

电子科技大学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

抓包分析报告



专 业	计算机科学与技术(互联网+)
学 号	2018080901006
作者姓名	刘文晨

目录

一、 WIRESHARK 软件基础	3
二、 实验目的.....	6
三、 层次化封装分析	6
四、 TCP 报文交互时序分析	17
五、 不同类型报文对应的协议及功能	24
六、 流量成分或变化的分析	27
七、 TCP 通信中异常情况整理	43
八、 对 QICQ 包的分析	45
九、 总结及心得体会	46

一、Wireshark 软件基础

1. 软件介绍

Wireshark（前称 Ethereal）是一个网络封包分析软件。网络封包分析软件的功能是撷取网络封包，并尽可能显示出最为详细的网络封包资料。Wireshark 使用 WinPCAP 作为接口，直接与网卡进行数据报文交换。

2. 软件功能

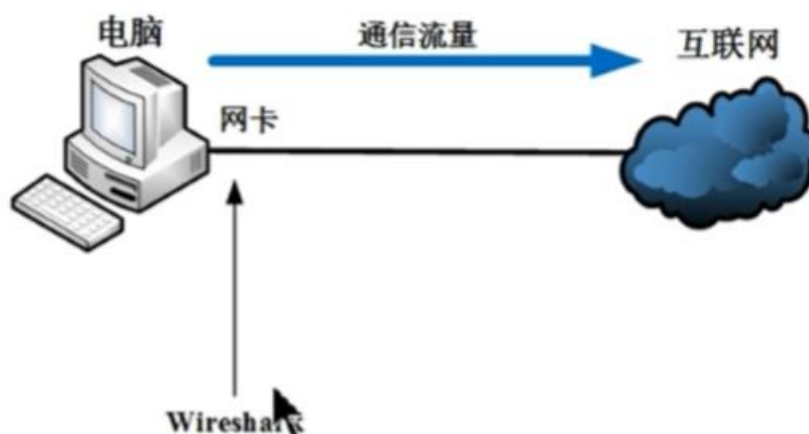
- 分析网络底层协议
- 解决网络故障问题
- 找寻网络安全问题

3. 抓包原理

抓包原理分为网络原理和底层原理。

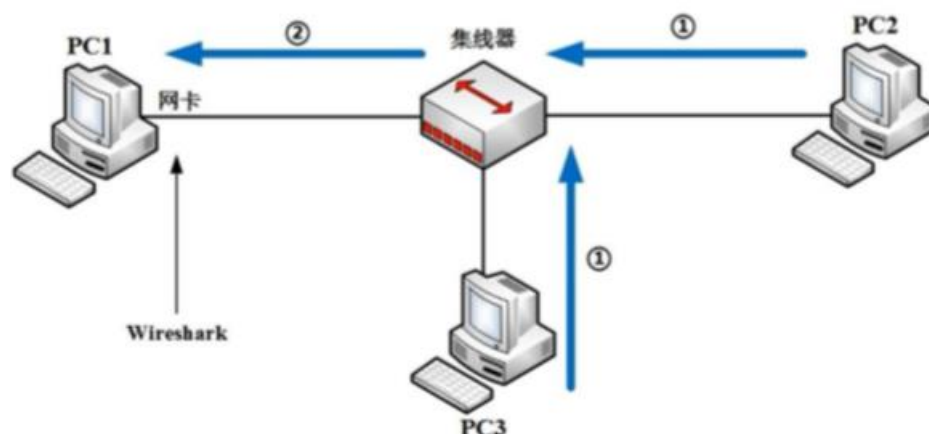
1. 网络原理：

- 1) 本机环境-直接抓本机网卡进出的流量：直接在终端安装 ws，然后 ws 抓本机网卡的与互联网通信的流量。



- 2) 集线器环境（老网络）-集线器：向其他所有端口都会

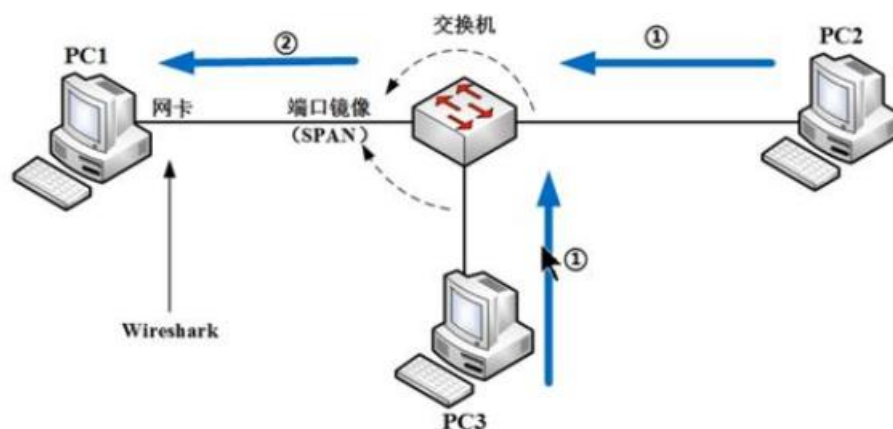
泛洪，抓整个局域网里面的包。



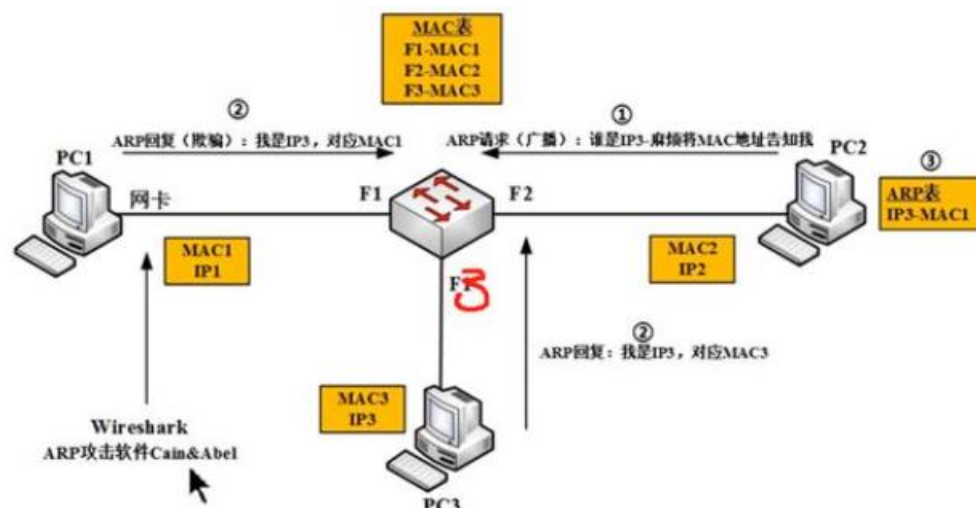
· 集线器- (hub) (多端口的信号放大设备) 属于纯硬件网络底层设备，基本上不具有类似于交换机的"智能记忆"能力和"学习"能力。它也不具备交换机所具有的 MAC 地址表，所以它发送数据时都是没有针对性的，而是采用广播方式发送。也就是说当它要向某节点发送数据时，不是直接把数据发送到目的节点，而是把数据包发送到与集线器相连的所有节点。

3) 交换机环境：

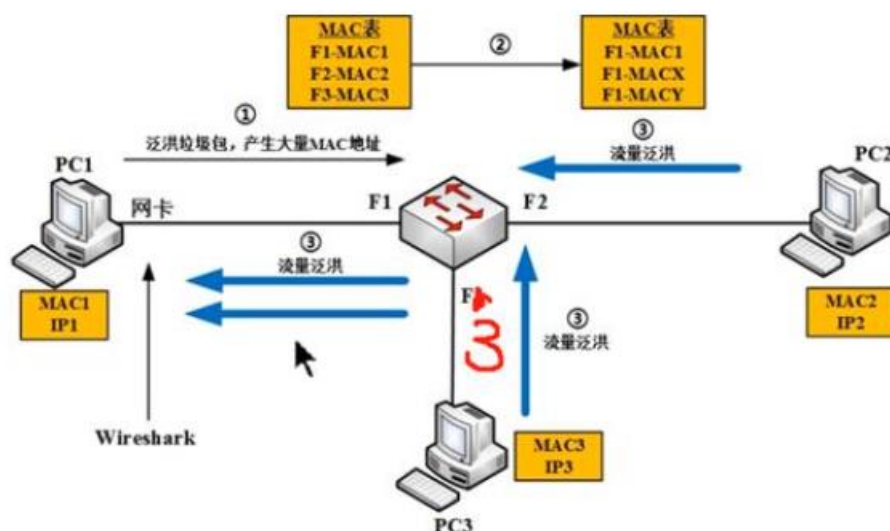
· 端口镜像 (安全)：SPAN 技术，复制其他端口的数据包到特定端口。



- ARP 欺骗 (攻击): 需安装 arp 欺骗软件, 错位欺骗, 如图, PC1 会不断发送欺骗, 毒化 PC2 的 arp 表, 会产生错的绑定, 交换机根据 mac 表就会把数据包乖乖丢给 PC1。

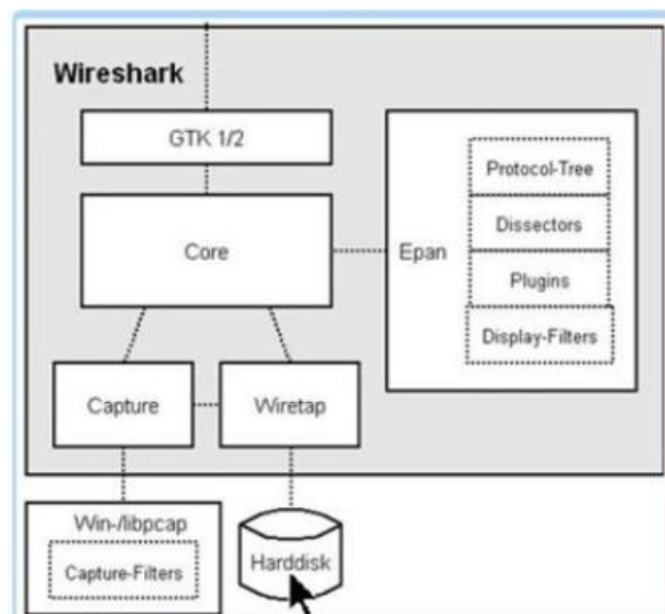


- MAC 泛洪: 泛洪大量垃圾包, 产生大量的 mac 地址, 改变了交换机原有的 mac 地址表, 如图, 这样流量就泛洪给 F1 了。



2. 底层原理: Wireshark 底层抓包工具

结构图如下所示:



二、实验目的

在本次实验中，我们利用 WireShark 软件，抓取自己计算机中的包，选取其中的内容进行分析。本次实验的实验目的有三个：

1. 了解真实应用的网络中，有哪些类型的报文。
2. 了解“封装”的实际效果。
3. 掌握利用 Wireshark 软件进行网络认知和网络分析的基本方法。

三、层次化封装分析

打开 Wireshark 对 WLAN 进行抓包会看到如下展开内容：

No.	Time	Source	Destination	Protocol	Length	Info
118	1.230165	fe80::1cb4:29d0:888...	ff02::fb	MDNS	435	Standard query response 0x0000 PTR iPad (2)..._companion-link._tcp.local TXT TXT, cache flush SRV, cache fL...
119	1.230165	fe80::4a49:e3ff:fe4...	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
120	1.230501	113.54.210.89	224.0.0.251	MDNS	957	Standard query 0x0000 PTR _companion-link._tcp.local, "QM" question TXT \351\200\270\347\232\204\345\260\...
121	1.230501	fe80::1483:4889:3c4...	ff02::fb	MDNS	977	Standard query 0x0000 PTR _companion-link._tcp.local, "QM" question TXT \351\200\270\347\232\204\345\260\...
122	1.231228	182.92.244.228	113.54.209.128	TCP	60	80 → 49765 [FIN, ACK] Seq=1 Ack=1 Win=58 Len=0
123	1.231262	113.54.209.128	182.92.244.228	TCP	54	49765 → 80 [ACK] Seq=1 Ack=2 Win=513 Len=0
124	1.382469	fe80::cbf:cc59:5d8...	ff02::fb	MDNS	505	Standard query response 0x0000 PTR tinoryj\342\200\231s MacBook Pro._companion-link._tcp.local TXT TXT, c...
125	1.382470	113.54.207.250	224.0.0.251	MDNS	485	Standard query response 0x0000 PTR tinoryj\342\200\231s MacBook Pro._companion-link._tcp.local TXT TXT, c...
126	1.382471	113.54.198.226	224.0.0.251	IGMPv2	56	Membership Report group 224.0.0.251
127	1.382882	113.54.209.70	224.0.0.251	MDNS	412	Standard query response 0x0000 PTR easy._companion-link._tcp.local TXT TXT, cache flush SRV, cache flush ...
128	1.382883	fe80::9f:66d2:e3f5...	ff02::fb	MDNS	432	Standard query response 0x0000 PTR easy._companion-link._tcp.local TXT TXT, cache flush SRV, cache flush ...

