Linux 下 HTTP服务器设计毕业论文

目录

摘要

Linux 操作系统是一个开放源代码的免费操作系统。它不仅有安全、稳定、成本低的特点，而且很少发现有病毒传播。 HTTP服务器是 web服务器的一种，它是基于超文本传输协议 HTTP的服务器。基于 Linux 具有稳定、可靠、安全和强大的网络功能这些优点， 使得其主要应用于服务器领域。 所以本文选择在 Linux 环境下实现一个HTTP服务器。

本文研究了 Linux 下 HTTP服务器的设计与实现。 在 Linux 系统中采用 HTTP协议和浏览器完成数据的传输。阐述了 Linux 套接字编程的方法、 EPOLL等 I/O 复用编程模型。详细分析了 HTTP协议内容以及客户端与服务器之间的通信过程。本文实现了客户端浏览器和服务器端以 HTTP协议进行请求和响应的功能。同时对服务器进行了一个简单的压力测试。所有程序代码均为 Linux 下的 C语言编程。

关键字： Linux、HTTP 服务器、 HTTP 协议、 EPOLL

Abstract

The Linux operate system is a free operate system which opens a source code. Not only it has characteristics such as safe , stability,and the low cost,but also it seldom disseminates the Virus . HTTPserver is one of the Web servers and it bases on HTTP protocol . As the Linux operating system has the function of stable , reliable , safe and powerful network, it mainly used in servers.To realize a HTTPserver in the Linux environment is the best choice .

This paper introduces design and implement of HTTP server in Linux operating system. In the Linux system and browser used HTTPprotocol for data transmission. This paper expounds the method of Linux socket programming and EPOLL I/O multiplexing programming model.Detailed analysis the communication process between client and server and HTTP protocol.This paper realizes the function that the client browser requests and the server responds by HTTP agreement.Make a simple pressure test on the server.All program code use the C language programming in Linux operating system.

Keyword: Linux, HTTP Server, HTTP protocol, EPOLL

前言

随着 Internet 的迅速发展与普及，网络已经延伸到世界的各个角落。在该技术基础上发展起来的 www 通过超文本向用户提供全方位的多媒体、超媒体信息，从而为全世界的 Internet 用户提供了一种获取信息、共享资源的途径。由于用户在通过Web浏览器访问信息资源的过程中， 无需再关心一些技术性的细节， 而且界面非常友好，因而 Web 在 Internet 上一推出就受到了热烈的欢迎，走红全球，并迅速得到了爆炸性的发展。 所以 Web服务器在网络中的地位日益重要。当今社会中已有了许多知名的商用服务器，如 Microsoft IIS 、IBM WebSphere、BEAWebLogic、Apache、Tomcat 等。但往往这些功能强大的服务器其结构也相当复杂，规模较大，在一些特定应用情景下 （如嵌入式设备） 就不太适合了。 同时一个简单小巧的服务器也有利于我们学习网络编程的相关知识，对服务器原理也能有进一步的了解。

本文设计实现了一个功能简单、结构小巧的 HTTP服务器，采用 EPOLL多路 I/O 复用机制来实现并发服务。网络编程采用 socket ，服务器端创建套接字、绑定套接口、设置套接口为监听模式， 将该监听套接字加入 EPOLL事件列表， 然后无限循环等待 EPOLL返回， 对返回事件的套接字进行读或写的处理。 若为新连接， 则将其加入到EPOLL事件列表；若为已有连接则读取其请求或向其发送响应；若客户端已断开或已发送完响应， 服务器端就断开该连接， 并将该套接字从 EPOLL事件列表中移除。 主要提供对静态请求的处理，解析客户端请求报文，回送请求的文件和响应报文的功能。

第一章　绪　论

1.1 课题背景

随着 Internet 的迅速发展与普及，网络已经延伸到世界的各个角落。在该技术基础上发展起来的 www 通过超文本向用户提供全方位的多媒体、超媒体信息，从而为全世界的 Internet 用户提供了一种获取信息、共享资源的途径。随着计算机网络技术的发展，客户/服务器(Client/Server) 结构逐渐向浏览器/服务器(Browser/Server) 结构迁移，B/S 方式已成为一种时尚，大部分网络应用系统都是以这种 B/S 方式与网络用户交换信息。B/S 的基础是客户端要有一个浏览器程序，服务器端要有一个与之对应的 Web服务器。所以，Web服务器在 B/S 方式下起着决定性的作用，且其应用地位日益重要。

Linux 系统凭借其开源、稳定、高效的特点，在服务器市场拥有较大的市场份额。Linux 作为网络服务器市场的佼佼者，网络服务应用是其精华与核心。

当前商用的服务器大都规模比较大，功能强大的同时结构也比较复杂。自己实现一个简单小巧的 HTTP服务器，对于理解服务器工作原理知识，或针对一些特定情景下的 HTTP服务器应用来说，不失为一种比较好的方式。

1.2　课题研究的目的和意义

随着 Internet 的普及，网络已经深入到了我们的生活，跟我们息息相关。Linux 系统作为网络应用的重要平台，如何更好的去学习和使用 Linux 系统便是我们要面对的问题。同时，作为当今互联网最主要的应用——www服务，其为全世界的 Internet 用户提供了一种获取信息、共享资源的途径。www服务主要的提供者就是其后端的 HTTP服务器，所以如何更为高效的研究和学习 Web服务器的相关知识，也是我们必须要关注的。

