四 川 大 学 计 算 机 学 院、软 件 学 院

实 验 报 告

学号：2017141051019 姓名：王崇智 专业：计算机科学与技术 班级：173040104 第 9 周

课程名称	计算机网络课程设计	实验课时	2
实验项目	TCP 的连接管理分析 与 UDP 协议分析	实验时间	2019/10/29
实验目的	本实验通过 Wireshark 捕获 TCP 协议和 UDP 协议的分组， 完成对 TCP 的连接管理分析和 UDP 协议工作原理的掌握。在 实验过程中，需要完成对以下知识点的掌握： 1) 了解 TCP 连接的建立和释放过程； 2) 掌握 TCP 报文到达确认(ACK)机制； 3) 了解 UDP 协议的报文段结构；		
实验环境	WINDOWS10 + wireshark		

实验内容（算法、程序、步骤和方法）

1. TCP 连接管理分析

保持联网状态与 wireshark 开启状态，在浏览器中地址栏中输入 www.scu.edu.cn,在浏览器成功显示页面之后停止分组捕获。由于未过滤时，wireshark 同时捕获了很多其于的数据包，我们在次需要进行必要的过滤工作。输入 tcp，并对在 info 中含有[SYN]的信息，点击右键与相关操作进行 http 追踪后，数据框中的结果得到更新即呈如下效果

tcp.stream eq 9

No.	Time	Source	Destination	Protoc	Length	Info
52	7.182038	240e:d8:a56:…	2001:250:2003…	TCP	86	51199 → 80 [SYN] Seq=0 Win=64800 Len=0 MSS=1350 WS=256 SACK_PERM=1
57	7.451948	2001:250:200…	240e:d8:a56:6…	TCP	86	80 → 51199 [SYN, ACK] Seq=0 Ack=1 Win=14400 Len=0 MSS=1312 SACK_PERM=1 WS=128
58	7.452071	240e:d8:a56:…	2001:250:2003…	TCP	74	51199 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=0
59	7.452536	240e:d8:a56:…	2001:250:2003…	HTTP	1184	GET / HTTP/1.1
64	7.651089	2001:250:200…	240e:d8:a56:6…	TCP	74	80 → 51199 [ACK] Seq=1 Ack=1111 Win=17280 Len=0
65	7.651174	2001:250:200…	240e:d8:a56:6…	HTTP	344	HTTP/1.1 304 Not Modified
66	7.693050	240e:d8:a56:…	2001:250:2003…	TCP	74	51199 → 80 [ACK] Seq=1111 Ack=271 Win=130816 Len=0
68	7.869341	240e:d8:a56:…	2001:250:2003…	HTTP	1117	GET /system/resource/code/datainput.jsp?owner=1420436181&e=1&w=1536&h=864&treeid=1072&r
120	8.118593	2001:250:200…	240e:d8:a56:6…	HTTP	424	HTTP/1.1 200 OK
124	8.158611	240e:d8:a56:…	2001:250:2003…	TCP	74	51199 → 80 [ACK] Seq=2154 Ack=621 Win=130560 Len=0

已知 tcp 连接建立需要“三次握手”才能实现最后的数据传输工作

Protoc	Length	Info
TCP	86	51199 → 80 [SYN] Seq=0 Win=64800 Len=0 MSS=1350 WS=256 SACK_PERM=1
TCP	86	80 → 51199 [SYN, ACK] Seq=0 Ack=1 Win=14400 Len=0 MSS=1312 SACK_PERM=1 WS=128
TCP	74	51199 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=0
HTTP	1184	GET / HTTP/1.1

