|  |
| --- |
| 1. **Purpose of experiment** 2. 熟悉HTTP的请求和响应的结构，学会编写简单的HTTP网页server服务器的编写 3. 了解UDP传输协议的工作原理以及结构，学会编写简单的UDP套接字链接server和client并完成两者之间的传输实验 4. 了解FTP传输协议的工作原理以及结构，学会编写简单的TCP套接字链接server和client并完成两者之间的传输实验 5. **Experimental principle** 6. **socket**表示网络通信的端点。socket编程就是通过调用一系列的函数来创建、绑定、监听、接收、发送和关闭socket的过程。使用Python中的socket模块可以使用到一些疯传好的函数来完成套接字编程 7. **Web服务器**：Web服务器的作用是接收客户端（浏览器）的HTTP请求，根据请求返回相应的文件或错误信息。HTTP是一种应用层协议，它的请求和响应都是由起始行、头部和主体三部分组成的文本消息。HTTP使用TCP作为传输层协议，因此需要使用socket编程来建立和管理TCP连接 8. **UDP Pinger**：测试网络延迟，客户端发送一个简单的ping消息给服务器，服务器回复一个相应的pong消息给客户端，并计算客户端发送ping消息和接收pong消息之间的时间差。这个时间差就叫做往返时延（RTT）。UDP是一种无连接、不可靠、面向报文的传输层协议，它只负责将数据报从一台主机发送到另一台主机，不保证数据是否能到达、不考虑顺序和重复性的问题。同时，它还要处理丢包和超时发送的问题并返回错误信息。 9. **FTP protocol**：FTP协议包括两个组成部分，其一为FTP服务器，其二为FTP客户端。其中FTP服务器用来存储文件，用户可以使用FTP客户端通过FTP协议访问位于FTP服务器上的资源。 10. **Content**   **对于这三个实验，首先要做的是：**创建一个socket对象，指定使用IPv4地址（AF\_INET）和数据报套接字（SOCK\_DGRAM）  1.web server：  (1) 绑定socket对象到一个特定的IP地址和端口号，这样就可以在这个地址和端口上监听客户端的连接请求。  (2) 调用listen方法，让socket对象进入监听模式，等待客户端的连接请求。  (3) 调用accept方法，接受一个客户端的连接请求，并返回一个新的socket对象和客户端的地址。这个新的socket对象用来和客户端进行通信，而原来的socket对象继续监听其他客户端的连接请求。  (4) 调用recv方法，从新的socket对象中接收客户端发送的HTTP请求，并解析请求中的文件名。  (5) 根据文件名，从服务器的文件系统中获取相应的文件，并创建一个HTTP响应消息，包含文件内容和一些头部信息。如果文件不存在，则创建一个“404 Not Found”错误消息。然后调用send方法，通过新的socket对象发送HTTP响应消息给客户端。  2.UDP Pinger：  (1) 客户端使用settimeout方法，设置socket对象的超时时间为1秒，即如果在1秒内没有收到服务器回复，则认为数据丢失或超时。  (2)客户端使用sendto方法，向服务器发送包含“Ping”字符串和当前时间的数据，并记录发送时间。  (3) 客户端使用recvfrom方法，尝试从服务器接收数据，并记录接收时间。  (4) 若请求未超时，则服务器返回“Pong”字符串和响应信息  (5) 如果成功接收到服务器回复，则计算RTT为接收时间减去发送时间，并打印出RTT和服务器回复的内容。  (6) 如果没有成功接收到服务器回复，则捕获socket.timeout异常，并打印出“Request timed out”信息。  (7) 客户端重复以上步骤10次，每次发送不同的数据，以测试网络延迟的变化。  (8) 计算丢包的次数，丢包概率和平均往返延迟时间。  3.FTP Protocol：  (1) 绑定socket对象到一个特定的IP地址和端口号，这样就可以在这个地址和端口上监听客户端的连接请求。  (2) 调用listen方法，让socket对象进入监听模式，等待客户端的连接请求。  (3) 客户端使用connect方法，向服务器发起控制连接，并指定服务器的IP地址和端口号。 。  (4) 客户端使用recv方法，从socket对象中接收服务器返回的响应消息，并打印出来。  (5) 客户端使用recv方法，从socket对象中接收服务器返回的响应消息，并解析其中的IP地址和数据端口号。  (6) 客户端使用send方法，向服务器发送一个命令指定要下载的文件。  (7) 服务器打开指定的文件，用read方法读取文件并使用send将文件数据发送回客户端  (8) 客户端使用recv方法，从第一个socket对象中接收服务器返回的响应消息，并打印出来。  (9) 客户端使用recv方法，从第二个socket对象中接收文件内容，写入到一个新建的文件中并保存到本地文件系统。 |
| 1. **Conclusion and discussion**   本实验的目的是通过三个子实验，熟悉HTTP、UDP和FTP三种不同的网络协议，学习使用Python的socket模块进行网络编程，掌握创建、绑定、监听、接收、发送和关闭socket的基本操作，以及处理网络延迟、丢包和错误信息的方法。  第一个子实验是编写一个简单的Web服务器，它可以接收客户端（浏览器）的HTTP请求，并根据请求返回相应的文件或错误信息。HTTP是一种应用层协议，它的请求和响应都是由起始行、头部和主体三部分组成的文本消息。HTTP使用TCP作为传输层协议，因此需要使用socket编程来建立和管理TCP连接。本实验中，我们使用了socket模块中的socket、bind、listen、accept、recv和send等函数，实现了一个基于TCP套接字的Web服务器。  第二个子实验是编写一个UDP Pinger程序，它可以测试网络延迟，即客户端发送一个简单的ping消息给服务器，服务器回复一个相应的pong消息给客户端，并计算客户端发送ping消息和接收pong消息之间的时间差，即往返时延（RTT）。UDP是一种无连接、不可靠、面向报文的传输层协议，它只负责将数据报从一台主机发送到另一台主机，不保证数据是否能到达、不考虑顺序和重复性的问题。同时，它还要处理丢包和超时发送的问题并返回错误信息。本实验中，我们使用了socket模块中的socket、settimeout、sendto和recvfrom等函数，实现了一个基于UDP套接字的Pinger程序。此外，由于网络真实丢包率比较低，实验中还使用randint生成随机数来模拟丢包机制  第三个子实验是编写一个FTP协议程序，它可以实现文件传输功能。FTP协议包括两个组成部分，其一为FTP服务器，其二为FTP客户端。其中FTP服务器用来存储文件，用户可以使用FTP客户端通过FTP协议访问位于FTP服务器上的资源。本实验中，我们使用了socket模块中的socket、connect、recv和send等函数，实现了一个基于TCP套接字的FTP协议程序。  通过本实验，我们对HTTP、UDP和FTP三种网络协议有了更深入的理解和应用，也掌握了Python语言进行网络编程的基本技能。本实验还有一些可以改进和扩展的地方，例如增加Web服务器支持多线程处理多个客户端请求的功能；增加UDP Pinger程序支持多个客户端同时发送ping消息给服务器并接收pong消息的功能；增加FTP协议程序支持上传文件到服务器或删除服务器上文件的功能等。这些都可以作为我们今后学习网络编程的方向和目标。 |