本文在 Linux 环境下设计和实现了一个简单的 HTTP服务器。使用 socket 库完成网络底层的通信，使用 HTTP协议来和客户端进行数据传输，使用 EPOLL多路 I/O 复

用机制来完成并发服务，　最后还进行了一个简单的并发性能测试。　通过对此课题的研究学习，我们能更好的掌握　Linux 系统的使用，加深对　HTTP协议的理解。同时能对服务器设计的相关理论和实践有一定了解。

第二章 HTTP 服务器的相关理论基础

本章主要介绍设计 HTTP服务器的相关理论知识。包括 Linux 系统简介、 TCP/IP 协议分析、 Linux 下网络编程介绍、 HTTP协议分析。

2.1 Linux 系统简介

简单地说， Linux 是一套免费使用和自由传播的类 Unix 操作系统，它主要用于基于 x86 系列 CPU的计算机上。这个系统是由世界各地的成千上万的程序员设计和实现的。其目的是建立不受任何商品化软件的版权制约的、 全世界都能自由使用的 Unix 兼容产品。

Linux 以它的高效性和灵活性著称。 Linux 模块化的设计结构，使得它既能在价格昂贵的工作站上运行， 也能够在廉价的 PC机上实现全部的 Unix 特性，具有多任务、多用户的能力。 Linux 是在 GNU(GNUs Not Unix) 公共许可权限下免费获得的，是一个符合 POSIX标准的操作系统。 Linux 操作系统软件包不仅包括完整的 Linux 操作系统，而且还包括了文本编辑器、 高级语言编译器等应用软件。 它还包括带有多个窗口管理器的 X. Windows图形用户界面，如同我们使用 WindowsNT一样，允许我们使用窗口、图标和菜单对系统进行操作。 Linux 具有 Unix 的优点：稳定、可靠、安全，有强大 的网 络功 能。 在相关 软件 的支 持下 ，可实 现 WWW FTP(File Transfer Protoc01) 、DNS(Domain Name System) DHCP((Dynamic Host Configure Protocol ，动态主机配置协议)、Email 等服务， 还可作为路由器使用， 利用 ipchains/iptables 可构建 NAT(Network Address Translation ，网络地址转换) 及功能全面的防火墙。

现在， Linux 已经成为了一种受到广泛关注和支持的操作系统。包括国际商用机器公司和惠普、戴尔在内的一些计算机业巨头也陆续支持 Linux ，并且成立了一些组织支持其发展， 如 OpenInvention Network（OIN)（成员有 IBM，索尼，NEC Philips ，Novell ， Red hat 等）购买了微软专利，允许任何个体以开放的原则使用。很多人认为，和微软 Windows相比，作为自由软件的 Linux 具有低软件成本，高安全性，更加可信赖等优势，但是同时却需要更多的人力成本。

2.2 TCP/IP 协议分析

由于当今世界上的绝大部分网络程序都是建立 TCP/IP（传输控制协议 / 网际协议）协议的基础上的。所以有必要对 TCP/IP 的协议内容有所了解。这一节主要对 TCP/IP 协议进行分析。

2.2.1 TCP/IP 协议概述

TCP/IP 协议并不完全符合 OSI 的七层参考模型。传统的开放式系统互连参考模型，是一种通信协议的 7 层抽象的参考模型，其中每一层执行某一特定任务。该模型的目的是使各种硬件在相同的层次上相互通信。这 7 层是：物理层、数据链路层、网络层、传输层、会话层、表示层和应用层。而 TCP/IP 通讯协议采用了 4 层的层级结构，每一层都呼叫它的下一层所提供的网络来完成自己的需求。这 4 层分别为：

应用层：应用程序间沟通的层，如简单电子邮件传输（SMTP）、文件传输协议（FTP）、网络远程访问协议（Telnet）等。

传输层：在此层中，它提供了节点间的数据传送，应用程序之间的通信服务，主要功能是数据格式化、数据确认和丢失重传等。如传输控制协议（TCP）、用户数据报协议（UDP）等，TCP和 UDP给数据包加入传输数据并把它传输到下一层中，这一层负责传送数据，并且确定数据已被送达并接收。

互连网络层：负责提供基本的数据封包传送功能，让每一块数据包都能够到达目的主机（但不检查是否被正确接收），如网际协议（IP）。

链路接口层：接收 IP 数据报并进行传输，从网络上接收物理帧，抽取 IP 数据报转交给下一层，对实际的网络媒体的管理，定义如何使用实际网络（如 Ethernet 、Serial Line 等）来传送数据。

