

**计算机网络**

实验报告

|  |  |
| --- | --- |
| 题目 | A3-简单Web服务器 |
| 学生姓名 | 谭哲文 |
| 学 号 | 8202191123 |
| 专业班级 | 计科2105 |
| 指导教师 | 王建新，王伟平 |
| 学 院 | 计算机学院 |
| 完成时间 | 2023-05-22 |

目录

[一、实验目的 2](#_Toc104125881)

[二、实验要求 2](#_Toc104125882)

[三、实验内容 2](#_Toc104125883)

[四、实验方案设计与实施 3](#_Toc104125884)

[1. Socket编程接口及Web服务器总体设计 3](#_Toc104125885)

[HttpServer.java 5](#_Toc104125886)

[2. HTTP传输协议 6](#_Toc104125887)

[3、服务端处理信息部分设计 7](#_Toc104125888)

[Request.java 8](#_Toc104125889)

[Response.java 9](#_Toc104125890)

[index.html 10](#_Toc104125891)

[五、实验结果展示与分析 11](#_Toc104125892)

[六、实验心得体会 12](#_Toc104125893)

# 一、实验目的

1、理解web服务的工作原理，为今后开发网站打下基础

2、掌握在Windows实现web服务

# 二、实验要求

Web 服务器的基本功能是接受并解析客户端的 HTTP 请求，然后从服务器的文件系统获取所请求的文件，生成一个由头部和响应文件内容所构成成的 HTTP 响应消息，并将该响应消息发送给客户端。如果请求的文件不存在于服务器中，则服务器应该向客户端发送“404 Not Found”差错报文。 具体的过程和步骤分为：

