《网络与通信》课程实验报告

实验 2：Socket 通信编程

<table>
<tr><td>姓名</td><td>严昕宇</td><td>院系</td><td>计算机学院</td><td>学号</td><td>20121802</td></tr>
<tr><td>任课教师</td><td colspan="2">曹晨红</td><td>指导教师</td><td colspan="2">曹晨红</td></tr>
<tr><td>实验地点</td><td colspan="2">计 708</td><td>实验时间</td><td colspan="2">2022 年 9 月 28 日</td></tr>
<tr><td rowspan="2">实验课表现</td><td>出勤、表现得分(10)</td><td></td><td rowspan="2">实验报告得分(40)</td><td rowspan="2"></td><td rowspan="2">实验总分</td><td rowspan="2"></td></tr>
<tr><td>操作结果得分(50)</td><td></td></tr>
</table>

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：
  ① 调用socket库，构造socket对象；
  ② 调用socket.bind()方法，将socket与本机端口绑定；
  ③ 调用socket.connect()方法，连接到服务端。
  ④ 调用socket.recv()方法，接收来自服务单的数据；
  ⑤ 调用socket.send()方法，向服务端发送数据；
  ⑥ 调用socket.close()方法，关闭socket。
- Socket编程服务器端的主要步骤：
  ① 调用socket库，构造socket对象；
  ② 调用socket.bind()方法，将socket与本机端口绑定；
  ③ 调用socket.listen()方法，进行socket监听；
  ④ 调用socket.accpet()方法，等待接收客户端连接，若连接成功，创建新的socket；
  ⑤ 调用socket.recv()方法，接收来自客户端的数据；
  ⑥ 调用socket.send()方法，向客户端发送数据；
  ⑦ 调用socket.close()方法，关闭socket。

实验过程中遇到的问题如何解决的？（10 分）　　得分：

问题 1：如何判断客户端的用户身份是合法的数据？
答：在 SQLite 数据库中事先存储合法用户的数据，在客户端登录时，将其传递的 Hash 解密后的用户和密码进行验证，如果符合，则说明是合法的数据，建立可靠的连接；如果不符合，则拒绝建立连接。
（由于受时间限制，修改口令的功能未能完成）

问题 2：如何设计服务器端与客户端的交互访问界面？
答：本实验采用了 PyQt5 ——创建 Python GUI 应用程序的工具包作为设计 GUI 的工具。
其中主要用到了 QtCore：包含了核心的非 GUI 功能；QtWidgets：模块包含创造经典桌面风格的用户界面提供了一套 UI 元素的类。

问题 3：如何合理分配工程文件的内容？
答：由于最初将所有函数都集中与一个文件中，导致在 Debug 遇到很大阻力和困难。因此决定将项目分为 3 个.py 文件，分别负责创建 socket、服务器端及客户端的建立。通过此方法，减少了第一版项目代码容易遇到的异常情况，大大提高了设计效率。同时这个设计思路容易处理多线程的情况，让项目在运行时出错的几率大大降低。

本次实验的体会（结论）（10 分）	得分：

此次实验为计算机网络课程的第二次实验课，本次实现利用了 Python 的 socket 库实现 socket 的 TCP/UDP 通信，同时利用 PyQt5 库实现图形化界面。相比第一次实验，我感觉项目的操作难度较高。首先需要清晰的设计逻辑，要将项目分成几个模块，妥善安排每个模块之间的交互，因此不仅要对 socket 编程比较熟悉，更要有过硬的代码实力。由于 socket 库的封装较好，TCP/UDP 通信部分实现相对简单。
其次，为了完善用户使用体验，本次实验中使用了 GUI 编程和数据库。我发现图形化界面设计的工程量相比 socket 编程的只增不减。虽然我花费了较多时间设计服务器端与客户端的交互界面，但也从中学会了新的技术。

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。
编程环境：Python 3.9.13
Python 提供了两个级别访问的网络服务：

- 低级别的网络服务支持基本的 Socket，提供了标准的 BSD Sockets API，可以访问底层操作系统 Socket 接口的全部方法。
- 高级别的网络服务模块 SocketServer，提供了服务器中心类，可以简化网络服务器的开发。

主要类：socket
构造：socket.socket([*family*[, *type*[, *proto*]]])
参数：

- *family*: 套接字家族可以是 AF_UNIX 或者 AF_INET。
- *type*: 套接字类型可以根据是面向连接的还是非连接分为 SOCK_STREAM 或 SOCK_DGRAM。
- *protocol*: 一般不填默认为 0。