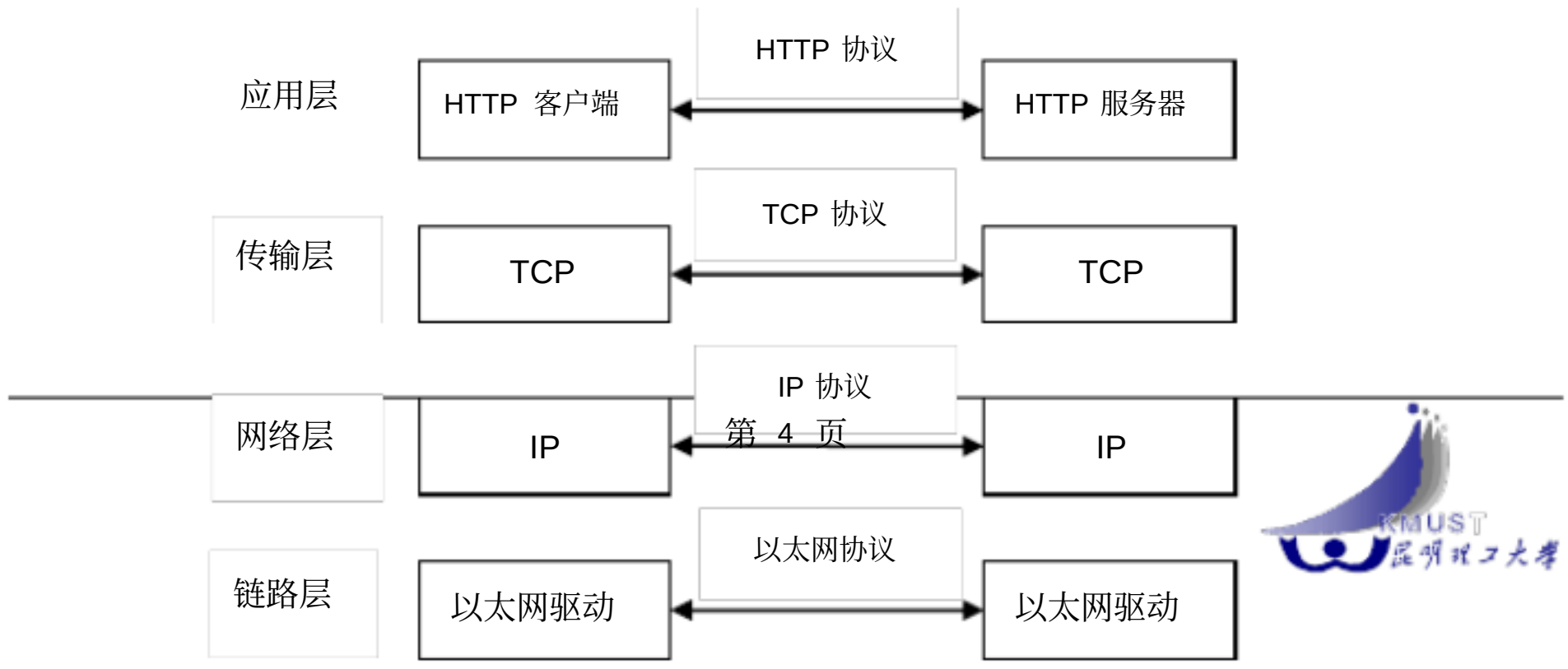


图 2-1 TCP/IP 层次结构图

2.2.2 网络层协议（ IP 协议）

IP 协议是网络层的主要协议，是 Internet 最重要的协议。在 IP 协议中规定了在 Internet 上进行通信时应遵守的规则。例如 IP 数据包的组成、路由器如何将 IP 数据包送到目的主机等。 IP 协议在主机和网络之间寻址和路由数据包。 IP 是一个无连接的协议， 主要负责在主机间寻址并为数据包设定路由， 在交换数据前它并不建立会话。因为它不保证正确传递。另一方面，数据在被收到时， IP 不需要收到确认，所以它是不可靠的。

IP 层接收由更低层（网络接口层例如以太网设备驱动程序）发来的数据包，并把该数据包发送到更高层——TCP或 UDP层；相反，IP 层也把从 TCP或 UDP层接收来的数据包传送到更低层。 IP 数据包中含有发送它的主机的地址（源地址）和接收它的主机的地址（目的地址） 。

IP 协议的数据格式如下：

版本号（4）	IHL（4）	服务类型（ 8）	数据包长度（ 16）	
标识（ 16）			Flag(3)	偏移量（ 13）
生存时间（ 8）		传输协议（ 8）	校验和（ 16）	
源地址（ 32）				
目的地址（ 32）				
选项（ 8） + 填充				
数据				