当一个客户（浏览器）连接时，创建一个连接套接字；

1、从这个连接套接字接收 HTTP 请求；

2、解释该请求以确定所请求的特定文件；

3、从服务器的文件系统获得请求的文件；

4、创建一个由请求的文件组成的 HTTP 响应报文，报文前面有首部行；

5、经 TCP 连接向请求浏览器发送响应；

6、如果浏览器请求一个在该服务器中不存在的文件，服务器应当返回一个“404 Not Found”差错报文。

# 三、实验内容

整个Web服务器设计可分为以下几个部分：

1、初始化套接字，为TCP连接的建立做准备,服务器首先启动，根据定义的端口号与套接字进行绑定。

2、启动监听，接受客户端请求，建立TCP连接即建立套接字，接受客户HTTP请求报文

3、根据套接字获取输入输出流，从而实现接收报文。

4、根据请求报文进行解析

5、关闭输入流、输出流、套接字，完成此次请求处理。

# 四、实验方案设计与实施

## 1. Socket编程接口及Web服务器总体设计

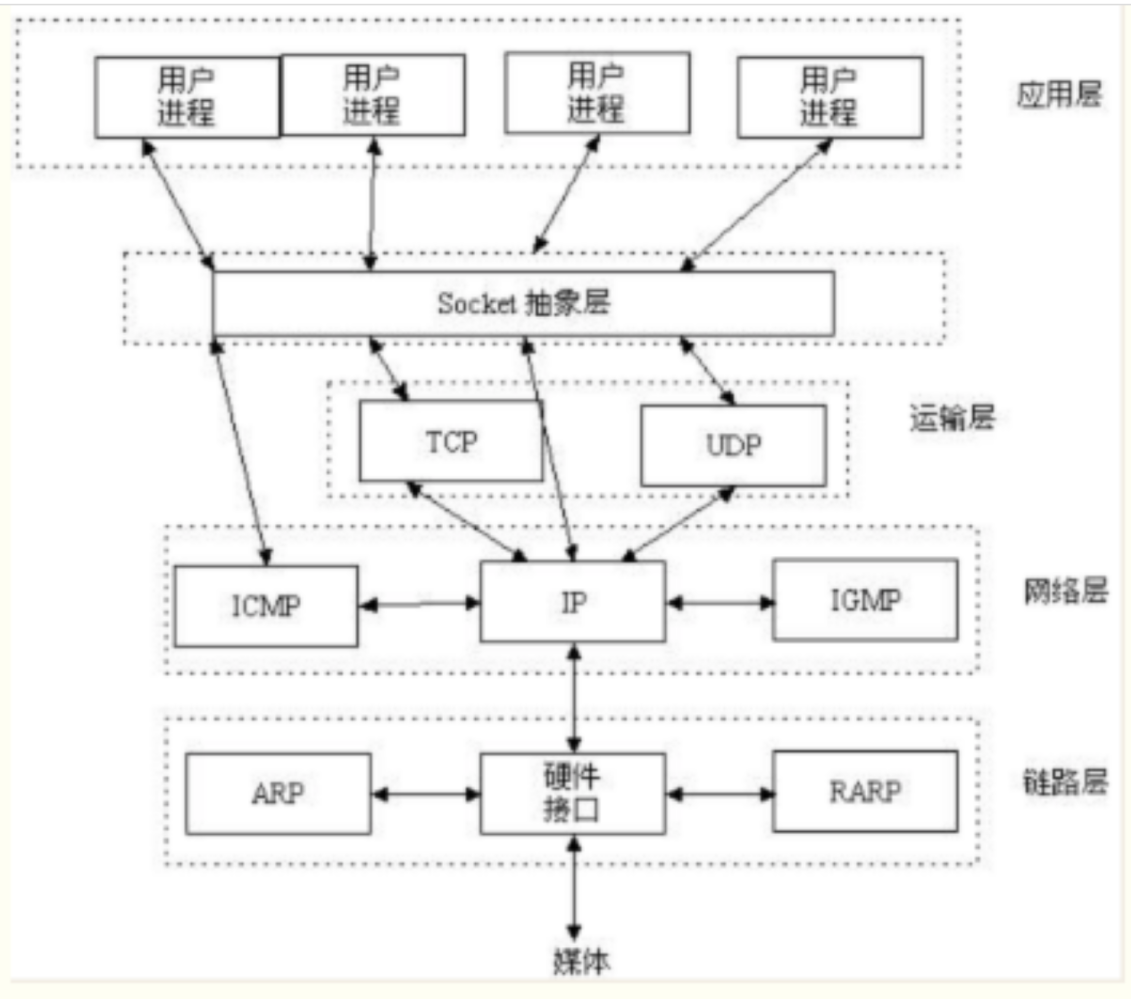
要实现 Web 服务器，需使用套接字 Socket编程接口来使用操作系统提供的网络通信功能。 Socket 是应用层与 TCP/IP 协议族通信的中间软件抽象层，是一组编程接口。它把复杂的 TCP/IP 协议族隐藏在 Socket 接口后面，对用户来说，一组简单的接口就是全部，让 Socket 去组织数据，以符合指定的协议。使用 Socket 后，无需深入理解 TCP/UDP 协议细节（因为Socket 已经为我们封装好了），只需要遵循 Socket 的规定去编程，写出的程序自然就是遵循 TCP/UDP 标准的。Socket 的地位如下图所示：

图4.1 TCP协议在协议栈中的地位

从某种意义上说，Socket 由地址IP和端口Port构成。IP 是用来标识互联网中的一台主机的位置，而 Port 是用来标识这台机器上的一个应用程序，IP 地址是配置到网卡上的，而 Port 是应用程序开启的，IP 与 Port 的绑定就标识了互联网中独一无二的一个应用程序。

套接字类型 流式套接字（SOCK\_STREAM）：用于提供面向连接、可靠的数据传输服务。 数据报套接字（SOCK\_DGRAM）：提供了一种无连接的服务。该服务并不能保证数据传输的可靠性，数据有可能在传输过程中丢失或出现数据重复，且无法保证顺序地接收到数据。 原始套接字（SOCK\_RAW）：主要用于实现自定义协议或底层网络协议。

