

实验 2 真实网络环境协议分析

实验目的

了解和熟悉常见网络协议的内部通信过程

实验内容

安装网络包捕获软件，观察网络中的数据包

实验环境

PC 机、Ethereal 软件（wireshark），Windows8.1 操作系统

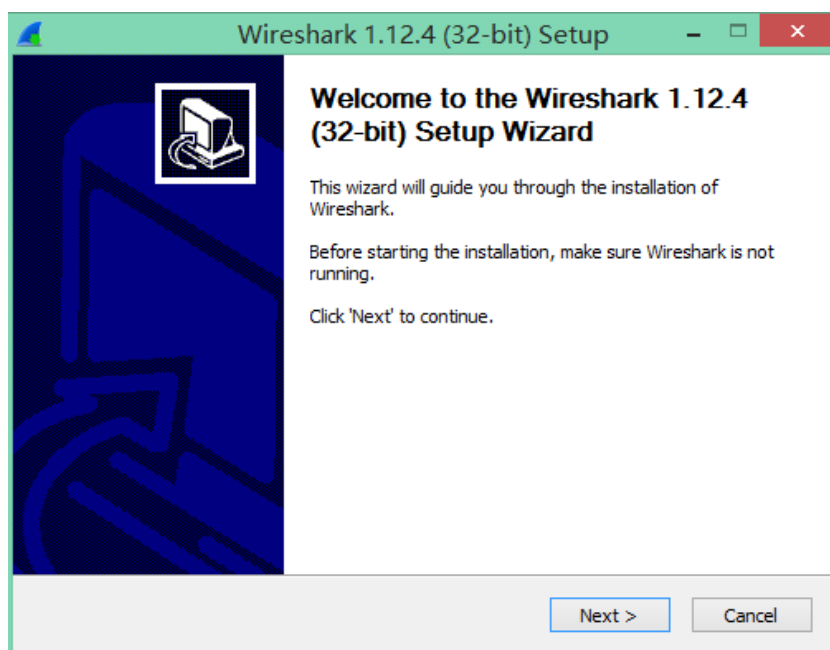
实验时间

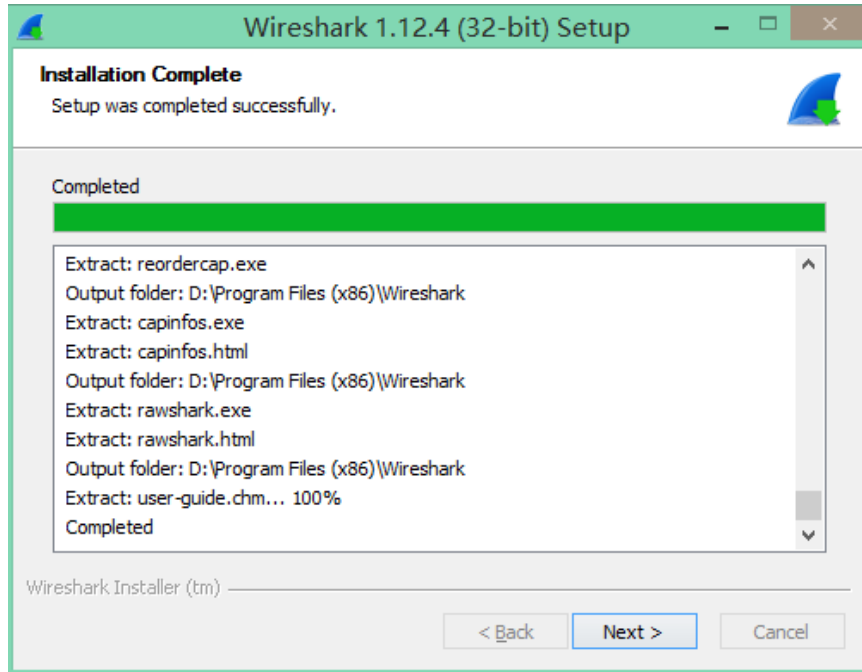
2 机时。

实验步骤：

- 安装网络包捕获软件 **Ethereal**

下载 wireshark 软件并安装。



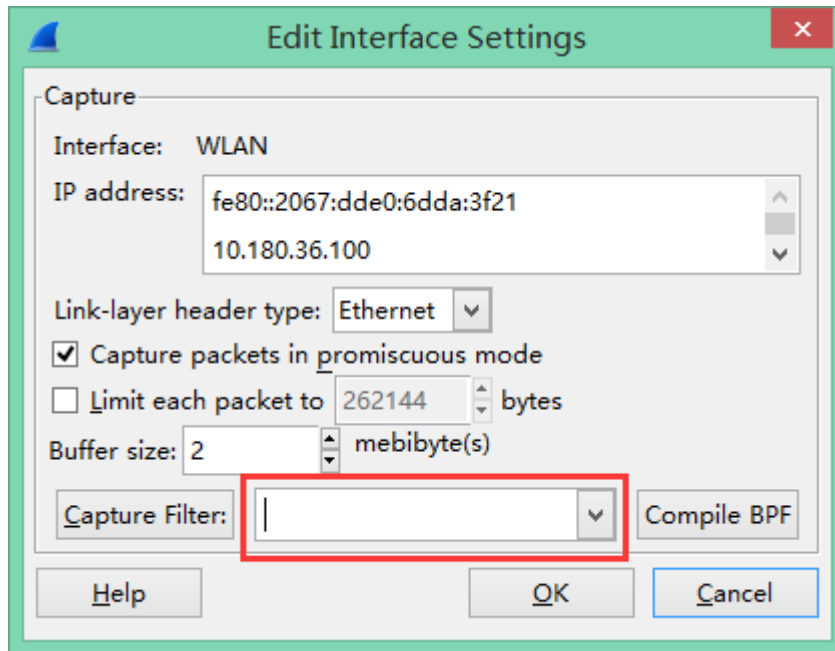


- 配置网络包捕获软件，捕获所有机器的数据包

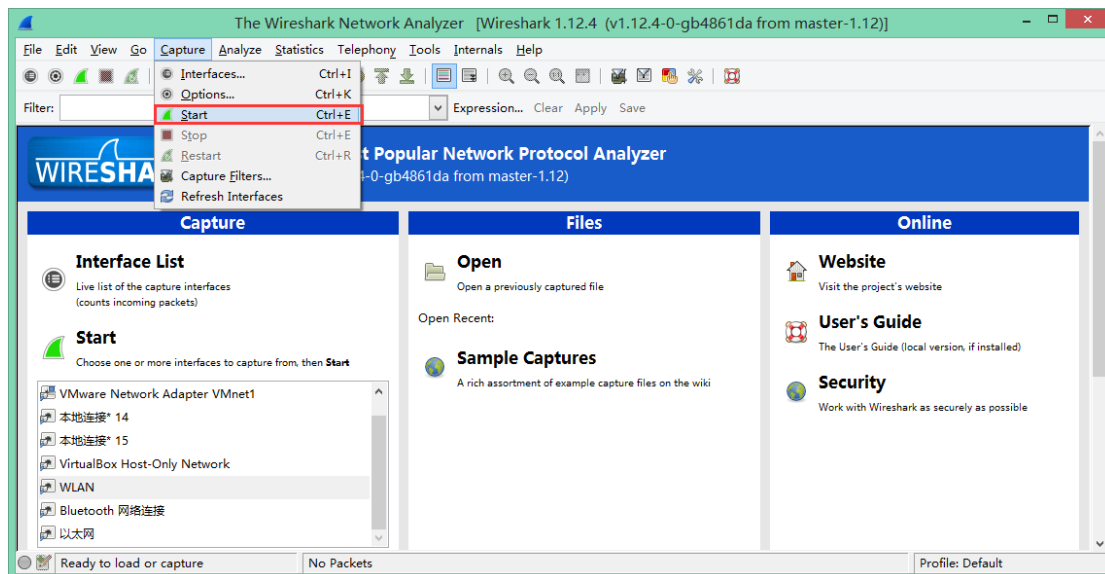
首先，需要在界面中选择当前的网络接口，由于本次实验是在无线网络的环境下完成的，因此这里我们选择 WLAN。