图 2-2 网际协议 IP 数据格式

版本号：协议的版本号，不同版本的协议格式或语言可能不同，现在常用的是IPV4。

生存时间（ Time To Live ， TTL）：8bit ， 即 IP 分组在 IP 网络中的寿命。

协议（Protocol ）:8bit ，指明 IP 分组中数据字段携带的是哪种高层协议的数据。

首部检查和（ header checksum）：16bit 。此字段只用于检查 IP 分组的首部，不包括数据字段。

源 IP 地址（ source IP address ）：32bit ， 填入源主机的 IP 地址。

目标 IP 地址（ destination IP address ）：32bit ， 填入目标主机的 IP 地址。

可选字段（ IP options ）：可选，可变长， 1 字节－ 40 字节，但加上填充字段（填充 0）后两个字段长度必须为 4Bytes 的整数倍。

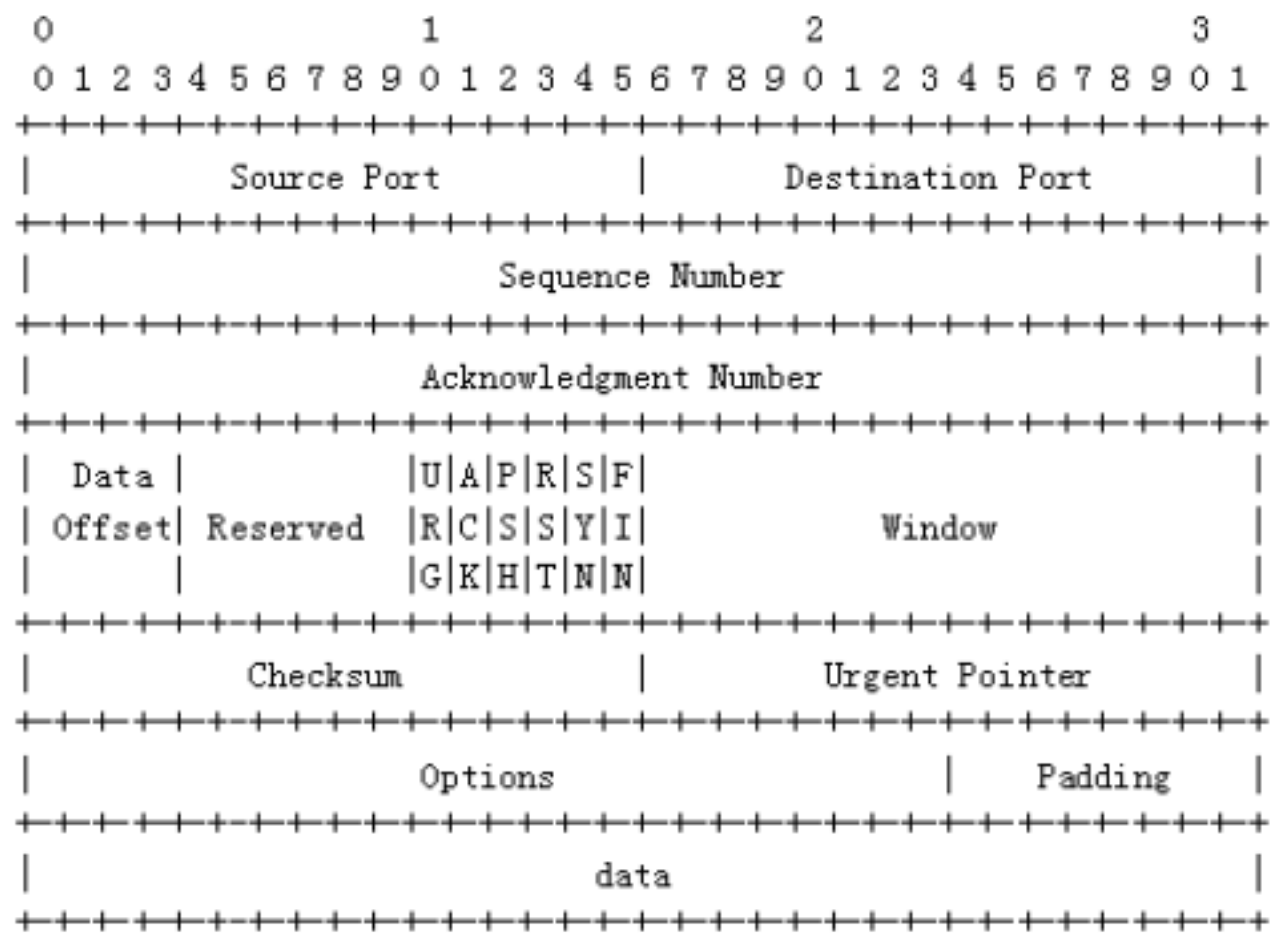
IP 地址标识着网络中一个系统的位置。 我们知道每个 IP 地址都是由两部分组成的：网络号和主机号。 其中网络号标识一个物理的网络， 同一个网络上所有主机需要同一个网络号， 该号在互联网中是唯一的； 而主机号确定网络中的一个工作端、 服务器、路由器及其它 TCP/IP 客户端。对于同一个网络号来说，主机号是唯一的。每个 TCP/IP 主机由一个逻辑 IP 地址确定。

2.2.3 传输层协议（ TCP 和 UDP）

1.TCP 协议

传输控制协议 TCP是一种面向连接（连接导向）的、可靠的、基于字节流的运输层通信协议，由 IETF 的 RFC 793说明。它在传送数据时是分段进行的，主机交换数据必须建立一个会话。它用比特流通信，即数据被作为无结构的字节流。通过每个TCP传输的字段指定顺序号，以获得可靠性。如果一个分段被分解成几个小段，接收主机会知道是否所有小段都已收到。通过发送应答，用以确认别的主机收到了数据。对于发送的每一个小段， 接收主机必须在一个指定的时间返回一个确认。 如果发送者未收到确认，数据会被重新发送；如果收到的数据段损坏，接收主机会舍弃它，因为确认未被发送，发送者会重新发送分段。 TCP端口为信息的传送指定端口，端口号小于 256 的定义为常用端口。

