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	计算机学院	班 级	人工智能与大数据	组长	陈欣宇
学号	21307347	21307350	21307100		
学生	陈欣宇	高宇	陈华清		

编程实验

【实验内容】

（1） 参考教材 P98 例 3-1 内容，基于 P67 图 2-10 实验拓扑，完成 P103 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。

实验要求：

局域网：发送 100 个 1000 字节的数据包接收到了 67 个，丢失 33 个

校园网：发送 100 个 1000 字节的数据包接收到了 95 个，丢失 5 个

```
count= 58
count= 59
count= 60
count= 61
count= 62
count= 63
count= 64
count= 65
count= 66
count= 67
```

```
count= 86
count= 87
count= 88
count= 89
count= 90
count= 91
count= 92
count= 93
count= 94
count= 95
```

用 wireshark 捕获数据包并分析，以下对局域网传送数据进行分析

1、客户端抓包部分数据，用 ip.dst==169.254.69.95 过滤

No.	Time	Source	Destination	Protocol	Length	Info
7	11.278164	169.254.238.10	169.254.69.95	UDP	1042	57036 → 9999 Len=1000
8	11.278267	169.254.238.10	169.254.69.95	UDP	1042	57036 → 9999 Len=1000
9	11.278305	169.254.238.10	169.254.69.95	UDP	1042	57036 → 9999 Len=1000
10	11.278346	169.254.238.10	169.254.69.95	UDP	1042	57036 → 9999 Len=1000
11	11.278375	169.254.238.10	169.254.69.95	UDP	1042	57036 → 9999 Len=1000

上面的数据表明发送源 ip：169.254.238.10，目的 ip：169.254.69.95，传输协议 UDP，传输 1042 字节的数据，从 57036 端口传输到 9999 端口，data 大小为 1000 字节

2、客户端抓包部分数据，用 ip.src==169.254.238.10 过滤

No.	Time	Source	Destination	Protocol	Length	Info
4	4.225571	169.254.238.10	169.254.69.95	UDP	1042	57036 → 9999 Len=1000
5	4.225571	169.254.238.10	169.254.69.95	UDP	1042	57036 → 9999 Len=1000
6	4.225571	169.254.238.10	169.254.69.95	UDP	1042	57036 → 9999 Len=1000
7	4.225571	169.254.238.10	169.254.69.95	UDP	1042	57036 → 9999 Len=1000

分析与客户端相同

此外，可以看见没有建立连接的过程，服务端持续侦听某端口，客户端直接向服务端侦听的端口发送数据，UDP

协议没有流控制，没有应答确认机制，不能解决丢包、重发、错序问题，因此它是不可靠的传输。但是它不需要建立连接，也不需要释放连接，减少了开销和发送数据的时延。

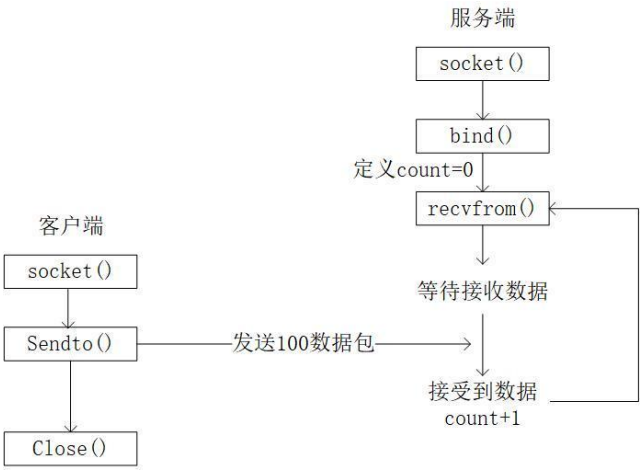
a. 问题和解决方法

1. 对 socket 函数调用缺乏使用经验，参照书上代码解决

2. 在统计丢包率中，在 100 个数据包里面一直不会出现丢包的情况，无法进行分析，通过查找 udp 丢包的原因，主要有发送频率过高 recv 处理不过来、发送包过大超过接受者缓存导致的丢包，因此我们通过加大发送包的大小来加大了丢包的概率

b. 流程图和关键函数说明

流程图如下：



代码通过 python 实现：

服务器和客户端都使用到 socket()创建套接字，设置了 AF_INET，SOCK_DGRAM 表示“无连接的套接字”，表示 IPV4 的 UDP 网络协议

```
from socket import *
serverPort = 9999
serverSocket = socket(AF_INET,SOCK_DGRAM)
serverSocket.bind(('',serverPort))
print("The server is ready to receive")
count = 0
while True:
    message, clineAddress = serverSocket.recvfrom(2048)
    count+=1
    modifiedMessage = message.decode().upper()
    # print(modifiedMessage)
    print("count=",count)
```

```
1  from socket import *
2  #serverName = '172.16.26.2'
3  serverName = '169.254.69.95'
4  serverPort = 9999
5  clientSocket = socket(AF_INET, SOCK_DGRAM)
6  for i in range(100):
7      message = '0'.zfill(1000)
8      if message=='break':break
9      clientSocket.sendto(message.encode(), (serverName, serverPort))
10 clientSocket.close()
```

