通信网络实验报告

姓名：陆超、潘豪

学号：01101486 、01101491

班级：011015

2013.12.28

实验内容

1.广域网实验
2.网络数据获取
3.socket 编程实验
4.UDP/TCP 通信实验
5.数据链路层协议的设计与实现
6.滑动窗口协议实验
7.嗅探器的实现

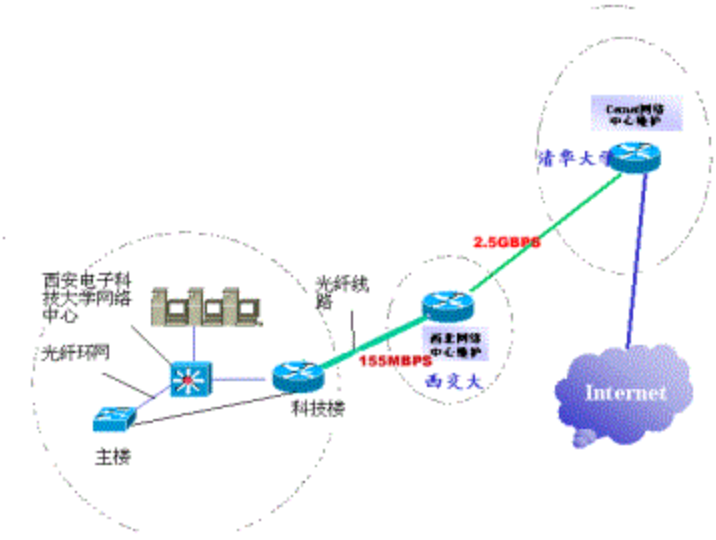
1、广域网实验

实验目的
1.了解广域网的组成和工作原理。
2.学会使用广域网中常用的软件以及协议配置。

实验设备
1.微机
2.广域网络

实验原理
在广域网中中讨论的主要问题
1.网络的结构
2.路由器的配置
3.分组数据的转发方式
4.网络的连通性

广域网的网络结构
教育网是典型的广域网络。
1.主楼和科技楼内部有各自的局域网络，这些网络又通过路由器连接在一起，最后这两个网络节点又和网管中心的网络设备之间通过光纤环网连接
2.这个光纤环网构成了我们校园网的骨干传输网



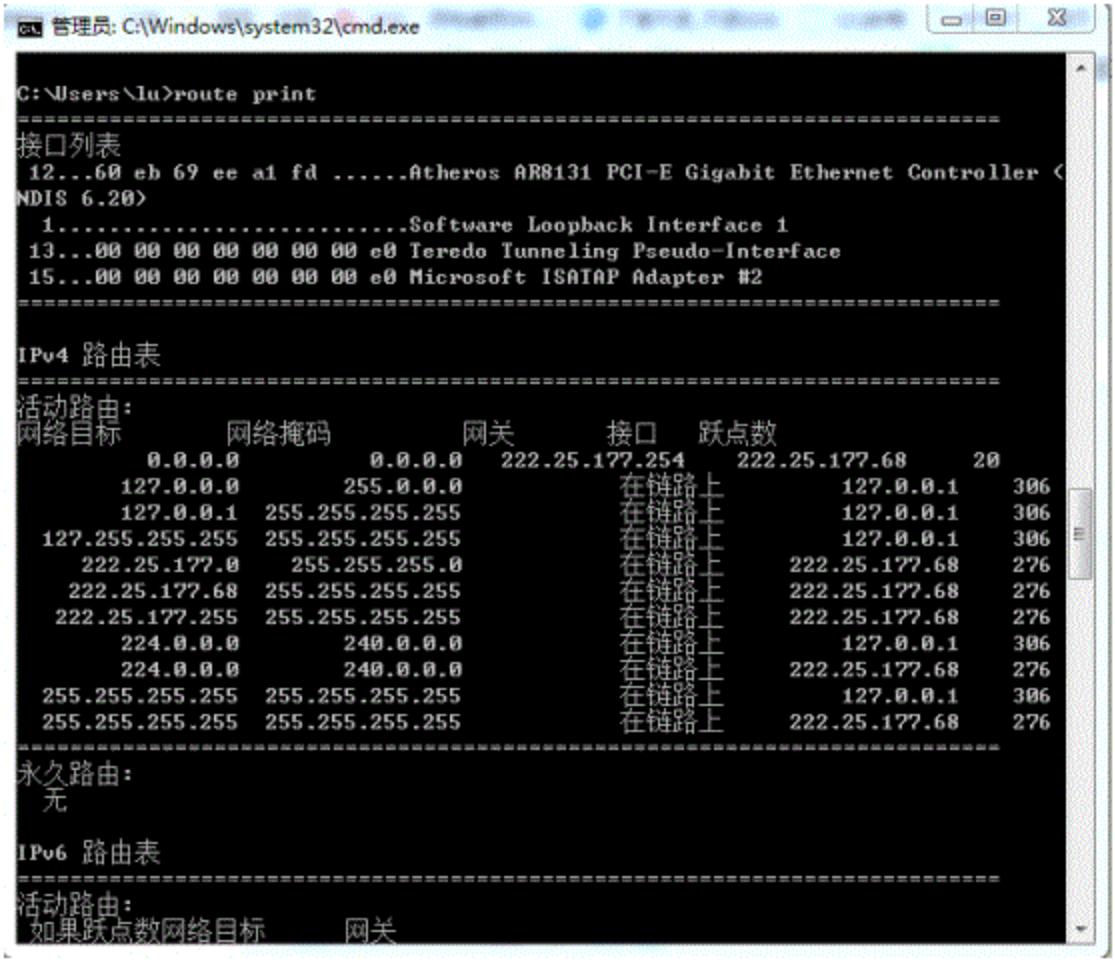
3.校园网又和西北网管中心之间通过 155MBPS 的光纤线路连接。
4.西北网管中心又通过 2.5GBPS 的光纤线路和中国教育网络中心相连接，通过教育网联入 INTERNET。
5.我们的实验环境是一个局域网。局域网中的主机通过网管中心分配的网关可以登录到 INTERNET。