接下来会弹出一个窗口，在这个窗口中，不对抓包过滤器添加任何内容，则默认抓取网络中的所有数据包。

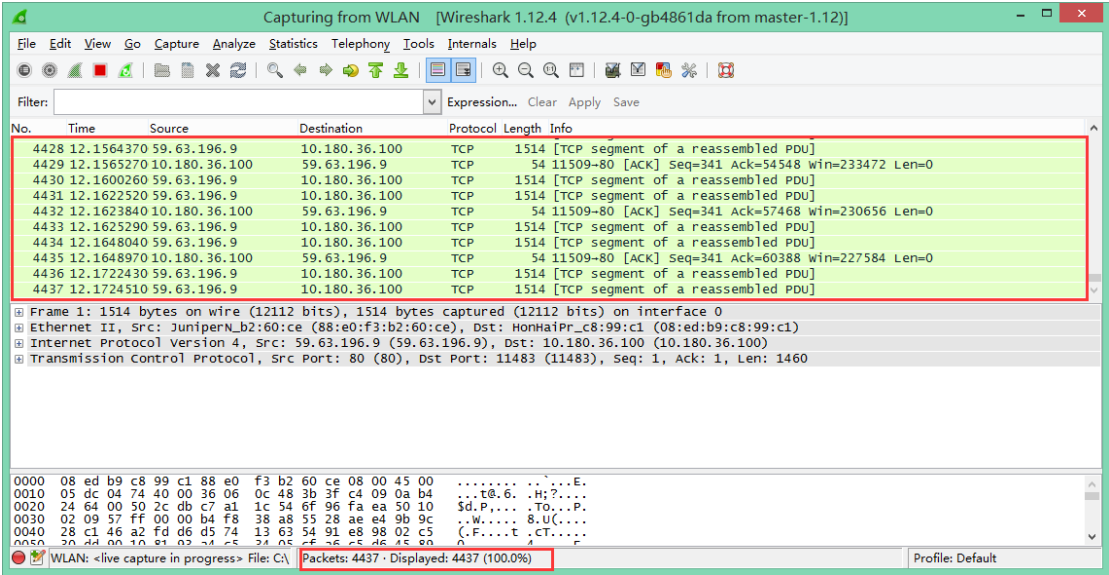


接下来在 capture 一栏中，选择 Start 开始抓包。



可以在下图中的红色方框中看到所有捕获的包，主要信息包括源地址，目的地址，协议名，长度以及包的主要信息内容。

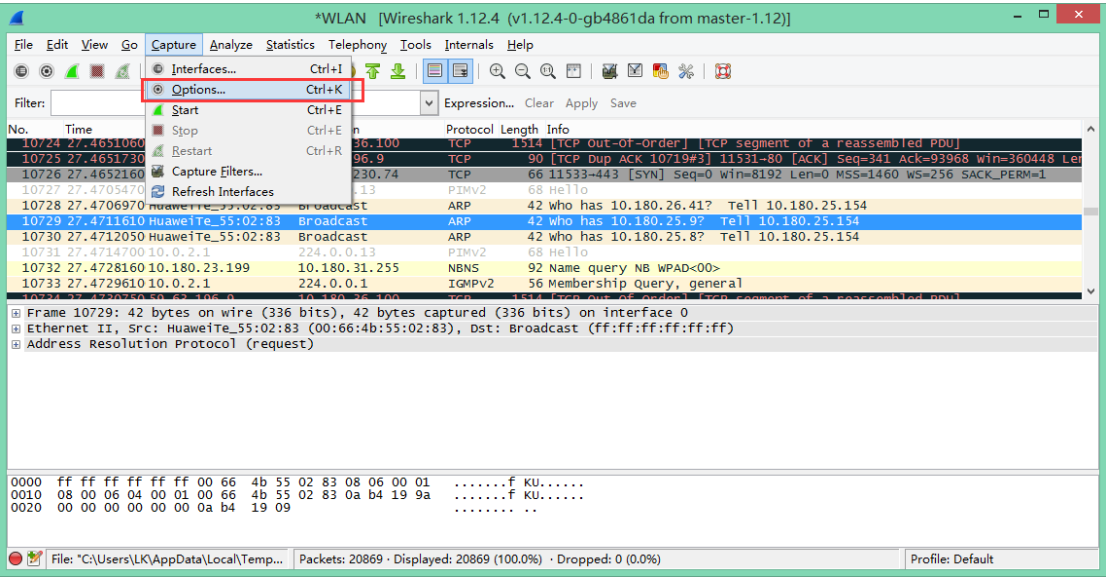
点击其中一个包，可以在下方看到该包的详细内容。其中紧挨着包列表的方框中，可以看到包的大小，捕获大小，包的源地址，目的地址，协议信息，源端口，目的端口等内容。以及包的大致内容等信息。



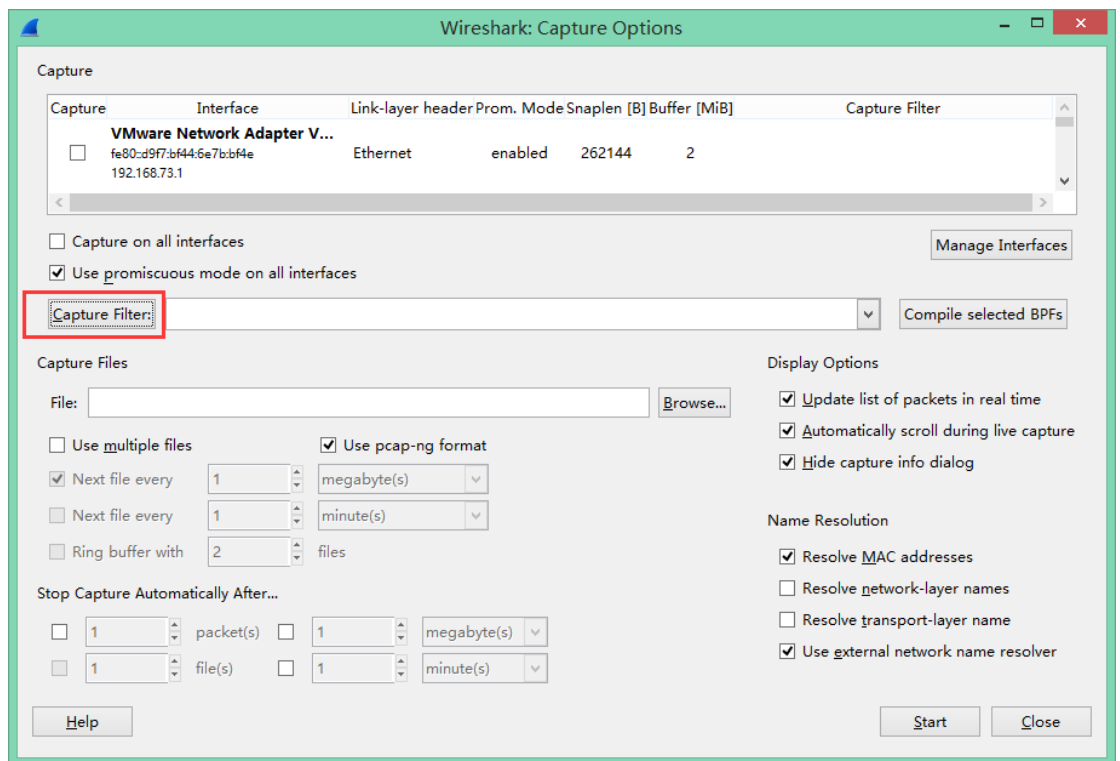
在最下方红框圈出的部分，显示了总共抓取到的包的数量以及显示的包的数量，其中抓取的包可以通过抓取过滤器进行过滤，而显示包数量可以通过显示过滤器进行过滤。

- 配置网络包捕获软件，只捕获特定 IP 或特定类型的包

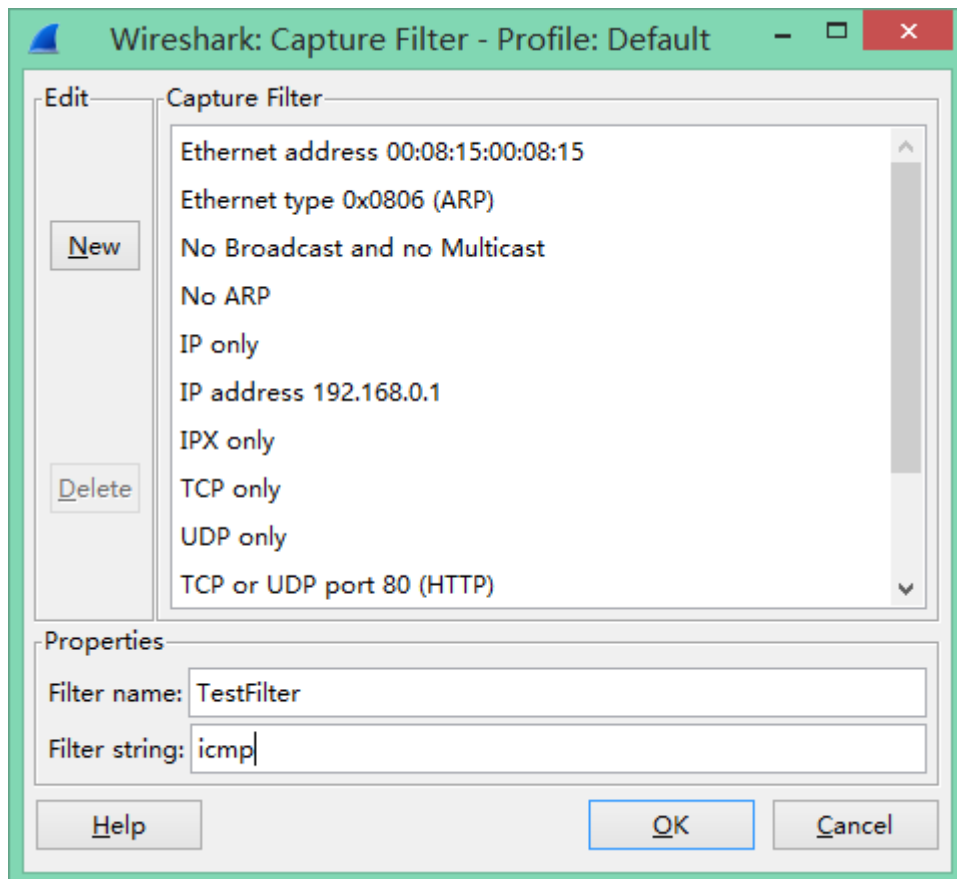
在 capture 捕获一栏中的 option 选项中



选择 capture filter 过滤器添加一个过滤器



输入过滤器的名字可以方便在下次直接使用, 在 Filter string 中输入 icmp 表明该过滤器只捕获 icmp 协议的数据包. 当然, 还可以通过 and, or 等符号添加更多的条件, 扩大或缩小捕获数据包的范围, 使捕获对象更加具体



