

计算机网络实验报告（二）

专业：计算机科学与技术

学号：2011188

姓名：邵琦

- 一、实验要求
- 二、Web服务器的搭建
- 三、报文
 - 3.1 TCP报文
 - 3.1.1 物理层数据帧概况
 - 3.1.2 数据层头部信息
 - 3.1.3 互联网层IP包头部信息
 - 3.1.4 传输层数据段头部信息
 - 3.2 HTTP报文
 - 3.2.1 HTTP请求报文
 - 3.2.2 HTTP响应报文
- 四、wireshark捕获
 - 4.1 三次握手
 - 4.2 http请求
 - 4.3 四次挥手

一、实验要求

(1) 搭建Web服务器（自由选择系统），并制作简单的Web页面，包含简单文本信息（至少包含专业、学号、姓名）和自己的LOGO。

(2) 通过浏览器获取自己编写的Web页面，使用Wireshark捕获浏览器与Web服务器的交互过程，并进行简单的分析说明。

(3) 提交实验报告。

二、Web服务器的搭建

html文件如下:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>专业: 计算机科学与技术</h1>
  <h1>姓名: 邵琦</h1>
  <h1>学号: 2011188</h1>
  
</body>
</html>
```

js文件如下:

```

//引入http模块
var http=require('http');
var fs=require('fs');
var url=require('url');
//创建服务
var server=http.createServer(function(req,res){
    var url=req.url;
    if(url==='/{
        fs.readFile('lab2.html',(err,data)=>{
            if(err){
                res.setHeader('Content-Type','text/plain;charset=utf-8');
                res.end('文件读取失败');
            }else{
                res.setHeader('Content-Type','text/html;charset=utf-8');
                res.end(data);
            }
        });
    }else {
        fs.readFile('SHOXIE.jpg',(err,data)=>{
            if(err){
                res.setHeader('Content-Type','text/plain;charset=utf-8');
                res.end('文件读取失败');
            }else{
                res.setHeader('Content-Type','image/jpeg');
                res.end(data);
            }
        });
    }
}).listen(8001);
console.log('Server running at http://127.0.0.1:8001/');

```

用html文件与js文件搭建服务器，在集成终端运行后，在浏览器中输入"<http://127.0.0.1:8001/>"即可进入该网站，该网站包括了姓名、学号、专业，以及自己设计的一张图片，其截图如下：



127.0.0.1:8001

专业： 计算机科学与技术

姓名： 邵琦

学号： 2011188



SHOXDE

三、 报文

3.1 TCP报文

如图所示，TCP报文分为四部分：

> Frame 8: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NP	0000	02 00 00 00 45 00 00 34	e5 36 40 00 80 06 00 00E..4.6@....
> Null/Loopback	0010	7f 00 00 01 7f 00 00 01	cc 24 1f 41 a9 e1 3d 28\$.A..=(
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	0020	00 00 00 00 80 02 ff ff	a4 7a 00 00 02 04 ff d7-z.....
> Transmission Control Protocol, Src Port: 52260, Dst Port: 8001, Seq: 0, Len: 0	0030	01 03 03 08 01 01 04 02	

3.1.1 物理层数据帧概况

▼ Frame 8: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NP
Section number: 1
> Interface id: 0 (\Device\NPF_Loopback)
Encapsulation type: NULL/Loopback (15)
Arrival Time: Oct 28, 2022 14:30:03.138771000 中国标准时间
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1666938603.138771000 seconds
[Time delta from previous captured frame: 0.001641000 seconds]
[Time delta from previous displayed frame: 0.000000000 seconds]
[Time since reference or first frame: 2.343098000 seconds]
Frame Number: 8
Frame Length: 56 bytes (448 bits)
Capture Length: 56 bytes (448 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: null:ip:tcp]
[Coloring Rule Name: TCP SYN/FIN]
[Coloring Rule String: tcp.flags & 0x02 tcp.flags.fin == 1]
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 52260, Dst Port: 8001, Seq: 0, Len: 0

3.1.2 数据层头部信息

由于是本地回环，所以头部并没有什么信息。

> Frame 8: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NP
▼ Null/Loopback
Family: IP (2)
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 52260, Dst Port: 8001, Seq: 0, Len: 0

3.1.3 互联网层IP包头部信息

