Types of socket programming and socket designs: A Review

Ameya Gokhale
Bal Shikshan Mandir
English Medium School
Pune, Maharashtra,
India

ABSTRACT
Sockets are used to connect multiple devices on a network. This concept is used widely in networking and network based applications. Complex networks can be developed using APIs (Application Programming Interface) in the applications using network. Socket Programming is supported in many languages, namely C, C++, C#, Python, Java and others. Sockets are created using the host IPA (Internet Protocol Address) and the TCP (Transmission Control Protocol) port number. Socket Programming can be done in numerous ways, as there are different designs of socket programming. Also, there are many different types of sockets. In this paper, Python and Java are used instead of other powerful programming languages, due to multiple reasons, including the fact that these two languages are extensively used in networking. This paper also discusses varied reasons for choosing Python and Java over other programming languages. A code presenting a simple client-server connection using socket programming in both, Python and Java has also been included in this paper. This research paper focuses on Applications of socket, their designs, types of socket and a simple socket server using Python and Java.

Keywords
Sockets, network, Application Programming Interface, host IPA, Internet Protocol Address, Transmission Control Protocol port number, Python and Java, client-server, socket programming, Applications of Socket.

1. INTRODUCTION
Client-Server program has become a very common concept in computer networks and in distributed computing. A client is a computer terminal, permitting the user to access the interface(s) created by the server. A client initiates communication with the server through a connection. The server, who is waiting for clients, further responds to the client request [10]. A socket is an endpoint of a two-way communication link between two programs running on a network, which allows sending and receiving of data. In socket programming, socket APIs are used to establish communication between remote and local processes. Socket is used for connecting the clients and the server together, either on one or multiple devices. It is used for both, connection-oriented and connection-less communication between the applications that are running on different compilers. A socket is connected to a specific TCP port, making the TCP layer identify the application to which the data is destined to be sent to. The server IPA is also needed for establishing the connection [4]. A socket program can send or receive data; irrespective of the programming language used for it. That is, a socket which is programmed using Java, can communicate and establish a connection with another program, which is programmed using Python, or any other programming language [10]. This paper presents socket programming using Python and Java.

2. BENEFITS OF USING PYTHON AND JAVA FOR SOCKET PROGRAMMING
Both, Python and Java are powerful and widely used programming languages. Python and Java, both support OOP (Object Oriented Programming) and are used in various fields like web development (User and Server side programming in Web apps, websites etc.), android and other software development and also in networking. There are 6 key benefits of using Python and Java for socket programming:

Sending and receiving of any object can be done through socket programming in Java, just that the object class needs to implement Serializable interface [1].

Basic data type conversion into integer and float can be done using built-in methods like writeInt() and readInt(), reducing the overhead of converting from byte stream to the required data type [1].

Java is already being used widely for networking in Android. Hence, the code for socket connection compatible with PC can be used for socket programming in android applications with minimum changes [1].

Standardized exception handling makes debugging easier in Java [1].

Python packages are already being considered competent tools for network programming and is used in network management systems. Learning basic socket programming in Python may help in development of higher applications later, using complex networking packages and libraries in python, like Twisted [2].

Even though Python programs tend to run slower than Java programs, the time needed to develop them is less [1].

3. SET-UP FOR SOCKET PROGRAMMING
3.1 Set-up for socket programming in Python:
In python, socket programming is done by importing a module, 'socket'. All the inbuilt functions needed for a simple socket connection are present in this module. The server is bound to a specific port number, using the bind() method, which takes 2 parameters, the IP address of the server and the port number. The server has 2 more methods, accept() and

close() which initiate and end the connection respectively [5]. The module needed for python is imported as follows:

```
import socket
```

Fig 1: Importing Modules for Python

3.2 Set-up for socket programming in Java:

In Java, the modules needed for socket programming can be imported from the Java API networking package (java.net). Also, for performing operations related to input and output, input-output module also has to be imported. All the connections and input-output processes need to be ended after the completion of their task [3]. The modules needed for java are imported as follows:

```
import java.io.*;
import java.net.*;
```

Fig 2: Importing Modules for Java

4. SOCKET PROGRAMMING

4.1 Server Side Programming

The socket connection is created by entering the host (IP Address of the server) and the TCP port number. The server initiates the connection and waits for the clients to join. On receiving the client request, the server communicates back through the connection to the client. For achieving this, 2 sockets are needed: A ServerSocket and a main socket. The ServerSocket waits for client connections, and the main socket is used for communication between the server and the client. In Java, getOutputStream() method is used to send a message through the socket connection, while in Python, send() is used for sending the message [3] [5].

