FILE TRANSFER FROM PC TO MOBILE USING SOCKET PROGRAMMING

Bhupathi Shwejan Raj[1], Bolisetty Sujith[2], Janagama Vamshi Krishna[3]

[1,2,3] Department of Computer Science Engineering
Amrita Vishwa Vidyapeetham
amenu4aie20017@am.students.amrita.edu; amenu4aie20018@am.students.amrita.edu
;amenu4aie20036@am.students.amrita.edu

Abstract

This paper demonstrates a very useful daily life application of socket programming i.e., file/data transfer from computer to mobile which are on the same network. This is done by using socket programming. By default sockets establish TCP/IP connection and ipv4 but can be changed to UDP or IPV6 on will. The files transferred from the PC can be received in the mobile app, the path of the file and the file itself is available in the app, it is possible to view the received files from the PC on the app.

Keywords: TCP/IP, python, flutter, file transfer.

Introduction

The Internet is all about connecting machines together.One of the most existing features of java is that it cooperates with any easy to use cross platform model for network communication that makes it possible to learn network programming without a year of study.A socket is the most popular operating system to give programs access to the network.That allows messages to be sent and received between applications on different networked machines. The TCP socket is part of socket programming. Since they provide a connection oriented service with both flow and congestion control.Socket programming is the APL for application to the read and write data from transmission control protocol, Internet protocol and the User datagram protocol, Internet protocol in socket programming. The socket programming file abstraction (open, read, write, close) then abstraction operating system resources.Socket allows the communication between two different processes on the same or different machine.Java socket model is developed from BSD Unix.A server is a process that performs some function on request from a client.It is a computer program that needs to connect to a local or wide area network such as the Internet;it uses a software component called a socket system.The socket opens the network connection for the program, allowing data to be read and written over the network.It is important to note that

sockets are software, but not hardware, like a wall socket.

Socket Programming

The socket programming only allows a single computer to serve many different clients at once as well as serving many different types of information. A socket provides the communication mechanism between two computers that use TCP. A client program creates a socket on its end of the communication. The attempt to connect that socket to a server. When the connection is made the server creates a socket object on its end of the communication. The client and server can now communicate by writing to and reading from the socket.

Client/Server Communication

The basic level of network-based systems consists of a server / client and a media for communication as shown in the client/server communication. A computer running a program that makes a request for services is called the client machine. The computer running a program that offers requested services from one or more clients is called a server machine. The media for communication between wired or wireless networks. There are generally programs running on client machines that are machines that make requests to a program (often called a server program) running on a server machine. That involves networking services provided by the transport layer that is part of the Internet software stack often called Transmission Control Protocol/Internet Protocol (TCP/IP).

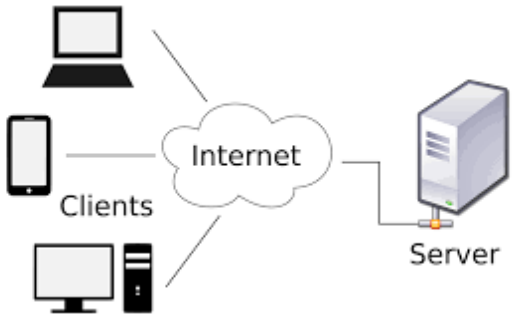


Fig 1: client server communication.

The transport layer comprises two types of protocols, TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). The most widely used programming interfaces for these protocols are sockets**.**

The TCP is a connection-oriented protocol that provides a reliable flow of data between two computers that is communication between client and server.

The UDP is a protocol that sends independent packets of data, called datagram's, from one computer to another with no guarantees about arrival and sequencing. Examples of applications that use such services include Clock server and Ping.

Literature review

In the paper named Socket Programming by Pooja Parashar and sarvesh sign was published on 15 july 2017 discusses the Network programming in socket programming describe in detail about concepts used in network programming in socket. It also describes socket programming network Programming and its socket types. The network programming is Client server programming, so it describes different client server models. It also explains about Socket programming, different types of sockets used like TCP or UDP. There is a comparison of Network programming using Java and Network programming using C, provided the TCP and UDP of both.and they have concluded that Network programming using Java have lots of advantages than that of C Programming.

