计算机网络实验二

express安装与练习demo

采用express框架，参考Node.js搭建Web服务器 - 简书 (jianshu.com)

期间踩坑：

1. npm install卡住不动：使用代理，目标路径下命令行，输入

   npm config set registry https://registry.npm.taobao.org

2. cannot find module express：在当前目录下再次npm install express
3. 获取本地ip：cmd->ipconfig，或直接用localhost
4. body-parser deprecated undefined extended...：参考网络资源

https://blog.csdn.net/weixin_43654123/article/details/122340088

html编写

编写一个简单的html即可。

由于需要使用本地服务器，对图片的导入不能是静态页面，直接导入相对路径。由于要包含一个logo，需要将图片转换为base64编码，传入img的src参数即可。可以编写后端代码进行转换，但不是本实验的重点，这里直接百度“图片转换base64”完成。

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>EX2</title>
</head>
<body>
  <a href="https://cc.nankai.edu.cn/" title="南开计算机">
  <img src="这里填入转换后的base64" width="150" alt="姚翔载" class="panel-
cover__logo logo">
  </a>
  <hr class="panel-cover__divider">
  <p align="center"><font size="8">姚翔载</font><br />
  <font size="6" align="center">专业：计算机科学与技术</font><br/>
  <font size="5" align="center">学号：2011273</font><br/></p>
<p><i><font color="blue">计算机网络课程实验二</font></i></p>
</body>
</html>
```

使用express搭建服务器

经过实测，还需要使用npm install分别安装

express-session

cookie-parser

morgan

ejs

在安装好express的文件夹或子文件夹下,打开命令行

express -e 项目名称

打开项目文件中./bin/www设置ip和端口。为了好区分，我这里这样设置：

```
var port = normalizePort(process.env.PORT || '4321');
app.set('port', port);
app.set('host','127.6.7.8');
```

编写app.js，如下

```
//导入express
var express =require("express");
var app =express();
app.use(express.static('public'))

//参数‘/’可当作设置url的根显示页面，这里即”http://localhost:3000/“访问的页面设置为
index.html
app.get('/',(req,res)=>{
    res.sendFile(__dirname+"/"+"my1.html")        //设置/ 下访问文件位置
});

app.get("/yhz",(req,res)=>{
    res.sendFile(__dirname+"/"+"my2.html")
})