> Frame 123: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
 > Ethernet II, Src: HomaIPr.8d:47:d3 (80:2b:f9:8d:47:d3), Dst: HuaweiTe.fad:ccc (44:6a:2e:fad:ccc)
 > Internet Protocol Version 4, Src: 113.54.209.128, Dst: 182.92.244.228
 > Transmission Control Protocol, Src Port: 49765, Dst Port: 80, Seq: 1, Ack: 2, Len: 0

其中，113.54.209.128 是 UESTC_WIFI 的 IP 地址。

在开始具体的分析之前，有必要熟悉以下三个面板：

- "Packet List"面板（包列表）

Packet list/包列表面板显示所有当前捕捉的包。列表中的每行显示捕捉文件的一个包。如果您选择其中一行，该包得更多情况会显示在"Packet Detail/包详情"，"Packet Byte/包字节"面板 在分析(解剖)包时，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，您通常在包列表面板看到的都是每个包的最高层协议描述（在这里高层是应用层，底层是数据链路层）。

- "Packet Details"面板（包详情）

"Packet Details/包详情"面板显示当前包(在包列表面板被选中的包)的详情列表。该面板显示包列表面板选中包的协议及协议字段，协议及字段以树状方式组织。你可以展开或折叠它们。右击它们会获得相关的上下文菜单。某些协议字段会以特殊方式显示

- "Packet Byte"面板（包字节）

面板以 16 进制转储方式显示当前选择包的数据。通常在 16 进制转储形式中，左侧显示包数据偏移量，中间栏以 16 进制表示，右侧显示为对应的 ASCII 字符（包数据偏移量是相对第一个包进行偏移）。

下面我们开始具体的分析：

1、 TCP 报文：

点击一个 Protocol 为 TCP 的数据包，查看这个数据包的具体内容，我们发现共有四层：

- Frame：物理层的数据帧概况。
- Ethernet II：数据链路层以太网帧头部信息。
- Internet Protocol Version 4：互联网层 IP 包头部信息。
- Transmission Control Protocol：传输层的数据段头部信息，此处是 TCP 协议。

下面我们将分别对这四层进行分析：

1.1、物理层

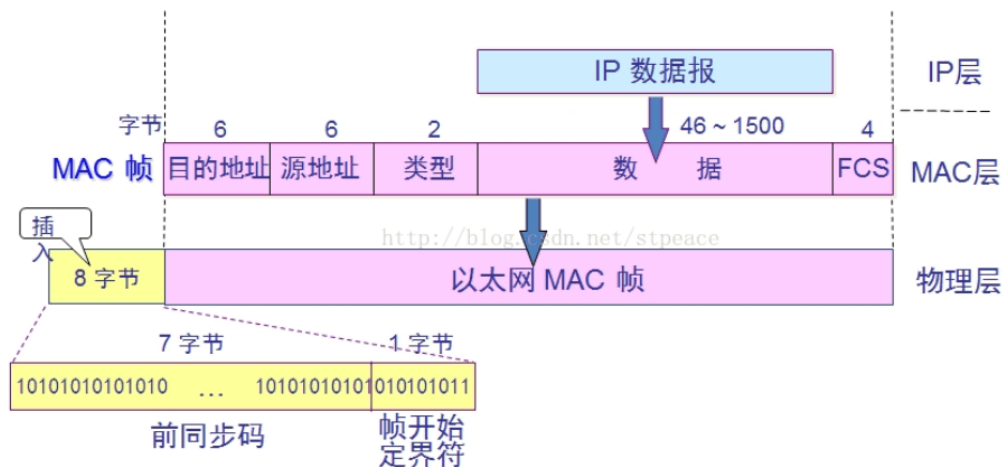
物理层，为设备之间的数据通信提供传输媒体及互连设备。点击 Frame 123 一行，我们能够看到具体传输信息：

```

v Frame 123: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
  > Interface id: 0 (\Device\NPF_{02F498FF-111B-4D71-ADE6-BFCFCDEB2478}) //接口id为0
    Encapsulation type: Ethernet (1) //封装类型
    Arrival Time: Sep  9, 2019 08:31:46.169974000 中国标准时间 //捕获日期和时间
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1567989106.169974000 seconds
    [Time delta from previous captured frame: 0.000034000 seconds] //与前一个包时间相隔
    [Time delta from previous displayed frame: 0.000034000 seconds]
    [Time since reference or first frame: 1.231262000 seconds] //此包与第一帧的时间间隔
    Frame Number: 123 //帧序号
    Frame Length: 54 bytes (432 bits) //帧长度
    Capture Length: 54 bytes (432 bits) //捕获字节长度
    [Frame is marked: False] //是否作了标记
    [Frame is ignored: False] //是否被忽略
    [Protocols in frame: eth:ethertype:ip:tcp] //帧内封装的协议体系结构
    [Coloring Rule Name: HTTP] //着色标记的协议名称
    [Coloring Rule String: http || tcp.port == 80 || http2] //着色规则显示的字符串
  
```

1.2、数据链路层以太网帧头部信息

数据链路层在物理层和网络层的中间层，保证网路层的数据能可靠的透明的在物理层传输。以太网帧格式为：



```

v Ethernet II, Src: HonHaiPr_8d:47:d3 (80:2b:f9:8d:47:d3), Dst: HuaweiTe_fa:dc:cc (44:6a:2e:fa:dc:cc)
  v Destination: HuaweiTe_fa:dc:cc (44:6a:2e:fa:dc:cc) //目的MAC地址
    Address: HuaweiTe_fa:dc:cc (44:6a:2e:fa:dc:cc)
    ....0. .... = LG bit: Globally unique address (factory default)
    ....0. .... = IG bit: Individual address (unicast)
  v Source: HonHaiPr_8d:47:d3 (80:2b:f9:8d:47:d3) //源MAC地址
    Address: HonHaiPr_8d:47:d3 (80:2b:f9:8d:47:d3)
    ....0. .... = LG bit: Globally unique address (factory default)
    ....0. .... = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800) //0x0800表示使用IP协议

```

点击 Ethernet II 一行，我们可以看到更加具体的信息：

```

> Frame 123: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
> Ethernet II, Src: HonHaiPr_8d:47:d3 (80:2b:f9:8d:47:d3), Dst: HuaweiTe_fa:dc:cc (44:6a:2e:fa:dc:cc)
> Internet Protocol Version 4, Src: 113.54.209.128, Dst: 182.92.244.228
> Transmission Control Protocol, Src Port: 49765, Dst Port: 80, Seq: 1, Ack: 2, Len: 0

```

0000	44 6a 2e fa dc cc 80 2b f9 8d 47 d3 08 00 45 00	Dj.....+..G...E.
0010	00 28 68 8d 40 00 80 06 a4 4a 71 36 d1 80 b6 5c	.(h.@....Jq6...\
0020	f4 e4 c2 65 00 50 c4 9d ad 99 34 13 db ec 50 10	...e.P...4...P.
0030	02 01 7a ee 00 00	..Z...

其中，44 6a 2e fa dc cc 是源 MAC 地址，而 80 2b f9 8d 47 d3 是目的 MAC 地址。

1.3、互联网层 IP 包头头部信息

IP 数据报包含两个部分：一个头和一个正文。其中，头是由一个 20 字节的定长部分和一个可选的变长部分组成。如下图：



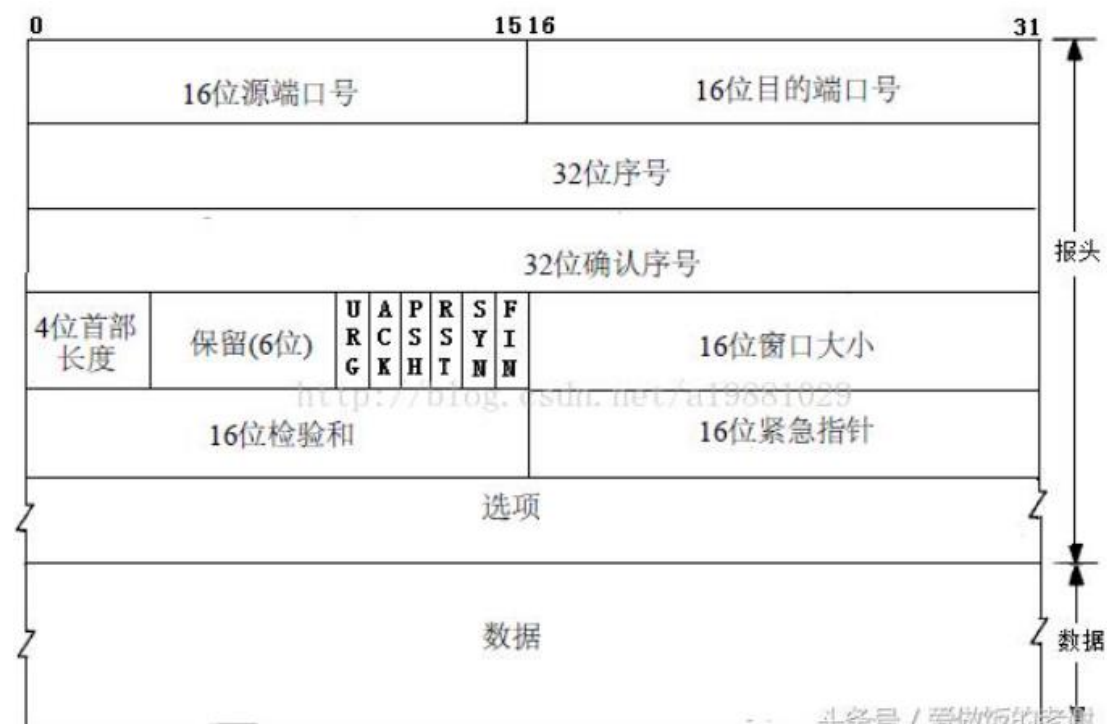
点击 Internet Protocol Version 4 一行，我们可以看到 IPV4 协议各字段的信息：

```

v Internet Protocol Version 4, Src: 113.54.209.128, Dst: 182.92.244.228
  0100 .... = Version: 4 //IPV4协议
  .... 0101 = Header Length: 20 bytes (5) //包头长度
  v Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) //差分服务字段
    0000 00.. = Differentiated Services Codepoint: Default (0)
    .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
  Total Length: 40 //IP包总长度
  Identification: 0x688d (26765) //标志字段
  v Flags: 0x4000, Don't fragment //标记字段
    0... .... .... = Reserved bit: Not set
    .1.. .... .... = Don't fragment: Set
    ..0. .... .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment offset: 0 //分的偏移量
  Time to live: 128 //生存期TTL
  Protocol: TCP (6) //此包内封装的上层协议为TCP
  Header checksum: 0xa44a [validation disabled] //头部数据的校验和
  [Header checksum status: Unverified] //头部数据校验状态
  Source: 113.54.209.128 //源IP地址
  Destination: 182.92.244.228 //目的IP地址
  
```

1.4、传输层的数据段头部信息

TCP 报文格式为：



点击 Transmission Control Protocol 一行，我们可以看到 TCP 协议各字段的信息：

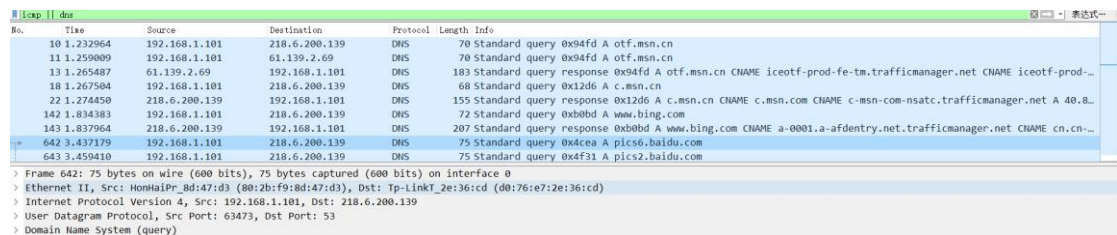
```

▼ Transmission Control Protocol, Src Port: 50180, Dst Port: 443, Seq: 215, Ack: 1461, Len: 0
  Source Port: 50180 //源端口号
  Destination Port: 443 //目标端口号
  [Stream index: 5]
  [TCP Segment Len: 0]
  Sequence number: 215 (relative sequence number) //序列号
  [Next sequence number: 215 (relative sequence number)] //下一个序列号
  Acknowledgment number: 1461 (relative ack number) //确认序列号
  0101 .... = Header Length: 20 bytes (5) //头部长度
  ▼ Flags: 0x010 (ACK) //TCP标记字段
    000. .... = Reserved: Not set
    ...0 .... = Nonce: Not set
    .... 0... = Congestion Window Reduced (CWR): Not set
    .... .0.. = ECN-Echo: Not set
    .... ..0. = Urgent: Not set //紧急指针标志
    .... ...1 = Acknowledgment: Set //确认号字段
    .... .... 0... = Push: Not set //推送功能
    .... .... .0.. = Reset: Not set //重置连接
    .... .... ..0. = Syn: Not set //同步序列号
    .... .... ...0 = Fin: Not set
    [TCP Flags: .....A....]
  Window size value: 1024 //窗口大小
  [Calculated window size: 262144]
  [Window size scaling factor: 256]
  Checksum: 0xa09f [unverified] //TCP数据段的校验和
  [checksum Status: Unverified]
  Urgent pointer: 0
  ▼ [SEQ/ACK analysis]
    [This is an ACK to the segment in frame: 59]
    [The RTT to ACK the segment was: 0.000049000 seconds]
    [iRTT: 0.186984000 seconds]
  ▼ [Timestamps]
    [Time since first frame in this TCP stream: 0.289414000 seconds]
    [Time since previous frame in this TCP stream: 0.000049000 seconds]

```

2、 DNS 报文

在过滤器中输入 icmp || dns 过滤出 Protocol 为 DNS 的数据包，如下图所示：



No.	Time	Source	Destination	Protocol	Length	Info
10	1.232964	192.168.1.101	218.6.200.139	DNS	70	Standard query 0x94fd A oftf.msn.cn
11	1.259080	192.168.1.101	61.139.2.69	DNS	70	Standard query 0x94fd A oftf.msn.cn
13	1.265487	61.139.2.69	192.168.1.101	DNS	183	Standard query response 0x94fd A oftf.msn.cn CNAME iceotf-prod-fe-tm.trafficmanager.net CNAME iceotf-prod-...
18	1.267504	192.168.1.101	218.6.200.139	DNS	68	Standard query 0x12d6 A c.msn.cn
22	1.274450	218.6.200.139	192.168.1.101	DNS	155	Standard query response 0x12d6 A c.msn.cn CNAME c.msn.com CNAME c.msn-com-nsatc.trafficmanager.net A 40.8...
142	1.834383	192.168.1.101	218.6.200.139	DNS	72	Standard query 0xb0bd A www.bing.com
143	1.837964	218.6.200.139	192.168.1.101	DNS	207	Standard query response 0xb0bd A www.bing.com CNAME a-0001.a-afdentry.net.trafficmanager.net CNAME cn.cn...
642	3.437179	192.168.1.101	218.6.200.139	DNS	75	Standard query 0x4cea A pics6.baidu.com
643	3.459410	192.168.1.101	218.6.200.139	DNS	75	Standard query 0x4f31 A pics2.baidu.com

▶ Frame 642: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface 0
 ▶ Ethernet II, Src: HonHaiPr_8d:47:d3 (80:2b:f9:8d:47:d3), Dst: Tp-LinkT_2e:36:cd (d0:76:e7:2e:36:cd)
 ▶ Internet Protocol Version 4, Src: 192.168.1.101, Dst: 218.6.200.139
 ▶ User Datagram Protocol, Src Port: 63473, Dst Port: 53
 ▶ Domain Name System (query)

随机点击一个数据包，查看这个数据包的具体内容，我们发现共有五层：

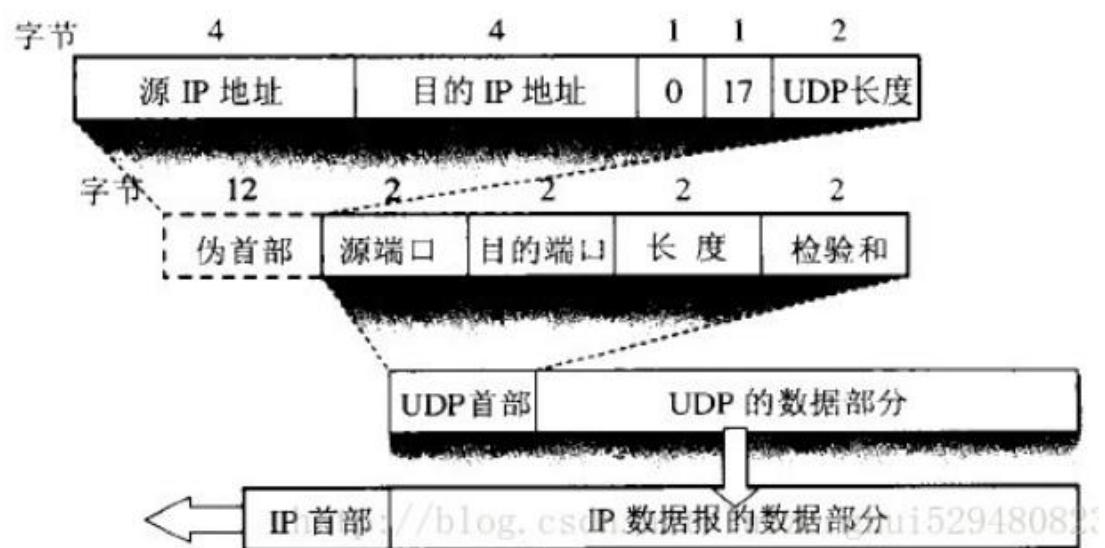
- Frame：物理层的数据帧概况。
- Ethernet II：数据链路层以太网帧头部信息。