In paper named Design and Implementation of Client-Server Based Application using Socket Programming by Ronald Cordova and Balaji Sudramurthy was published on 10 December 2017 discusses about the the overview and the detail implementation of the OBS(OpTel Billing Service) application using socket programming method. In the development of this study, They have chosen Java NetBeans programming language as it covers a wide range of functions and classes.In this, researchers found out that programming style,concepts,functions, datagram and socket programming are easily done in Java programming language.The designed application is used to simulate scenarios and is able to illustrate the use of socket programming and how it works in the real-world environment.Thus, the researchers would like to recommend that the application can be designed in an object oriented approach.

In the paper named Simple File Transfer System Using GUI and Socket Programming discusses the file transferring through the WAN without using a server whereby two or multiple users connect and transfer files to each other using the system designed. Here the system also saves logs so that it will be easy to read the history of the file transfer any time the user logs into the computer. They have used Graphic User Interface (GUI) programming in java and socket programming 4. Java language is used so that the design codes can run on any system. They have shown the results for each file transfer from computer with IP address 192.168.0.2 to 192.168.0.3.They have also discussed the difficulties they faced during this process like always checking the port numbers and turning off window firewalls.

Requirements

This system is a file-transfer system that can help users to share data among the same network.The system could help

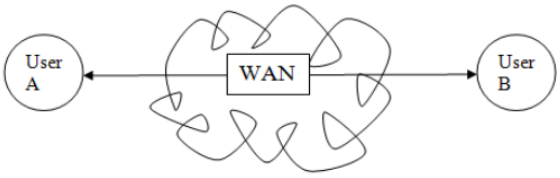
users find anyone who uses the same system in the WAN, meanwhile it can transfer files, The file-transfer module allows the user to transfer files to the other. This module could provide the user with the log of the file-transfer including time, the name of the file, as well as the directory where the file is saved if you are the receiver.

Since file transferring includes either server-client model or client client model, this system is a little different from the above mentioned software-designs, this time when implementing it, the user could not use any server to transfer the file, in other words the user transfer file directly without any transfer-station.

Through the use of these sharing systems things become much easier especially at an economic level in such a way that people started to share information and data such as files containing electronic books, mp3 etc., this is simply by one or few people buying the product and sharing it with the rest hence economically better. Examples of existing systems include Bittorrent, Utorrent etc. These services also allow users to download files other than music, such as movies and games.

Architecture of the design

As two machines in the WAN using remote IPs and this composed of participants that make a portion of their files directly available to other network participants, without the need for central coordination instances. Both users are suppliers and consumers of files, in contrast to the traditional client-server model where only servers supply, and clients consume.



Methodology

Sockets were first invented in 1971 and later became an API in the Berkeley Software Distribution(BSD) operating system released in 1983 called Berkeley sockets.

When the Internet took off in the 1990s with the World Wide Web, so did network programming. Web servers and browsers weren't the only applications taking advantage of newly connected networks and using sockets. Client-server applications of all types and sizes came into widespread use.

Today, although the underlying protocols used by the socket API have evolved over the years, and new ones have developed, the low-level API has remained the same.

The primary socket API functions and methods in this module are:

socket( )

The first step is to call the socket function, specifying the type of communication protocol (TCP based on IPv4, TCP based on IPv6, UDP).

bind( )

The bind() assigns a local protocol address to a socket. With the Internet protocols, the address is the combination of an IPv4 or IPv6 address (32-bit or 128-bit) address along with a 16 bit TCP port number.

listen( )

The listen() function converts an unconnected socket into a passive socket, indicating that the kernel should accept incoming connection requests directed to this socket

accept( )

The accept() is used to retrieve a connection request and convert that into a request.

connect( )

The connect() used to create a connection to the specified destination

send( )

The send() function shall initiate transmission of a message from the specified socket to its peer.

recv( )

The recv() used to read incoming data on connection-oriented sockets, or connectionless sockets.

close( )

The normal close() function is used to close a socket and terminate a TCP socket.

TCP Sockets

You're going to create a socket object using socket.socket(), specifying the socket type as socket.SOCK_STREAM. When you do that, the default protocol that's used is the Transmission Control Protocol (TCP). This is a good default and probably what you want.

