

计算机网络 第一次实验

姓名：周末

专业：计算机科学与技术

学号：2211349

一、网页服务器搭建

二、三次握手

三、GET请求

四、资源解析

五、四次挥手

网页服务器搭建

这里我使用Node.js搭建服务器，它能够使用HTTP模块来处理用户请求，创建一个静态文件服务器，从而在本地搭建了一个可以访问的网页服务器。在项目中添加js文件如下

```

const http = require('http');
const fs = require('fs');
const path = require('path');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  const filePath = path.join(__dirname, req.url === '/' ? 'index.html' : req.url);

  const extname = String(path.extname(filePath)).toLowerCase();
  const mimeTypes = {
    '.html': 'text/html',
    '.js': 'text/javascript',
    '.css': 'text/css',
    '.png': 'image/png',
    '.jpg': 'image/jpeg',
    '.gif': 'image/gif',
    '.mp4': 'video/mp4'
  };

  fs.readFile(filePath, (err, content) => {
    if (err) {
      if (err.code === 'ENOENT') {
        res.writeHead(404, { 'Content-Type': 'text/plain' });
        res.end('404 Not Found');
      } else {
        res.writeHead(500);
        res.end(`Server Error: ${err.code}`);
      }
    } else {
      res.writeHead(200, { 'Content-Type': mimeTypes[extname] || 'application/octet-stream' });
      res.end(content);
    }
  });
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});

server.on('error', (err) => {

```

```
console.error('Server error:', err);  
});
```

通过终端输入node server.js就可以访问<http://127.0.0.1:3000>，即可看到页面



HTML个人页面

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>Personal Page</title>

    <style>
        .info {
            max-width: 800px;
            margin: auto;
            padding: 30px;
            border: 1px solid;
            box-shadow: 0 0 10px;
            font-size: 16px;
            line-height: 24px;
            font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif;
            color: black;
        }

        .info table {
            width: 42%;
            line-height: inherit;
            text-align: center;
            margin: auto;
        }

        .info table td {
            padding: 5px;
            vertical-align: top;
        }

        .info table tr.heading td {
            background: rgb(182, 182, 183);
            border-bottom: 1px solid #ddd;
            font-weight: bold;
        }

        .info table tr.details td {
            padding-bottom: 20px;
        }
    
```

```
</style>
</head>

<body>
  <div class="info">
    <center>
      
    </center>
    <br>
    <p><center>
      <font style="font-family:楷体">
        <br>
        计算机网络-Lab2<br>
      </font></center>
    </p>

    <table cellpadding="0" cellspacing="0">

      <tr class="heading" >
        <td>
          姓名
        </td>

      </tr>

      <tr class="details">
        <td style="font-family:楷体">
          周末
        </td>

      </tr>
      <tr class="heading">
        <td>
          学号
        </td>

      </tr>

      <tr class="details">
        <td style="font-family:楷体">
          2211349
        </td>

      </tr>
    </table>
  </div>
</body>
</html>
```

```

        <tr class="heading">
            <td>
                专业
            </td>

        </tr>

        <tr class="details">
            <td style="font-family:楷体">
                计算机科学与技术
            </td>

