《网络与通信》课程实验报告

实验 2：Socket 通信编程

姓名	李昀哲	院系	计算机学院	学号	20123101
任课教师	刘通		指导教师	刘通	
实验地点	计 708		实验时间	2022.9.23-9.30	
实验课表现	出勤、表现得分(10)		实验报告得分(40)		实验总分	
	操作结果得分(50)					

实验目的：

1. 掌握 Socket 编程过程；
2. 编写简单的网络应用程序。

实验内容：

利用你选择的任何一个编程语言，分别基于 TCP 和 UDP 编写一个简单的 Client/Server 网络应用程序。具体程序要求参见《实验指导书》。

要求以附件形式给出：

- 系统概述：运行环境、编译、使用方法、实现环境、程序文件列表等；
- 主要数据结构；
- 主要算法描述；
- 用户使用手册；
- 程序源代码；

实验要求：（学生对预习要求的回答）（10 分） 得分：

- Socket编程客户端的主要步骤
  - 通过指明协议族、socket类型，使用*socket.socket()*函数建立*Socket*；
  - 通过服务器域名获得服务器的IP地址，使用*connect()*函数建立链接；
  - 链接完成后，接收从服务器端发送的数据并发送需要传输的数据。
- Socket编程服务器端的主要步骤
  - 通过指明协议族、socket类型，调用*socket.socket()*函数建立Socket；
  - 调用*bind()*函数将其与本机地址以及一个本地端口号绑定；
  - 调用*listen()*函数在相应的socket上监听；
  - 调用*accept()*函数接收客户端链接请求；
  - 接收到请求后，服务器端显示客户端IP地址，并发送相关连接信息；
  - 关闭socket。

Socket通信实验-Python实现源代码：

基于TCP：

客户端：Client

```
def socket_client():
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(('127.0.0.1',6666))
    except socket.error as msg:
        print(msg)
```

```
            sys.exit(1)
        print(s.recv(1024))
        while 1:
            data = input("please input work: ").encode()
            s.send(data)
            print('reveived from server', s.recv(1024))
            if data == 'exit':
                break
        s.close

if __name__ == '__main__':
    socket_client()
```

服务端：Server

```
def socket_server():
    try:
        # 首先调用socket函数创建一个socket
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        # 防止socket server重启端口被占用（socket.error:[Errno 98] Address already in use）
        s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        # 调用bind函数将其与本机地址以及一个端口号绑定
        s.bind(('127.0.0.1', 6666))
        # listen在相应的socket上监听
        s.listen(10)
    except socket.error as msg:
        print(msg)
        sys.exit(1)
    print("waiting connection...")

    while 1:
        conn, addr = s.accept()
        t = threading.Thread(target=deal_data, args=(conn, addr))
        t.start()
# 处理数据
def deal_data(conn, addr):
    print("Accept new connection from {0}".format(addr))
    conn.send(("Welcome to the server.").encode())
    while 1:
        data = conn.recv(1024)
        print('{0} client send data is {1}'.format(addr, data.decode()))
        time.sleep(1)
        if data == 'exit' or not data:
            print(f'{addr} connection close.')
            conn.send(bytes('Connetion closed!'), 'UTF-8')
            break
```

```
            conn.send(bytes(f'Hello, {data}', 'UTF-8'))
        conn.close()

if __name__ == '__main__':
        socket_server()
```