```

> Frame 8: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF
> Null/Loopback
√ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 52
    Identification: 0xe536 (58678)
    > 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: TCP (6)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 127.0.0.1
    Destination Address: 127.0.0.1
> Transmission Control Protocol, Src Port: 52260, Dst Port: 8001, Seq: 0, Len: 0

```

这部分信息包含：指明为IPv4协议，包头长度为20bytes，IP包总长度为52，标识字段58678，标记字段为0x2，生存周期为128，包内封装的上层协议为TCP，头部校验和，源IP地址（127.0.0.1），目的IP地址（127.0.0.1）。

3.1.4 传输层数据段头部信息

```

> Frame 8: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
√ Transmission Control Protocol, Src Port: 52260, Dst Port: 8001, Seq: 0, Len: 0
    Source Port: 52260
    Destination Port: 8001
    [Stream index: 3]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0 (relative sequence number)
    Sequence Number (raw): 2850110760
    [Next Sequence Number: 1 (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
    > Flags: 0x002 (SYN)
    Window: 65535
    [Calculated window size: 65535]
    Checksum: 0xa47a [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
    > Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation
    > [Timestamps]

```

这部分信息包含：源端口，目的端口，标志为SYN（发送SYN报文到服务器），seq=0（此处分析的是第一次握手）

3.2 HTTP报文

3.2.1 HTTP请求报文

> Frame 11: 774 bytes on wire (6192 bits), 774 bytes captured (6192 bits) on interface \Device\NPF{...}	0020 56 fc 15 09 50 18 04 ff 41 ad 00 00 47 45 54 20 V...P... A...GET
> Null/Loopback	0030 2f 20 48 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74 / HTTP/1.1..Host
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	0040 3a 20 31 32 37 2e 30 2e 30 2e 31 3a 38 30 30 31 : 127.0. 0.1:8001
> Transmission Control Protocol, Src Port: 52260, Dst Port: 8001, Seq: 1, Ack: 1, Len: 730	0050 0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 6b 65 ..Connec tion: ke
> Hypertext Transfer Protocol	0060 65 70 2d 61 6c 69 76 65 0d 0a 73 65 63 2d 63 68 ep-alive ..sec-ch
