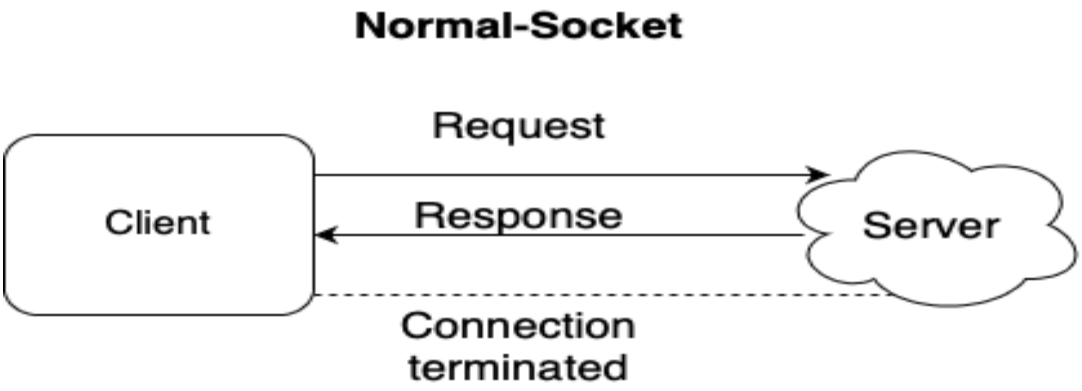
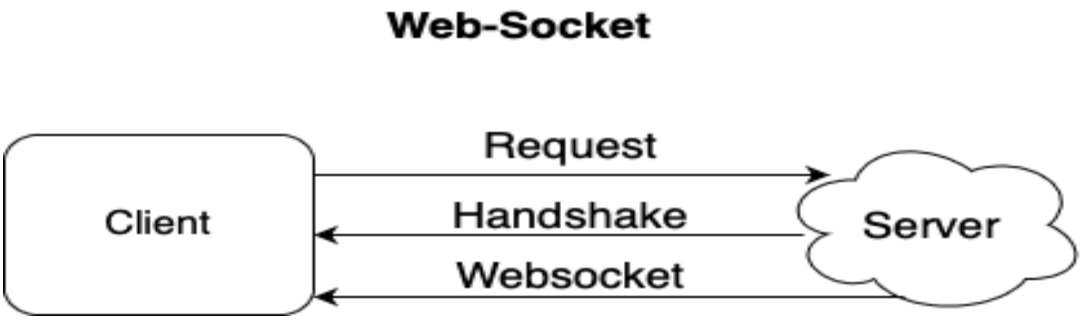
WebSocket Report

1. What are the differences between a WebSocket and a normal network socket?

One of the main differences between web sockets and normal sockets is that normal sockets have a lot more freedom than web sockets do. WebSocket's on the other hand operate in a browser and are event driven[*Malte Ubl*]. Sockets don't operate solely in browsers; they are a lot more dynamic than that. They are not restricted to running in browsers, they can implement a whole variety of connections. In normal sockets you must specify the type of connection (e.g., Internet) and you must also state the protocol involved in the socket (e.g., if its TCP/UDP). WebSocket's have specific events that can be executed. An example of an event involved in a web socket is 'open' where an event is fired when the socket first connects. While they do share differences a web socket is fundamentally just a normal socket, that uses a socket in order for a web sockets to be implemented. A WebSocket is used as a form of communication between two applications and most notably between a client and a server. Normal sockets on the other hand connects two individual nodes and allows them to connect and communicate with each other. One would have to use web sockets if they were to implement an application online through a browser as a normal web-browser can't open a normal TCP or UDP socket. This is because it would provide a major security threat, as random TCP connections being open all over the internet would not do well with security measures in place inside of web-browsers. For normal web applications it is necessary to implement a server-side and a client side. For web sockets this functions properly as you can use Socket.IO to establish a connection between the server and the clients. On the contrary normal sockets cannot use Socket.IO to establish such a connection.

WebSocket's are built on a protocol that defines a connection, known as a "handshake" and a message which is known as a "frame"[*Gabriel L. Muller*]. Both of these process go through the handshake procedure in order to accept a connection and then use the message "frame" in order for messages to pass back and forth. Sockets on the other hand work by creating a socket through a family value and a protocol type that are passed through as parameters. Sockets are then binded(by port and address) and must listen and accept for new connections, in order for sockets to communicate. Sockets can then duly send and receive messages between each other afterwards. WebSocket's were created before sockets. A major problem in sockets was that it was quite difficult to maintain a constant open connection between a client and a server. This was because a server can only respond to a client's request, they did not have the capabilities to send data from a server to a client. A normal socket's method to deal with this issue was to poll. WebSocket's provide full-duplex communication allowing sockets to communicate with each other simultaneously, while sockets provide half-duplex communication, allowing sockets to communicate but not simultaneously. This concept will be covered more in the second question of this report.



