**北京邮电大学软件学院**

**2019-2020 学年第 1学期实验报告**

**课程名称： 计算机网络\_\_\_\_\_\_\_\_**

**实验名称： 实验三 传输层实验\_\_\_\_\_\_\_**

**实验完成人：**

**姓名：**\_\_\_\_吴志镛\_\_\_\_**学号：**\_\_\_\_\_2017211869\_\_\_

**成绩：**\_\_\_\_\_\_\_\_

**指导教师：**\_\_\_\_\_\_\_ \_\_**\_\_雷友珣\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**日 期： 2019 年 12 月 12 日**

1. **实验目的**

通过本实验使学生理解并掌握TCP连接，掌握TCP报文（TCP Segment）字段组成、字段的作用和使用方法。

1. **实验任务**

编写基于TCP套接字（Socket）的TCP服务器应用程序和TCP客户端应用程序，TCP服务器应用程序支持与多个TCP客户端应用程序同时建立TCP连接；将TCP客户端应用程序和服务器应用程序分别部署到本课程实验二（“网络层实验”）搭建的网络环境中的主机H1和H2上。分别观测下列情况下TCP协议过程及TCP报文字段取值：

1. TCP服务器应用程序未启动情况下，TCP客户端应用程序向服务器发出TCP连接建立请求；
2. TCP服务器应用程序已启动，但主机H1和H2之间的通路上的路由器丢弃主机H1发出的IP包的情况下，TCP客户端应用程序向服务器发出TCP连接建立请求；
3. 多个TCP客户端应用程序与TCP服务器应用程序建立TCP连接后，TCP客户端与TCP服务器间传递数据；并模拟应用程序一次发送数据量大于链路数据包最大长度情况下的TCP报文的发送/接收。
4. 关闭TCP连接。
5. **实验内容**
6. 恢复本课程实验二（“网络层实验”）中的网络环境，并使主机H1和主机H2在网络层连通。
7. 编写基于TCP套接字的TCP客户端应用程序和TCP服务器应用程序。TCP客户端程序和TCP服务器程序可以采用不同编程语言开发；TCP服务器应用程序支持与多个TCP客户端应用程序同时建立TCP连接；TCP连接建立后，可以在TCP连接上传递结构化消息，消息中的字段取值长度为可变长度。
8. 将TCP客户端应用程序和TCP服务器应用程序分别部署到主机H1和主机H2上。
9. 在未启动TCP服务器应用程序的情况下，在主机H1上启动TCP客户端应用程序，观测TCP客户端应用程序在TCP连接建立过程中的收发的TCP报文、TCP连接建立结果。
10. 在启动TCP服务器应用程序的情况下，在路由器R1上设置丢弃主机H1发出的IP分组的防火墙规则，使得TCP客户端应用程序发出的TCP连接建立请求无法到达主机H2，然后在主机H1上启动TCP客户端程序，观测TCP客户端应用程序在TCP连接建立过程中的收发的TCP报文、TCP连接建立结果。
11. 删除在路由器R1中设置的丢弃主机H1发出的IP分组的防火墙规则，使得主机H1发出IP分组能够到达主机H2。在主机H2启动TCP服务器应用程序，在主机H1上分别多次运行TCP客户端应用程序，在TCP服务器和TCP客户端间建立多条TCP连接。观测TCP连接建立的三次握手过程、观测TCP连接建立后TCP连接上TCP报文传递过程。
12. TCP连接建立后，在客户端与TCP服务器间传递数据，观测应用程序一次发送数据量大于链路数据包最大长度情况下的TCP报文的发送/接收。
13. 分别先后关闭TCP客户端程序和TCP服务器程序，观测TCP连接拆除过程。
14. **实验环境**
15. Windows系统主机。
16. Wireshark软件，软件下载网址：https://www.wireshark.org/
17. Linux环境下Socket网络编程开发和运行环境：Java程序开发环境VScode 。

**五、 实验步骤**

a)

首先先说明IP设置，我们的IP设置如下：

1. H3
   1. 在 VMnet2 中的地址为 192.168.21.30
2. R1
   1. 在VMnet2 中的地址为 192.168.21.10
   2. 在VMnet3 的地址为 192.168.136.10
3. R2
   1. 在VMnet3 的地址为 192.168.136.20
   2. 在VMnet4 的地址为 192.168.194.20
4. H5
   1. 在 VMnet4 的地址为 192.168.194.40

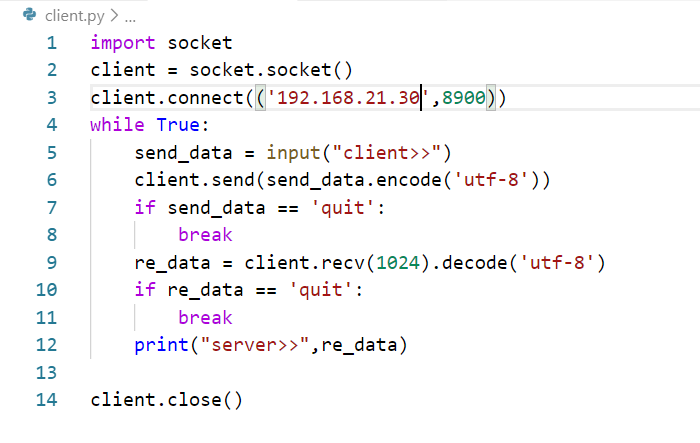
**标注：因为实验二之后就删除了上次的实验内容，重新创建了新的虚拟机H3和H5其中H3代表实验实验指导书中的H1，H5代表指导书中的H2.，对本实验没有太多影响。**

b)