> GET / HTTP/1.1\r\n	0070 2d 75 61 3a 20 22 43 68 72 6f 6d 69 75 6d 22 3b -ua: "Ch romium";
> [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]	0080 76 3d 22 31 30 36 22 2c 20 22 4d 69 63 72 6f 73 v="106", "Micros
> Request Method: GET	0090 6f 66 74 20 45 64 67 65 22 3b 76 3d 22 31 30 36 oft Edge ";v="106
> Request URI: /	00a0 22 2c 20 22 4e 6f 74 3b 41 3d 42 72 61 6e 64 22 ", "Not; A=Brand"
> Request Version: HTTP/1.1	00b0 3b 76 3d 22 39 39 22 0d 0a 73 65 63 2d 63 68 2d ;v="99" ..sec-ch
> Host: 127.0.0.1:8001\r\n	00c0 75 61 2d 6d 6f 62 69 6c 65 3a 20 3f 30 0d 0a 73 ua-mobil e: ?0..s
> Connection: keep-alive\r\n	00d0 65 63 2d 63 68 2d 75 61 2d 70 6c 61 74 66 6f 72 ec-ch-ua -platfor
> sec-ch-ua: "Chromium";v="106", "Microsoft Edge";v="106", "Not;A=Brand";v="99"\r\n	00e0 6d 3a 20 22 57 69 6e 64 6f 77 73 22 0d 0a 55 70 m: "Wind ows"..Up
> sec-ch-ua-mobile: ?0\r\n	00f0 67 72 61 64 65 2d 49 6e 73 65 63 75 72 65 2d 52 grade-In secure-R
> sec-ch-ua-platform: "Windows"\r\n	0100 65 71 75 65 73 74 73 3a 20 31 0d 0a 55 73 65 72 equests: 1..User
> Upgrade-Insecure-Requests: 1\r\n	0110 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 6c 6c 61 2f -Agent: Mozilla/
> User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36 Edg/106.0.1370.52\r\n	0120 35 2e 30 20 28 57 69 6e 64 6f 77 73 20 4e 54 20 5.0 (Win dows NT
> Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;purpose=fetch\r\n	0130 31 30 2e 30 3b 20 57 69 6e 36 34 3b 20 78 36 34 10.0; Wi n64; x64
> Sec-Fetch-Site: none\r\n	0140 29 20 41 70 70 6c 65 57 65 62 4b 69 74 2f 35 33) AppleW ebKit/53
> Sec-Fetch-Mode: navigate\r\n	0150 37 2e 33 36 20 28 4b 48 54 4d 4c 2c 20 6c 69 6b 7.36 (KH TML, lik
> Sec-Fetch-User: ?1\r\n	0160 65 20 47 65 63 6b 6f 29 20 43 68 72 6f 6d 65 2f e Gecko) Chrome/
> Sec-Fetch-Dest: document\r\n	0170 31 30 36 2e 30 2e 30 2e 30 20 53 61 66 61 72 69 106.0.0. 0 Safari
> Accept-Encoding: gzip, deflate, br\r\n	0180 2f 35 33 37 2e 33 36 20 45 64 67 2f 31 30 36 2e /537.36 Edg/106.
> Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6,zh-TW;q=0.5\r\n	0190 30 2e 31 33 37 30 2e 35 32 0d 0a 41 63 63 65 70 0.1370.5 2..Accep
> [Full request URI: http://127.0.0.1:8001/]	01a0 74 3a 20 74 65 78 74 2f 68 74 6d 6c 2c 61 70 70 t: text/ html,app
> [HTTP request 1/2]	01b0 6c 69 63 61 74 69 6f 6e 2f 78 68 74 6d 6c 2b 78 lication /xhtml+x
> [Response in frame: 18]	01c0 6d 6c 2c 61 70 70 6c 69 63 61 74 69 6f 6e 2f 78 ml,appli cation/x
> [Next request in frame: 20]	01d0 6d 6c 3b 71 3d 30 2e 39 2c 69 6d 61 67 65 2f 77 ml;q=0.9 ,image/w
	01e0 65 62 70 2c 69 6d 61 67 65 2f 61 70 6e 67 2c 2a ebp,imag e/apng.*
	01f0 2f 2a 3b 71 3d 30 2e 38 2c 61 70 70 6c 69 63 61 /*;q=0.8 ,applica
	0200 74 69 6f 6e 2f 73 69 67 6e 65 64 2d 65 78 63 68 tion/sig ned-exch
	0210 61 6e 67 65 3b 76 3d 62 33 3b 71 3d 30 2e 39 0d ange;v=b 3;q=0.9..

这部分信息在原有的TCP基础上增加了差文本传输协议部分。

请求行指明方法为GET，HTTP版本为HTTP1.1，指明请求源host为127.0.0.1:8001，连接类型，一些Cookie信息。

3.2.2 HTTP响应报文

> Frame 18: 619 bytes on wire (4952 bits), 619 bytes captured (4952 bits) on interface \Device\NPF{...}	0000 02 00 00 00 45 00 02 67 e5 40 40 00 80 06 00 00E..g @@.....
> Null/Loopback	0010 7f 00 00 01 7f 00 00 01 1f 41 cc 24 56 fc 15 09 :A:\$V....
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	0020 a9 e1 40 03 50 18 27 f9 da 3c 00 00 48 54 54 50 ..@.P...'<..HTTP
> Transmission Control Protocol, Src Port: 8001, Dst Port: 52260, Seq: 1, Ack: 731, Len: 575	0030 2f 31 2e 31 20 32 30 30 20 4f 4b 0d 0a 43 6f 6e /1.1 200 OK..Con
> Hypertext Transfer Protocol	0040 74 65 6e 74 2d 54 79 70 65 3a 20 74 65 78 74 2f tent-Typ e: text/
> Line-based text data: text/html (15 lines)	0050 68 74 6d 6c 3b 63 68 61 72 73 65 74 3d 75 74 66 html;cha rset=utf
> <!DOCTYPE html>\r\n	0060 2d 38 0d 0a 44 61 74 65 3a 20 46 72 69 2c 20 32 -8..Date : Fri, 2
> <html lang="en">\r\n	0070 38 20 4f 63 74 20 32 30 32 32 20 30 36 3a 33 30 8 Oct 20 22 06:30
