计算机网络实验报告

警示

1. 实验报告如有雷同，雷同各方当次实验成绩均以 0 分计。
2. 当次小组成员成绩只计学号、姓名登录在下表中的。
3. 在规定时间内未上交实验报告的，不得以其他方式补交，当次成绩按 0 分计。
4. 实验报告文件以 PDF 格式提交。

院系	数据科学与计算机学院	班 级	电子政务	组长	刘硕
学号	16340154	16340148	16340171	15331183	
学生	刘硕	刘虹奇	聂博业	梁峻华	

编程实验

【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？

本次实验完成后，请根据组员在实验中的贡献，请实事求是，自评在实验中应得的分数。（按百分制）

一、实验过程

1. **设计思路**:

实验的代码部分分为两部分，客户端程序以及服务器程序。

1）客户端程序编程流程：

- 创建套接字（socket）
- 将套接字和 IP 地址、端口号绑定在一起（bind）
- 等待客户端发起数据通信（recvfrom/recvto）
- 关闭套接字

2）服务器程序编程流程：

- 创建套接字（socket）
- 向服务器发起通信（recvfrom/recvto）
- 关闭套接字

2. **实验步骤**：

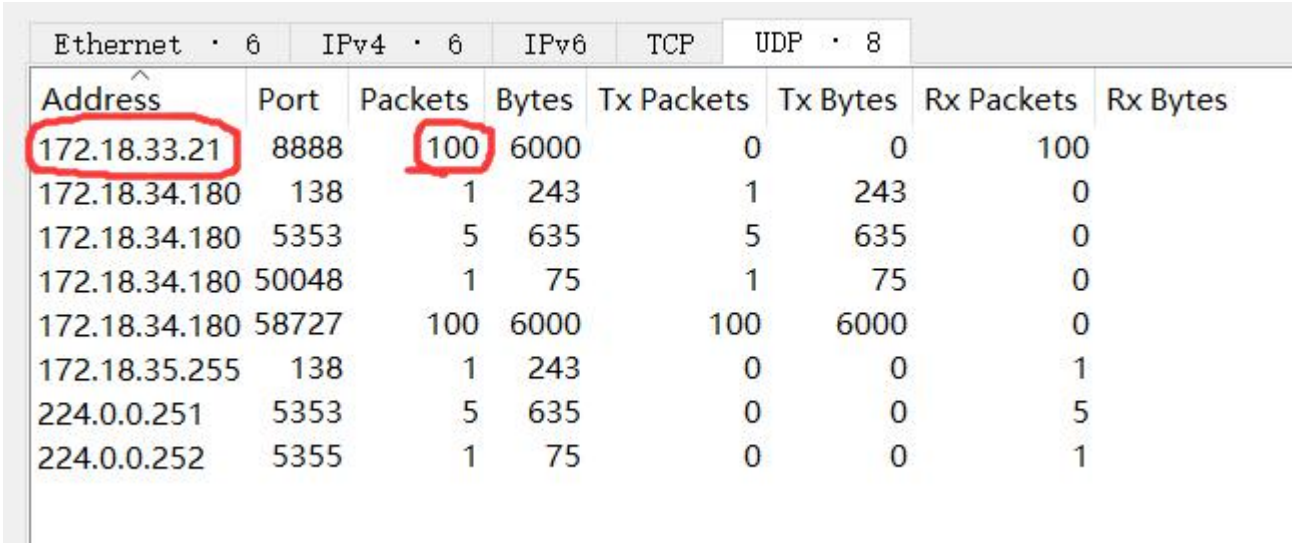
1）通过对老师给的 PPT 与书上的有关 TCP 协议的 socket 编程事例进行研究，查找资料和博客，了解 UDP 客户端与服务器模式的工作原理，以及在编程中二者的不同。

2）编写 UDP 客户端与服务器模式的通信程序。

3）调试并运行自己编写的实现程序，对异常情况进行分析，并且对实验中出现的问题进行总结。

• 要在局域网的环境下测试 udp 丢包的统计，所以我们把服务端放到了宿舍的一台 PC 上，客户端在宿舍的另一台 PC 中，然后在服务端上运行 wireshark 软件来统计收到到来自客户端的 udp 数据包。

```
received: Hello Server 93

current num: 94
Server is running.....
received: Hello Server 94

current num: 95
Server is running.....
received: Hello Server 95

current num: 96
Server is running.....
received: Hello Server 96

current num: 97
Server is running.....
received: Hello Server 97

current num: 98
Server is running.....
received: Hello Server 98

current num: 99
Server is running.....
received: Hello Server 99

current num: 100
Server is running.....
```