var server =app.listen(5678,()=>{
    var port =server.address().port
    console.log("访问地址http://localhost:%s",port)
})
module.exports=app;
```

将编写好的html文件以及图片资源放在同级目录下

该目录下打开命令行，输入

npm start

此时可以通过上述www中设置的express服务器ip127.6.7.8，端口4321或监听的端口5678进入

也可以直接通过本地ip进入，例如localhost:5678（为了好在抓包中进行区分，不访问这个）

如图

get传入参数/yhz，则进入my2.html界面，如图

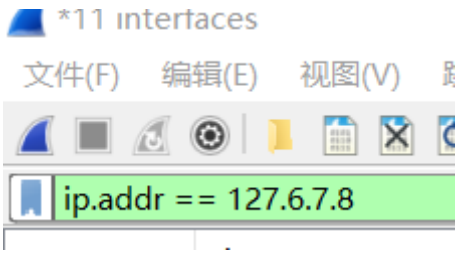
WireShark抓包

安装wireshark，官网下载安装包安装，无脑next

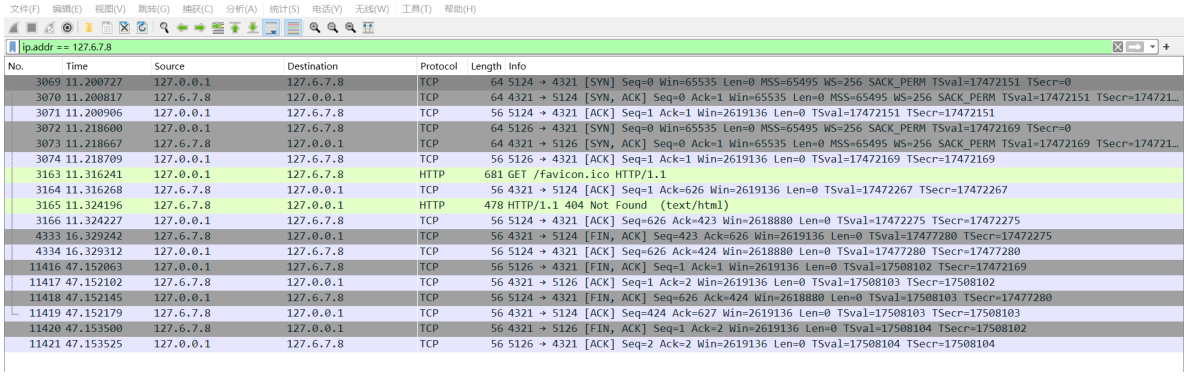
安装后进行监听，打开wireshark，监听所有端口

之后再统一执行启动本地服务器。浏览器访问服务器ip和端口（127.6.7.8:4321）、关闭服务器等操作

停止抓包后，在wireshark中输入过滤器语句，过滤只有127.6.7.8的包



捕获后大致如图



No.	Time	Source	Destination	Protocol	Length	Info
3069	11.200727	127.0.0.1	127.6.7.8	TCP	64	5124 → 4321 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM TSval=17472151 TSecr=0
3070	11.200817	127.6.7.8	127.0.0.1	TCP	64	4321 → 5124 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM TSval=17472151 TSecr=174721…
3071	11.200906	127.0.0.1	127.6.7.8	TCP	56	5124 → 4321 [ACK] Seq=1 Ack=1 Win=2619136 Len=0 TSval=17472151 TSecr=17472151
3072	11.218600	127.0.0.1	127.6.7.8	TCP	64	5126 → 4321 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM TSval=17472169 TSecr=0
3073	11.218667	127.6.7.8	127.0.0.1	TCP	64	4321 → 5126 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM TSval=17472169 TSecr=174721…
3074	11.218709	127.0.0.1	127.6.7.8	TCP	56	5126 → 4321 [ACK] Seq=1 Ack=1 Win=2619136 Len=0 TSval=17472169 TSecr=17472169
3163	11.316241	127.0.0.1	127.6.7.8	HTTP	681	GET /favicon.ico HTTP/1.1