接下来进行测试，我们使用基于 ICMP 协议的 ping 命令来进行测试。
这个命令一共发送了 4 个数据包包到 baidu.com 的服务器

```
C:\Users\LK>ping baidu.com

正在 Ping baidu.com [220.181.57.217] 具有 32 字节的数据:
来自 220.181.57.217 的回复: 字节=32 时间=44ms TTL=52
来自 220.181.57.217 的回复: 字节=32 时间=55ms TTL=52
来自 220.181.57.217 的回复: 字节=32 时间=111ms TTL=52
来自 220.181.57.217 的回复: 字节=32 时间=36ms TTL=52

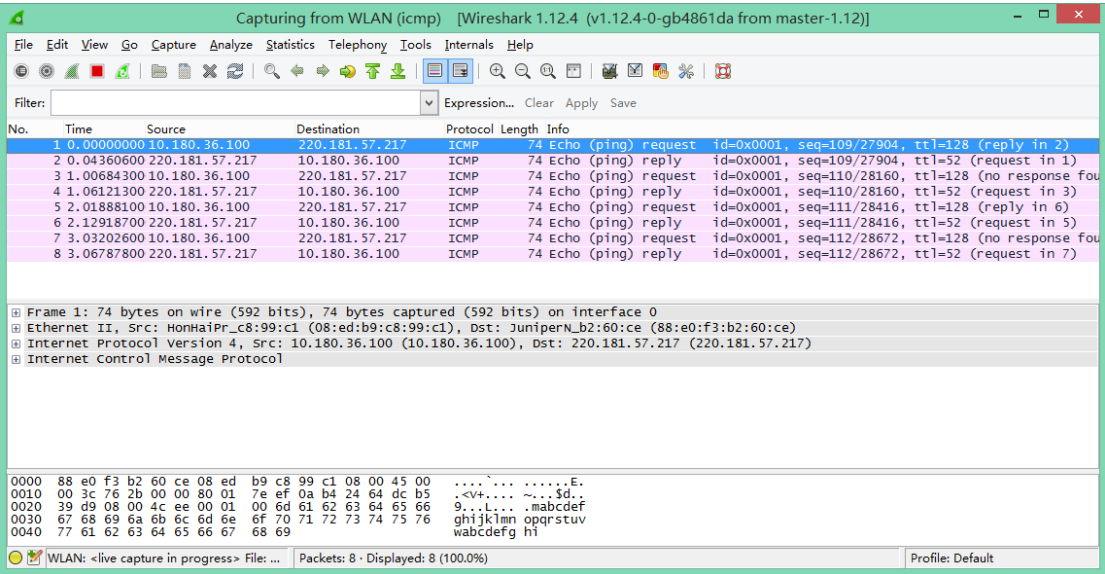
220.181.57.217 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
    往返行程的估计时间<以毫秒为单位>:
        最短 = 36ms, 最长 = 111ms, 平均 = 61ms
```

我们使用 ipconfig 命令查看本机的 IP 地址，为 10.180.36.100

```
无线局域网适配器 WLAN:

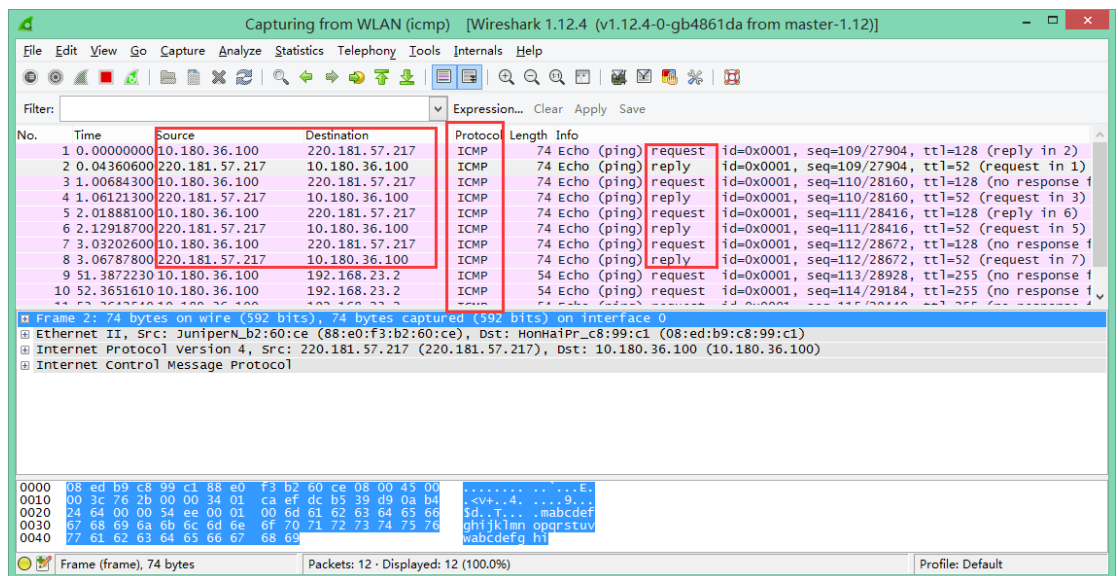
连接特定的 DNS 后缀 . . . . . :
本地连接 IPv6 地址. . . . . : fe80::2067:dde0:6dda:3f21%27
IPv4 地址 . . . . . : 10.180.36.100
子网掩码 . . . . . : 255.255.240.0
默认网关. . . . . : fe80::25c7:fb9d:a833:8a0%27
                    fe80::f4d1:8efb:6f89:9835%27
                    10.180.32.1
```

接下来返回主界面，发现已经捕捉到了一些数据包，由于这次我们限定了捕捉数据包的类型，而 ICMP 协议数据包数量很少，事实上这里除了我们自己发送的数据包意外，并没有接收到任何其他基于 ICMP 协议的数据包，因此总数据包的量很少。

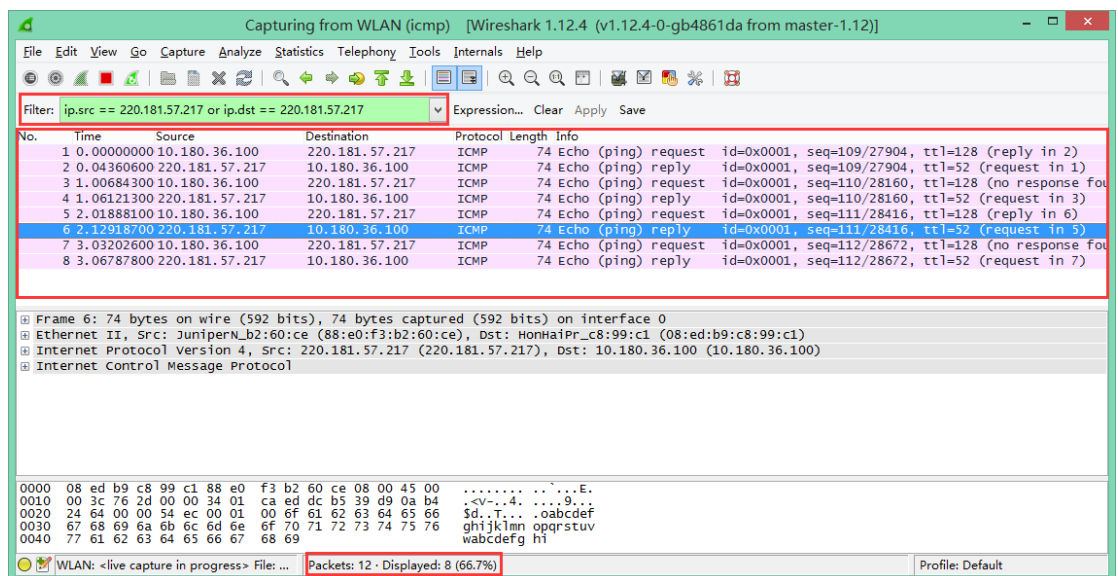


在图中，我们可以发现数据包的一些特点和内容，如下图所示，首先，所有数据包的协议都是 ICMP 的，这是我们设置了过滤器后筛选的结果，另外，我们发现前几个数据包的 Source 源地址和 Destination 目的地址都是本机地址（10.180.36.100）和百度的 IP 地址（220.181.57.217），总共 8 个包，分别对应了 4 次的本机请求包和服务端回应的数据包，也就是 ping 命令发出得 4 个包和接收 4 个包。