在本 WEB 服务器程序实验中，采用流式套接字进行通信。其基本模型如下图所示：

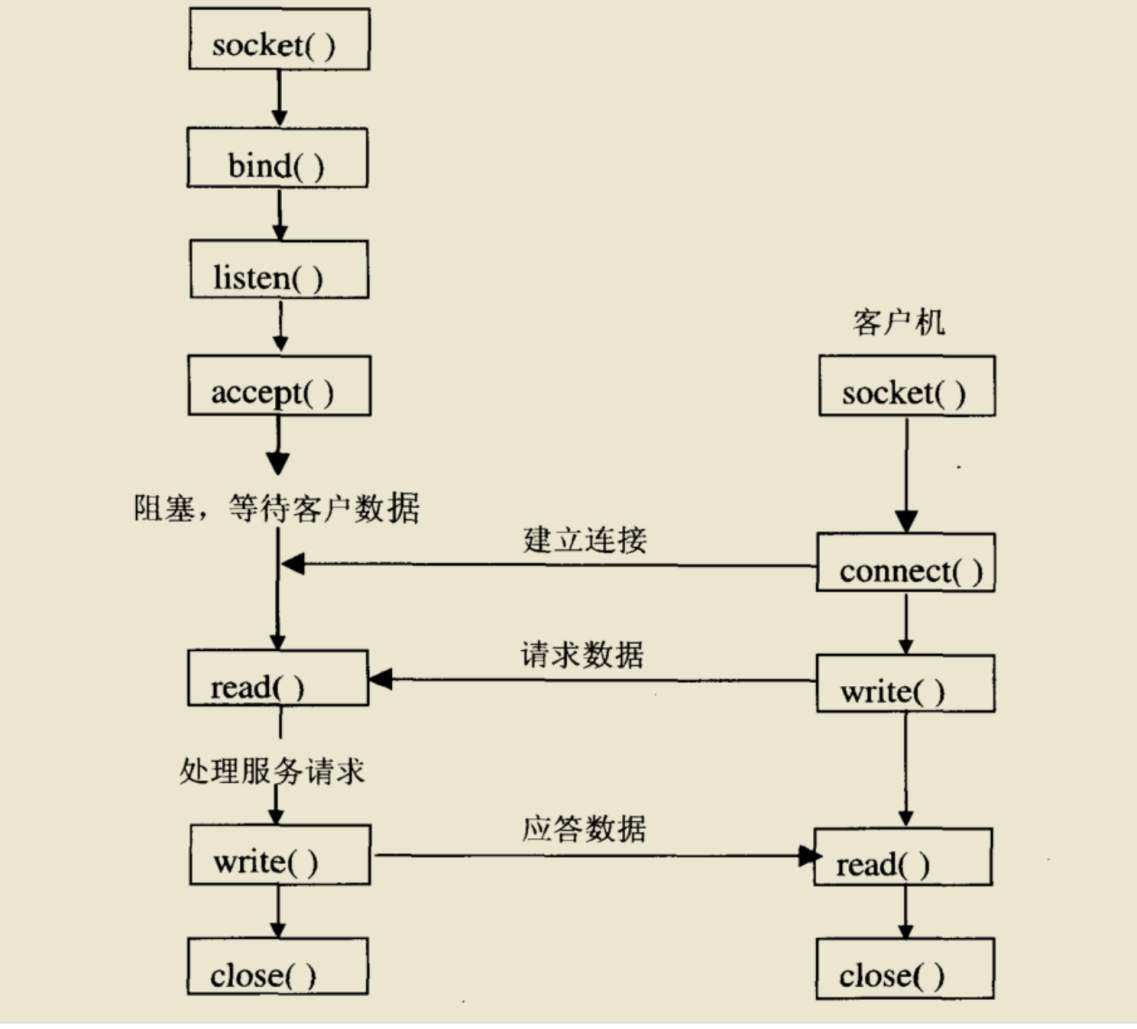


图4.2 Web服务器工作流程图

其工作过程如下：服务器首先启动，通过调用 socket() 建立一个套接字，然后调用绑定方法 bind() 将该套接字和本地网络地址联系在一起，再调用 listen() 使套接字做好侦听连接的准备，并设定的连接队列的长度。客户端在建立套接字后，就可调用连接方法 connect() 向服务器端提出连接请求。服务器端在监听到连接请求后，建立和该客户端的连接，并放入连接队列中，并通过调用 accept() 来返回该连接，以便后面通信使用。客户端和服务器连接一旦建立，就可以通过调用接收方法 recv()／recvfrom() 和发送 方法send()／sendto() 来发送和接收数据。最后，待数据传送结束后，双方调用 close() 关闭套接字。