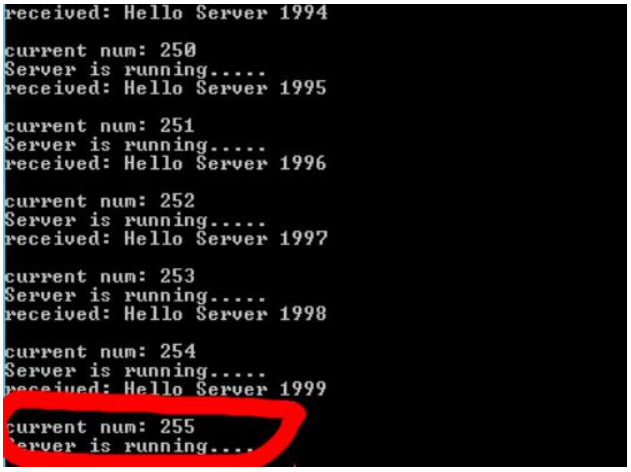


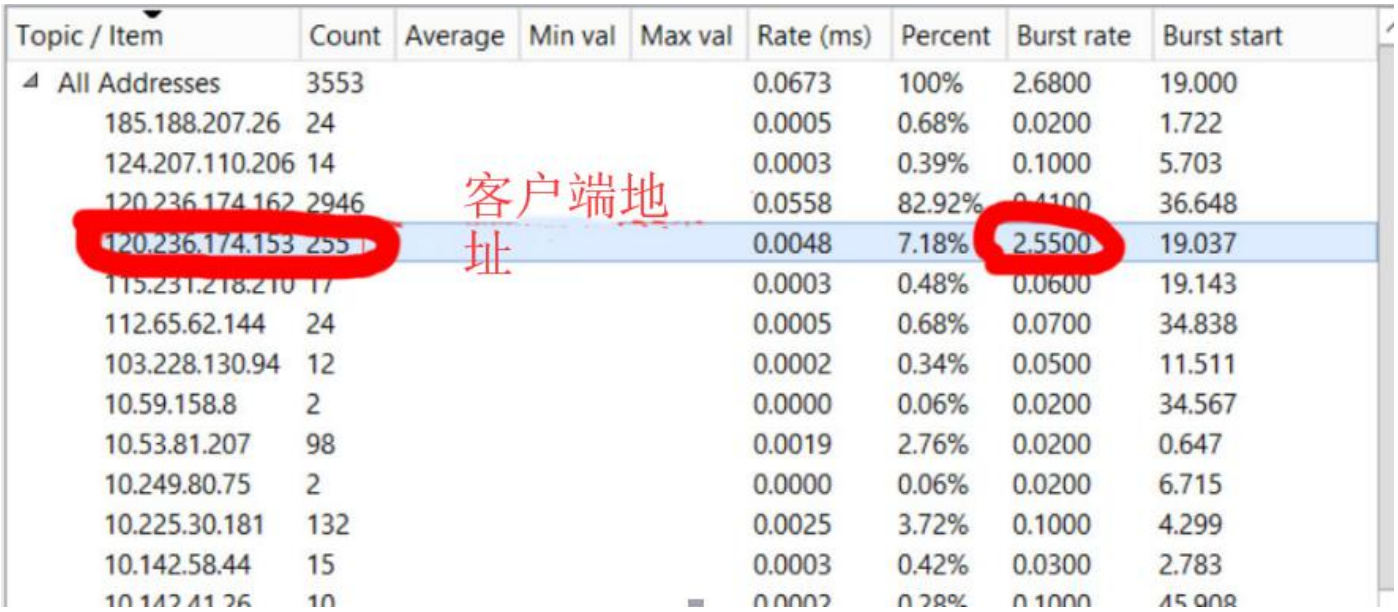
Ethernet · 6 | IPv4 · 6 | IPv6 | TCP | UDP · 8

Address	Port	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes
172.18.33.21	8888	100	6000	0	0	100	
172.18.34.180	138	1	243	1	243	0	
172.18.34.180	5353	5	635	5	635	0	
172.18.34.180	50048	1	75	1	75	0	
172.18.34.180	58727	100	6000	100	6000	0	
172.18.35.255	138	1	243	0	0	1	
224.0.0.251	5353	5	635	0	0	5	
224.0.0.252	5355	1	75	0	0	1	