Socket 对象(内建)方法：

函数	描述
服务器端套接字	
s.bind()	绑定地址（host,port）到套接字， 在 AF_INET 下，以元组（host,port）的形式表示地址
s.listen()	开始 TCP 监听，backlog 指定在拒绝连接之前，操作系统可以挂起的最大连接数量。
s.accept()	被动接受 TCP 客户端连接，(阻塞式)等待连接的到来
客户端套接字	
s.connect()	主动初始化 TCP 服务器连接，。一般 address 的格式为元组（hostname, port），如果连接出错，返回 socket.error 错误
s.connect_ex()	connect()函数的扩展版本,出错时返回出错码,而不是抛出异常
公共用途的套接字函数	
s.recv()	接收 TCP 数据，数据以字符串形式返回，bufsize 指定要接收的最大数据量。flag 提供有关消息的其他信息，通常可以忽略
s.send()	发送 TCP 数据，将 string 中的数据发送到连接的套接字。返回值是要发送的字节数量，该数量可能小于 string 的字节大小
s.sendall()	完整发送 TCP 数据。将 string 中的数据发送到连接的套接字，但在返回之前会尝试发送所有数据。成功返回 None，失败则抛出异常
s.recvfrom()	接收 UDP 数据，与 recv()类似，但返回值是（data, address）。其中 data 是包含接收数据的字符串，address 是发送数据的套接字地址
s.sendto()	发送 UDP 数据，将数据发送到套接字，address 是形式为（ipaddr，port）的元组，指定远程地址。返回值是发送的字节数。
s.close()	关闭套接字
s.getpeername()	返回连接套接字的远程地址。返回值通常是元组（ipaddr, port）
s.getsockname()	返回套接字自己的地址。通常是一个元组(ipaddr, port)
s.setsockopt(level,optname,value)	设置给定套接字选项的值。
s.getsockopt(level,optname[.buflen])	返回套接字选项的值。
s.settimeout(timeout)	设置套接字操作的超时期，timeout 是一个浮点数，单位是秒。值为 None 表示没有超时期。一般，超时期应该在刚创建套接字时设置，因为它们可能用于连接的操作（如 connect()）

s.gettimeout()	返回当前超时期的值，单位是秒，如果没有设置超时期，则返回 None。
s.fileno()	返回套接字的文件描述符。
s.setblocking(flag)	如果 flag 为 0，则将套接字设为非阻塞模式，否则将套接字设为阻塞模式（默认值）。非阻塞模式下，如果调用 recv()没有发现任何数据，或 send()调用无法立即发送数据，那么将引起 socket.error 异常。
s.makefile()	创建一个与该套接字相关连的文件

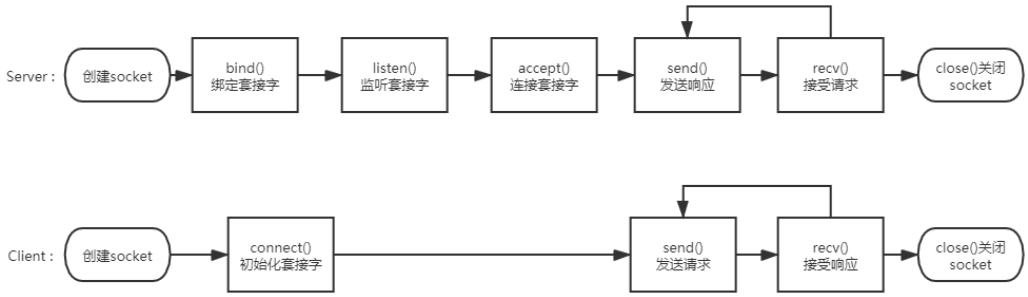
思考题 2：（6 分）　　得分：

说明 TCP 和 UDP 编程的主要差异和特点。

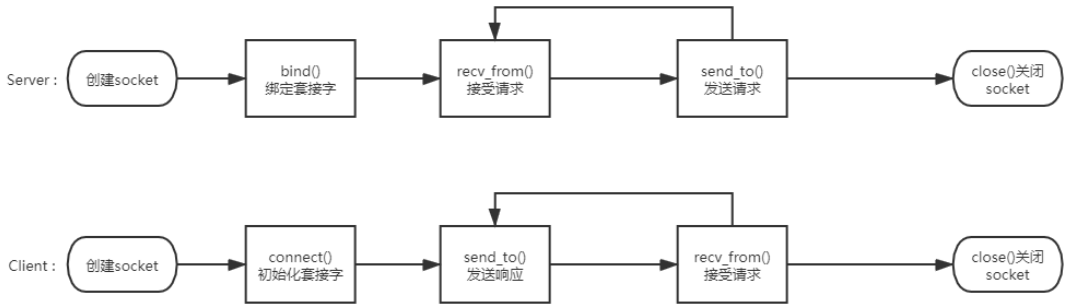
TCP(Transmission Control Protocol)：传输控制协议，UDP(User Datagram Protocol)：用户数据报协议，都是位于传输层的协议，它们的区别主要有以下方面：

① 流程差异

TCP 流程图：



UDP 流程图：



② 数据传输差异

	TCP	UDP
可靠性	可靠	不可靠
连接方式	面向连接	无连接
对象个数	一对一	多对多
头部长度	20Byte~60Byte	8Byte
传输方式	面向字节流	面向报文
传输速度	慢	快
流量控制	滑动窗口	无
拥塞控制	慢开始、拥塞避免、快重传、快恢复	无

③ 应用场景差异

应用层协议	应用	传输层协议
SMTP	电子邮件	TCP
TELNET	远程终端接入	
HTTP	万维网	
FTP	文件传输	
DNS	域名转换	UDP
TFTP	文件传输	
SNMP	网络登录	
NFS	远程文件服务器	

指导教师评语：

日期：