- Internet Protocol Version 4: 互联网层 IP 包头部信息。
- User Datagram Protocol: 传输层的数据段头部信息，此处是 UDP 协议。
- Domain Name System(query): 应用层头部信息，此处是 DNS 协议。

因为前三层在 TCP 报文中已经分析过了，下面我们将分别对后两层进行分析：

2.1、传输层的数据段头部信息

UDP 报文格式为：



点击 User Datagram Protocol 一行，我们可以看到 UDP 协议各字段的信息：

```

v User Datagram Protocol, Src Port: 63473, Dst Port: 53
  Source Port: 63473 //源端口
  Destination Port: 53 //目标端口
  Length: 41 //数据报长度
  Checksum: 0x32cf [unverified] //校验和
  [Checksum Status: Unverified]
  [Stream index: 8]
v [Timestamps]
  [Time since first frame: 0.000000000 seconds]
  [Time since previous frame: 0.000000000 seconds]

```

2.2、应用层头部信息

DNS 报文格式为：



DNS协议报文格式

点击 Domain Name System (query) 一行，我们可以看到 DNS 协议各字段的信息：

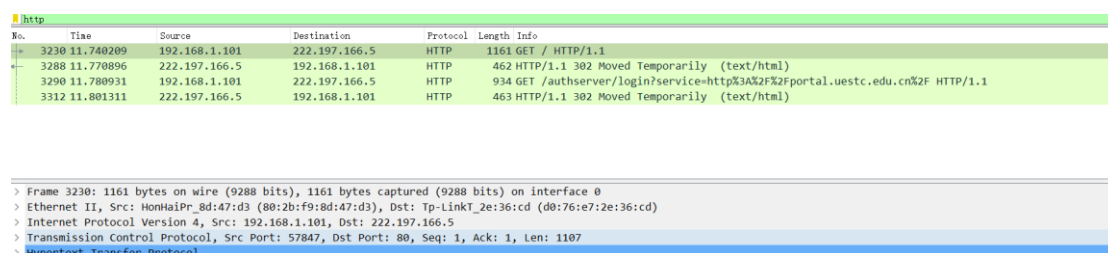

```

Domain Name System (query)
  Transaction ID: 0x4cea //标识字段
  Flags: 0x0100 Standard query //标志字段
    0... .. = Response: Message is a query
    .000 0... .. = Opcode: Standard query (0)
    .... ..0. .... = Truncated: Message is not truncated
    .... ..1 .... = Recursion desired: Do query recursively
    .... ..0.. .... = Z: reserved (0)
    .... ..0 .... = Non-authenticated data: Unacceptable
  Questions: 1 //问题数
  Answer RRs: 0 //资源记录数
  Authority RRs: 0 //授权资源记录数
  Additional RRs: 0 //额外资源记录数
  Queries
    > pics6.baidu.com: type A, class IN
    [Response In: 648]

```

3、 HTTP 报文

在过滤器中输入 http 过滤出 Protocol 为 HTTP 的数据包，如下图所示：



随机点击一个 HTTP 数据包，查看这个数据包的具体内容，我们发现共有四层：

- Frame：物理层的数据帧概况。
- Ethernet II：数据链路层以太网帧头部信息。
- Internet Protocol Version 4：互联网层 IP 包头部信息。
- Hypertext Transfer Protocol：应用层头部信息，此处是 HTTP 协议。

因为前三层在 TCP 报文中已经分析过了，下面我们将分别对最后一层进行分析：

3.1、应用层头部信息

HTTP 报文分为请求报文和响应报文。

HTTP 请求报文格式为：



HTTP 响应报文格式为：



右键点击这个数据包，选择追踪流-->HTTP 流，弹出窗口就是完整的 HTTP 报文，红色字体为 HTTP 请求报文，蓝色字体为 HTTP 响应报文：


```

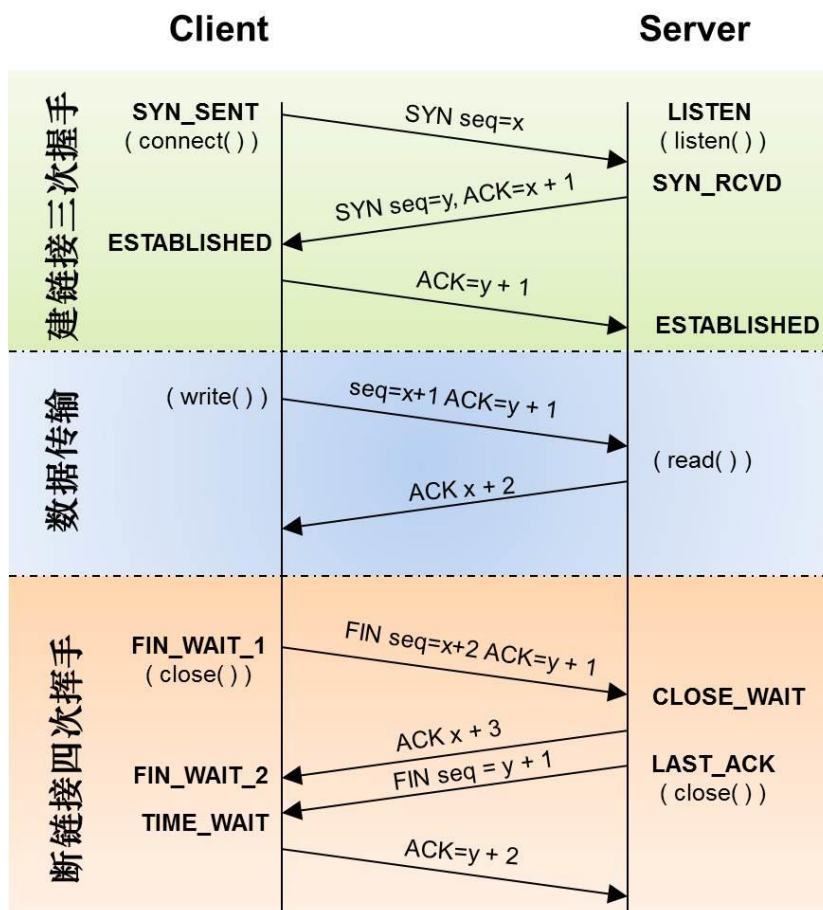
GET / HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.102 Safari/537.36 Edge/18.18362 //请求的浏览器类型
Accept-Encoding: gzip, deflate //客户端可接受的编码压缩格式
Host: portal.uestc.edu.cn //请求的主机名
Connection: Keep-Alive //连接方式
Cookie: __utmsz=108824541.1554128841.2.2.utmscr=idas.uestc.edu.cn|utmccn=(referral)|utmcmd=referral|utmcct=/authserver/login;__utma=108824541.331175326.1552987814.1567954489.1567992473.24;UM_distinctid=169d94c33d978-0d3df7c9365c34-784a5037-1a25b6-169d94c33da12;FSSBB11Ugzbn7N80T=44MqirourzD.YKxpy5azZy30usfDs8JTVuEW2s0qWDEAZH.A9EJ238jufc_qMOepMRFTi1vYdt0_Tk_b8PMLQLBHKJqY0y2nqYW7zFcF6f4ka0V7wnIqRPEgiX5pZ3LcTno6wtpUTYl.t_vNZiO7FeR8Le_Q_c1HbBf9OwBV8zX.NS18ZKXenTVJ6r82BNDqhffWxCba830UvvHAAw10uhQfguquNV0jOr18Qy6izCln_4q5DKT8Zz0EN1IqK_LXLgHnZTM_FGew.J7bOULScjHqzk1Wp.WIWS1xf7tnb3fD6tNfb4dPiKp0_D2weBwC1g_jRRUK4h8ScasA20Pw_DSHBYeLeq1aJg_qdmx10_qbnq; FSSBB11Ugzbn7N80S=.Yd0VPuk5gz1tL15nSWiIdIjerf1Edz4FMNEhrMMXUR29IOkP9kUTmvSdhaCkQ0
//存储于客户端扩展字段，向同一域名的服务端发送属于该域的cookie
HTTP/1.1 302 Moved Temporarily
Server: ***** //Server响应报头域包含了服务器用来处理请求的软件信息及其版本
Content-Type: text/html
Content-Length: 154
Connection: keep-alive //连接方式
Date: Tue, 10 Sep 2019 09:09:51 GMT
Location: http://idas.uestc.edu.cn/authserver/login?service=http%3A%2F%2Fportal.uestc.edu.cn%2F

<html>
<head><title>302 Found</title></head>
<body bgcolor="white">
<center><h1>302 Found</h1></center>
<hr><center>nginx</center>
</body>
</html>

```

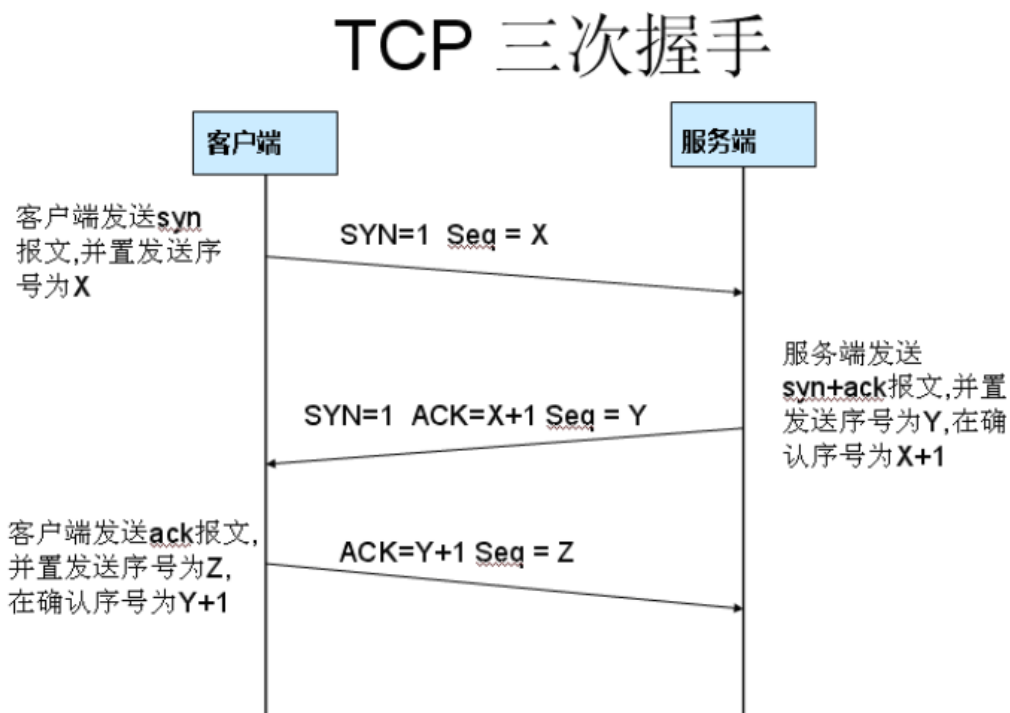
四、 TCP 报文交互时序分析

TCP 通信流程大致如下：



1、 建立连接协议（三次握手）：

- 1.1、 第一次握手：客户端发送 syn 包（ $\text{syn}=x$ ）的数据包到服务器，并进入 SYN_SEND 状态，等待服务器确认；
- 1.2、 第二次握手：服务器收到 syn 包，必须确认客户的 SYN（ $\text{ack}=x+1$ ），同时自己也发送一个 SYN 包（ $\text{syn}=y$ ），即 SYN+ACK 包，此时服务器进入 SYN_RECV 状态；
- 1.3、 第三次握手：客户端收到服务器的 SYN + ACK 包，向服务器发送确认包 ACK($\text{ack}=y+1$)，此包发送完毕，客户端和服务端进入 ESTABLISHED 状态，完成三次握手。



在 Wireshark 软件中，为了在成百上千个数据包中发现这三次握手过程，我们在显示过滤框输入： $\text{ip.dst}=192.168.1.100$ or $\text{ip.src}=192.168.1.100$ ，这样做的目的是为了得到与浏览器打开网站

相关的数据包。

于是我们得到如下图所示的数据包：

No.	Time	Source	Destination	Protocol	Length	Info
9	5.632414	192.168.1.100	117.184.242.221	TCP	66	55624 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=2 SACK_PERM=1
10	5.733566	117.184.242.221	192.168.1.100	TCP	66	443 → 55624 [SYN, ACK] Seq=0 Ack=1 Win=14400 Len=0 MSS=1440 SACK_PERM=1 WS=128
11	5.733748	192.168.1.100	117.184.242.221	TCP	54	55624 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0
12	5.734472	157.255.173.227	192.168.1.100	TCP	54	443 → 55621 [FIN, ACK] Seq=1 Ack=2 Win=130 Len=0
13	5.734583	192.168.1.100	157.255.173.227	TCP	54	55621 → 443 [ACK] Seq=2 Ack=2 Win=32483 Len=0
14	5.734953	220.194.111.242	192.168.1.100	TCP	54	443 → 55622 [FIN, ACK] Seq=1 Ack=2 Win=130 Len=0
15	5.735058	192.168.1.100	220.194.111.242	TCP	54	55622 → 443 [ACK] Seq=2 Ack=2 Win=32483 Len=0
16	5.735339	192.168.1.100	117.184.242.221	TLSv1.2	385	Client Hello
17	5.834902	117.184.242.221	192.168.1.100	TCP	54	443 → 55624 [ACK] Seq=1 Ack=332 Win=15488 Len=0
18	5.843065	117.184.242.221	192.168.1.100	TLSv1.2	1494	Server Hello
19	5.843066	117.184.242.221	192.168.1.100	TCP	1494	443 → 55624 [ACK] Seq=1441 Ack=332 Win=15488 Len=1440 [TCP segment of a reassembled PDU]
20	5.843066	117.184.242.221	192.168.1.100	TLSv1.2	528	Certificate, Server Key Exchange, Server Hello Done