我们发现，在局域网的情况下，用 UDP 协议来传输 100 个包并没有发生丢包的情况。之后我们又将数量扩大为 10000 个数据包，这测传输的丢包率为百分之三十，可见在庞大的数据传输下，丢包现象是会产生的。

• 要在互联网的环境下测试 UDP 丢包的统计，所以我们把服务端放到了远程服务器上，客户端在校园网中，然后在远程服务器上运行 wireshark 软件来统计接收到的客户端发来的 UDP 数据包。





Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
All Addresses	3553				0.0673	100%	2.6800	19.000
185.188.207.26	24				0.0005	0.68%	0.0200	1.722
124.207.110.206	14				0.0003	0.39%	0.1000	5.703
120.236.174.162	2946				0.0558	82.92%	0.4100	36.648
120.236.174.153	255				0.0048	7.18%	2.5500	19.037
115.231.218.210	17				0.0003	0.48%	0.0600	19.143
112.65.62.144	24				0.0005	0.68%	0.0700	34.838
103.228.130.94	12				0.0002	0.34%	0.0500	11.511
10.59.158.8	2				0.0000	0.06%	0.0200	34.567
10.53.81.207	98				0.0019	2.76%	0.0200	0.647
10.249.80.75	2				0.0000	0.06%	0.0200	6.715
10.225.30.181	132				0.0025	3.72%	0.1000	4.299
10.142.58.44	15				0.0003	0.42%	0.0300	2.783
10.142.41.26	10				0.0002	0.28%	0.1000	45.908

所以在互联网环境下 UDP 的丢包率为 1 - 255 / 2000 = 87.25%。

3. **引起 UDP 丢包的可能原因：**

1）UDP 包途径的路由器负载过高，有过多的包需要处理，所以可能会把某些包给丢掉。

2）服务器负载过高，处理时间过长，导致丢包。

3）客户端发包频率过快，或者发送的报文太大。

二、实验思考

1.说明在实验中遇到的问题和解决方法：

1）客户端接收信息失败，找不出原因：后来将代码改成即使 recvfrom 返回值为-1，也打印 recBUFF 中的消息，发现 recBUFF 的前面部分是“客户端…”，而后面部分则成了乱码。于是推断是 recBUFF 数据缓冲区的大小不够大。修改缓冲区的大小后，成功接收客户端信息

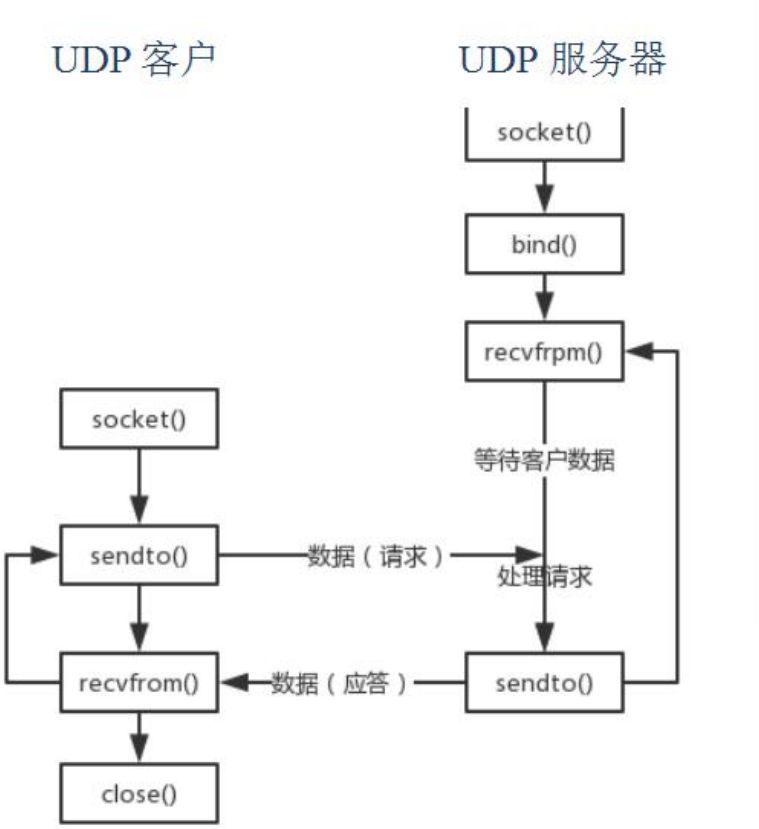
2）为什么客户端不需绑定 socket，而服务器端则需要绑定 socket：因为需要向服务器端分配一个端口，客户