即位于 http 传输前的三次数据包传输为三次握手，下对其内容进行分析

第一个包，即由客户端发向服务器端的［SYN］包

```
0000  a8 9c ed 8b 3d fe 9c b6  d0 1e c1 2d 86 dd 60 0e   ····=··· ···-··`·
0010  c4 fc 00 20 06 40 24 0e  00 d8 0a 56 6a 54 14 07   ··· ·@$· ···VjT··
0020  e4 93 7b 36 ad 9d 20 01  02 50 20 03 00 00 00 00   ··{6·· · ·P ·····
0030  00 00 00 00 00 80 c7 ff  00 50 a2 f3 a6 28 00 00   ·····••· ·P···(··
0040  00 00 80 02 fd 20 63 1e  00 00 02 04 05 46 01 03   ····· c· ·····F··
0050  03 08 01 01 04 02                                 ••••••
```

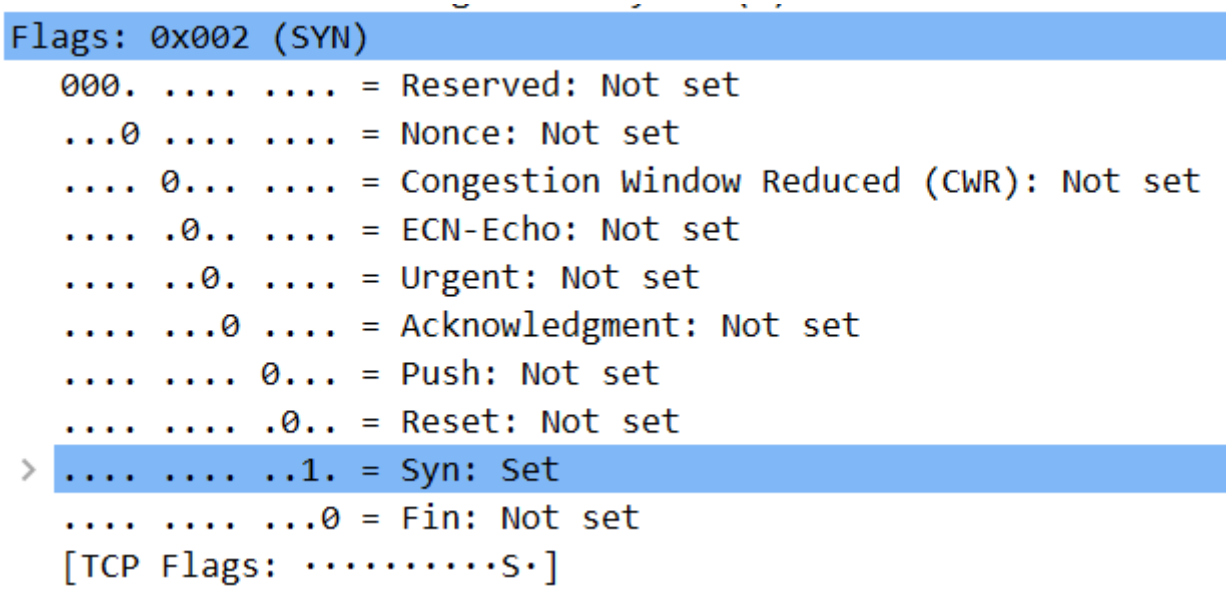
其原始传输数据形式即为上图中的蓝色部分

```
Transmission Control Protocol, Src Port: 51199, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 51199
    Destination Port: 80
    [Stream index: 9]
    [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: 64800
    [Calculated window size: 64800]
    Checksum: 0x631e [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  > Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted
  > [Timestamps]
```