**接下来是写的TCP通信程序：**

**注意:本程序使用的是python语言，python语言socket通信如果不指定的话默认的是TCP通信.**

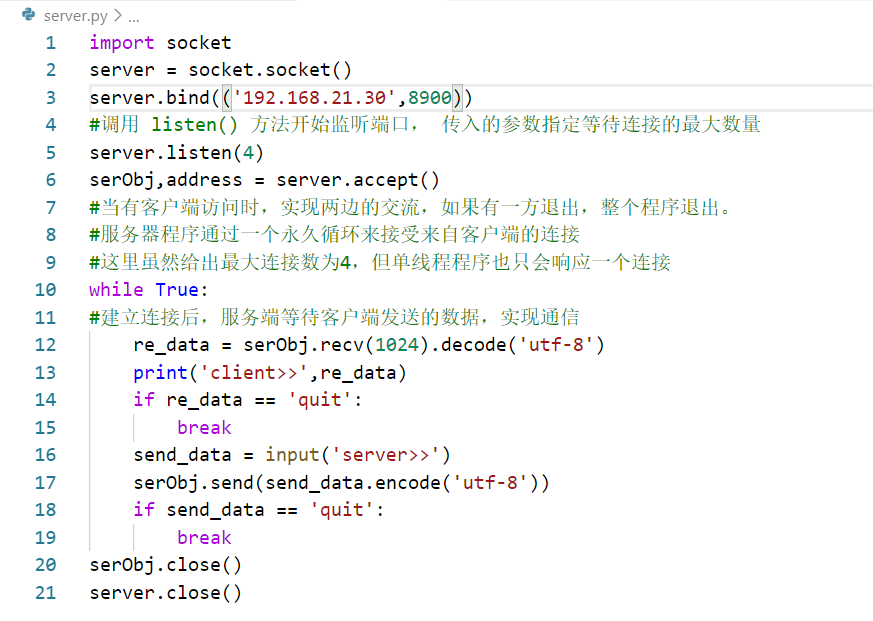
客户端：



程序说明：客户端通过socket.connect向服务器端指定端口建立连接，

当出现client>>代表客户端可以向服务端发送消息，当输入quit时取消与服务端的连接。

服务端程序如下：



程序说明：服务端通过socket.bind将进程绑定到自己ip的指定端口，之后可以等待客户端与服务端建立连接，客户端发送完消息后，服务端出现server>>代表服务端可以向客户端发送数据。当输入quit时，取消与客户端的连接。

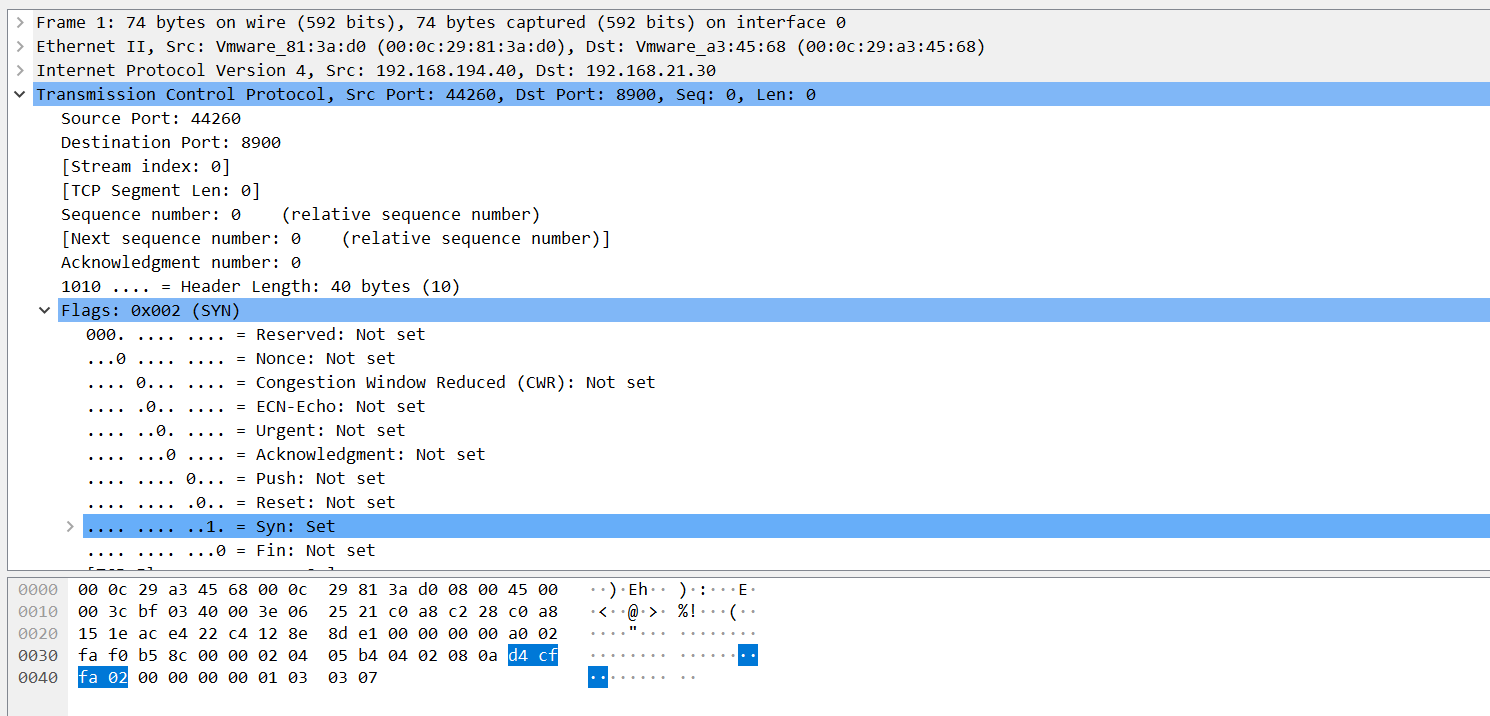
c)TCP的交互模拟(三次握手)