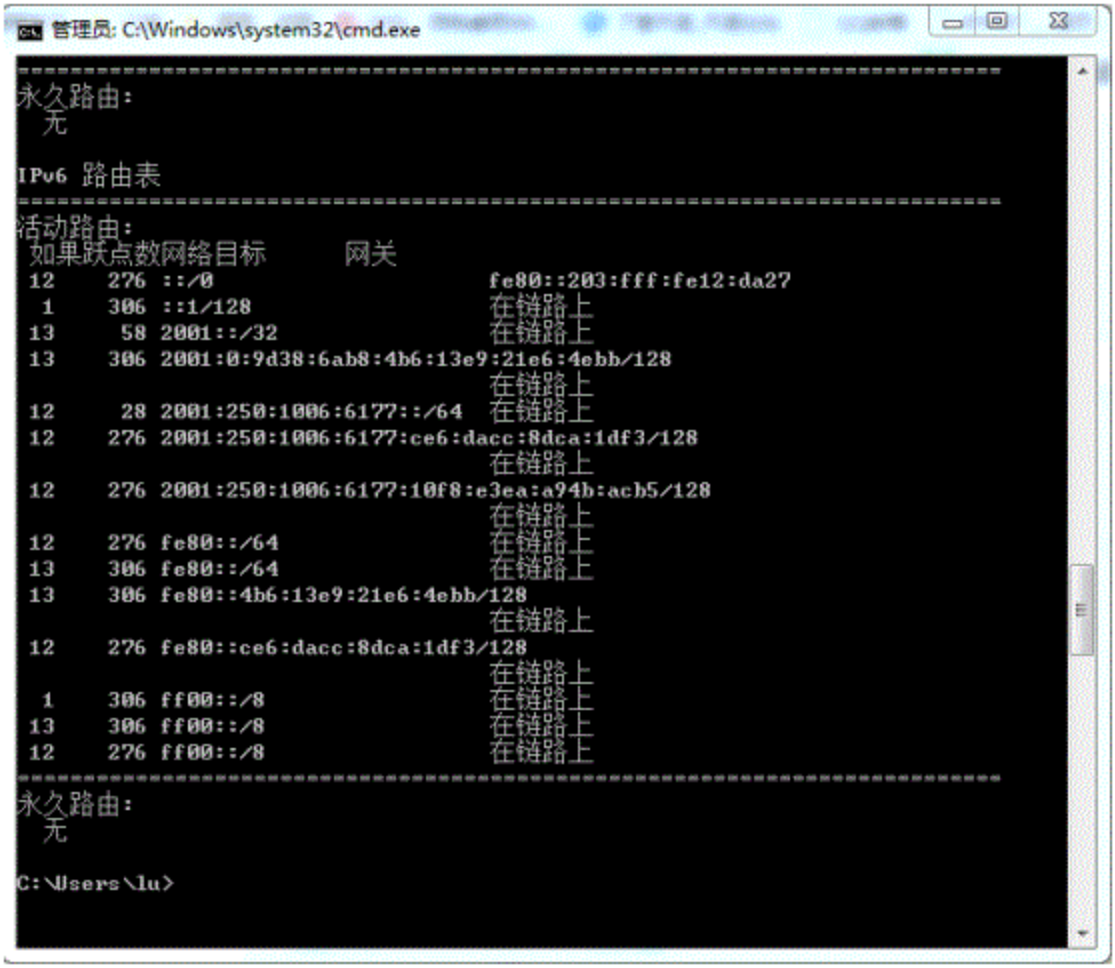
IP 地址的分配
1.通常，内部的 IP 地址采用的是 INTERNET 网络的专用网地址。这些地址通常是 192.168.0 段。
2.本次实验主要通过 PING、NETSTAT、ROUTE 命令来了解 IP 数据报在网络中路由选择的过程。

常用命令
1.PING：主要用来测试网络的连通性。它使用了 ICMP 回送请求与回送回答报文。
2.NETSTAT：主要功能是显示路由表的信息
3.ROUTE ：主要功能是显示、修改、删除、添加路由表

实验要求
1.利用 PING 命令在网关机工作前后分别测试网络的连通性。（网关不会关）

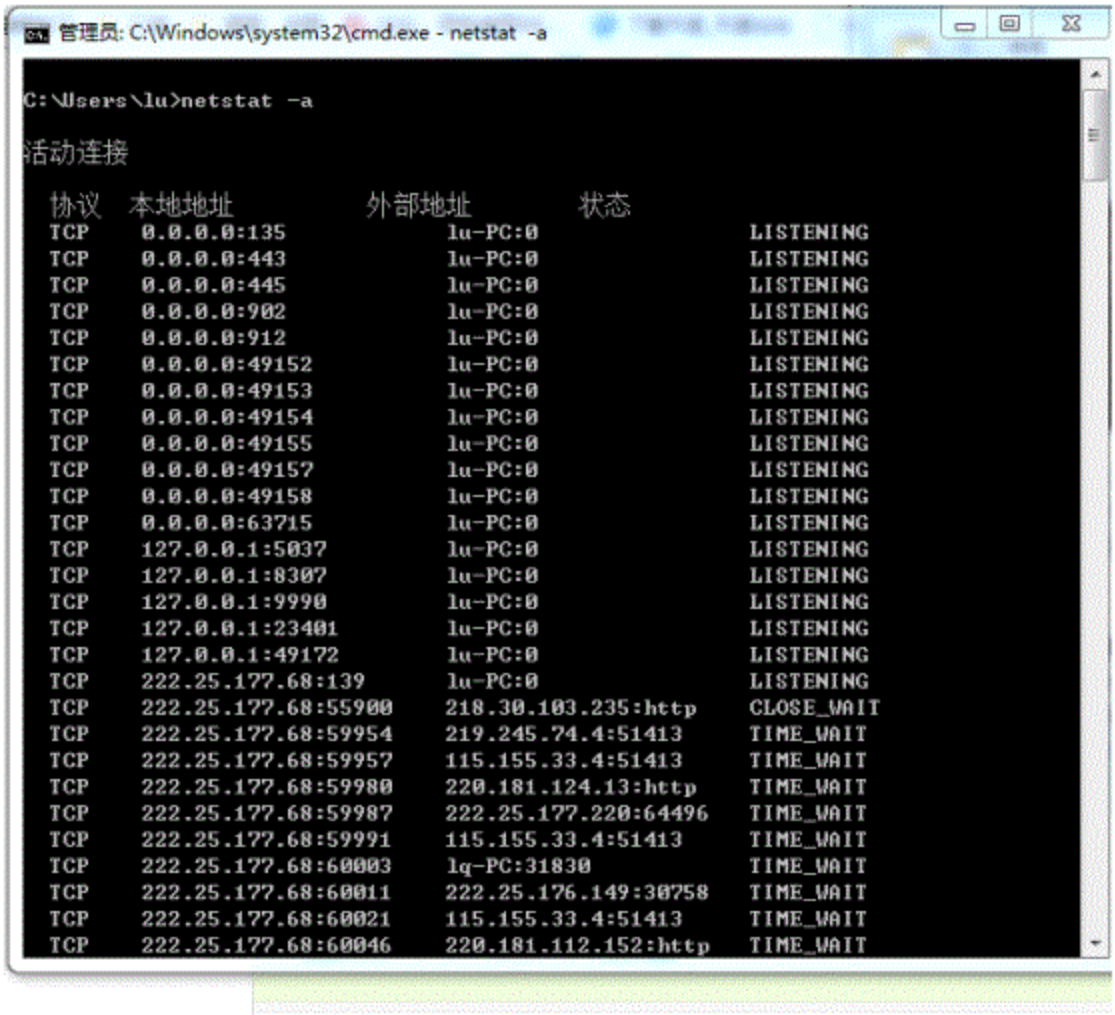
2.利用 route 命令配置主机的路由器表，并写出实验室内 IP 分组传送经过的所有可能路由。


