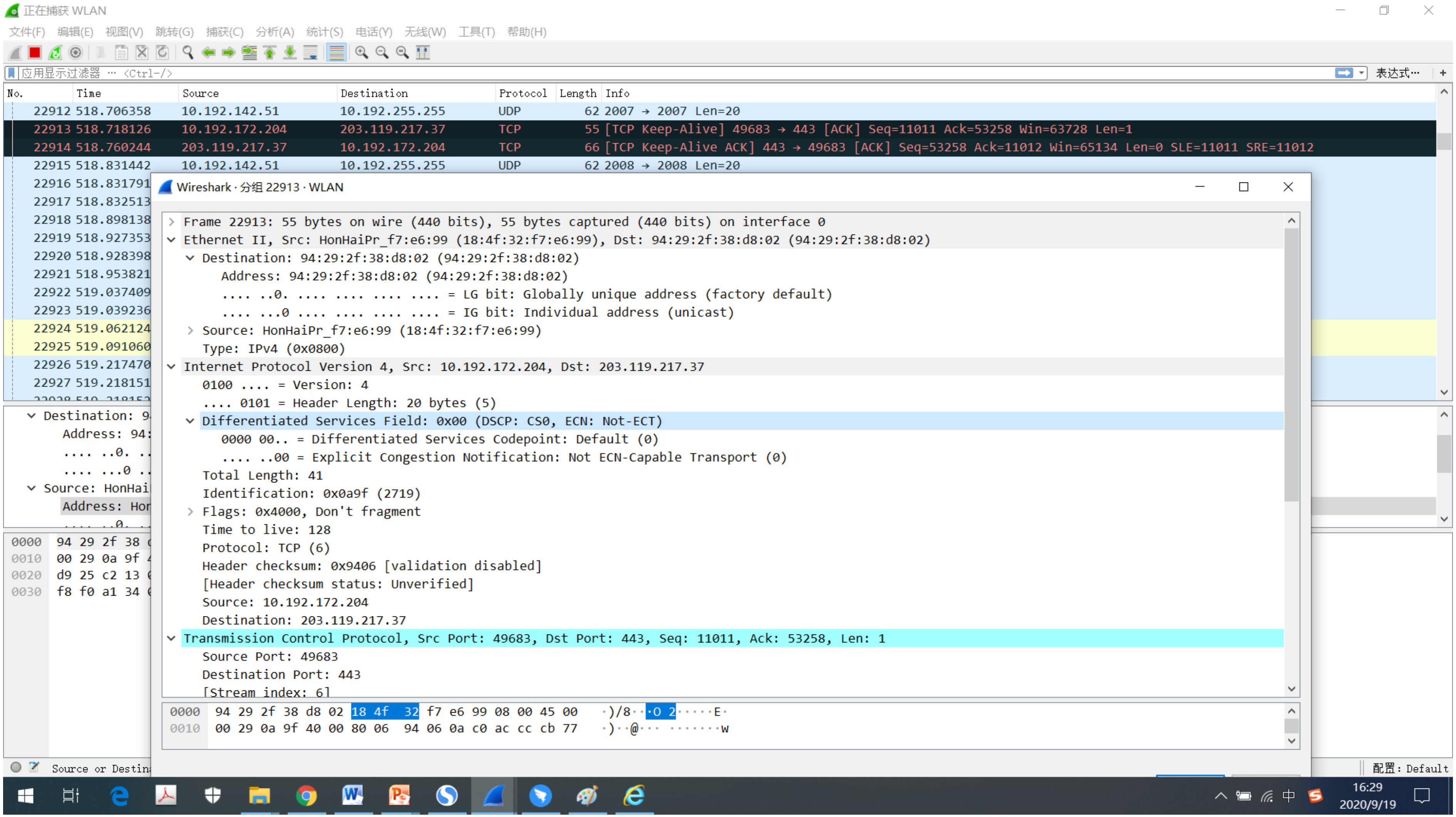

正在捕获 WLAN
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
应用显示过滤器 … <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
22987	520.547453	IntelCor_8a:d7:2f	Broadcast	ARP	56	Who has 10.192.248.174? Tell 0.0.0.0
22988	520.549069	10.192.213.131	10.192.255.255	BROWSER	216	Get Backup List Request
22989	520.596596	10.192.17.171	10.192.255.255	NBNS	92	Name query NB BRN30055C720634<00>

ARP: Address Resolution Protocol [2]

- Each node (host and router) has **an ARP table** in its memory, which contains mappings of IP addresses to MAC addresses.
- The ARP table contains a time-to-live (**TTL**) value, which indicates when each mapping will be deleted from the table.
  - A typical expiration time for an entry is 20 minutes from when an entry is placed in an ARP table.
- ARP vs. DNS
  - ARP resolves an IP address to a MAC address only for nodes on the same subnet.
  - DNS resolves host names to IP addresses for hosts anywhere in the Internet.
- ARP is probably best considered a protocol that straddles the boundary between the link and network layers.

Example II: TCP


正在捕获 WLAN
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
应用显示过滤器 … <Ctrl-/>
表达式…
Wireshark · 分组 22913 · WLAN
Source or Destin
配置：Default
16:29
2020/9/19

```
No.    Time        Source          Destination      Protocol Length Info
 22912 518.706358  10.192.142.51   10.192.255.255   UDP         62 2007 → 2007 Len=20
 22913 518.718126  10.192.172.204  203.119.217.37   TCP         55 [TCP Keep-Alive] 49683 → 443 [ACK] Seq=11011 Ack=53258 Win=63728 Len=1
 22914 518.760244  203.119.217.37  10.192.172.204   TCP         66 [TCP Keep-Alive ACK] 443 → 49683 [ACK] Seq=53258 Ack=11012 Win=65134 Len=0 SLE=11011 SRE=11012
 22915 518.831442  10.192.142.51   10.192.255.255   UDP         62 2008 → 2008 Len=20
 22916 518.831791
 22917 518.832513
 22918 518.898138
 22919 518.927353
 22920 518.928398
 22921 518.953821
 22922 519.037409
 22923 519.039236
 22924 519.062124
 22925 519.091060
 22926 519.217470
 22927 519.218151
```

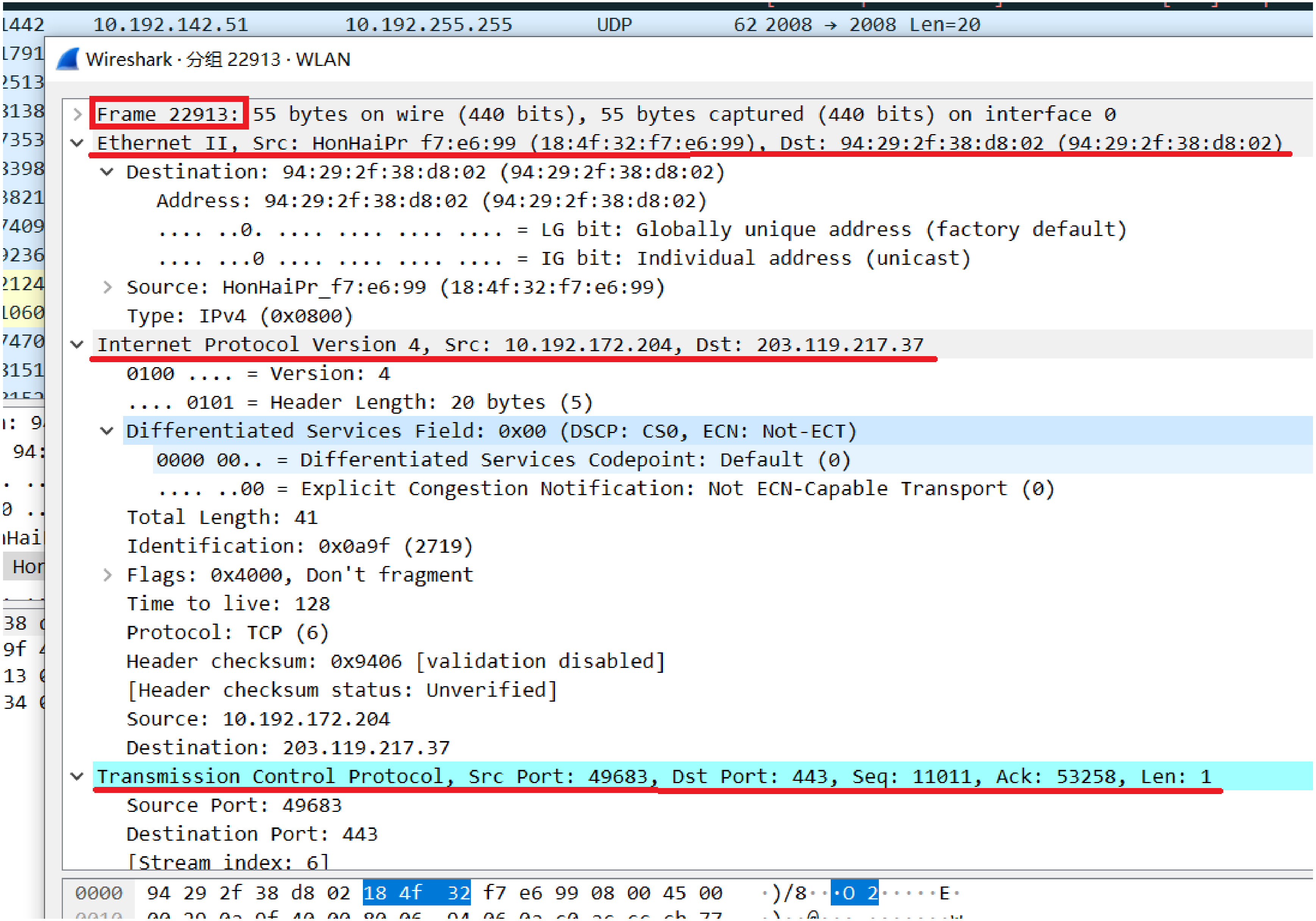
```
> Frame 22913: 55 bytes on wire (440 bits), 55 bytes captured (440 bits) on interface 0
v Ethernet II, Src: HonHaiPr_f7:e6:99 (18:4f:32:f7:e6:99), Dst: 94:29:2f:38:d8:02 (94:29:2f:38:d8:02)
  v Destination: 94:29:2f:38:d8:02 (94:29:2f:38:d8:02)
      Address: 94:29:2f:38:d8:02 (94:29:2f:38:d8:02)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  > Source: HonHaiPr_f7:e6:99 (18:4f:32:f7:e6:99)
    Type: IPv4 (0x0800)
v Internet Protocol Version 4, Src: 10.192.172.204, Dst: 203.119.217.37
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  v Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      0000 00.. = Differentiated Services Codepoint: Default (0)
      .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    Total Length: 41
    Identification: 0x0a9f (2719)
  > Flags: 0x4000, Don't fragment
    Time to live: 128
    Protocol: TCP (6)
    Header checksum: 0x9406 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.192.172.204
    Destination: 203.119.217.37
v Transmission Control Protocol, Src Port: 49683, Dst Port: 443, Seq: 11011, Ack: 53258, Len: 1
    Source Port: 49683
    Destination Port: 443
    [Stream index: 6]
```

```
0000  94 29 2f 38 d8 02 18 4f  32 f7 e6 99 08 00 45 00   ·)/8···O 2····E·
0010  00 29 0a 9f 40 00 80 06  94 06 0a c0 ac cc cb 77   ·)··@··· ·······w
```

```
v Destination: 9
    Address: 94:
    .... ..0. .
    .... ...0 .
v Source: HonHai
    Address: Hon
    .... ..0. .
```

```
0000  94 29 2f 38
0010  00 29 0a 9f
0020  d9 25 c2 13
0030  f8 f0 a1 34
```

