计算机网络实验报告2

网络空间安全专业 20337251 伍建霖

1 实验内容

（1）完成实验教程实例3-2的实验(考虑局域网、互联网两种实验环境), 回答实验提出的问题及实验思考(P103)。

（2）注意实验时简述设计思路。

（3）引起UDP丢包的可能原因是什么？

2 实验分析&思路

实验3-2需要实现一个基于UDP的丢包统计程序， 思路如下：

2.1 服务端

1、WSAStartup()加载套接字库，socket()创建套接字;

2、设置sockaddr_in中的端口和IP地址，bind()绑定套接字和sockaddr_in;

3、recvfrom()接收远程主机经指定的socket传来的数据;

4、sendto()和客户端进行通信;

5、等待另一个连接请求；

6、closesocket()关闭套接字，WSACleanup()关闭加载的套接字库；

2.2 客户端

1、WSAStartup()加载套接字库，socket()创建套接字；

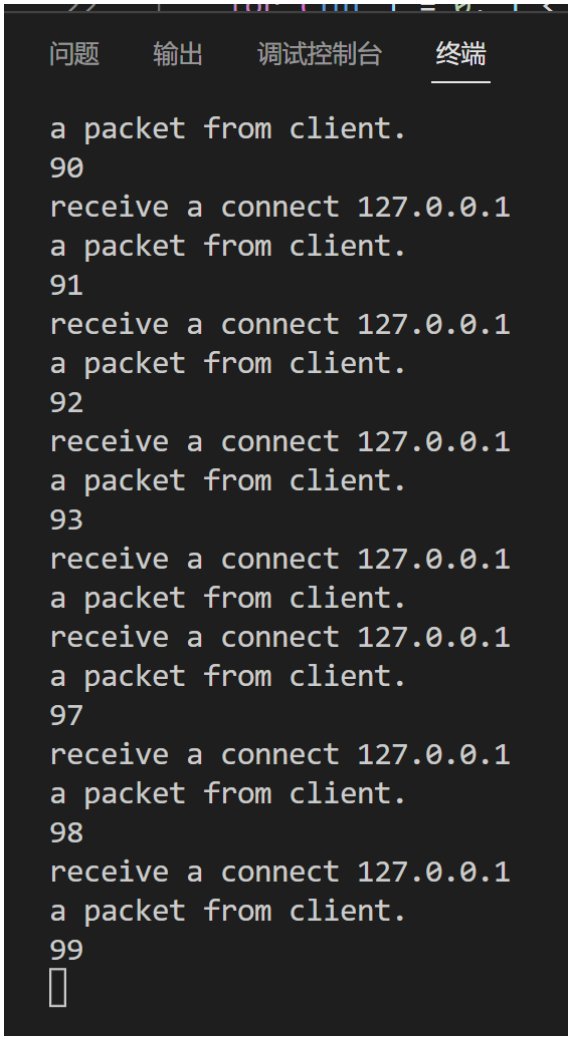
2、设置sockaddr_in中的端口和IP地址;

3、send()/recvfrom()和服务器进行通信；

4、closesocket()关闭套接字，WSACleanup()关闭加载的套接字库；

3 实验截图

3.1 服务端


问题
输出
调试控制台
终端
a packet from client.
90
receive a connect 127.0.0.1
a packet from client.
91
receive a connect 127.0.0.1
a packet from client.
92
receive a connect 127.0.0.1
a packet from client.
93
receive a connect 127.0.0.1
a packet from client.
receive a connect 127.0.0.1
a packet from client.
97
receive a connect 127.0.0.1
a packet from client.
98
receive a connect 127.0.0.1
a packet from client.
99

3.2 客户端

```
问题    输出    调试控制台    终端

85
a packet from server.
86
a packet from server.
87
a packet from server.
88
a packet from server.
89
a packet from server.
90
a packet from server.
91
95
a packet from server.
96
a packet from server.
97
a packet from server.
98
a packet from server.
99
(base) PS D:\CodeField\ComputerNetwork\lab>
```

4 实验细节

4.1 sockaddr_in

```
struct sockaddr_in
{
    sa_family_t         sin_family;     //地址族（Address Family）
    uint16_t            sin_port;       //16 位 TCP/UDP 端口号
    struct in_addr      sin_addr;       //32 位 IP 地址
    char                sin_zero[8];    //不使用
};
```

该结构体中提到的另一个结构体in_addr定义如下，它用来存放32位IP地址。

```
struct in_addr
{
    In_addr_t           s_addr;         //32 位 IPv4 地址
};
```

4.2 sendto()

定义函数：int sendto(int s, const void * msg, int len, unsigned int flags, const struct sockaddr * to, int tolen);

函数说明：sendto()用来将数据由指定的socket传给对方主机。参数s为已建好连线的socket，如果利用UDP协议则不需经过连线操作。参数msg指向欲连线的数据内容，参数flags一般设0，详细描述请参考send()。参数to用来指定欲传送的网络地址，结构sockaddr请参考bind()。参数tolen为sockaddr的结果长度。

4.3 recvfrom()