从中找出 Wireshark 截获到的三次握手过程中的三个数据包：

9	5.632414	192.168.1.100	117.184.242.221	TCP	66	55624 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=2 SACK_PERM=1
10	5.733566	117.184.242.221	192.168.1.100	TCP	66	443 → 55624 [SYN, ACK] Seq=0 Ack=1 Win=14400 Len=0 MSS=1440 SACK_PERM=1 WS=128
11	5.733748	192.168.1.100	117.184.242.221	TCP	54	55624 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0

i. 第一次握手：

客户端发送一个 TCP，标志位为 SYN，序列号为 0，代表客户端请求建立连接。如下图：

>	Frame 9: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
>	Ethernet II, Src: HonHaiPr_8d:47:d3 (80:2b:f9:8d:47:d3), Dst: Tp-LinkT_2e:36:cd (d0:76:e7:2e:36:cd)
>	Internet Protocol Version 4, Src: 192.168.1.100, Dst: 117.184.242.221
>	Transmission Control Protocol, Src Port: 55624, Dst Port: 443, Seq: 0, Len: 0
	Source Port: 55624
	Destination Port: 443
	[Stream index: 6]
	[TCP Segment Len: 0]
	Sequence number: 0 (relative sequence number)
	[Next sequence number: 0 (relative sequence number)]
	Acknowledgment number: 0
	1000 = Header Length: 32 bytes (8)
>	Flags: 0x002 (SYN)
	Window size value: 65535
	[Calculated window size: 65535]

ii. 第二次握手：

服务器发回确认包，标志位为 SYN，ACK。将确认序号设置为客户的 ISN 加 1，即 0+1=1，如下图：

```

> Frame 10: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
> Ethernet II, Src: Tp-LinkT_2e:36:cd (d0:76:e7:2e:36:cd), Dst: HonHaiPr_8d:47:d3 (80:2b:f9:8d:47:d3)
> Internet Protocol Version 4, Src: 117.184.242.221, Dst: 192.168.1.100
v Transmission Control Protocol, Src Port: 443, Dst Port: 55624, Seq: 0, Ack: 1, Len: 0
    Source Port: 443
    Destination Port: 55624
    [Stream index: 6]
    [TCP Segment Len: 0]
    Sequence number: 0 (relative sequence number)
    [Next sequence number: 0 (relative sequence number)]
    Acknowledgment number: 1 (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
> Flags: 0x012 (SYN, ACK)
    Window size value: 14400
    [Calculated window size: 14400]

```

iii. 第三次握手:

客户端再次发送确认包(ACK), SYN 标志位为 0, ACK 标志位为 1。并且把服务器发来 ACK 的序号字段+1, 放在确定字段中发送给对方。并且在数据段放写 ISN 的+1, 如下图:

```

> Frame 11: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
> Ethernet II, Src: HonHaiPr_8d:47:d3 (80:2b:f9:8d:47:d3), Dst: Tp-LinkT_2e:36:cd (d0:76:e7:2e:36:cd)
> Internet Protocol Version 4, Src: 192.168.1.100, Dst: 117.184.242.221
v Transmission Control Protocol, Src Port: 55624, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
    Source Port: 55624
    Destination Port: 443
    [Stream index: 6]
    [TCP Segment Len: 0]
    Sequence number: 1 (relative sequence number)
    [Next sequence number: 1 (relative sequence number)]
    Acknowledgment number: 1 (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
> Flags: 0x010 (ACK)
    Window size value: 32768
    [Calculated window size: 65536]

```

就这样通过了 TCP 三次握手, 建立了连接。

2、 数据传输:

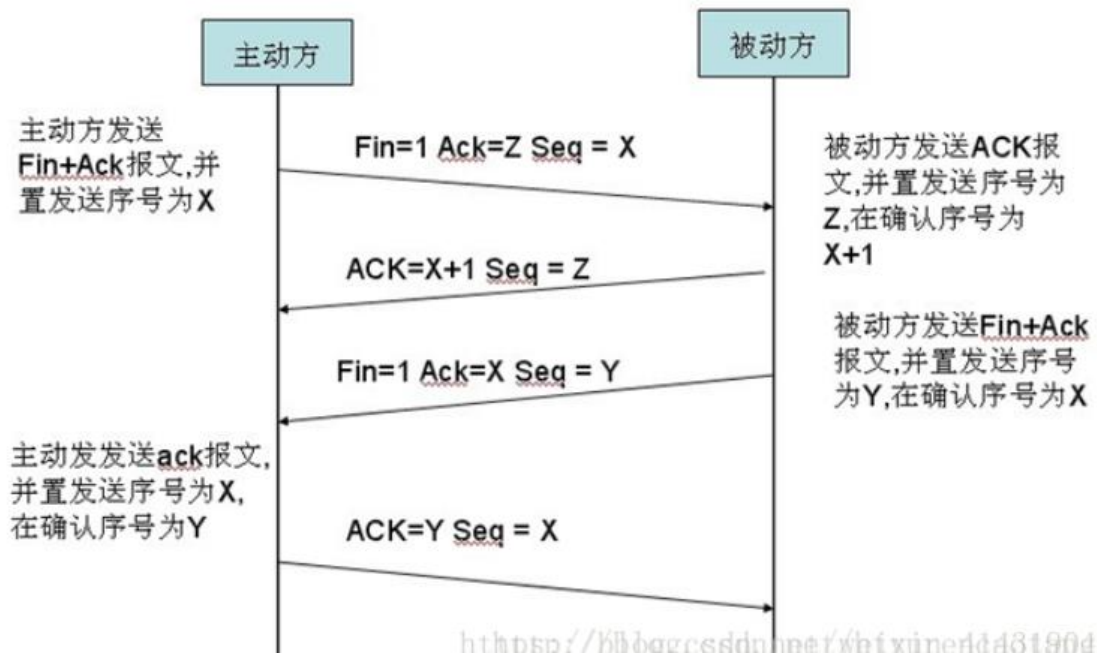
握手过程中传送的包里不包含数据, 三次握手完毕后, 客户端与服务器才正式开始传送数据。理想状态下, TCP 连接一旦建立, 在通信双方中的任何一方主动关闭连接之前, TCP 连接都将被一直保持下去。

3、 连接终止协议 (四次挥手):

由于 TCP 连接是全双工的，因此每个方向都必须单独进行关闭。这原则是当一方完成它的数据发送任务后就能发送一个 FIN 来终止这个方向的连接。收到一个 FIN 只意味着这一方向上没有数据流动，一个 TCP 连接在收到一个 FIN 后仍能发送数据。首先进行关闭的一方将执行主动关闭，而另一方执行被动关闭。

- 3.1、第一次挥手：主动关闭方发送一个 FIN，用来关闭主动方到被动关闭方的数据传送（当然，在 fin 包之前发送出去的数据，如果没有收到对应的 ack 确认报文，主动关闭方依然会重发这些数据）。但是，此时主动关闭方还可以接受数据。
- 3.2、第二次挥手：被动关闭方收到 FIN 包后，发送一个 ACK 给对方，确认序号为收到序号+1（与 SYN 相同，一个 FIN 占用一个序号，SYN 和 FIN 都有 seq 序号）。
- 3.3、第三次挥手：被动关闭方发送一个 FIN，用来关闭被动关闭方到主动关闭方的数据传送。
- 3.4、第四次挥手：主动关闭方收到 FIN 后，发送一个 ACK 给被动关闭方，确认序号为收到序号+1，至此，完成四次挥手。

TCP 四次挥手



在 Wireshark 软件中，为了在成百上千个数据包中发现这三次握手过程，我们在显示过滤框输入：ip.dst==192.168.1.102 or ip.src==192.168.1.102，这样做的目的是为了得到与浏览器打开网站相关的数据包。

我们找到一个属于挥手过程的数据包，右键点击这个数据包，选择追踪流-->TCP 流，于是我们得到了四次挥手的数据包：

tcp.stream eq 11						
No.	Time	Source	Destination	Protocol	Length	Info
78	2.645763	192.168.1.102	202.89.233.96	TCP	54	50946 → 443 [FIN, ACK] Seq=1 Ack=1 Win=1023 Len=0
93	2.689959	202.89.233.96	192.168.1.102	TCP	54	443 → 50946 [ACK] Seq=1 Ack=2 Win=1024 Len=0
96	2.690449	202.89.233.96	192.168.1.102	TCP	54	443 → 50946 [FIN, ACK] Seq=1 Ack=2 Win=1024 Len=0
98	2.690531	192.168.1.102	202.89.233.96	TCP	54	50946 → 443 [ACK] Seq=2 Ack=2 Win=1023 Len=0

i. 第一次挥手：

```

v Transmission Control Protocol, Src Port: 50946, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
  Source Port: 50946
  Destination Port: 443
  [Stream index: 11]
  [TCP Segment Len: 0]
  Sequence number: 1 (relative sequence number)
  [Next sequence number: 1 (relative sequence number)]
  Acknowledgment number: 1 (relative ack number)
  0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x011 (FIN, ACK)
  Window size value: 1023
  [Calculated window size: 1023]
  [Window size scaling factor: -1 (unknown)]
  Checksum: 0x2629 [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  v [Timestamps]
    [Time since first frame in this TCP stream: 0.000000000 seconds]
    [Time since previous frame in this TCP stream: 0.000000000 seconds]

```

发送 FIN ACK 报文, Seq=1,Ack=1。

ii. 第二次挥手:

```

v Transmission Control Protocol, Src Port: 443, Dst Port: 50946, Seq: 1, Ack: 2, Len: 0
  Source Port: 443
  Destination Port: 50946
  [Stream index: 11]
  [TCP Segment Len: 0]
  Sequence number: 1 (relative sequence number)
  [Next sequence number: 1 (relative sequence number)]
  Acknowledgment number: 2 (relative ack number)
  0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
  Window size value: 1024
  [Calculated window size: 1024]
  [Window size scaling factor: -1 (unknown)]
  Checksum: 0x2628 [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  > [SEQ/ACK analysis]
  v [Timestamps]
    [Time since first frame in this TCP stream: 0.044196000 seconds]

```

发送 ACK 报文, Seq=1,Ack+1=2。

iii. 第三次挥手:


```

v Transmission Control Protocol, Src Port: 443, Dst Port: 50946, Seq: 1, Ack: 2, Len: 0
  Source Port: 443
  Destination Port: 50946
  [Stream index: 11]
  [TCP Segment Len: 0]
  Sequence number: 1 (relative sequence number)
  [Next sequence number: 1 (relative sequence number)]
  Acknowledgment number: 2 (relative ack number)
  0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x011 (FIN, ACK)
  Window size value: 1024
  [Calculated window size: 1024]
  [Window size scaling factor: -1 (unknown)]
  Checksum: 0x2627 [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  v [Timestamps]
    [Time since first frame in this TCP stream: 0.044686000 seconds]
    [Time since previous frame in this TCP stream: 0.000490000 seconds]

```

发送 ACK 报文, Seq=1,Ack=2。

iv. 第四次挥手:

```

v Transmission Control Protocol, Src Port: 50946, Dst Port: 443, Seq: 2, Ack: 2, Len: 0
  Source Port: 50946
  Destination Port: 443
  [Stream index: 11]
  [TCP Segment Len: 0]
  Sequence number: 2 (relative sequence number)
  [Next sequence number: 2 (relative sequence number)]
  Acknowledgment number: 2 (relative ack number)
  0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
  Window size value: 1023
  [Calculated window size: 1023]
  [Window size scaling factor: -1 (unknown)]
  Checksum: 0x2628 [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  > [SEQ/ACK analysis]
  > [Timestamps]

```

发送 FIN ACK 报文, Seq+1=2,Ack=1。

就这样通过 TCP 四次挥手, 终止了连接。

五、 不同类型报文对应的协议及功能

1、 TCP 报文：

对应协议为传输控制协议，其功能是在不可靠的互联网络上提供可靠的端到端字节流。

2、 UDP 报文：

对应协议是用户数据报协议，其功能是将网络数据流量压缩成数据包的形式。一个典型的数据包就是一个二进制数据的传输单位。每一个数据包的前 8 个字节用来包含报头信息，剩余字节则用来包含具体的传输数据。

3、 IP 报文：

对应协议是网际协议，其功能是定义了数据传输时的基本单元和格式，还定义了数据报的递交方法和路由选择。此外，在 TCP/IP 网络中，主机之间进行通信所必需的地址，也是通过 IP 协议来实现的。

4、 DNS 报文：

对应协议是域名解析协议，其功能是把域名指向网站空间 IP，让人们通过注册的域名可以方便地访问到网站。互联网中的地址是数字的 IP 地址，域名解析的作用主要就是为了便于记忆。

5、 HTTP 报文：

对应协议是超文本传输协议，其功能是指定客户端可能发送给服务器什么样的消息以及得到什么样的响应。

6、 ARP 报文

对应协议是地址解析协议，其功能是根据 IP 地址获取物理地

址。主机发送信息时将包含目标 IP 地址的 ARP 请求广播到局域网络上的所有主机，并接收返回消息，以此确定目标的物理地址；收到返回消息后将该 IP 地址和物理地址存入本机 ARP 缓存中并保留一定时间，下次请求时直接查询 ARP 缓存以节约资源。地址解析协议是建立在网络中各个主机互相信任的基础上的，局域网络上的主机可以自主发送 ARP 应答消息，其他主机收到应答报文时不会检测该报文的真实性就会将其记入本机 ARP 缓存；由此攻击者就可以向某一主机发送伪 ARP 应答报文，使其发送的信息无法到达预期的主机或到达错误的主机，这就构成了一个 ARP 欺骗。ARP 命令可用于查询本机 ARP 缓存中 IP 地址和 MAC 地址的对应关系、添加或删除静态对应关系等。相关协议有 RARP、代理 ARP。NDP 用于在 IPv6 中代替地址解析协议。

7、 ICMP 报文

对应协议是控制报文协议，其功能是 IP 主机、路由器之间传递控制消息。控制消息是指网络通不通、主机是否可达、路由是否可用等网络本身的消息。这些控制消息虽然并不传输用户数据，但是对于用户数据的传递起着重要的作用。

8、 BOOTP 报文

对应协议是一种引导协议，基于 IP/UDP 协议，也称自举协议，是 DHCP 协议的前身。BOOTP 用于无盘工作站的局域网中，可以让无盘工作站从一个中心服务器上获得 IP 地址。通过

BOOTP 协议可以为局域网中的无盘工作站分配动态 IP 地址，这样就不需要管理员去为每个用户去设置静态 IP 地址。

六、 流量成分或变化的分析

1、流量成分分析

经过上面的分析，我们发现数据包之间的差异体现在协议、端口、分组长度等方面上。为了更好地分析流量成分及变化，我们利用 Wireshark 抓取了 1005 个数据包。

为了对这 1005 个数据包的属性有初步的了解，我们打开统计→捕获文件属性，该窗口的上半部分包含以下内容：

- 文件：通过该区域中的信息，可以了解抓包文件的各种属性，比如，抓包文件的名称、路径信息，以及抓包文件所含数据包的“规模”等信息。
- 时间：通过该区域中的信息，可以获悉抓包的开始、结束以及持续时间。
- 捕获：通过该区域中的信息，可以得知安装了 Wireshark 的主机的硬件及操作系统信息。
- 接口：通过该区域中的信息，可以了解到有关抓包网卡的信息，包括该网卡在操作系统注册表中的信息（左侧）、在抓包时是否启用了抓包过滤器、网卡类型以及对所抓数据包大小的限制。
- 统计：通过该区域中的信息，可以了解到本次抓包的常规统计信息，比如，所抓数据包的数量、Wireshark 显示出的数据包的数量等。