4.2 Client Side Programming

The client is connected to the server connection using the functions, 'connect()' and 'new Socket()' in Python and Java respectively. Here, the client waits till the server starts the connection. When the socket connection is made, it sends a request to the server, and then joins the connection when the server grants access [3] [5].

This is how a basic socket connection is initiated in Python and Java.

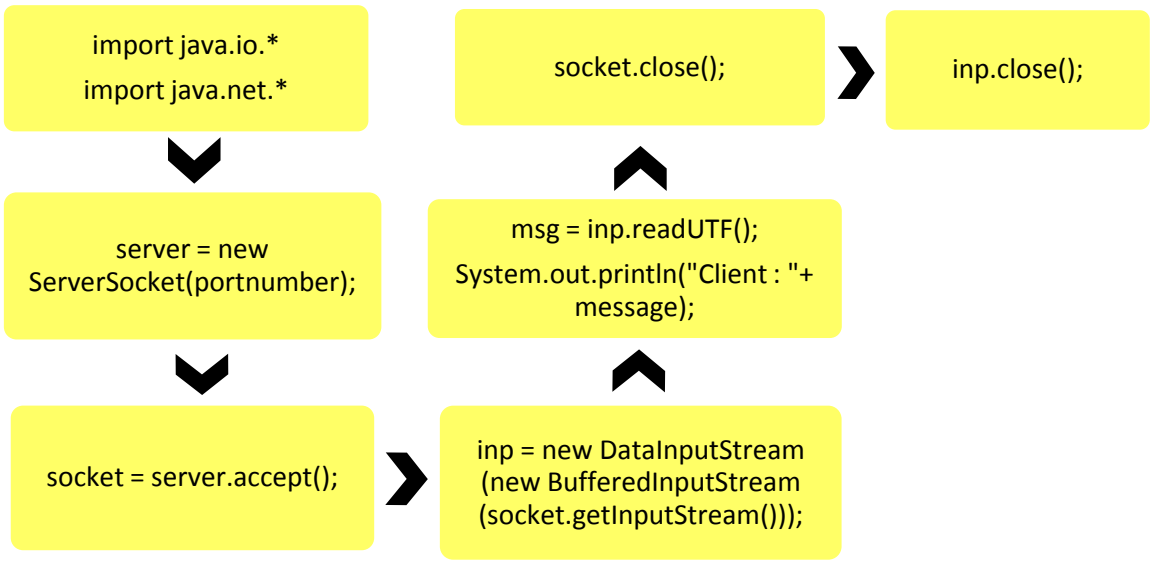

Fig 3: Important commands for server socket in Java

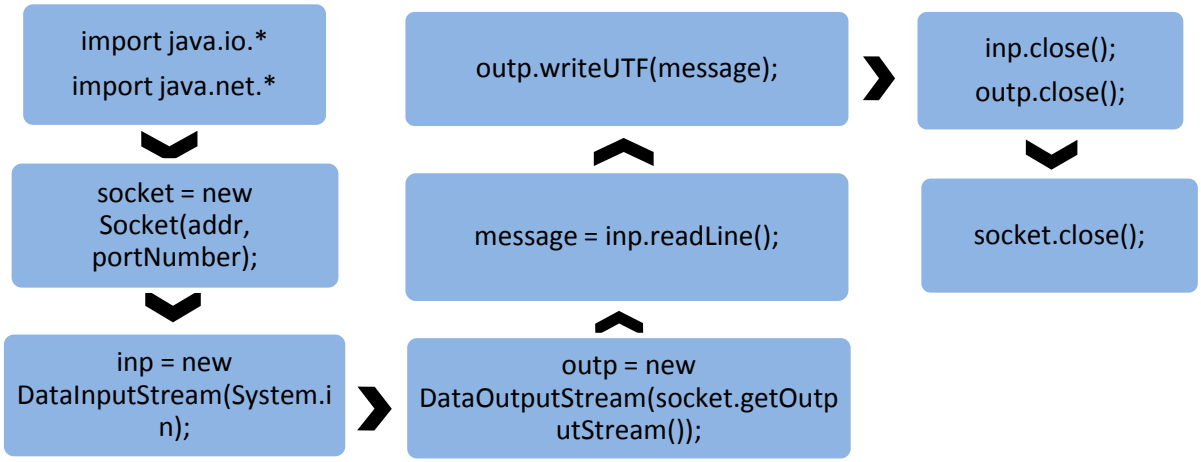

Fig 4: Important commands for Client socket in Java

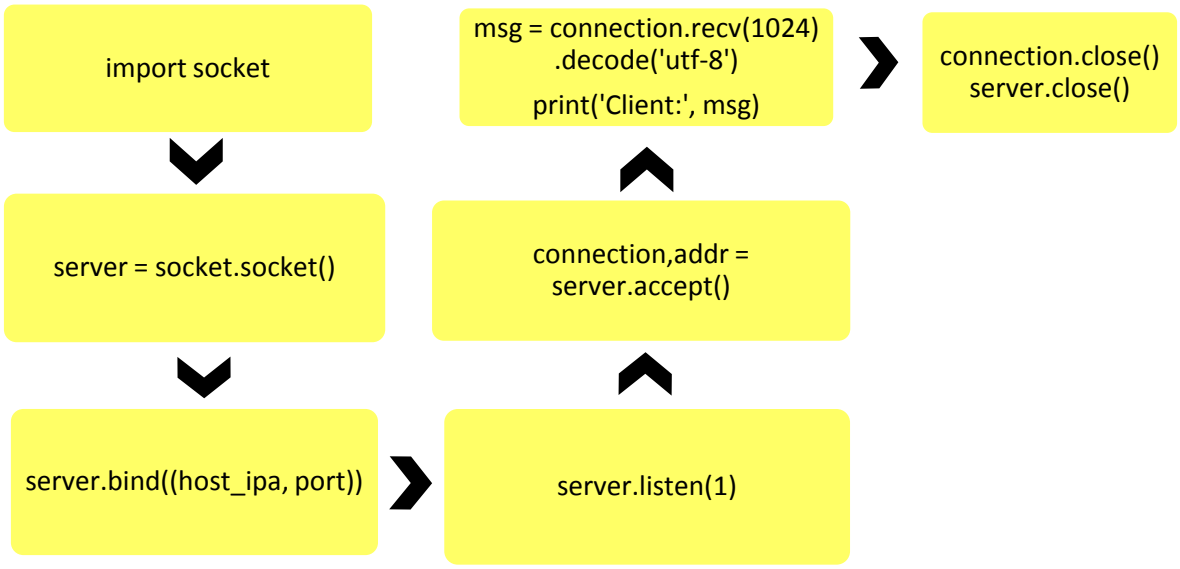

Fig 5: Important commands for Server socket in Python

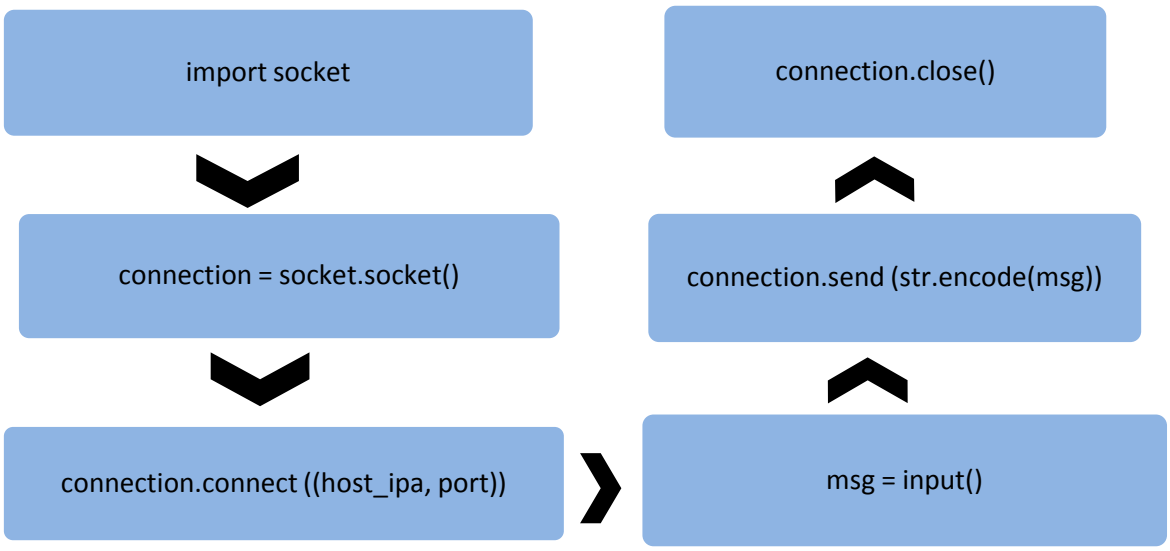

Fig 6: Important commands for Client socket in Python

In the above diagrams, the key commands needed for a simple client-server socket connection are given for both, Python and Java. In Java, the socket for server is created using function ServerSocket(), which takes one parameter, the TCP port number, while in Python, the function, bind() takes 2 parameters, the host IPA and the TCP port number. It binds the server to the specified port number, and the listen() function waits for and queues the client connection request(s). At the end of the program, the connection is closed using close() function. The commands in the chart allow an input from the client side, which is then sent through the connection to the server and then printed out on the server side. The message is encoded before sending it through the connection, and it is decoded when it is received by the server. This is done in utf-8 format. The encoding and decoding is done using the functions, str.encode() and decode('utf-8') respectively in Python. However, the task of encoding and decoding is done using the functions writeUTF() and readUTF() respectively in Java. In this example, only single way messaging (client to server) can be seen. However, a connection, where input and output is allowed on both, the server and client side, can be programmed using multithreading, which is supported by both the languages, Python and Java. That is, there will be two-way flow of messages, server to client and client to server. When these programs are run on a console, they show the following output:

```
C:\Users\hp\Desktop\Ameya\Java>java Server
Waiting for clients...
Client : Hello, server
```

Fig 7: Output seen on server side

