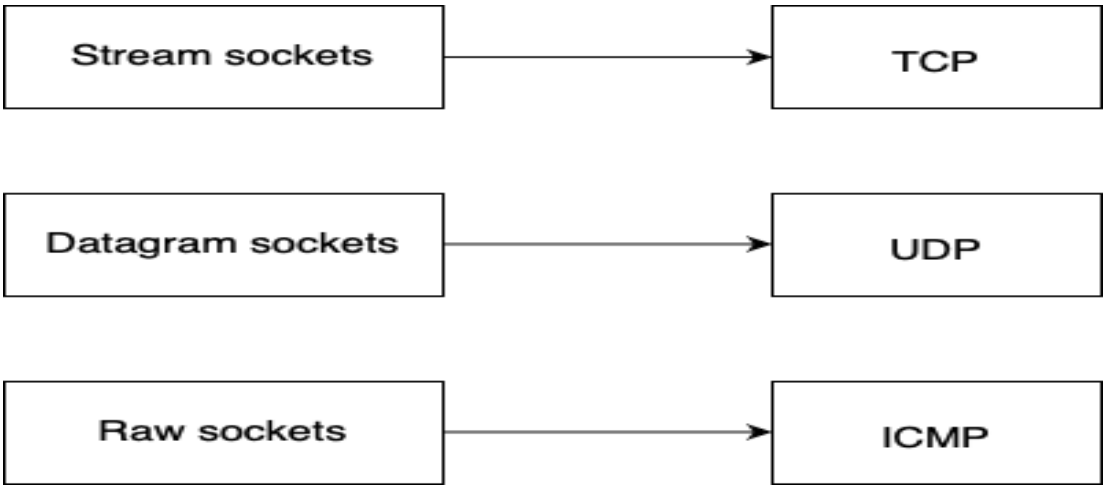
Sockets Report

What are the three types of network socket and what are their differences?

There are three types of network sockets with each individual network socket fulfilling a different purpose. There contains a vast variety of differences in each socket. The three types of sockets are stream sockets, datagram sockets and raw sockets. Each socket communicates with different protocols highlighting the differences between them. Stream sockets communicate with the TCP (transmission control protocol) protocol by allowing communication between the processes and the protocol. Datagram sockets on the other hand communicate with the UDP (User Datagram Protocol) protocol by allowing communication between the process and the protocol. Raw sockets differ by communicating with ICMP (Internet Control Message Protocol) protocols, they are distinctly different from the first two sockets as they provide access to the ICMP protocol and can be used if you want to make your own transport layer protocol. To use the various types of network sockets in python, each socket must be called by a different name in python. To use stream sockets in python you must use SOCK_STREAM', if however, you want to use datagram sockets you must use 'SOCK_DGRAM' and 'SOCK_RAW' is used for raw sockets[*IBM 2020*]. Stream sockets deliver its deliverables in the order that it was sent, so if you send deliverables in a certain order, they will arrive in the order that was originally defined. In contrast, this is not guaranteed to occur in Datagram sockets. This socket delivers its deliverables and the information is sent out. The process receiving the messages may be met with messages that are not in the order in which they were sent from. The reason why they don't arrive in the order in which they are sent is because messages sent in this socket can be sent without establishing a connection between the socket and the process it is sending it to. This process is not used at all in raw sockets as this socket does not send out any information, instead it is used to build new protocols. Raw sockets allow access to the underlying transport provider, which in turn can be used to manipulate the underlying transport provider, you are basically responsible for creating and parsing the underlying transport headers and logic, thus allowing you to create new protocols [Jens Heuschkel].