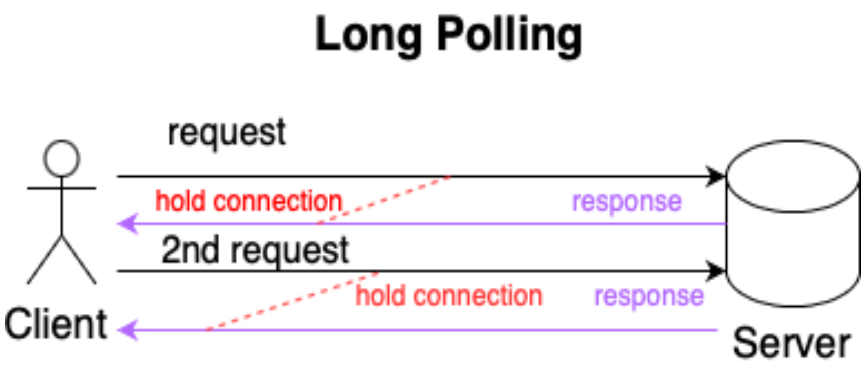


Another difference between sockets and web sockets is the time that they were created. Sockets were created around 1989 through UC Berkley which released versions of it on its operating system. WebSocket's were finalized in December 2011. The difference in dates between a normal socket and a WebSocket highlight the modernity of WebSocket's, while normal sockets being created almost thirty years prior showcase the outdated nature of normal sockets in today's day and age.

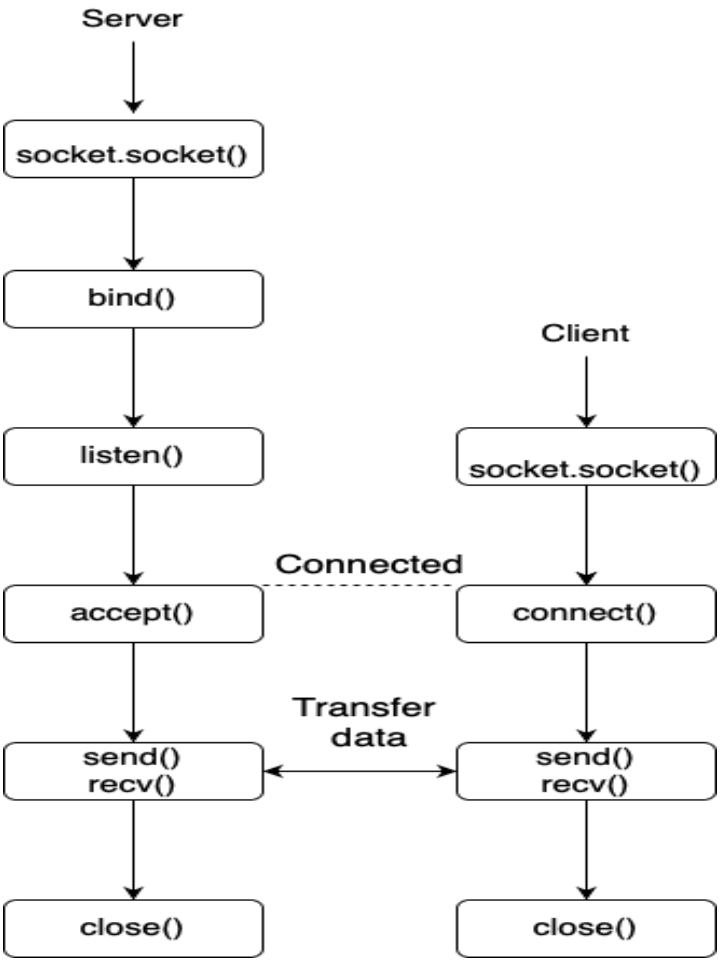
2. *WebSocket's provide full-duplex communication, what does this mean, and how did we achieve similar functionality before WebSocket's were released?*

