



南开大学
Nankai University

计算机网络实验2：配置Web服务器



姓名：陈睿颖

学号：2013544

专业：计算机科学与技术

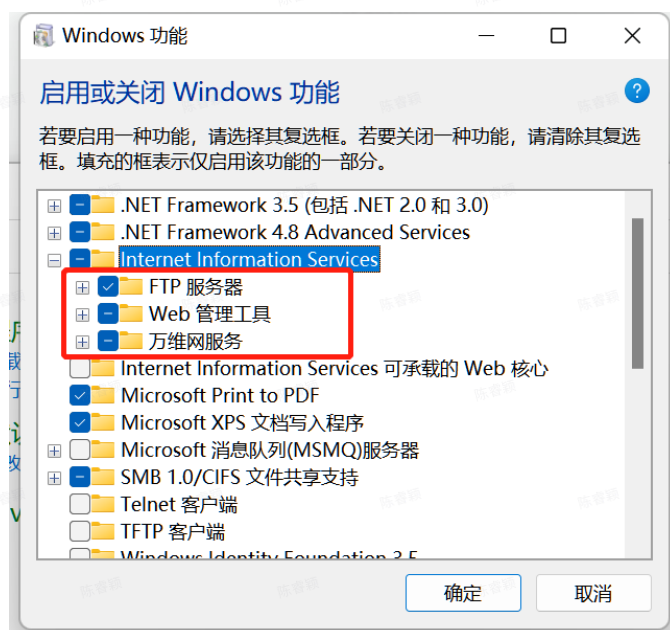
1. 实验内容

- 搭建Web服务器（自由选择系统）；
- 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；
- 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；