        </tr>
    </table>

    <center>
        <p>
            <audio controls>
                <source src="audio.mp3" type="audio/mpeg">
            </audio>
        </p>
    </center>
</div>
</body>
</html>

```

三次握手连接

TCP的三次握手确认客户端和服务端都具备发送和接收数据的能力，并确认各自的初始序列号,用于在客户端和服务端之间建立可靠的连接。

- 客户端发送 SYN：客户端向服务器发送一个SYN标志位为1的报文，表示请求建立连接，并携带初始序列号（seq），准备同步序列号。
- 服务器回应 SYN+ACK：服务器收到SYN后，向客户端回送一个带有SYN和ACK标志的报文，表示确认收到客户端的SYN，同时自己也发送一个SYN请求，与客户端建立同步。这一报文同样带有服务器的初始序列号和对客户端序列号的确认号， $ack = 客户端seq + 1$ 。
- 客户端发送 ACK：客户端收到服务器的SYN+ACK后，向服务器发送ACK报文，表示确认连接建立。此时客户端和服务端进入通信状态，完成三次握手。

选择**Adapter for Loopback Traffic Capture**捕获本地回环流量，即设备与自身的通信流量，并通过指令 `ip.addr == 127.0.0.1&&tcp.port==3000`，过滤出目的地址为 127.0.0.1，即本地主机，以及端口为

3000的网络报文。下图所示三次握手过程。

正在捕获 Adapter for loopback traffic capture

文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(V) 无线(W) 工具(T) 帮助(H)

ip.addr == 127.0.0.1&&tcp.port==3000

No. Time Source Destination Protocol Length Info

4359 110.216361 127.0.0.1 127.0.0.1 TCP 56 52126 → 3000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM

4360 110.216412 127.0.0.1 127.0.0.1 TCP 56 3000 → 52126 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PER

4361 110.216451 127.0.0.1 127.0.0.1 TCP 44 52126 → 3000 [ACK] Seq=1 Ack=1 Win=2161152 Len=0

4362 110.216868 127.0.0.1 127.0.0.1 TCP 56 52127 → 3000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM

4363 110.216909 127.0.0.1 127.0.0.1 TCP 56 3000 → 52127 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PER

4364 110.216955 127.0.0.1 127.0.0.1 TCP 44 52127 → 3000 [ACK] Seq=1 Ack=1 Win=2161152 Len=0

4365 110.217214 127.0.0.1 127.0.0.1 HTTP 708 GET / HTTP/1.1

4366 110.217243 127.0.0.1 127.0.0.1 TCP 44 3000 → 52126 [ACK] Seq=1 Ack=665 Win=2160640 Len=0

4367 110.226814 127.0.0.1 127.0.0.1 HTTP 2591 HTTP/1.1 200 OK (text/html)

4368 110.226907 127.0.0.1 127.0.0.1 TCP 44 52126 → 3000 [ACK] Seq=665 Ack=2548 Win=2158592 Len=0

4380 110.269991 127.0.0.1 127.0.0.1 HTTP 632 GET /image.jpg HTTP/1.1

4381 110.270054 127.0.0.1 127.0.0.1 TCP 44 3000 → 52126 [ACK] Seq=2548 Ack=1253 Win=2159872 Len=0

4382 110.272923 127.0.0.1 127.0.0.1 HTTP 28247 HTTP/1.1 200 OK (JPEG JFIF image)

4383 110.273012 127.0.0.1 127.0.0.1 TCP 44 52126 → 3000 [ACK] Seq=1253 Ack=30751 Win=2130432 Len=0

4387 110.472394 127.0.0.1 127.0.0.1 HTTP 584 GET /audio.mp3 HTTP/1.1

4388 110.472498 127.0.0.1 127.0.0.1 TCP 44 3000 → 52126 [ACK] Seq=30751 Ack=1793 Win=2159360 Len=0

4389 110.473965 127.0.0.1 127.0.0.1 HTTP 231 HTTP/1.1 404 Not Found (text/plain)

4390 110.474030 127.0.0.1 127.0.0.1 TCP 44 52126 → 3000 [ACK] Seq=1793 Ack=30938 Win=2130176 Len=0

> Frame 4359: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface Null/Loopback

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> Transmission Control Protocol, Src Port: 52126, Dst Port: 3000, Seq: 0, Len: 0

0000 02 00 00 00 45 00 00 34 5e 55 40 00 80 06 00 00E..4 ^U@.....

0010 7f 00 00 01 7f 00 00 01 cb 9e 0b b8 3e e4 3d 93>..=.

0020 00 00 00 00 80 02 ff ff 23 1c 00 00 02 04 ff d7#.....

0030 01 03 03 08 01 01 04 02

第一次握手

Wireshark · 分组 4359 · Adapter for loopback traffic capture

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> Transmission Control Protocol, Src Port: 52126, Dst Port: 3000, Seq: 0, Len: 0