A stream socket provides for the bidirectional, reliable, sequenced, and unduplicated flow of data without record boundaries, this type of socket is very reliable for transmitting data through sockets. The variance in a datagram socket is that it is not a reliable way of transmitting data, this is as a consequence of UDP being a much simpler protocol whereas TCP allows for an ordered and reliable connection and is also not sequenced and not unduplicated, all of these features vary from a stream socket. This form of thinking can be applied to real world problems to highlight the variance between both sockets, for example say there is some extremely important information that you want to ensure gets across to the destination it is trying to send to, you would be much more inclined to use TCP as it is reliable in comparison to UDP which is not and can lose information in the process of sending the information. In contrast if you want to send information to thousands of processes every x number of seconds, where some missing information does not matter UDP is a preferable option as it is a much simpler protocol and does not have to handle error checking like TCP. A raw socket varies each time it is used as the features are heavily dependent on the protocol it implements, so as a result this socket can contrast both or one or none of these socket features just defined dependent on the features implemented in the raw socket protocol. A datagram socket is generally used for short messages as messages are connectionless and the messages sent are of a fixed length as well as the order and reliability of a message not being guaranteed. A stream socket varies from this logic, the order and reliability of a message is guaranteed meaning that messages sent in this socket can be of substantially larger size [*Panagiota Fatourou*]. Once again, the generality of the length of a message in the raw socket is dependent on the form of protocol this socket implements.

A stream socket has an open connection, meaning that when the socket builds its packet and sends the information it has the destination information contained inside it. This varies from a datagram socket, as they don't have an open connection thus making them connectionless. This contributes and negates from the reliability of both sockets, a stream socket being more

reliable, with a datagram socket having a more unreliable nature. A raw socket is dependent on the protocol it elects to build from.

2. What is the process of creating a network socket

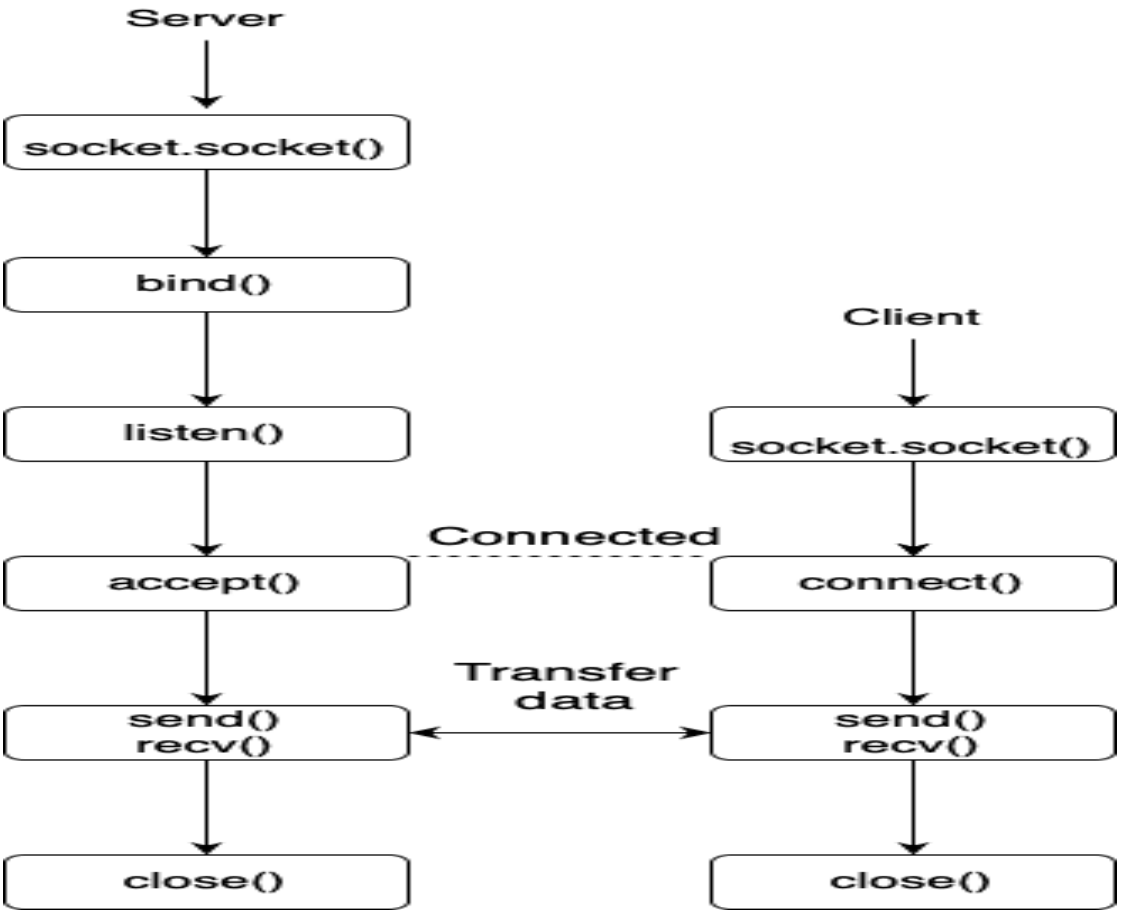
The first step in order to create a network socket is to import the socket library, and from here you can create sockets from the library. Once you have imported the library, we need to call the socket constructor, in order for the constructor to function properly a family parameter and a type parameter need to be passed into it[*Microsoft Docs*]. Examples of family address values are 'AF_INET' and also 'AF_INET6', there exists many more examples of family addresses, allowing for variety in socket programming. The type parameter values are the types of network sockets that exist, so the first question and answer should give a good synopsis of the different types of network sockets and the varying features of these sockets, the main three types of network sockets are, Stream sockets, datagram sockets and finally raw sockets. Now that we have the basic fundamentals of creating a socket, we can now apply this process in order to produce a socket. First, we assign the socket we are using to a variable, then use the socket library to create a socket. The socket needs to have two parameters (the family of the socket and the type of the socket) passed to it. For example