Full-duplex communication means that two or more entities can communicate with each other simultaneously. This basically means that all entities in the system can communicate with each other in all directions, allowing for an open communication between entities in a system. This is an important concept when it comes to sockets and web sockets as sockets main purpose is to allow an easy connection between every socket. Both entities communicating with each other can both send and receive information simultaneously. Without full-duplex communication sockets had to receive the data, process it and then send a response. With full-duplex communication implemented sockets can receive the data and send a message back to the entity that sent that data at the exact same time. Before web sockets were released around 2011, ‘polling’ and ‘long polling’ were used as a precursor for full-duplex communication in

WebSocket's. Polling allows simultaneous connection with all connections in the queue who are listening within the socket. Therefore, it is a good way of counteracting a normal sockets ineptitude for communication between a server and a client. 'Polling' however is very wasteful. Polling is wasteful because a server must first establish a connection with a client before sending across their information. For every new connection established HTTP headers must be parsed, a query for new data must be performed and a response must be generated and delivered. As you can see this can be quite strenuous and excessive. Long polling is a technique used by sockets that looks to expand on the polling technique. Long polling works by holding a client's connection with the server for as long as it possibly can, thus helping to alleviate the constraints that lie in polling, where a server has to constantly establish a connection with a client[*Kieran Kilbride-Singh*]. Web sockets were created and helped relieve this problem, they keep the connection open between a server and a client while simultaneously removing any latency problems between the two. Alongside this important attribute, headers are not sent when a client needs information from the server. Both of these elements make clients sending information to the server less costly. These are big differences between web sockets and normal sockets. Polling and Long polling in normal sockets are a lot more resource exhaustive compared to web sockets.



Another method for implementing the process of full-duplex before it was invented was streaming. Streaming allows for a free pipe for data to be passed through between two entities. Both entities can send information back and forth, and no data will be lost. To enable this sort of connection a tcp socket shares a common ip address and a common port number. Once a connection has been properly established both entities can both send and receive information from each other. Once both entities decide that the communication is over, they can close the connection between each other. This is a good way to implement a server with corresponding clients. The client side of the script can create a socket and connect to the server, they can then start to send and receive messages from the server starting the streaming process. The server-sided script can listen and accept connections, they can then send and receive information and continue the streaming process.



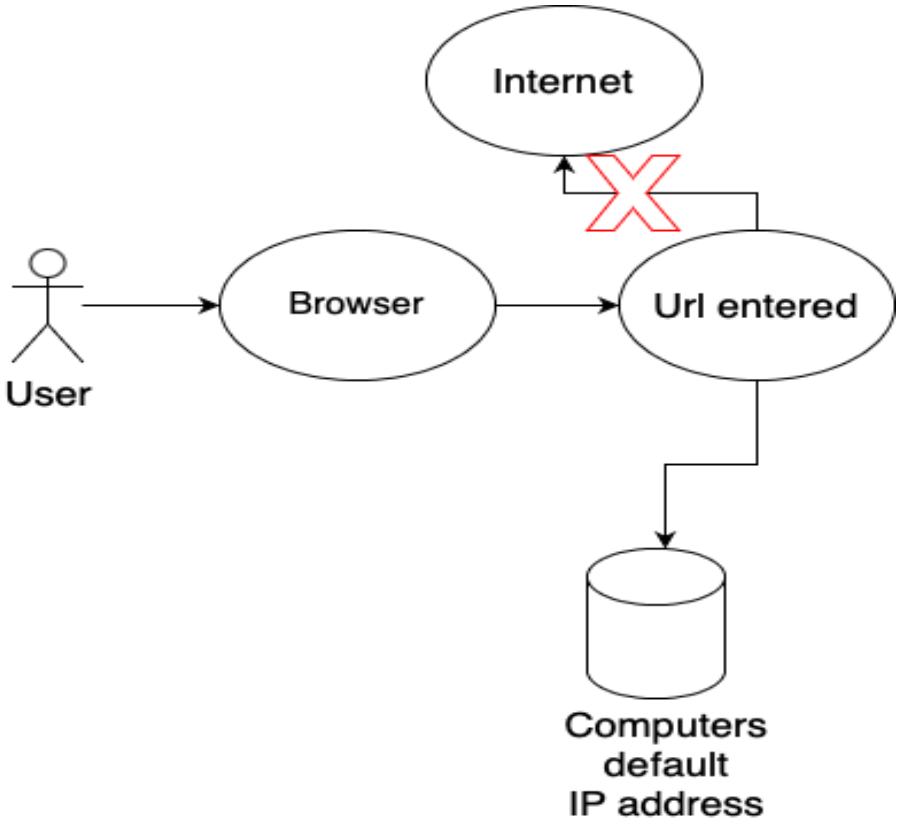
Another method that was used was ‘AJAX’. Ajax is an abbreviation for Asynchronous Javascript and XML[*Darshan G. Puranik*]. Ajax uses a response-request method to transmit communication between two different entities, while full-duplex communication works in real-time to enable communication. This is a simplified version of what full-duplex communication tries to implement. It creates a connection from the socket to the server, and sends request headers and any other optional data, once it gets a response from the server, it will close the connection. The drawback of ‘AJAX’ comes in the form of security. There is high security risk with this process as each request sent is a new TCP connection that is going throughout the internet and might be picked up by attackers. AJAX also provides the benefit of running much faster than websockets and full-duplex communication do nowadays. This is because they provide less overhead and are therefore much faster. Overhead is basically an excess of memory or bandwidth, resulting in a longer computation time. AJAX is a derivative of of JavaScript's XmlHttpRequest which is essentially a java script object. An XmlHttpRequest essentially allows execution of javascript code with completely reloading the webpage, which is very usefully when it comes to sending information from clients to servers using javascript code. Ajax is however half-duplex and not full-duplex. The difference pertaining to half-duplex and full-duplex comes in the lack of ability for both entities to communicate with each other