**未开启服务端时:**

客户端抓包结果如下:

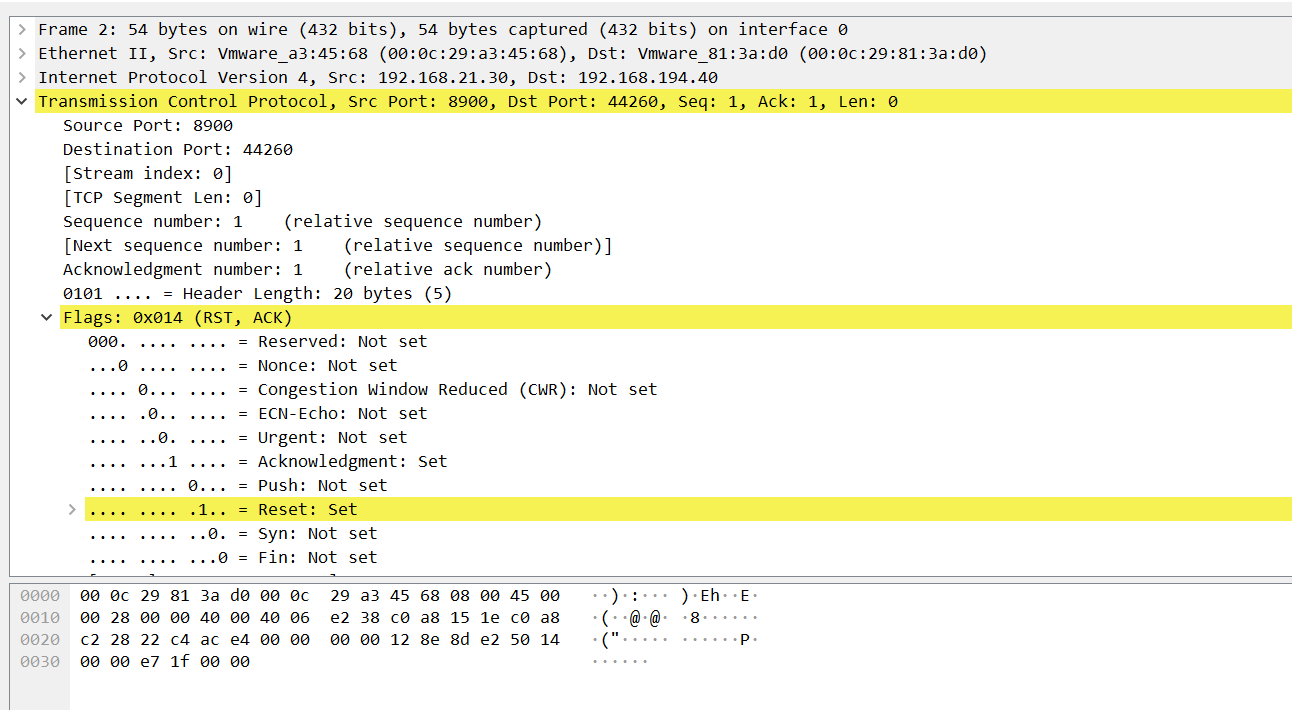


请求的包如下:



SYN为1,Reset为0.SYN为1说明这是个请求消息帧.

响应包如下:

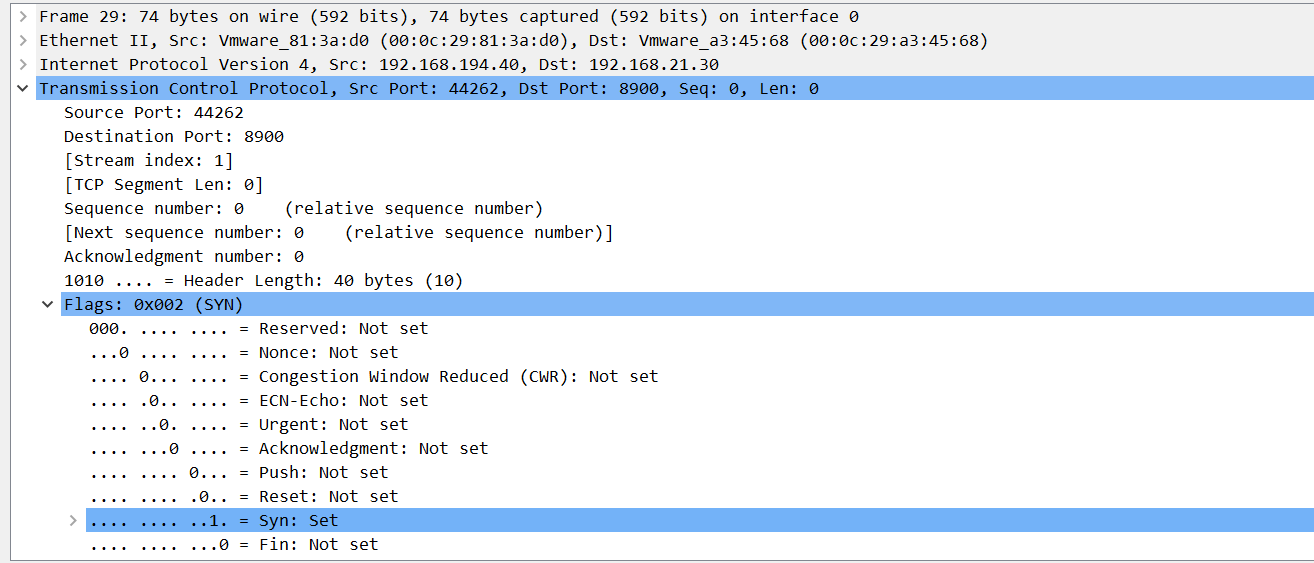


Reset为1表示发生了错误

SYN为0

**建立连接时:**

**客户向服务端发送的包:**

****

**分析如下:**

源端口为 44262 ，目标端口为 8900(程序设定的)