在每个数据包的最后，对应的数据包的大致内容，包括延迟时间和数据包生存最长时间等。



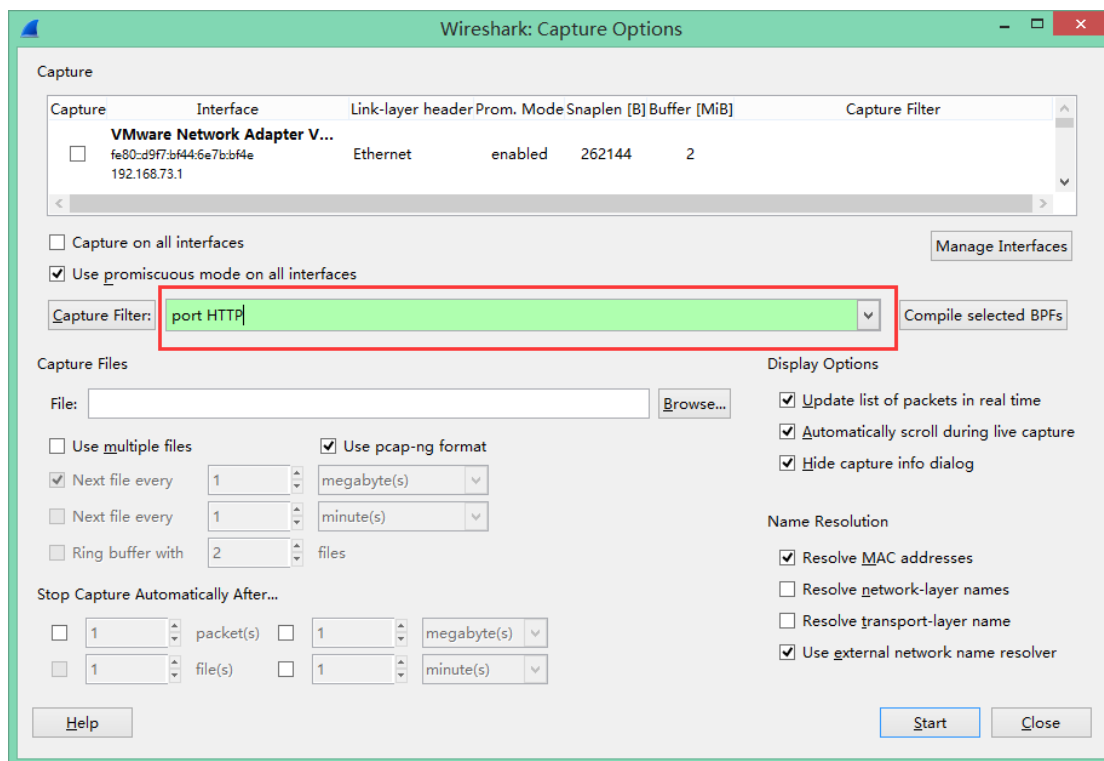
如果我们希望仅仅查看收发的数据包，还可以在 display filter 中输入相应的指令进行筛选。不同于 capture filter, display filter 只是在捕获数据包后对要显示的数据包进行筛选，但事实上那些没有显示的数据包依然被捕获。如下图所示：



如图，捕获了 12 个数据包，但只显示了 8 个。因为我们将显示的数据包限定为和百度有关的数据包，即源地址和目的地址中必须有一个是百度服务器的地址。通过 or 和 ip.src, ip.dst 来设置源 IP 地址和目的 IP 地址。

- 跟踪一次 HTTP 会话数据包

将 capture filter 中的筛选条件改为 port HTTP, 使过滤器过滤掉那些和 HTTP 协议无关的数据包。



由于现在绝大多数情况下使用的都是 HTTPS 协议，因此本次实验使用 telnet 的方法连接服务器，并使用 GET 命令，以 HTTP1.1 的协议传送文本，来捕获数据包。

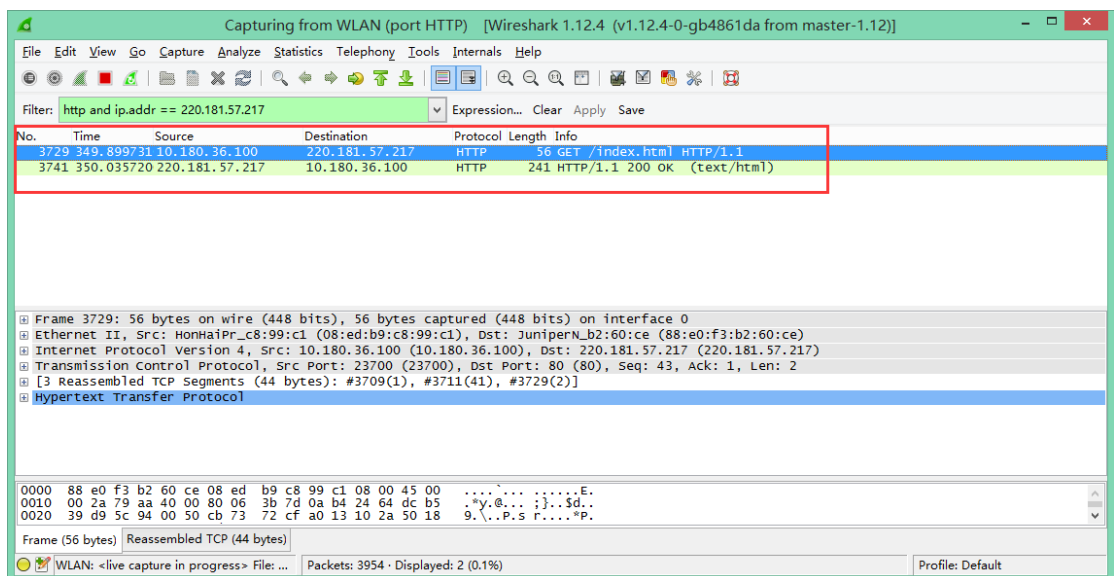
```
Telnet baidu.com

GET /index.html HTTP/1.1
HOST:baidu.com

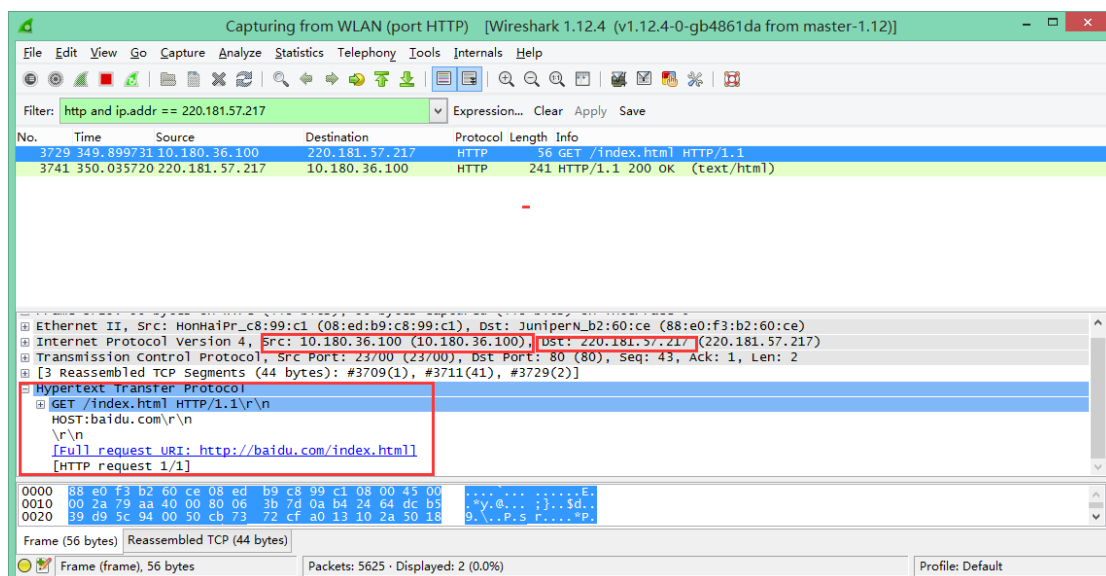
HTTP/1.1 200 OK
Date: Wed, 15 Apr 2015 11:01:24 GMT
Server: Apache
Cache-Control: max-age=86400
Expires: Thu, 16 Apr 2015 11:01:24 GMT
Last-Modified: Sat, 31 Dec 2011 09:54:00 GMT
ETag: "1cdb-4efedbb8"
Accept-Ranges: bytes
Content-Length: 7387
Connection: Keep-Alive
Content-Type: text/html

<!doctype html><html><head><meta http-equiv="Content-Type" content="text/html; charset=gb2312"><title>
百度一下，你就知道</title><style>html{overflow-y:auto}body{font:12px arial;text-align:center;
background:#fff}body,p,form,ul{margin:0;padding:0}body,form,#fm{position:relative}td{text-align:left
}img{border:0}a{color:#00c}a:active{color:#f60}#u{padding:7px 10px 3px 0;text-align:right}#m{width:6
80px;margin:0 auto}#nv{font-size:16px;margin:0 0 4px;text-align:left;text-indent:117px}#nv a,#nv b,.
btn,#lk{font-size:14px}#fm{padding-left:90px;text-align:left}#kw{width:404px;height:22px;padding:4px
7px;padding:6px 7px 2px\9;font:16px arial;background:url(http://www.baidu.com/img/i-1.0.0.png) no-r
epeat -304px 0;background-attachment:fixed;border:1px solid #cdcdcd;border-color:#9a9a9a #cdcdcd #c
dcdcd #9a9a9a;vertical-align:top}.btn{width:95px;height:32px;padding:0;padding-top:2px\9;border:0;ba
ckground:#ddd url(http://www.baidu.com/img/i-1.0.0.png) no-repeat;cursor:pointer}.btn_h{background-p
osition:-100px 0}#kw,.btn_wr{margin:0 5px 0 0}.btn_wr{width:97px;height:34px;display:inline-block;ba
ckground:url(http://www.baidu.com/img/i-1.0.0.png) no-repeat -202px 0;top:1px;position:relative}#l
k{margin:33px 0}#lk span{font:14px "宋体"}#l1{height:60px}#l1h{margin:16px 0 5px;word-spacing:3px}#mC
on{height:18px;line-height:18px;position:absolute;right:7px;top:8px;top:10px\9;cursor:pointer;paddin
g:0 18px 0 0;background:url(http://www.baidu.com/img/bg-1.0.0.gif) no-repeat right -134px;background
-position:right -136px\9}#mCon span{color:#00c;cursor:default;display:block}#mCon .hw{text-decoration
n:underline;cursor:pointer}#mMenu{width:56px;border:1px solid #9a99ff;list-style:none;position:absol
ute;right:7px;top:28px;display:none;background:#fff}#mMenu a{width:100%;height:100%;display:block;li
ne-height:22px;text-indent:6px;text-decoration:none}#mMenu a: hover{background:#d9e1f6}#mMenu .ln{hei
ght:1px;background:#ccf;overflow:hidden;margin:2px;font-size:1px;line-height:1px}#ep,#cp a{color:#77
c}#sh{display:none;behavior:url(#default#homepage)}</style></head>
<body><p id="u"><a href="/gaoji/preferences.html">搜索设置</a>&nbsp;<a href="http://passport.
baidu.com/?login&tpl=mn">登录</a></p><p id="lg"></p><p id="nv"><a href="http://news.baidu.com">?
nbs;闻</a> <b>网&nbsp;&nbsp;&nbsp;页</b> <a href="http://tieba.baidu.com">贴&nbsp;&nbsp;&nbsp;吧</a> <a href="http://z
hidao.baidu.com">知&nbsp;&nbsp;&nbsp;道</a> <a href="http://mp3.baidu.com">MP3</a> <a href="http://image.baidu
.com">图&nbsp;&nbsp;&nbsp;片</a> <a href="http://video.baidu.com">视&nbsp;&nbsp;&nbsp;频</a> <a href="http://map.baidu.com
">地&nbsp;&nbsp;&nbsp;图</a></p><div id="fm"><form name="f" action="/s"><input type="text" name="wd" id="kw" max
length="100"><input type="hidden" name="rsv_bp" value="0"><span class="btn_wr"><input type="submit"
value="百度&nbsp;呢?" id="su" class="btn" onmousedown="this.className='btn btn_h'" onmouseout="this.class
搜狗拼音输入法 全 :
```

