《计算机网络》课程设计--协议设计与编码实现
指导书

实验 1. Windows 平台简单套接字编程

实验类型：设计性

一、实验环境

操作系统:Windows
编程工具及集成开发环境:VC++

二、实验目的和要求

实验目的：掌握 Windows 平台上简单的客户机端和服务器端的套接字编程。
实验要求：
（1）在 Windows 上，编写、编译 1 个客户机端程序和 1 个服务器端程序。要求客户机端程序能发送请求给服务器端程序，服务器端程序收到后能发送本机时间给客户机端程序。
（2）在相同或不同机子上，先运行服务器端程序可执行文件，后运行客户机端程序可执行文件。

TCP/UDP 赋予每个服务一个唯一的协议端口号。服务器程序通过协议端口号来指定它所提供的服务，然后被动地等待通信。客户在发送连接请求时，必须说明服务器程序运行主机的 IP 地址和协议端口号来指定它所希望的服务。服务器端计算机通过此端口号将收到的请求转向正确的服务器程序。

大多数网络编程语言都提供或者使用控件封装了套接字应用程序接口（Socket API），应用程序通过套接字接口调用来实现和传输层交互。用户目前可以使用两种套接口，即流套接字 TCP 和数据报套接字 UDP。流式套接字定义了一种可靠的面向连接的服务，提供了双向的，有序的，无重复的数据流服务。数据报套接字定义了一种无连接的服务，支持双向的数据流，但并不保证是可靠，有序，无重复的。也就是说，一个从数据报套接字接收信息的进程有可能发现信息重复了，或者和发出时的顺序不同。

套节字 API 主要包括表 1 所示的接口。

使用面向连接的套接字编程，通过图 2 来表示其时序。套接字工作过程如下：服务器首先启动，通过调用 socket()建立一个套接字，然后调用 bind()将该套接字和本地网络地址联系在一起，再调用 listen()使套接字做好侦听的准备，并规定它的请求队列的长度，之后就调用 accept()来接收连接。客户在建立套接字后就可调用 connect()和服务器建立连接，连接一旦建立，客户机和服务器之间就可以通过调用 read()和 write()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。

表 1 套节字 API 主要接口

接口	解释
SOCKET	创建一个新的套接字
BIND	给服务器绑定一个传输层地址
LISTEN	将服务器设为被动模式
ACCEPT	接收客户的一个请求
CONNET	客户向服务器发起连接
SEND	向一个连接的套接字发送数据
RECV	从一个连接的套接字接收数据
CLOSE	终止一个连接

