实验报告

|  |  |
| --- | --- |
| 实验名称：UDP网络程序设计 | |
| 实验台号：2 | 实验时间：2019年11月24日 |
| 实验小组：第2组  成员及本次实验分工:於文卓 | |
| 实验目的：  1.进一步学习UDP协议的工作原理  2.学习SOCKET编程的基本方法  3.使用Python编程语言的socket包，实现客户端对服务端的数据传输 | |
| 实验环境说明：  笔记本电脑一台(Win10系统)，Wireshark(Version 3.0.3)，Python（3.6） | |
| 实验过程、步骤（可另附页、使用网络拓扑图等辅助说明）及结果： 1.相关概念简介1.1 UDP UDP用户数据报协议，是一个无连接的简单的面向数据报的运输层协议。UDP不提供可靠性，它只是把应用程序传给IP层的数据报发送出去，但是并不能保证它们能到达目的地。由于UDP 在传输数据报前不用在客户和服务器之间建立一个连接，且没有超时重发等机制，故而传输速度很快。 2.实验步骤  2.1 相关模块介绍 1）创建套接字  和Tcp一样用socket.socket来创建服务套接字，指定套机字的类型和通信类型socket.SOCK\_DGRAM代表是采用udp方式的套接字通信。   |  | | --- | | import socket  server\_socket = socket.socket(AF\_INET, SOCK\_DGRAM) |   2）绑定ip和端口  基于套接字通过bind函数对外公布地址和端口号   |  | | --- | | sockobj.bind((myHost， myPort)) |   3）接受数据  udp收数据用recvfrom函数   |  | | --- | | data, addr = server\_socket.recvfrom(字节数) |   recvfrom的返回值有两个，第一个是收到的数据，第二个是客户端套接字(地址和端口号)，即从那个客户端收到了数据。  4）发送数据  这里的addr是某客户端的套接字(地址和端口号)，可以直接指出也可使用recvfrom的第二个返回值   |  | | --- | | server\_socket.sendto(数据，addr) |   5）关闭服务套接字   |  | | --- | | server\_socket.close() |  2.2 实现UDP多对多数据传输 1）server.py 服务端代码  在服务端中，设置ip为localhost，端口号为50007。当服务端收到客户端传来的消息时，打印消息，并向客户端传送：Got it。   |  | | --- | | from socket import \*  from time import sleep, ctime  host = ""  port = 50007  addr = (host, port)  server = socket(AF\_INET, SOCK\_DGRAM)  server.bind(addr)  print("服务器建立成功")  while True:  data, address = server.recvfrom(1024)  print("从{}中获得数据{}".format(address, data.decode()))  server.sendto("Got it!".encode(), address)  print("\*" \* 20) |   2）客户端代码  客户端想服务器发送字用户输入的数据，发送出成功后，得到服务器的回复Got it！   |  | | --- | | from socket import \*  from time import ctime, sleep  host = "localhost"  port = 50007  addr = (host, port)  client = socket(AF\_INET, SOCK\_DGRAM)  while True:  msg = input("输入要发送的消息:")  print("正在发送...")  sleep(1)  client.sendto(msg.encode(), addr)  print("发送成功")  data, address = client.recvfrom(1024)  print("Server: ", data.decode())  print("\*" \* 20) |   3）实现结果    可以看到创建了一个服务端和三个客户端，服务端收到了客户端发来的消息。客户端发送消息后，收到了来自服务端的Got it！说明UDP通讯建立成功 | |
| 实验总结（遇到的问题及解决办法、体会）： TCP VS UDP 1) TCP  a.基于链接，则需要listen（backlog），指定连接池的大小。  b.基于链接，必须先运行的服务端，然后客户端发起链接请求。  c.如果一端断开了链接，那另外一端的链接也跟着断开(在mac系统中，recv收到的是空，而在windows和linux中，会直接报错，需要在代码中加入异常处理)。  2)UDP  a.无链接，因而无需listen（backlog）  b.无链接，udp的sendinto不用管是否有一个正在运行的服务端，可以己端可以不停地发消息，只不过数据丢失。如图所示，并没有开启服务器，但客户端依然可以发送数据。 | |
| 器材、工具领用及归还负责人：於文卓 | 实验记录人：於文卓 |
| 实验执笔人：於文卓 | 报告协助人：於文卓 |
| 小组成员签名：於文卓 | |
| 验收人： | 成绩评定： |