使用 telnet 连接服务器并请求 HTTP 页面数据的方法已经在上次实验中描述过了，这里不多做赘述。请求得到响应并成功收到 HTML 数据后，停止数据包的捕获，回到 wireshark 主界面，可以看到：

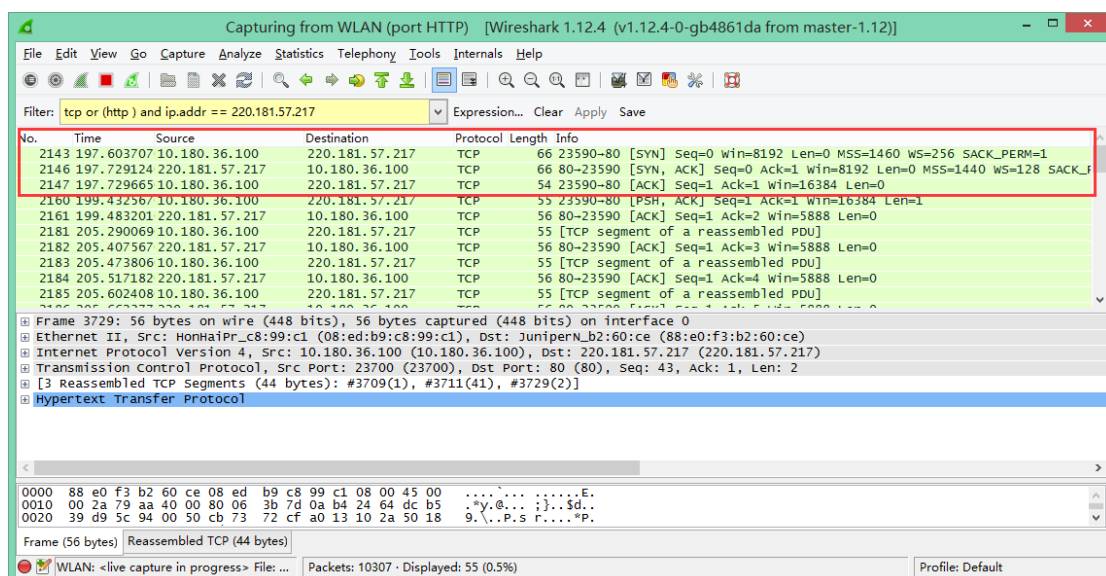


两个包中，第一个包是主机向服务器发送请求的包，可以在最右侧看到内容包括 GET /index.html HTTP/1.1 的内容，第二个包是服务器响应，200 代表成功。下面查看第一个包的内容。



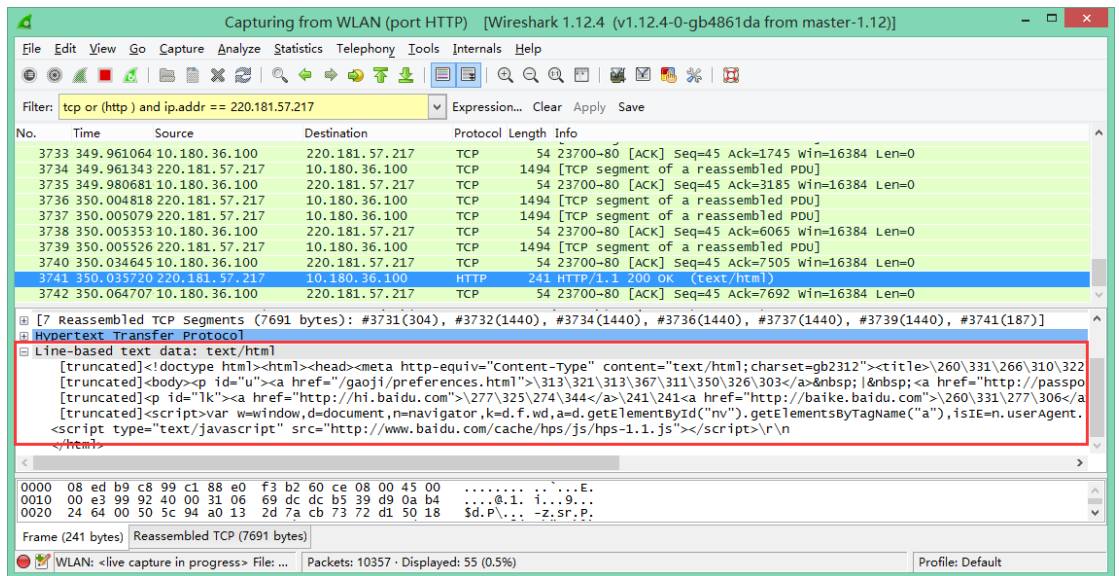
可以在包的详细内容中看到数据包源地址，目的地址，并在 HTTP 中找到了请求的详细内容。