Import socket

Socket_var = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

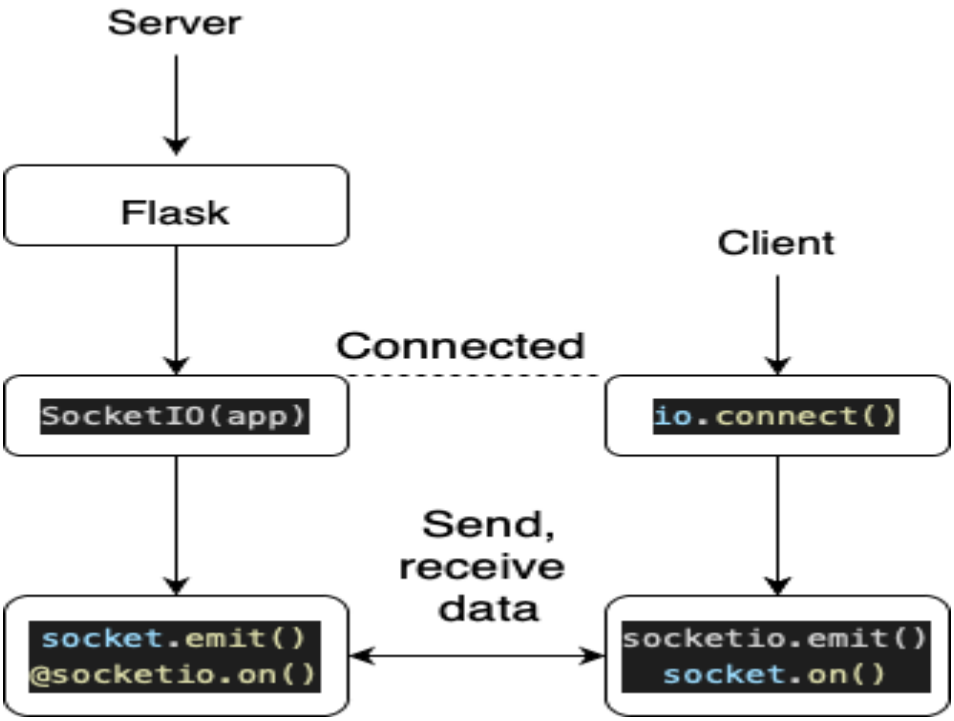
This socket opens up a UDP socket (under the SOCK_DGRAM parameter variable) while using the IPV4 addressing protocol. This is the process when creating an open socket which will be the same for both the client and also the server. The client side is now finished, however there are still tasks to be completed in the server side to ensure that it can properly process any incoming connections that comes its way. It must ensure that it binds to an address and that it can properly listen for any incoming connections that might come its way, while it is in the process of listening for connections it needs to make sure that it can accept the connection requests as well. We can bind the address using the bind() function. This function takes in two parameters an ip address and finally a port. The address is in the form of a string while the port is an int. To reduce errors, you should be more disposed to picking a port number that is an arbitrary large number, as any ports that are reserved reside in the lower numbers. For the address, the local host is a solid choice. If you use the localhost for your address then the connection is not open on your network (preventing any unwanted traffic). If it is the localhost, then only machines on your own local host can interact with you. The next step involves listening for any connections. To apply this process, one must use the listen() function, so it will be the variable of the socket calling the listen function[*Limi Kalita*]. Once this has been properly applied, we can use the accept() function to accept an incoming connection. To