我们打开统计→已解析的地址，该窗口包含以下内容：

- Comment（注释信息）：若要查看注释信息，请点击 Show 按钮，激活 Comment 菜单项。
- hosts（IPv4/IPv6 地址 DNS 解析信息）：提供了所抓数据包的 IP 地址的 DNS 名称。
- IPv4/IPv6 Hash Table（IPv4/IPv6 地址哈希表）：提供了所抓数据包的 IP 地址的哈希值。
- Service（服务或应用程序名称信息）：提供了公认的 TCP 或 UDP 端口名称。
- Ethernet Addresses、Ethernet Manufacturers 和 Ethernet Well-Known Addressses：提供了 MAC 地址信息以及拥有 MAC 地址的网卡制造商的信息。

在对这 1005 个数据包的属性有初步的了解之后，我们开始详细的分析：

首先我们打开统计→协议分级，我们可以看到各协议按分组百分比分类及按字节百分比分类的内容：

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	结束 位/秒
▼ Frame	100.0	1005	100.0	355767	40 k	0	0	0
▼ Ethernet	100.0	1005	4.0	14070	1588	0	0	0
▼ Internet Protocol Version 4	98.5	990	5.6	19800	2235	0	0	0
▼ User Datagram Protocol	25.3	254	0.6	2032	229	0	0	0
Simple Service Discovery Protocol	2.6	26	1.9	6846	773	26	6846	773
OICQ - IM software, popular in China	13.3	134	5.6	19914	2248	134	19914	2248
Domain Name System	2.2	22	0.5	1737	196	22	1737	196
Data	7.2	72	6.4	22848	2579	72	22848	2579
▼ Transmission Control Protocol	73.2	736	75.4	268100	30 k	512	155571	17 k
Transport Layer Security	21.9	220	55.1	195935	22 k	212	173692	19 k
▼ Hypertext Transfer Protocol	1.0	10	3.3	11712	1322	8	3560	401
Line-based text data	0.2	2	7.1	25331	2860	2	8152	920
Data	0.2	2	0.8	2920	329	2	2920	329
Address Resolution Protocol	1.5	15	0.1	420	47	15	420	47

我们可以看到最顶层显示出来的是 Frame，这是因为我们使用的网络接口是以太网，数据都是以数据帧的形式呈现，所以无论是

按分组还是按字节，Frame 都占 100%。同理，第二层 Ethernet 也都是占分组和字节的 100%。

到了第三层，流量出现了划分：按分组百分比，98.5%的流量属于 Ipv4 协议，而 1.5%的流量属于 ARP 协议。

在 Ipv4 协议中，UDP 协议占 25.3%，TCP 协议占 73.2%，两项相加等于 98.5%，即等于 Ipv4 协议的占比。在 UDP 协议中，SSDP 协议占了 2.6%，OICQ 占了 13.3%，DNS 协议占了 2.2%，Data 占了 7.2%，四项相加等于 25.3%，即等于 UDP 协议的占比。

但在 TCP 协议中，TLS 协议占 21.9%，HTTP 协议占 1.0%，Data 占 0.2%，三项相加等于 23.1%，小于 TCP 协议的占比。**通过查询资料，我找到了原因：因为 TCP 协议有分段传输，一个大的数据块使用 TCP 协议时会被分成很多段，协议分级只会识别这些分段当中的第一段，而后续的 TCP 分段是没有包含进来的，所以 TCP 协议以下的内容之和没有占到 TCP 协议的 100%。**

然后我们打开统计→会话，我们可以看到基于会话进行分类的统计信息：

Ethernet		5	IPv4	23	IPv6	TCP	39	UDP	16	
Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	
192.168.1.102	54136	120.92.150.127	7823	12	1271	7	572	5	6990	
192.168.1.102	54209	202.89.233.96	443	29	9946	14	1956	15	79901	
192.168.1.102	54210	202.89.233.96	443	25	8005	12	1133	13	68721	
192.168.1.102	54211	40.81.31.55	443	7	655	6	589	1	661	
192.168.1.102	54212	40.81.31.55	443	24	9801	13	1960	11	78411	
192.168.1.102	54213	23.99.125.55	443	19	7240	12	1129	7	61111	
192.168.1.102	54214	23.99.125.55	443	16	7066	10	1021	6	60451	
192.168.1.102	54215	125.56.201.96	443	18	4718	11	1111	7	36071	
192.168.1.102	54216	125.56.201.96	443	91	57 k	48	4387	43	52 k1	
192.168.1.102	54217	23.99.125.55	443	35	16 k	21	8012	14	85811	
192.168.1.102	54219	204.79.197.203	443	15	5568	9	734	6	48341	
192.168.1.102	54218	204.79.197.203	443	12	5300	8	692	4	46081	
192.168.1.102	54220	23.99.125.55	443	6	388	3	198	3	1901	
192.168.1.102	54221	23.99.125.55	443	47	32 k	31	22 k	16	98591	
192.168.1.102	54222	23.99.125.55	443	32	18 k	21	10 k	11	76061	
192.168.1.102	54223	204.79.197.200	443	8	2159	5	525	3	16341	
192.168.1.102	54224	204.79.197.200	443	12	5335	7	673	5	46621	
192.168.1.102	53993	120.92.150.127	7823	12	926	6	500	6	4261	
192.168.1.102	54187	120.92.150.127	7823	12	926	6	500	6	4261	
192.168.1.102	54226	220.181.76.83	80	14	7793	7	919	7	68741	
192.168.1.102	54225	220.181.76.83	80	8	1299	5	1011	3	2881	
192.168.1.102	54227	220.181.76.82	80	8	1331	5	996	3	3351	
192.168.1.102	54228	220.181.76.82	80	8	1333	5	998	3	3351	
192.168.1.102	54229	183.66.109.166	80	11	2722	6	772	5	19501	
192.168.1.102	54230	101.206.201.36	443	16	5548	9	1226	7	43221	
192.168.1.102	54231	183.66.109.166	443	20	6985	10	1312	10	56731	

可以看到，Ethernet 的会话有 5 个，Ipv4 的会话有 23 个，Ipv6 的会话有 1 个，TCP 的会话有 39 个，UDP 的会话有 16 个。

我们点击 TCP 的会话，发现每一个会话都是从一个 IP 地址指向另一个 IP 地址，除此之外，我们还可以看到两个地址之间传输的数据包的大小和比特数。

最后我们打开统计→分组长度，可以看到以数据包的长度分类的统计信息：

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
▼ Packet Lengths	1005	354.00	42	1514	0.0142	100%	0.6500	13.996
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	410	58.30	42	79	0.0058	40.80%	0.2900	13.933
80-159	239	109.31	81	153	0.0034	23.78%	0.1200	14.060
160-319	90	230.69	161	317	0.0013	8.96%	0.1200	13.932
320-639	63	463.21	328	601	0.0009	6.27%	0.1000	15.108
640-1279	73	819.66	651	1276	0.0010	7.26%	0.1200	15.731
1280-2559	130	1507.36	1355	1514	0.0018	12.94%	0.1700	14.272
2560-5119	0	-	-	-	0.0000	0.00%	-	-
5120 and greater	0	-	-	-	0.0000	0.00%	-	-

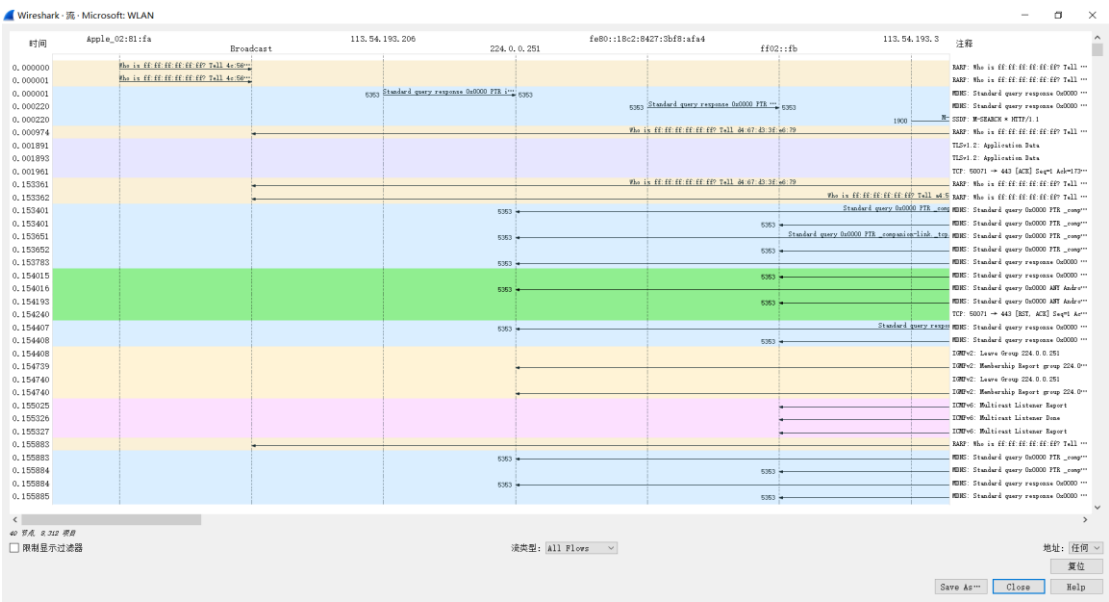
在抓取到的 1005 个数据包中，平均长度为 354.00，最长的数据包为 1514，最短的数据包为 42。其中，长度在 40-79 的数据包有 410 个，占总数的 40.80%；长度在 80-159 的数据包有 239 个，占总数的 23.78%；长度在 160-319 的数据包有 90 个，占总数的 8.96%；长度在 320-639 的数据包有 63 个，占总数的 6.27%；长度在 640-1279 的数据包有 73 个，占总数的 7.26%；长度在 1280-2559 的数据包有 130 个，占总数的 12.94%。

2、流量变化分析

为了探究流量随时间变化的过程，我们打开 wireshark 进行了时间为 180s 的抓包，期间打开了许多网站。

2.1、流量图查看 TCP 流

首先我们打开统计→流量图，查看 TCP 流。在 Flow 窗口中，可以看到数据包的抓取时间、数据包的源、目的地址、数据包的源和目的端口号。



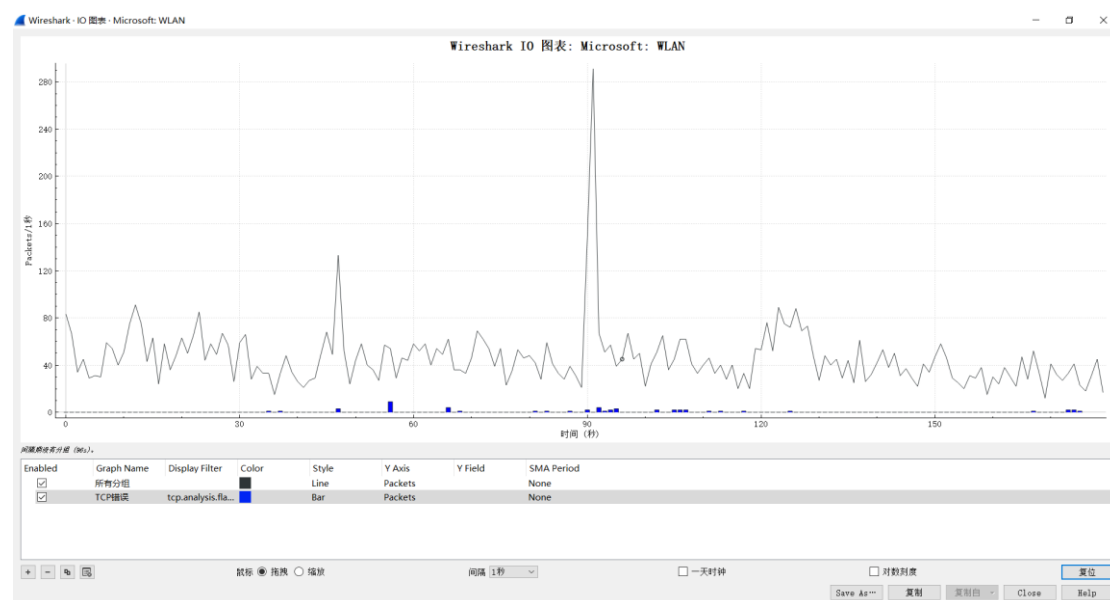
Flow 窗口内置有若干功能项（复选框）可供选择，以下是对这

些功能项的解释：

- Flow type：可在下拉菜单中选择所要查看的各种数据流。若点选 TCP Flow 菜单项，则 Wireshark 会根据抓包文件中的所有数据包或所有经过显示过滤器过滤的数据包，来生成含 TCP 标记、序列号、ACK 号以及报文段长度的 TCP 数据流图。在针对抓包文件应用显示过滤器 http.request，且同时勾选 Limit to display 复选框并将 Flow type 设置为 TCP Flows 的情况下，Flow 窗口将只会显示 PSH 位置 1 的 TCP 流。
- Address：只有 IP 地址一种选项。

2.2、I/O Graphs 初级应用

然后我们打开统计→IO 图表，我们可以看到流量（每秒传输的数据包数）随时间变化的折线图：

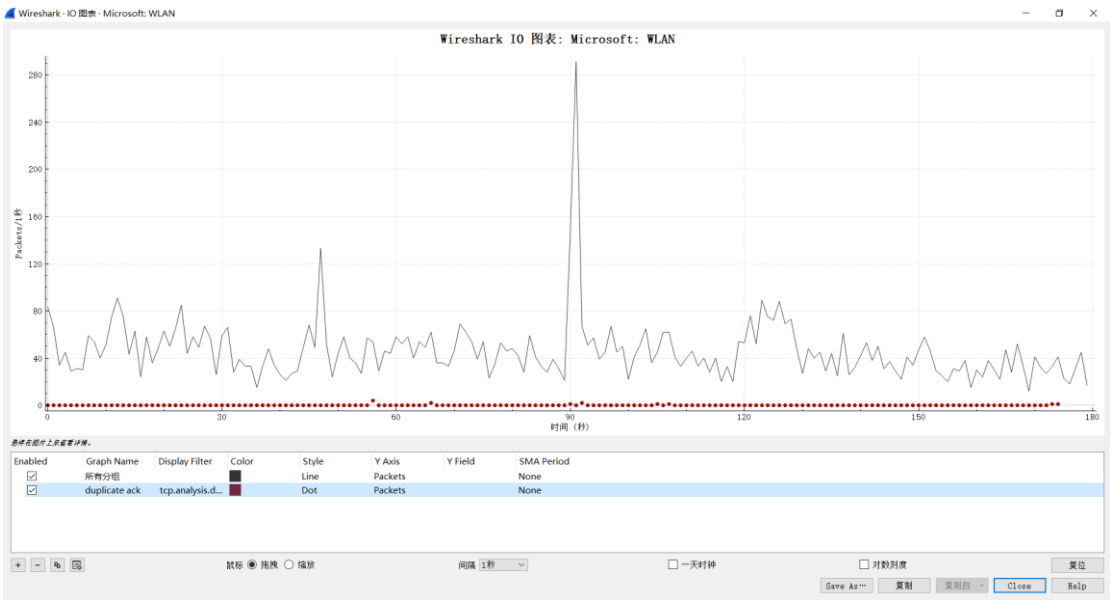


I/O Graphs 的作用是配置该工具来排除网络故障。在 I/O Graphs 窗口中，上半部分为图形显示区域，下半部分是过滤器配置区域，可以在此配置显示过滤器，并根据显示过滤器来展示相关图

