WebSockets in System Design

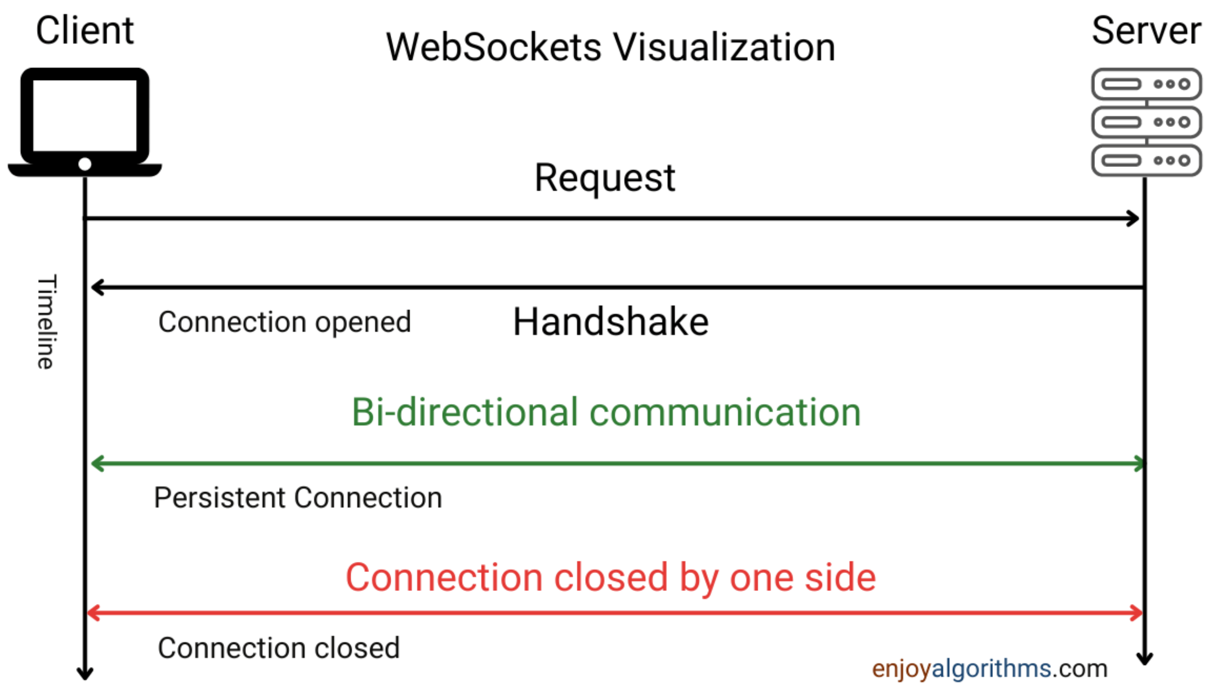
What is WebSockets?

The traditional model of the web is built around the HTTP request and response paradigm. In an HTTP connection, the client sends a request to the server and the server responds with the requested data. This happens in a separate connection for each request and response, which means that HTTP is not well-suited for real-time applications that require continuous, bidirectional communication. To solve this problem of HTTP, we use WebSockets.

WebSocket is a communication protocol that enables bidirectional communication between client and server. It provides a full-duplex communication channel over a single TCP connection and allow for data transfer in both directions. Once WebSocket connection is established, it remains open until either server or client terminates it. This help us in continuous, real-time communication between the client and the server without the need for frequent re-establishing of the connection.

- WebSocket and HTTP protocols have significant differences, but both rely on TCP protocol at the fourth layer (transport layer) of OSI model, and they exist at the seventh layer (application layer).
- WebSocket protocol reduces the overhead associated with HTTP by allowing for low-weight communication between a client and a server.
- By maintaining a persistent single TCP socket connection between the client and server, WebSocket protocol enables efficient distribution of bi-directional messages. This results in a low-latency connection that allows for quick and responsive communication.
- WebSockets have become a crucial component in developing real-time applications, such as multiplayer games or any application that requires quick and reliable data transfer.

How does WebSocket works?



1. To establish a WebSocket connection, client first initiate a handshake with the server by sending an HTTP request with an "Upgrade" header to upgrade the connection to a WebSocket connection. Note: Client opens WebSocket connection by calling WebSocket constructor, which uses the URL schemas "ws:" or "wss:" for WebSockets (Just like "http:" or "https:" for HTTP).
2. Once client request is received by the server, server accepts it by sending a response with a "101 Switching Protocols" status code and a header indicating the use of WebSockets.
3. With established WebSocket connection, both client and server can send messages to each other without the need for a new request-response cycle. The messages are sent in a binary format using the send() method on the connection object. In the latest specification, the WebSocket connection supports sending both string and binary messages using Blob or ArrayBuffer objects.
4. The WebSocket connection remains open until either the client or server closes it.

Key Features of WebSockets

- **Two-Way Data Exchange:** WebSockets are essential in reducing

network traffic by enabling the simultaneous transfer of data in both directions using a single connection.
- **HTTP Compatibility:** WebSockets are highly compatible with previous versions of HTTP connections, enabling us to switch between HTTP and WebSockets.
- **Publish-Subscribe Event Pattern:** WebSockets provide an efficient data transfer model by establishing communication channels. This allows messages to be sent to and from a server, as well as receiving event-driven responses without having to continuously poll the server.

Real-life applications of WebSockets

WebSockets are commonly used in various types of applications that require real-time, bidirectional communication between the client and server. Here are some examples:

- **Real-time messaging:** Web Sockets are essential in enabling real-time communication, such as instant chat applications. They provide efficient and fast data transfer, making it possible to build complex features like encrypted messages, typing indicators, and more.
- **Multiplayer gaming:** Web Sockets allow for low-latency and real-time synchronization of game states between players. This results in seamless and interactive online gaming experiences across multiple devices.
- **Live online maps:** Real-time location data is crucial in building live online maps. Web Sockets allow for efficient and fast transfer of this data, enabling the routing and navigation of moving assets in real-time.
- **Live updates:** Web Sockets are valuable in providing real-time updates for various platforms, such as live election results, live score updates, and more. They ensure users are always kept informed with the latest information.

Conclusion

Web Sockets have genuinely revolutionized web development. Using their stateful and bi-directional nature, they are beneficial in systems that require real-time updates or continuous data streams. However, if we only need to fetch only once, then a simple HTTP request would be considered compared to Web Sockets.

Thanks Suyash Namdeo for his contribution in creating the first version of this content. Please write in the message below if you find anything incorrect or want to share more insight. Enjoy learning. Enjoy algorithms!