simultaneously, even though they can communicate with each other effectively. If one or more entities emits information at the same time, then a collision will occur and messages will either be lost or they will be distorted.

3. Detail the steps, requests and protocols involved when you:

Start your app server, visit it in a browser and then send a message.

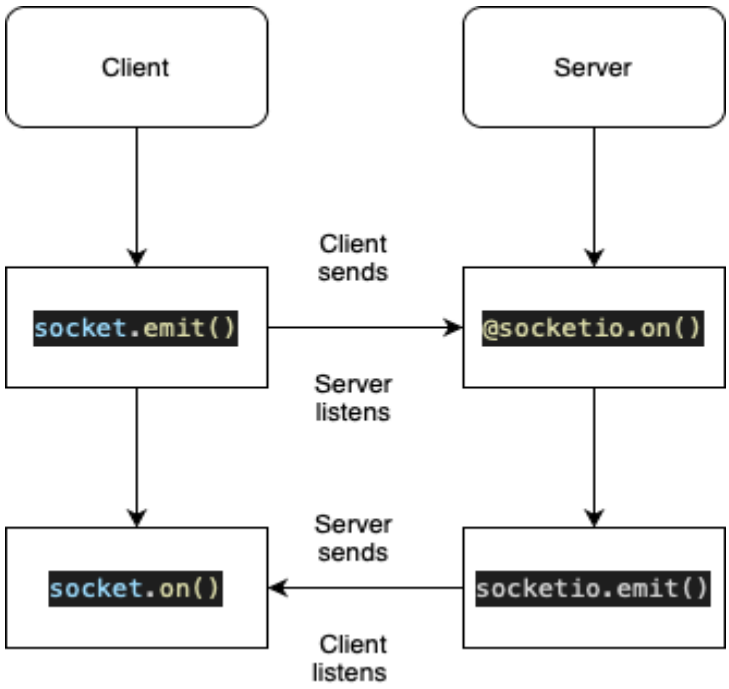
To start my app's server, I simply used terminal on my MacBook and called my server script using python which I installed using pip. I had to ensure that I was in the right directory where my script was by using "cd" followed by the folder's that my script was placed in. I called the server script using python3. I ran the server on my local-host and on the port 5000. The terminal then conveyed various information showing if the server was correctly started or not. Messages like "(40936) wsgi starting up on http://127.0.0.1:5000/" Showed that the server was starting up and indicated the Ip address and the port that the server used to start. A message such as "(40936) accepted ('127.0.0.1', 62720)", showed that the server was accepted and had properly started. The terminal was a good way of knowing what was happening inside the server, it showed a variety of information such as get and post requests, as well as any print statements that had occurred. A localhost means that it is not connected to the internet, rather it is installed on the computer, hence the term 'local'. A localhost uses the internet protocol loopback device which enables TCP/IP applications to communicate with each other[*Joseph Boone*]. An internet protocol is a protocol for routing packets of data between connections which must ensure that they arrive at their correct destination. The default Ip address of a localhost is 127.0.0.1, essentially with localhost you are connecting to your own machine or computer. Instead of sending a request to the internet, you are sending a request to your own machine.