### HttpServer.java

|  |
| --- |
| import java.net.ServerSocket;  import java.net.Socket;  import java.io.IOException;  import java.io.InputStream;  import java.io.OutputStream;  public class HttpServer {      public static void main(String[] args) {          ServerSocket serverSocket = null;          try {              // 创建服务器套接字              serverSocket = new ServerSocket(8080);          } catch (IOException ioe) {              ioe.printStackTrace();              System.exit(0);          }          Socket socket = null;          InputStream inputStream = null;          OutputStream outputStream = null;          while (true) {              try {                  // 接受套接字请求并创建相应的服务器套接字来启动监听                  socket = serverSocket.accept();                  // 获取输入流                  inputStream = socket.getInputStream();                  // 获取输出流                  outputStream = socket.getOutputStream();                  // 实例化请求类                  Request request = new Request(inputStream);                  // 实例化应答类                  Response response = new Response(outputStream, request);                  // 解析请求                  request.parse();                  // 根据请求做响应应答                  response.sendStaticSourse();                  // 关闭套接字                  socket.close();              } catch (IOException ioe) {                  ioe.printStackTrace();              }          }      }  } |

## 2. HTTP传输协议

超文本传输协议（HTTP）是用于Web上进行通信的协议：它定义Web浏览器如何从Web服务器请求资源以及服务器如何响应。为简单起见，在该实验中将处理HTTP协议的1.0版。HTTP通信以事务形式进行，其中事务由客户端向服务器发送请求，然后读取响应组成。 请求和响应消息共享一个通用的基本格式：

* 初始行（请求或响应行）
* 零个或多个头部行
* 空行（CRLF）
* 可选消息正文。

对于大多数常见的HTTP事务，协议归结为一系列相对简单的步骤：

首先，客户端创建到服务器的连接；然后客户端通过向服务器发送一行文本来发出请求。这请求行包HTTP方法(比如GET，POST、PUT等)，请求URI(类似于URL)，以及客户机希望使用的协议版本(比如HTTP/1.0)；接着，服务器发送响应消息，其初始行由状态线（指示请求是否成功)，响应状态码(指示请求是否成功完成的数值)，以及推理短语(一种提供状态代码描述的英文消息组成)；最后一旦服务器将响应返回给客户端，它就会关闭连接。

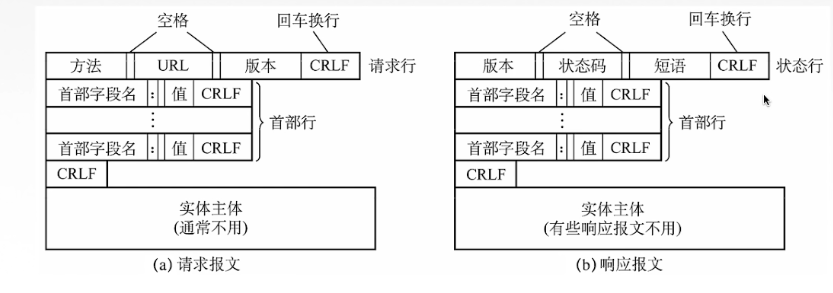


图4.3 HTTP传输协议请求报文格式

## 3、服务端处理信息部分设计

（1）处理HTTP请求报文的请求行，提取“请求方法”、“URL”、“HTTP版本“三个关键要素；

（2）判断处理“请求方法”，根据判断处理的结果，构造相应的响应报文，反馈相关给客户端，并且在服务器端打印反馈的结果。

（3）判断处理“URL”，根据判断处理的结果，构造相应的响应报文，反馈相关给客户端，并且在服务器端打印反馈的结果。比如如果请求的文件存在，之后发送文件数据。如果请求的文件不存在，则发送“文件不存在”相关信息。