形。在默认情况下，折线图的 X 轴表示的是时间（单位为秒），Y 轴表示的是流量速率（单位为数据包/秒）。

折线图反映的是所有类型的数据包每秒传输的个数，而下面的柱形图反映的是在数据传输过程中错误的 TCP 包的个数。我们可以看出，流量在第 91 秒达到了最高峰：291 个数据包/秒。

修改过滤器：名为 duplicate ack 的图形体现了对抓包文件施加过滤器 tcp.analysis.duplicate_ack 时的流量状况，该图的显示风格为点状（Dot）。该图反映了 TCP 重复确认事件的次数。



2.3、I/O Graphs 高级应用

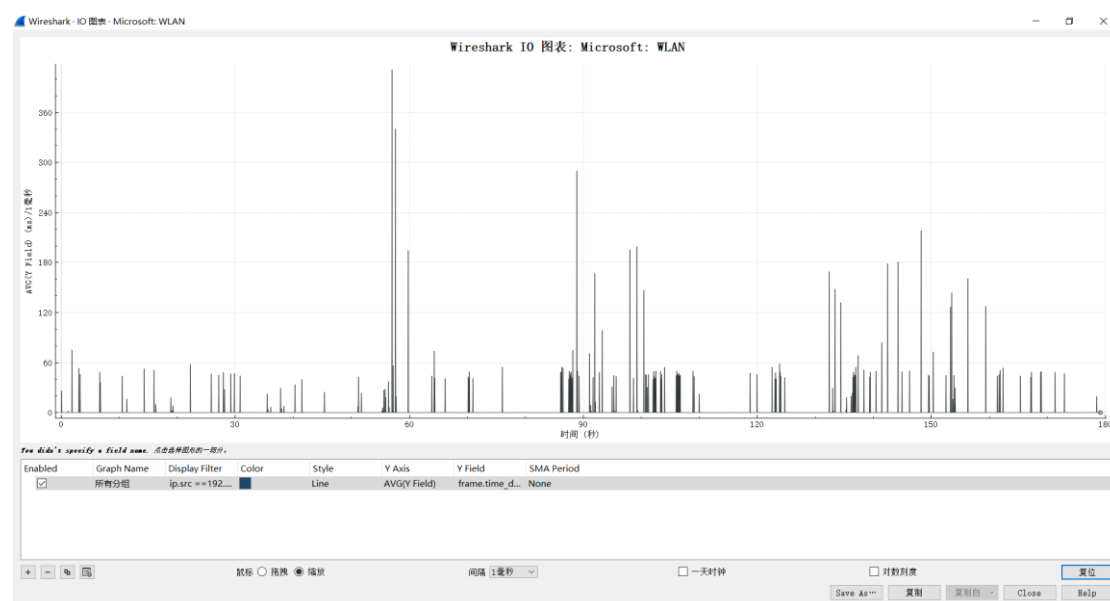
排除网络故障时，通过观察 Wireshark 抓包文件中相关数据帧（封装 TCP 报文段的数据帧）的抓取时间间隔，通常有助于判断出是否存在与 TCP 的性能和语音/视频等交互式应用的性能有关的问题。让 I/O Graphs 工具生成（封装 TCP/UDP 报文段的）数据帧抓取间隔时间统计图形，无疑是一种比较直观的了解 TCP/UDP 性能的方法。在使用 I/O Graphs 工具时，要配搭显示过滤参数

frame.time_delta 和 frame.time_delta_displayed。

我们在 I/O Graphs 窗口中配置了以下参数：

- 设置显示过滤器为 ip.src == 192.168.1.102，其作用是从抓包文件中筛选出源 IP 地址为 192.168.1.102 的 IP 数据包。
- 在 Y Axis 一栏的下拉菜单中选择 AVG (Y Axis)菜单项，它表示显示平均帧间间隔时间。
- 在 Y Field 输入栏内，填入了显示过滤参数 frame.time_delta，该参数是指当前帧与上一帧的时间间隔。
- 在 Interval 下拉菜单中，选择 1ms。

配置好参数之后，I/O Graphs 窗口如下所示：



该窗口显示了在单位时间（1ms）内抓到的源 IP 地址为 192.168.1.102 的 IP 数据包（帧）的平均时间间隔图。

重新设置显示过滤器：

1) 在 Display filter 输入栏里填入了显示过滤语句

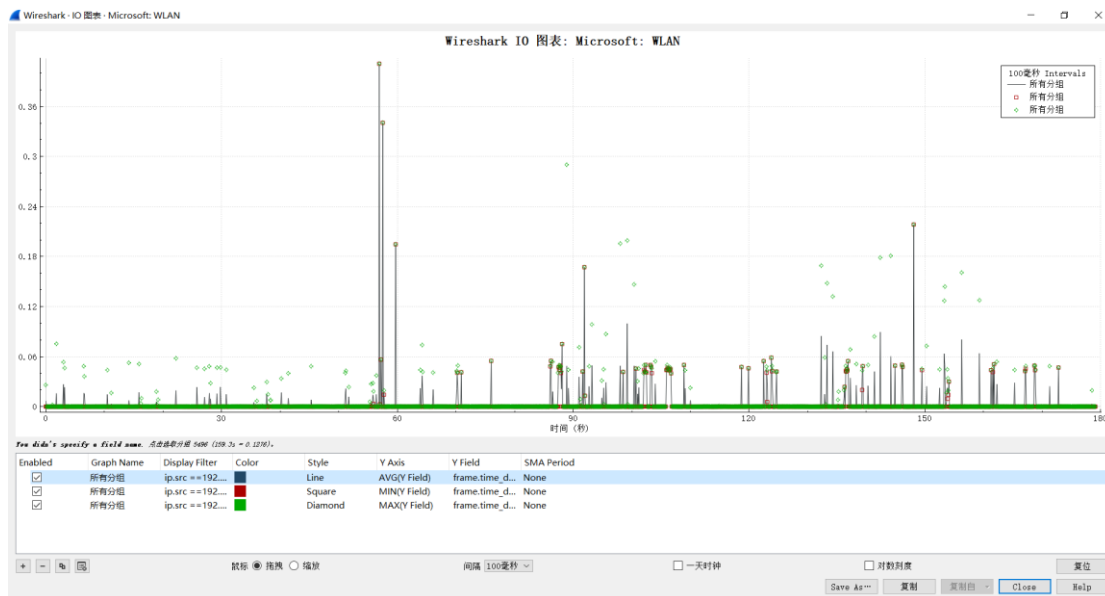
ip.src==192.168.1.102，目的是先在抓包文件中筛选出源 IP 地址为

192.168.1.102 的所有 IP 数据包。在 Y Axis 一栏的下拉菜单里选择了 AVG(Y Field)菜单项。在其右边的 Y Field 输入栏内输入了显示过滤参数 frame.time_delta，目的是让 Wireshark 生成在单位时间（100ms）内抓到的源 IP 地址为 192.168.1.102 的 IP 数据包（帧）的平均时间间隔图。样式设置为 Line。

2) 在 Display filter 输入栏里填入了显示过滤语句 ip.src==192.168.1.102，目的是先在抓包文件中筛选出源 IP 地址为 192.168.1.102 的所有 IP 数据包。在 Y Axis 一栏的下拉菜单里选择了 MIN(Y Field)菜单项。在其右边的 Y Field 输入栏内输入了显示过滤参数 frame.time_delta，目的是让 Wireshark 生成在单位时间（100ms）内抓到的源 IP 地址为 192.168.1.102 的 IP 数据包（帧）的最短时间间隔图。样式设置为 Square。

3) 在 Display filter 输入栏里填入了显示过滤语句 ip.src==192.168.1.102，目的是先在抓包文件中筛选出源 IP 地址为 192.168.1.102 的所有 IP 数据包。在 Y Axis 一栏的下拉菜单里选择了 MAX(Y Field)菜单项。在其右边的 Y Field 输入栏内输入了显示过滤参数 frame.time_delta，目的是让 Wireshark 生成在单位时间（100ms）内抓到的源 IP 地址为 192.168.1.102 的 IP 数据包（帧）的最长时间间隔图。样式设置为 Diamond。

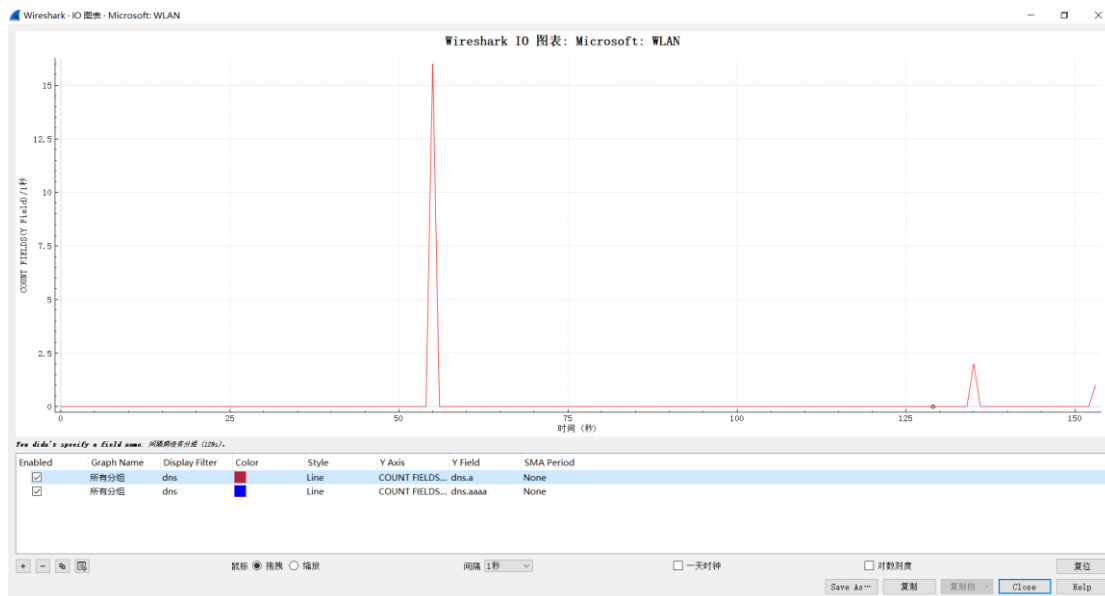
由此，I/O Graphs 窗口将显示三个图像，如下所示：



2.4、统计 Y Field 输入栏所指定的数据包属性

Y Axis 下拉菜单的 COUNT FIELDS (Y Field) 菜单项的功能是统计 Y Field 输入栏所指定的数据包的属性（或协议的特征）在抓包文件（包含数据包）中出现的次数。在进行统计之前，可在 Display filter 输入栏内输入显示过滤器，对抓包文件做第一步的筛选。

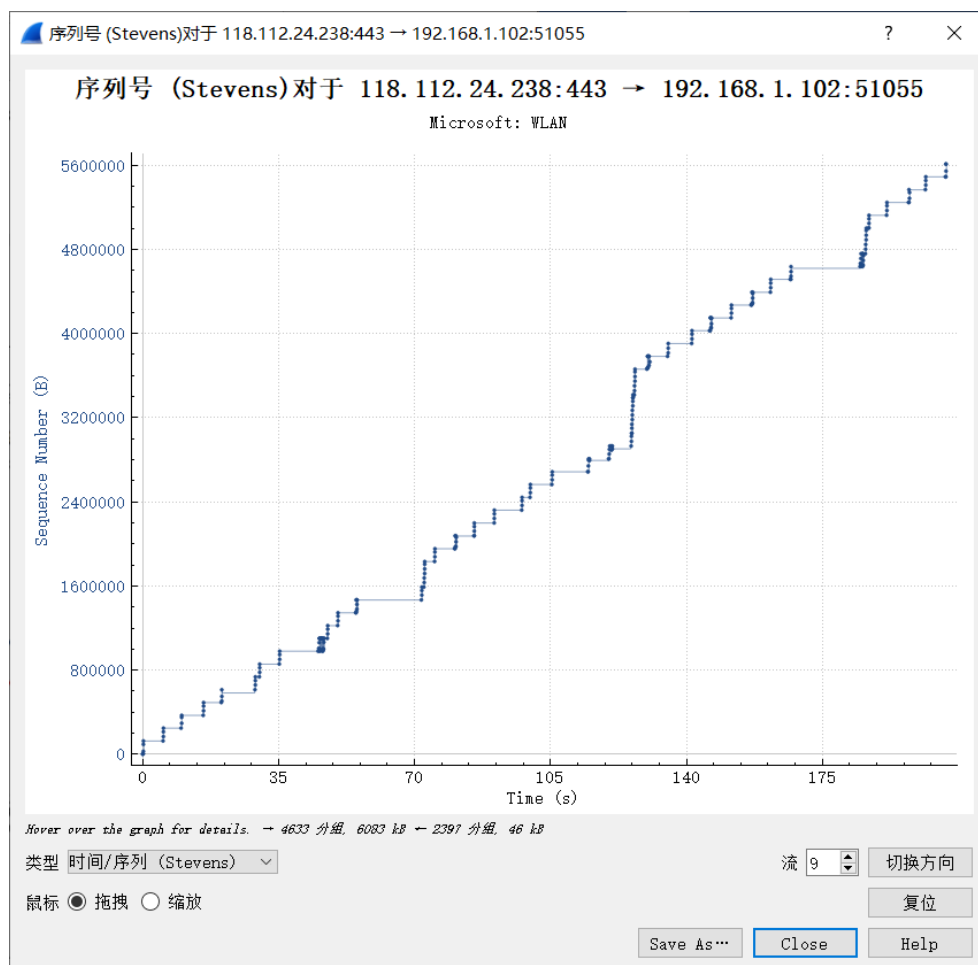
- 1) 在图形的 Display filter 输入栏内输入 dns，在图形的 Y Axis 下拉菜单中，选择 COUNT FIELDS (Y Field) 菜单项。样式选择 Line。在图形的 Y 字段输入栏内输入 dns.a。
- 2) 在图形的 Display filter 输入栏内输入 dns，在图形的 Y Axis 下拉菜单中，选择 COUNT FIELDS (Y Field) 菜单项。样式选择 Line。在图形的 Y 字段输入栏内输入 dns.aaaa。