3164	11.316268	127.6.7.8	127.0.0.1	TCP	56	4321 → 5124 [ACK] Seq=1 Ack=626 Win=2619136 Len=0 TSval=17472267 TSecr=17472267
3165	11.324196	127.6.7.8	127.0.0.1	HTTP	478	HTTP/1.1 404 Not Found (text/html)
3166	11.324227	127.0.0.1	127.6.7.8	TCP	56	5124 → 4321 [ACK] Seq=626 Ack=423 Win=2618880 Len=0 TSval=17472275 TSecr=17472275
4333	16.329242	127.6.7.8	127.0.0.1	TCP	56	4321 → 5124 [FIN, ACK] Seq=423 Ack=626 Win=2619136 Len=0 TSval=17477280 TSecr=17472275
4334	16.329312	127.0.0.1	127.6.7.8	TCP	56	5124 → 4321 [ACK] Seq=626 Ack=424 Win=2618880 Len=0 TSval=17477280 TSecr=17477280
11416	47.152063	127.0.0.1	127.6.7.8	TCP	56	5126 → 4321 [FIN, ACK] Seq=1 Ack=1 Win=2619136 Len=0 TSval=17508102 TSecr=17472169
11417	47.152102	127.6.7.8	127.0.0.1	TCP	56	4321 → 5126 [ACK] Seq=1 Ack=2 Win=2619136 Len=0 TSval=17508103 TSecr=17508102
11418	47.152145	127.0.0.1	127.6.7.8	TCP	56	5124 → 4321 [FIN, ACK] Seq=626 Ack=424 Win=2618880 Len=0 TSval=17508103 TSecr=17477280
11419	47.152179	127.6.7.8	127.0.0.1	TCP	56	4321 → 5124 [ACK] Seq=424 Ack=627 Win=2619136 Len=0 TSval=17508103 TSecr=17508103
11420	47.153500	127.6.7.8	127.0.0.1	TCP	56	4321 → 5126 [FIN, ACK] Seq=1 Ack=2 Win=2619136 Len=0 TSval=17508104 TSecr=17508102
11421	47.153525	127.0.0.1	127.6.7.8	TCP	56	5126 → 4321 [ACK] Seq=2 Ack=2 Win=2619136 Len=0 TSval=17508104 TSecr=17508104

WireShark分析

使用Edge浏览器访问Express的服务器，浏览器相当于客户端。

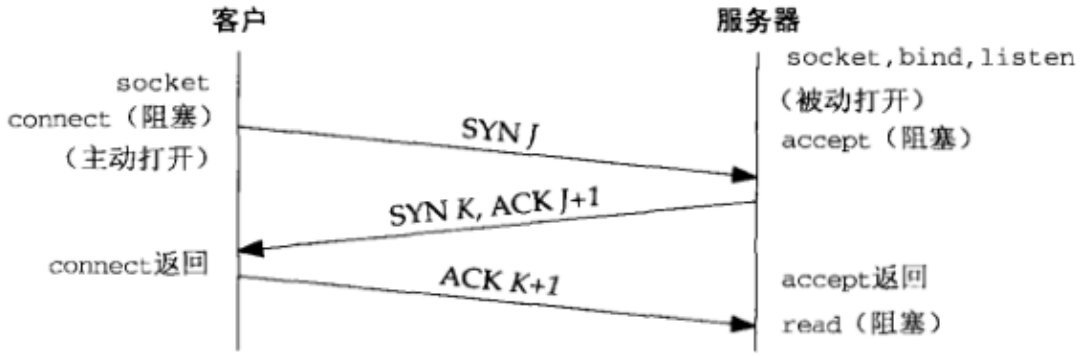
浏览器ip:127.0.0.1

服务器ip:127.6.7.8

三次握手

Protocol一项为TCP，表示使用TCP协议。上图前三行为TCP协议三次握手过程。