函数定义：int recvfrom(int s, void *buf, int len, unsigned int flags, struct sockaddr *from, int *fromlen);

函数说明：recvfrom()用来接收远程主机经指定的socket传来的数据，并把数据存到由参数buf指向的内存空间，参数len为可接收数据的最大长度。参数flags一般设0。参数from用来指定欲传送的网络地址，结构sockaddr请参考bind()。参数fromlen为sockaddr的结构长度。

4.4 服务端的IP地址

使用htonl(INADDR_ANY)来自动获得IP地址

4.5 UDP和TCP的小区别

UDP无需在连接状态下交换数据，因此基于UDP的服务器端和客户端也无需经过连接过程。也就是说，不必调用listen()和accept()函数。UDP中只有创建套接字的过程和数据交换的过程。

4.6 UDP丢包原因及解决办法

1、接收端处理时间过长导致丢包：调用recvfrom()接收端收到数据后，处理数据花了一些时间，处理完后再次调用recvfrom()，在这二次调用间隔里，发过来的包可能丢失。对于这种情况可以修改接收端，将包接收后存入一个缓冲区，然后迅速返回继续recvfrom()。

2、发送的包巨大丢包：虽然sendto()会帮你做大包切割成小包发送的事情，但包太大也不行。例如超过50K的一个UDP包，不切割直接通过sendto()发送也会导致这个包丢失。这种情况需要切割成小包再逐个send。

3、发送的包较大，超过接受者缓存导致丢包：包超过mtu size数倍，几个大的UDP包可能会超过接收者的缓冲，导致丢包。这种情况可以设置socket接收缓冲。以前遇到过这种问题，我把接收缓冲设置成64K就解决了。

4、发送的包频率太快：虽然每个包的大小都小于mtu size 但是频率太快，例如40多个mut size的包连续发送中间不sleep，也有可能导致丢包。这种情况也有时可以通过设置socket接收缓冲解决，但有时解决不了。所以在发送频率过快的时候还是考虑sleep一下吧。

5、局域网内不丢包，公网上丢包。这个问题我也是通过切割小包并sleep发送解决的。如果流量太大，这个办法也不灵了。总之UDP丢包总是会有的，如果出现了用我的方法解决不了，还有这个几个方法：要么减小流量，要么换TCP协议传输，要么做丢包重传的工作。

5 代码

```
// server
#include <stdio.h>
#include <winsock2.h>

#pragma comment(lib, "ws2_32.lib")

int main(int argc, char *argv[])
{
    WSADATA wsaData;
    WORD sockVersion = MAKEWORD(2, 2);
    if (WSAStartup(sockVersion, &wsaData) != 0)
    {
        return 0;
    }
```

```
    SOCKET serSocket = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
    if (serSocket == INVALID_SOCKET)
    {
        printf("socket error");
        return 0;
    }

    sockaddr_in serAddr;
    serAddr.sin_family = AF_INET;
    serAddr.sin_port = htons(8888);
    serAddr.sin_addr.S_un.S_addr = INADDR_ANY;
    if (bind(serSocket, (sockaddr *)&serAddr, sizeof(serAddr)) == SOCKET_ERROR)
    {
        printf("bind error");
        closesocket(serSocket);
        return 0;
    }

    sockaddr_in remoteAddr;
    int nAddrLen = sizeof(remoteAddr);
    int count = 0;
    while (true)
    {
        char recvData[255];
        int ret = recvfrom(serSocket, recvData, 255, 0, (sockaddr *)&remoteAddr,
&nAddrLen);
        if (ret > 0)
        {
            recvData[ret] = 0x00;
            printf("receive a connect %s \r\n", inet_ntoa(remoteAddr.sin_addr));
            printf(recvData);
            printf("%d\n", count);
            count++;
        }
        const char *sendData = "a packet from server. \n";
        sendto(serSocket, sendData, strlen(sendData), 0, (sockaddr
*)&remoteAddr, nAddrLen);
    }
    closesocket(serSocket);
    WSACleanup();
    return 0;
}
```

```
// client
#include <stdio.h>
#include <winsock2.h>

#pragma comment(lib, "ws2_32.lib")

int main(int argc, char *argv[])
{
    WORD socketVersion = MAKEWORD(2, 2);
    WSADATA wsaData;
    if (WSAStartup(socketVersion, &wsaData) != 0)
    {
```

```
        return 0;
    }
    SOCKET sclient = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);

    sockaddr_in sin;
    sin.sin_family = AF_INET;
    sin.sin_port = htons(8888);
    sin.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");
    int len = sizeof(sin);

    for (int i = 0; i < 100;)
    {
        const char *sendData = "a packet from client.\n";
        sendto(sclient, sendData, strlen(sendData), 0, (sockaddr *)&sin, len);

        char recvData[255];
        int ret = recvfrom(sclient, recvData, 255, 0, (sockaddr *)&sin, &len);
        if (ret > 0)
        {
            recvData[ret] = 0x00;
            printf(recvData);
            printf("%d\n", i);
            i++;
        }
    }

    closesocket(sclient);
    WSACleanup();
    return 0;
}
```