2.5、用 TCP 流图形中的时间序列 (Stevens) 进行分析

选择统计菜单下的 TCP 流图形，点击其下的时间序列

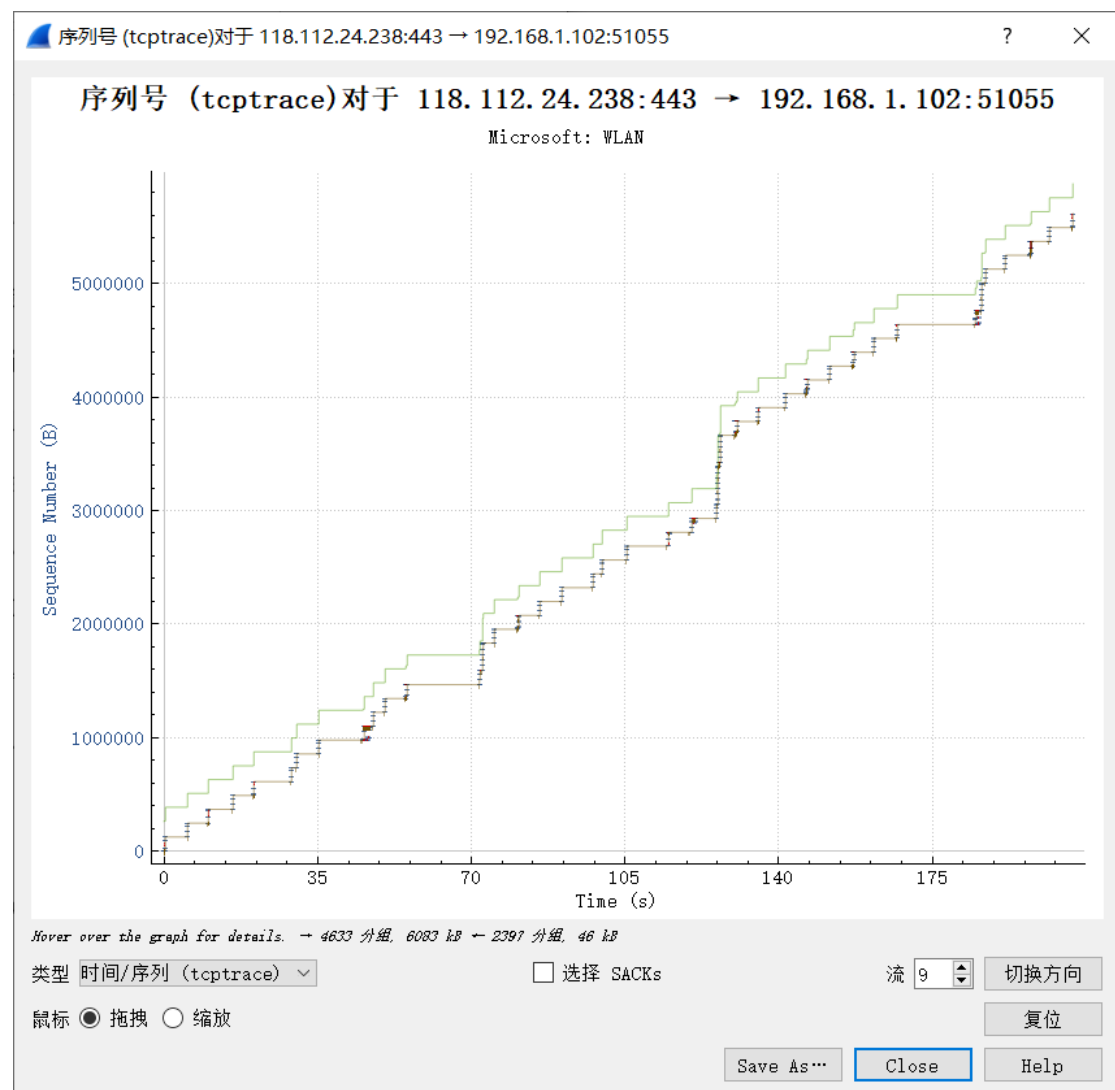
(Stevens)，窗口如下所示：



该图形反映了随着时间的推移，受监控的 TCP 对话在某个方向所传数据的字节数。出现在图中的是一条逐渐上升的折线，中间有许多断裂。Y 轴表示 (TCP) 序列号，作用是统计通过指定 TCP 对话传递的数据的字节数。将鼠标放在图形的最后一个点上，我们得知：该 TCP 连接 (对话) 在 207.9 秒内传输了 5575296 字节，传输速率约为 26817 字节/秒，或 26KB/s。

2.6、用 TCP 流图形中的时间序列 (tcptrace) 进行分析

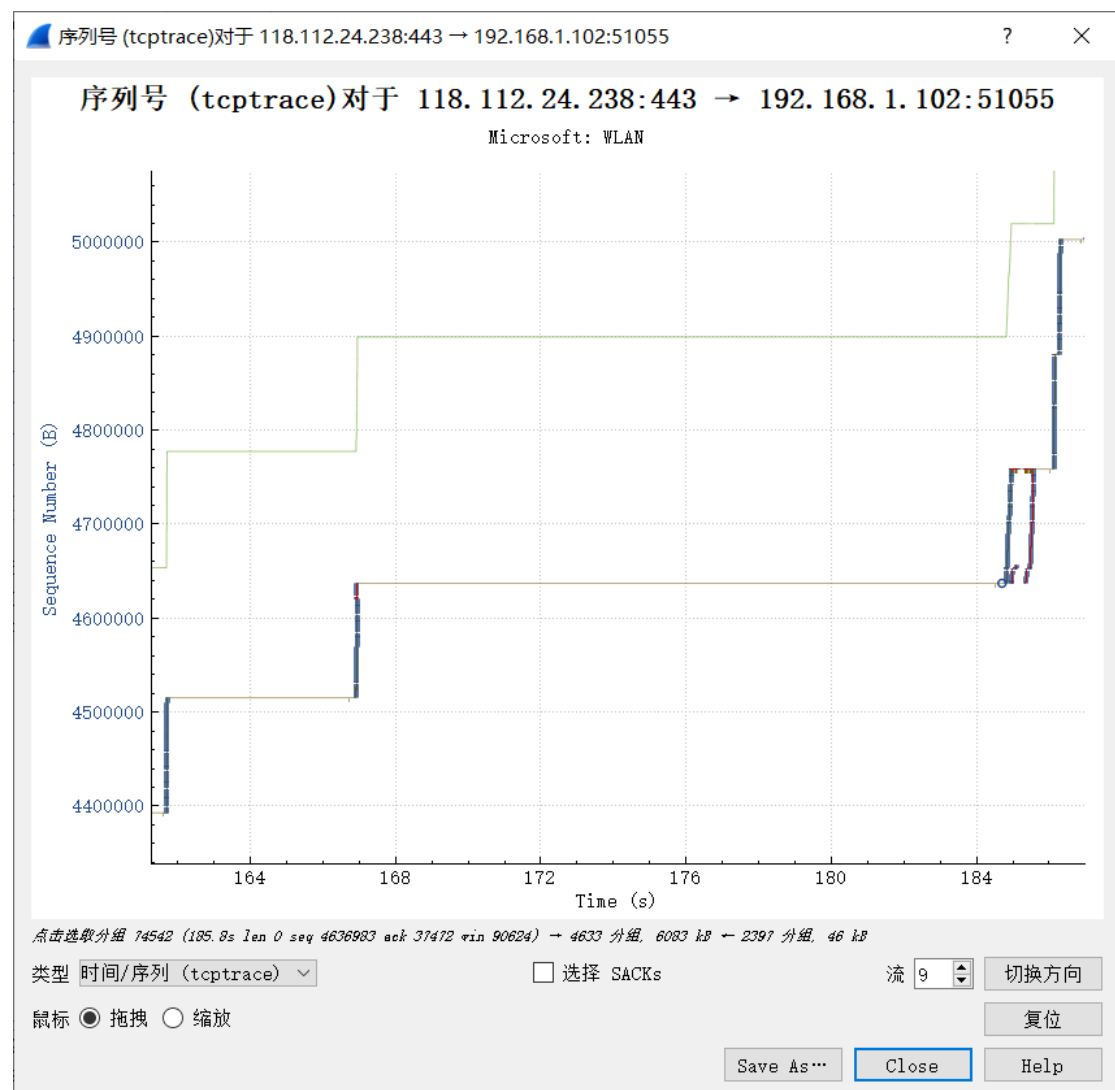
选择统计菜单下的 TCP 流图形，点击其下的时间序列 (tcptrace)，窗口如下所示：



该图形反映了抓包时段内受监控的 TCP 对话在某个方向上传输数据的进展情况，提供了有待监控的 TCP 连接的诸多详细信息。我们可以用这些信息来分析与此 TCP 连接有关的种种问题，包括 TCP 确认、TCP 重传，以及 TCP 窗口大小等信息。

我们选取一块区域并放大，可以看到三种类型的线：

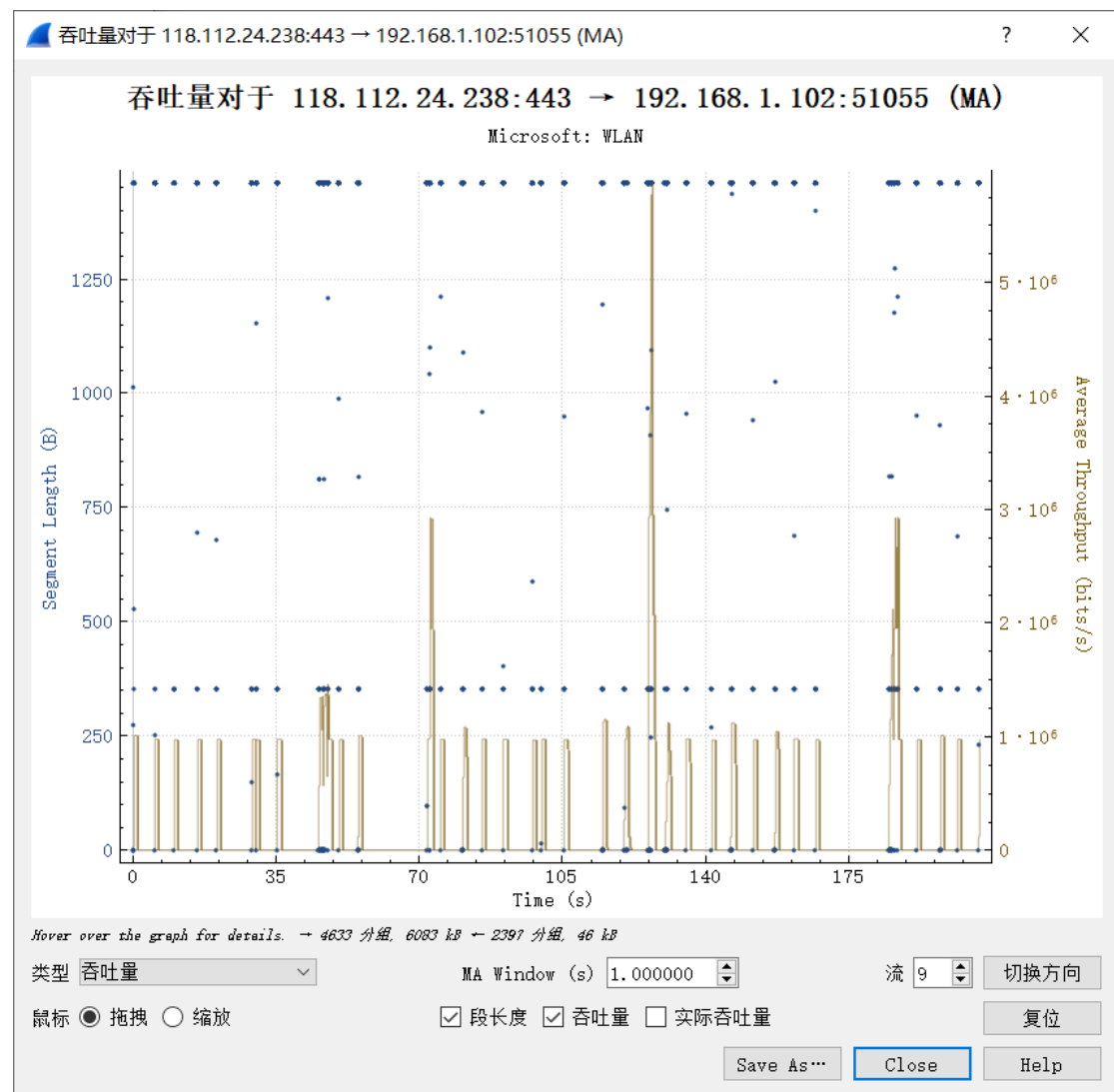
- 垂直蓝线：表示通过受监控的 TCP 对话（连接）发送的 TCP 报文段。
- 水平棕线：表示逆向的 TCP 确认报文段。
- 绿线：表示数据包发送过程中接收端 TCP 窗口大小。



棕线和绿线之间的空间表示 TCP 接收端剩余的 TCP 缓冲区的大小。TCP 接收端的 TCP 缓冲区用来限制 TCP 发送端向 TCP 接收端发送的数据量。当棕线和绿线彼此靠拢直至重叠时，就表示接收端窗口渐满，发送端需降速或停止发送数据。

2.7、用 TCP 流图形中的吞吐量进行分析

选择统计菜单下的 TCP 流图形，点击其下的吞吐量，窗口如下所示：



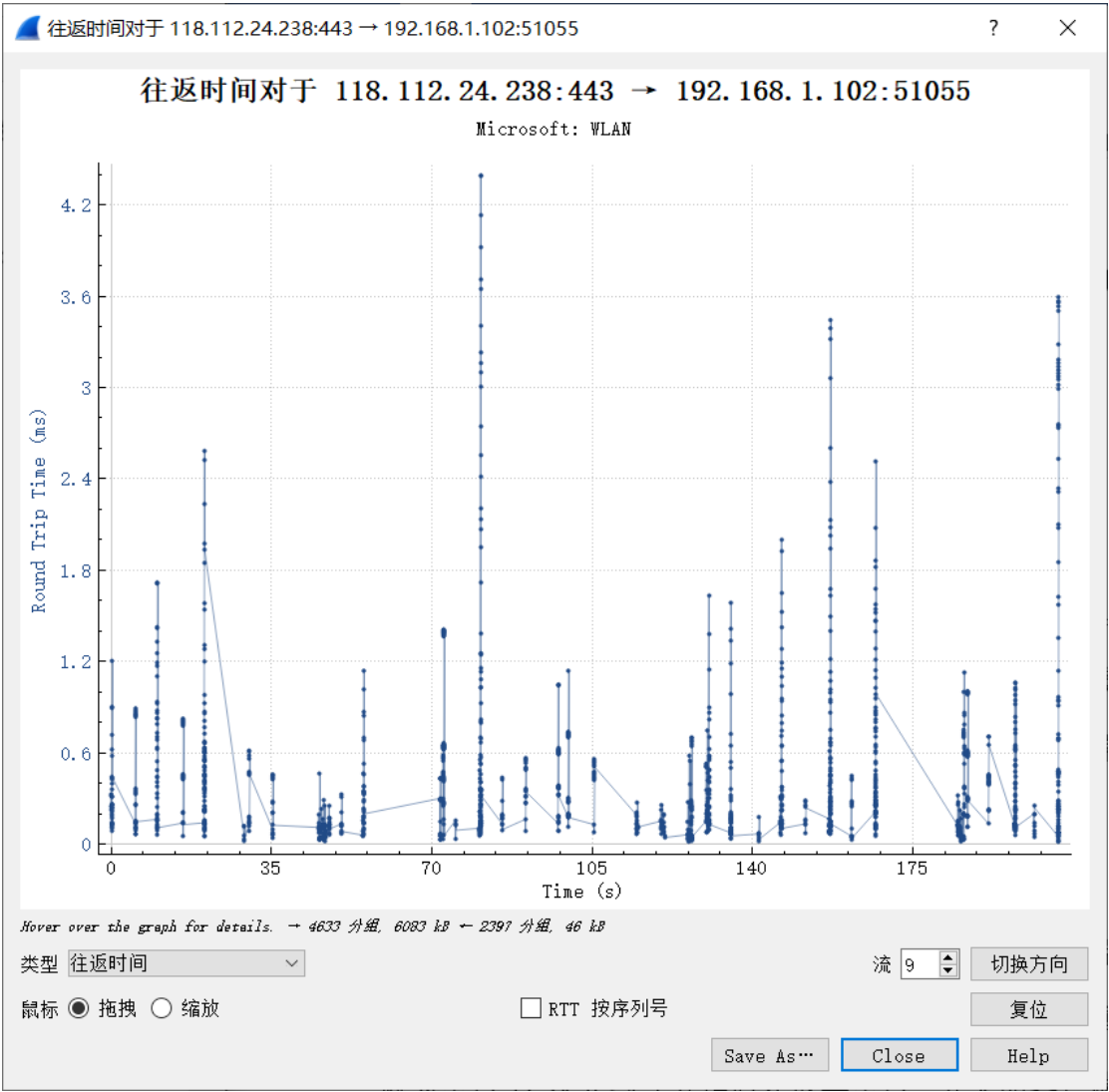
吞吐量工具所生成的图形所能呈现的网络状况或许不如时间序列 (Stevens) 和时间序列 (tcptrace) 工具所反映的全面，但可以即