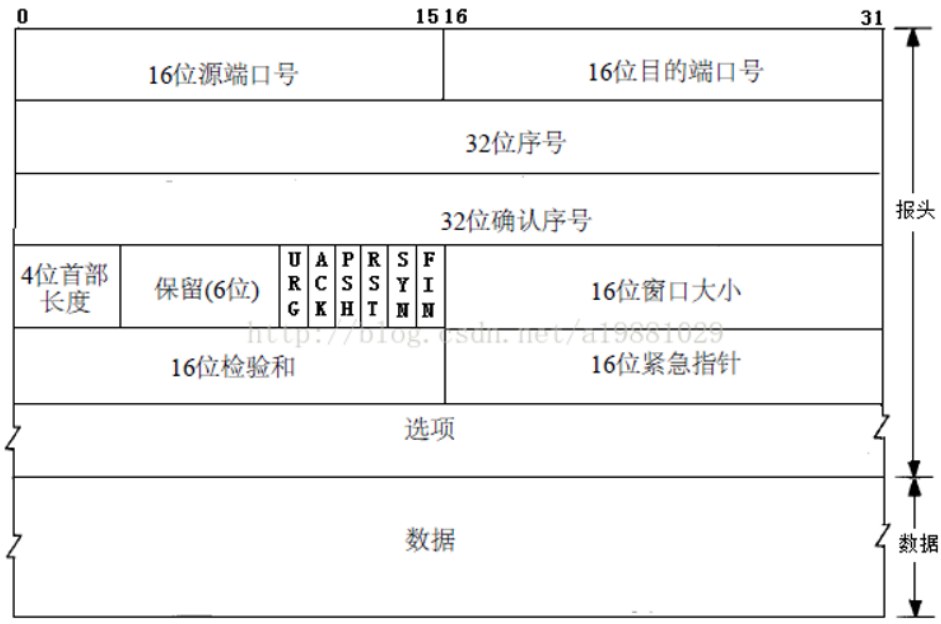
这是原始数据经 wireshark 工具格式化之后的 TCP 部分详细信息

其中序号：即 sequence number 为 0

确认序号：即 Acknowledgement 为 0 报文中有效的标志位为 SYN=1，其于标志位为 0. 这个可以观察该图，即标志位的完整构成



这里做一些关于 TCP 结构的补充



如图所示，正常的 tcp 中，标志位的长度为 12 位，存在 6 位的保留位与 6 位固定的信息，即分别为

U / Urgent：紧急指针有效性标志

A / Acknowledgment：确认序号有效性标志，一旦一个连接建立起来，该标志总被置为 1，即除了请求建立连接报文（仅设置 Syn 标志位为 1），其它所有报文的该标志总为 1

P / Push：Push 标志（接收方应尽快将报文段提交至应用层）

R / Reset：重置连接标志

S / Syn：同步序号标志

F / Fin：传输数据结束标志

则该第一个被发送以请求建立连接的报文中，仅有 syn 标志位为 1

第二个包，即从服务器端发回客户端的包

```
Transmission Control Protocol, Src Port: 80, Dst Port: 51199, Seq: 0, Ack: 1, Len: 0
   Source Port: 80
   Destination Port: 51199
   [Stream index: 9]
   [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]
   Checksum: 0xd25d [unverified]
   [Checksum Status: Unverified]
   Urgent pointer: 0
 > Options: (12 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP), SACK pe
 > [SEQ/ACK analysis]
 > [Timestamps]
```

序号为：0

确认序号为：0+1=1

标志位中 SYN=1， ACK=1,其余　标志位为 0

第三个包，即从客户端回传服务器端的包

```
Transmission Control Protocol, Src Port: 51199, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
   Source Port: 51199
   Destination Port: 80
   [Stream index: 9]
   [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: 512
   [Calculated window size: 131072]
   [Window size scaling factor: 256]
   Checksum: 0x48dc [unverified]
   [Checksum Status: Unverified]
   Urgent pointer: 0
 > [SEQ/ACK analysis]
 > [Timestamps]
```

序号为：1

确认序号为：1

标志位中 SYN=1，其余标志位为 0

四次挥手

因为更换了网络环境，需要重新对刚刚的网站进行访问

```
14217 10.236037    202.115.32.43    10.132.253.229   TCP    60 80 → 56592 [FIN, ACK] Seq=30469 Ack=3865 Win=39424 Len=0
14218 10.236202    10.132.253.229   202.115.32.43    TCP    54 56592 → 80 [ACK] Seq=3865 Ack=30470 Win=130816 Len=0
14346 34.955871    10.132.253.229   202.115.32.43    TCP    54 56592 → 80 [FIN, ACK] Seq=3865 Ack=30470 Win=130816 Len=0
14348 35.019719    202.115.32.43    10.132.253.229   TCP    60 80 → 56592 [ACK] Seq=30470 Ack=3866 Win=39424 Len=0
```