properly accept a connection the right way you must assign two variables to the function call. A connection object and the address of the object making the connection are returned by the accept() function, which can duly be assigned to variables. Once the socket is created and is up and running, then the client just needs to call the function connect() with the appropriate ip address and port name as its parameters. From here the client and server can transmit data through its sockets, so for example they can use the send() function to send a message, and the recv() function to receive the message that was sent, which can then be stored inside a variable. To close the sockets and free up any ports one can use the close() function to do so.



If you want to create a socket in python another option resides in the flask library, and more specifically socket.io. To implement this, you would originally create the socket inside of the python code. In the python side, you can create the server which would first create an application using flask which will run the html pages. From here create a socket using socketio, and assign it to a variable. This socket will take the application defined earlier as its parameter. One would also have to implement a html script, which will connect to the socket you created using socket.io by using 'io.connect', connecting it to the appropriate ip address and port. from here flask can create local html pages and emit data to every client inside the server

through the emit() function. Emit will take a function as a parameter, which will be defined by the html code, it will also take the data to be sent as a parameter. The html code will handle the data and based on the function will decide what to do with it. The html code will handle the client side, and can call the server by using its socket variable it defined earlier alongside the appropriate emit function, with the appropriate parameters it wants to pass.

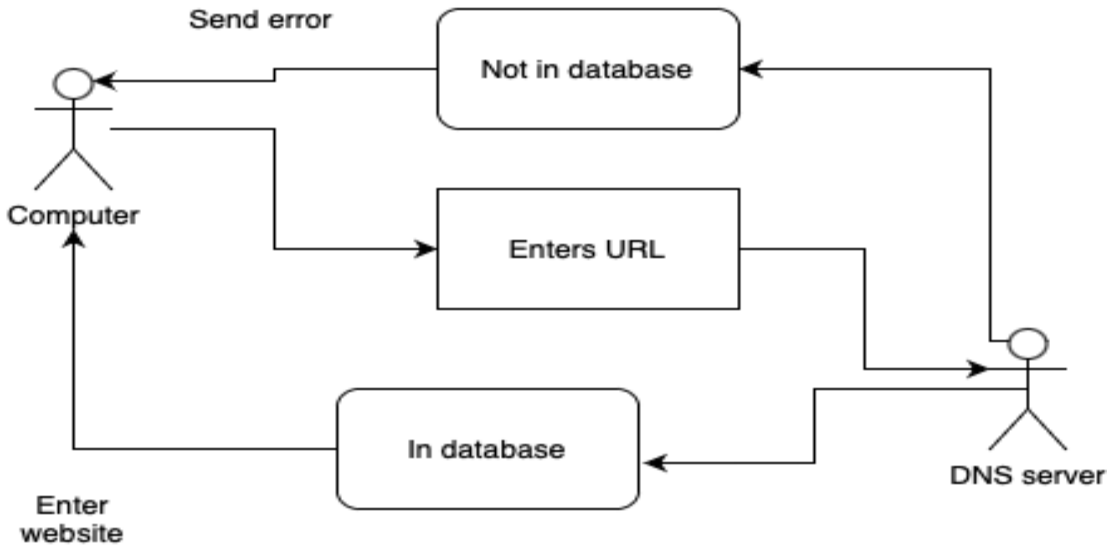


3. What are the steps involved when the user opens their browser and visits the website discord.com, make sure to take account of the protocols involved in each step.