Example II: TCP



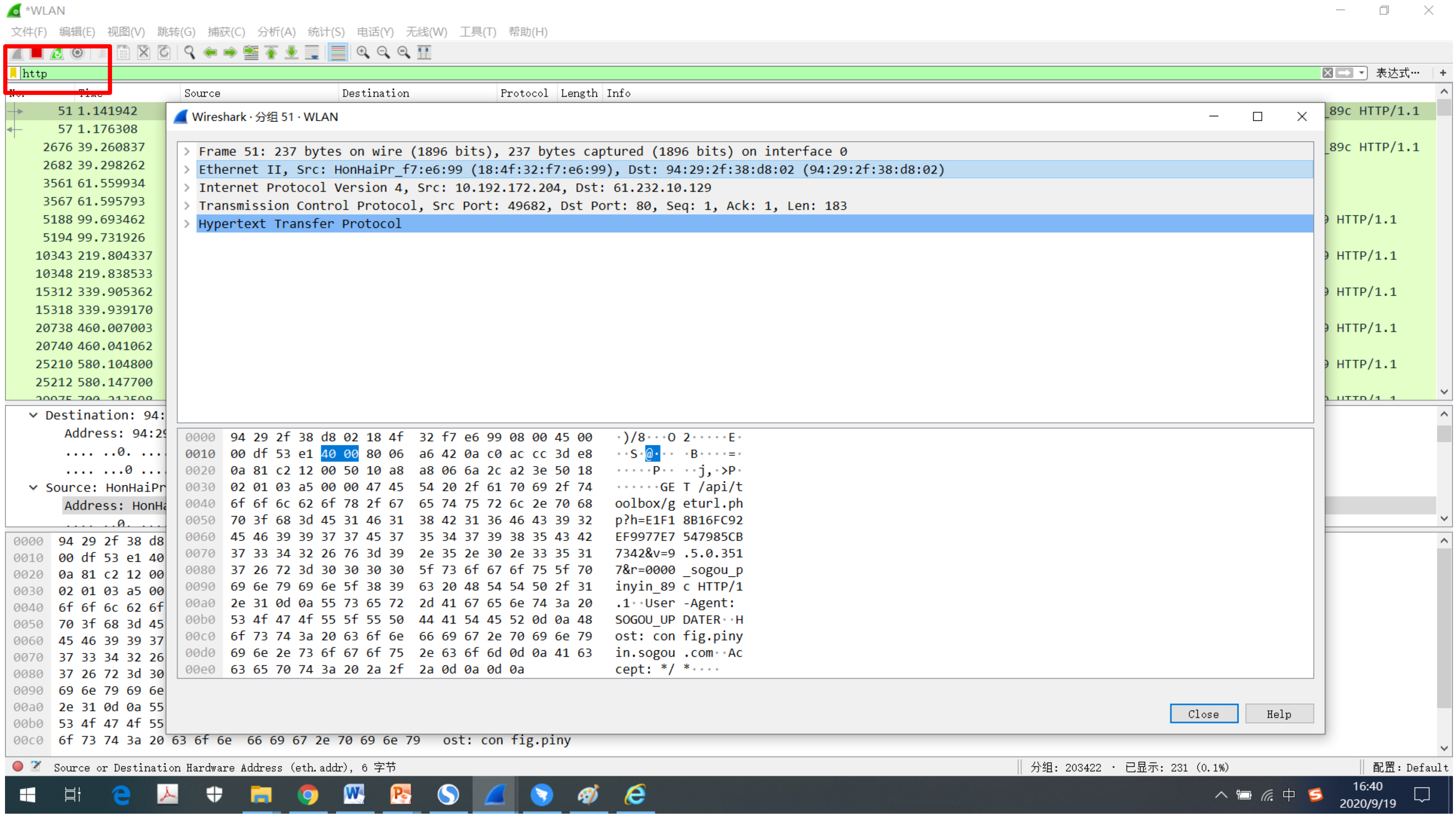
```
1442   10.192.142.51        10.192.255.255       UDP        62 2008 → 2008 Len=20
```

Wireshark · 分组 22913 · WLAN

```
> Frame 22913: 55 bytes on wire (440 bits), 55 bytes captured (440 bits) on interface 0
v Ethernet II, Src: HonHaiPr_f7:e6:99 (18:4f:32:f7:e6:99), Dst: 94:29:2f:38:d8:02 (94:29:2f:38:d8:02)
    v Destination: 94:29:2f:38:d8:02 (94:29:2f:38:d8:02)
        Address: 94:29:2f:38:d8:02 (94:29:2f:38:d8:02)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    > Source: HonHaiPr_f7:e6:99 (18:4f:32:f7:e6:99)
      Type: IPv4 (0x0800)
v Internet Protocol Version 4, Src: 10.192.172.204, Dst: 203.119.217.37
      0100 .... = Version: 4
      .... 0101 = Header Length: 20 bytes (5)
    v Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
        0000 00.. = Differentiated Services Codepoint: Default (0)
        .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
      Total Length: 41
      Identification: 0x0a9f (2719)
    > Flags: 0x4000, Don't fragment
      Time to live: 128
      Protocol: TCP (6)
      Header checksum: 0x9406 [validation disabled]
      [Header checksum status: Unverified]
      Source: 10.192.172.204
      Destination: 203.119.217.37
v Transmission Control Protocol, Src Port: 49683, Dst Port: 443, Seq: 11011, Ack: 53258, Len: 1
      Source Port: 49683
      Destination Port: 443
      [Stream index: 6]
```

```
0000  94 29 2f 38 d8 02 18 4f  32 f7 e6 99 08 00 45 00   ·)/8···O 2····E·
```