Why should you use TCP? The Transmission Control Protocol (TCP):
Is reliable: Packets dropped in the network are detected and retransmitted by the sender.
Has in-order data delivery: Data is read by your application in the order it was written by the sender.

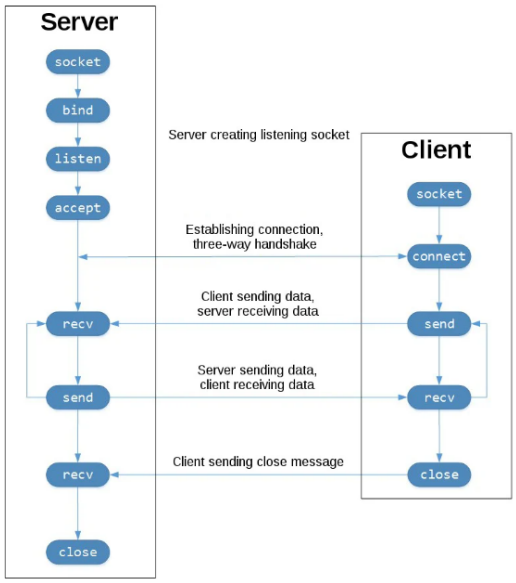


Fig 2: Interaction between client and server.

Echo Client and Server

SOcket.socket() creates a socket object that supports the context manager type, so you can use it in a with statement. There's no need to call s.close():

The arguments passed to socket() are constants used to specify the address family and socket type. AF_INET is the Internet address family for IPv4. SOCK_STREAM is the socket type for TCP, the protocol that will be used to transport messages in the network.

The .bind() method is used to associate the socket with a specific network interface and port number:

The values passed to .bind() depend on the address family of the socket. In this example, you're using socket.AF_INET (IPv4). So it expects a two-tuple: (host, port).

host can be a hostname, IP address, or empty string. If an IP address is used, the host should be an IPv4-formatted address string. The IP address 127.0.0.1 is the standard IPv4 address for the loopback interface, so only processes on the host will be able to connect to the server. If you pass an empty string, the server will accept connections on all available IPv4 interfaces.

port represents the TCP port number to accept connections from clients. It should be an integer from 1 to 65535, as 0 is reserved. Some systems may require superuser privileges if the port number is less than 1024.

The .listen() method has a backlog parameter. It specifies the number of unaccepted connections that the system will allow before refusing new connections. Starting in Python 3.5, it's optional. If not specified, a default backlog value is chosen.

The .accept() method blocks execution and waits for an incoming connection. When a client connects, it returns a new socket object representing the connection and a tuple holding the address of the client. The tuple will contain (host, port) for IPv4 connections or (host, port, flowinfo, scopeid) for IPv6. See Socket

Address Families in the reference section for details on the tuple values.

One thing that's imperative to understand is that you now have a new socket object from .accept(). This is important because it's the socket that you'll use to communicate with the client. It's distinct from the listening socket that the server is using to accept new connections:

Results

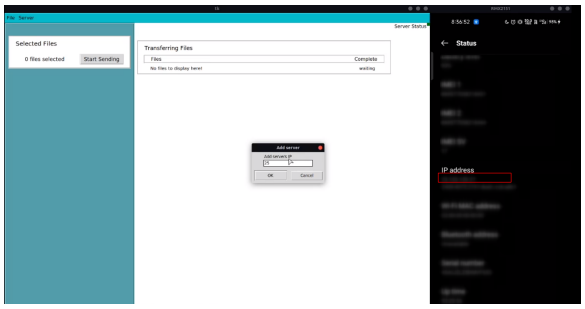


Fig 3: Adding client to server.

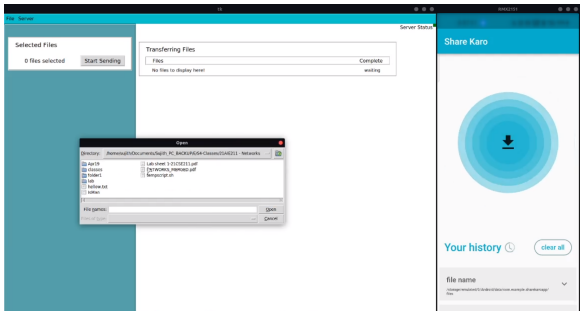


Fig 4: Adding files on client side which will be transfered to server.