```
C:\Users\hp\Desktop\Ameya\Java>java Client
Connected with server at port 9999
Hello, server
Thank you for connecting to the server
```

Fig 8: Output seen on client side

In the above outputs, the server prints “waiting for clients” till it gets client requests. Later, when the client is connected, it prints the message, which is sent by the client to the server. On the client side, the port number used for the connection is printed as soon as the client connects to the server. At the end, when the connection is closed, an acknowledgement message is printed on the client side. The above programs allow sharing of a single message, but the chat process can be kept repetitive using loops.

5. TYPES OF SOCKETS

Sockets can be classified according to the properties of the communication which are visible to the user. The Internet family for the sockets, connecting over both IPv6 and IPv4, is recognized by the value AF_INET6. Also, the Internet Family Sockets allow accessibility to transport protocols, TCP/IP. AF_INET permits source compatibility with the older applications and raw access to IPv4 [12].

5.1 Stream Sockets

These sockets permit the connection of processes over TCP. A stream socket allows bidirectional, reliable, sequenced, and unduplicated transfer of data without any record boundaries. The data from the socket can be read and written (edited) in the form of a byte stream after the establishment of the connection. The type of socket is SOCK_STREAM [12].

5.2 Datagram Sockets

These sockets permit the connection of processes over UDP (User Datagram Protocol). Similar to stream socket, these sockets also allow bidirectional flow of messages. However, the sequence of receiving of the messages on the socket may be different from the sequence of sending the messages from the other end. There may be duplication of messages in this type of socket connection. Record boundaries in the data are stored. The type of the socket is SOCK_DGRAM [12].

5.3 Raw Sockets

These sockets permit the connection of processes over ICMP (Internet Control Message Protocol) for communication. Usually, these sockets are based on datagram. However, their specific characteristics depend on the interface by the

protocol. These sockets are used for supporting development of new communication protocols. Also, these are used for accessing cryptic facilities of other protocol(s). Raw Sockets are used very rarely, as only super user processes can use them. The type of the socket is SOCK_RAW [12].