管理员: C:\Windows\system32\cmd.exe
C:\Users\lu>route print
接口列表
12...60 eb 69 ee a1 fd ......Atheros AR8131 PCI-E Gigabit Ethernet Controller (
NDIS 6.20)
1...........................Software Loopback Interface 1
13...00 00 00 00 00 00 00 e0 Teredo Tunneling Pseudo-Interface
15...00 00 00 00 00 00 00 e0 Microsoft ISATAP Adapter #2
IPv4 路由表
活动路由:
网络目标 网络掩码 网关 接口 跃点数
0.0.0.0 0.0.0.0 222.25.177.254 222.25.177.68 20
127.0.0.0 255.0.0.0 在链路上 127.0.0.1 306
127.0.0.1 255.255.255.255 在链路上 127.0.0.1 306
127.255.255.255 255.255.255.255 在链路上 127.0.0.1 306
222.25.177.0 255.255.255.0 在链路上 222.25.177.68 276
222.25.177.68 255.255.255.255 在链路上 222.25.177.68 276
222.25.177.255 255.255.255.255 在链路上 222.25.177.68 276
224.0.0.0 240.0.0.0 在链路上 127.0.0.1 306
224.0.0.0 240.0.0.0 在链路上 222.25.177.68 276
255.255.255.255 255.255.255.255 在链路上 127.0.0.1 306
255.255.255.255 255.255.255.255 在链路上 222.25.177.68 276
永久路由:
无
IPv6 路由表
活动路由:
如果跃点数网络目标 网关



3.删除默认网关后，运行 HTTP 高层进程，观察结果。恢复默认网关。

4.利用 netstat 命令在网关机工作前后分别测试路由器的工作情况，开启几个高层应用进程，对高层协议的运行情况进行统计。指出其地址、端口号、当前的状态等信息。（网关不会关）



2、网络数据获取

1.网络监听是一种常用的被动式网络攻击方法，能帮助入侵者轻易获得用其他方法很难获得的信息，包括用户口令、账号、敏感数据、IP 地址、路由信息、TCP 套接字号等。

2.管理员使用网络监听工具可以监视网络的状态、数据流动情况以及网络上传输的信息。

实验目的

1.掌握 sniffer 捕获数据包的技术

2.了解一般局域网内监听手段

3.掌握如何防范攻击

原理简介

1.嗅探器（sniffer）是利用计算机的网络接口截获目的地为其他计算机的数据报文的一种技术。它工作在网络的底层，把网络传输的全部数据记录下来。

2.嗅探器可以帮助网络管理员查找网络漏洞和检测网络性能。嗅探器可以分析网络的流量，以便找出所关心的网络中潜在的问题。

3.不同传输介质网络的可监听性是不同的。
4.一般来说，以太网被监听的可能性比较高，因为以太网是一个广播型的网络
5.微波和无线网被监听的可能性同样比较高，因为无线电本身是一个广播型的传输媒介，弥散在空中的无线电信号可以被很轻易的截获。
6.在以太网中，嗅探器通过将以太网卡设置成混杂模式来捕获数据。
7.因为以太网协议的工作方式是将要发送的数据包发往连接在一起的所有主机，包中包含着应该接收数据包主机的正确地址，只有与数据包中目标地址一致的那台主机才能接收。
8.但是，当主机工作监听模式下，无论数据包中的目标地址是什么，主机都将接收（当然只能监听经过自己网络接口的那些包）。
9.因特网上有很多使用以太网协议的局域网，许多主机通过电缆、集线器连在一起。
10.当同一网络中的两台主机通信的时候，源主机将写有目的主机地址的数据包直接发向目的主机。但这种数据包不能在 IP 层直接发送，必须从 TCP/IP 协议的 IP 层交给网络接口，也就是数据链路层，而网络接口是不会识别 IP 地址的，

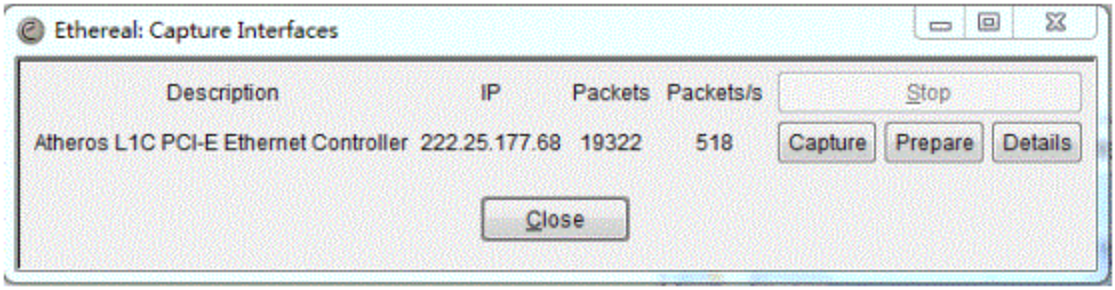
11.因此在网络接口数据包又增加了一部分以太帧头的信息。在帧头中有两个域，分别为只有网络接口才能识别的源主机和目的主机的物理地址，这是一个与 IP 地址相对应的 48 位的以太地址。
传输数据时，包含物理地址的帧从网络接口（网卡）发送到物理的线路上
如果局域网是由一条粗缆或细缆连接而成，则数字信号在电缆上传输，能够到达线路上的每一台主机。
当使用集线器时，由集线器再发向连接在集线器上的每一条线路，数字信号也能到达连接在集线器上的每一台主机
当数字信号到达一台主机的网络接口时，正常情况下，网络接口读入数据帧，进行检查，如果数据帧中携带的物理地址是自己的或者是广播地址，则将数据帧交给上层协议软件，也就是 IP 层软件，否则就将这个帧丢弃。对于每一个到达网络接口的数据帧，都要进行这个过程。
然而，当主机工作在监听模式下，所有的数据帧都将被交给上层协议软件处理。