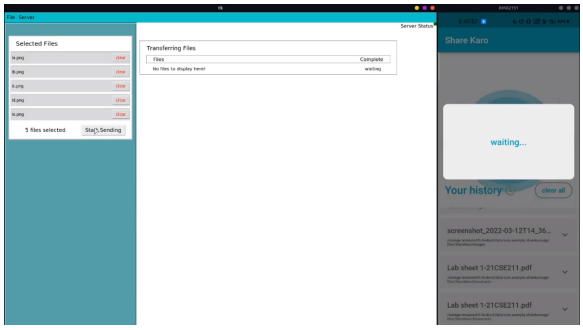


Fig 5: Transferring files to server.

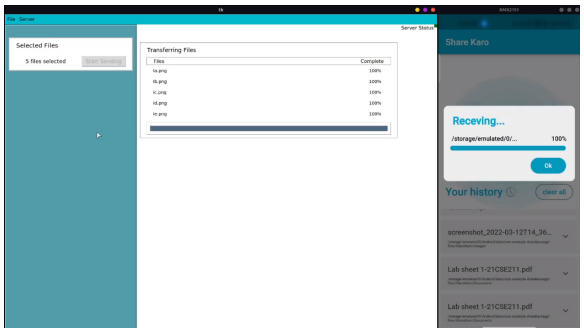


Fig 6: Files has been transfered to sever.

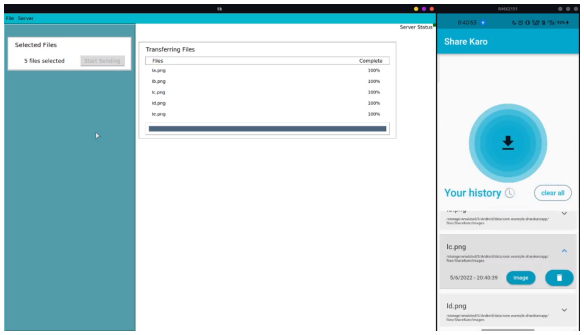


Fig 7: Received files on sever side can be viewed from app.

Conclusion

In this paper we have presented File Transfer from Pc to Mobile using Socket Programming.Here after entering the server IP address on the python interface a connection will be established between client and server we can then add multiple files on the client side and hit start sending. We can then see the client sending files to the mobile app, we can also see the files have been successfully received by the mobile app. We can also open the transfer from the app, even the path of the files are available in the app.

Future scope

Right now the program is capable of transferring file/data from one computer to one mobile. In the bigger picture we want to transfer data from one computer to multiple mobiles. We can even extend this for sharing data from computer to computer or even mobile to mobile. The suggested model in this paper is unidirectional which means that we can only send data from computer to mobile, we can't send data from mobile to computer, we have a strong hope of implementing the bidirectional functionality in the future.

References

1. https://realpython.com/python-sockets/
2. https://aws.amazon.com/what-is/api/#:~:text=API%20stands%20for%20Application%20Programming,other%20using%20requests%20and%20responses.
3. https://realpython.com/python-sockets/
4. https://www.geeksforgeeks.org/socket-programming-cc/#:~:text=Socket%20programming%20is%20a%20way,reaches%20out%20to%20the%20server.
5. https://www.techtarget.com/searchnetworking/definition/TCP-IP
6. https://www.techtarget.com/searchnetworking/definition/UDP-User-Datagram-Protocol#:~:text=User%20Datagram%20Protocol%20(UDP)%20is,provided%20by%20the%20receiving%20party.
7. https://intellipaat.com/blog/what-is-client-server-architecture/#:~:text=Client%20server%20architecture%20is%20a,are%20delivered%20over%20a%20network.
8. https://www.investopedia.com/terms/i/ip-address.asp#:~:text=IP%20address%20stands%20for%20internet,to%20send%20and%20receive%20information.
9. https://aws.amazon.com/what-is/api/#:~:text=API%20stands%20for%20Application%20Programming,other%20using%20requests%20and%20responses.