> <head>\r\n	0080 3a 30 33 20 47 4d 54 0d 0a 43 6f 6e 6e 65 63 74 :03 GMT..Connect
> <meta charset="UTF-8">\r\n	0090 69 6f 6e 3a 20 6b 65 65 70 2d 61 6c 69 76 65 0d ion: kee p-alive..
> <meta http-equiv="X-UA-Compatible" content="IE=edge">\r\n	00a0 0a 4b 65 65 70 2d 41 6c 69 76 65 3a 20 74 69 6d .Keep-Al ive: tim
> <meta name="viewport" content="width=device-width, initial-scale=1.0">\r\n	00b0 65 6f 75 74 3d 35 0d 0a 43 6f 6e 74 65 6e 74 2d eout=5.. Content-
> <title>Document</title>\r\n	00c0 4c 65 6e 67 74 68 3a 20 34 31 32 0d 0a 0d 0a 3c Length: 412....<
> </head>\r\n	00d0 21 44 4f 43 54 59 50 45 20 68 74 6d 6c 3e 0d 0a !DOCTYPE html>..
> <body>\r\n	00e0 3c 68 74 6d 6c 20 6c 61 6e 67 3d 22 65 6e 22 3e <html la ngs="en">
> <h1>专业：计算机科学与技术</h1>\r\n	00f0 0d 0a 3c 68 65 61 64 3e 0d 0a 20 20 20 3c 6d ..<head> .. <m
> <h1>姓名：邵琦</h1>\r\n	0100 65 74 61 20 63 68 61 72 73 65 74 3d 22 55 54 46 eta char set="UTF
> <h1>学号：2011188</h1>\r\n	0110 2d 38 22 3e 0d 0a 20 20 20 20 3c 6d 65 74 61 20 -8">.. <meta
> \r\n	0120 68 74 74 70 2d 65 71 75 69 76 3d 22 58 2d 55 41 http-equ iv="X-UA
> </body>\r\n	0130 2d 43 6f 6d 70 61 74 69 62 6c 65 22 20 63 6f 6e -Compati ble" con
> </html>	0140 74 65 6e 74 3d 22 49 45 3d 65 64 67 65 22 3e 0d tent="IE =edge">..
	0150 0a 20 20 20 20 3c 6d 65 74 61 20 6e 61 6d 65 3d <me ta name=
	0160 22 76 69 65 77 70 6f 72 74 22 20 63 6f 6e 74 65 "viewpor t" conte
	0170 6e 74 3d 22 77 69 64 74 68 3d 64 65 76 69 63 65 nt="wid t=device

响应报文的消息格式又在HTTP请求报文中增加了响应体，包含请求需要得到的数据，这里是HTML文件内容。

四、wireshark捕获

在实验中首先开启Wireshark，并开启抓包，在浏览器中输入自己设计的网页的网址，观察Wireshark界面如下图所示：



由于客户端和服务端都是本机，所以我们选择“Adapter for loopback traffic capture”进入。进入后我们使用过滤器，取出本机端口与本机交互的数据包显示，以分析浏览器与Web服务器的交互过程。

8	2.343098	127.0.0.1	127.0.0.1	TCP	56 52260 → 8001 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
9	2.343167	127.0.0.1	127.0.0.1	TCP	56 8001 → 52260 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
10	2.343214	127.0.0.1	127.0.0.1	TCP	44 52260 → 8001 [ACK] Seq=1 Ack=1 Win=327424 Len=0