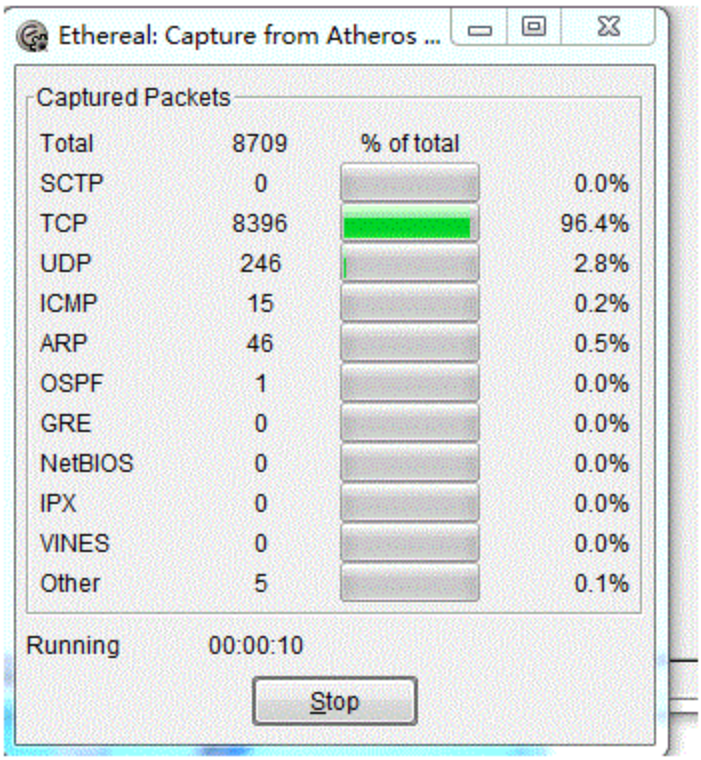
而且，当连接在同一条电缆或集线器上的主机被逻辑地分为几个子网时，如果一台主机处于监听模式下，它还能接收到发向与自己不在同一子网（使用了不同的掩码、IP 地址和网关）的主机的数据包。也就是说，在同一条物理信道上传输的所有信息都可以被接收到。
另外，现在网络中使用的大部分协议都是很早设计的，许多协议的实现都是基于一种非常友好的、通信的双方充分信任的基础之上，许多信息以明文发送。因此，如果用户的账户名和口令等信息也以明文的方式在网上传输，而此时一个黑客或网络攻击者正在进行网络监听，只要具有初步的网络和 TCP/IP 协议知识，便能轻易地从监听到的信息中提取出感兴趣的部分。
同理，正确的使用网络监听技术也可以发现入侵并对入侵者进行追踪定位，在对网络犯罪进行侦查取证时获取有关犯罪行为的重要信息，成为打击网络犯罪的有力手段。

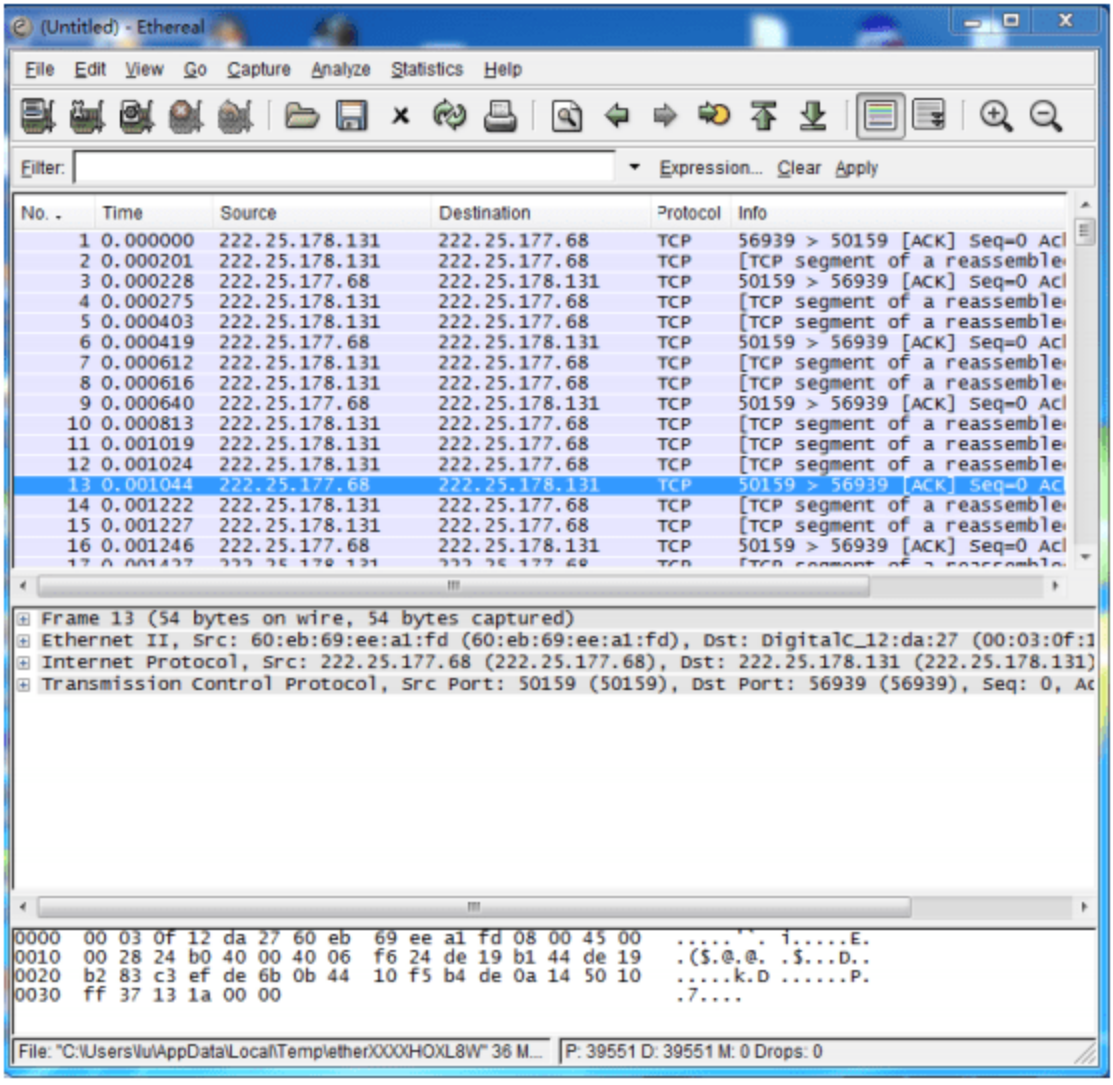
实验环境

网内设有 3 台主机，IP 地址分别为 192.168.0.61，192.168.0.101，192.168.0.92
其中 192.168.0.92 主机中安装 sniffer 软件 Ethereal
在主机 192.168.0.101 上运行 FTP 服务器

实验报告
详细描述实验过程。
仔细查看截获的数据，分析通过嗅探器能够获取哪些信息。
（本子太烂无法在机器上运行两台以上的模拟器，所以就只弄了一台的）







思考题

1.根据嗅探器工作原理，分析如何抵御嗅探器攻击？

网络嗅探器属于第二层的攻击,可以从以下几方面进行预防: 1)交换网络。随着交换机的成本和价格的大幅度降低,交换机已成为非常有效的使 sniffer 失效的设备。只要防止以上所提到的几种欺骗基本可避免交换网络中嗅探器的运行,进而避免秘密信息的泄漏。如果交换机具有端口安全保护功能,开启这种功能,交换机的每个端口将只与唯一的地址 MAC 相关联,从而有效地防范 MAC 地址欺骗和 MAC 地址淹没。对交换机每个端口都手工配置 MAC 地址,会给管理员带来很大的工作量。一般只对连接路由器的端口手工配置 MAC 地址,以防止路由欺骗。 2)加密。黑客对用户的口令、账户等信息比较敏感。目前有许多软件包可用于加密连接,可以采用对用户的重要数据进行加密,从而使入侵者即使捕获到数据,也无法将数据解密而失去窃听的意义。 3)安全意识。全部网络用户,必须加强网络安全意识,即对系统进行升级,安装补丁。对机密信息必须加密传送。网