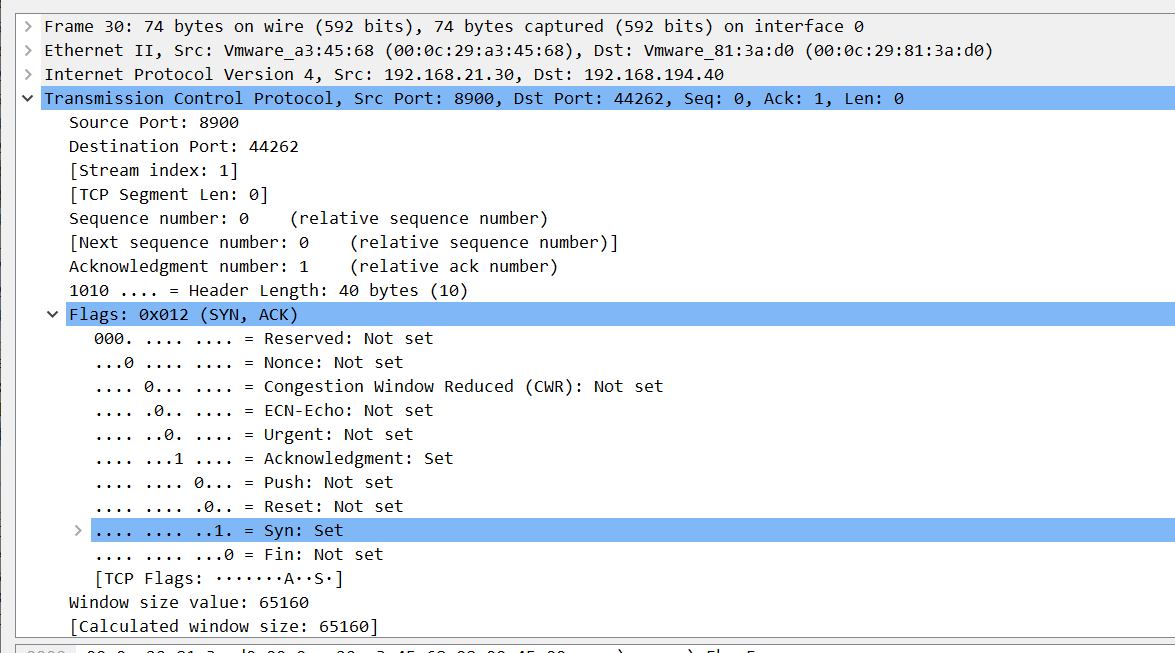
序列号字段为 0

Ack 字段为 0

Syn = 1 表示连接建立请求

fin = 0表示消息还未传输完

**服务端响应客户端的包:**

****

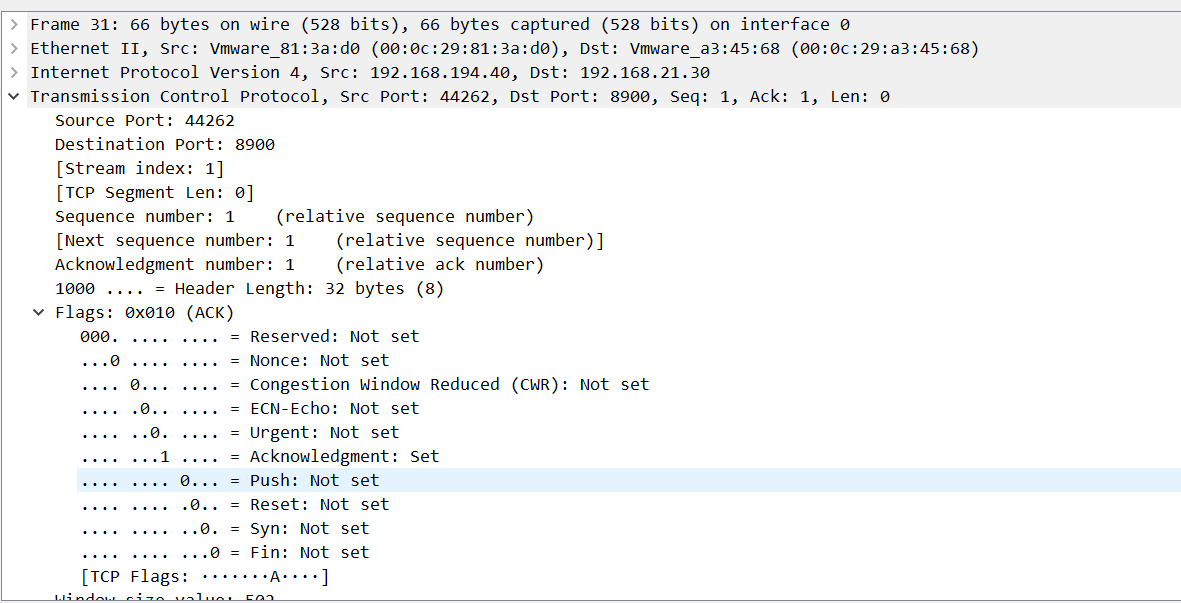
源端口为 8900 目标端口为44262

Seq 为0

ack字段 = 1，表示希望下一次受到的字段的序列号为1

syn = 1 ack=1 表示对建立连接的确定，syn = 1 默认这是一个 连接恳求或是毗邻接管报文，而 ack =1 则识别了其为 连接恳求报文，并且只有 ack位=1 确认字段才有效

**客户端再次响应服务端的包:**

****

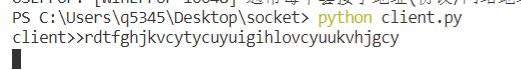
**以上是代表TCP通信的三次握手建立连接的过程,具体的分析如下:**

Seq序列号字段变成了1 这和 服务器之前发送的帧的 ack字段相同。Ack 字段=1 同样表示希望接收到来自服务器的 seq =1 的帧

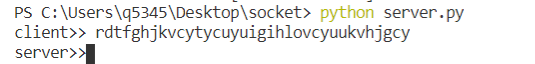
Syn = 0表示不再是理解请求的帧。Ack 位 = 1 确认号(ack)字段有意义

**发送消息时:**

客户端传送消息:

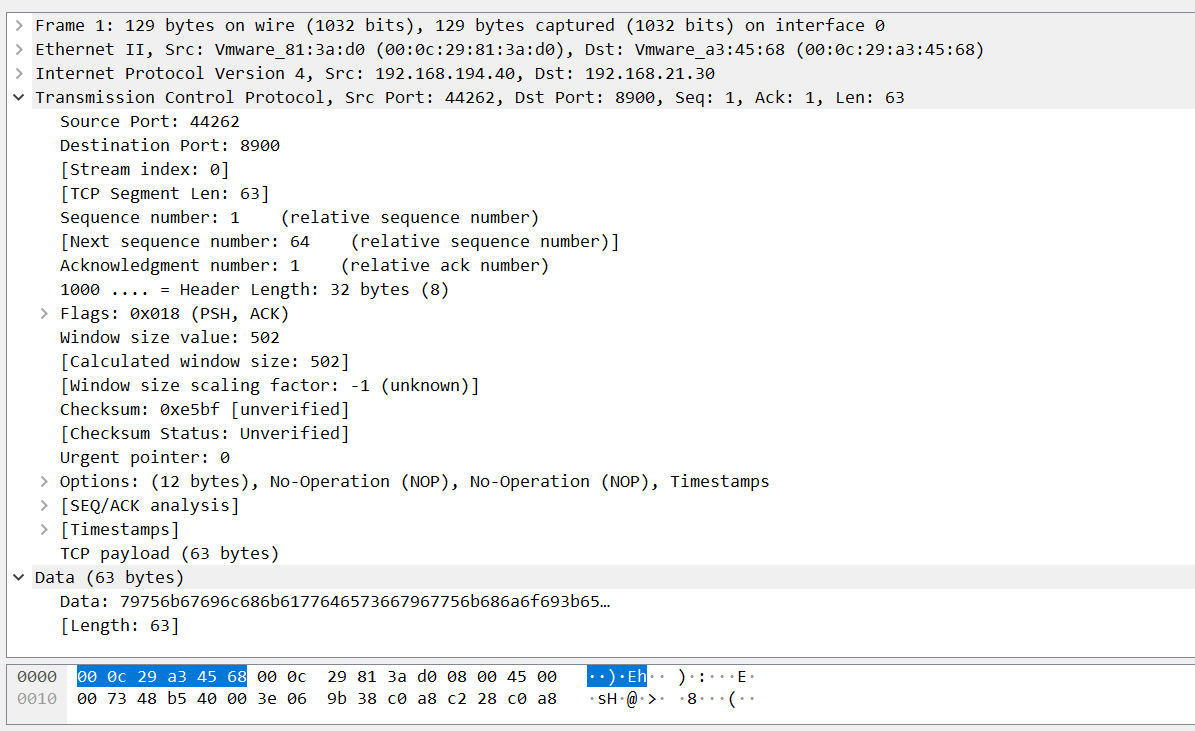
****

服务端接收到消息:

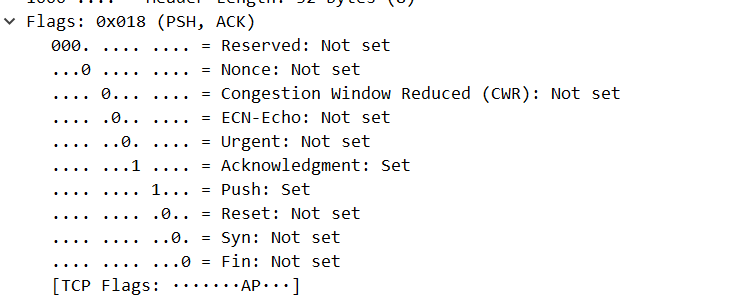
****

抓包结果如下:

**客户端向服务端发送的帧:**

****

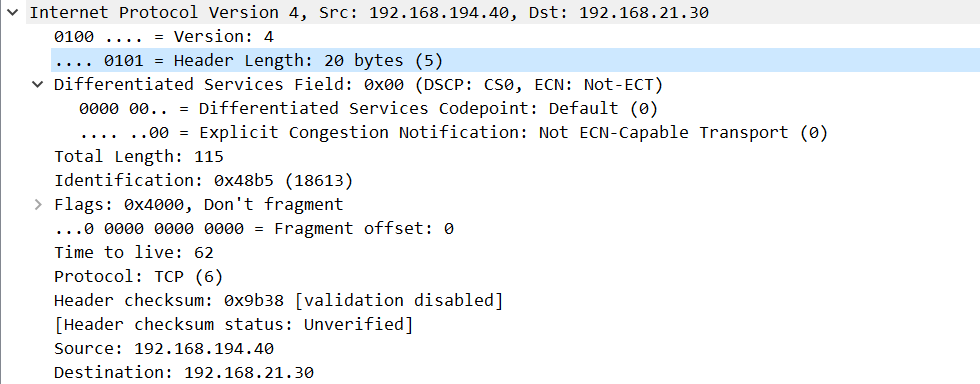
**分析如下:**

****

Ack和PUSH都设置为1表示,而客户端收到了服务端传送回来的建立连接消息,并且要发送消息.Data(63 bytes)还要发送一个长度为63bytes的数据.