Source Port: 52126

Destination Port: 3000

[Stream index: 109]

[Stream Packet Number: 1]

> [Conversation completeness: Complete, WITH_DATA (63)]

[TCP Segment Len: 0]

Sequence Number: 0 (relative sequence number)

Sequence Number (raw): 1055145363

[Next Sequence Number: 1 (relative sequence number)]

Acknowledgment Number: 0

Acknowledgment number (raw): 0

1000 = Header Length: 32 bytes (8)

> Flags: 0x002 (SYN)

000. = Reserved: Not set

...0 = Accurate ECN: Not set

... 0... = Congestion Window Reduced: Not set

.... .0.. = ECN-Echo: Not set

.... ..0. = Urgent: Not set

.... ...0 = Acknowledgment: Not set

.... 0... = Push: Not set

....0.. = Reset: Not set

>1. = Syn: Set

0 = Fin: Not set

0000 02 00 00 00 45 00 00 34 5e 55 40 00 80 06 00 00E..4 ^U@.....

0010 7f 00 00 01 7f 00 00 01 cb 9e 0b b8 3e e4 3d 93>..=.

0020 00 00 00 00 80 02 ff ff 23 1c 00 00 02 04 ff d7#.....

0030 01 03 03 08 01 01 04 02

No.: 4359 · Time: 110.216361 · Source: 127.0.0.1 · Destination: 127.0.0.1 · Protocol: TCP · Length: 56 · Info: 52126 → 3000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM

Show packet bytes

Layout: Vertical (Stacked)

关闭

帮助

客户端向服务器发送的 TCP 连接请求，相关报文信息

- 源端口：52126
- 目标端口：3000
- TCP 标志 (Flags)：SYN 位被设置为 1。表示这是一个连接请求报文，即三次握手中的第一次握手，用于建立连接。其他标志位均未设置，例如 ACK、FIN 等，这与 SYN 报文的性质一致。
- 序列号 (Sequence Number)：相对序列号为 0，表示这是该连接的初始序列号。下一个序列号为 1，表示客户端准备发送的下一个数据包的序列号。
- 最大报文段长度 (MSS)：MSS 值为 65495，表示客户端所能接收的最大数据段大小为 65495 字节。

第二次握手

Wireshark · 分组 4360 · Adapter for loopback traffic capture

> Frame 4360: 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: 3000, Dst Port: 52126, Seq: 0, Ack: 1, Len: 0

Source Port: 3000

Destination Port: 52126

[Stream index: 109]

[Stream Packet Number: 2]

> [Conversation completeness: Complete, WITH_DATA (63)]

[TCP Segment Len: 0]

Sequence Number: 0 (relative sequence number)

Sequence Number (raw): 3686796847

[Next Sequence Number: 1 (relative sequence number)]

Acknowledgment Number: 1 (relative ack number)

Acknowledgment number (raw): 1055145364

1000 = Header Length: 32 bytes (8)

> Flags: 0x012 (SYN, ACK)

000. = Reserved: Not set

...0 = Accurate ECN: Not set

.... 0... = Congestion Window Reduced: Not set

.... .0.. = ECN-Echo: Not set

.... ..0. = Urgent: Not set

.... ...1 = Acknowledgment: Set

.... 0... = Push: Not set

....0.. = Reset: Not set

>1. = Syn: Set

....0 = Fin: Not set

[TCP Flags:A..S.]

0000 02 00 00 00 45 00 00 34 5e 56 40 00 80 06 00 00E..4 ^V@.....

0010 7f 00 00 01 7f 00 00 01 0b b8 cb 9e db c0 0e 2f/

0020 3e e4 3d 94 80 12 ff ff 39 1b 00 00 02 04 ff d7 >.=.....9.....

0030 01 03 03 08 01 01 04 02

No.: 4360 · Time: 110.216412 · Source: 127.0.0.1 · Destination: 127.0.0.1 · Protoc...fo: 3000 → 52126 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM

☒ Show packet bytes

Layout: Vertical (Stacked)

关闭

帮助

服务器接收到了客户端发起的连接请求后，给出应答

- TCP 标志：SYN 和 ACK 位都被设置为 1，这是三次握手中的第二次握手。服务器确认收到客户端的 SYN 请求，并向客户端发送了一个 SYN 请求以建立连接。
- 序列号：0，表明这是服务器向客户端发送的初始数据段。
- 确认号（Acknowledgment Number）：确认号为接收到的 Seq+1，因此设置为1，表示服务器确认收到客户端的初始序列号 0，并期望客户端接下来的数据从序列号 1 开始。

第三次握手

Wireshark · 分组 4361 · Adapter for loopback traffic capture