络管理员经常检查网络设施与配置,关闭交换机未使用的端口,尽量停止使用共享式 HUB。

3、Socket 编程实验
实验目的：熟练掌握 socket 编程命令
实验内容：一个简单的客户机/服务器程序的实现
实验原理：
1.Sockets 编程基础知识
2.基于 C 的 Socket 编程相关函数和数据类型

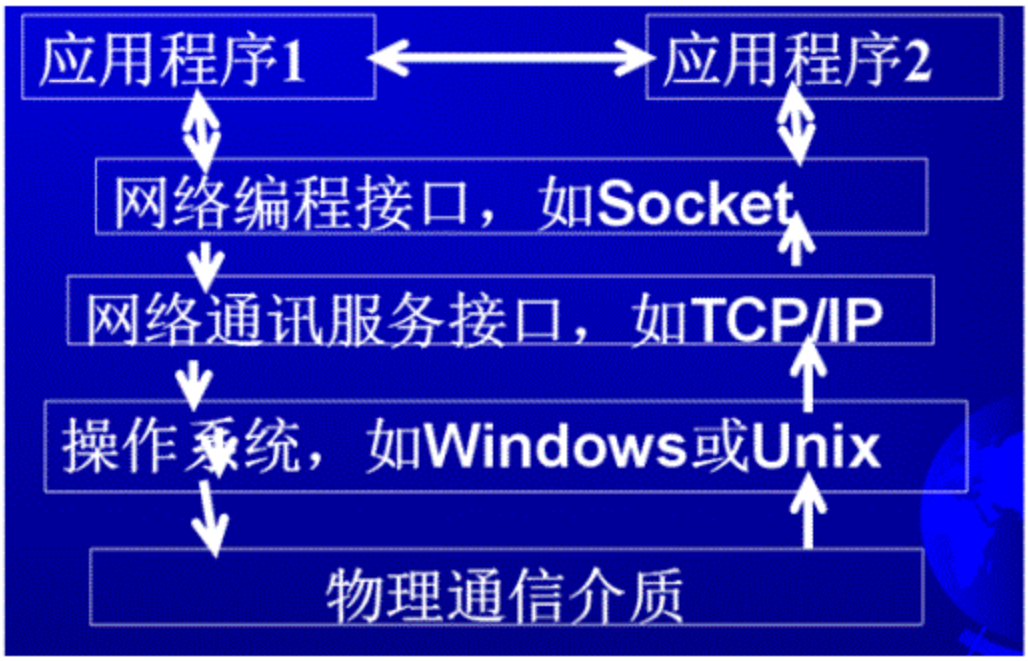
实验内容
一个简单的客户机/服务器程序的实现
1.它用套接字接口在 TCP 连接上发送消息。
2.允许用户在一端的机器上输入并把文本发送给另一端机器的用户。它是 UNIX 中 talk 的一个简化版本，类似于 WEB 聊天室的核心程序。

客户端
1.客户端用远端的机器名作为参数。它调用 UNIX 程序 gethostbyname 把该名字转化为远端主机的 IP 地址。
2.下一步构造套接字接口所需的地址数据结构（sin）。注意这个数据结构表明我们将一直用套接字与因特网（AF_INET）连接。在这个例子中，我们用 TCP 端口号 5432 作为共知的服务器端口号；它恰好不是分配给其他因特网服务的端口号。
3.建立连接的最后一步是调用 socket 和 connect。一旦 connect 操作返回，建立起连接，客户程序将进入主循环，不断从标准输入读文本并通过套接字发送。

服务器
1.服务器首先填上自己的端口号(SERVER_PORT)构造地址数据结构。
2.其次，它并不指明 IP 地址,从而使应用程序可以接受来自本地任一 IP 地址的连接。
3.然后，服务器执行与被动打开有关的初始步骤：建立一个套接字，将它绑定到本地地址，然后设置允许同时连接的最大数。
4.最后，主循环等待远端主机试图与之连接，当远端有一台主机试图与之连接时，它就接收并输出连接上送来的字符

实验原理
Sockets 编程基础知识
网络编程就是通过计算机网络与其他程序
进行通信的程序，Socket 编程是网络编程
的主流工具。Socket API 是实现进程间通信的一种编程设施，也是一种为进程间提供底层抽象的机制。
尽管应用开发人员很少需要在该层编写代码，但是理解 socket API 还是非常重要的。
第一，高层设施是构建于 socket API 之上的，它们是利用 socket API 提供的操

作来实现。
第二，对于响应时间要求较高或运行于有限资源平台上的应用，甚至 socket API 是唯一可用的进程间通信设施。
socket API 出现于 20 世纪 80 年代早期，作为 Berkeley Unix（BSD 4.2）操作系统程序库来通过进程间通信功能。
现在主流操作系统都提供 socket API。
在基于 Unix 系统中，如 BSD、Linux 系统，socket API 是操作系统内核的一部分
在 MS-DOS、Windows OS、OS/2 等操作系统中，socket API 是以程序库形式提供的，如在 Windows 系统中，socket API 被称为 Winsock。
Socket 接口规范可以适用多种通讯协议，主要是 TCP/IP。
TCP/IP 是计算机互联最常适用的网络通讯协议，TCP/IP 的核心部分由网络操作系统的内核实现，应用程序通过编程接口来访问 TCP/IP。



TCP/IP 使用一个网络地址和一个服务端口号来惟一地标识设备。
网络地址标识网络上的特定设备
端口号标识要连接到的该设备上的特定服务
网络通讯的基本模式如下：每一台通讯的主机都有一个本网络环境中惟一的 IP 地址，一台主机上往往有多个通讯程序存在，每个这样的程序都要占用一个通讯端口。因此，一个 IP 地址，一个通讯端口，就能确定一个通讯程序的位置。

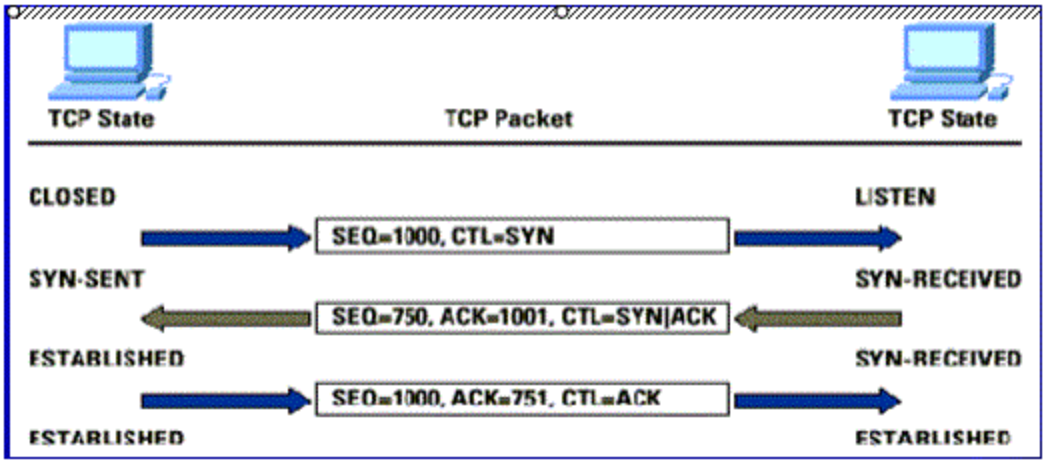
UDP/TCP 通信实验

实验目的
熟练掌握 UDP、TCP Client/Server 模式的通信原理。

实验内容
关于 UDP 的介绍可以参见教科书。
传输控制协议(Transport Control Protocol)是一种面向连接的，可靠的传输层协议。

面向连接是指一次正常的 TCP 传输需要通过在 TCP 客户端和 TCP 服务端建立特定的虚电路连接来完成，该过程通常被称为“三次握手”。
可靠性可以通过很多种方法来提供保证，在这里我们关心的是数据序列和确认。