其中,我们从包中可以看到:  
源地址为192.168.192.40 端口为44262

目的地址为192.168.21.30 端口为我们程序中设定的8900

****

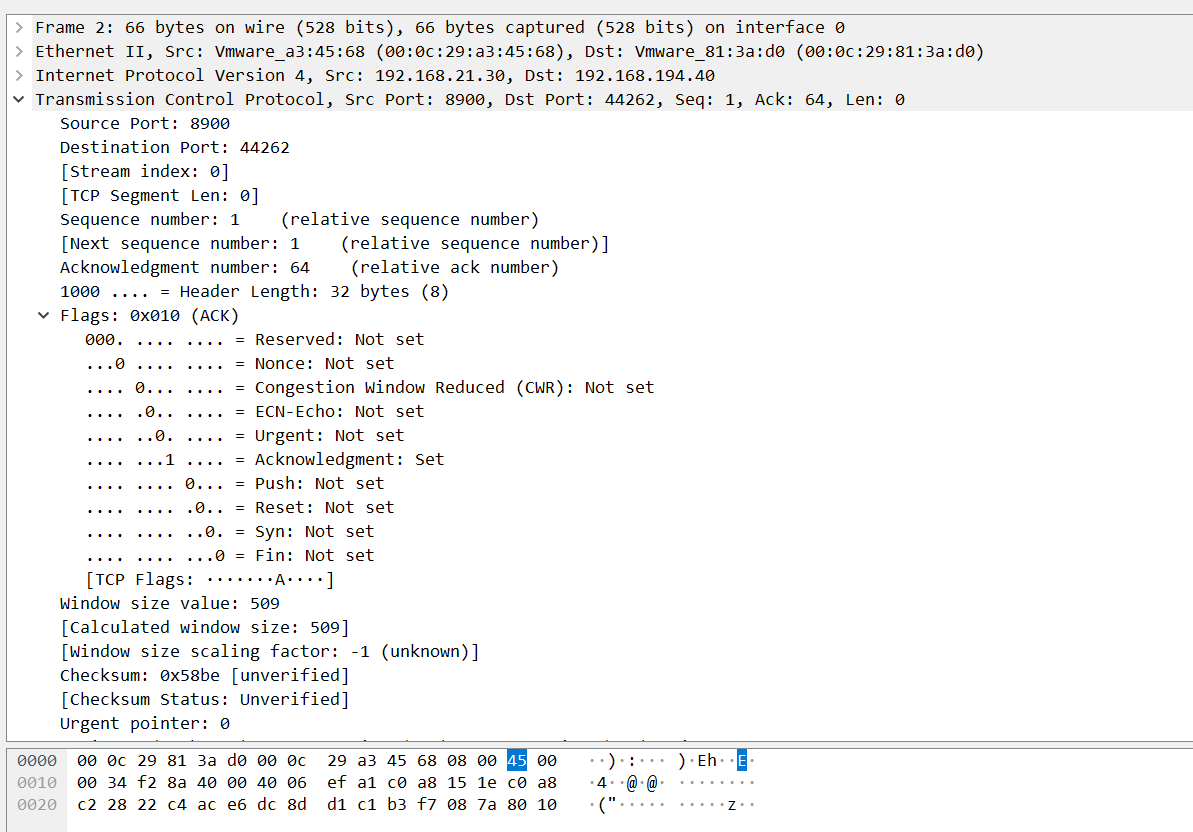
协议字段为 0100 表示 IPV4

IHL 字段为0101 表示 5 即 包头长度为20bytes

Total length 字段 为 115 表示整个 ip分组长度为 115byte

Protocol 字段为6 表示使用的是TCP协议

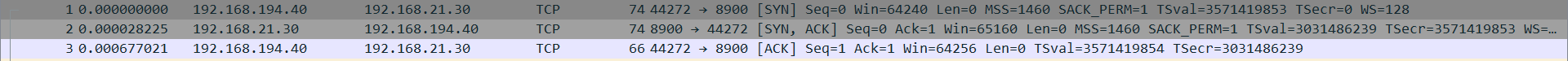
**服务端响应客户端的包:**

****

其中ackonwledgement设置为1代表收到了消息

**另一个客户端的三次握手:**

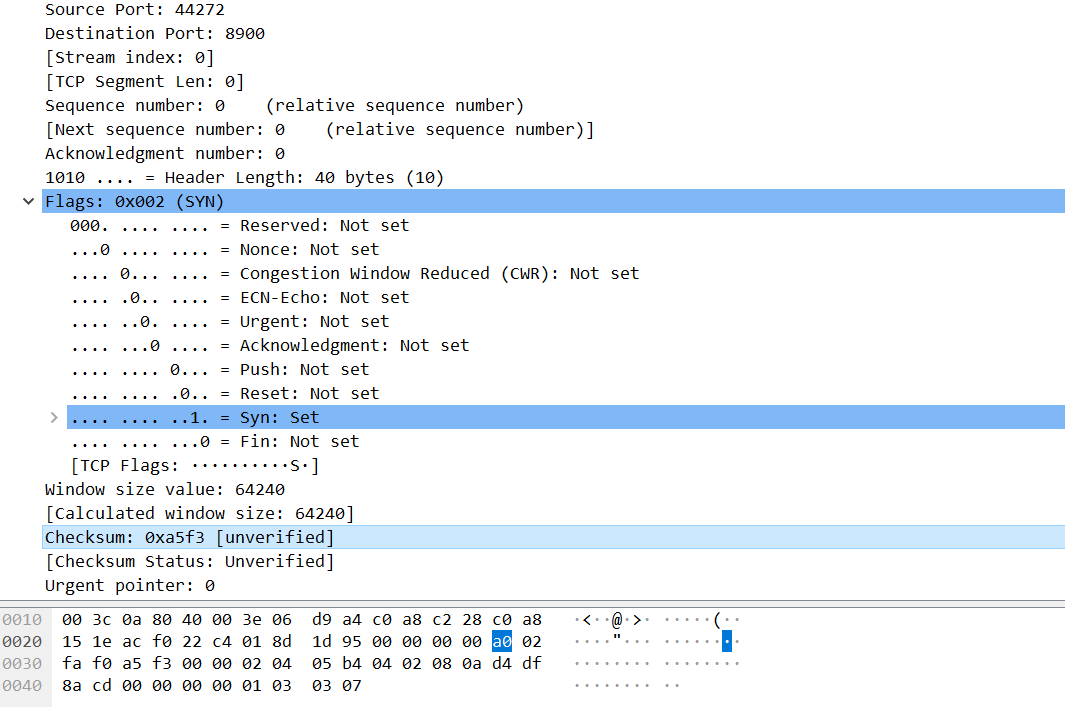
抓包如下:

****

包内的大致信息于第一个客户端的三次握手大致相同,这里讲述一些两者的不同:

首先从大致信息我们就发现了虽然发送方ip一样但是端口却却不一样,两次的端口分别为44264和44272.

**具体如下:**

****

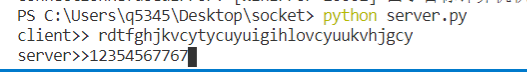
**端口为44272**

Id，因为这是唯一标识帧的 字段

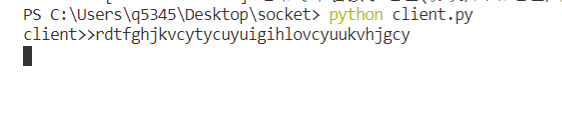
Header 校验和，校验和是根据所有的位来计算的，故不会相同  
Sequence number，虽然seq 在 wireshark 中显示的是 0 ，但是0只是相对位，而真实的 seq 的值则为 二进制中描述的那样，不尽相同