> Frame 4361: 44 bytes on wire (352 bits), 44 bytes captured (352 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: 52126, Dst Port: 3000, Seq: 1, Ack: 1, Len: 0

Source Port: 52126

Destination Port: 3000

[Stream index: 109]

[Stream Packet Number: 3]

> [Conversation completeness: Complete, WITH_DATA (63)]

[TCP Segment Len: 0]

Sequence Number: 1 (relative sequence number)

Sequence Number (raw): 1055145364

[Next Sequence Number: 1 (relative sequence number)]

Acknowledgment Number: 1 (relative ack number)

Acknowledgment number (raw): 3686796848

0101 = Header Length: 20 bytes (5)

> Flags: 0x010 (ACK)

000. = Reserved: Not set

...0 = Accurate ECN: Not set

.... 0... = Congestion Window Reduced: Not set

.... .0.. = ECN-Echo: Not set

.... ..0. = Urgent: Not set

.... ...1 = Acknowledgment: Set

.... 0... = Push: Not set

....0.. = Reset: Not set

....0. = Syn: Not set

....0 = Fin: Not set

[TCP Flags:A.....]

Window: 8442

0000 02 00 00 00 45 00 00 28 5e 57 40 00 80 06 00 00E..(^W@.....

0010 7f 00 00 01 7f 00 00 01 cb 9e 0b b8 3e e4 3d 94>.=.

0020 db c0 0e 30 50 10 20 fa 53 18 00 00 ...0P. .S...

Flags (tcp.flags), 12 bit(s)

☒ Show packet bytes

Layout: Vertical (Stacked)

关闭

帮助

客户端收到服务器回应的确认报文后，对服务器回应

- TCP 标志：只有 ACK 一位有效，表示这只是一条确认报文，值为1

- 序列号：由于之前已经消耗了一个序号，因此这条报文的序号 Seq 为1

GET请求

正在捕获 Adapter for loopback traffic capture

文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)

ip.addr == 127.0.0.1 & tcp.port == 3000

No.	Time	Source	Destination	Protocol	Length	Info
168	14.342396	127.0.0.1	127.0.0.1	TCP	56	51570 → 3000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
169	14.342462	127.0.0.1	127.0.0.1	TCP	56	3000 → 51570 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
170	14.342528	127.0.0.1	127.0.0.1	TCP	44	51570 → 3000 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
171	14.342986	127.0.0.1	127.0.0.1	TCP	56	51571 → 3000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
172	14.343044	127.0.0.1	127.0.0.1	TCP	56	3000 → 51571 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
173	14.343094	127.0.0.1	127.0.0.1	TCP	44	51571 → 3000 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
174	14.343526	127.0.0.1	127.0.0.1	HTTP	708	GET / HTTP/1.1
175	14.343569	127.0.0.1	127.0.0.1	TCP	44	3000 → 51570 [ACK] Seq=1 Ack=665 Win=2160640 Len=0
176	14.348462	127.0.0.1	127.0.0.1	HTTP	2590	HTTP/1.1 200 OK (text/html)
177	14.348612	127.0.0.1	127.0.0.1	TCP	44	51570 → 3000 [ACK] Seq=665 Ack=2547 Win=2158592 Len=0
178	14.363699	127.0.0.1	127.0.0.1	HTTP	631	GET /logo.jpg HTTP/1.1
179	14.363765	127.0.0.1	127.0.0.1	TCP	44	3000 → 51570 [ACK] Seq=2547 Ack=1252 Win=2159872 Len=0
180	14.366217	127.0.0.1	127.0.0.1	HTTP	28247	HTTP/1.1 200 OK (JPEG JFIF image)
181	14.366378	127.0.0.1	127.0.0.1	TCP	44	51570 → 3000 [ACK] Seq=1252 Ack=30750 Win=2130432 Len=0
182	14.664344	127.0.0.1	127.0.0.1	HTTP	584	GET /audio.mp3 HTTP/1.1
183	14.664410	127.0.0.1	127.0.0.1	TCP	44	3000 → 51570 [ACK] Seq=30750 Ack=1792 Win=2159360 Len=0
184	14.665480	127.0.0.1	127.0.0.1	HTTP	231	HTTP/1.1 404 Not Found (text/plain)
185	14.665556	127.0.0.1	127.0.0.1	TCP	44	51570 → 3000 [ACK] Seq=1792 Ack=30937 Win=2130176 Len=0

> Frame 170: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 51570, Dst Port: 3000, Seq: 1, Ack:

0000 02 00 00 00 45 00 00 28 68 a9 40 00 80 06 00 00E..h@.....
0010 7f 00 00 01 7f 00 00 01 c9 72 0b b8 fc c2 05 dbP.....
0020 da 58 cd c6 50 10 20 fa 10 f0 00 00X..P.....

wireshark_Adapter for loopback traffic captureSTMIW2.pcapng 分组: 652 · Displayed: 34 (5.2%) 配置: Default

通过三次握手建立 TCP 连接后，客户端然后再通过 HTTP 协议发送请求获取 HTML 文件资源。这里可以观察到是从客户端到服务器端口号的一次请求，客户端首先通过 HTTP 的 GET 请求向服务端请求 HTML 文件资源。

接着还有一条由服务器向客户端的报文，这是服务器通过TCP回应一条确认报文。

Wireshark · 分组 174 · Adapter for loopback traffic capture

> Frame 174: 708 bytes on wire (5664 bits), 708 bytes captured (5664 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: 51570, Dst Port: 3000, Seq: 1, Ack: 1, Len: 664

Source Port: 51570

Destination Port: 3000

[Stream index: 19]

[Stream Packet Number: 4]

> [Conversation completeness: Complete, WITH_DATA (63)]

[TCP Segment Len: 664]

Sequence Number: 1 (relative sequence number)

Sequence Number (raw): 4240573915

[Next Sequence Number: 665 (relative sequence number)]

Acknowledgment Number: 1 (relative ack number)

Acknowledgment number (raw): 3663252934

0101 = Header Length: 20 bytes (5)

> Flags: 0x018 (PSH, ACK)

Window: 8442

[Calculated window size: 2161152]

[Window size scaling factor: 256]

Checksum: 0xb003 [unverified]

[Checksum Status: Unverified]

Urgent Pointer: 0

> [Timestamps]

> [SEQ/ACK analysis]

TCP payload (664 bytes)

▼ Hypertext Transfer Protocol

> GET / HTTP/1.1\r\n

Host: 127.0.0.1:3000\r\n

0010 7f 00 00 01 7f 00 00 01 c9 72 0b b8 fc c2 05 dbr.....

0020 da 58 cd c6 50 18 20 fa b0 03 00 00 47 45 54 20 .X..P.GET

0030 2f 20 48 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74 / HTTP/1 .1..Host

0040 3a 20 31 32 37 2e 30 2e 30 2e 31 3a 33 30 30 30 : 127.0. 0.1:3000

No.: 174 · Time: 14.343526 · Source: 127.0.0.1 · Destination: 127.0.0.1 · Protocol: HTTP · Length: 708 · Info: GET / HTTP/1.1

☒ Show packet bytes Layout: Vertical (Stacked) ▼

关闭 帮助

- PSH (Push Flag) : Flags新增PSH信号，表示该数据报文需要立即交付给上层应用程序。在 TCP 中，数据可以暂时缓存在接收端的 TCP 缓冲区中，而 PSH 标志的作用是通知接收端，立即将此数据交给应用层，不要等待更多的数据。
- Seq: 这次发送的是一个 HTTP 请求，因此 Seq 从 1 开始。同时，因为 TCP 报文携带了 HTTP 数据（比如请求头和内容），数据段长度为 664 字节，因此下一次发送的 Seq 值会变成 $1 + 664 = 665$ 。

Wireshark packet capture details for an HTTP GET request (packet 0020):

- IP: 127.0.0.1 → 127.0.0.1
- TCP: 54321 → 80
- HTTP: GET / HTTP/1.1

Request details:

- Request Method: GET
- Request URI: /
- Request Version: HTTP/1.1
- Host: 127.0.0.1:3000
- Connection: keep-alive
- sec-ch-ua: "Chromium";v="130", "Google Chrome";v="130", "Not?A_Brand";v="99"
- sec-ch-ua-mobile: ?0
- sec-ch-ua-platform: "Windows"
- Upgrade-Insecure-Requests: 1
- User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.0.../
- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,app...
- Sec-Fetch-Site: none
- Sec-Fetch-Mode: navigate
- Sec-Fetch-User: ?1
- Sec-Fetch-Dest: document
- Accept-Encoding: gzip, deflate, br, zstd
- Accept-Language: zh-CN,zh;q=0.9

Full request URI: http://127.0.0.1:3000/

Packet bytes (hex/ASCII):

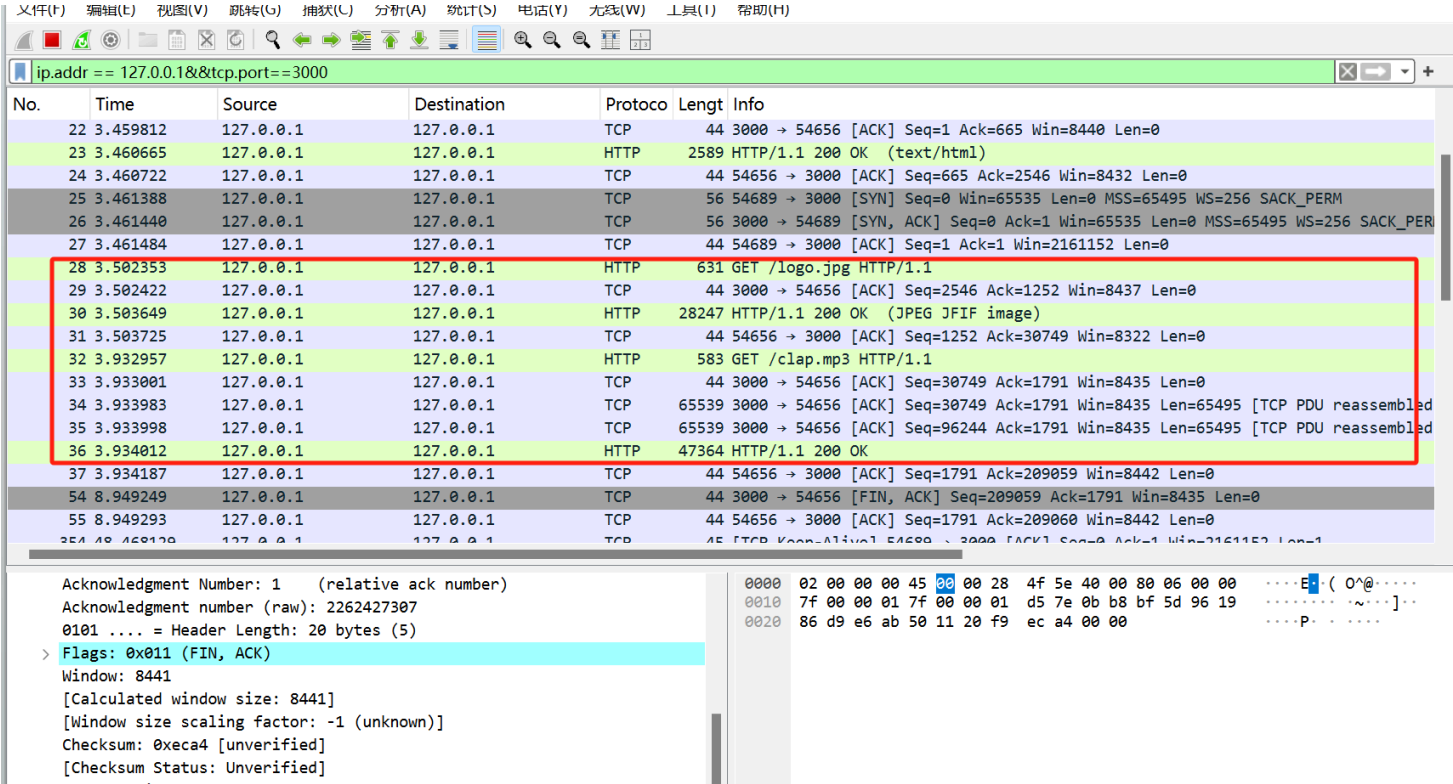
```
0020  da 58 cd c6 50 18 20 fa b0 03 00 00 47 45 54 20  ·X·P· · · · GET
0030  2f 20 48 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74  / HTTP/1.1 · Host