端才能向指定端口发起请求。而客户端的端口有系统随机分配，如果服务期端的端口也是随机分配的，则客户端无法知道该向哪个端口发起连接请求。

3）关于 INADDR_ANY：指定地址为 0.0.0.0，这个地址表示的是任意地址，就是本机的所有 IP，因为有些电脑不止一块网卡，多个网卡的情况下就表示本机的所有网卡 IP 地址。INADDR_ANY 是指监听本机任意一个 IP 地址所收到的信号。

2.给出程序详细的流程图和对程序关键函数的详细说明：

1）程序流程图:



2）程序关键函数的详细说明:

• Socket 创建套接字

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

domain 指明地址簇类型。

type 指明 socket 的类型。

protocol 指明数据传输协议，该参数取决于 af 和 type 参数的类型。

返回值：如果不出错，socket 函数将返回 socket 的描述符（句柄），否则，将返回 INVALID_SOCKET。

• Bind：服务端将 socket 与地址关联

int bind(OCKET s,const struct sockaddr* name,int namelen);

S：指定一个未绑定的 socket。

Name：指向 sockaddr 地址的指针，该结构含有 IP 和 PORT。

Namelen：参数 name 的字节数。

返回值：无错误返回 0，有错误返回 SOCKET_ERROR。

• sendto：使用指定的 SocketFlags，将指定字节数的数据发送到指定的终结点

int sendto (SOCKET s, char* buf, int size, int flags, sockaddr *toaddr, int tolen);

S： 套接字。

Buff： 待发送数据的缓冲区。
Size： 缓冲区长度。
Flags： 调用方式标志位, 一般为 0, 改变 Flags，将会改变 Sendto 发送的形式。
toaddr：（可选）指针，指向目的套接字的地址。
tolen：所指地址的长度。
返回值：如果成功，则返回发送的字节数，失败则返回 SOCKET_ERROR。

- **Recvfrom：接收一个数据报并保存源地址**

int recvfrom(SOCKET s, char * buf, int len, int flags,
sockaddr * from, int* fromlen);
s：标识一个已连接套接口的描述字。
buf：接收数据缓冲区。
len：缓冲区长度。
flags：调用操作方式。
from：（可选）指针，指向装有源地址的缓冲区。
fromlen：（可选）指针，指向 from 缓冲区长度值。
返回值：若无错误发生，recvfrom()返回读入的字节数。如果连接已中止，返回 0。否则的话，返回 SOCKET_ERROR 错误。

- **Close：关闭一个套接字**

int closesocket(SOCKET s);
s：一个套接口的描述字。
返回值：如无错误发生，则返回 0。否则的话，返回 SOCKET_ERROR 错误。

3.**使用 Socket API 开发通信程序中的客户端程序和服务端程序时，各需要哪些不同的函数：**

客户端程序需要的函数：socket(); recvfrom(); sendto(); close();
服务端程序需要的函数：socket(); bind(); recvfrom(); sendto();

4.**解释 connet(),bind()等函数中 struct sockaddr*addr 参数各个部分的含义，并用具体的数据举例说明：**

*addr 实际指向的是一个 sockaddr_in 的结构体，这个结构体如下：

```
struct sockaddr_in {
short int          sin_family;                /* Address family */
unsigned short int      sin_port;             /* Port number */
struct   n_addr     sin_addr;                 /* Internet address */
unsigned char       sin_zero[8];              /* Same size as struct sockaddr */
};
struct in_addr {
unsigned long s_addr;
};
```

1）意义：

- sin_family 是指选取可用网络的类型，指代协议族，在 socket 编程中只能是 AF_INET。
- sin_port 指端口（使用网络字节顺序），在 bind 中就是服务器要把自己的哪个端口开放用于接收数据，在 connect 中就是指出客户端要连接服务器的哪个端口，用于通讯。
- sin_addr 存储 IP 地址，使用 in_addr 这个数据结构，bind 的时候，这个就是包含本机的 IP 地址一个结构体，connect 的时候就是包含服务器的 IP 地址一种结构体其中 IP 地址就是 sin_addr.S_un.S_addr。
- sin_zero 是为了让 sockaddr 与 sockaddr_in 两个数据结构保持大小相同而保留的空字节。

2）例子：

如 client 代码中绑定前的参考设定

• 可用网络的类型：

serverAddr.sin_family = AF_INET;

• 将一个 16 位数从主机字节顺序转换成网络字节顺序，此处设欲绑定的端口号为 13：

serverAddr.sin_port = htons(13);

• 存储 IP 地址 127.0.0.1:

serverAddr.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");