三次握手过程如图：



对其进行分析：

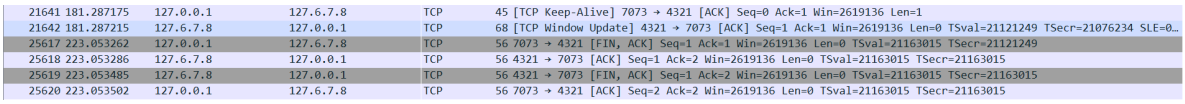
1. 浏览器想要通过TCP协议，访问127.6.7.8:4321，向服务器发出请求，第一次握手：客户端给服务端发送一个SYN（synchronize，请求同步）段，即在 TCP 标头中 SYN 位字段为 1 的 TCP/IP 数据包, 同时包含客户端初始序列号seq=j。由第一行可知，这里的seq=0。
2. 第二次握手：服务端返回客户端 一个SYN +ACK （acknowledge，确认同步）段，即 TCP 标头中 SYN和ACK位字段都为 1 的 TCP/IP 数据包。 这部分中包含了服务器的初始序列号seq=k；同时使 ack= J + 1来表示确认已收到客户端的 SYN段（seq=j）。由第二行可知，服务端127.6.7.8返回客户端[SYN,ACK]，服务器的seq=0，ack段=客户端seq+1=0+1=1。
3. 第三次握手：客户端给服务端发送一个ACK段，即在 TCP 标头中 ACK 位字段为 1的 TCP/IP 数据包。使ack=k+1（服务端的syn+1）来表示确认已收到服务器的 SYN段。服务器端发过去的syn中seq=0，那么客户端发回的ack=0+1=1，与第三行相同。

4-6行也为三次握手，但是可以看到客户端即浏览器的端口号不同，5124变为5126

Get请求

7-10行为浏览器发送一个get请求，可以看到浏览器通过HTTP协议进行get请求，请求参数为/favicon.ico。但服务器里并没有对它进行定义；因此返回一个HTTP的404。实际上不知道为什么会传入favicon.ico，一般都是传入/或/yhz，这时服务器会传回一个HTTP/1.1 304 Not Modified。查阅资料得知，客户端浏览器缓存了当前界面，第二次访问这个界面时，get页面未做修改，服务器返回HTTP/1.1 304 Not Modified，客户端从本地缓存中调取页面。

TCP keep-alive和TCP window update

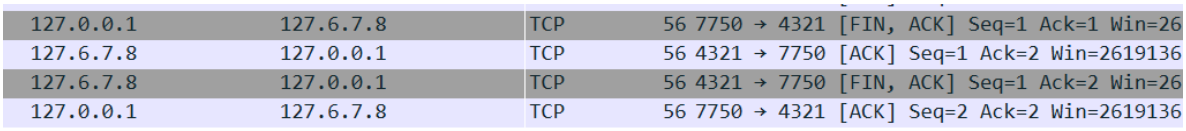


21641 181.287175	127.0.0.1	127.6.7.8	TCP	45 [TCP Keep-Alive] 7073 → 4321 [ACK] Seq=0 Ack=1 Win=2619136 Len=1
21642 181.287215	127.6.7.8	127.0.0.1	TCP	68 [TCP Window Update] 4321 → 7073 [ACK] Seq=1 Ack=1 Win=2619136 Len=0 TSval=21121249 TSecr=21076234 SLE=0…
25617 223.053262	127.0.0.1	127.6.7.8	TCP	56 7073 → 4321 [FIN, ACK] Seq=1 Ack=1 Win=2619136 Len=0 TSval=21163015 TSecr=21121249
25618 223.053286	127.6.7.8	127.0.0.1	TCP	56 4321 → 7073 [ACK] Seq=1 Ack=2 Win=2619136 Len=0 TSval=21163015 TSecr=21163015