0040  3a 20 31 32 37 2e 30 2e 30 2e 31 3a 33 30 30 30  : 127.0.0.1:3000
0050  0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 6b 65  ··Connec tion: ke
```

可以从GET请求头中看到一些信息：

- Host：请求的主机名，也就是服务器的地址
- Connection：表示连接方式， keep-alive 表示长连接的方式
- User-Agent：客户端代理类型，系统信息以及浏览器的类型
- Accept：客户端可以识别的响应内容，可以看到html文件， image图片等资源

接着服务器在接收到了客户端发送的GET请求之后，会通过TCP回应一条确认报文。客户端在收到服务器的响应之后，同样会给服务器端发送确认报文。服务器在收到了客户端发送来的确认报文之后，然后再将HTML文件数据返回给客户端。接下来分析资源解析。

资源解析



- 接着客户端会再次通过HTTP协议中的GET方法和服务器请求相关的资源，如接下来的图片和音频，与前面大致相同，这里不再赘述。
- 但可以观察到，为了请求 logo.jpg 之前，客户端又和服务器建立了一个三次握手的TCP连接过程。是因为在 HTTP/1.1 及之前的 HTTP 协议中，浏览器通常会为每一个资源（例如 HTML 文件、图片等）与服务器建立一个单独的 TCP 连接，为了以避免过多资源请求导致的阻塞。并且浏览器通常会同时加载页面上的多个资源，以提升页面加载速度。
- 同时观察到图片资源的TCP报文被拆分成了三段进行传输，客户端在接收到服务器发来的完整TCP报文之后才会给服务端发送 ACK 确认报文。这是因为数据量比较大，超过了 MSS，服务器端通过分别发送RGB三个通道的方式来传输图片。
- mps音频格式原理相同，传输方式大致类似，也不再赘述。

四次挥手

四次挥手用于在客户端和服务端之间断开连接，目的是确保双方都完成数据传输并同意断开。（发起断开的一方可以是客户端也可以是服务器，客户端完成了请求和接收数据，例如网页加载等；或是服务器完成了数据的发送，且判断该连接不再需要维持，都会主动断开连接）

正在捕获 Adapter for loopback traffic capture

文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(V) 无线(W) 工具(T) 帮助(H)

ip.addr == 127.0.0.1&&tcp.port==3000

No.	Time	Source	Destination	Protocol	Length	Info
20	5.895642	127.0.0.1	127.0.0.1	TCP	44	60371 → 3000 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
21	5.895867	127.0.0.1	127.0.0.1	HTTP	708	GET / HTTP/1.1
22	5.895887	127.0.0.1	127.0.0.1	TCP	44	3000 → 60370 [ACK] Seq=1 Ack=665 Win=2160640 Len=0
23	5.904752	127.0.0.1	127.0.0.1	HTTP	2589	HTTP/1.1 200 OK (text/html)
24	5.904873	127.0.0.1	127.0.0.1	TCP	44	60370 → 3000 [ACK] Seq=665 Ack=2546 Win=2158592 Len=0
27	5.945880	127.0.0.1	127.0.0.1	HTTP	631	GET /logo.jpg HTTP/1.1
28	5.945916	127.0.0.1	127.0.0.1	TCP	44	3000 → 60370 [ACK] Seq=2546 Ack=1252 Win=2159872 Len=0
29	5.947460	127.0.0.1	127.0.0.1	HTTP	28247	HTTP/1.1 200 OK (JPEG JFIF image)
30	5.947588	127.0.0.1	127.0.0.1	TCP	44	60370 → 3000 [ACK] Seq=1252 Ack=30749 Win=2130432 Len=0
33	6.137946	127.0.0.1	127.0.0.1	HTTP	583	GET /clap.mp3 HTTP/1.1
34	6.137981	127.0.0.1	127.0.0.1	TCP	44	3000 → 60370 [ACK] Seq=30749 Ack=1791 Win=2159360 Len=0
35	6.139293	127.0.0.1	127.0.0.1	TCP	65539	3000 → 60370 [ACK] Seq=30749 Ack=1791 Win=2159360 Len=65495 [TCP PDU reassemb
36	6.139326	127.0.0.1	127.0.0.1	TCP	65539	3000 → 60370 [ACK] Seq=96244 Ack=1791 Win=2159360 Len=65495 [TCP PDU reassemb
37	6.139348	127.0.0.1	127.0.0.1	HTTP	47364	HTTP/1.1 200 OK
38	6.139547	127.0.0.1	127.0.0.1	TCP	44	60370 → 3000 [ACK] Seq=1791 Ack=209059 Win=2161152 Len=0
78	11.141017	127.0.0.1	127.0.0.1	TCP	44	3000 → 60370 [FIN, ACK] Seq=209059 Ack=1791 Win=2159360 Len=0
79	11.141120	127.0.0.1	127.0.0.1	TCP	44	60370 → 3000 [ACK] Seq=1791 Ack=209060 Win=2161152 Len=0
151	31.156594	127.0.0.1	127.0.0.1	TCP	44	60371 → 3000 [FIN, ACK] Seq=1 Ack=1 Win=2161152 Len=0
152	31.156632	127.0.0.1	127.0.0.1	TCP	44	3000 → 60371 [ACK] Seq=1 Ack=2 Win=2161152 Len=0
153	31.156678	127.0.0.1	127.0.0.1	TCP	44	60370 → 3000 [FIN, ACK] Seq=1791 Ack=209060 Win=2161152 Len=0
154	31.156710	127.0.0.1	127.0.0.1	TCP	44	3000 → 60370 [ACK] Seq=209060 Ack=1792 Win=2159360 Len=0
157	31.157316	127.0.0.1	127.0.0.1	TCP	44	3000 → 60371 [FIN, ACK] Seq=1 Ack=2 Win=2161152 Len=0
158	31.157352	127.0.0.1	127.0.0.1	TCP	44	60371 → 3000 [ACK] Seq=2 Ack=2 Win=2161152 Len=0