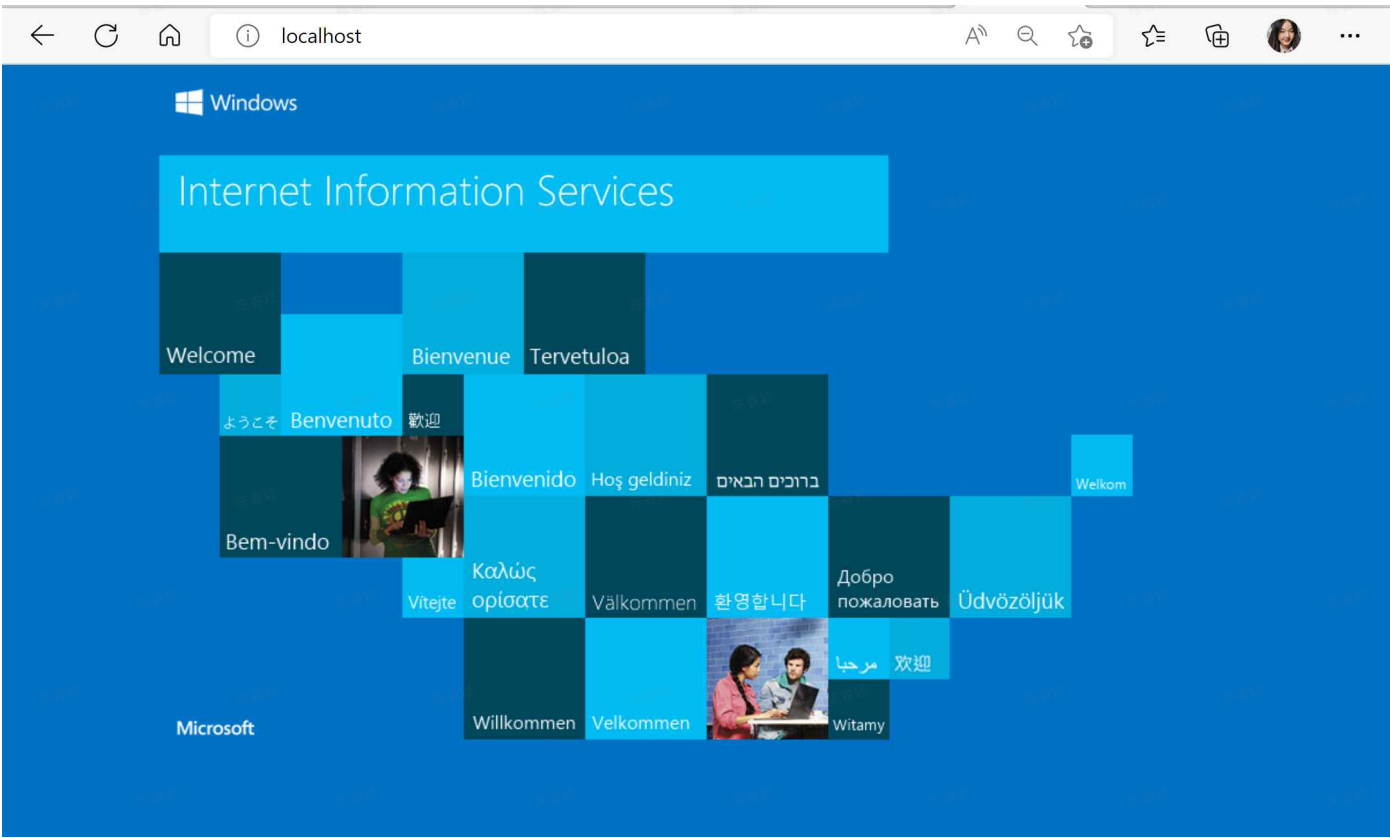
2. 实验步骤

2.1 配置Web服务器

- 首先打开Internet information services

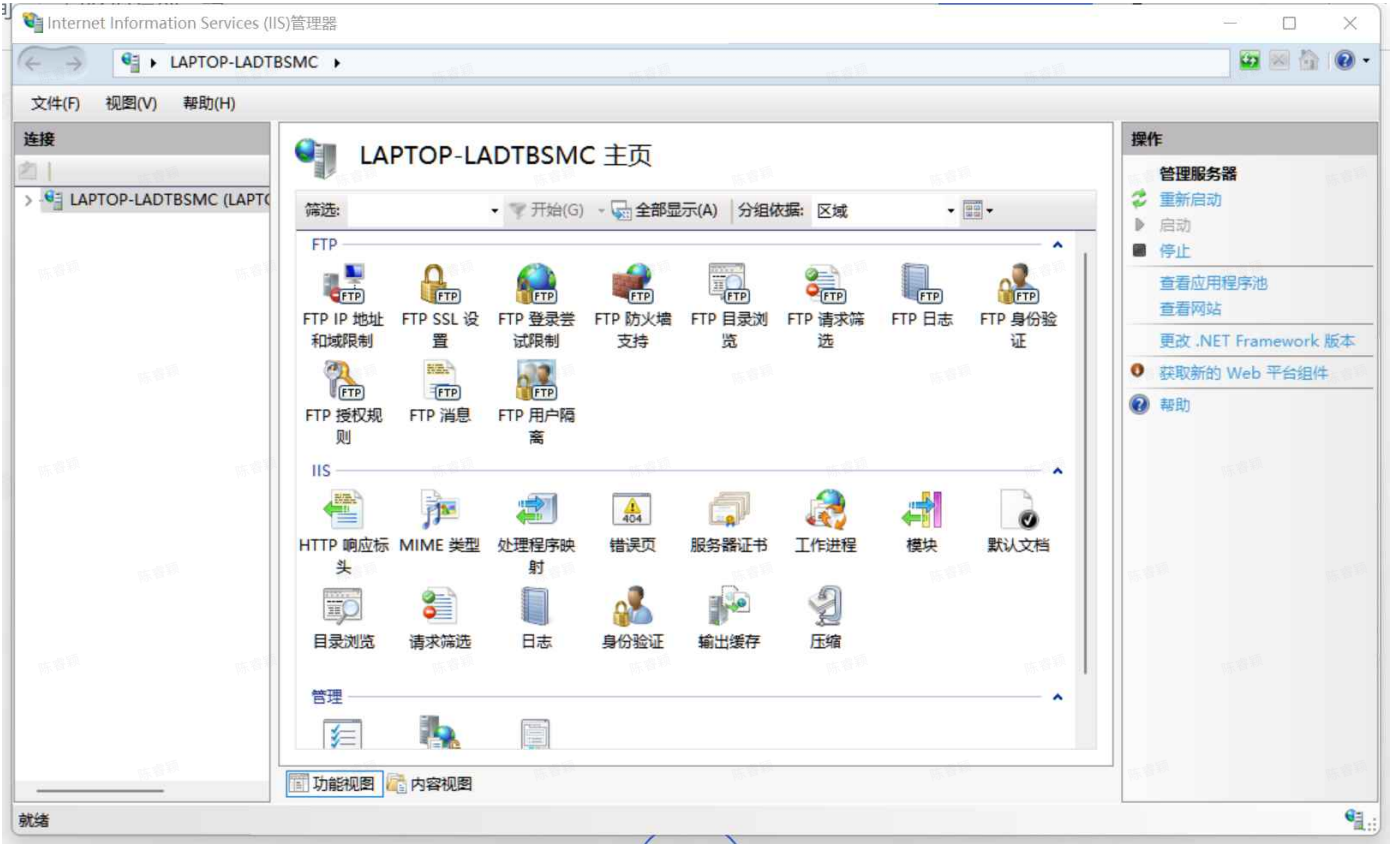


- 进入浏览器并输入localhost

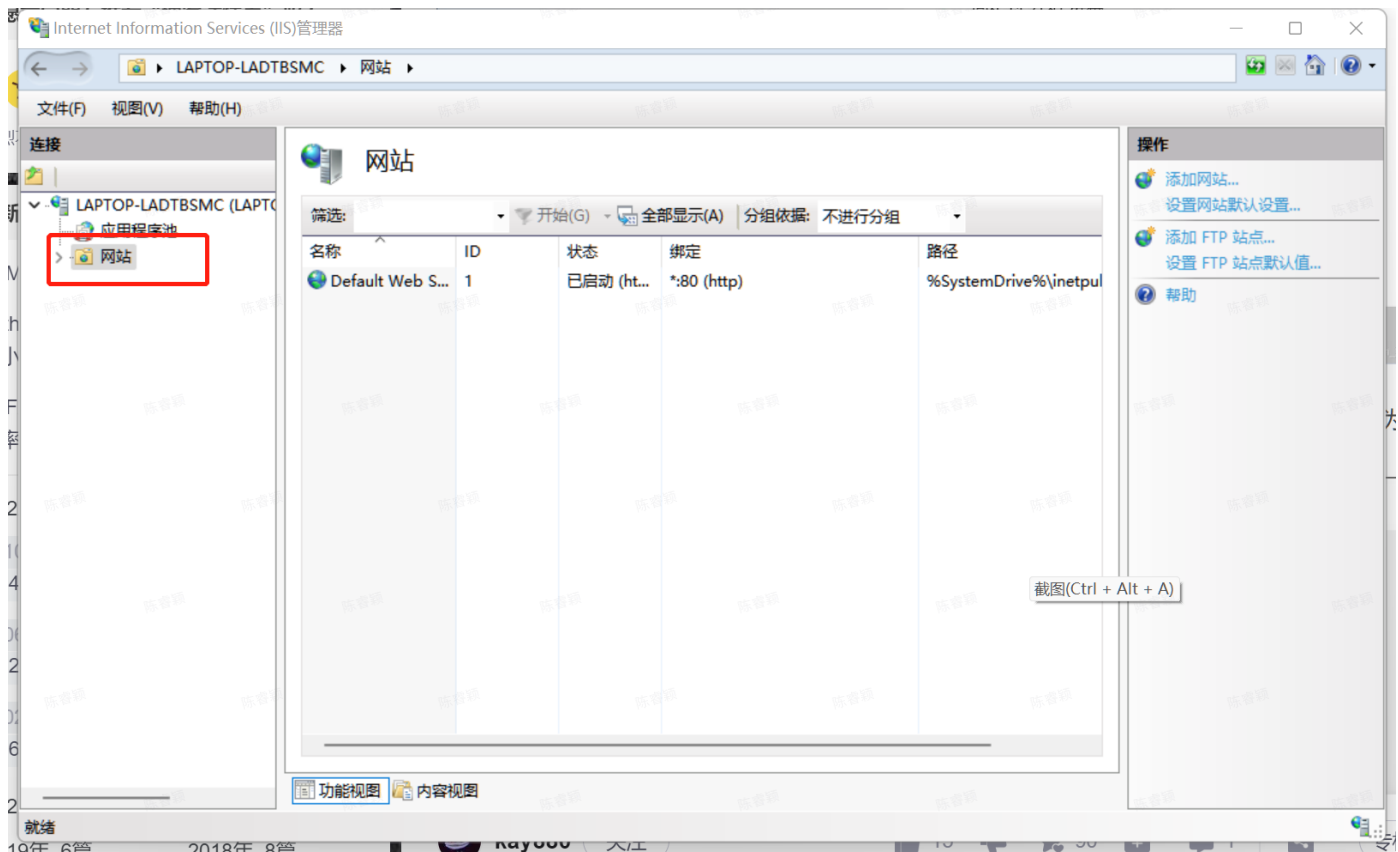


表明开启成功。

c. 打开iis管理工具，界面如图所示：



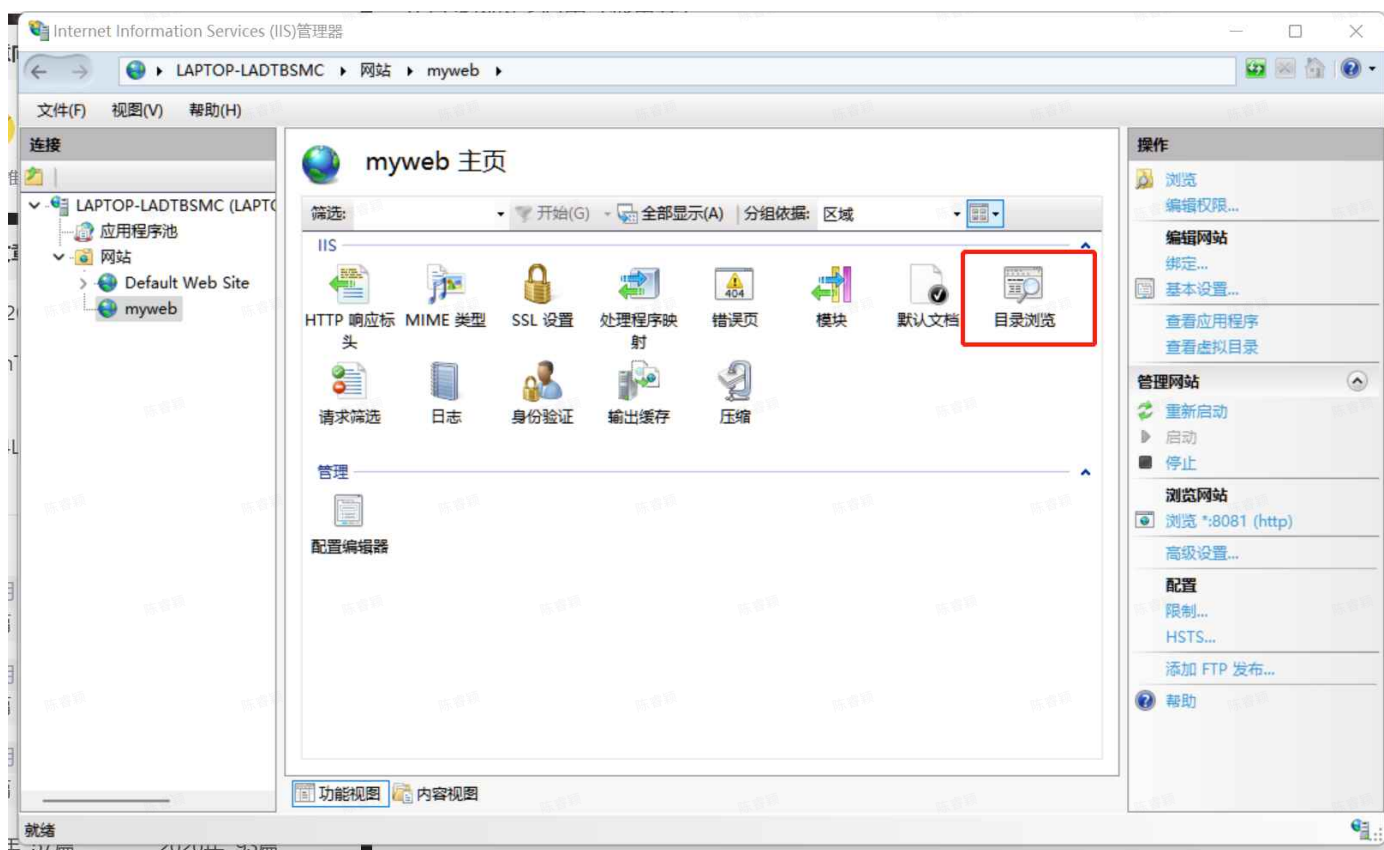
d. 找到网络选项并右击点击添加网络：