TCP 通过数据分段(Segment)中的序列号保证所有传输的数据可以在远端按照正常的次序进行重组，而且通过确认保证数据传输的完整性。
要通过 TCP 传输数据，必须在两端主机之间建立连接。举例说明，TCP 客户端需要和 TCP 服务端建立连接，过程如图所示



TCP 客户端与服务端连接过程
第一步中，客户端向服务端提出连接请求。这时 TCP SYN 标志置位。客户端告诉服务端序列号区域合法，需要检查。客户端在 TCP 报头的序列号区中插入自己的 ISN。
服务端收到该 TCP 分段后，在第二步以自己的 ISN 回应(SYN 标志置位)，同时确认收到客户端的第一个 TCP 分段(ACK 标志置位)。
在第三步中，客户端确认收到服务端的 ISN(ACK 标志置位)。到此为止建立完整的 TCP 连接，开始全双工模式的数据传输过程。

实验内容
根据以上内容编写一个 TCP Client/Server 模式的通信程序。
事实上网络程序是由两个部分组成的--客户端和服务器端。它们的建立步骤如下
服务器端
socket-->bind-->listen-->accept
客户端
socket-->connect

实验步骤
实验按下述步骤进行：
（1） 编写 UDP、TCP Client/Server 模式的通信程序；

TCP Client/Server 模式通信程序（我用的是 vc 6.0 编写的，加载时别忘了加上 ws2_32.lib）

客户端源代码:

```
#include<stdio.h>
#include<Winsock2.h>


void main(){
//-----------------------------------------------------------------加载套接字（Socket）
    WORD wVersionRequested;
    WSADATA wsaData;
    int err;

    wVersionRequested = MAKEWORD(1,1);

    err = WSAStartup( wVersionRequested, &wsaData );
    if(err != 0){
        return;
    }

    if(LOBYTE(wsaData.wVersion) != 1 ||
        HIBYTE(wsaData.wVersion) != 1){
        WSACleanup();
        return;
    }

//-----------------------------------------------------------------

    SOCKET sockClient = socket(AF_INET,SOCK_STREAM,0);          //创建
Socket

    SOCKADDR_IN addrSrv;
    addrSrv.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");
    addrSrv.sin_family = AF_INET;
    addrSrv.sin_port = htons(5000);

    connect(sockClient,(SOCKADDR*)&addrSrv,sizeof(SOCKADDR));
//连接 Socket

    char recvBuffer[100];

    recv(sockClient,recvBuffer,100,0);//接收数据
    printf("%s\n",recvBuffer);
```

```
    send(sockClient,"I got it !",strlen("I got it !")+1,0); //发送数据
    closesocket(sockClient);
    WSACleanup();

}
```


服务器端代码:

```
#include<stdio.h>
#include<Winsock2.h>


void main(){
//-----------------------------------------------------------------------加载套接字（Socket）
    WORD wVersionRequested;
    WSADATA wsaData;
    int err;

    wVersionRequested = MAKEWORD(1,1);

    err = WSAStartup( wVersionRequested, &wsaData );
    if(err != 0){
        return;
    }

    if(LOBYTE(wsaData.wVersion) != 1 ||
        HIBYTE(wsaData.wVersion) != 1){
        WSACleanup();
        return;
    }

//-----------------------------------------------------------------------

    SOCKET sockSrv = socket(AF_INET,SOCK_STREAM,0);          //创建
Socket

    SOCKADDR_IN addrSrv;
    addrSrv.sin_addr.S_un.S_addr = htonl(INADDR_ANY);
```

```
    addrSrv.sin_family = AF_INET;
    addrSrv.sin_port = htons(5000);

    bind(sockSrv,(SOCKADDR*)&addrSrv,sizeof(SOCKADDR));
//绑定 Socket

    listen(sockSrv,10);                    //监听 socket

    SOCKADDR_IN addrClient;
    int len = sizeof(SOCKADDR);

    while(true){
        SOCKET sockConn = accept(sockSrv,(SOCKADDR*)&addrClient,&len); //
等待接收数据地址
        char sendBuffer[100];
        sprintf(sendBuffer,"Hello %s !",inet_ntoa(addrClient.sin_addr));
        send(sockConn,sendBuffer,strlen(sendBuffer)+1,0); //发送数据
        char recvBuffer[100];
        recv(sockConn,recvBuffer,100,0);//接收数据
        printf("%s\n",recvBuffer);
        closesocket(sockConn);
    }

}
```

UDP Client/Server 模式的通信程序

客户端源代码:

```
#include<stdio.h>
#include<Winsock2.h>


void main(){
//-----------------------------------------------------------------加载套接字（Socket）
    WORD wVersionRequested;
    WSADATA wsaData;
    int err;

    wVersionRequested = MAKEWORD(1,1);

    err = WSAStartup( wVersionRequested, &wsaData );
    if(err != 0){
```

```
        return;
    }

    if(LOBYTE(wsaData.wVersion) != 1 ||
        HIBYTE(wsaData.wVersion) != 1){
        WSACleanup();
        return;
    }

//-----------------------------------------------------------------

    SOCKET sockClient = socket(AF_INET,SOCK_DGRAM,0);           //创建
Socket

    SOCKADDR_IN addrSrv;
    char s[20];
//  printf("请输入服务器的 ip 地址: ");
//  scanf("%s",s);
    addrSrv.sin_addr.S_un.S_addr = inet_addr("222.25.177.68");
    addrSrv.sin_family = AF_INET;
    addrSrv.sin_port = htons(6000);

    char recvBuffer[100];
    char sendBuffer[100];
    char Buffer[100];

    int len = sizeof(SOCKADDR);

    while(true){
        printf("\n 请输入消息: ");
        gets(sendBuffer);

    sendto(sockClient,sendBuffer,strlen(sendBuffer)+1,0,(SOCKADDR*)&addrSrv,s
izeof(SOCKADDR));
        recvfrom(sockClient,recvBuffer,100,0,(SOCKADDR*)&addrSrv,&len);
        if('#' == recvBuffer[0]){

    sendto(sockClient,"#",strlen("#")+1,0,(SOCKADDR*)&addrSrv,sizeof(SOCKA
DDR));
            printf("Chat end !\n");
            break;
        }
        sprintf(Buffer,"Server say : %s",recvBuffer);
        printf("\n%s\n",Buffer);
```

```
    }

    closesocket(sockClient);
    WSACleanup();


}
```