上图即为得到的四次挥手结果，此时本地 ip 已经切换为校园网环境，如图可以看除本地分配的内网地址为 10.132.253.229，而目标网址为 202.115.32.43

本次挥手由服务器端发起

第一个数据包，即代表服务器端说明已经发送完需要的数据，请求客户端回复

```
Transmission Control Protocol, Src Port: 80, Dst Port: 56592, Seq: 30469, Ack: 3865, Len: 0
    Source Port: 80
    Destination Port: 56592
    [Stream index: 101]
    [TCP Segment Len: 0]
    Sequence number: 30469    (relative sequence number)
    [Next sequence number: 30469    (relative sequence number)]
    Acknowledgment number: 3865    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x011 (FIN, ACK)
    Window size value: 77
    [Calculated window size: 39424]
    [Window size scaling factor: 512]
    Checksum: 0x0741 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  > [Timestamps]
```

序号：30469

确认号：3865

标志位 FIN=1，ACK=1，其于标志位为 0

第二个数据包，由客户端向服务器端发送

```
Transmission Control Protocol, Src Port: 56592, Dst Port: 80, Seq: 3865, Ack: 30470, Len: 0
    Source Port: 56592
    Destination Port: 80
    [Stream index: 101]
    [TCP Segment Len: 0]
    Sequence number: 3865    (relative sequence number)
    [Next sequence number: 3865    (relative sequence number)]
    Acknowledgment number: 30470    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
    Window size value: 511
    [Calculated window size: 130816]
    [Window size scaling factor: 256]
    Checksum: 0x058f [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  > [SEQ/ACK analysis]
  > [Timestamps]
```

序号：3865

确认号：30470

标志位 ACK=1，其余标志位为 0

即说明客户端已经收到服务器端的指令，但此时未发出 FIN，即表明其还想继续接收数据，而继续进行等待过程

第三个数据包，仍由客户端向服务器端发送

```
Transmission Control Protocol, Src Port: 56592, Dst Port: 80, Seq: 3865, Ack: 30470, Len: 0
    Source Port: 56592
    Destination Port: 80
    [Stream index: 101]
    [TCP Segment Len: 0]
    Sequence number: 3865    (relative sequence number)
    [Next sequence number: 3865    (relative sequence number)]
    Acknowledgment number: 30470    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x011 (FIN, ACK)
    Window size value: 511
    [Calculated window size: 130816]
    [Window size scaling factor: 256]
    Checksum: 0x058e [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  > [Timestamps]
```

序号：3865

确认号 30470

标志位 ACK=1,FIN=1 其余标志位为 0

此时客户端向服务器端发送了携带 FIN=1 的 ACK 数据包，即代表其已经完成了所有的接收动作并向服务器端说明其将要关闭

第四个数据包，由服务器端向客户端发送

```
Transmission Control Protocol, Src Port: 80, Dst Port: 56592, Seq: 30470, Ack: 3866, Len: 0
    Source Port: 80
    Destination Port: 56592
    [Stream index: 101]
    [TCP Segment Len: 0]
    Sequence number: 30470    (relative sequence number)
    [Next sequence number: 30470    (relative sequence number)]
    Acknowledgment number: 3866    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
    Window size value: 77
    [Calculated window size: 39424]
    [Window size scaling factor: 512]
    Checksum: 0x0740 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  > [SEQ/ACK analysis]
  > [Timestamps]
```

序号：30470

确认号：3866 = 3865+1

标志位 ACK=1，其余标志位为 0

此时服务器端向客户端发送 ACK 表示其成功接收客户端发来的数据，整个挥手过程结束，因为在挥手的第一次操作已经客户端已经成功接收了服务器端发送的数据，已经证明了其发信能力没有问题，所以第四次包的接收问题就不用再考虑了。