6. SOCKET DESIGNS

Sockets can be programmed in a more advanced way, in order to establish secure and stronger connections. Graphics and listing of calls, illustrating the flow of events in the applications can also be observed in socket programming. Xsockets tool can be used interactively, using some of the following APIs in the programs, or specific changes can be made for the environment [6]. Following are a few examples of socket application designs:

6.1 Connection oriented designs

A connection oriented server can be created using an Iterative Server and/or a concurrent server [7].

6.1.1 Iterative Server

A server job handles all the incoming client-requests and connections indigenously. All the data in this connection flows in client jobs. This server is comparatively easier and faster to develop, but has some disadvantages. When the server is handling request from a client, some other client may try to connect to the server, making the requests fill the listen() backlog, making the server eventually reject some clients [7].

6.1.2 Concurrent Server

Many threads are used for handling the connection requests from the clients. Mostly, multiple clients connect to the server simultaneously in this connection. For multiple concurrent clients in a network, asynchronous I/O (Input Output) socket APIs can be used. These APIs provide the best network performance for multiple concurrent clients [7].

6.2 Asynchronous Input-Output

The application using asynchronous I/O (input-output) starts an I/O function, specifying a port handle. On its completion, the I/O completion port is posted with the information of status and an application-defined handle, activating one of the waiting threads. Then, a buffer is supplied on the original request. The buffer, length of data processed to/from the buffer, type of completed I/O operation and application-defined handle is received by the application. This application handle can be used to identify the client connection and also to store the information of the state of the connection. This passed handle makes the worker thread determine the next step to establish the client connection. Worker threads processing the completed asynchronous operations can handle multiple client requests. Copying to and from user buffers occurs asynchronously to the server processes, diminishing the waiting time of client. This is useful in multi-processor systems [8].