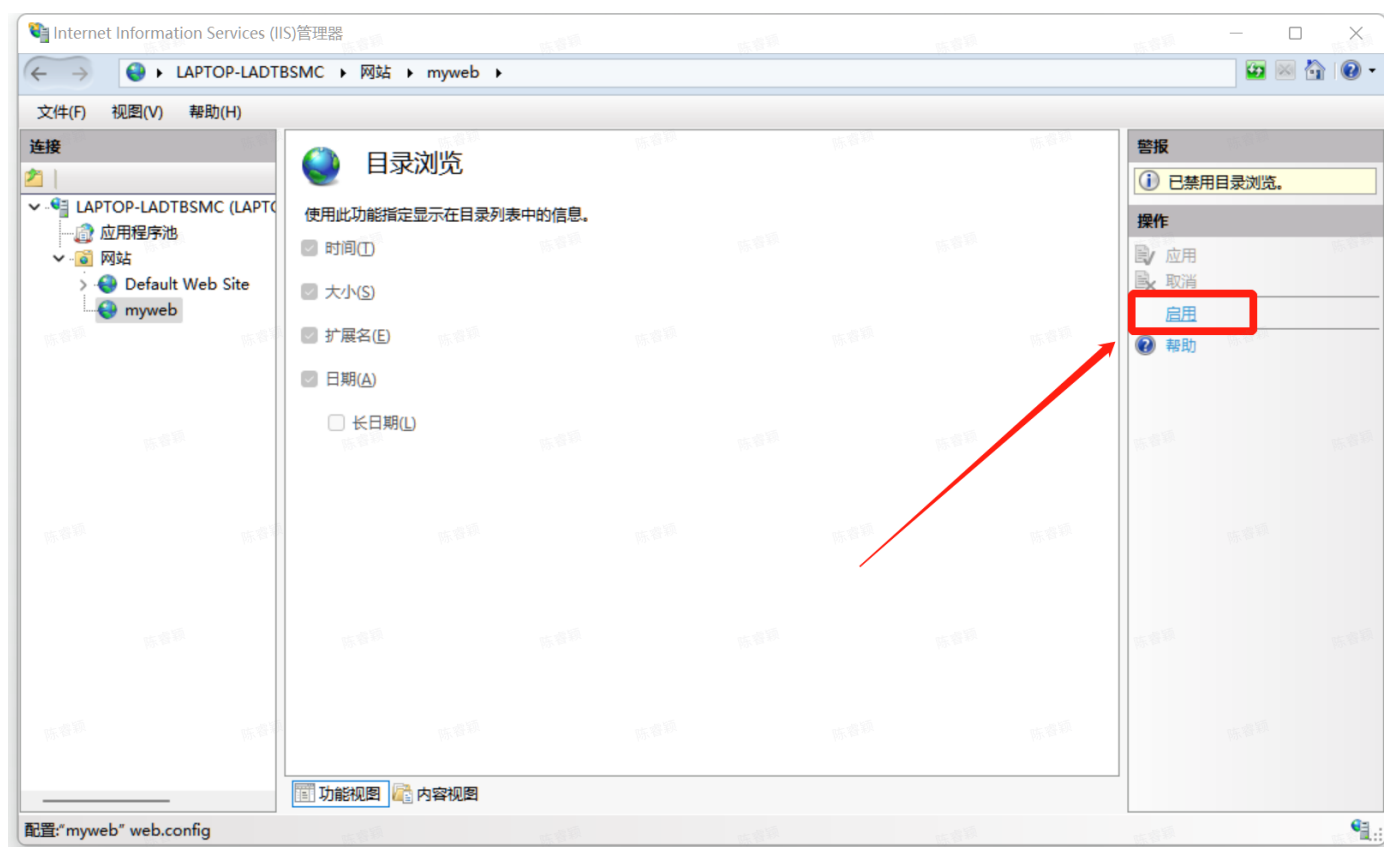
e. 设置网站名称、物理路径及端口，如下图所示：

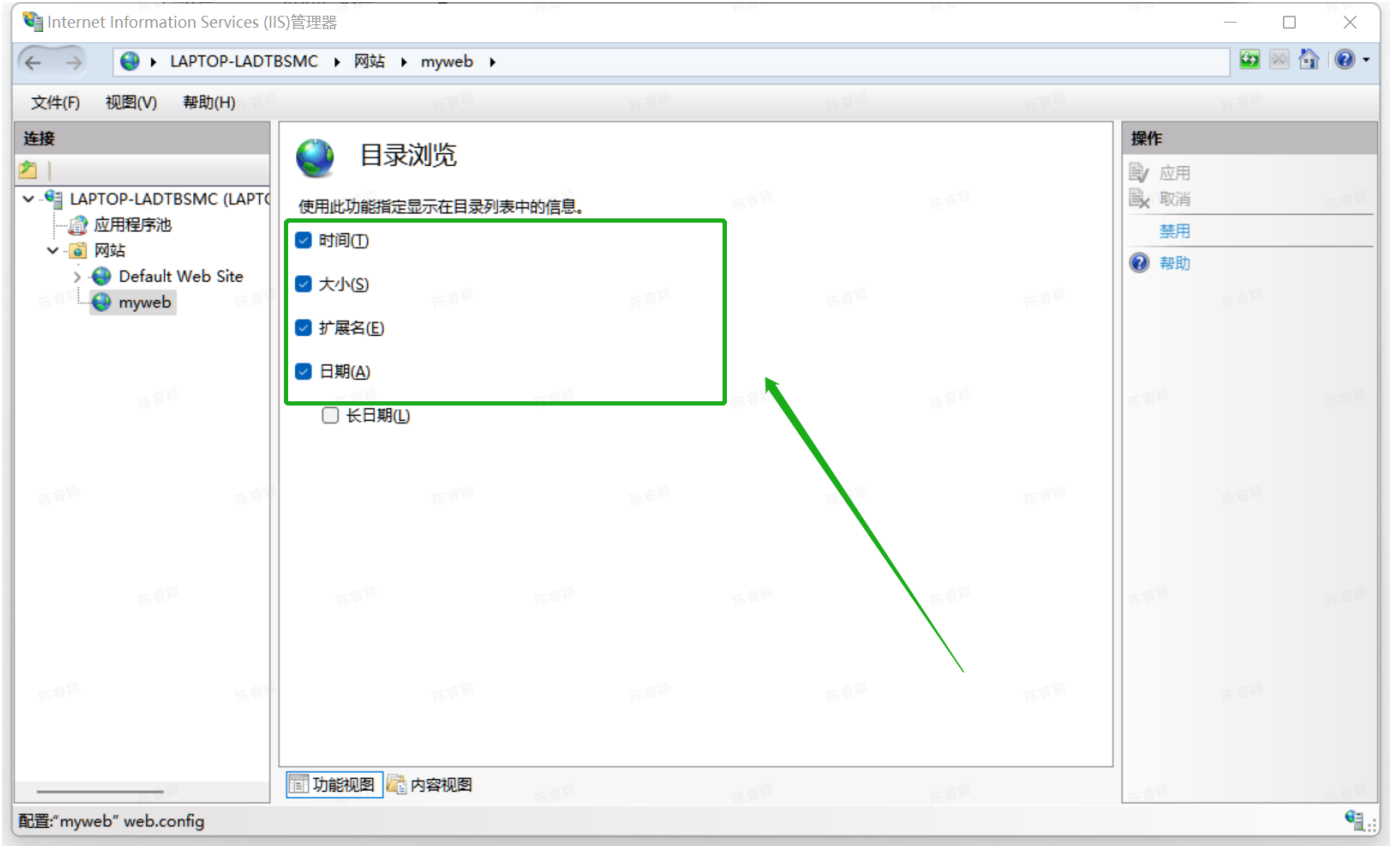


f. 点开myWeb详情页，找到目录浏览并打开：



g. 点击启用:





h. 进入浏览器，输入 `localhost:8081`，显示下面的内容，表示配置成功：



2.2 编写Web页面

```
1 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
2 <html>
3   <head>
4     <title>Welcome to my web</title>
5     <h1>2013544 陈睿颖</h1>
6   </head>
7   <body>
8     <img src=./myname.jpg width="273" height="364">
9   </body>
10 </html>
```

界面如下图所示：



2013544 陈睿颖



2.3 使用Wireshark捕获浏览器与Web服务器的交互过程：

2.3.1 进行抓包

打开wireshark并开始抓包，再使用浏览器访问编写好的页面

由于配置的web服务器的端口为8081，我们需要分析的数据包即为源端口为8081或目的端口为8081的数据包。在过滤条件中输入：

```
1 tcp.port==8081
```