服务器端代码:

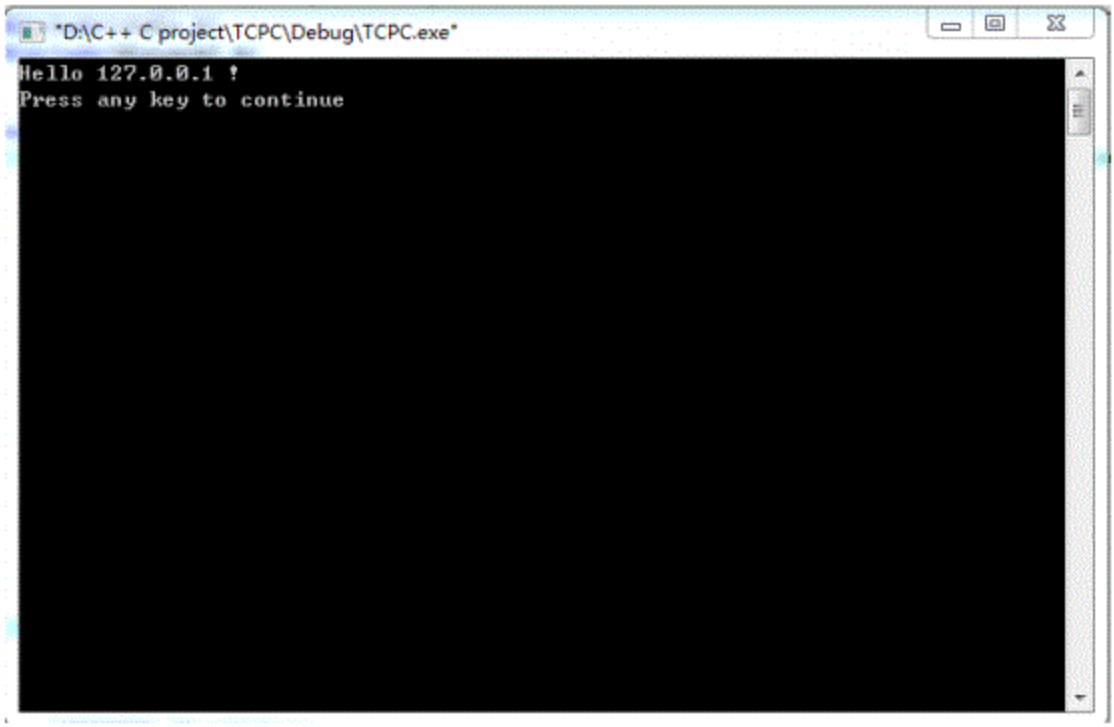
```
#include<stdio.h>
#include<Winsock2.h>


void main(){
//-----------------------------------------------------------------加载套接字（Socket）
    WORD wVersionRequested;
    WSADATA wsaData;
    int err;

    wVersionRequested = MAKEWORD(1,1);

    err = WSAStartup( wVersionRequested, &wsaData );
    if(err != 0){
        return;
    }

    if(LOBYTE(wsaData.wVersion) != 1 ||
        HIBYTE(wsaData.wVersion) != 1){
        WSACleanup();
        return;
    }

//-----------------------------------------------------------------


    SOCKET sockSrv = socket(AF_INET,SOCK_DGRAM,0);  //创建 Socket

    SOCKADDR_IN addrSrv;
    addrSrv.sin_addr.S_un.S_addr = htonl(INADDR_ANY);
    addrSrv.sin_family = AF_INET;
    addrSrv.sin_port = htons(6000);

    bind(sockSrv,(SOCKADDR*)&addrSrv,sizeof(SOCKADDR));     //绑定 Socket
```

```
    char recvBuffer[100];
    char sendBuffer[100];
    char Buffer[100];
    SOCKADDR_IN addrClient;
    int len = sizeof(SOCKADDR);

    while(true){
        recvfrom(sockSrv,recvBuffer,100,0,(SOCKADDR*)&addrClient,&len);
        if('#' == recvBuffer[0]){
            sendto(sockSrv,"#",strlen("q")+1,0,(SOCKADDR*)&addrClient,len);
            printf("Chat end!\n");
            break;
        }
        sprintf(Buffer,"%s say : %s",inet_ntoa(addrClient.sin_addr),recvBuffer);
        printf("\n%s\n",Buffer);
        printf("\n 请输入消息: ");
        gets(sendBuffer);

    sendto(sockSrv,sendBuffer,strlen(sendBuffer)+1,0,(SOCKADDR*)&addrClient,l
en);

    }

    closesocket(sockSrv);
    WSACleanup();

}
```