c. 客户端和服务端需要的不同函数

在客户端中使用了 sendto()函数，函数参数有发送的数据、服务器 ip 地址、服务器端口号，能将指定数据发送至服务器端，发送完最后用 close 关闭套接字

在服务端中使用了 recvfrom()函数，参数有接收数据缓冲区大小，接受发送至本端口号的数据，使用 bind()将该套接字绑定 2000 以上端口

d. connect()，bind()中 struct sock addr * addr 参数各个部分含义

int connect(socket, struct sockaddr * addr, int addr_len)

connect 一般是客户端调用的，用来建立与服务器的连接，struct sockaddr * addr 结构体中包含了两个变量：

unsigned short sa_family；　用来表示地址族，即 ip 类型，ipv4、ipv6 等

char sa_data[14];　　用来指明服务器 ip 地址及端口号来建立连接请求

举例：

```
struct sockaddr serv_addr;
memset(&serv_addr, 0, sizeof(serv_addr));
serv_addr.sa_family = AF_INET;  //使用 IPv4 地址
serv_addr.sa_data = string("172.0.0.1:1234");  //具体的 IP 地址和端口号
```

该例表明了向 IP 为 172.0.0.1 的服务器的 1234 端口发送连接请求

e. 面向连接的客户端和面向非连接的客户端在建立 socket 时有什么区别

面向连接的 socket 套接字的 type 类型为流式，面向连接的比特流，用于 TCP 通信；面向非连接的 socket 套接字的 type 类型为数据报式，无连接，用于 UDP 通信。

面向连接的客户端建立 socket 后要与服务端用 connect()建立连接后才可以发数据，面向连接的客户端建立 socket 后可以直接发数据

f. 面向连接的客户端和面向非连接的客户端收发数据时的区别，面向非连接的客户端如何判定数据发送结束

面向连接的客户端在收发数据之前必须用 connect()建立一个连接才能进行收发数据，而面向无连接的客户端　在收发数据之前不需要建立连接。面向连接的客户端收发的数据是可靠的、有序的，而非连接的客户端收发的数据可能是无序的。数据收发完成后，面向连接的客户端需要与服务器之间断开连接，而非连接客户端不需要。非连接客户端调用 sendto()函数时会有一个返回值，得到返回值说明数据发送结束。

g. 面向连接的通信和非连接通信的优缺点以及适用场合

无连接通信的优点是传输效率高，收发数据快，响应速度快，缺点是不可靠，有丢失数据包、捣乱数据的风　险，接收的数据也可能是无序的；

面向连接通信的优点是非常可靠，万无一失，数据发送和接收都是有序的，丢包、数据混乱的风险小，缺点是传输效率低，耗费资源多，收发数据耗费的时间长。

面向连接通信适用于对可靠性要求比较高，要求必须数据包能够完整无误地送达的场合，且对于数据收发时间没有太大的要求，比如 HTTP、FTP 、电子邮件等；面向非连接的通信适用于对数据的可靠性要求不高，接受部分混乱数据，但对效率和收发数据速度有很高的需求的场合，比如 DNS、即时聊天工具、多媒体视频等。

h. socket 是阻塞方式还是非阻塞方式，有什么不同

实验过程中使用的 socket 是阻塞方式。

阻塞模式和非阻塞模式的主要区别在于无请求来到时，程序的运行方式。阻塞模式下 socket 会一直停在接收函数，直到有请求到来才会继续向下进行处理。而非阻塞模式下，运行接收函数时如果有请求，则会接收请求，如果无请求，会返回一个负值或是一个错误提醒发送端，但不等待，而是继续向下运行。

（2） 注意实验时简述设计思路。

编写服务端客户端两个文件，分别建立套接字并准备接收和发送数据，在实验室两台连接主机上，根据该主机的 IP 地址，设置好并开始传输 100 个数据包，在接收端统计收到的数据包个数，期间用 wireshark 抓包并分析传输数据的特征。

（3）引起 UDP 丢包的可能原因是什么？

1、接收端处理数据包的时间过长，然后下一个数据包就到来了，可能会导致丢包

2、发送端的发送的数据包过于巨大，在传输过程中丢失，同时也会接收端处理时间变长，都有可能丢包，或是数据包被分组发送，但其中的某一块丢失了，则整个数据包都有丢包的风险。

3、发送端发送的数据包较大，超过了接受者缓存，接收端无法接收导致丢包。

4、发送端发送数据包的频率太快，接收端还没来得及接收上一个数据包，下一个数据包就到来了，也会导致其中的某些数据包丢失。

5、网络流量太大，网络连接不佳，防火墙没有关闭，导致数据无法接收等情况下也可能会导致丢包。

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）

学号	学生	自评分
21307347	陈欣宇	90
21307350	高宇	90
21307100	陈华清	90