The first step involved with accomplishing this process is to enter the website 'discord.com' into the address bar which is usually located in the top of the browser. This address is commonly known as a URL or a Uniform Resource Locator.[T. Berners-Lee] Every webpage in the internet can be accessed by a URL. A URL has many different components, with each component having a different functionality. The first section of a URL defines the protocol of the url, it is usually 'http://', the next section defines the third(www.),second(discord) and top(.com) level domain. The first section is either the hypertext transfer protocol(http) or else it is either the hypertext transfer protocol secure(https). Https is the most commonly used protocol, as it can encrypt http requests and responses so it is far less likely that any data will be stolen by the likes of hackers. This is because it uses TLS(SSL). It can also verify the identity of the clients using the

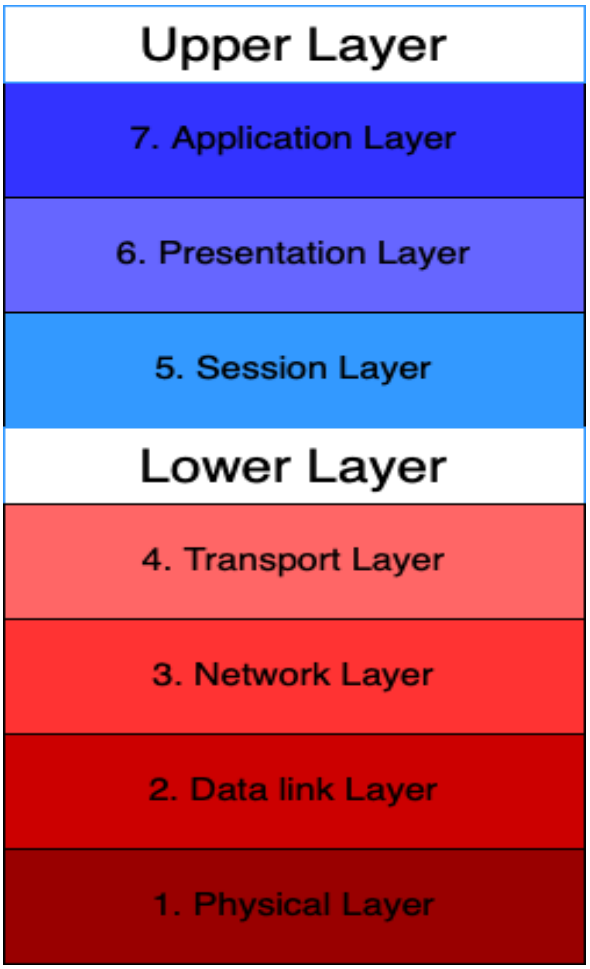
certificate it was issued upon entering the website. This protocol is used to get resources like html documents. It sends a request to the server through a proxy and the server returns its response. “www.” is usually the third domain however other subdomains can be used. The second domain and the third domain is the website name(‘discord’) and its associated domain in this case discord uses the most common domain (‘.com’) although this can vary from address to address.

Next the browser calls upon a software module called DNS (Domain Name System), which subsequently queries for the Ip address that corresponds to discords Ip address. DNS translates domain names into their ip addresses. The browser therefore when it has the corresponding IP address can initiate a connection to the IP using a port such as 80(default) or 443. To fully establish a connection a process called a three-way handshake is implemented. The browser starts by sending a SYN request[*Lawrence Williams*] (a synchronize request) to its specified port that it is trying to connect to. If the server is listening on that specific port and also accepting any incoming requests it will send a response back which is known as SYN-ACK(synchronize and acknowledge). Finally in order for the connection to be fully established the browser must then send an ACK(acknowledgment) and bind its port to its host. The connection is either fully established or an error code response is sent back. The server can then process the url and through the http or https protocol which sends data packets through the protocol. The http can use protocols such as TCP (transmission control protocol), which is a protocol which exchanges data between two computers in an error-free fashion. Other protocols include UDP, which do not maintain a reliable connection but does have its uses such as in streaming content. The webserver processes the information sent in the data packet and emits a http status code. If the status code returns successful then the protocol functions by passing the html document to the backend of the server, the server then can send the html page to the browser again using the same http or https protocol. If the request is unsuccessful, it return a 404-error code. The html is usually not a file but a resource that uses the backend of a server to send queries to a database which returns data and creates the html. So, html is generated very swiftly using a variety of differing techniques.