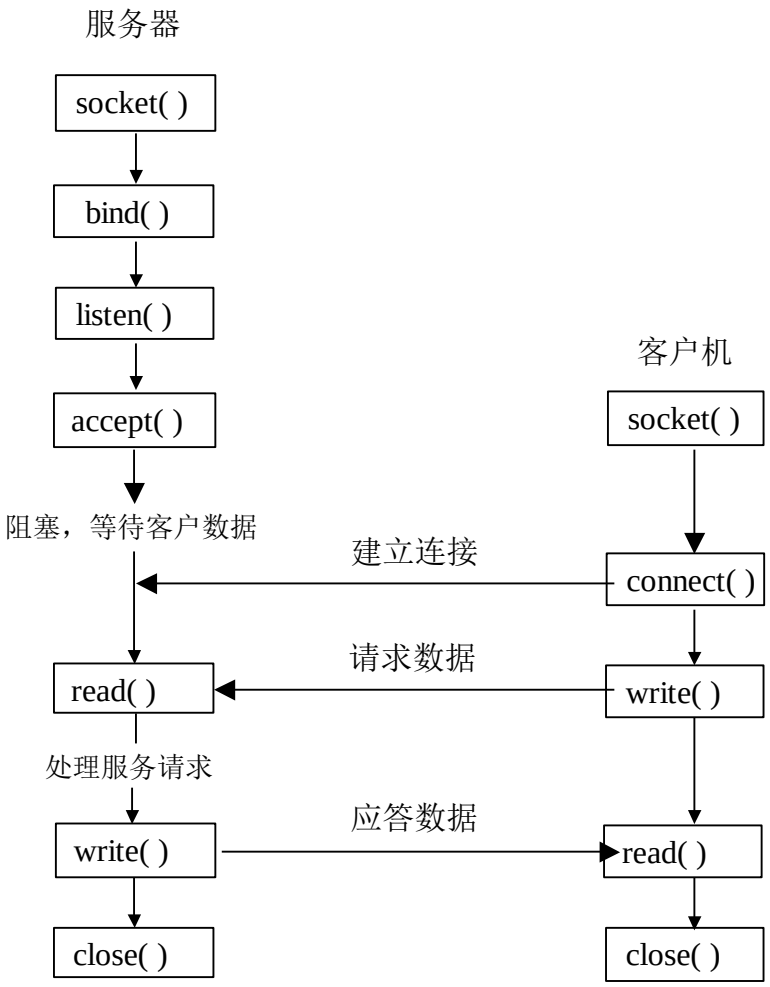


图 1 面向连接套接口应用程序时序图

实验 2. 并发套接字编程

实验类型：设计性

一、实验环境

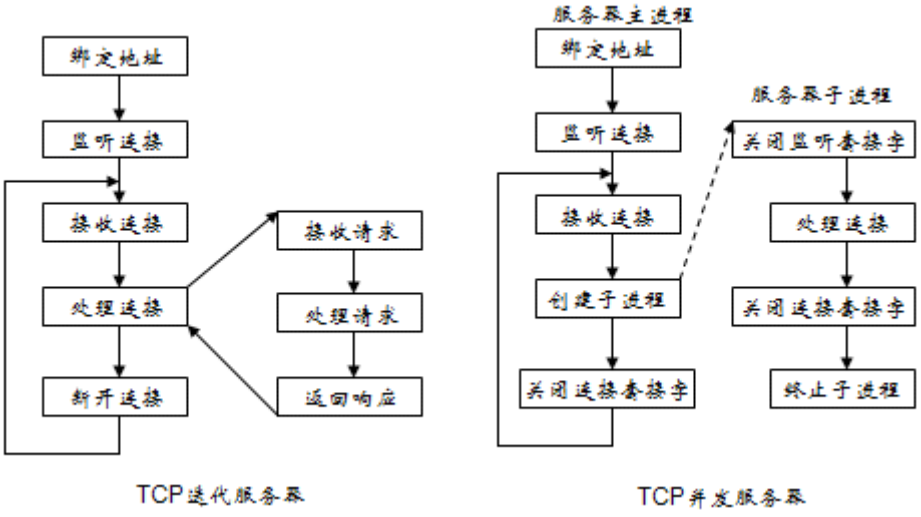
操作系统:Windows 或 Linux
编程工具及集成开发环境:VC++

二、实验目的和要求

实验目的：掌握 Linux 或 Windows 平台上多线程、多进程或异步 I/O 的套接字编程。
实验要求：
（1）在 Windows 上，编写、编译 1 个客户机端程序和 1 个服务器端程序。要求客户机端程序能发送请求给服务器端程序，服务器端程序收到后能发送本机时间给客户机端程序。
（2）在相同或不同机子上，先运行服务器端程序可执行文件，后运行客户机端程序可执行文件。
可选内容：
（1）编写 Linux 或 Windows 平台上采取各并发方式的 1 种程序。
（2）测试多线程、多进程或异步 I/O 的性能。

三、实验提示

（1）多线程程序编译连接需要用-pthread 选项



TCP 迭代服务器　　TCP 并发服务器

创建 TCP 并发服务器的算法如下：

```
socket(……);   //创建一个 TCP 套接字
bind(……);     //邦定公认的端口号
listen(……）; //倾听客户端连接
while(1）      //开始循环接收客户端的接收
```

```
{
        accept(……)；//接收一个客户端的连接
        if(fork(……)==0)  //创建子进程
        {
            while(1)
              {          //子进程处理某个客户端的连接
                  read(……)；
                  process(……);
                  write(……);
               }
              close(……);  //关闭子进程处理的客户端连接
               exit(……) ；//终止该子进程
          }
         close（……);          //父进程关闭连接套接字描述符，准备接收下一个客户端连接
    }
```