6.3 Using signals with blocking APIs

When a process or an application is blocked, the signals report the user and provide a time limit for blocking the processes. A signal is created every five seconds on the accept() call. As there is a specific time limit, the call blocks only for five seconds at a time. Signals can be used to reduce the impact, as blocked programs can obstruct the performance of an application of a server [9].

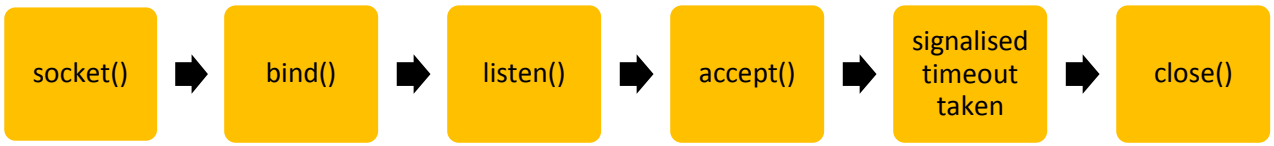

Fig 9: Processes in socket connection while using signals with blocking socket APIs

7. APPLICATIONS OF SOCKET PROGRAMMING

Socket Programming has multiple applications in networking, wherever computer networks and/or distributed computing comes into picture. However, currently many layers are applied on sockets, like HTTP (Hypertext Transfer Protocol) in network-based applications. There are many other ways/designs of sockets than mentioned in this paper. These include using poll() instead of select() API, using multicasting with AF_INET and some others. The poll() API is included in the Single Unix Specification and the UNIX 95/98 standard. The poll() API has the same performance as the currently existing select() API. In an application, an IP (Internet Protocol) datagram can be sent, which is received by a group of hosts by using IP multicasting [6].

8. CONCLUSION

Socket Programming has multiple applications in computer technology, wherever computer networks and/or distributed computing is needed. Many programming languages support Socket Programming, including Python, Java, C and C++. Sockets can be created using multiple languages, however the connection within sockets does not depend upon the programming language used. That is, a server programmed using C programming language can communicate and establish connection with clients programmed in other languages like Python, Java, C++ and others. There are multiple designs and ways of creating socket connections today. Currently plain socket connections are not made for communications in a network. Many layers are applied on sockets like HTTP. This is done for increasing the complexity of connections, thereby increasing the capacity of data exchanged, making the connection more powerful and sometimes, enhancing security. There are many types of sockets, too which are used according to their intended function.

9. REFERENCES

[1] DebjyotiBhattacharjee, "Which language should I choose for Socket programming, Python or Java?" May 29, 2015 [online] available: https://www.quora.com/Which-language-should-I-choose-for-Socket-programming-Python-or-Java

[2] SatishAnnigeri, "Is python a good choice for socket programming?" February 15, 2019 [online] available: https://www.quora.com/Is-python-a-good-choice-for-socket-programming

[3] SouradeepBarua "Socket Programming in Java" February 26, 2018, [online] available: https://www.geeksforgeeks.org/socket-programming-in-java

[4] Neha Vaidya "Know all about Socket Programming in Java" June 17, 2021 [online] available: https://www.edureka.co/blog/socket-programming-in-java/

[5] KishlayVerma "Socket Programming in Python" August 31, 2021 [online] available: https://www.geeksforgeeks.org/socket-programming-python/

[6] https://www.ibm.com/docs/en/i/7.1?topic=programming-examples-socket-application-designs

[7] https://www.ibm.com/docs/en/i/7.1?topic=designs-examples-connection-oriented

[8] https://www.ibm.com/docs/en/i/7.1?topic=designs-example-using-asynchronous-io

[9] https://www.ibm.com/docs/en/i/7.1?topic=designs-example-using-signals-blocking-socket-apis

[10] Rolou Lyn R. Maata, Ronald Cordova, BalajiSudramurthy, AlrenceHalibas, "Design and Implementation of Client-Server Based Application using Socket Programming in a Distributed Computing Environment", IEEE International Conference on Computational Intelligence and Computing Research, 2017

[11] https://softwareengineering.stackexchange.com/questions/109442/difference-between-networking-programming-and-socket-programming

[12] https://docs.oracle.com/cd/E19455-01/806-1017/sockets-4/index.html