下图展示了 TCP首部的数据格式。如果不计任选 (Options) 字段，那么，它的大小是 20 个字节。



TCP 包头格式

图 2-3 TCP 包头格式

TCP 协议通过三个报文段完成连接的建立，这个过程称为三次握手 (three-way handshake)，过程如下图所示。

(1) 客户机向服务器发送一个 TCP数据包，表示请求建立连接。

(2) 服务器收到了数据包，知道这是一个建立请求的连接，服务器也通过发回具有以下项目的数据包表示回复： 同步标志置位、 即将发送的数据段的起始字节的顺序号、应答并带有将收到的下一个数据段的字节顺序号。

(3) 客户机收到了服务器的 TCP，知道是从服务器来的确认信息。于是客户机也向服务器发送确认信息。至此客户端完成连接。

(4) 服务器收到确认信息，也完成连接。

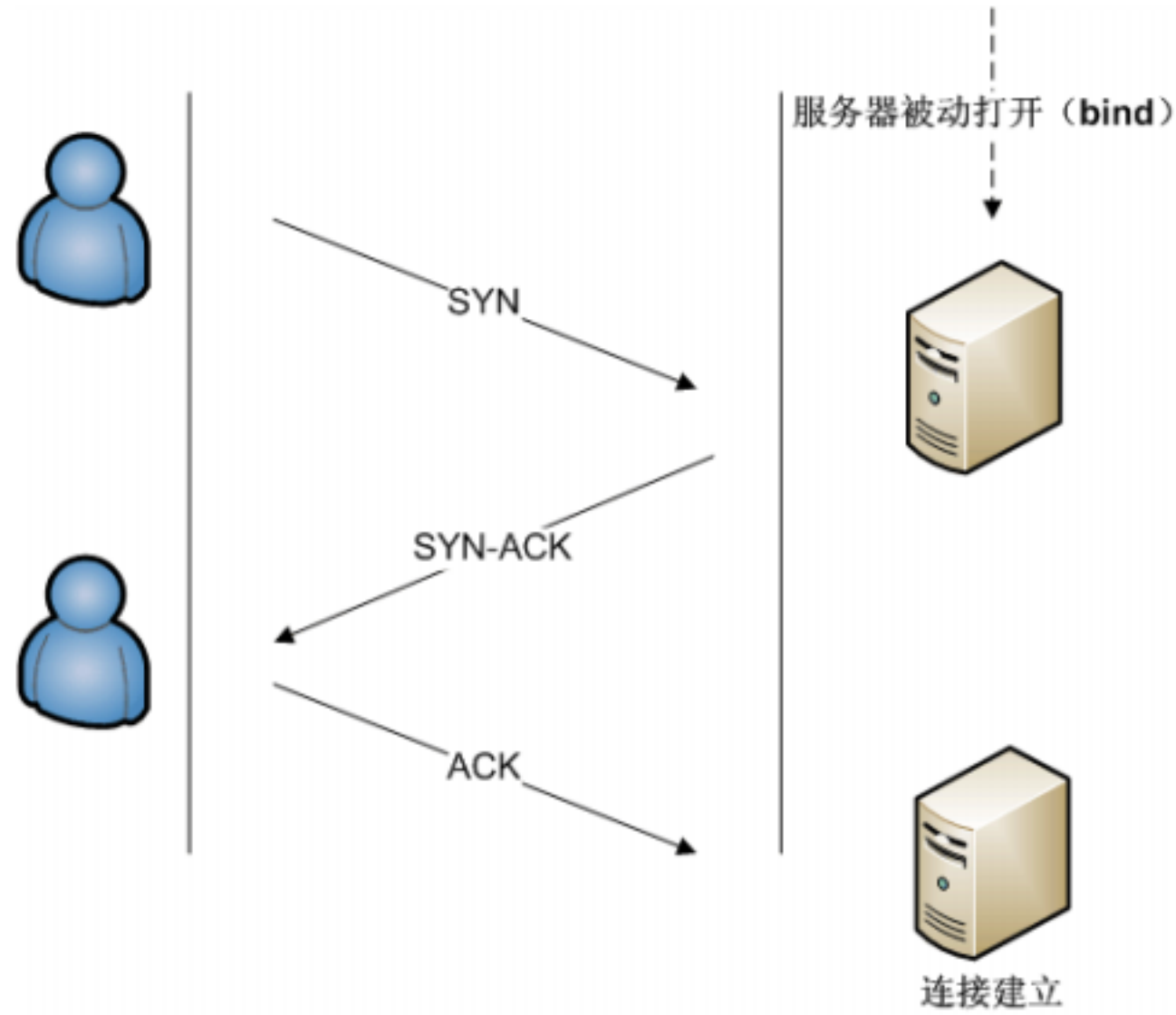


图 2-4 TCP 建立连接（三次握手）

TCP协议建立一个连接需要三次握手，而终止一个连接要经过四次握手，这是由TCP的半关闭 (half-close) 造成的。具体过程如下图所示。

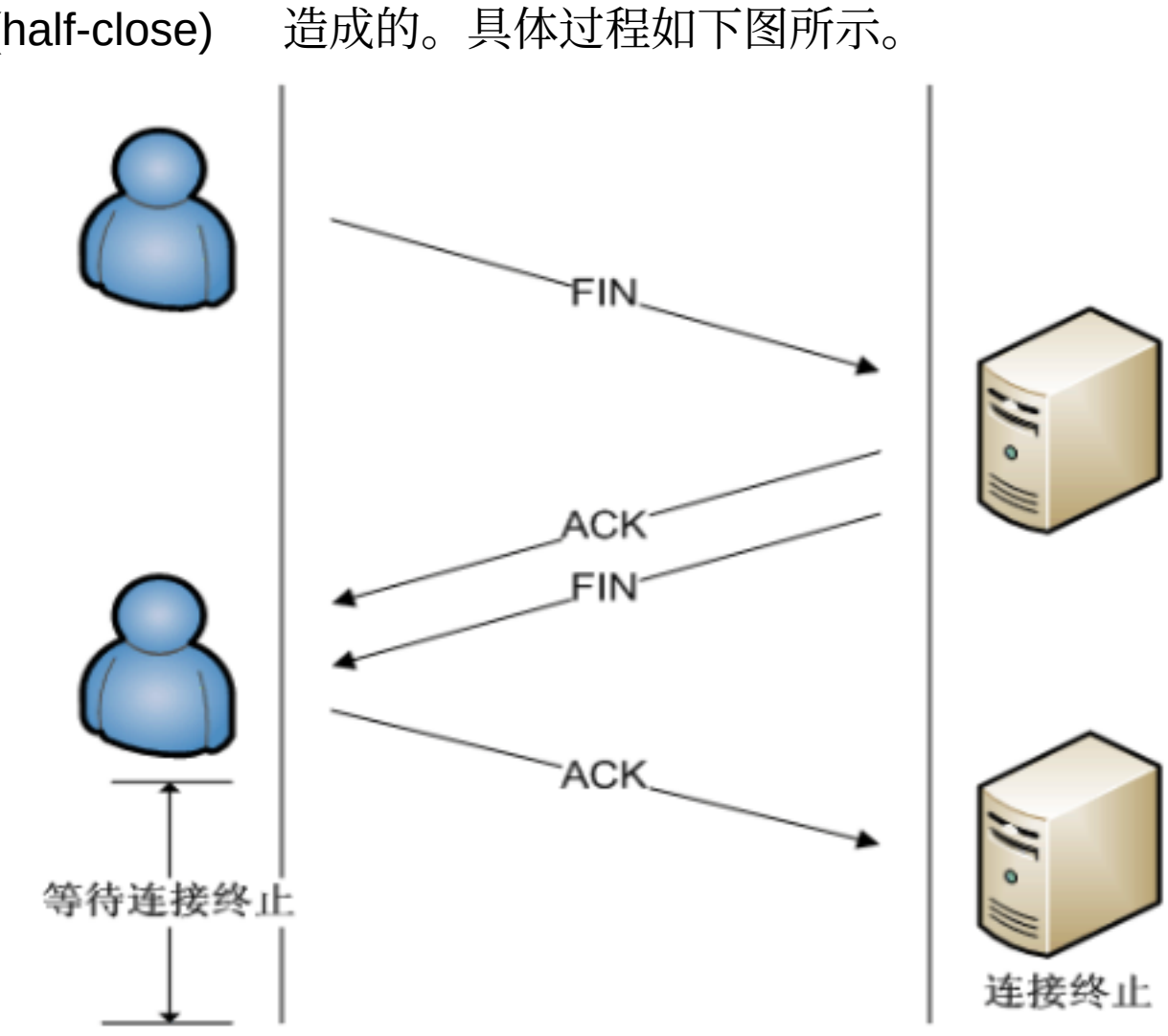


图 2-5 TCP 断开连接

2.UDP协议

UDP是 User Datagram Protocol 的简称，中文名是用户数据包协议，是 OSI 参考模型中一种无连接的传输层协议，提供面向事务的简单不可靠信息传送服务。