25619 223.053485	127.6.7.8	127.0.0.1	TCP	56 4321 → 7073 [FIN, ACK] Seq=1 Ack=2 Win=2619136 Len=0 TSval=21163015 TSecr=21163015
25620 223.053502	127.0.0.1	127.6.7.8	TCP	56 7073 → 4321 [ACK] Seq=2 Ack=2 Win=2619136 Len=0 TSval=21163015 TSecr=21163015

在闲置浏览器页面一定时间后，会看到浏览器向服务端发送了一个[TCP Keep-Alive]段，它用来检测死连接。如果主机可达，服务端相应ACK应答，认为是存活的；主机可达但应用程序退出，对方发RST应答，发送TCP撤销连接；可达但程序崩溃，发送FIN；对方主机不响应，继续发送直到超时撤销连接。

TCP Window Update是零窗口检测报文，发生的原因是TCP Window为0或接近0，发送方停止发送数据，接收方Window出现空闲空间，则接收方发送TCP Window Update来更新发送方的Window

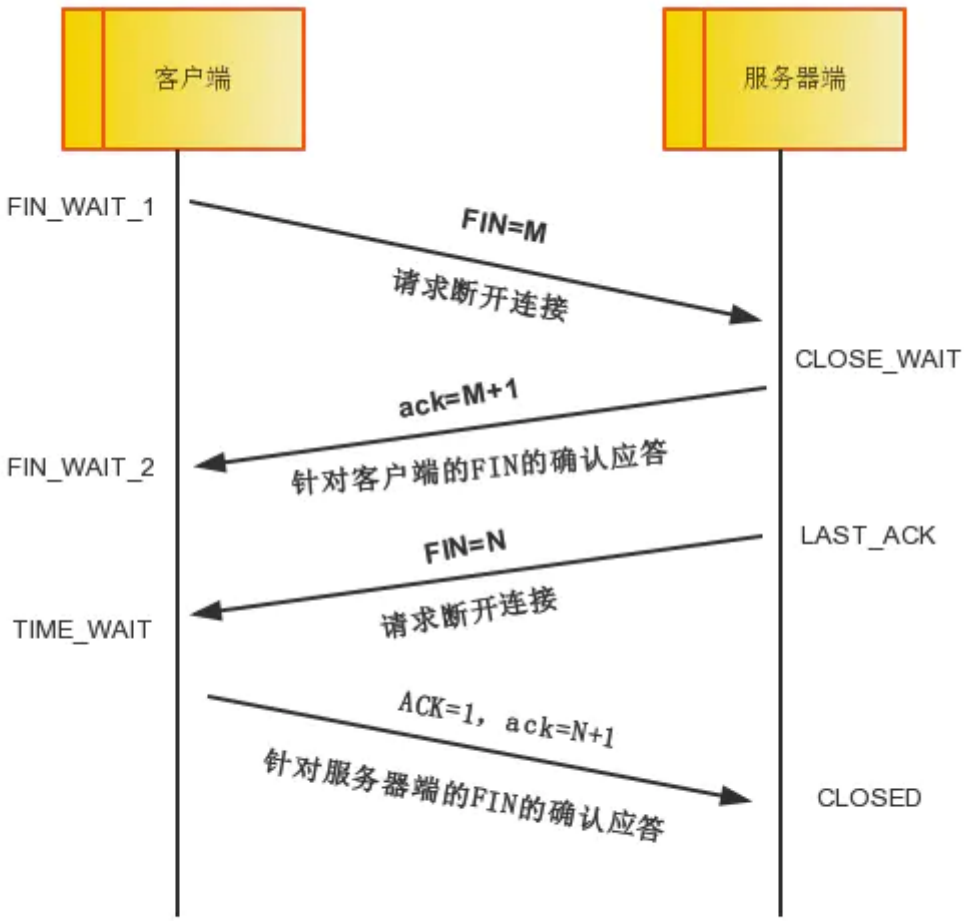
四次挥手



127.0.0.1	127.6.7.8	TCP	56 7750 → 4321 [FIN, ACK] Seq=1 Ack=1 Win=26
127.6.7.8	127.0.0.1	TCP	56 4321 → 7750 [ACK] Seq=1 Ack=2 Win=2619136
127.6.7.8	127.0.0.1	TCP	56 4321 → 7750 [FIN, ACK] Seq=1 Ack=2 Win=26
127.0.0.1	127.6.7.8	TCP	56 7750 → 4321 [ACK] Seq=2 Ack=2 Win=2619136

关闭浏览器的所有页面，能得到客户端主动断开的四次挥手报文。（这里端口跟上面不一样是因为不是对这个页面同一次访问，因此port不同）

四次挥手的过程如图：

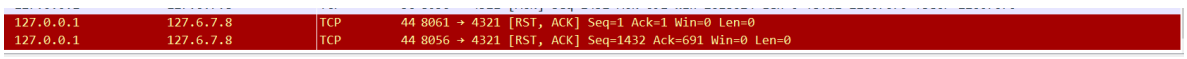


四次挥手

1. 客户端断开连接，主动请求，客户端给服务端发送FIN=M，关闭客户端到服务器端的数据传送，客户端进入FIN_WAIT_1状态。可以从图第一行看到127.0.0.1向127.6.7.8发送了[FIN,ACK]段，序列号seq=1
2. 第二次挥手：服务端收到FIN，返回一个ACK，ACK=M+1。此时客户端进入FIN_WAIT_2 状态，继续等待服务器端的FIN报文。可以看到返回的ACK=2=M+1=1+1，能对应上发送过来的FIN=1
3. 第三次挥手：服务器端确认所有剩余数据已经发送，向客户端发送FIN=N，请求断开连接。服务器端进入LAST_ACK状态。可以看到第三行服务端发回了[FIN,ACK]，seq=1，ack=2
4. 第四次挥手：客户端收到断开连接信息，向服务器端发送ACK=N+1进行应答确认，客户端进入TIME_WAIT状态，经过2*MSL（最大报文生存时间）后没有收到回复，说明服务端也正常关闭，客户端关闭。

一种其他情况

如果使用任务管理器强制关闭浏览器进程，则wireshark会显示如下报文



查阅资料得知，RST一般是在不正常情况下发送的，如connect不存在的端口、向一个关闭的连接发送数据、或向一个已经崩溃的对端发送数据（连接之前已经被建立）等。一方收到RST后，TCP socket立即关闭。