### Request.java

|  |
| --- |
| import java.io.InputStream;    public class Request {      private InputStream input;        private String uri;        public Request(InputStream input){          this.input=input;      }        public void parse(){          //Read a set of characters from the socket          StringBuffer request=new StringBuffer(2048);          int i;          byte[] buffer=new byte[2048];          try {              i=input.read(buffer);          } catch (Exception e) {              e.printStackTrace();              i=-1;          }          for(int j=0;j<i;j++){              request.append((char)buffer[j]);          }          System.out.print(request.toString());          uri=parseUri(request.toString());      }        public String parseUri(String requestString){          int index1,index2;          index1=requestString.indexOf(" ");          if(index1!=-1){              index2=requestString.indexOf(" ",index1+1);              if(index2>index1){                  return requestString.substring(index1+2,index2);              }          }          return null;      }        public String getUri(){          return this.uri;      }  } |

### Response.java

|  |
| --- |
| import java.io.OutputStream;  import java.io.BufferedInputStream;  import java.io.BufferedOutputStream;  import java.io.File;  import java.io.FileInputStream;  import java.io.FileNotFoundException;  import java.io.IOException;  public class Response {      private OutputStream outputStream;      private Request request;      public Response(OutputStream outputStream, Request request) {          this.outputStream = outputStream;          this.request = request;      }      public void sendStaticSourse() {          BufferedOutputStream bos = new BufferedOutputStream(outputStream);          BufferedInputStream bis = null;          try {              // 如果请求的文件存在              if (new File(this.request.getUri()).exists()) {                  try {                      // 获取文件输入流                      bis = new BufferedInputStream(new FileInputStream(this.request.getUri()));                  } catch (FileNotFoundException fnfe) {                  }                  byte[] data = new byte[1024];                  int length = 0;                  // 发送可以发送文件的报文                  String msg = "HTTP/1.1 200 OK \r\n" +                          "content-type: text/html; charset=utf-8 \r\n\r\n";                  // 发送文件数据                  bos.write(msg.getBytes());                  while ((length = bis.read(data)) != -1) {                      bos.write(data, 0, length);                  }              } else {                  // 发送未找到文件报文                  String errormsg = "HTTP/1.1 404 File Not Found\r\n" +                          "Content-Type:text/html\r\n" +                          "Content-Length:23\r\n" +                          "\r\n" +                          "<h2>File Not Found</h2>";                  bos.write(errormsg.getBytes());              }              // 关闭输出流              bos.close();              // 关闭输入流              if (bis != null) {                  bis.close();              }          } catch (IOException ioe) {              ioe.printStackTrace();          }      }  } |

### index.html

|  |
| --- |
| <html>  <head>        <link href="favicon.ico" rel="shortcut icon">        <meta content="text/html" charset="UTF-8">        <title>谭哲文web服务器</title>        </style>  </head>  <body background="1.jpeg" style="background-repeat:no-repeat        background-attachment fixed;        background-size:100% 100%; ">        <div style="left: 40%; position: absolute; top:20%;font-weight:bold;font-size:30px"> web服务器 实验界面：</div>        <div style="left: 45%; position: absolute; top:40%; color: red;font-weight:bold;font-size:24px">姓名：谭哲文</div>        <div style="left: 45%; position: absolute; top:60%; color: blue;font-weight:bold;font-size:18px">班级：计科2105班        </div>        <div style="left: 45%; position: absolute; top:80%;color:blue;font-weight:bold;font-size:18px">学号：8202191123        </div>  <img src="https://ts4.cn.mm.bing.net/th?id=OIP-C.B3ghsniJ91U5MMBLNsSmnAHaFT&w=295&h=211&c=8&rs=1&qlt=90&o=6&dpr=1.5&pid=3.1&rm=2" width="450px">  </body>  </html> |