Example III: HTTP


*WLAN
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
表达式…
No.
Time
Source
Destination
Protocol
Length
Info
Wireshark · 分组 51 · WLAN
Close
Help
16:40
2020/9/19

http

```
   51 1.141942
   57 1.176308
 2676 39.260837
 2682 39.298262
 3561 61.559934
 3567 61.595793
 5188 99.693462
 5194 99.731926
10343 219.804337
10348 219.838533
15312 339.905362
15318 339.939170
20738 460.007003
20740 460.041062
25210 580.104800
25212 580.147700
```

```
> Frame 51: 237 bytes on wire (1896 bits), 237 bytes captured (1896 bits) on interface 0
> Ethernet II, Src: HonHaiPr_f7:e6:99 (18:4f:32:f7:e6:99), Dst: 94:29:2f:38:d8:02 (94:29:2f:38:d8:02)
> Internet Protocol Version 4, Src: 10.192.172.204, Dst: 61.232.10.129
> Transmission Control Protocol, Src Port: 49682, Dst Port: 80, Seq: 1, Ack: 1, Len: 183
> Hypertext Transfer Protocol
```

```
0000  94 29 2f 38 d8 02 18 4f  32 f7 e6 99 08 00 45 00   ·)/8···O 2·····E·
0010  00 df 53 e1 40 00 80 06  a6 42 0a c0 ac cc 3d e8   ··S·@··· ·B····=·
0020  0a 81 c2 12 00 50 10 a8  a8 06 6a 2c a2 3e 50 18   ·····P·· ··j,·>P·
0030  02 01 03 a5 00 00 47 45  54 20 2f 61 70 69 2f 74   ······GE T /api/t
0040  6f 6f 6c 62 6f 78 2f 67  65 74 75 72 6c 2e 70 68   oolbox/g eturl.ph
0050  70 3f 68 3d 45 31 46 31  38 42 31 36 46 43 39 32   p?h=E1F1 8B16FC92
0060  45 46 39 39 37 45 37 35  34 37 39 38 35 43 42      EF9977E7 547985CB
0070  37 33 34 32 26 76 3d 39  2e 35 2e 30 2e 33 35 31   7342&v=9 .5.0.351
0080  37 26 72 3d 30 30 30 30  5f 73 6f 67 6f 75 5f 70   7&r=0000 _sogou_p
0090  69 6e 79 69 6e 5f 38 39  63 20 48 54 54 50 2f 31   inyin_89 c HTTP/1
00a0  2e 31 0d 0a 55 73 65 72  2d 41 67 65 6e 74 3a 20   .1··User -Agent: 
00b0  53 4f 47 4f 55 5f 55 50  44 41 54 45 52 0d 0a 48   SOGOU_UP DATER··H
00c0  6f 73 74 3a 20 63 6f 6e  66 69 67 2e 70 69 6e 79   ost: con fig.piny
00d0  69 6e 2e 73 6f 67 6f 75  2e 63 6f 6d 0d 0a 41 63   in.sogou .com··Ac
00e0  63 65 70 74 3a 20 2a 2f  2a 0d 0a 0d 0a            cept: */ *····
```

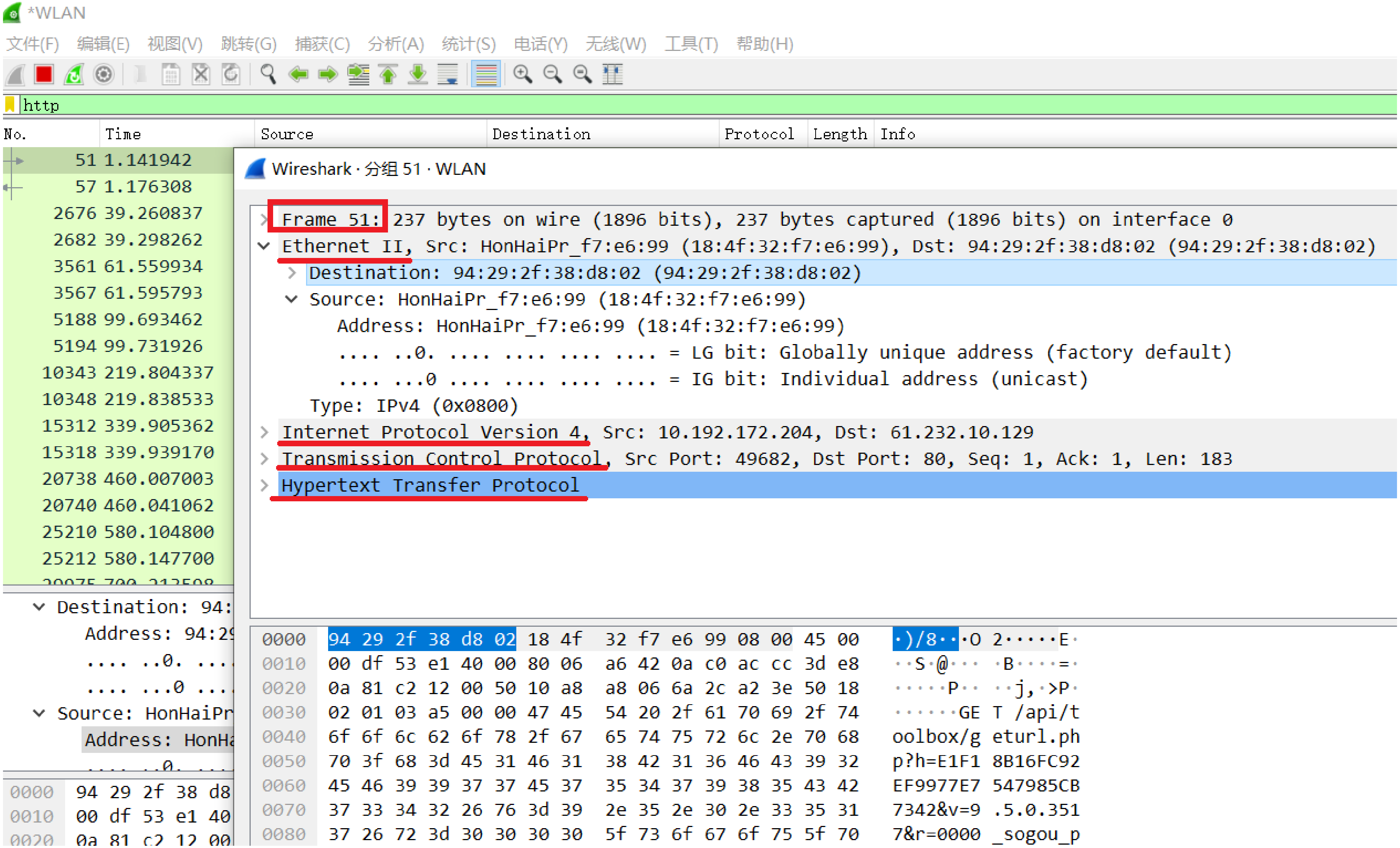
```
v Destination: 94:
    Address: 94:2
    .... ..0. ...
    .... ...0 ...
v Source: HonHaiPr
    Address: HonHa
```

Source or Destination Hardware Address (eth.addr), 6 字节

分组: 203422 · 已显示: 231 (0.1%)

配置: Default

Example III: HTTP


*WLAN
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
http
No. Time Source Destination Protocol Length Info
Wireshark · 分组 51 · WLAN