No.	Time	Source	Destination	Protocol	Length	Info
31	1.156177	127.0.0.1	127.0.0.1	TCP	56	19217 → 8081 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
32	1.156224	127.0.0.1	127.0.0.1	TCP	56	8081 → 19217 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
33	1.156255	127.0.0.1	127.0.0.1	TCP	44	19217 → 8081 [ACK] Seq=1 Ack=1 Win=327424 Len=0
46	1.165925	127.0.0.1	127.0.0.1	HTTP	629	GET /info.html HTTP/1.1
47	1.165944	127.0.0.1	127.0.0.1	TCP	44	8081 → 19217 [ACK] Seq=1 Ack=586 Win=2160640 Len=0
48	1.166591	127.0.0.1	127.0.0.1	HTTP	541	HTTP/1.1 200 OK (text/html)
49	1.166620	127.0.0.1	127.0.0.1	TCP	44	19217 → 8081 [ACK] Seq=586 Ack=498 Win=326912 Len=0
78	3.948977	127.0.0.1	127.0.0.1	TCP	44	19217 → 8081 [FIN, ACK] Seq=586 Ack=498 Win=326912 Len=0
79	3.949012	127.0.0.1	127.0.0.1	TCP	44	8081 → 19217 [ACK] Seq=498 Ack=587 Win=2160640 Len=0
80	3.949034	127.0.0.1	127.0.0.1	TCP	44	8081 → 19217 [FIN, ACK] Seq=498 Ack=587 Win=2160640 Len=0
81	3.949073	127.0.0.1	127.0.0.1	TCP	44	19217 → 8081 [ACK] Seq=587 Ack=499 Win=326912 Len=0

得到如图所示的结果。

2.3.2 实验原理

2.3.2.1 TCP可靠数据传输

1. 基本机制
 - 发送端：发送数据、等待确认、超时重传
 - 接收端：进行差错检测，采用累积确认机制
2. 数据单元（段）格式

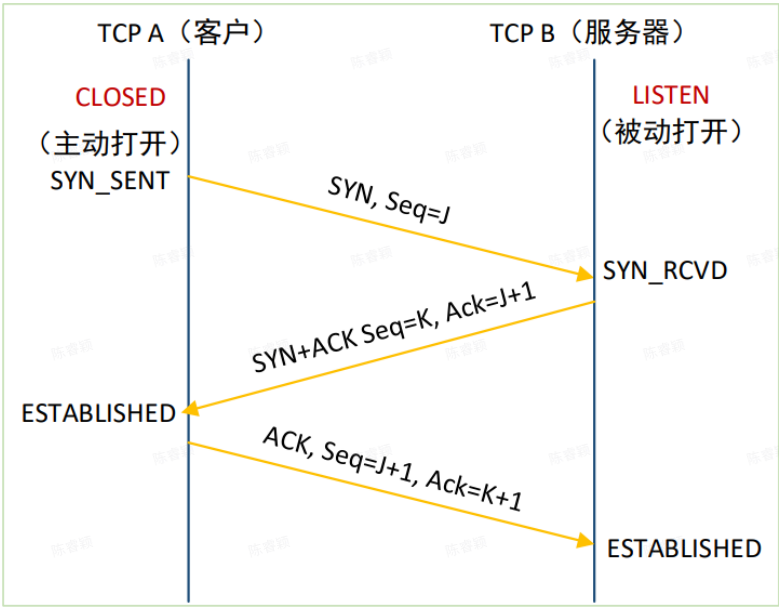
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
源端口号（Source Port）																目的端口号（Destination Port）															
发送序号（sequence number）																															
确认序号（length）																															
头长度		未用				U	A	P	R	S	F	接收窗口通告（rcvr window size）																			
校验和（checksum）																紧急数据指针（ptr urgent data）															
选项（options）																															
数据（data）																															

- 头长度：四个字节为计数单位，包含选项部分
- 接收窗口通告：指示接收缓冲区可接收的字节数
- 标志位：URG,ACK,PUSH,RESET,SYN,FIN
- 选项格式：

Kind(1字节)	Length(1字节)	Info(1字节)
-----------	-------------	-----------

3. 三次握手

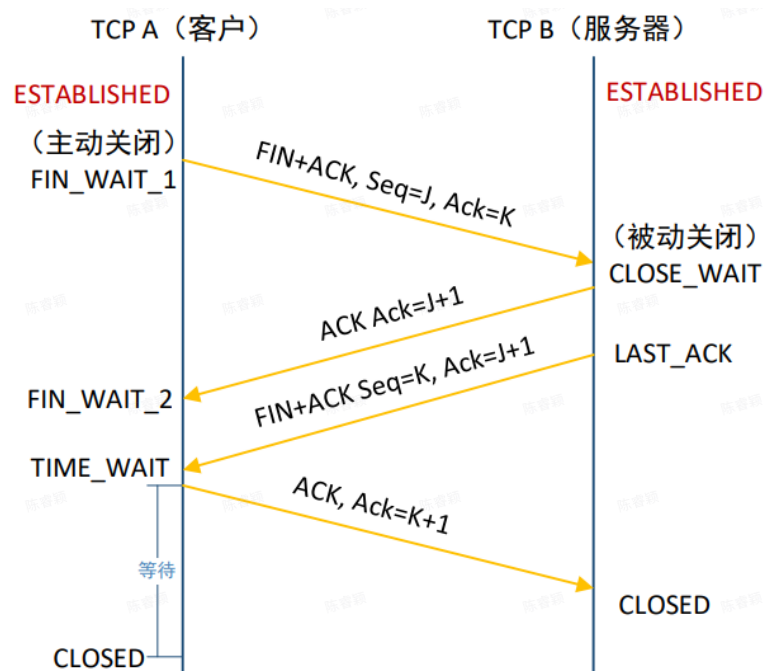
a. 示意图



- 第一次握手：**客户端发送一个TCP的SYN标志位置1的包指明客户打算连接的服务器的端口，以及初始序号X,保存在包头的序列号(Sequence Number)字段里。
- 第二次握手：**服务器发回确认包(ACK)应答。即SYN标志位和ACK标志位均为1同时，将确认序号(Acknowledgement Number)设置为客户的I S N加1以.即X+1。
- 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