TCP 并发服务器可以解决 TCP 循环服务器客户端独占服务器的情况。但同时也带来了一个不小的问题，即响应客户机的请求，服务器要创建子进程来处理，而创建子进程是一种非常消耗资源的操作。

多路复用 I/O 并发服务器：创建子进程会带来系统资源的大量消耗，为了解决这个问题，采用多路复用 I/O 模型的并发服务器。采用 select 函数创建多路复用 I/O 模型的并发服务器的算法如下：

```
初始化(socket,bind,listen);
while(1)
{
  设置监听读写文件描述符（FD_*)；
  调用 select;
  如果是倾听套接字就绪，说明一个新的连接请求建立
  {
     建立连接（accept)；
     加入到监听文件描述符中去；
   }
  否则说明是一个已经连接过的描述符
   {
        进行操作(read 或者 write);
   }
}
```

```
int select(int n, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);

int poll(struct pollfd *ufds, unsigned int nfds, int timeout);

int epoll_wait(int epfd, struct epoll_event * events, int maxevents, int timeout)
```

实验 3. 原始套接字编程

实验类型：设计性

一、实验环境

操作系统:Windows
编程工具及集成开发环境:VC++

二、实验目的和要求

实验目的：掌握原始套接字编程。
实验要求：完成下列功能：
（1）利用 RAW SOCKET 捕获网络数据包的程序模型

SOCKET_STREAM 流式套接字
SOCKET_DGRAM
SOCKET_RAW 原始套接字
IPPROTO_IP IP 协议
IPPROTO_ICMP INTERNET 控制消息协议，配合原始套接字可以实现 ping 的功能
IPPROTO_IGMP INTERNET 网关服务协议，在多播中用到

在 AF_INET 地址族下，有 SOCK_STREAM、SOCK_DGRAM、SOCK_RAW 三种套接字类型。SOCK_STREAM 也就是通常所说的 TCP，而 SOCK_DGRAM 则是通常所说的 UDP，而 SOCK_RAW 则是用于提供一些较低级的控制的；第 3 个参数依赖于第 2 个参数，用于指定套接字所用的特定协议，设为 0 表示使用默认的协议。

RAW SOCKET 能够对较低层次的协议直接访问，网络监听技术很大程度上依赖于它。

（2）能够抓取第二节课的并发服务器程序的服务器端或客户端的应用层数据，即：时间值，打印输出。

输出示例：
Applications++
Data: XXXXXXX

Transportation++
Src Port: XXXXXX
Dest Port: XXXXXX
…….

Network++
Src IP: XXXXXXX
Dest IP: XXXXXX

1 引言

随着信息技术的快速发展，网络已成为信息交换的主要手段，一些网络新业务在不断地兴起，如电子商务、移动支付等，这些都对网络安全提出了较高的要求。与此同时，黑客对网络的攻击从未停止，网络的安全问题变得日趋严峻。
很多网络攻击都是从监听开始的，网络监听最重要一步就是捕获局域网中的数据帧，因此，研究数据捕获技术对于保障网络安全有着重要的意义。

2 RAW SOCKET 简介

同一台主机不同进程可以用进程号来唯一标识，但是在网络环境下进程号并不能唯一标识该进程。TCP/IP 主要引入了网络地址、端口和连接等概念来解决网络间进程标识问题。套接字（Socket）是一个指向传输提供者的句柄，TCP/IP 协议支持 3 种类型的套接字，分别是流式套接字、数据报式套接字和原始套接字。

流式套接字（SOCKET_STREAM）提供了面向连接、双向可靠的数据流传输服务。数据报式套接字(SOCKET_DGRAM)提供了无连接服务，不提供无错保证。原始套接字(SOCKET_RAW)允许对较低层次的协议直接访问，比如 IP、 ICMP 协议，它常用于检验新的协议实现，或者访问现有服务中配置的新设备，因为 RAW SOCKET 可以自如地控制 Windows 下的多种协议，能够对网络底层的传输机制进行控制，所以可以应用原始套接字来操纵网络层和传输层应用。比如，我们可以通过 RAW SOCKET 来接收发向本机的 ICMP、IGMP 协议包，或者接收 TCP/IP 栈不能够处理的 IP 包，也可以用来发送一些自定包头或自定协议的 IP 包。网络监听技术很大程度上依赖于 SOCKET_RAW。