11	2.343484	127.0.0.1	127.0.0.1	HTTP	774 GET / HTTP/1.1
12	2.343527	127.0.0.1	127.0.0.1	TCP	44 8001 → 52260 [ACK] Seq=1 Ack=731 Win=2619648 Len=0
18	2.360828	127.0.0.1	127.0.0.1	HTTP	619 HTTP/1.1 200 OK (text/html)
19	2.360878	127.0.0.1	127.0.0.1	TCP	44 52260 → 8001 [ACK] Seq=731 Ack=576 Win=326656 Len=0
20	2.400642	127.0.0.1	127.0.0.1	HTTP	694 GET /SHOXIE.jpg HTTP/1.1
21	2.400706	127.0.0.1	127.0.0.1	TCP	44 8001 → 52260 [ACK] Seq=576 Ack=1381 Win=2619136 Len=0
22	2.407812	127.0.0.1	127.0.0.1	HTTP	17227 HTTP/1.1 200 OK (JPEG JFIF image)
23	2.407899	127.0.0.1	127.0.0.1	TCP	44 52260 → 8001 [ACK] Seq=1381 Ack=17759 Win=309504 Len=0
35	7.414801	127.0.0.1	127.0.0.1	TCP	44 8001 → 52260 [FIN, ACK] Seq=17759 Ack=1381 Win=2619136 Len=0
36	7.414883	127.0.0.1	127.0.0.1	TCP	44 52260 → 8001 [ACK] Seq=1381 Ack=17760 Win=309504 Len=0
68	21.956645	127.0.0.1	127.0.0.1	TCP	44 52260 → 8001 [FIN, ACK] Seq=1381 Ack=17760 Win=309504 Len=0
69	21.956731	127.0.0.1	127.0.0.1	TCP	44 8001 → 52260 [ACK] Seq=17760 Ack=1382 Win=2619136 Len=0

TCP协议提供的是按序、可靠的服务，是一种面向连接的传输方式，即其发送数据之前发送方和接收放需要握手，断开链接时需要四次挥手。

4.1 三次握手

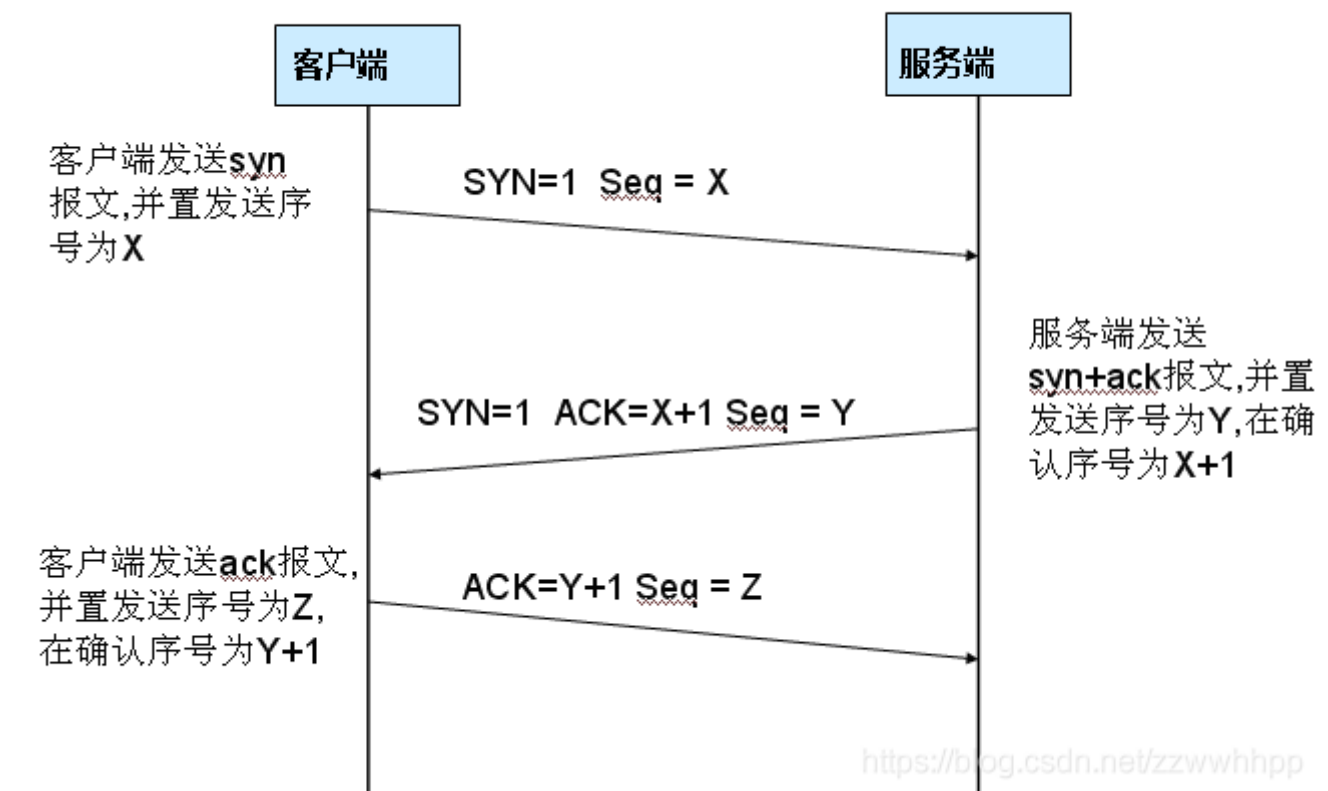
首先，TCP进行三次握手：

第一次握手：客户端发送syn包(seq=x)到服务器，并进入SYN_SEND状态，等待服务器确认；

第二次握手：服务器收到syn包，必须确认客户的SYN(ack=x+1)，同时自己也发送一个SYN包(seq=y)，即SYN+ACK包，此时服务器进入SYN_RECV状态；

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(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。

TCP 三次握手



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。

8	2.343098	127.0.0.1	127.0.0.1	TCP	56	52260 → 8001 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
9	2.343167	127.0.0.1	127.0.0.1	TCP	56	8001 → 52260 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
10	2.343214	127.0.0.1	127.0.0.1	TCP	44	52260 → 8001 [ACK] Seq=1 Ack=1 Win=327424 Len=0

如上图所示，即为tcp的三次握手过程：

首先由52260向8001端口发送了一条syn包，并令seq = x = 0（本机初始序列号），报文长度为0，滑动窗口为65535，最大窗口长度为65495，窗口扩大因子为256，此时浏览进入了SYN_SEND状态，这是第一次握手；

然后服务器8001端口接收到了浏览器发送的syn包后，确认客户的syn，使ack = x + 1 = 1，seq = y = 0，对序列号为1（ack）之前的报文进行确认，同时向客户发送一个（syn，ack）包，其中报文长度为0，滑动窗口为65535，最大窗口长度为65495，这是第二次握手；