时反映出应用程序吞吐量的变化，从而预示网络问题。

该吞吐量窗口呈现的是抓包文件中编号为 9 的 TCP 流的吞吐量图。可以看到，这条编号为 9 的 TCP 连接的吞吐量约为 100~200 KB/s。

2.8、用 TCP 流图形中的往返时间进行分析

选择统计菜单下的 TCP 流图形，点击其下的往返时间，窗口如下所示：

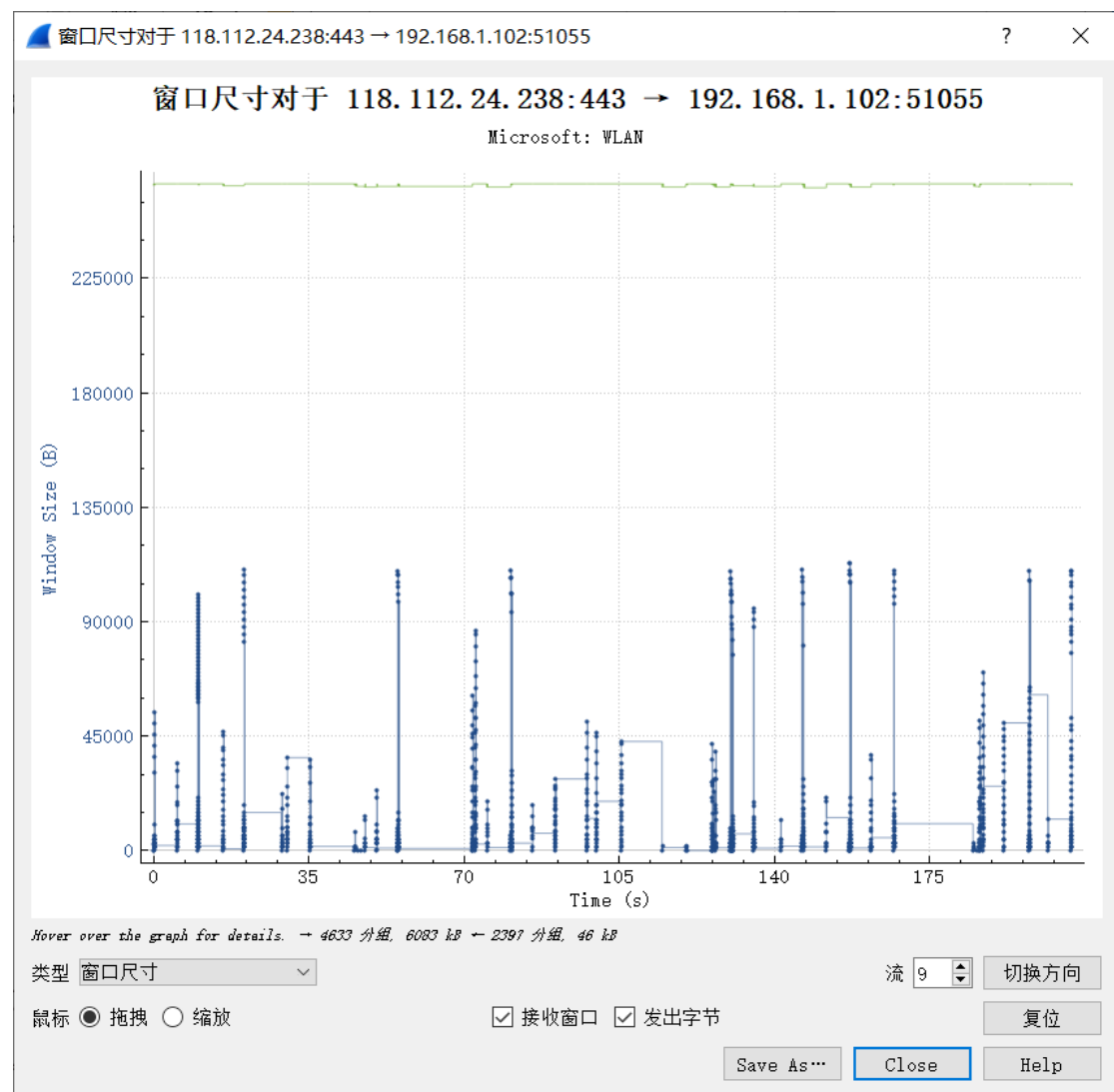


该往返时间窗口呈现的是抓包文件中编号为 9 的 TCP 流的往返时间图。可以看到，这条编号为 9 的 TCP 连接的大多数字节（序列

号) 在很短的时间内都得到了确认, 只是有些不太稳定, 这会影响 TCP 的性能。

2.9、用 TCP 流图形中的窗口尺寸进行分析

选择统计菜单下的 TCP 流图形, 点击其下的窗口尺寸, 窗口如下所示:



该窗口反映了由接收方或发送方所导致的数据传输性能下降问题。原因可能是服务器或客户端主机反应较慢, 不能迅速处理收到的所有数据。于是, 接收方便以降低接收窗口的形式, 告知发送方: 自己的接收能力有限, 请不要发得太快。

七、 TCP 通信中异常情况整理

在流量变化分析中，TCP 错误的柱形图引起了我的注意。为了更加详细地了解错误的类型。我查阅资料，总结出客户端与服务器端在 TCP 链接已经建立且正常通信的过程中，发生以下几种异常情况时 TCP 链接会产生的各种现象。

1. 服务器主机崩溃

客户端在给服务器发送数据时，由于收不到服务器端回传的 ACK 确认报文，正常情况下，客户端 TCP 均会进行超时重传，一般为重传 12 次大约 9 分钟后才放弃重传，并关闭客户端 TCP 链接。

2. 服务器主机崩溃后重启

如果服务器主机在崩溃重启的这段时间里，客户端没有向服务器发送数据，即客户端没有因重传次数超过限制关闭 TCP 链接。则在服务器重启后，当客户端再向服务器发送 TCP 报文时，由于服务器中的 TCP 链接已经关闭，会直接向客户端回复 RST 报文，客户端在接收 RST 报文后关闭自己的 TCP 链接。

3. 服务器主机断网或者中间路由器出现故障

与情况 1 类似，客户端会进行超时重传，直到重传次数超过后放弃重传，并关闭客户端 TCP 链接。(因为 TCP 中会忽略目的主机不可达和目的网络不可达的 ICMP 报文，并进行重传，直到重传总时间超过限制)

4. 服务器主机断网或者中间路由器出现故障后又恢复

如果在服务器主机断网或者中间路由器出现故障这段时间内, 客户端和服务端之间没有进行相互通信, 即双方均没有察觉对方目的不可达, 则在恢复网络链接后两端的 TCP 链接均有效, 能够正常继续进行通信。

如果在服务器主机断网或者中间路由器出现故障这段时间内, 客户端因向服务器发送数据超时, 并重传总时间超过限制关闭 TCP 链接。则在网络恢复后, 服务器再向客户端发送 TCP 报文时, 客户端也会直接恢复 RST 报文, 服务器再收到 RST 报文后关闭自己的 TCP 链接。

5. 服务器关机或服务器进程被终止

正常情况下服务器主机被关机时, 操作系统都会事先通知所有仍在运行的进程, 并先将所有进程终止后, 再继续关闭电脑。而所有的进程在被终止时, Unix 操作系统内核都会事先去关闭所有已经打开的 TCP 链接, 即向客户端发送 FIN 标志报文, 进行四次握手关闭连接。

因此, 对于这种情况, 客户端是能够察觉到并正常关闭 TCP 链接。

6. 服务器的端口被关闭

如果在通信过程中, 服务器的监听端口被管理员或系统禁掉, 则当客户端再向服务器发送 TCP 报文时, 服务器在收到该报文后, 由于发送该目的端口没有处于监听状态, 则会直接向客户端发送 RST 报文, 客户端在收到 RST 报文后会直接关闭自己 TCP 链接。

7. TCP 的保活机制

TCP 中的保活机制是一个可选项，并不是必须的。主要用在服务器端，用于检测已建立 TCP 链接的客户端的状态，防止因客户端崩溃或者客户端网络不可达，而服务器端一直保持该 TCP 链接，占用服务器端的大量资源。

八、对 QICQ 包的分析

在使用 Wireshark 的过程中无意发现了 QQ 独有的 QICQ 包，为了分析这种包，我做了如下实验：

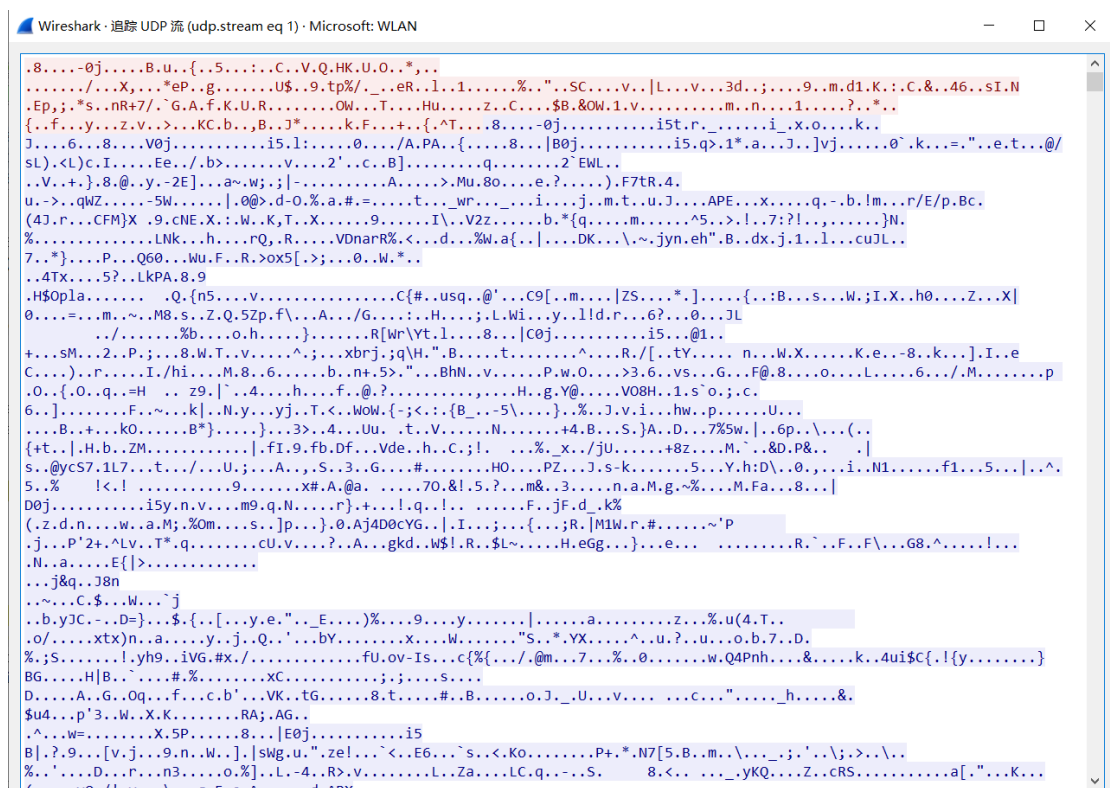
打开 Wireshark 进行抓包，打开 QQ 接收信息，再关闭 QQ，停止抓包。随机点击一个 QICQ 包，具体数据如下所示：

```
> Frame 223: 89 bytes on wire (712 bits), 89 bytes captured (712 bits) on interface 0
▼ Ethernet II, Src: HonHaiPr_8d:47:d3 (80:2b:f9:8d:47:d3), Dst: Tp-LinkT_2e:36:cd (d0:76:e7:2e:36:cd)
  > Destination: Tp-LinkT_2e:36:cd (d0:76:e7:2e:36:cd)
  > Source: HonHaiPr_8d:47:d3 (80:2b:f9:8d:47:d3)
  Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 192.168.1.102, Dst: 59.36.128.109
▼ User Datagram Protocol, Src Port: 4027, Dst Port: 8000
  Source Port: 4027
  Destination Port: 8000
  Length: 55
  Checksum: 0x0b71 [unverified]
  [Checksum Status: Unverified]
  [Stream index: 1]
  > [Timestamps]
▼ OICQ - IM software, popular in China
  Flag: Oicq packet (0x02)
  Version: 0x3803
  Command: Heart Message (2)
  Sequence: 1114
  Data(OICQ Number,if sender is client): 812288728
▼ Data: \002
  ▼ [Expert Info (Warning/Undecoded): Trailing stray characters]
    [Trailing stray characters]
    [Severity level: Warning]
    [Group: Undecoded]
```

在 Ethernet II 一层，可以看出路由器是 Tp-Link 和 HonHai。

在 User Datagram Protocol 一层，Destination Port:8000，这指的是 QQ 使用的端口号，长度为 55 字节，检验和显示为禁用。

在 QICQ 一层，可以看到自己的 QQ 号码为：812288728。右键点击这一层，选择追踪流→UDP 流，内容如下所示：



The image shows a Wireshark packet capture window titled "Wireshark · 追踪 UDP 流 (udp.stream eq 1) · Microsoft: WLAN". The packet list on the left shows a single packet of size 1440 bytes. The packet details pane shows the structure of the UDP stream, including the Ethernet II header, Internet Protocol Version 4 header, and User Datagram Protocol header. The main pane displays the raw data of the UDP stream, which is a sequence of hexadecimal bytes representing encrypted data. The data is shown in a hex dump format, with the first few lines of the hex dump visible.

可以看出，这些信息是加密的，需要知道加密算法才能破解这些信息。

九、 总结及心得体会

在本次项目的过程中，我掌握了利用 Wireshark 软件进行网络认知和网络分析的基本方法，包括抓包、层次化封装分析、交互时序分析等，并且学习了许多协议及其功能。与此同时，我还用一定数量的数据包进行流量成分及变化的分析，并从中发现了感兴趣的问题，并且通过查找资料的方式进行分析。