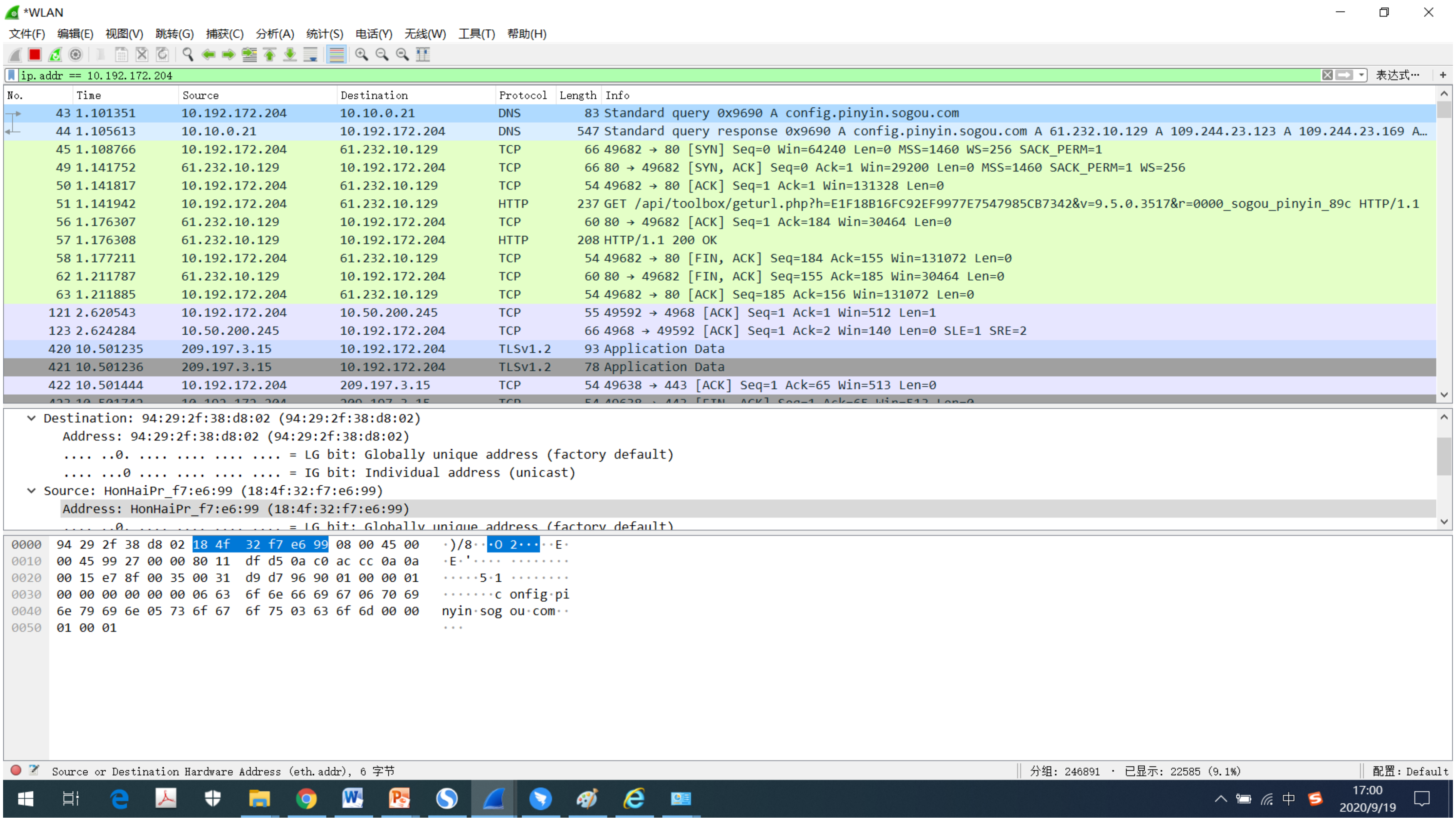
```
   51 1.141942
   57 1.176308
 2676 39.260837
 2682 39.298262
 3561 61.559934
 3567 61.595793
 5188 99.693462
 5194 99.731926
10343 219.804337
10348 219.838533
15312 339.905362
15318 339.939170
20738 460.007003
20740 460.041062
25210 580.104800
25212 580.147700
29975 700.213508
```

```
  ˅ Destination: 94:
      Address: 94:29
      .... ..0. ....
      .... ...0 ....
  ˅ Source: HonHaiPr
      Address: HonHa
      .... ..0. ....

0000  94 29 2f 38 d8
0010  00 df 53 e1 40
0020  0a 81 c2 12 00
```

```
> Frame 51: 237 bytes on wire (1896 bits), 237 bytes captured (1896 bits) on interface 0
˅ Ethernet II, Src: HonHaiPr_f7:e6:99 (18:4f:32:f7:e6:99), Dst: 94:29:2f:38:d8:02 (94:29:2f:38:d8:02)
  > Destination: 94:29:2f:38:d8:02 (94:29:2f:38:d8:02)
  ˅ Source: HonHaiPr_f7:e6:99 (18:4f:32:f7:e6:99)
      Address: HonHaiPr_f7:e6:99 (18:4f:32:f7:e6:99)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 10.192.172.204, Dst: 61.232.10.129
> Transmission Control Protocol, Src Port: 49682, Dst Port: 80, Seq: 1, Ack: 1, Len: 183
> Hypertext Transfer Protocol
```

```
0000  94 29 2f 38 d8 02 18 4f  32 f7 e6 99 08 00 45 00   ·)/8···O 2·····E·
0010  00 df 53 e1 40 00 80 06  a6 42 0a c0 ac cc 3d e8   ··S·@···  ·B····=·
0020  0a 81 c2 12 00 50 10 a8  a8 06 6a 2c a2 3e 50 18   ·····P··  ··j,·>P·
0030  02 01 03 a5 00 00 47 45  54 20 2f 61 70 69 2f 74   ······GE T /api/t
0040  6f 6f 6c 62 6f 78 2f 67  65 74 75 72 6c 2e 70 68   oolbox/g eturl.ph
0050  70 3f 68 3d 45 31 46 31  38 42 31 36 46 43 39 32   p?h=E1F1 8B16FC92
0060  45 46 39 39 37 37 45 37  35 34 37 39 38 35 43 42   EF9977E7 547985CB
0070  37 33 34 32 26 76 3d 39  2e 35 2e 30 2e 33 35 31   7342&v=9 .5.0.351
0080  37 26 72 3d 30 30 30 30  5f 73 6f 67 6f 75 5f 70   7&r=0000 _sogou_p
```

Example IV: ip.addr == 10.192.172.204



```
ip.addr == 10.192.172.204
```

```
No.   Time       Source          Destination     Protocol Length Info
   43 1.101351   10.192.172.204  10.10.0.21      DNS         83 Standard query 0x9690 A config.pinyin.sogou.com
   44 1.105613   10.10.0.21      10.192.172.204  DNS        547 Standard query response 0x9690 A config.pinyin.sogou.com A 61.232.10.129 A 109.244.23.123 A 109.244.23.169 A…
   45 1.108766   10.192.172.204  61.232.10.129   TCP         66 49682 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
   49 1.141752   61.232.10.129   10.192.172.204  TCP         66 80 → 49682 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=256
   50 1.141817   10.192.172.204  61.232.10.129   TCP         54 49682 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0
   51 1.141942   10.192.172.204  61.232.10.129   HTTP       237 GET /api/toolbox/geturl.php?h=E1F18B16FC92EF9977E7547985CB7342&v=9.5.0.3517&r=0000_sogou_pinyin_89c HTTP/1.1
   56 1.176307   61.232.10.129   10.192.172.204  TCP         60 80 → 49682 [ACK] Seq=1 Ack=184 Win=30464 Len=0
   57 1.176308   61.232.10.129   10.192.172.204  HTTP       208 HTTP/1.1 200 OK
   58 1.177211   10.192.172.204  61.232.10.129   TCP         54 49682 → 80 [FIN, ACK] Seq=184 Ack=155 Win=131072 Len=0
   62 1.211787   61.232.10.129   10.192.172.204  TCP         60 80 → 49682 [FIN, ACK] Seq=155 Ack=185 Win=30464 Len=0
   63 1.211885   10.192.172.204  61.232.10.129   TCP         54 49682 → 80 [ACK] Seq=185 Ack=156 Win=131072 Len=0
  121 2.620543   10.192.172.204  10.50.200.245   TCP         55 49592 → 4968 [ACK] Seq=1 Ack=1 Win=512 Len=1
  123 2.624284   10.50.200.245   10.192.172.204  TCP         66 4968 → 49592 [ACK] Seq=1 Ack=2 Win=140 Len=0 SLE=1 SRE=2
  420 10.501235  209.197.3.15    10.192.172.204  TLSv1.2     93 Application Data
  421 10.501236  209.197.3.15    10.192.172.204  TLSv1.2     78 Application Data
  422 10.501444  10.192.172.204  209.197.3.15    TCP         54 49638 → 443 [ACK] Seq=1 Ack=65 Win=513 Len=0
  423 10.501742  10.192.172.204  209.197.3.15    TCP         54 49638 → 443 [FIN, ACK] Seq=1 Ack=65 Win=513 Len=0
```

```
  ∨ Destination: 94:29:2f:38:d8:02 (94:29:2f:38:d8:02)
      Address: 94:29:2f:38:d8:02 (94:29:2f:38:d8:02)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  ∨ Source: HonHaiPr_f7:e6:99 (18:4f:32:f7:e6:99)
      Address: HonHaiPr_f7:e6:99 (18:4f:32:f7:e6:99)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
```