但在之前的时候会发现四次挥手出现异常，某一方会向另一方发送仅携带标志位 RST 的数据包，

i. 查阅资料后发现 rst 表示复位，用来异常的关闭连接。而发送 RST 包关闭连接时，不必等待缓冲区的包都发出去（这里不像 FIN 包），直接丢弃缓存区的包，发送 RST 包。而接收端收到 RST 包后，也不必发送 ACK 包来进行确认。

ii. 经常出现的两种情况如下：

1) a 向 b 发起连接，但 b 上未监听相应的端口，这时 b 上的 tcp 处理程序会发 rst 包

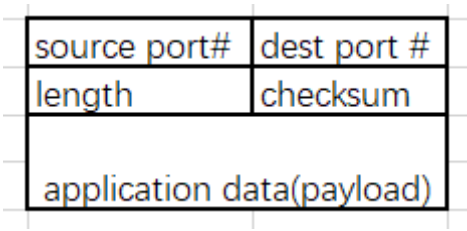
2) ab 已经建立正常连接，在通讯时，a 向 b 发送 fin 包要求关闭连接，b 在发出 ack 后，网断开了，即 a 并未实际收到。a 通过若干原因放弃了这个连接，网通后，B 又开始发送数据包，A 因为逻辑上已经断开与 B 的连接，此时不知道该数据包的来源，就发 RST 数据包再把该连接给关闭掉

2. UDP 协议分析

因为实验过程中切换了网络环境，且还不知道清除手机热点 DNS 高速缓存的方式，尝试访问其他网站，cn.bing.com 为例

1817	107.207325	192.168.43.2…	192.168.43.1	DNS	71	Standard query 0xa5a8 A cn.bing.com
1819	107.249185	192.168.43.1	192.168.43.211	DNS	200	Standard query response 0xa5a8 A cn.bing.com CNAME cn-bing-com.cn.a-0001.a-msedge.net

在对 UDP 分析之前需要补充对 UDP 结构的了解



上图即为基本的 udp 结构

其长度位 32bits,头部占 4 个字的空间

其中 source port 为源端口号，长度 16bits

Dest port 为目的端口号，长度 16bits

length 为数据报长度，长度 16bits，在仅有首部的条件下，其最小值为 8，注意 length 描述的长度是以 bytes 即字节为单位的。而回顾知识，一个字节即为 8 位

checksum 为校验和，长度为 16bits,用来检测 udp 用户数据报在传输中是否出错

而计算 checksum 的步骤有点复杂，需要在原 udp 报文中添加 12 个子节的伪首部进行操作

在伪首部添加到 UDP 上之后
在计算初始化时将检验和字段添零（即不计算该表中接收到的 checksum）
把所有位划分为长度为两个字节的字
把所有的 16 位的字相加，如果遇到进位则将高于 16 字节的进位部分的值加到最低位上
所有字相加得到的结果应该位一个 16 位的数，将该数取反即得到检验和 checksum
具体来进行实验测试，编写一段 java 代码发送一段 udp 数据包

```java
package clienttest;

import java.io.IOException;
import java.net.*;

public class client {

    private DatagramSocket socket;

    public client() throws SocketException {
        socket = new DatagramSocket();
```

```
    }

    public void run() throws IOException {
        InetAddress ip = InetAddress.getByName("11.112.112.112");
        String message = "my first UDP";
        byte[] buffer = message.getBytes();
        DatagramPacket sendPacket = new DatagramPacket(buffer, buffer.length, ip,
12333);
        socket.send(sendPacket);

    }

    public static void main(String[] args) throws IOException {
        new client().run();
    }

}
```