4. 四次挥手

a. 示意图



- **第一次挥手：**客户端打算断开连接，向服务器发送FIN报文(FIN标记位被设置为1，1表示为FIN，0表示不是)，FIN报文中会指定一个序列号，之后客户端进入FIN_WAIT_1状态。客户端打算断开连接，向服务器发送FIN报文(FIN标记位被设置为1，1表示为FIN，0表示不是)，FIN报文中会指定一个序列号，之后客户端进入FIN_WAIT_1状态。
- **第二次挥手：**服务器收到连接释放报文段(FIN报文)后，就向客户端发送ACK应答报文，以客户端的FIN报文的序列号 seq+1 作为ACK应答报文段的确认序列号ack = seq+1 = J + 1。接着服务器进入CLOSE_WAIT(等待关闭)状态，此时的TCP处于半关闭状态，客户端到服务器的连接释放。客户端收到来自服务器的ACK应答报文段后，进入FIN_WAIT_2状态。
- **第三次挥手：**服务器也打算断开连接，向客户端发送连接释放(FIN)报文段，之后服务器进入LAST_ACK(最后确认)状态，等待客户端的确认。服务器也打算断开连接，向客户端发送连接释放(FIN)报文段，之后服务器进入LAST_ACK(最后确认)状态，等待客户端的确认。
- **第四次挥手：**服务器也打算断开连接，向客户端发送连接释放(FIN)报文段，之后服务器进入LAST_ACK(最后确认)状态，等待客户端的确认。之后客户端进入TIME_WAIT(时间等待)状态，服务器收到ACK应答报文段后，服务器就进入CLOSE(关闭)状态，到此服务器的连接已经完成关闭。客户端处于TIME_WAIT状态时，此时的TCP还未释放掉，需要等待2MSL后，客户端才进入CLOSE状态。

2.3.2.2 HTTP协议

1. 工作原理

HTTP协议定义Web客户端如何从Web服务器请求Web页面，以及服务器如何把Web页面传送给客户端。HTTP协议采用了请求/响应模型。客户端向服务器发送一个请求报文，请求报文包含请求的方法、URL、协议版本、请求头部和请求数据。服务器以一个状态行作为响应，响应的内容包括协议的版本、成功或者错误代码、服务器信息、响应头部和响应数据。

2. 请求/响应步骤

- 客户端连接到Web服务器 一个HTTP客户端，通常是浏览器，与Web服务器的HTTP端口（默认为80）建立一个TCP套接字连接。例如，<http://www.baidu.com>。
- 发送HTTP请求 通过TCP套接字，客户端向Web服务器发送一个文本的请求报文，一个请求报文由请求行、请求头部、空行和请求数据4部分组成。

- c. 服务器接受请求并返回HTTP响应 Web服务器解析请求，定位请求资源。服务器将资源复本写到TCP套接字，由客户端读取。一个响应由状态行、响应头部、空行和响应数据4部分组成。
- d. 释放连接TCP连接 若connection 模式为close，则服务器主动关闭TCP连接，客户端被动关闭连接，释放TCP连接;若connection 模式为keepalive，则该连接会保持一段时间，在该时间内可以继续接收请求;
- e. 客户端浏览器解析HTML内容 客户端浏览器首先解析状态行，查看表明请求是否成功的状态代码。然后解析每一个响应头，响应头告知以下为若干字节的HTML文档和文档的字符集。客户端浏览器读取响应数据HTML，根据HTML的语法对其进行格式化，并在浏览器窗口中显示。

3. HTTP请求方法

- a. **GET**：向指定的资源发出“显示”请求。使用GET方法应该只用在读取数据，而不应当被用于产生“副作用”的操作中，例如在Web Application中。其中一个原因是GET可能会被网络蜘蛛等随意访问。
- b. **HEAD**：与GET方法一样，都是向服务器发出指定资源的请求。只不过服务器将不传回资源的本文部分。它的好处在于，使用这个方法可以在不必传输全部内容的情况下，就可以获取其中“关于该资源的信息”（元信息或称元数据）
- c. **POST**：与GET方法一样，都是向服务器发出指定资源的请求。只不过服务器将不传回资源的本文部分。它的好处在于，使用这个方法可以在不必传输全部内容的情况下，就可以获取其中“关于该资源的信息”（元信息或称元数据）

此外，还有PUT,DELETE,TRACE,OPTIONS,CONNECT等方法。

4. 请求格式



5. 响应格式



2.3.3 消息分析

2.3.3.1 TCP消息

No.	Time	Source	Destination	Protocol	Length	Info
31	1.156177	127.0.0.1	127.0.0.1	TCP	56	19217 → 8081 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM

消息分为4个部分

```
> Frame 31: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 19217, Dst Port: 8081, Seq: 0, Len: 0
```

1. Frame：物理层的数据帧概况

Frame 31: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on	0000	02 00 00 00 45 00 00 34 7c d1 40 00 80 06 00 00E..4 . @.....
Section number: 1	0010	7f 00 00 01 7f 00 00 01 4b 11 1f 91 95 ce a3 97 K.....
> Interface id: 0 (\Device\NPF_{Loopback})	0020	00 00 00 00 80 02 ff ff d2 e1 00 00 02 04 ff d7
Encapsulation type: NULL/Loopback (15)	0030	01 03 03 08 01 01 04 02
Arrival Time: Oct 26, 2022 15:35:27.139950000 中国标准时间			
[Time shift for this packet: 0.000000000 seconds]			
Epoch Time: 1666769727.139950000 seconds			
[Time delta from previous captured frame: 0.000287000 seconds]			
[Time delta from previous displayed frame: 0.000000000 seconds]			
[Time since reference or first frame: 1.156177000 seconds]			
Frame Number: 31			
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]			

2. Null/Loopback：环回接口

Null/Loopback
Family: IP (2)

协议类型为IPv4

0000	02 00 00 00	45 00 00 34	7c d1 40 00 80 06 00 00E..4 . @.....
0010	7f 00 00 01	7f 00 00 01	4b 11 1f 91 95 ce a3 97 K.....
0020	00 00 00 00	80 02 ff ff	d2 e1 00 00 02 04 ff d7
0030	01 03 03 08	01 01 04 02	

3. Internet Protocol Version 4:互联网层IP包头部信息

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: 0x7cd1 (31953)
> 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

0000	02 00 00 00	45 00 00 34	7c d1 40 00 80 06 00 00E..4 . @.....
0010	7f 00 00 01	7f 00 00 01	4b 11 1f 91 95 ce a3 97 K.....
0020	00 00 00 00	80 02 ff ff	d2 e1 00 00 02 04 ff d7
0030	01 03 03 08	01 01 04 02	

对应的数据结构为：

```
1  typedef struct IPHeader_t { //IP首部
2      BYTE Ver_HLen;
3      BYTE TOS;
4      WORD TotalLen WORD ID;
5      WORD Flag_Segment
6      BYTE TTL;
7      BYTE Protocol;
8      WORD Checksum; //校验和
9      ULONG SrcIP; //源ip地址
10     ULONG DstIP; //目的ip地址
11 }IPHeader_t;
```