Once your browser receives the html page from discord.com, it renders it onto the screen, thus closing the connection between the browser and the webserver. HTML uses javascript, thus the javascript code is executed and the page is duly rendered. Webpages are usually coded in HTML, CSS and javascript. Each browser also has a cache, in which data is temporarily stored when the webpage is accessed. So therefore, when a webpage is re-visited, not all the data needs to be requested from the web server.

The OSI model which is a model of network communication and describes the functions of a network system. There are seven layers to the model. It provides a thorough insight into the functionality of opening a browser and visiting a website in terms of network functionality.



Each layer has a differing level of abstraction. Browsers exist in the application layer, as one progressively goes down the layers the data is duly converted into the form in which it will be transported. So, in this layer we can find the process of opening the browser and entering the website('discord.com'). The upper layers appropriately handle the process of http trying to fetch the html from the server and displaying it in the browser. The transport layer handles the TCP and the UDP protocols which performs its protocols duly and is used in the http protocol to exchange data between two computers.

References

1. Socket Types – IBM 2020

https://www.ibm.com/docs/en/aix/7.1?topic=protocols-socket-types

"SOCK_DGRAM" - "Provides datagrams, which are connectionless messages of a fixed maximum length"

"SOCK_STREAM" - "Provides sequenced, two-way byte streams with a transmission mechanism for stream data"

"SOCK_RAW" - "Provides access to internal network protocols and interfaces."

2. Introduction to raw sockets- Jens Heuschkel, Technische Universtat Darmstadt, May 17, 2017

https://tuprints.ulb.tu-darmstadt.de/6243/1/TR-18.pdf

"IP-headers of RAW sockets can be manually created by the programmer if the option IP_HDRINCL is enabled

[1]. This way, Raw sockets allow a programmer to implement new IP based protocols. If IP_HDRINCL is not

enabled the IP header will be generated automatically." - page 11.

3. Introduction to Sockets Programming in C using TCP/IP, Panagiota Fatourou, May 2012, CSD.

https://www.csd.uoc.gr/~hy556/material/tutorials/cs556-3rd-tutorial.pdf

"Stream sockets (e.g uses TCP)", provide reliable byte-stream service"

4.MicrosoftDocs, ‘How to create a socket’, 09/15/2021

https://docs.microsoft.com/en-us/dotnet/framework/network-programming/how-to-create-a-socket

“The constructor for the Socket class has parameters that specify the address family, socket type, and protocol type that the socket uses to make connections.”

5.Socket Programming, Limi Kalita, 2014.

https://ijcsit.com/docs/Volume%205/vol5issue03/ijcsit20140503462.pdf

“bind() is typically used on the server side, and

associates a socket with a socket address structure,

i.e. a specified local port number and IP address.

 listen() is used on the server side, and causes a

bound TCP socket to enter listening state.

” - page 4806.

6. Uniform Resource Locators (URL), T. Berners-Lee, december 1994, University of Minesota.

https://www.researchgate.net/publication/224001129_Uniform_Resource_Locators_URL_Online

“URLs are used to `locate' resources, by providing an abstract identification of the resource location. Having

located a resource, a system may perform a variety of operations on the resource, as might be characterized

by such words as `access', `update', `replace', `find attributes'. In general, only the `access' method needs to

be specified for any URL scheme.”, page 2.

7. TCP 3-Way Handshake (SYN, SYN-ACK,ACK), Lawrence Williams, October 7, 2021.

https://www.guru99.com/tcp-3-way-handshake.html

“

Syn	Used to initiate and establish a connection. It also helps you to synchronize sequence numbers between devices.

”