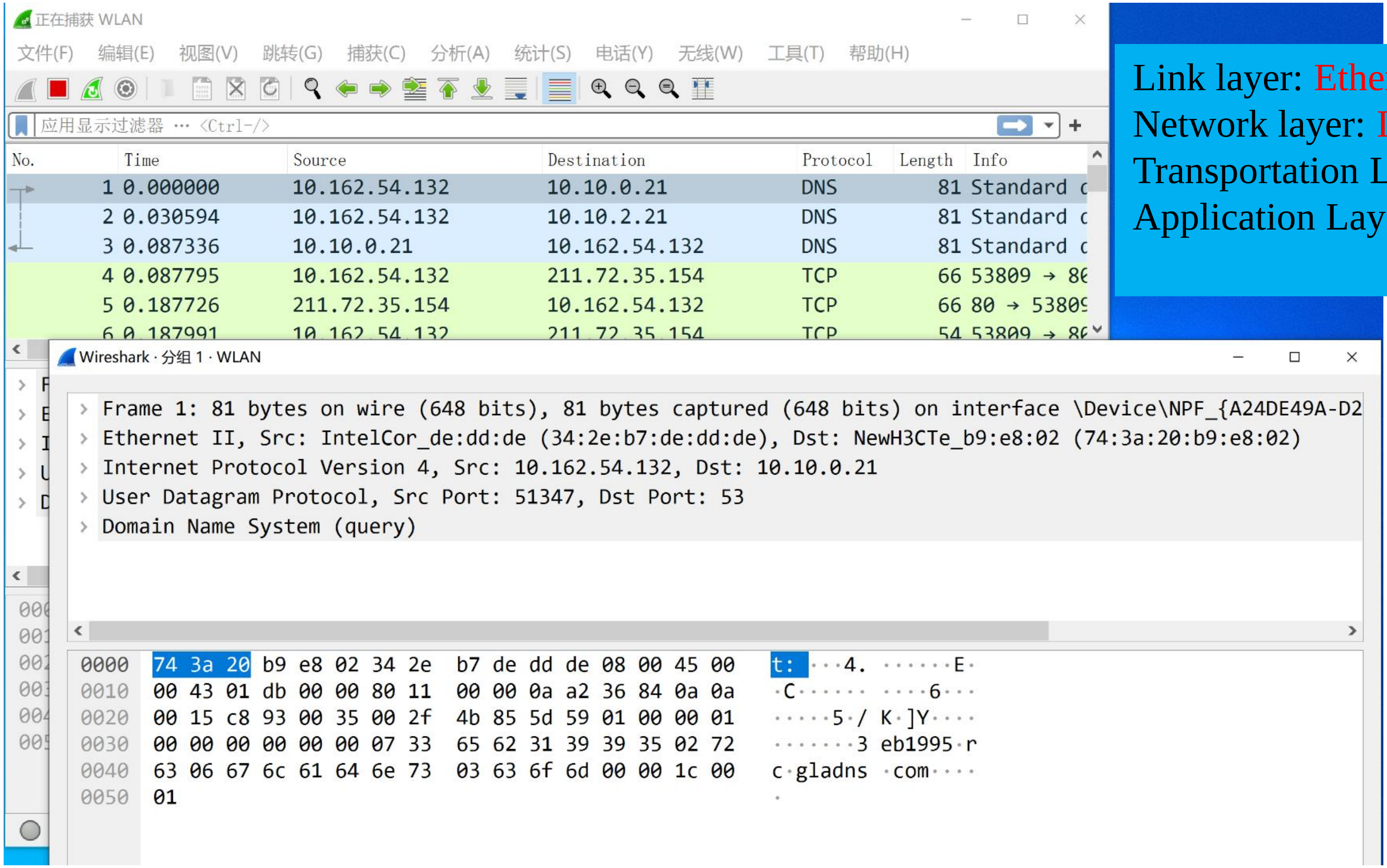
```
0000  94 29 2f 38 d8 02 18 4f  32 f7 e6 99 08 00 45 00   ·)/8··O 2·····E·
0010  00 45 99 27 00 00 80 11  df d5 0a c0 ac cc 0a 0a   ·E·'···· ········
0020  00 15 e7 8f 00 35 00 31  d9 d7 96 90 01 00 00 01   ·····5·1 ········
0030  00 00 00 00 00 00 06 63  6f 6e 66 69 67 06 70 69   ·······c onfig·pi
0040  6e 79 69 6e 05 73 6f 67  6f 75 03 63 6f 6d 00 00   nyin·sog ou·com··
0050  01 00 01                                           ···
```

Source or Destination Hardware Address (eth.addr), 6 字节 | 分组: 246891 · 已显示: 22585 (9.1%) | 配置: Default

ip.addr == x.x.x.x vs. host.addr == x.x.x.x

- 实验中第4部分和第5部分相比，区别在于ip.addr == x.x.x.x是捕获所有数据包，但是只显示与ip地址为x.x.x.x有关的数据包，而host.addr == x.x.x.x只捕获ip地址为x.x.x.x的数据包。检查一下实验结果， host.addr = x.x.x.x命令下抓获数据包量要小很多。
- 注意命令“host.addr = x.x.x.x”已经停用，改为“ip.host = x.x.x.x”
- 或者用这两个命令：ip.src_host == x.x.x.x 只抓数据包中源地址为x.x.x.x的数据包；或 ip.dst_host == x.x.x.x只抓数据包中目标地址为 x.x.x.x的数据包。

Example: DNS



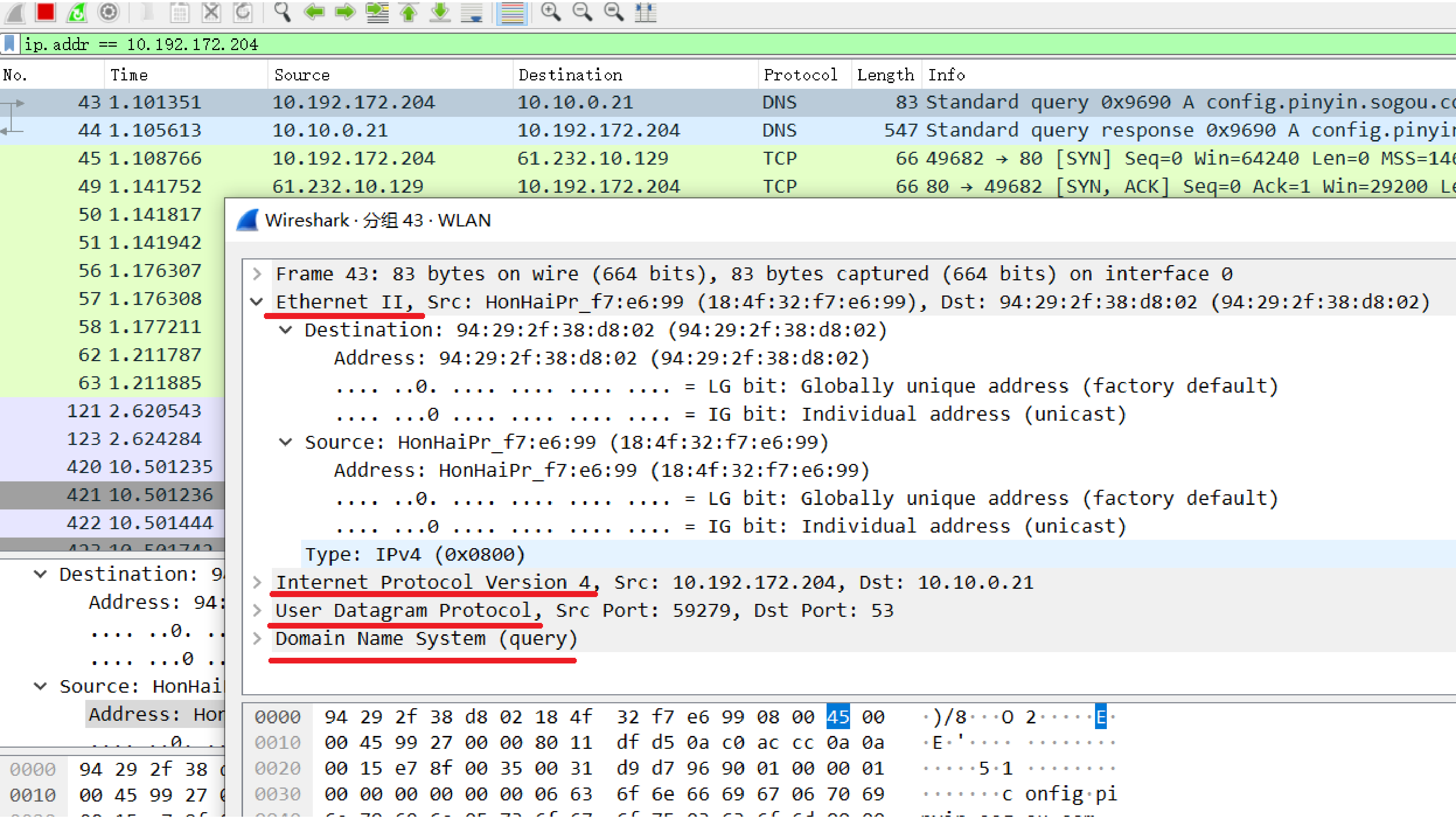
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.162.54.132	10.10.0.21	DNS	81	Standard c
2	0.030594	10.162.54.132	10.10.2.21	DNS	81	Standard c
3	0.087336	10.10.0.21	10.162.54.132	DNS	81	Standard c
4	0.087795	10.162.54.132	211.72.35.154	TCP	66	53809 → 80
5	0.187726	211.72.35.154	10.162.54.132	TCP	66	80 → 53809
6	0.187991	10.162.54.132	211.72.35.154	TCP	54	53809 → 80

```
> Frame 1: 81 bytes on wire (648 bits), 81 bytes captured (648 bits) on interface \Device\NPF_{A24DE49A-D2
> Ethernet II, Src: IntelCor_de:dd:de (34:2e:b7:de:dd:de), Dst: NewH3CTe_b9:e8:02 (74:3a:20:b9:e8:02)
> Internet Protocol Version 4, Src: 10.162.54.132, Dst: 10.10.0.21
> User Datagram Protocol, Src Port: 51347, Dst Port: 53
> Domain Name System (query)
```

```
0000  74 3a 20 b9 e8 02 34 2e  b7 de dd de 08 00 45 00   t:···4. ······E·
0010  00 43 01 db 00 00 80 11  00 00 0a a2 36 84 0a 0a   ·C······ ····6···
0020  00 15 c8 93 00 35 00 2f  4b 85 5d 59 01 00 00 01   ·····5·/ K·]Y····
0030  00 00 00 00 00 00 07 33  65 62 31 39 39 35 02 72   ·······3 eb1995·r
0040  63 06 67 6c 61 64 6e 73  03 63 6f 6d 00 00 1c 00   c·gladns ·com····
0050  01                                                 ·
```

Link layer: Ethernet II
Network layer: IPv4
Transportation Layer: UDP
Application Layer: DNS