发送信息为"my first UDP",发送目标 ip 地址为 11.112.112.112，目标端口为 12333

```
User Datagram Protocol, Src Port: 57041, Dst Port: 12333
   Source Port: 57041
   Destination Port: 12333
   Length: 20
   Checksum: 0x983b [unverified]
   [Checksum Status: Unverified]
   [Stream index: 44]
 > [Timestamps]
Data (12 bytes)
   Data: 6d7920666972737420554450
```

```
000  58 69 6c 4c 47 53 9c b6  d0 1e c1 2d 08 00 45 b8   XilLGS·· ···-··E·
010  00 28 03 f5 00 00 80 11  ac f8 0a 84 02 bc 0b 70   ·(······ ·······p
020  70 70 de d1 30 2d 00 14  98 3b 6d 79 20 66 69 72   pp··0-·· ·;my fir
030  73 74 20 55 44 50                                 st UDP
```

该图即为抓到的数结果，下方的数据原始信息直接能看到发送到的 msg 为 myfirst UDP

结合上面所说的伪首部数据，且注意部分信息需要在 IP 协议中寻找

```
Protocol: UDP (17)
Header checksum: 0xacf8 [validation disabled]
[Header checksum status: Unverified]
Source: 10.132.2.188
Destination: 11.112.112.112
```

得到具体的 16 进制数据表

Source	0a84 02bc
Destination	0b70 7070
Protocol	11
Length	14
Source port	Ded1
Destination port	30 2d
Length	14
Data	6d 79 20 66 69 72 73 74 20 55 40 50

将以上 16 进制数据按照 16 位加法的方式进位相加，其结果与原 checksum 进行比较即可完成校验工作

注意，UDP 头部总共 8 个字节，但通常说 UDP 数据包的大小时，整体是包含了头部空间的

第一个发送包的 udp 结构如下

```
User Datagram Protocol, Src Port: 55449, Dst Port: 53
   Source Port: 55449
   Destination Port: 53
   Length: 37
   Checksum: 0xf1fd [unverified]
   [Checksum Status: Unverified]
   [Stream index: 6]
 ⌄ [Timestamps]
      [Time since first frame: 0.000000000 seconds]
      [Time since previous frame: 0.000000000 seconds]
```

其 checksum 为 0xf1fd，处于未检验状态

该 UDP 的协议号，在 IP 层中查看为 UDP（17）；同时注意到 ip 层中也有一个 header checksum，在之后的学习中进行进一步了解 ip 的检验方式

```
Internet Protocol Version 4, Src: 192.168.43.211, Dst: 192.168.43.1
   0100 .... = Version: 4
   .... 0101 = Header Length: 20 bytes (5)
 › Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
   Total Length: 57
   Identification: 0x1da9 (7593)
 › Flags: 0x0000
   Time to live: 128
   Protocol: UDP (17)
   Header checksum: 0x44e6 [validation disabled]
   [Header checksum status: Unverified]
   Source: 192.168.43.211
   Destination: 192.168.43.1
```

接收的 UDP 包结构如下

```
User Datagram Protocol, Src Port: 53, Dst Port: 55449
   Source Port: 53
   Destination Port: 55449
   Length: 166
   Checksum: 0xfce1 [unverified]
   [Checksum Status: Unverified]
   [Stream index: 6]
 ⌄ [Timestamps]
      [Time since first frame: 0.041860000 seconds]
      [Time since previous frame: 0.041860000 seconds]
```