When you want to visit localhost in your browser and connect to the server the first step involved lies in opening your browser. Once you have completed this step one must enter the localhost into the address bar which will usually e located in the top of the browser. This address field is usually known as a URL or a Uniform Resource Locator. The URL that will be entered into this field will be http://127.0.0.1:5000/ which can be found when one first starts the app server. The first section is the hypertext transfer protocol(http) and the next section of the URL is the default ip address of your computer which is 127.0.0.1 the last section is the port number that the server uses, which is usually an arbitrary large number. The first section that defines the http protocol is important as it used to get resources like html documents. Once the app is visited in a browser the protocol will send a request to the server in order to display the html associated with the app[*Tim Berners-Lee*]. The server will then duly send a response and display the html for the application. Once you enter the local address in the URL form and click enter, the request will not be send from your router to the internet, instead it will loopback onto itself aka your own computer. The address will refer back to your own server that one has already started. The first part of the local address '127' informs the TCP protocol that you do not want to access the internet, instead you want to access your own local machine/server.

Once one has visited the app on the localhost, you are met with the html code for the application. From this webpage you can send a message to any of the clients that are connected to the server, and the message will be displayed within the html page and displayed on the screen. To send a message you must first type your message into a html text field and also

enter a username into a different html text field. A ‘send message’ html button can then send this message out to everyone connected to the server. When a user sends a message it calls on a function written in java script in the html-sided script. This function takes the data you submitted into fields and sends a request with the data to the server using sockets and specifically the ‘emit’ function used in sockets. The server will be listening for requests from its clients and will respond to the request by calling the emit function for the html to display the message depending on what channel the user is in[*Flask-SocketIO*]. As we are using socket-io the protocols involved when sending messages are either http or WebSocket. These are both TCP connections. A socketio client can only talk to a socketio server and vice versa. The first step involved in the WebSocket transport include a handshake between server and client which will establish the connection. In order to send and receive the data it will use HTPP long polling. The socket in socketio can upgrade to a WebSocket and duly send and receive data.



References

1. Introducing WebSockets: Bringing Sockets to the Web, Malte Ubl, October 20th,2010.

https://www.html5rocks.com/en/tutorials/websockets/basics/

“The WebSocket specification defines an API establishing "socket" connections between a web browser and a server”

2. HTML5 WebSocket protocol and its application to distributed computing, Cranfield University, Gabriel L. Muller, September 2014

"The protocol consists of an opening handshake followed by message framing, layered over TCP" page 8.

3. Web Sockets vs Long Polling, Kieran Kilbride-Singh , Oct 27 2021.

https://ably.com/blog/websockets-vs-long-polling

"long polling is a technique where the server elects to hold a client's connection open for as long as possible, delivering a response only after data becomes available or a timeout threshold is reache"

4. Real-time Monitoring using AJAX and WebSockets, Darshan G. Puranik, Indiana U.-Purdue U. Indianapolis, may 17 2017

https://silo.tips/download/real-time-monitoring-using-ajax-and-websockets

"Web 2.0 [1] technologies, such as Asynchronous JavaScript and XML (AJAX) [2], are revolutionizing how end-users interact with Web sites and Web applications. Instead of using many different pages and server callbacks to deliver content, Web 2.0 technologies enable Web sites to deliver content in real-time to Web clients while the end-user remains on the same web page."

5. What Is Localhost and How Can You Use It?, HelpDeskGeek, Joseph Boone, October 9th, 2019

https://helpdeskgeek.com/networking/what-is-localhost-and-how-can-you-use-it/

"The localhost – also referred to as 'the loopback address' – is used to establish an IP connection or call, to your own computer or machine. The loopback address is typically used in the context of networking and provides a computer the capability to validate the IP stack."

6.Uniform Resource Locators (URL) [Online], **Tim Berners-Lee,** December 1994

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

"While the syntax for the rest of the URL may vary depending on the particular scheme selected, URL

schemes that involve the direct use of an IP-based protocol to a specified host on the Internet use a common

syntax for the scheme-specific data:

”

7. Flask-SocketIO- GettingStarted

https://flask-socketio.readthedocs.io/en/latest/getting_started.html

“SocketIO event handlers defined as shown in the previous section can send reply messages to the connected client using the `send()` and `emit()` functions.”