Example V: DNS



ip.addr == 10.192.172.204

```
No.    Time         Source            Destination       Protocol  Length Info
    43 1.101351     10.192.172.204    10.10.0.21        DNS           83 Standard query 0x9690 A config.pinyin.sogou.co
    44 1.105613     10.10.0.21        10.192.172.204    DNS          547 Standard query response 0x9690 A config.pinyi
    45 1.108766     10.192.172.204    61.232.10.129     TCP           66 49682 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=14
    49 1.141752     61.232.10.129     10.192.172.204    TCP           66 80 → 49682 [SYN, ACK] Seq=0 Ack=1 Win=29200 Le
    50 1.141817
    51 1.141942
    56 1.176307
    57 1.176308
    58 1.177211
    62 1.211787
    63 1.211885
   121 2.620543
   123 2.624284
   420 10.501235
   421 10.501236
   422 10.501444
```

Wireshark · 分组 43 · WLAN

```
> Frame 43: 83 bytes on wire (664 bits), 83 bytes captured (664 bits) on interface 0
v Ethernet II, Src: HonHaiPr_f7:e6:99 (18:4f:32:f7:e6:99), Dst: 94:29:2f:38:d8:02 (94:29:2f:38:d8:02)
  v Destination: 94:29:2f:38:d8:02 (94:29:2f:38:d8:02)
      Address: 94:29:2f:38:d8:02 (94:29:2f:38:d8:02)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  v Source: HonHaiPr_f7:e6:99 (18:4f:32:f7:e6:99)
      Address: HonHaiPr_f7:e6:99 (18:4f:32:f7:e6:99)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 10.192.172.204, Dst: 10.10.0.21
> User Datagram Protocol, Src Port: 59279, Dst Port: 53
> Domain Name System (query)
```

```
0000  94 29 2f 38 d8 02 18 4f  32 f7 e6 99 08 00 45 00   ·)/8···O 2·····E·
0010  00 45 99 27 00 00 80 11  df d5 0a c0 ac cc 0a 0a   ·E·'···· ········
0020  00 15 e7 8f 00 35 00 31  d9 d7 96 90 01 00 00 01   ·····5·1 ········
0030  00 00 00 00 00 00 06 63  6f 6e 66 69 67 06 70 69   ·······c onfig·pi
```

```
v Destination: 9
    Address: 94:
    .... ..0. .
    .... ...0 .
v Source: HonHai
    Address: Hon
    .... ..0. .
0000  94 29 2f 38 
0010  00 45 99 27 
```

注意：在DNS数据包中传输层用的协议是UDP，不是TCP协议！Port number: 53

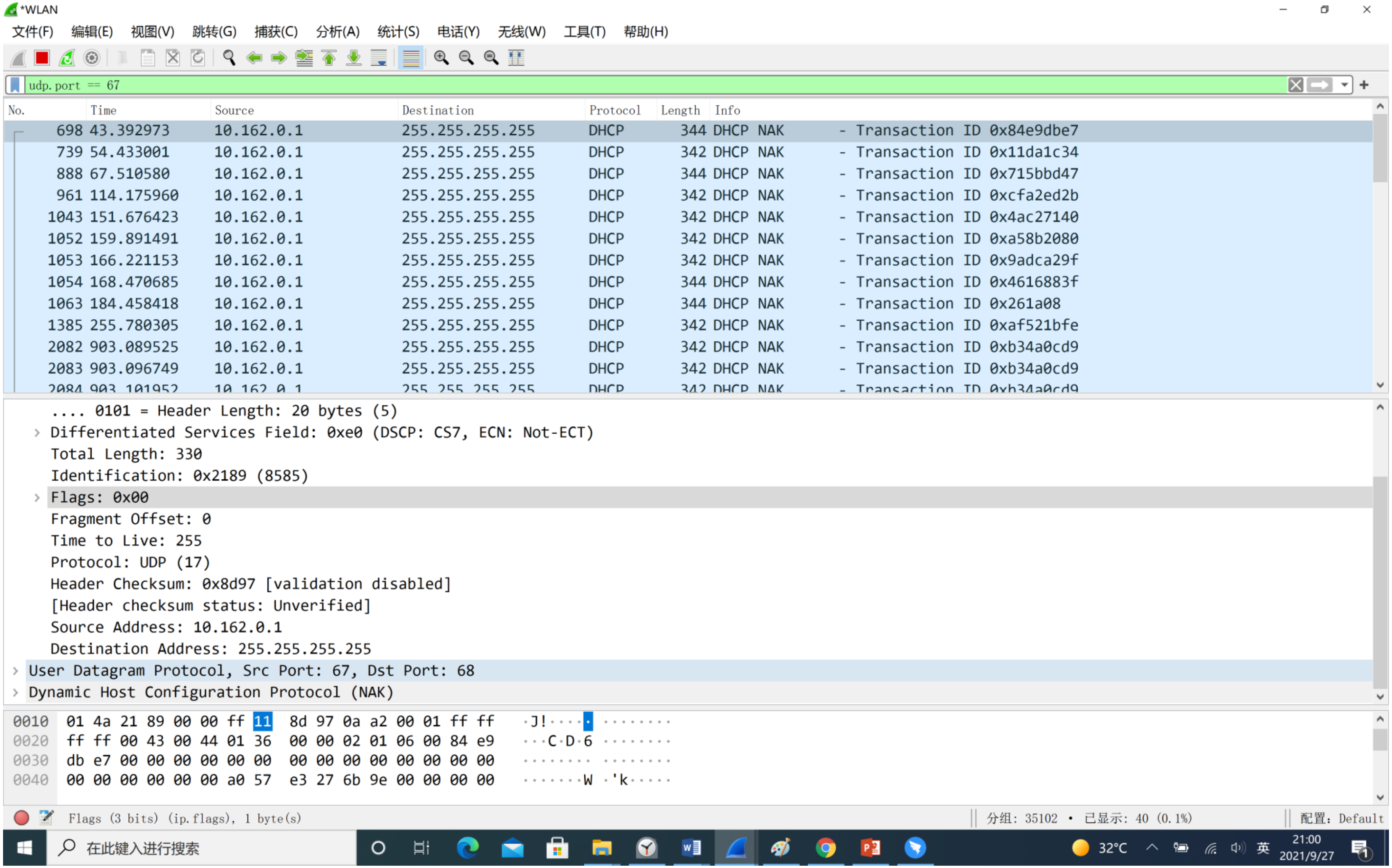
tcp.port == 443 (or 80, or 25)

- 实验1中Part 1第6题变为tcp.port == X 和udp.port == X

TCP 21 = 文件传输
TCP 22 = 远程登录协议
TCP 23 = 远程登录
TCP 25 = 电子邮件 (SMTP)
TCP 80 = http
TCP 110 = 电子邮件(Pop3)
TCP 179 = Border 网关协议 (BGP)
TCP 443 = 网页安全服务
TCP 546 = DHCP Client
TCP 547 = DHCP Server

UDP 53 = 域名解析
UDP 67 = 动态IP服务 DHCP
UDP 68 = 客户端向68端口DHCP服务器广播请求地址配置， DHCP服务器向67端口广播回应请求。

“udp.port == 67” Example



```
udp.port == 67
```

```
No.   Time         Source       Destination       Protocol  Length Info
  698 43.392973    10.162.0.1   255.255.255.255   DHCP      344 DHCP NAK      - Transaction ID 0x84e9dbe7
  739 54.433001    10.162.0.1   255.255.255.255   DHCP      342 DHCP NAK      - Transaction ID 0x11da1c34
  888 67.510580    10.162.0.1   255.255.255.255   DHCP      344 DHCP NAK      - Transaction ID 0x715bbd47
  961 114.175960   10.162.0.1   255.255.255.255   DHCP      342 DHCP NAK      - Transaction ID 0xcfa2ed2b
 1043 151.676423   10.162.0.1   255.255.255.255   DHCP      342 DHCP NAK      - Transaction ID 0x4ac27140
 1052 159.891491   10.162.0.1   255.255.255.255   DHCP      342 DHCP NAK      - Transaction ID 0xa58b2080
 1053 166.221153   10.162.0.1   255.255.255.255   DHCP      342 DHCP NAK      - Transaction ID 0x9adca29f
 1054 168.470685   10.162.0.1   255.255.255.255   DHCP      344 DHCP NAK      - Transaction ID 0x4616883f
 1063 184.458418   10.162.0.1   255.255.255.255   DHCP      344 DHCP NAK      - Transaction ID 0x261a08
 1385 255.780305   10.162.0.1   255.255.255.255   DHCP      342 DHCP NAK      - Transaction ID 0xaf521bfe
 2082 903.089525   10.162.0.1   255.255.255.255   DHCP      342 DHCP NAK      - Transaction ID 0xb34a0cd9
 2083 903.096749   10.162.0.1   255.255.255.255   DHCP      342 DHCP NAK      - Transaction ID 0xb34a0cd9
 2084 903.101952   10.162.0.1   255.255.255.255   DHCP      342 DHCP NAK      - Transaction ID 0xb34a0cd9
```