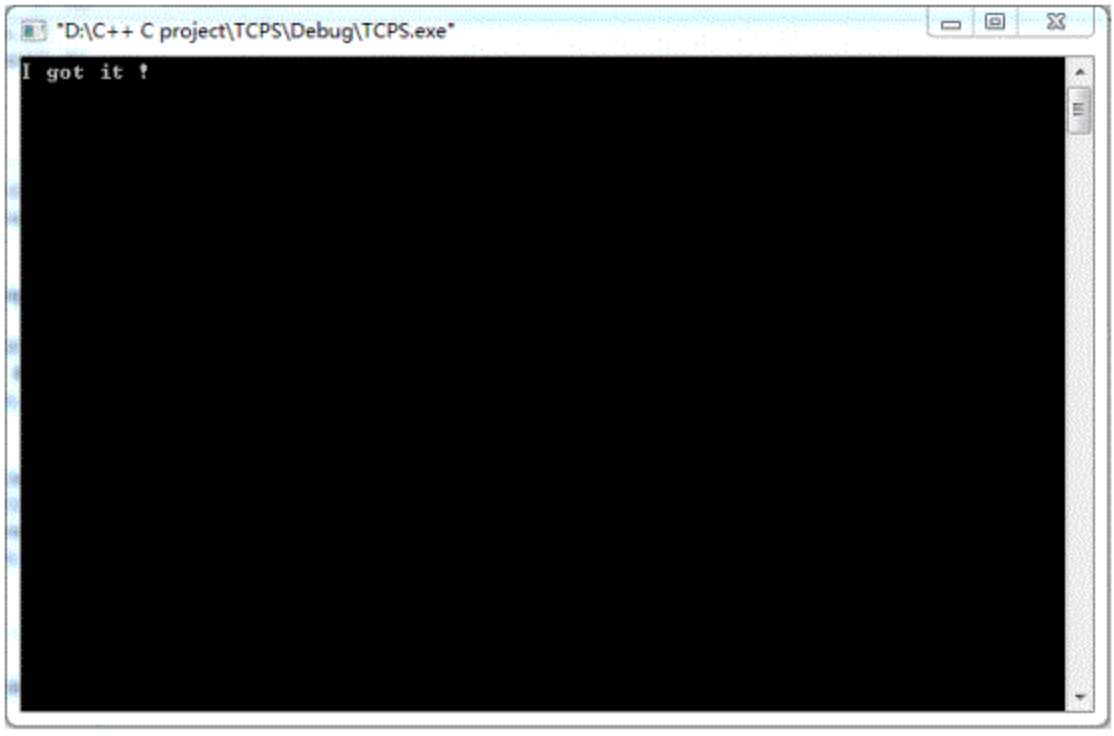
（2） 调试并运行自己编写的实现程序；

Tcp 程序运行结果（我用的是 127.0.0.1 会送地址在一台机子上运行测试）。

客户端截图：

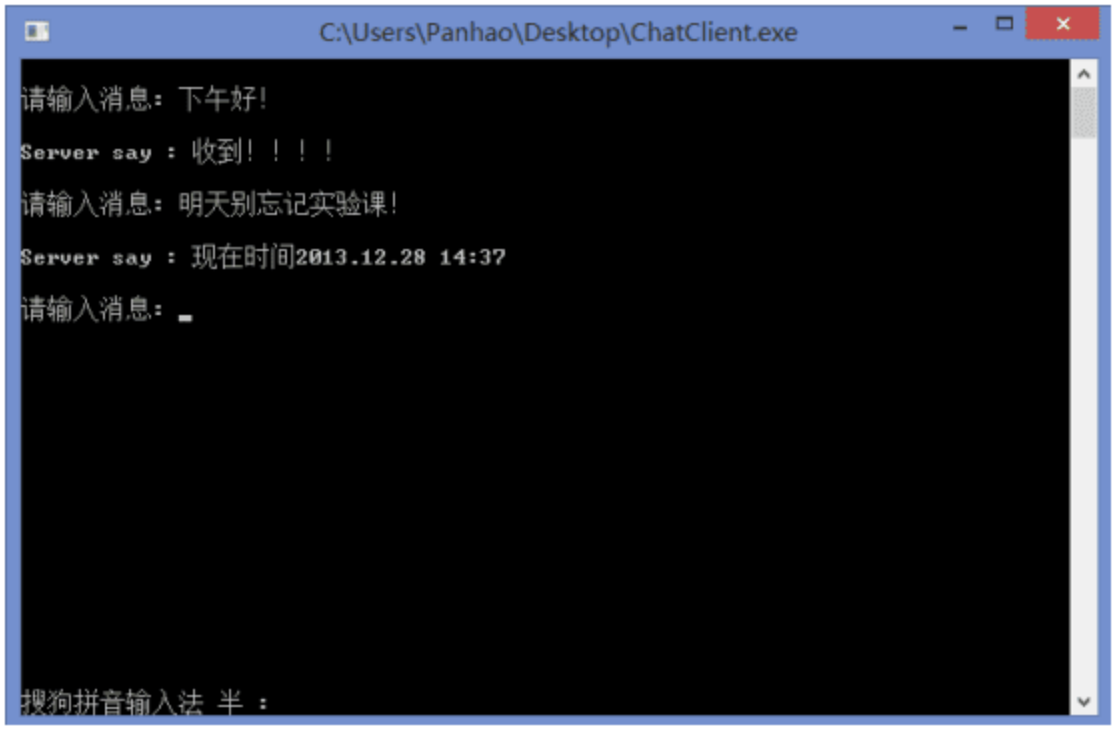


服务器截图：

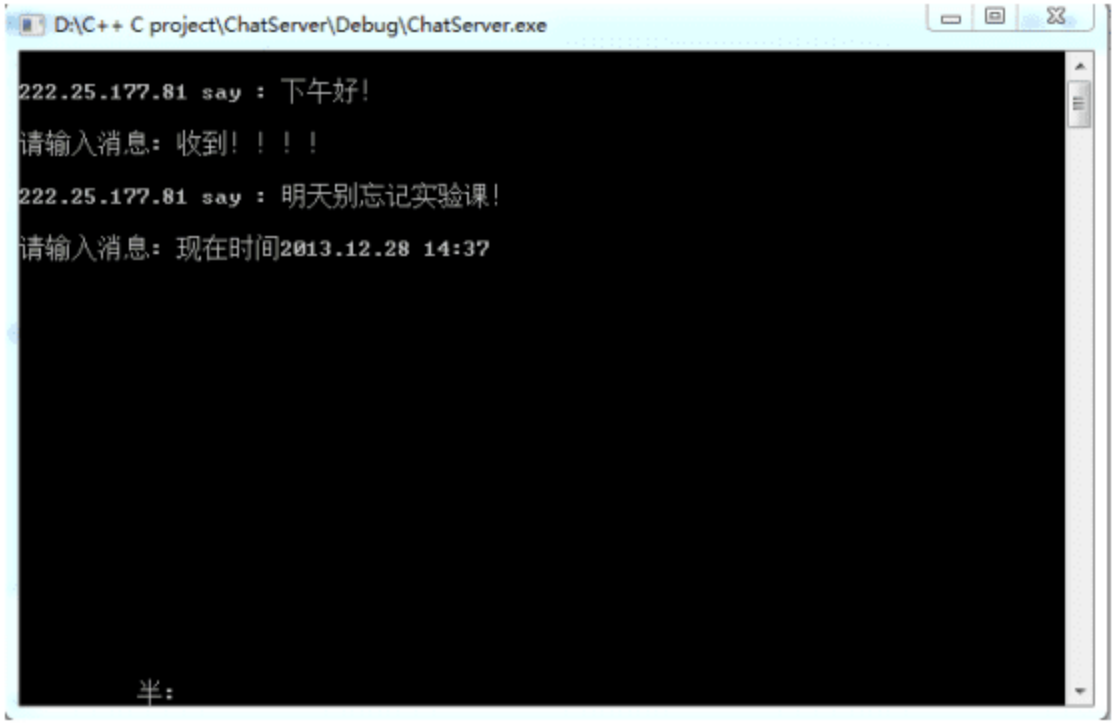


Udp 程序运行结果（我在两台机子上测试，我的 ip 是 222.25.177.68,。一台机子是 win7 系统（服务器），一台是 win8（客户端）所以界面有些不同）。

客户端截图：



服务器截图：



（3）了解 TCP Client/Server 模式的工作原理，比较二者的不同，如出现异常情况，在实验报告中写出原因分析；

TCP 是面向连接的传输控制协议，而 UDP 提供了无连接的数据报服务；TCP 具有高可靠性，确保传输数据的正确性，不出现丢失或乱序；UDP 在传输数据前不建立连接，不对数据报进行检查与修改，无须等待对方的应答，

所以会出现分组丢失、重复、乱序，应用程序需要负责传输可靠性方面的所有工作；也正因为以上特征，UDP 具有较好的实时性，工作效率较 TCP 协议高；UDP 段结构比 TCP 的段结构简单，因此网络开销也小。

报告中出现的异常：开始主要是在一台电脑中用回送地址测的时候没有如何问题，然后在两台电脑测的时候发现客户端老收到乱码，服务器没有收到任何信息。最后发现是服务器和客户端的启动顺序问题。（先启服务器后启动客户端），我自己在自己电脑上用 127.0.0.1 测没事是因为我知道顺序，但我的队友不知道，老是在我没打开前打开-_-|||。

分工：我（陆超）负责写代码和测试。潘豪负责测试。