> Frame 13: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 60209, Dst Port: 3000, Seq: 1, Ack: 2, Win: 2161152, Len: 0

0000 02 00 00 00 45 00 00 28 ff cb 40 00 80 06 00 00E..@.....
0010 7f 00 00 01 7f 00 00 01 eb 31 0b b8 40 3f 6b ab1..@?k.
0020 13 b4 7e c3 50 11 20 fa 5b 8b 00 00~.P.

Adapter for loopback traffic capture: <live capture in progress> 分组: 6588 · Displayed: 30 (0.5%) 配置: Default

第一次挥手

Wireshark · 分组 153 · Adapter for loopback traffic capture

> Frame 153: 44 bytes on wire (352 bits), 44 bytes captured (352 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: 60370, Dst Port: 3000, Seq: 1791, Ack: 209060, Len: 0

Source Port: 60370

Destination Port: 3000

[Stream index: 5]

[Stream Packet Number: 20]

> [Conversation completeness: Complete, WITH_DATA (31)]

[TCP Segment Len: 0]

Sequence Number: 1791 (relative sequence number)

Sequence Number (raw): 3016299169

[Next Sequence Number: 1792 (relative sequence number)]

Acknowledgment Number: 209060 (relative ack number)

Acknowledgment number (raw): 1997584458

0101 = Header Length: 20 bytes (5)

> Flags: 0x011 (FIN, ACK)

000. = Reserved: Not set

...0 = Accurate ECN: Not set

.... 0... = Congestion Window Reduced: Not set

.... .0.. = ECN-Echo: Not set

.... ..0. = Urgent: Not set

.... ...1 = Acknowledgment: Set

.... 0... = Push: Not set

....0.. = Reset: Not set

....0. = Syn: Not set

>1 = Fin: Set

> [TCP Flags:A...F]

Window: 8442

0000 02 00 00 00 45 00 00 28 00 57 40 00 80 06 00 00E.. .W@.....

0010 7f 00 00 01 7f 00 00 01 eb d2 0b b8 b3 c9 12 a1

0020 77 10 b8 4a 50 11 20 fa a3 86 00 00 w..JP.

No.: 153 · Time: 31.156678 · Source: 127.0.0.1 · Destination: 127.0.0.1 · Protocol: TCP · Length: 44 · Info: 60370 → 3000 [FIN, ACK] Seq=1791 Ack=209060 Win=2161152 Len=0

☒ Show packet bytes

Layout: Vertical (Stacked)

关闭

帮助

客户端加载到全部所需的资源后，客户端主动向服务器发起了断开连接的请求。

关闭连接时，发送一个带有 FIN 标志的报文，表示数据传输完成，并请求关闭连接。此时，客户端进入 FIN-WAIT-1 状态。FIN 报文还包含客户端的当前序列号1791。

第二次挥手

Wireshark · 分组 154 · Adapter for loopback traffic capture

> Frame 154: 44 bytes on wire (352 bits), 44 bytes captured (352 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: 3000, Dst Port: 60370, Seq: 209060, Ack: 1792, Len: 0

Source Port: 3000

Destination Port: 60370

[Stream index: 5]

[Stream Packet Number: 21]

> [Conversation completeness: Complete, WITH_DATA (31)]

[TCP Segment Len: 0]

Sequence Number: 209060 (relative sequence number)

Sequence Number (raw): 1997584458

[Next Sequence Number: 209060 (relative sequence number)]

Acknowledgment Number: 1792 (relative ack number)

Acknowledgment number (raw): 3016299170

0101 = Header Length: 20 bytes (5)

> Flags: 0x010 (ACK)

000. = Reserved: Not set

...0 = Accurate ECN: Not set

.... 0... = Congestion Window Reduced: Not set

.... .0.. = ECN-Echo: Not set

.... ..0. = Urgent: Not set

.... ...1 = Acknowledgment: Set

.... 0... = Push: Not set

....0.. = Reset: Not set

....0. = Syn: Not set

....0 = Fin: Not set

[TCP Flags:A.....]

Window: 8435

0000 02 00 00 00 45 00 00 28 00 58 40 00 80 06 00 00E..(.X@.....

0010 7f 00 00 01 7f 00 00 01 0b b8 eb d2 77 10 b8 4aw...J

0020 b3 c9 12 a2 50 10 20 f3 a3 8d 00 00P.

Flags (tcp.flags), 12 bit(s)