客户端接受到了服务器发送的（syn，ack）包后，客户端向服务器返回确认包，其中ack = y + 1 = 1，滑动窗口为327424，报文长度为0，这是第三次握手。至此三次握手完成，服务器和客户端建立了联系。

4.2 http请求

通过TCP三次握手后，建立了连接。开始进行数据交互。

11	2.343484	127.0.0.1	127.0.0.1	HTTP	774	GET / HTTP/1.1
12	2.343527	127.0.0.1	127.0.0.1	TCP	44	8001 → 52260 [ACK] Seq=1 Ack=731 Win=2619648 Len=0
18	2.360828	127.0.0.1	127.0.0.1	HTTP	619	HTTP/1.1 200 OK (text/html)
19	2.360878	127.0.0.1	127.0.0.1	TCP	44	52260 → 8001 [ACK] Seq=731 Ack=576 Win=326656 Len=0
20	2.400642	127.0.0.1	127.0.0.1	HTTP	694	GET /SHOXIE.jpg HTTP/1.1
21	2.400706	127.0.0.1	127.0.0.1	TCP	44	8001 → 52260 [ACK] Seq=576 Ack=1381 Win=2619136 Len=0
22	2.407812	127.0.0.1	127.0.0.1	HTTP	17227	HTTP/1.1 200 OK (JPEG JFIF image)
23	2.407899	127.0.0.1	127.0.0.1	TCP	44	52260 → 8001 [ACK] Seq=1381 Ack=17759 Win=309504 Len=0

首先可以看出，使用http1.1协议首先发送一个访问页面的get请求，客户端收到get请求后将所请求内容发送给客户端。

4.3 四次挥手

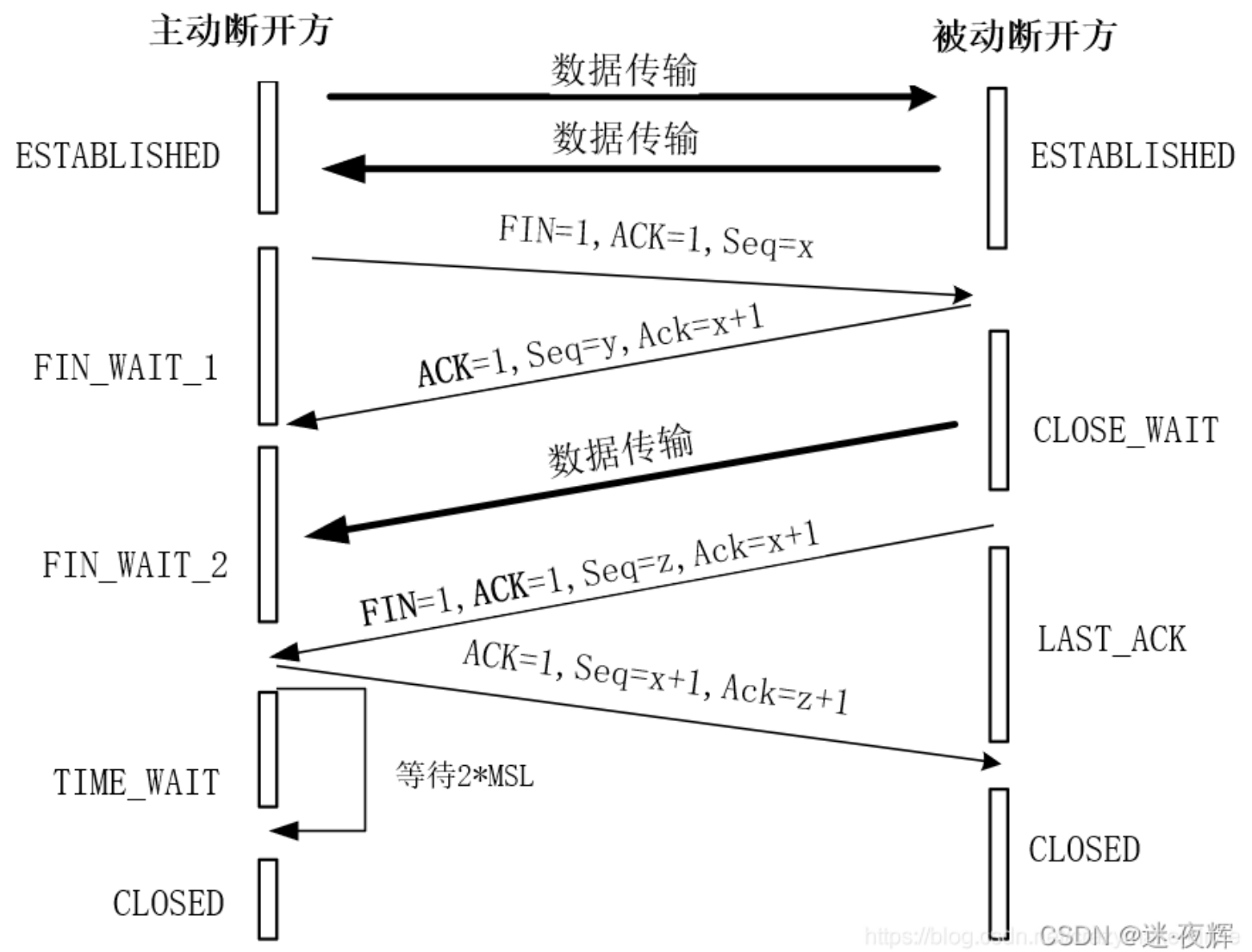
最后，TCP进行四次挥手：

第一次挥手：主动断开方发送一个FIN，用来关闭数据传送，主动断开方进入FIN_WAIT_1状态。

第二次挥手：被动断开方收到FIN后，发送一个ACK给主动断开方，确认序号为收到序号+1（与SYN相同，一个FIN占用一个序号），服务器进入CLOSE_WAIT状态。

第三次挥手：被动断开方发送一个FIN，用来关闭服务器到主动断开方的数据传送，服务器进入LAST_ACK状态。

第四次挥手：主动断开方收到FIN后，主动断开方进入TIME_WAIT状态，接着发送一个ACK给服务器，确认序号为收到序号+1，服务器进入CLOSED状态，完成四次挥手。



35	7.414801	127.0.0.1	127.0.0.1	TCP	44	8001 → 52260	[FIN, ACK] Seq=17759 Ack=1381 Win=2619136 Len=0
36	7.414883	127.0.0.1	127.0.0.1	TCP	44	52260 → 8001	[ACK] Seq=1381 Ack=17760 Win=309504 Len=0
68	21.956645	127.0.0.1	127.0.0.1	TCP	44	52260 → 8001	[FIN, ACK] Seq=1381 Ack=17760 Win=309504 Len=0
69	21.956731	127.0.0.1	127.0.0.1	TCP	44	8001 → 52260	[ACK] Seq=17760 Ack=1382 Win=2619136 Len=0

如上图所示，即为tcp四次挥手过程。

首先，服务器向客户端发送了一条请求结束的报文（FIN，ACK），报文序列号为seq = x = 17759，ack = y = 1381，报文长度为0，这是第一次挥手；

然后客户端接收到了发送的FIN包后，对收到的报文进行确认，其报文序列号为seq = y = 1381，ack = x + 1 = 17759 + 1 = 17760，这是第二次挥手；

然后客户端再向服务器发送一个FIN包，其内容与前一个基本相同，用来关闭服务器端到客户端的数据传送，这是第三次挥手；

服务器对收到的fin包进行确认，客户端进入关闭状态，这是第四次挥手。至此tcp的四次挥手结束，客户端和服务端断开连接，访问结束。