我们将筛选范围扩大，可以看到 TCP 协议的包：



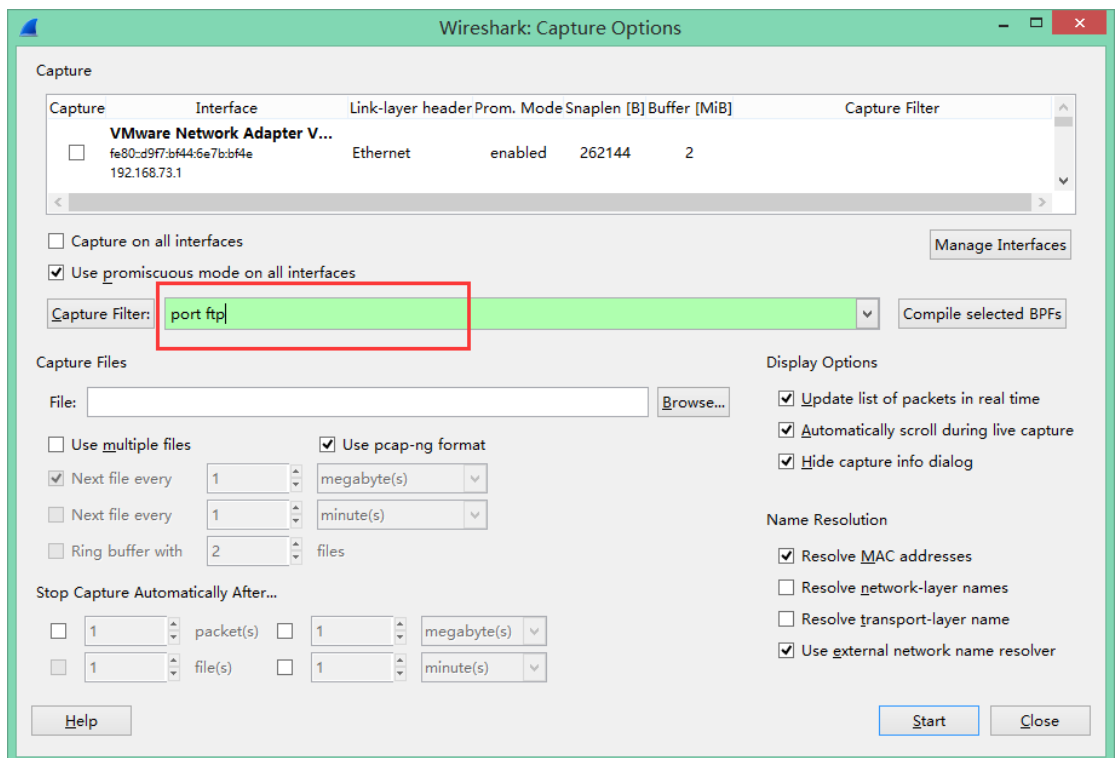
由于 HTTP 协议是基于 TCP 协议建立连接的，所以会有三个 TCP 的数据包，分别代表了三次握手协议，用于建立连接。

另外，可以在收到的 HTTP 数据包中找到 HTML 的详细内容。如下图所示：

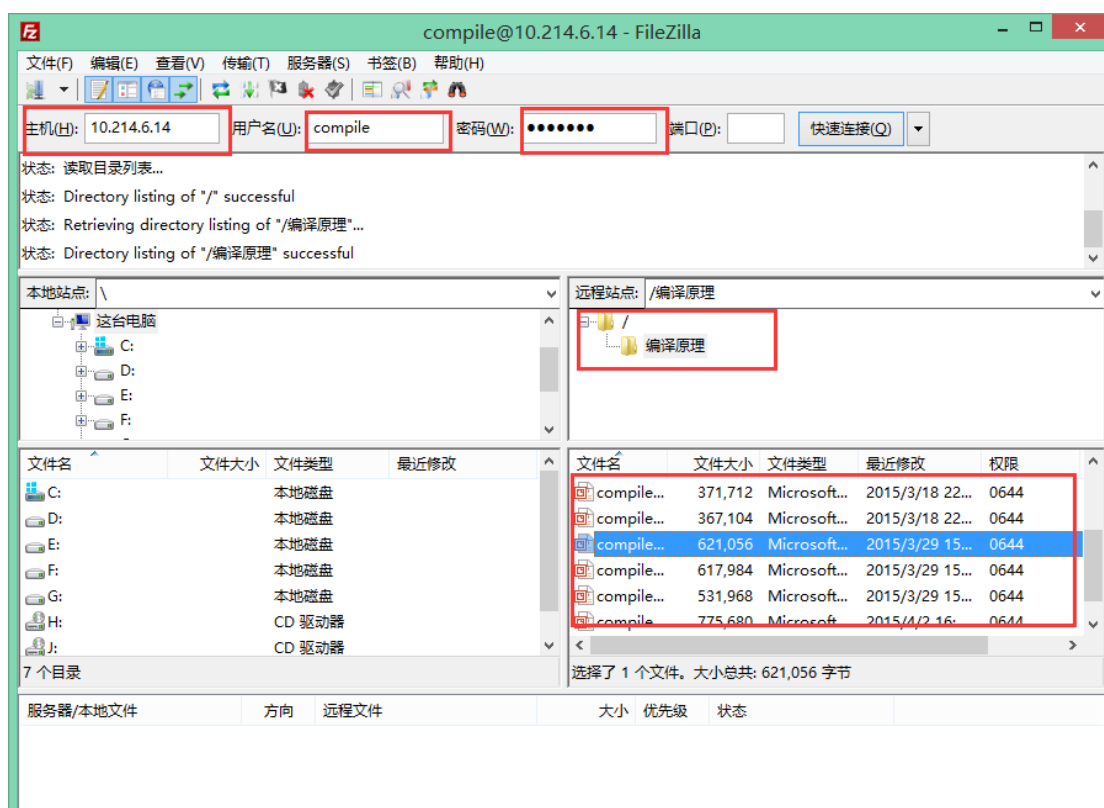


● 跟踪一次 FTP 会话的数据包

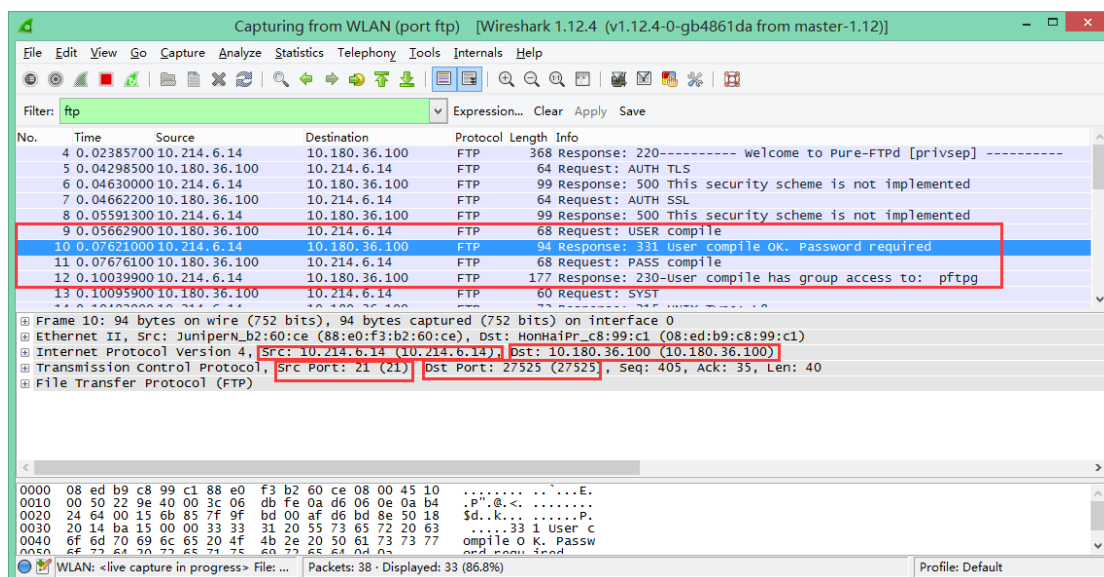
将抓取范围限定到 FTP 协议的数据包



用 FTP 客户端登陆 FTP:

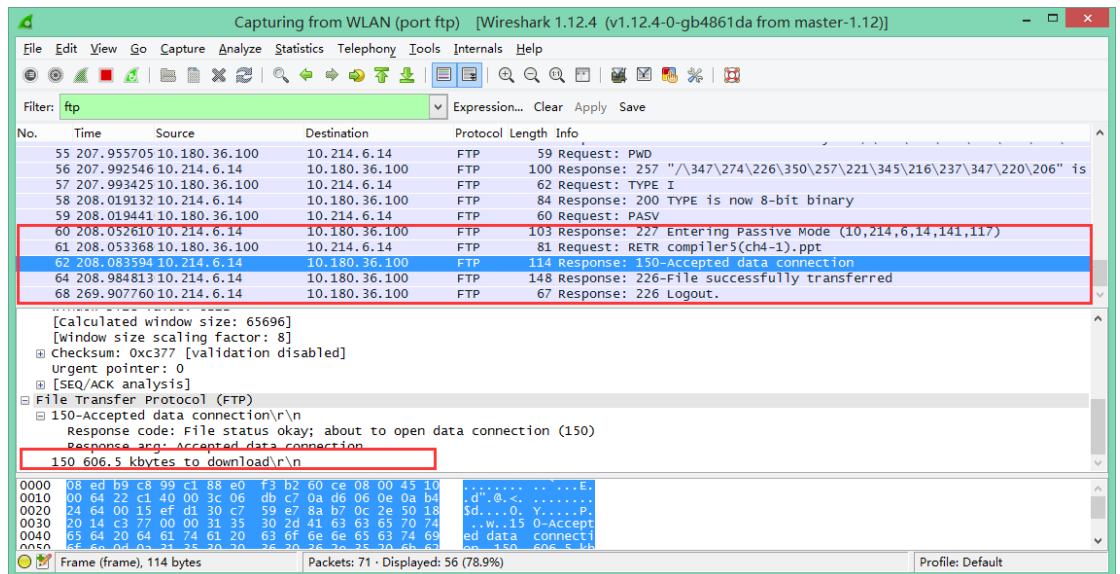


选中其中一个文件下载到本机，并结束数据包的抓取，返回主界面：



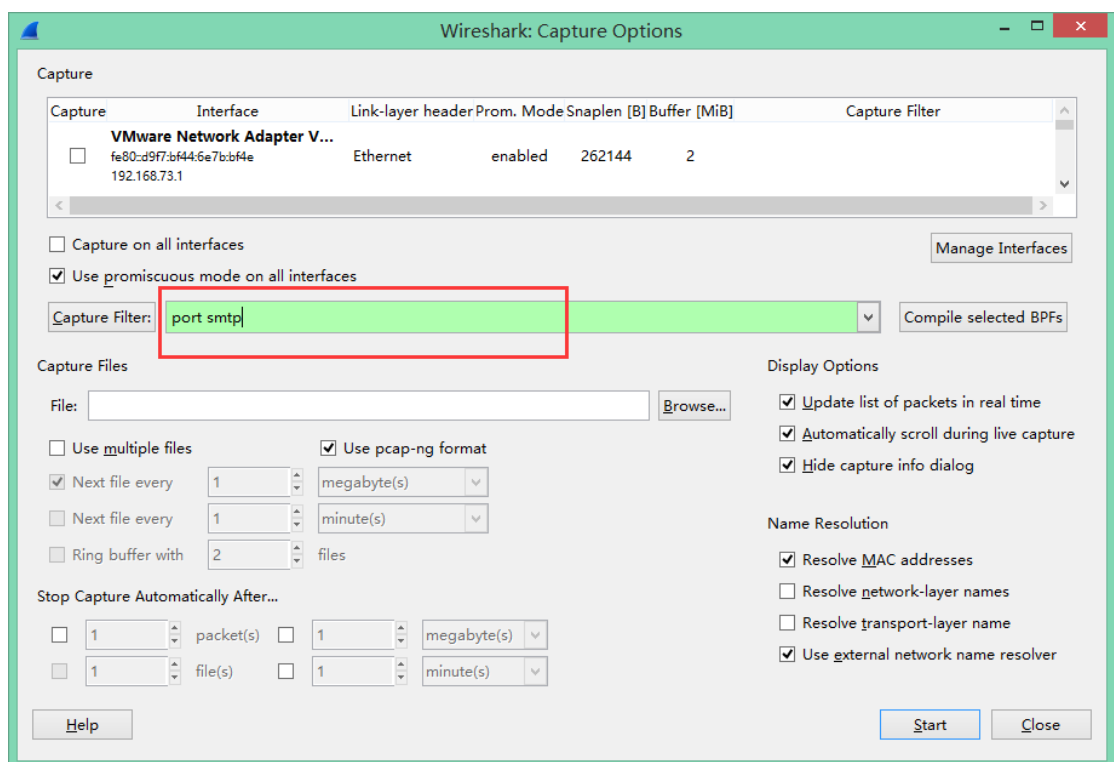
数据包列表中的红框部分的数据包主要用于身份的验证。点击其中一个数据包可以得到本机和服务器的 IP 地址，以及两者会话的端口。比如本机使用 27525 端口，服务器使用 21 端口

找到传送文件的数据包，可以获得数据包的大小等信息，如图：

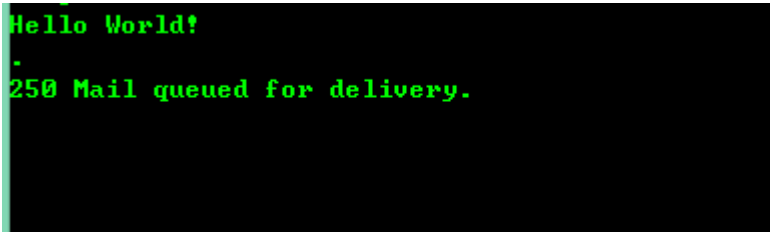


● 跟踪一次 SMTP 会话的数据包

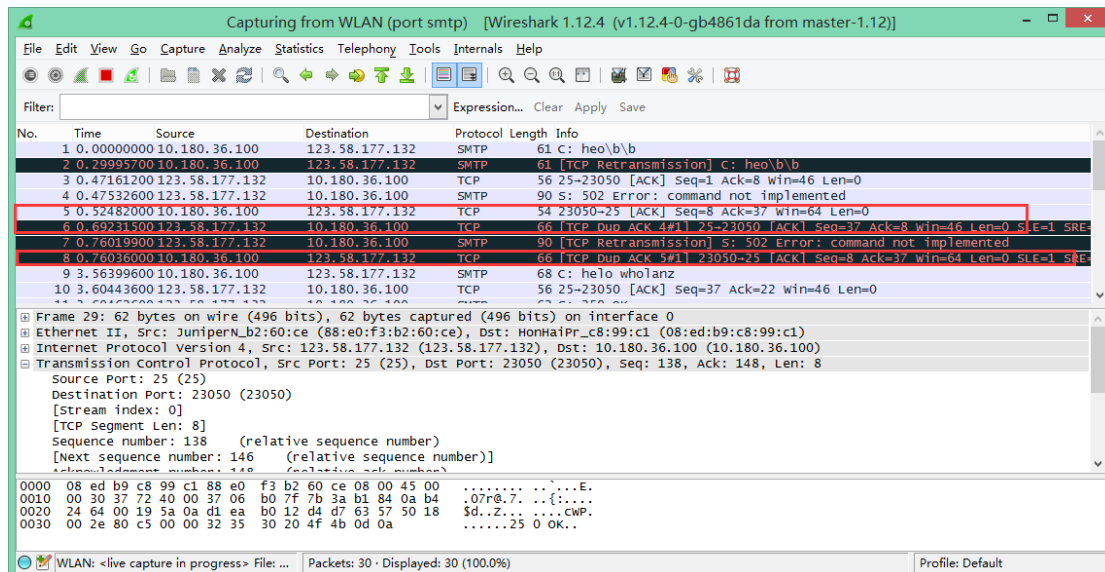
将抓包范围限定在 SMTP 协议的数据包：



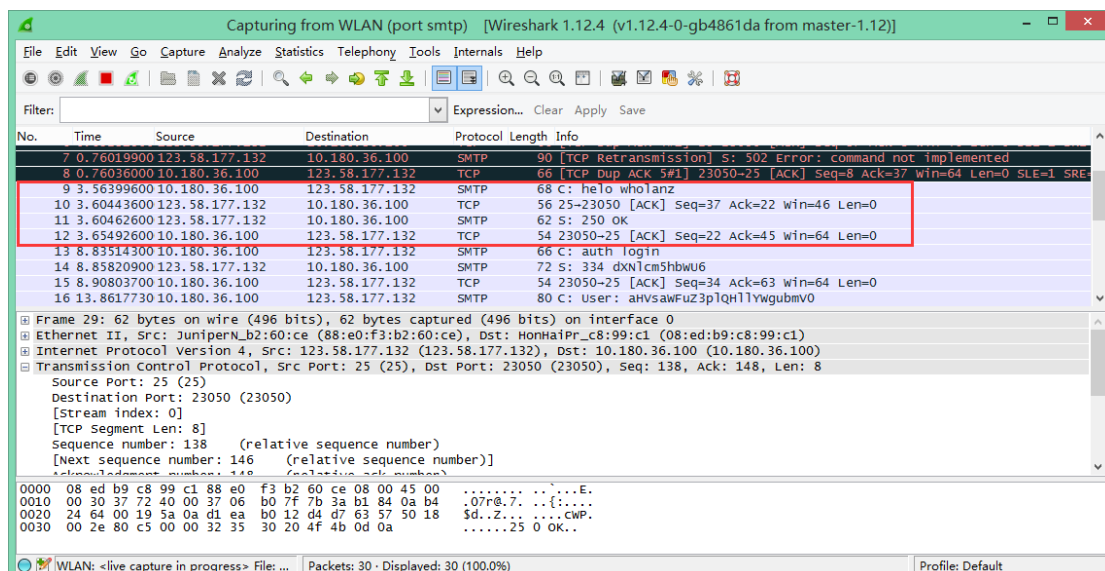
由于现在大多数邮件并不是通过 SMTP 协议发送的，所以本次实验通过 telnet 连接到服务器并发送一封邮件，过程已经在上次的实验中详细描述过，这里不多做赘述