☒ Show packet bytes

Layout: Vertical (Stacked)

关闭

帮助

服务器接收到客户端的 FIN 后，发送一个带有 ACK 标志的报文，表示确认收到关闭请求。此时，服务器进入 CLOSE-WAIT 状态，而客户端进入 FIN-WAIT-2 状态，等待服务器的 FIN 报文。这个 ACK 的确认号为客户端的 FIN 报文的序列号加 1。

第三次挥手

Wireshark · 分组 157 · Adapter for loopback traffic capture

> Frame 157: 44 bytes on wire (352 bits), 44 bytes captured (352 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: 3000, Dst Port: 60371, Seq: 1, Ack: 2, Len: 0

Source Port: 3000

Destination Port: 60371

[Stream index: 6]

[Stream Packet Number: 6]

> [Conversation completeness: Complete, NO_DATA (23)]

[TCP Segment Len: 0]

Sequence Number: 1 (relative sequence number)

Sequence Number (raw): 2054142672

[Next Sequence Number: 2 (relative sequence number)]

Acknowledgment Number: 2 (relative ack number)

Acknowledgment number (raw): 2158895659

0101 = Header Length: 20 bytes (5)

> Flags: 0x011 (FIN, ACK)

000. = Reserved: Not set

...0 = Accurate ECN: Not set

.... 0... = Congestion Window Reduced: Not set

.... .0.. = ECN-Echo: Not set

.... ..0. = Urgent: Not set

.... ...1 = Acknowledgment: Set

.... 0... = Push: Not set

....0.. = Reset: Not set

....0. = Syn: Not set

>1 = Fin: Set

> [TCP Flags:A...F]

Window: 8442

0000 02 00 00 00 45 00 00 28 00 5b 40 00 80 06 00 00E..(.[@.....

0010 7f 00 00 01 7f 00 00 01 0b b8 eb d3 7a 6f ba d0zo...

0020 80 ae 22 2b 50 11 20 fa c1 31 00 00 .."+P. .1..

No.: 157 · Time: 31.157316 · Source: 127.0.0.1 · Destination: 127.0.0.1 · Protocol: TCP · Length: 44 · Info: 3000 → 60371 [FIN, ACK] Seq=1 Ack=2 Win=2161152 Len=0

☒ Show packet bytes

Layout: Vertical (Stacked)

关闭

帮助

服务器确认数据传输完成并准备关闭连接时，再次发送一个带有 FIN 标志的报文给客户端，表示同意断开连接。此时，服务器进入 LAST-ACK 状态，等待客户端的最终确认。

第四次挥手

Wireshark · 分组 158 · Adapter for loopback traffic capture

> Frame 158: 44 bytes on wire (352 bits), 44 bytes captured (352 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: 60371, Dst Port: 3000, Seq: 2, Ack: 2, Len: 0

Source Port: 60371

Destination Port: 3000

[Stream index: 6]

[Stream Packet Number: 7]

> [Conversation completeness: Complete, NO_DATA (23)]

[TCP Segment Len: 0]

Sequence Number: 2 (relative sequence number)

Sequence Number (raw): 2158895659

[Next Sequence Number: 2 (relative sequence number)]

Acknowledgment Number: 2 (relative ack number)

Acknowledgment number (raw): 2054142673

0101 = Header Length: 20 bytes (5)

> Flags: 0x010 (ACK)

000. = Reserved: Not set

...0 = Accurate ECN: Not set

.... 0... = Congestion Window Reduced: Not set

.... .0.. = ECN-Echo: Not set

.... ..0. = Urgent: Not set

.... ...1 = Acknowledgment: Set

.... 0... = Push: Not set

....0.. = Reset: Not set

....0. = Syn: Not set

....0 = Fin: Not set

[TCP Flags:A.....]

Window: 8442

0000 02 00 00 00 45 00 00 28 00 5c 40 00 80 06 00 00E..(.\@.....

0010 7f 00 00 01 7f 00 00 01 eb d3 0b b8 80 ae 22 2b+

0020 7a 6f ba d1 50 10 20 fa c1 31 00 00 zo..P. .1..

No.: 158 · Time: 31.157352 · Source: 127.0.0.1 · Destination: 127.0.0.1 · Protocol: TCP · Length: 44 · Info: 60371 → 3000 [ACK] Seq=2 Ack=2 Win=2161152 Len=0

☒ Show packet bytes

Layout: Vertical (Stacked)

关闭

帮助

客户端接收到服务器的 FIN 报文后，发送最后一个 ACK 报文，确认连接已关闭。此时，客户端进入 TIME-WAIT 状态，等待一段时间以确保服务器接收到了 ACK 后关闭。完成等待后，客户端进入 CLOSED 状态，最终断开连接。