分组检验的和位 0xfce1 处于未验证状态

其 IP 结构如下

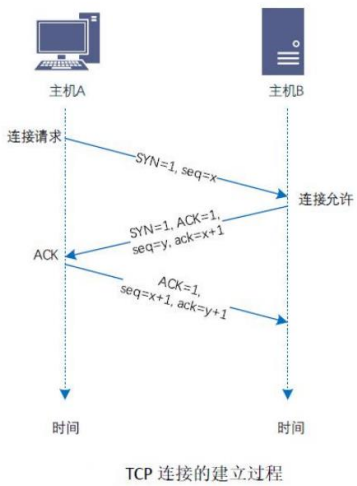
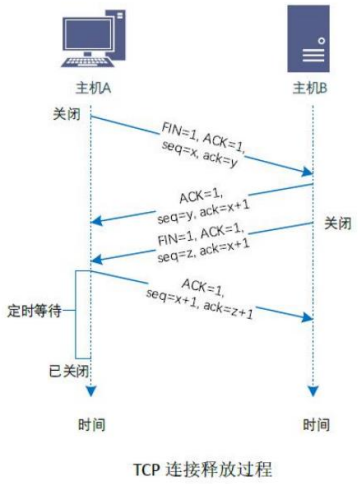
```
Internet Protocol Version 4, Src: 192.168.43.1, Dst: 192.168.43.211
   0100 .... = Version: 4
   .... 0101 = Header Length: 20 bytes (5)
 > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
   Total Length: 186
   Identification: 0x67f5 (26613)
 > Flags: 0x4000, Don't fragment
   Time to live: 64
   Protocol: UDP (17)
   Header checksum: 0xfa18 [validation disabled]
   [Header checksum status: Unverified]
   Source: 192.168.43.1
   Destination: 192.168.43.211
```

其 protocol 为 udp（17）与发送包的编号一致

5）问：udp 协议首部检验和字段是否冗余？请阐述原因

答：认为并不冗余，本身 tcp 为了加快网络传输，已经相比 ip 与 tcp 报文结构减少了很多内容。但为了保证基本的效率，加上检验机制可以保证其发现错误，以保证速度的同时仍具备一定的效率。

而且要注意到，链路层虽然也存在数据的校验机制，但是该机制只能保证数据从一个路由器到另一个路由器中间没有出错。

	但是数据包到了路由器之后，路由器会把这个数据包拆开，根据下一跳的地址，设置新的链路层头部的目的地址，crc 校验值，ip 首部的 ttl 值等，甚至可能会对数据包进行分片，这样就对原数据进行了较多修改，如果在路由器处理的过程中数据出错，那么链路层的校验时无法发现该错误的。 所以要保证数据包凑够发送方到目的地址均不出现错误，需要传输层自己的有效校验，而 checksum 就是 udp 实现该过程的必要条件。
数据记录和计算	实验原理  TCP 连接的建立过程  TCP 连接释放过程 UDP 提供无连接通信，且不对传送数据包进行可靠性保证，适合于一次传输少量数据，UDP 传输的可靠性由应用层负责。常用的 UDP 端口号有：DNS 53 等；

结论（结果）	1. 通过对建立连接时报文的抓取，成功分析了三次握手时数据包中标志位的变化 2. 通过对断开连接时报文的抓取，成功分析了四次挥手时数据包中标志位的变化 3. 通过对 dns 请求时 udp 报文的捕获与分析，成功认识了 udp 协议的结构，并了解到了伪首部的重要作用与 checksum 的计算方式
小结	通过本次实验。也是离开应用层后的第一次传输层实验。实验中配合 wireshark 成功捕获了步骤中需要的 tcp 与 udp 等报文。通过工具的转化成功看到了与课本上一致的结构，也通过实际的演算对实际传输中的 checksum 做了一定的验证工作。尤其是对标志位在握手挥手过程中的变化印象深刻，且过程中还掉入过一次坑内。就是发现服务器向本地客户端传输文件的时候标志位 ack 不变，但是其他内容都在变化。实际上那个时候是连接建立完毕，处于传输数据的状态，因为数据包实际过程中很可能大于协议约束的上限所以传输过程中必须进行一定的分包。且应明确，传输过程中不存在重复的 ACK，因为在握手阶段，确认的序号会将发送方传过来的序号增 1 以回答。在数据传输阶段，确认序号将发送方的序号加上发送的数据大小作为整体而回答以表示确实收到数据。 绝知此事要躬行，有的时候按照博客与书本的参考没法与实际验证的结果一致，于是就需要继续实验或者查阅更多的资料来证实某种观点，这个意识很重要，本次实验中继续锻炼了自己 debug 的能力并进一步加深了自己对计算机网络及相关技术的了解，同学之间的互相帮助也极大推动了本次实验的顺利完成。
指导老师评议	成绩评定：　　指导教师签名：