用户数据报协议 UDP提供了无连接的数据报服务。 它适用于无须应答并且通常一次只传送少量数据的应用软件。

2.3 Linux 下网络编程介绍

Linux 下的网络编程主要是基于 Linux 提供的 Socket API 函数来进行的。 所以，Linux 下的网络编程的基础就是对 socket API 函数的掌握，就必须理解和学会使用 socket 接口。同时针对并发服务， Linux 下提供了 I/O 复用等高效的形式来满足并发的要求。

2.3.1 Socket 简介

Socket 接口是 TCP/IP 网络的 API。Socket 接口定义了许多函数或例程，程序员可以用它们来开发 TCP/IP 网络上的应用程序。要学习 Internet 上的 TCP/IP 网络编程，必须理解 Socket 接口。

Socket 接口设计者最先是将接口放在 Unix 操作系统里面的。 如果了解 Unix 系统的输入和输出的话，就很容易了解 Socket 了。网络的 Socket 数据传输是一种特殊的 I/O ， Socket 也是一种文件描述符。 Socket 也具有一个类似于打开文件的函数调用 Socket() ， 该函数返回一个整型的 Socket 描述符，随后的连接建立、数据传输等操作都是通过该 Socket 实现的。常用的 Socket 类型有两种：流式 Socket（SOCK_STREAM）和数据报式 Socket（SOCK_DGRAM）。流式是一种面向连接的 Socket，针对于面向连接的 TCP服务应用；数据报式 Socket 是一种无连接的 Socket，对应于无连接的 UDP服务应用。