# 五、实验结果展示与分析

实验操作系统:Windows 10

实验工具：VSCode

实验结果展示：



图5.5 浏览器请求index.html

GET /C:/Users/Dell/Desktop/jiwang/web/index.html HTTP/1.1

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.7

Accept-Encoding: gzip, deflate, br

Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6

Connection: keep-alive

Host: localhost:8080

Sec-Fetch-Dest: document

Sec-Fetch-Mode: navigate

Sec-Fetch-Site: none

Sec-Fetch-User: ?1

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36 Edg/113.0.1774.50

sec-ch-ua: "Microsoft Edge";v="113", "Chromium";v="113", "Not-A.Brand";v="24"

sec-ch-ua-mobile: ?0

sec-ch-ua-platform: "Windows"

5.2 浏览器请求index.html的请求报文

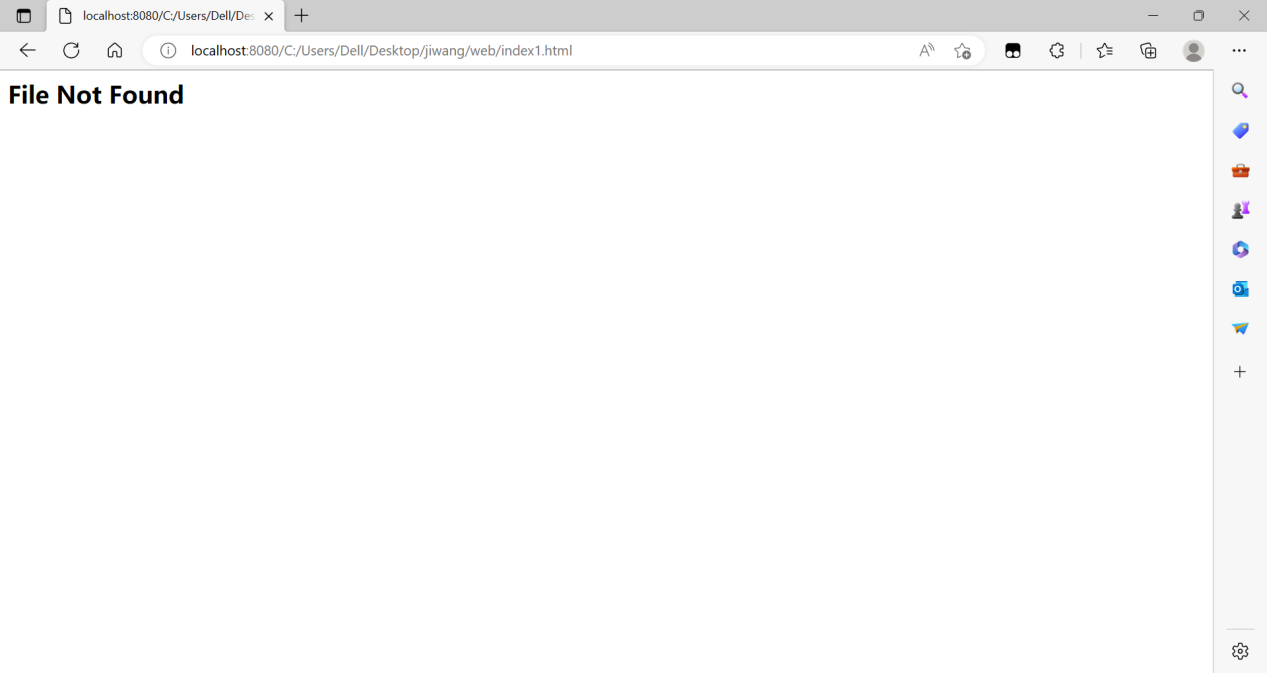


图5.3 浏览器请求不存在的文件

# 六、实验心得体会

本次实验我通过TCP协议、http协议和java网络编程的相关知识实现了简单的Web服务器，这个Web服务器能够实现初步的解析请求和响应，我也因此得到了对理论知识进行实践的机会，这使我对B/S结构有了更深的理解。但我认为这与内容丰富、功能多样、界面美观之间还有较大差距，Web服务器也应该能够实现更多的功能。因此，在下步学习中，我也应该更广泛、更深入地学习相关知识，并在以后继续对此次实验进行完善。