Option 下的 timestamp value，因为发送的时间不同 故 timestamp 时间戳的值不同

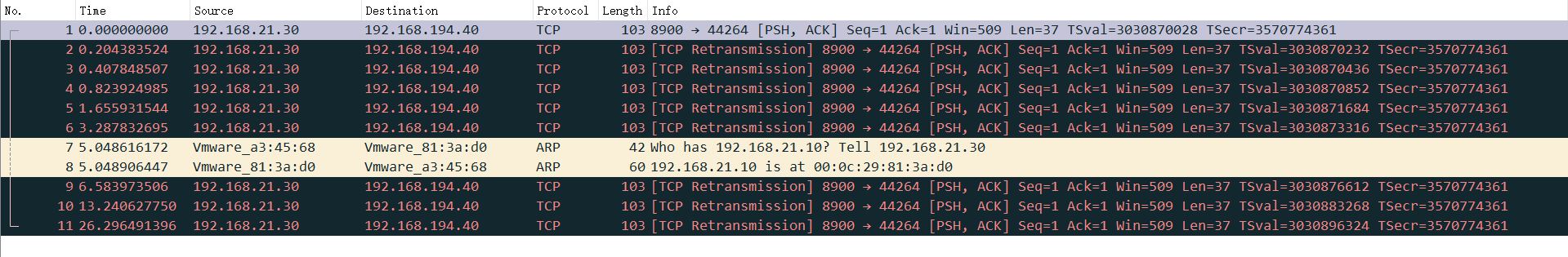
**开启防火墙之后,运行服务端结果如下:**

****

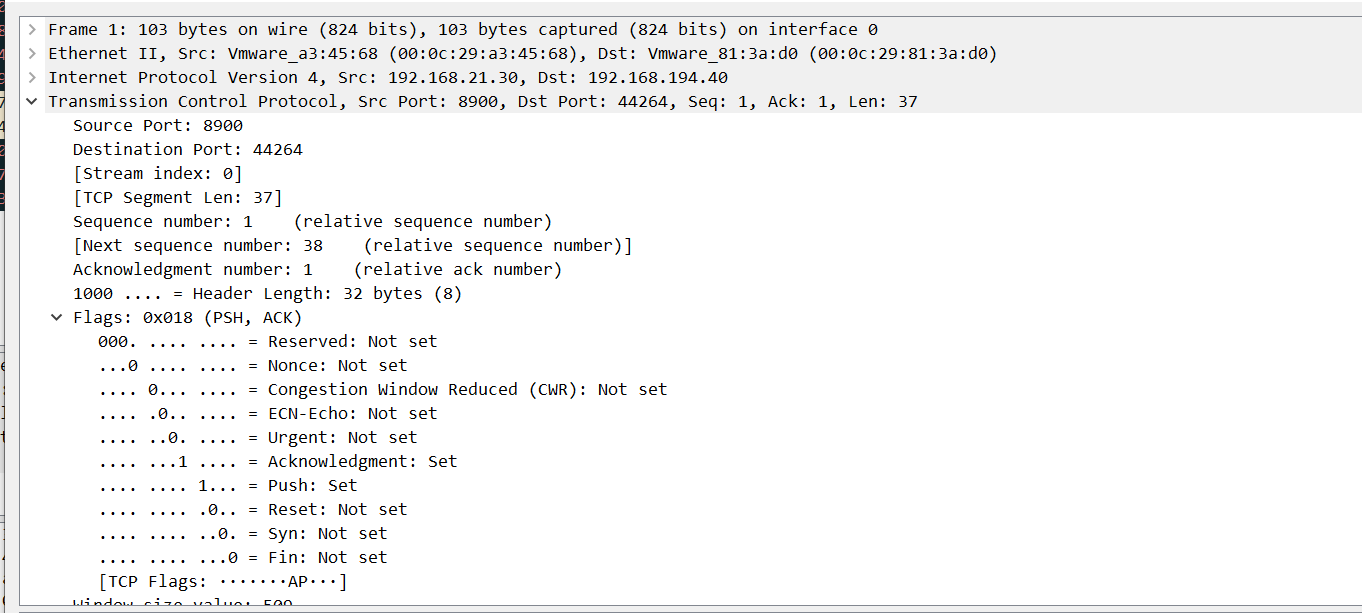
**客户端的结果:**

****

**客户端未接受到任何消息:**

服务端抓包结果如下:  


从抓包中我们发现,我们只抓到了服务端发送信息的包,而且客户端并没有做出响应:



分析如下:

包中ack和push为1表明在我们开启防火墙之前,还可以接收消息,并且向客户端发送了一些带有数据的包

从包中我们看到服务端向客户端发送了发送信息的消息,但是并未得到回应

1. **实验心得**

通过这次实验，对网络层ip地址的配置规划有了更直观的理解，通过在四台虚拟机之间路由转发的过程模拟，深入了解了网络层服务过程。同时也跟熟悉了Linux系统中配置IP的操作已经网络命令操作。

此外我想说，我实在是太难了，开始的这个实验只有2学时，但是虚拟机安装系统花了好久，而且同时启动4台虚拟机对电脑要求太高了，开始的时候经常闪退，最后花了566买了一个8G内存条安装上才能实验。

而且，老师的实验指导书该更新了，有一些解决方法对于Unbuntu18.04不适用，需要上网查才可以解决，我和好多同学都遇到这个问题，开始还以为自己电脑坏了。

当然通过本次实验，还是学到了很多的，上机操作真的有助于提升自己。