源地址为127.0.0.1，以十六进制显示在消息中为

1	7f 00 00 01
---	-------------

Source Address: 127.0.0.1
Destination Address: 127.0.0.1

0000	02 00 00 00 45 00 00 34	7c d1 40 00 80 06 00 00E..4 . @.....
0010	7f 00 00 01 7f 00 00 01	4b 11 1f 91 95 ce a3 97K.....
0020	00 00 00 00 80 02 ff ff	d2 e1 00 00 02 04 ff d7
0030	01 03 03 08 01 01 04 02	

4. Transmission Control Protocol：传输层的数据段头部信息

Transmission Control Protocol, Src Port: 19217, Dst Port: 8081, Seq: 0, Len: 0

Source Port: 19217
Destination Port: 8081
[Stream index: 5]
[Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 2513347479
[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: 0xd2e1 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation
Timestamps
[Time since first frame in this TCP stream: 0.000000000 seconds]
[Time since previous frame in this TCP stream: 0.000000000 seconds]

0000	02 00 00 00 45 00 00 34	7c d1 40 00 80 06 00 00E..4 . @.....
0010	7f 00 00 01 7f 00 00 01	4b 11 1f 91 95 ce a3 97K.....
0020	00 00 00 00 80 02 ff ff	d2 e1 00 00 02 04 ff d7
0030	01 03 03 08 01 01 04 02	

可以看到，消息是从主机的19217端口发送至主机的8081端口

Transmission Control Protocol, Src Port: 19217, Dst Port: 8081, Seq: 0, Len: 0
Source Port: 19217
Destination Port: 8081

0000	02 00 00 00 45 00 00 34	7c d1 40 00 80 06 00 00E..4 . @.....
0010	7f 00 00 01 7f 00 00 01	4b 11 1f 91 95 ce a3 97K.....
0020	00 00 00 00 80 02 ff ff	d2 e1 00 00 02 04 ff d7
0030	01 03 03 08 01 01 04 02	

2.3.3.2 TCP的三次握手

在捕获的数据包中

31	1.156177	127.0.0.1	127.0.0.1	TCP	56	19217 → 8081 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
32	1.156224	127.0.0.1	127.0.0.1	TCP	56	8081 → 19217 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
33	1.156255	127.0.0.1	127.0.0.1	TCP	44	19217 → 8081 [ACK] Seq=1 Ack=1 Win=327424 Len=0

主机向服务器发送连接建立请求SYN，服务器向主机恢复SYN消息并携带确认消息ACK，主机收到服务器的恢复并再次向服务器发送ACK。

2.3.3.3 HTTP消息

64	6.478275	:::1	:::1	HTTP	789	GET / HTTP/1.1
----	----------	------	------	------	-----	----------------

消息的格式为在原有TCP格式的基础上增加超文本传输协议部分

> Frame 46: 629 bytes on wire (5032 bits), 629 bytes captured (5032 bits) on interface \Device
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 19217, Dst Port: 8081, Seq: 1, Ack: 1, Len: 585
Hypertext Transfer Protocol
GET /info.html HTTP/1.1\r\nHost: localhost:8081\r\nUser-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:103.0) Gecko/20100101 Firefox/103\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\nAccept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2\r\nAccept-Encoding: gzip, deflate, br\r\nConnection: keep-alive\r\nUpgrade-Insecure-Requests: 1\r\nSec-Fetch-Dest: document\r\nSec-Fetch-Mode: navigate\r\nSec-Fetch-Site: none\r\nSec-Fetch-User: ?1\r\nIf-Modified-Since: Wed, 26 Oct 2022 07:20:24 GMT\r\nIf-None-Match: "e89e196abe9d81:0"\r\n\r\n[Full request URI: http://localhost:8081/info.html]
[HTTP request 1/1]
[Response in frame: 48]

0020	39 5e 82 66 50 18 04 ff	2d 1a 00 00 47 45 54 20	9^_fp...GET
0030	2f 69 6e 66 6f 2e 68 74	6d 6c 20 48 54 50 2f	/info.html HTTP/
0040	31 2e 31 0d 0a 48 6f 73	74 3a 20 6c 6f 63 61 6c	1.1..Host: local
0050	68 6f 73 74 3a 38 30 38	31 0d 0a 55 73 65 72 2d	host:8081..User-
0060	41 67 65 6e 74 3a 20 4d	6f 7a 69 6c 6c 61 2f 35	Agent: Mozilla/5
0070	2e 30 20 28 57 69 6e 64	6f 77 73 20 4e 54 20 31	.0 (Windows NT 1
0080	30 2e 30 3b 20 57 69 6e	36 34 3b 20 78 36 34 3b	0.0; Win 64; x64;
0090	20 72 76 3a 31 30 33 2e	30 29 28 47 65 63 6b 6f	rv:103.0) Gecko
00a0	2f 32 30 31 30 30 31 30	31 20 46 69 72 65 66 6f	/20100101 Firefo
00b0	78 2f 31 30 33 2e 30 0d	0a 41 63 63 65 70 74 3a	x/103.0..Accept-
00c0	20 74 65 78 74 2f 68 74	6d 6c 2c 61 70 70 6c 69	text/ht ml,appli
00d0	63 61 74 69 6f 6e 2f 78	68 74 6d 6c 2b 78 6d 6c	cation/x html+xml
00e0	2c 61 70 70 6c 69 63 61	74 69 6f 6e 2f 78 6d 6c	,applic ation/xml
00f0	3b 71 3d 30 2e 39 2c 69	6d 61 67 65 2f 61 76 69	;q=0.9,i mage/avi
0100	66 2c 69 6d 61 67 65 2f	77 65 62 70 2c 2a 2f 2a	f,image/ webp,*/*
0110	3b 71 3d 30 2e 38 0d 0a	41 63 63 65 70 74 2d 4c	;q=0.8.. Accept-L
0120	61 6e 67 75 61 67 65 3a	20 7a 68 2d 43 4e 2c 7a	anguage: zh-CN,s
0130	68 3b 71 3d 30 2e 38 2c	7a 68 2d 54 57 3b 71 3d	h;q=0.8, zh-TW;q=
0140	30 2e 37 2c 7a 68 2d 48	4b 3b 71 3d 30 2e 35 2c	0.7,zh-H K;q=0.5,
0150	65 6e 2d 55 53 3b 71 3d	30 2e 33 2c 65 6e 3b 71	en-US;q= 0.3,en;q
0160	3d 30 2e 32 0d 0a 41 63	63 65 70 74 2d 45 6e 63	=0.2..Ac cept-Enc
0170	6f 64 69 6e 67 3a 20 67	7a 69 70 2c 20 64 65 66	oding: g zip, def