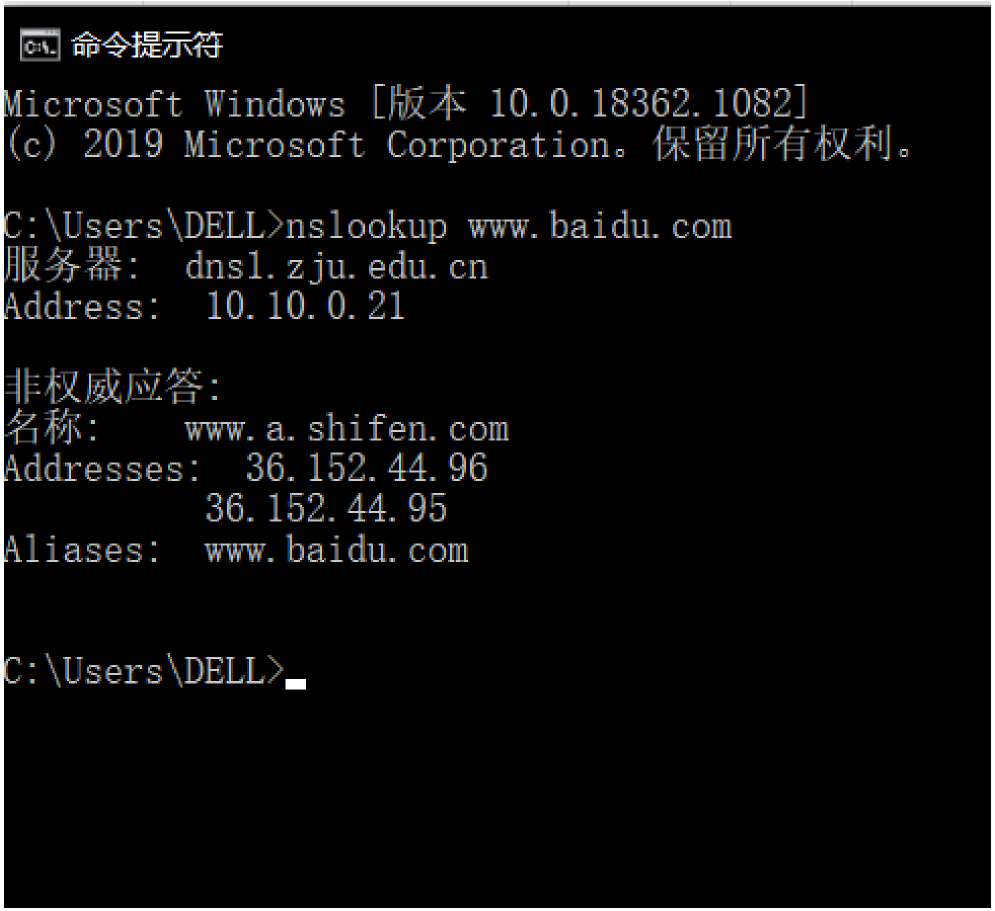
```
      .... 0101 = Header Length: 20 bytes (5)
    > Differentiated Services Field: 0xe0 (DSCP: CS7, ECN: Not-ECT)
      Total Length: 330
      Identification: 0x2189 (8585)
    > Flags: 0x00
      Fragment Offset: 0
      Time to Live: 255
      Protocol: UDP (17)
      Header Checksum: 0x8d97 [validation disabled]
      [Header checksum status: Unverified]
      Source Address: 10.162.0.1
      Destination Address: 255.255.255.255
> User Datagram Protocol, Src Port: 67, Dst Port: 68
> Dynamic Host Configuration Protocol (NAK)
```

```
0010  01 4a 21 89 00 00 ff 11  8d 97 0a a2 00 01 ff ff   ·J!····· ········
0020  ff ff 00 43 00 44 01 36  00 00 02 01 06 00 84 e9   ···C·D·6 ········
0030  db e7 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0040  00 00 00 00 00 00 a0 57  e3 27 6b 9e 00 00 00 00   ·······W ·'k·····
```

Flags (3 bits) (ip.flags), 1 byte(s) || 分组: 35102 · 已显示: 40 (0.1%) || 配置: Default

Example: nslookup

- 技巧：先退出WireShark，然后重新打开，再运行“nslookup www.baidu.com”在命令行。

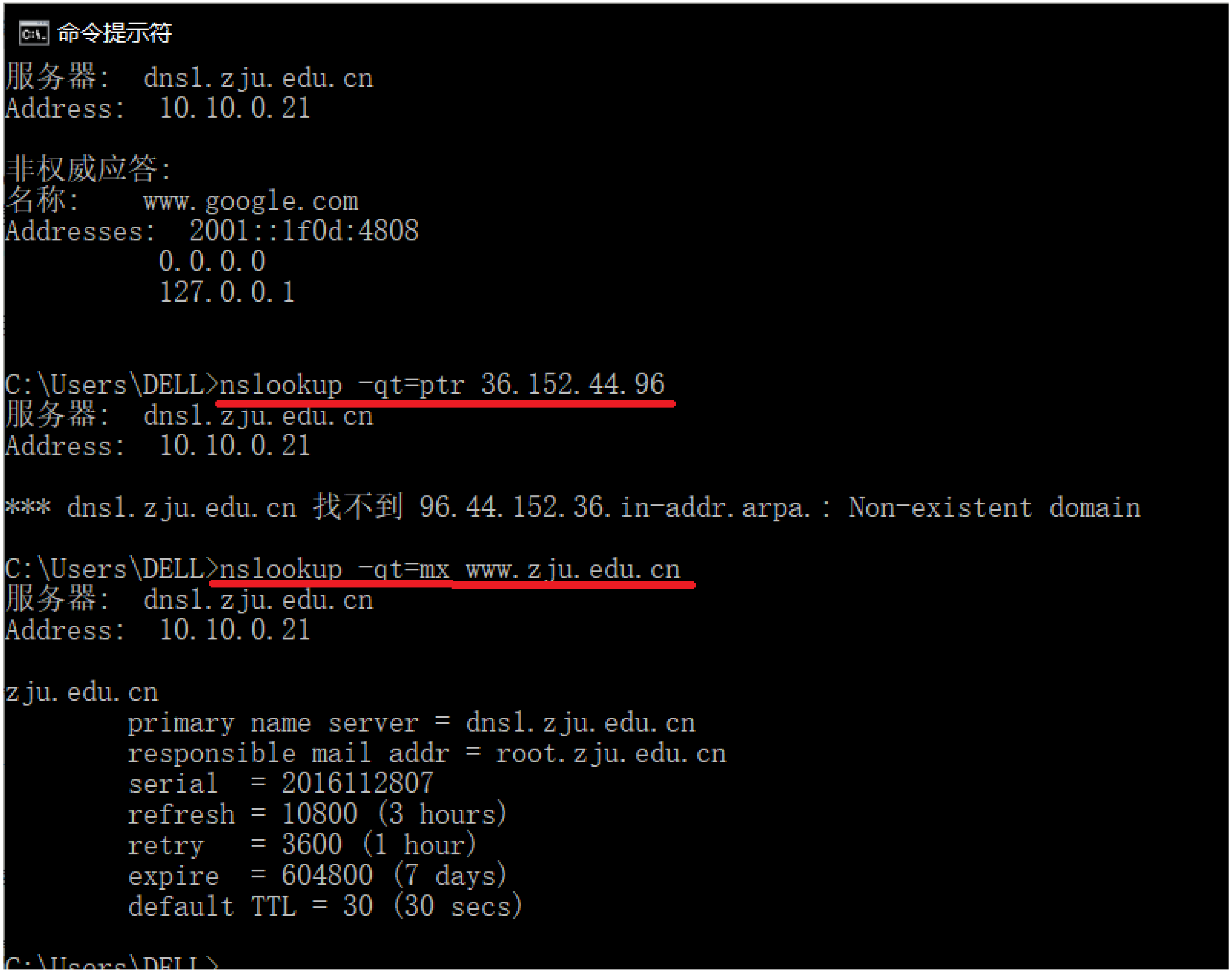


```
Microsoft Windows [版本 10.0.18362.1082]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\Users\DELL>nslookup www.baidu.com
服务器:  dns1.zju.edu.cn
Address:  10.10.0.21

非权威应答:
名称:    www.a.shifen.com
Addresses:  36.152.44.96
          36.152.44.95
Aliases:  www.baidu.com


C:\Users\DELL>
```

Example: nslookup [4]