2.3.2 Socket 创建

为了创建 Socket，程序可以调用 Socket 函数，该函数返回一个类似于文件描述符的句柄。 socket 函数原型为：

```
int socket(int domain, int type, int protocol);
```

domain 指明所使用的协议族，通常为 PF_INET，表示互联网协议族（ TCP/IP 协议族）；type 参数指定 socket 的类型： SOCK_STREAM 或 SOCK_DGRAM，Socket 接口还定义了原始 Socket（SOCK_RAW），允许程序使用低层协议； protocol 通常赋值“ 0 ”。 Socket() 调用返回一个整型 socket 描述符，你可以在后面的调用使用它。

Socket 描述符是一个指向内部数据结构的指针，它指向描述符表入口。调用 Socket 函数时，socket 执行体将建立一个 Socket，实际上 " 建立一个 Socket" 意味着为一个 Socket 数据结构分配存储空间。 Socket 执行体为你管理描述符表。

两个网络程序之间的一个网络连接包括五种信息： 通信协议、 本地协议地址、 本地主机端口、远端主机地址和远端协议端口。 Socket 数据结构中包含这五种信息。

2.3.3 Socket 配置

通过 socket 调用返回一个 socket 描述符后，在使用 socket 进行网络传输以前，必须配置该 socket 。面向连接的 socket 客户端通过调用 Connect 函数在 socket 数据结构中保存本地和远端信息。无连接 socket 的客户端和服务端以及面向连接 socket 的服务端通过调用 bind 函数来配置本地信息。

Bind 函数将 socket 与本机上的一个端口相关联， 随后你就可以在该端口监听服务请求。 Bind 函数原型为：

```
int bind(int sockfd,struct sockaddr *my_addr, int addrlen);
```

Sockfd 是调用 socket 函数返回的 socket 描述符 ,my_addr 是一个指向包含有本机 IP 地址及端口号等信息的 sockaddr 类型的指针；addrlen 常被设置为 sizeof(struct sockaddr) 。

struct sockaddr 结构类型是用来保存 socket 信息的：

```
struct sockaddr {
 unsigned short sa_family; /*                    地址族,     AF_xxx */
```

char sa_data[14]; /* 14 字节的协议地址 */

};

sa_family 一般为 AF_INET，代表 Internet （TCP/IP）地址族； sa_data 则包含该 socket 的 IP 地址和端口号。

另外还有一种结构类型：

struct sockaddr_in {

short int sin_family; /* 地址族 */

unsigned short int sin_port; /* 端口号 */

struct in_addr sin_addr; /* IP 地址 *

unsigned char sin_zero[8]; /* 填充 0 以保持与 struct sockaddr 同样大小 */

};

这个结构更方便使用。 sin_zero 用来将 sockaddr_in 结构填充到与 struct sockaddr 同样的长度，可以用 bzero() 或 memset()函数将其置为零。指向 sockaddr_in 的指针和指向 sockaddr 的指针可以相互转换，这意味着如果一个函数所需参数类型是 sockaddr 时，你可以在函数调用的时候将一个指向 sockaddr_in 的指针转换为指向 sockaddr 的指针；或者相反。

使用 bind 函数时，可以用下面的赋值实现自动获得本机 IP 地址和随机获取一个没有被占用的端口号：

my_addr.sin_port = 0; /* 系统随机选择一个未被使用的端口号 */

my_addr.sin_addr.s_addr = INADDR_ANY; /* 填入本机 IP 地址 */

通过将 my_addr.sin_port 置为 0，函数会自动为你选择一个未占用的端口来使用。同样，通过将 my_addr.sin_addr.s_addr 置为 INADDR_ANY，系统会自动填入本机 IP 地址。注意在使用 bind 函数是需要将 sin_port 和 sin_addr 转换成为网络字节优先顺序；而 sin_addr 则不需要转换。

计算机数据存储有两种字节优先顺序： 高位字节优先和低位字节优先。 Internet 上数据以高位字节优先顺序在网络上传输， 所以对于在内部是以低位字节优先方式存储数据的机器，在 Internet 上传输数据时就需要进行转换，否则就会出现数据不一致。下面是几个字节顺序转换函数：

- htonl()　：把 32 位值从主机字节序转换成网络字节序
- htons()　：把 16 位值从主机字节序转换成网络字节序
- ntohl()　：把 32 位值从网络字节序转换成主机字节序
- ntohs()　：把 16 位值从网络字节序转换成主机字节序

Bind() 函数在成功被调用时返回 0；出现错误时返回“ -1 ”并将 errno 置为相应的错误号。需要注意的是，在调用 bind 函数时一般不要将端口号置为小于 1024 的值，因为 1 到 1024 是保留端口号， 你可以选择大于 1024 中的任何一个没有被占用的端口号。