可以看到对应的十六进制消息与字符。

2.3.3.4 GET请求获取网页

Hypertext Transfer Protocol

GET /info.html HTTP/1.1\r\n

[Expert Info (Chat/Sequence): GET /info.html HTTP/1.1\r\n]

[GET /info.html HTTP/1.1\r\n]

[Severity level: Chat]

[Group: Sequence]

Request Method: GET

Request URI: /info.html

Request Version: HTTP/1.1

请求获取info.html页面，方式为GET

0020	39 5e 82 66 50 18 04 ff 2d 1a 00 00 47 45 54 20	9^·fp... -...GET
0030	2f 69 6e 66 6f 2e 68 74 6d 6c 20 48 54 54 50 2f	/info.ht ml HTTP/
0040	31 2e 31 0d 0a 48 6f 73 74 3a 20 6c 6f 63 61 6c	1.1·Hos t: local
0050	68 6f 73 74 3a 38 30 38 31 0d 0a 55 73 65 72 2d	host:808 1·User-

2.3.3.5 获取网页文字和图片信息

```
1 <html>
2   <head>
3     <title>Welcome to my web</title>
4     <h1>2013544 陈睿颖</h1>
5   </head>
6   <body>
7     <img src=./myname.jpg width="273" height="364">
8   </body>
9 </html>
```

对应数据包中：

Line-based text data: text/html (11 lines)

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />\r\n

\r\n

<html>\r\n

<head>\r\n

<title>Welcome to my web</title>\r\n

<h1>2013544 陈睿颖</h1>\r\n

</head>\r\n

<body>\r\n

\r\n

</body>\r\n

</html>

0080	4d 54 0d 0a 41 63 63 65 70 74 2d 52 61 6e 67 65	MT·Acce pt-Range
0090	73 3a 20 62 79 74 65 73 0d 0a 45 54 61 67 3a 20	s: bytes ··ETag:
00a0	22 63 32 31 35 39 62 36 31 64 65 39 64 38 31 3a	"c2159b6 1de9d81:
00b0	30 22 0d 0a 53 65 72 76 65 72 3a 20 4d 69 63 72	0"··Serv er: Micr
00c0	6f 73 6f 66 74 2d 49 49 53 2f 31 30 2e 30 0d 0a	osoft-II S/10.0·
00d0	44 61 74 65 3a 20 57 65 64 2c 20 32 36 20 4f 63	Date: We d, 26 Oc
00e0	74 20 32 30 32 32 20 30 37 3a 33 35 3a 32 37 20	t 2022 0 7:35:27
00f0	47 4d 54 0d 0a 43 6f 6e 74 65 6e 74 2d 4c 65 6e	GMT·Con tent-Len
0100	67 74 68 3a 20 32 37 33 0d 0a 0d 0a 3c 6d 65 74	gth: 273 ···<met
0110	61 20 68 74 74 70 2d 65 71 75 69 76 3d 22 43 6f	a http-e quiv="Co
0120	6e 74 65 6e 74 2d 54 79 70 65 22 20 63 6f 6e 74	ntent-Ty pe" cont
0130	65 6e 74 3d 22 74 65 78 74 2f 68 74 6d 6c 3b 20	ent="tex t/html;
0140	63 68 61 72 73 65 74 3d 75 74 66 2d 38 22 20 2f	charset= utf-8" /
0150	3e 0d 0a 0d 0a 3c 68 74 6d 6c 3e 0d 0a 20 20 20	>···<ht ml>·
0160	20 3c 68 65 61 64 3e 0d 0a 20 20 20 20 20 20	<head>·
0170	20 3c 74 69 74 6c 65 3e 57 65 6c 63 6f 6d 65 20	<title> Welcome
0180	74 6f 20 6d 79 20 77 65 62 3c 2f 74 69 74 6c 65	to my we b</title>
0190	3e 0d 0a 20 20 20 20 20 20 20 3c 68 31 3e 32	>·
01a0	30 31 33 35 34 3a 20 e9 99 88 e7 9d bf e9 a2 96	<h1>2
01b0	3c 2f 68 31 3e 0d 0a 20 20 20 20 3c 2f 68 65 61	013544 ·····
01c0	64 3e 0d 0a 20 20 20 20 3c 62 6f 64 79 3e 0d 0a	</h1>·
01d0	20 20 20 20 20 20 20 3c 69 6d 67 20 73 72 63	</hea d>·
		<body>·
		<img src

获取成功，返回状态码200代表请求成功

```

  ▾ Hypertext Transfer Protocol
    ▾ HTTP/1.1 200 OK\r\n
      ▾ [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
        [HTTP/1.1 200 OK\r\n]
        [Severity level: Chat]
        [Group: Sequence]
        Response Version: HTTP/1.1
        Status Code: 200
        [Status Code Description: OK]
        Response Phrase: OK

```

2.3.3.6 TCP的四次挥手

78	3.948977	127.0.0.1	127.0.0.1	TCP	44 19217 → 8081 [FIN, ACK] Seq=586 Ack=498 Win=326912 Len=0
79	3.949012	127.0.0.1	127.0.0.1	TCP	44 8081 → 19217 [ACK] Seq=498 Ack=587 Win=2160640 Len=0
80	3.949034	127.0.0.1	127.0.0.1	TCP	44 8081 → 19217 [FIN, ACK] Seq=498 Ack=587 Win=2160640 Len=0
81	3.949073	127.0.0.1	127.0.0.1	TCP	44 19217 → 8081 [ACK] Seq=587 Ack=499 Win=326912 Len=0

主机向服务器发送FIN，不再发送数据但可以接收数据；服务器接收FIN并回送ACK段；服务器发送FIN并等待主机返回ACK；主机接收FIN后回送ACK，等待两倍的段生存期后关闭连接；服务器接收ACK并关闭连接。

3. 实验总结

经过以上分析，将浏览器和web服务器的交互过程总结如下：

- 浏览器向DNS获取web服务器的IP地址。此处是在本机配置的服务器，所以为127.0.0.1；
- 浏览器与IP地址为127.0.0.1的服务器进行TCP连接，端口为8081，即为myweb设置的端口号；
- 浏览器执行HTTP协议，发送GET localhost:8081/info.html 命令,请求读取该文件，如下图所示：
- 服务器返回localhost:8081/info.html文件到客户端
- 释放TCP连接；
- 浏览器解释localhost:8081/info.html文件内容，并显示该文件表示的页面；