3 RAW SOCKET 编程

要使用原始套接字，必须经过创建原始套接字、设置套接字选项和创建并填充相应协议头这三个步骤，然后用 send、WSASend 函数将组装好的数据发送出去。接收的过程也很相似，只是需要用 recv 或 WSARecv 函数接收数据。下面介绍使用 RAW SOCKET 编程的几个步骤。

3.1 创建原始套接字

我们可以用 socket 或 WSASocket 函数来创建原始套接字，因为原始套接字能直接控制底层协议，因此只有属于“管理员”组的成员，才有权创建原始套接字。下面是用 socket 函数创建原始套接字的代码。

```
SOCKET sock;

Sock＝socket (AF_INET, SOCK_RAW, IPPROTO_UDP);
```

上述创建原始套接字的代码使用的是 UDP 协议，如果要使用其它的协议，比如 ICMP、IGMP、IP 等协议，只需要把相应的参数改为 IPPROTO_ICM、IPPROTO_ IGMP、IPPROTO_IP 就可以了。另外，IPPROTO_UDP、IPPROTO_IP、IPPROTO_RAW 这几个协议标志要求使用套接字选项 IP_HDRINCL，而目前只有 Windows 2000

和 Windows XP 提供了对 IP_HDRINCL 的支持，这意味着在 Windows 2000 以下平台创建原始套接字时是不能使用 IP、UDP、TCP 协议的。

3.2 设置套接字选项

创建了原始套接字后，就要设置套接字选项，这要通过 setsocketopt 函数来实现，setsocketopt 函数的声明如下：

```
int setsocketopt (
SOCKET s,
int level,
int optname,
const char FAR *optval,
int optlen
);
```

在该声明中，参数 s 是标识套接口的描述字，要注意的是选项对这个套接字必须是有效的。参数 Level 表明选项定义的层次，对 TCP/IP 协议族而言，支持 SOL_SOCKET、IPPROTO_IP 和 IPPROTO_TCP 层次。参数 Optname 是需要设置的选项名，这些选项名是在 Winsock 头文件内定义的常数值。参数 optval 是一个指针，它指向存放选项值的缓冲区。参数 optlen 指示 optval 缓冲区的长度

3.3 创建并填充相应协议头

这一步就是创建 IP 和 TCP 协议头的数据结构，根据相关协议的定义进行编写即可，下面是一个 TCP 协议头的数据结构。

```
struct TCP
{
    unsigned short    tcp_sport;
    unsigned short    tcp_dport;
    unsigned int      tcp_seq;
    unsigned int      tcp_ack;
    unsigned char     tcp_lenres;
    unsigned char     tcp_flag;
    unsigned short    tcp_win;
    unsigned short    tcp_sum;
    unsigned short    tcp_urp;
};
```

4　一个利用 RAW SOCKET 捕获网络数据包的程序模型

下面介绍一个利用 RAW SOCKET 捕获网络数据包的程序模型。这个程序模型演示了如何使用 RAW SOCKET 捕获局域网中的数据包，它完成了网络底层数据的接收，能显示源地址、目标地址、源端口、目标端口和接收的字节数等信息。这个程序模型也说明了网络监听的基本原理，给捕获局域网中的数据包提供了一种方法，即先把

网卡设置为混杂模式，然后利用 RAW SOCKET 接收 IP 层的数据。

程序在 Visual C++.net 2003 中调试并编译通过，运行环境为以太网，程序代码可同时在 Linux 与 windows 环境下编译和运行，当然在编译时需要不同的头文件以及需要对代码作相应的改动。本程序模型在 Windows 下能直接运行，如果在 Linux 下运行，则需要先用手工把网卡设置为混杂模式，在 root 权限下用如下命令设置：ifconfig eth0 promisc。