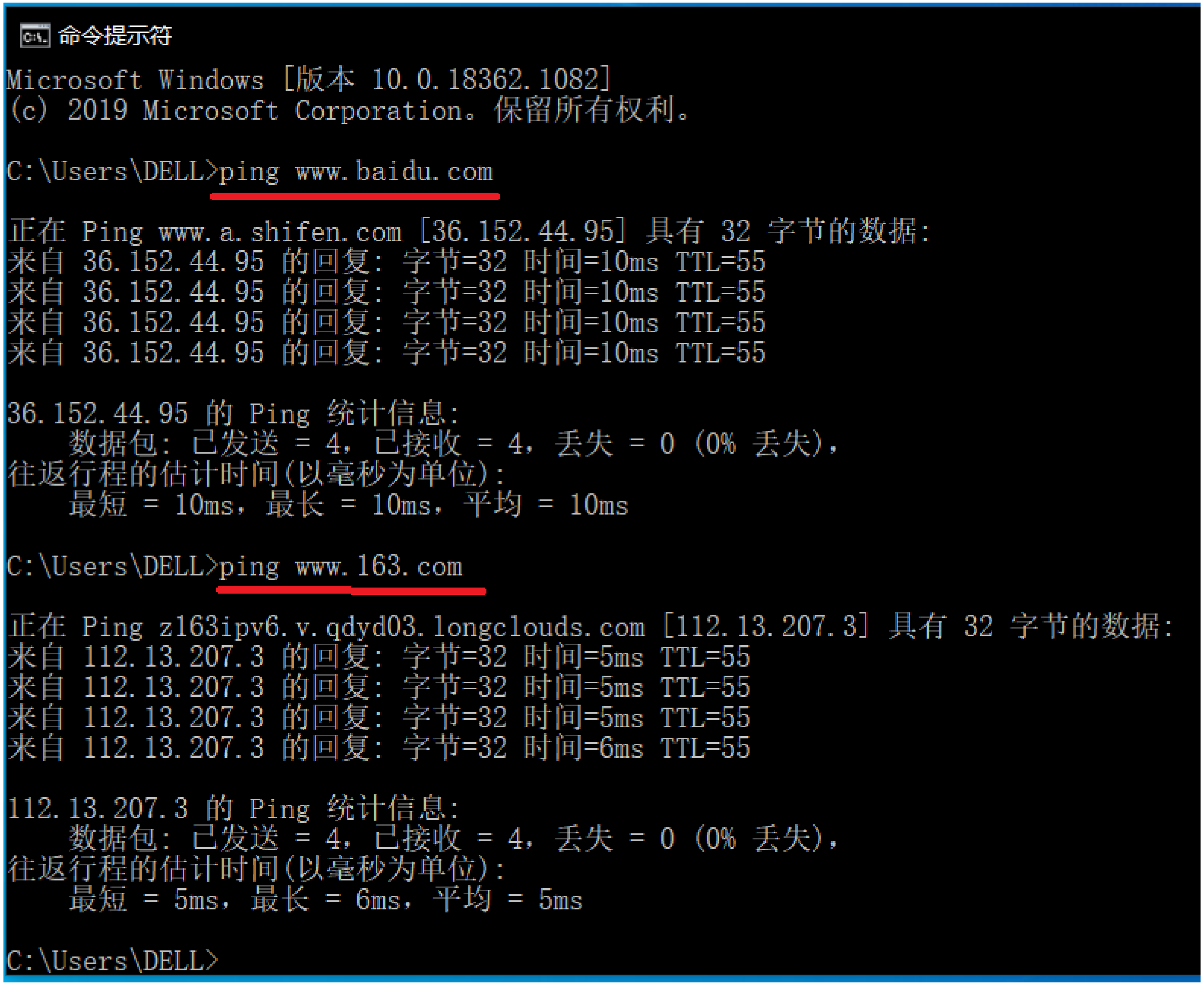
- 正向解析：通过域名查找ip；
- 反向解析：通过ip查找域名；
  - IP反向解析主要应用到邮件服务器中来阻拦垃圾邮件，特别是在国外。多数垃圾邮件发送者使用动态分配或者没有注册域名的IP地址来发送垃圾邮件，以逃避追踪，使用了域名反向解析后，就可以大大降低垃圾邮件的数量。
    - 比如你用 xxx@name.com 这个邮箱给我的邮箱 123@163.com 发了一封信。163邮件服务器接到这封信会查看这封信的信头文件，这封信的信头文件会显示这封信是由哪个IP地址发出来的。然后根据这个IP地址进行反向解析，如果反向解析到这个IP所对应的域名是name.com 那么就接受这封邮件，如果反向解析发现这个IP没有对应到name.com，那么就拒绝这封邮件。

Example: nslookup [4]



```
服务器:  dns1.zju.edu.cn
Address:  10.10.0.21

非权威应答:
名称:    www.google.com
Addresses:  2001::1f0d:4808
          0.0.0.0
          127.0.0.1


C:\Users\DELL>nslookup -qt=ptr 36.152.44.96
服务器:  dns1.zju.edu.cn
Address:  10.10.0.21

*** dns1.zju.edu.cn 找不到 96.44.152.36.in-addr.arpa.: Non-existent domain

C:\Users\DELL>nslookup -qt=mx www.zju.edu.cn
服务器:  dns1.zju.edu.cn
Address:  10.10.0.21

zju.edu.cn
        primary name server = dns1.zju.edu.cn
        responsible mail addr = root.zju.edu.cn
        serial  = 2016112807
        refresh = 10800 (3 hours)
        retry   = 3600 (1 hour)
        expire  = 604800 (7 days)
        default TTL = 30 (30 secs)

C:\Users\DELL>
```

Example: ping (ICMP, Internet Control Message Protocol)



```
Microsoft Windows [版本 10.0.18362.1082]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\Users\DELL>ping www.baidu.com

正在 Ping www.a.shifen.com [36.152.44.95] 具有 32 字节的数据:
来自 36.152.44.95 的回复: 字节=32 时间=10ms TTL=55
来自 36.152.44.95 的回复: 字节=32 时间=10ms TTL=55
来自 36.152.44.95 的回复: 字节=32 时间=10ms TTL=55
来自 36.152.44.95 的回复: 字节=32 时间=10ms TTL=55

36.152.44.95 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 = 10ms，最长 = 10ms，平均 = 10ms

C:\Users\DELL>ping www.163.com

正在 Ping z163ipv6.v.qdyd03.longclouds.com [112.13.207.3] 具有 32 字节的数据:
来自 112.13.207.3 的回复: 字节=32 时间=5ms TTL=55
来自 112.13.207.3 的回复: 字节=32 时间=5ms TTL=55
来自 112.13.207.3 的回复: 字节=32 时间=5ms TTL=55
来自 112.13.207.3 的回复: 字节=32 时间=6ms TTL=55

112.13.207.3 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 = 5ms，最长 = 6ms，平均 = 5ms

C:\Users\DELL>
```

Internet Control Message Protocol (**ICMP**) [2]

- ICMP is specified in RFC 792.
- The most typical use of ICMP is for **error reporting**.
  - For example, when running a Telnet, FTP, or HTTP session, you may have encountered an error message such as "Destination network unreachable".
- ICMP is often considered part of IP but architecturally it lies just above IP, as ICMP messages are carried inside IP datagrams.
- ICMP messages have a type and a code field, and contain the header and the first 8 bytes of the IP datagram.

ICMP Type	Code	Description
0	0	echo reply (to ping)
3	0	destination network unreachable
3	1	destination host unreachable
3	2	destination protocol unreachable
3	3	destination port unreachable
3	6	destination network unknown
3	7	destination host unknown
4	0	source quench (congestion control)
8	0	echo request
9	0	router advertisement
10	0	router discovery
11	0	TTL expired
12	0	IP header bad

Figure 4.23 ♦ ICMP message types

Example: Tracert (ICMP) (I)

- The **Tracert** program, which allows us to trace a route from a host to any other host in the world.
- Tracert is implemented with ICMP messages, to determine the names and addresses of the routers between source and destination,
  - 1) Tracert in the source sends *a series of ordinary IP datagrams* to the destination.
  - Each of these datagrams carries a **UDP** segment with an unlikely UDP port number.
  - The 1st of these datagrams has a TTL of 1, the 2nd of 2, the 3rd of 3, and so on. The source also starts timers for each of the datagrams.

Example: Tracert (ICMP) (II)

- Tracert is implemented with ICMP messages, to determine the names and addresses of the routers between source and destination,
  - 2) When the *n*th datagram arrives at the *n*th router, the *n*th router observes that *the TTL of the datagram has just expired*.
  - According to the rules of the IP protocol, the router discards the datagram and sends an ICMP warning message to the source (type 11 code 0)
  - This warning message includes the name of the router and its IP address.
  - 3) When this ICMP message arrives back at the source, the source obtains **the round-trip time** from the timer and the name and IP address of the *n*th router from the ICMP message

Example: Tracert (ICMP) (III)

- Tracert is implemented with ICMP messages, to determine the names and addresses of the routers between source and destination,
  - 4) How does a Tracert source know when to **stop** sending UDP segments?
  - Recall that the source increments the TTL field for each datagram it sends. Thus, one of the datagrams will eventually make it all the way to the destination host.
  - Because this datagram contains a UDP segment with an unlikely port number, the destination host sends **a port unreachable ICMP message** (type 3 code 3) back to the source.
  - When the source host receives this particular ICMP message, it knows it does not need to send additional probe packets.
  - The standard Tracert program actually sends sets of three packets with the same TTL; thus the Tracert output provides three results for each TTL.

References

- [1] https://www.wireshark.org/
- [2] J. F. Kurose and K.W. Ross, Computer Networking — A Top-down Approach, 5th Edition, Pearson Education Inc., 2010.
- [3] https://blog.csdn.net/gui951753/article/details/83070180 (这个博客中有解释多个站点对应一个IP地址的问题。)
- [4] https://www.cnblogs.com/machangwei-8/p/10353137.html