SMTP 协议同样是基于 TCP 协议建立连接的，因此我们同样可以看到代表三次握手协议的三个 TCP 数据包：

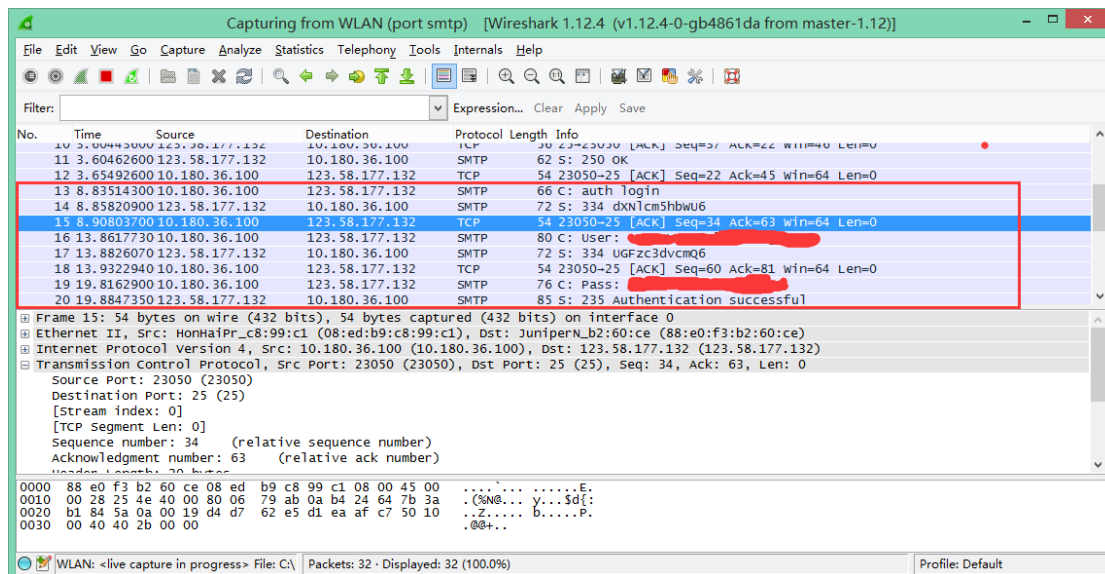


连接到服务器后，还需要进行身份确认：

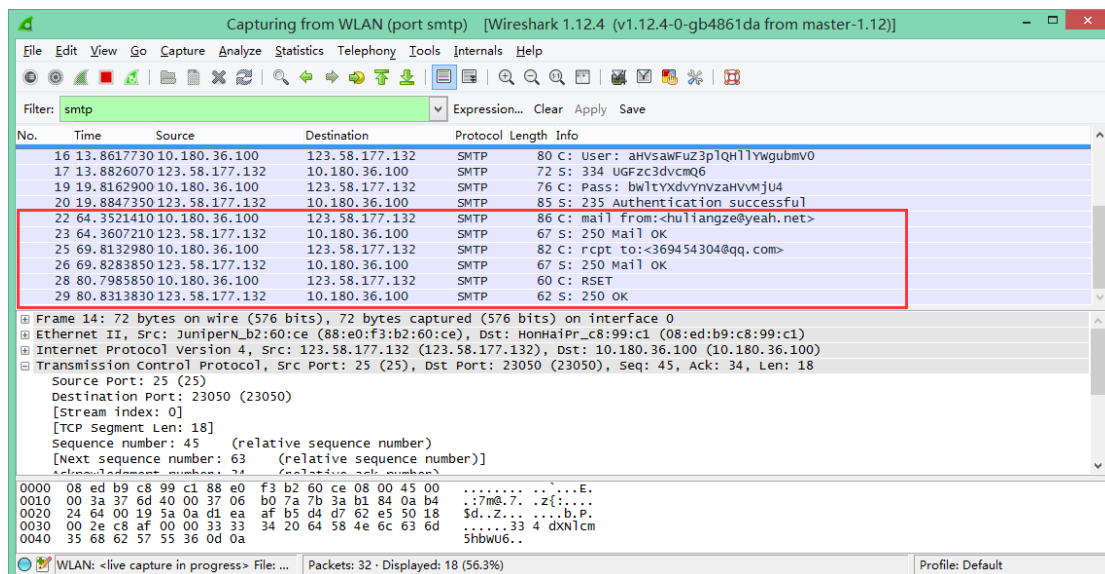


图中的两个 SMTP 包内容分别是本机向服务器主动表明身份以及服务器确认。

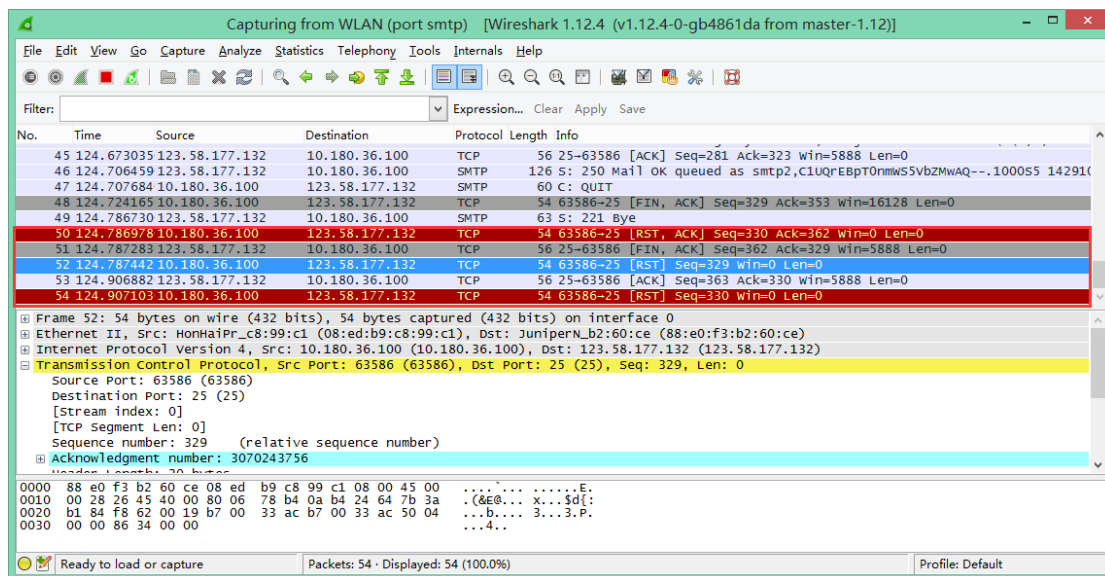
接下来需要用 base64 加密过后的邮箱账号密码登陆服务器：



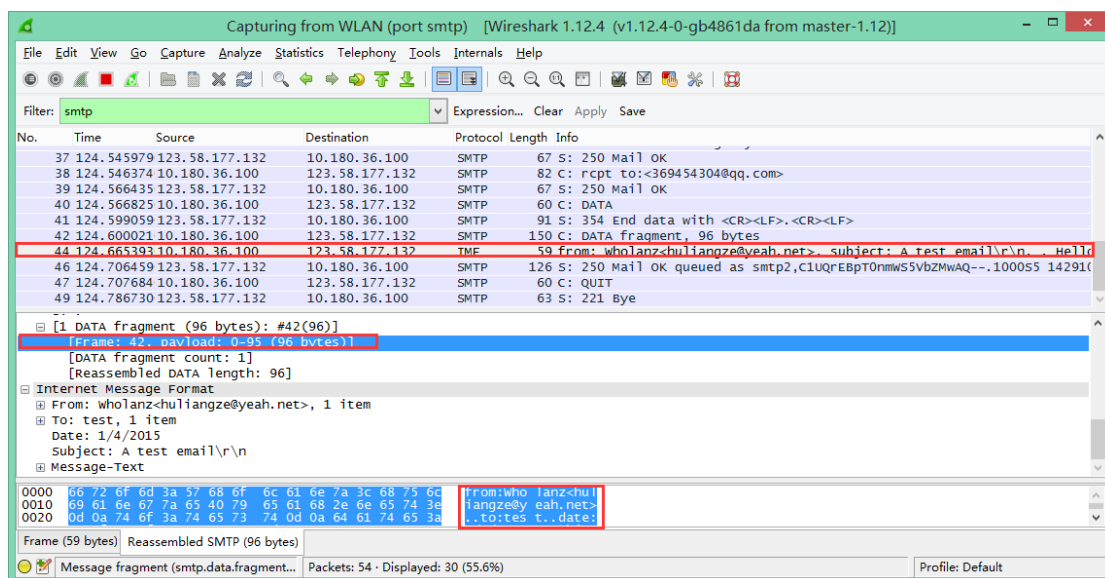
图中的几个 SMTP 包内容分别是服务器要求输入账号密码以及本机发送账号密码，具体过程和原理在上次实验中已经讲述过，这里不多做赘述
接下来确认邮件发送者以及邮件接收者：



几个 SMTP 包的内容分别包括了邮件发送者地址，邮件接收者地址，已经服务器确认的信息。
最后和服务器断开连接。



另外，还可以在下图的数据包中找到相应的邮件属性信息：



比如，在红框的数据包中，我们发现邮件大小有 96K，并且在数据包中找到了邮件的详细内容（下方红框中标注），在 16 进制内容中翻译后得到结果。

实验心得

本次实验让我学会了如何巧妙的使用 wireshark 软件抓取网络中的数据包，并分析数据包的内容，最大的收获包括如下几点：

1. 抓取数据包和分析数据包的过程让我对 HTTP 协议、ICMP 协议、FTP 协议以及 SMTP 协议有了更加深入的了解，并熟悉了协议的行为过程。
2. 学会了如何利用筛选器筛选数据包
3. 对网络协议产生了更加浓厚的兴趣，期望在以后的实验中学到更多