在 Unix/Linux 下程序要包含以下这几个进行调用系统和网络函数的头文件：

```
#include〈stdio.h〉
#include〈sys/socket.h〉
#include〈netinet/in.h〉
#include〈arpa/inet.h〉
#include"headers.h"
```

为了方便基于 Berkeley 套接口的已有源程序的移植，Windows Sockets 支持许多 Berkeley 头文件。这些 Berkeley 头文件被包含在 WINSOCK2.H 中，所以一个 Windows Sockets 应用程序只需包含 WINSOCK2.H 头文件就足够了，这也是目前推荐使用的一种方法。在 Windows 平台下程序改用以下这几个头文件：

```
#include "stdafx.h"
#include<stdio.h>
#include<Winsock2.h>
#include"headers.h"
```

headers.h 是自己编写的头文件，它的作用是定义 IP 和 TCP 包的头结构。在程序中首先定义几个变量和结构，然后调用函数 socket（）建立 socket 连接,主要代码如下：

```
int _tmain(int argc, _TCHAR* argv[])
{
int sock,bytes_recieved,fromlen;
char buffer[65535];
struct sockaddr_in from;
struct ip *ip;
struct tcp *tcp;
sock=socket(AF_INET,SOCK_RAW,IPPROTO_TCP);
        ……
return 0;
}
```

程序的第二步用一个 while(1)语句来建立一个死循环，用来不停地接收网络信息。首先用函数 sizeof()取出一个 socket 结构的长度，这个参数是 recvfrom()函数所必须的。从建立的 socket 连接中接收数据是通过函数 recvfrom（）是来实现的，因为 recvfrom()函数需要一个 sockaddr 数据类型，所以用了一个强制类型转换，代码如下：

```
fromlen=sizeof(from);
bytes_recieved=recvfrom(sock,buffer,sizeof(buffer),0,(struct sockaddr*)&from,&fromlen);
```

接下来用一条语句把接收到的数据转化为我们预先定义的结构，以便于查看，代码为：

```
ip=(struct ip *)buffer
```

还要用一条语句来指向 TCP 头，因为接收的数据中，IP 头的大小是固定的 4 字节，所以用 IP 长度乘以 4 就能指向 TCP 头部分，代码为：

```
tcp=(struct tcp *)(buffer+(4*ip->ip_length))
```

最后就可以用打印语句把接收的字节数、数据的源地址、目标地址、源端口、目标端口、IP 头长度和协议的类型输出来。

实验 4. FTP 综合应用编程

实验类型：设计性

一、实验环境

操作系统:Windows
编程工具及集成开发环境:VC++

二、实验目的和要求

实验目的：掌握 FTP 应用编程。
实验要求：完成下列功能：

（1） 在前三个实验的基础上，将其改造为一个能传输指定文件名称的点对点文件传输软件
（2） 设计并实现一个支持多个客户端的文件传输服务器
（3） 客户端等待键盘输入文件名称，然后将文件名称传输给服务器，服务器在预先设置好的文件夹下查找该文件，如果发现同名文件，开始传输回客户端，客户端接收完文件后将文件以输入的文件名称保存在本地某个目录即可，否则告诉客户端文件不存在。

注意事项：

（1） 可以事先在服务器上的某个目录存放一些已经名称的文件，用作客户端指下载。

附录 1. 程序评分标准

1．程序功能和原理（40％）-把具体要求分解，思路正确
2．程序质量（40%）
（1）用大括号和缩进来清楚地显示程序结构。（提示：按一次"tab"键产生一个缩进)
（2）各函数有功能说明和参数说明
（3）每个源程序文件都有说明（比如本程序功能，作者，包含哪些函数）
（4）每个函数长度不超过 100 行
（5）函数、变量取名前后一致并容易理解
（6）对不容易理解的常量、变量和语句有注释（比如全局常量、变量、if 语句）
（7）是否允许全部合法输入,是否能检测出所有不合理输入
3．程序编写和调试综合能力 （20％）,比如
（1）会单步运行到任何一个语句
（2）单步运行时能查看变量值
4. 3 人/组，给出详细的设计说明，并进行代码演示；报优、良需参加答辩。