5.说明面向连接的客户端和非连接的客户端在建立 Socket 时有什么区别：

用 socket 函数创建一个 socket 用于面向连接的网络协议通讯时，函数参数我们通常填为 SOCK_STREAM。即 socket(PF_INET, SOCK_STREAM, 0)，这表示建立一个 socket 用于流式网络通讯。SOCK_STREAM 这种的特点是面向连接的，即每次收发数据之前必须通过 connect 建立连接，也是双向的，即任何一方都可以收发数据，协议本身提供了一些保障机制保证它是可靠的、有序的，即每个包按照发送的顺序到达接收方。 而在 UDP 网络协议中，我们使用的建立 socket 函数的参数则是 SOCK_DGRAM。它是无连接的，不可靠的，因为通讯双方发送数据后不知道对方是否已经收到数据，是否正常收到数据。任何一方建立一个 socket 以后就可以用 sendto 发送数据，也可以用 recvfrom 接收数据。

6.说明面向连接的客户端和面向非连接的客户端在收发数据时有什么区别。在非连接的客户端又是如何判断数据发送结束的：

在面向 UDP 的 socket 程序中，在发送和接受数据时会用到 write/send/sendto 和 read/recv/recvfrom，同时也会用：send，recv，但在发送数据时，如果用 sendto 的话，就不用 connect 了。但是，在面向 TCP 的 sockert 程序中，在发送数据时，即使 sendto，也必须 connect，也就是说 connect 这一步是必不可少的。无论是 TCP 还是 UDP，默认情况下创建的都是阻塞模式（blocking）的套接字，执行到 accept，connect，write/send/sendto，read/recv/recvfrom 等语句时，会一直等待。

7. 比较面向连接的通信和无连接通信，它们各有什么优点和缺点？适合在那种场合下使用：

面向链接通信的优点：可靠，稳顶，比如 TCP 的可靠体现在传递数据之前的三次握手来建立连接，而且在数据传递时，有确认、窗口、重传、拥塞控制机制，在数据传完后，还会断开连接用来节约系统资源。缺点： 效率低，占用系统资源高，而且在传递数据之前，要先建连接，这会消耗时间，而且在数据传递时，确认机制、重传机制、拥塞控制机制等都会消耗大量的时间，而且要在每台设备上维护所有的传输连接，事实上，每个连接都会占用系统的 CPU、内存等硬件资源。而且，因为有了三次握手的机制，这些也导致容易被人利用，实现网络攻击。面向非链接的客户端的优点：比 TCP 稍安全且更快，UDP 没有 TCP 的握手、确认、窗口、重传、拥塞控制等机制，是一个无状态的传输协议，所以它在传递数据时非常快，同样，它被攻击者利用的漏洞就要少一些。但 UDP 也是无法避免攻击的。缺点：不可靠，不稳定，因为 UDP 并没有没有相对可靠的机制，在数据传递时，如果网络质量不好，就会很容易丢包。TCP 往往用于一些要求可靠的应用，比如 HTTP，FTP 等传输文件的协议，POP，SMTP 等邮件传输的协议。UDP 要求网络通讯速度能尽量的快，应用比如用微信发送语音，传输即时视频，TFTP 等。

8.实验过程中使用 Socket 的时候是工作在阻塞方式还是非阻塞方式，通过网络检索阐述这两种操作方式的不同：

实验过程中使用 socket 的时候是工作在阻塞方式。阻塞调用是指调用结果返回之前，当前线程会被挂起。函数只有在得到结果之后才会返回。非阻塞指在不能立刻得到结果之前，该函数不会阻塞当前线程，而会立刻返回。

计算机网络实验报告

学号	学生	自评分
16340154	刘硕	95
16340148	刘虹奇	95
16340171	聂博业	95
15331183	梁峻华	95

【交实验报告】

上传实验报告：ftp://222.200.180.109/　　截止日期（不迟于）：1 周之内

上传包括两个文件：

（1）小组实验报告。上传文件名格式：小组号_ Ftp 协议分析实验.pdf （由组长负责上传）

例如: 文件名“10_ Ftp 协议分析实验.pdf”表示第 10 组的 Ftp 协议分析实验报告

（2）小组成员实验体会。每个同学单独交一份只填写了实验体会的实验报告。只需填写自己的学号和姓名。

文件名格式：小组号_学号_姓名_ Ftp 协议分析实验.pdf （由组员自行上传）

例如: 文件名“10_05373092_张三_ Ftp 协议分析实验.pdf”表示第 10 组的 Ftp 协议分析实验报告。

注意：不要打包上传！