基于UDP：

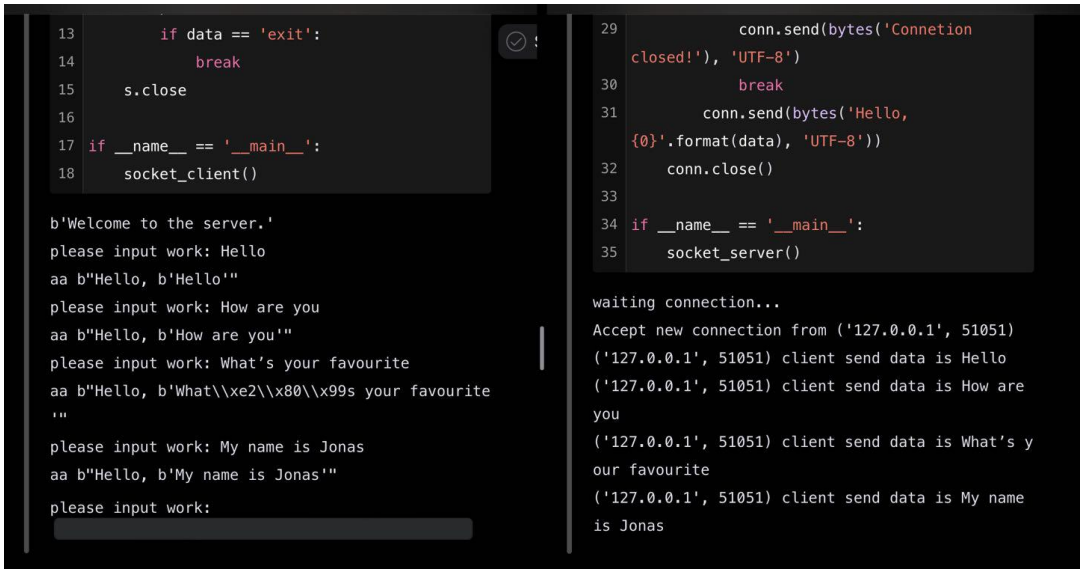
客户端：Client

```
serverName='127.0.0.1'
serverPort=12345
clientSocket=socket(AF_INET,SOCK_DGRAM)
while 1:
        message=input().encode()
        clientSocket.sendto(message,(serverName,serverPort))
        modifiedMessage,serverAddress=clientSocket.recvfrom(2048)
        print(modifiedMessage)
clientSocket.close()
```

服务端：Server

```
from socket import *
serverPort=12345
serverSocket=socket(AF_INET,SOCK_DGRAM)
serverSocket.bind(('127.0.0.1',serverPort))
print('ready')
while 1:
        message,clientAddress=serverSocket.recvfrom(2048)
        modifiedMessage=message.upper()
        print(modifiedMessage)
        serverSocket.sendto(modifiedMessage,clientAddress)
```

如下图所示为通信实验截图，可见客户端和服务端之间实现了数据传输。



实验过程中遇到的问题如何解决的？（10 分）	得分：

问题 1：socket 服务端与客户端之间的连接建立阶段报错。
A: 运行时，应先开启服务端，待准备就绪后，才能开启客户端，否则连接无法建立。

问题 2：连接建立后的瞬间建立消失。
A: 需要在服务端将接收请求写死，即：使用 *while True* 确保服务端始终处于接收状态，除非编写者有意将其停止。

问题 3：服务端无法接收客户端数据。
A: 编写程序时，误将 close()函数写在死循环里，导致每一次接收数据后，当前服务端建立的客户端 socket 会被关闭，导致无法接收。

问题 4：端口冲突。
A: 调用 *setsockopt()*防止 socket server 重启端口被占用。

本次实验的体会（结论）（10 分）	得分：

本实验中使用 Python 分别基于 TCP 和 UDP 编写了一个简单的 Client/Server 网络应用程序。熟悉了 Socket 编程过程以及定义的许多函数和例程。从建立、配置、连接再到最后的数据传输，均在此次实验中实现。编程过程中，也体验到了传输完成时的喜悦和遇到 bug 时的排疑解惑。同时，利用本次实验也对 python 的使用更加熟悉，为后续的课程打下基础。

思考题：（10 分）

思考题 1：（4 分）	得分：

你所用的编程语言在 Socket 通信中用到的主要类及其主要作用。
- Socket 类，用于实现客户端与服务端的通信；
- threading 类，用于创建新的线程处理客户端请求；

思考题 2：（6 分）	得分：

说明 TCP 和 UDP 编程的主要差异和特点。

从定义上看，TCP 是传输控制协议（Transmission Control Protocol），UDP 是用户数据协议（User Data Protocol）。TCP 在传送前需要服务端与客户端建立连接，通过 *send()*和 *recv()* 传送和接收数据；而 UDP 则不需要建立连接即可通信，使用 *sendto()*和 *recvfrom()*即可。

可靠性上，TCP 这种连接导向的更高；而传输速度上，则是 UDP 非连接导向的快。

指导教师评语：

日期：