2.3.4 建立连接

第六章　总结与体会

毕业设计最初的时候，　主要是对相关资料的收集和理论知识的学习。　在这个阶段，最好是同时结合资料和源码一起来看，　效果会比较好，　学习效率较高。 看资料和教程是从细节和基础上去学习知识，　而看相关程序的源码则是从整体和实现上去了解一个系统。这样才能做到“ 见树又见林 ” 。学习理论知识可以使我们掌握最基础的知识，能更深入的了解设计的底层实现。　当在具体实现的时候，　可以以模块或分层次的思想来分析系统。 重点掌握核心的模块，　其他模块可以采用现有的类库或开源的实现，　这样可以提高开发的效率。　软件开发其实对于代码量的积累是很重要的。　当积累了一定的代码量后，看问题就会比较有程序的思想，能够从层次，模块的角度来分析问题，这样思路就比较清晰了。

整个毕业设计的过程其实就是经历了一个项目的生命周期。从最初的选题确定后，开始进行相关资料的收集和理论知识的学习，　接着确定自己的方案设计和系统整体结构，然后开始编码实现，调试代码，直至顺利运行，再进行性能测试，最后写出论文。这些步骤其实和一个软件项目的开发是很类似的。　软件的开发同样会有这些步骤，需求分析，设计，编码，测试，发布，文档撰写等。

当完成了整个毕业设计后，　对如何把握一个项目的整体有了一点基本的认识。　同时从中体会到时间控制和进度安排都是很重要的，任何任务和项目都是有时间期限的，自己的想法和设计都是得基于按时完成这个前提的。

四年的大学时光即将结束，　心中还是有不舍。 回顾四年的学习时光，　感觉自己还是过的蛮充实的。 做过很多有意思的事，　也认识了一帮好友与同窗。　无论是做人还是学习，我的老师和同学朋友们都给了我很大的帮助，　我非常感谢他们。 作为即将踏上工作、步入社会的我，我想我会更加努力奋斗，不让我的家人、朋友、老师们失望。

最后，愿大家在今后的日子里，一帆风顺，身体健康。

谢辞

本论文的工作是在我的指导老师　　老师的悉心指导下完成的，　谢涛老师创新的学习思想和积极奋斗的人生理念给了我很大的影响，　　同时对我的工作和人生规划都有很大帮助。在此衷心的感谢谢涛老师。

在大学四年的学习过程中，　我学到了很多做人与做事的学问，　度过了一个充实而快乐的大学时光。非常感谢各位亲爱的老师对我的教诲和指导，无论是知识的学习，还是职业规划和人生理想，　你们都给了我很多意见和指导。　我也很荣幸我能和各位同学、朋友们一起走过大学四年，　我们共同经历了人生中最美好的时光。　感谢你们的帮助和鼓励，希望走出校门后的我们仍然是一辈子的好朋友。

最后，我要感谢我的父母和亲人。 是你们的辛勤工作为我创造了良好的学习条件，是你们的信任、 鼓励和理解， 我才会取得今天的成就。 我会用我的努力工作来回报你们的养育之恩，希望你们永远身体健康，快乐长寿。

参考文献

[1] JAMES F.KUROSE,KEITH W.ROSS.计算机网络——自顶向下方法与 Internet 特色. 北京：机械工业出版社， 2005 年

[2] W.RICHARD STEVENS,BILL FENNER,ANDREW M.RUDOFF. UNIX 网络编程 第 1 卷套接口 API. 北京：清华大学出版社， 2006 年 6 月第 3 版

[3] W.RICHARD STEVENS,STEPHEN A.RAGO.UNIX环境高级编程（第 2 版）. 北京：人民邮电出版社， 2006 年

[4] 鸟哥 . 鸟哥的 Linux 私房菜基础学习篇 . 北京： 人民邮电出版社 ,2007 年 9 月

[5] 林宇，郭凌云 .Linux 网络编程 . 北京：人民邮电出版社， 2000.45 —65

[6] 郑齐，方思行 . 通用多线程服务器的设计与实现 . 计算机工程与应用, 2003.16:146 —147

[7] 胥光辉等译. W.RICHARD STEVENS. TCP/IP 详解 (第 I 卷) ：协议. 北京：机械工业出版社， 2000. 15—25.

[8] 张南平，徐静 . 基于进程池的 Linux 并发服务器的研究 . 计算机与数字工程,2009.1 ：159—161

[9] 邵芬, 于国防，张宁 . 基于多线程的 HTTP服务器的设计与实现 . 工矿自动化, 2007.8:134 —136

[9] 孙霞 . 基于 java 的高效多线程 HTTP 服务器的研究及实现 . 福建电脑,2003.11:38 —39

[10] 李磊. 嵌入式 WEB服务器软件的设计与实现 . 计算机工程与设计 2003(10)

[11] 白小明， 邱桃荣 . 基于 Linux 的嵌入式实时操作系统的研究 . 微计算机信息 , 2006,2-2:78-7.

[12] 车飞锋 . 基于嵌入式 Linux 的 Web和邮件服务器的设计与实现 . 西安石油大学：计算机应用技术， 2008

[13] 曲波，吴兆芝 . Linux 环境下面向 Web服务器的设计与实现 . 小型微型计算机系统 , 2002

[14] Dan Kegel.The C10K problem. http://www.kegel.com/c10k.html