

An Overview of Web Sockets: The future of Real-Time Communication

Bhumij Gupta¹, Dr. M.P. Vani²

¹Student, Dept. of Information Technology, Vellore Institute of Technology, Tamil Nadu, India

²Associate Professor, Dept. of Information Technology, Vellore Institute of Technology, Tamil Nadu, India

Abstract - As the development and implementation of HTML 5 (Hypertext Markup Language 5), it has opened a new range of possibilities for real-time communication between client and server. Currently, vastly used methods for asynchronous real-time communication are HTTP Polling and HTTP Long Polling. But a new protocol is recently introduced called Web sockets (WS) and Web sockets secure (WSS). Web Sockets allow full-duplex communication in HTML5 compliant browser over a single socket. This paper will cover the basic overview of Web sockets, and see if it is a better option than its competitors.

Key Words: Web sockets, Real-time communication, HTML5, Duplex communication, HTTP

1. INTRODUCTION

Since a long time the web has contained pages which are static, the facts and the figures are updated once in a while, and the information gets stale and outdated. These pages do not reflect the changes of data in real-time until reloaded. If a user wants the frequent update of data or real-time update of data, they may have to reload the page every second to reflect the changes. In this era, there are certain sectors where real-time communication is very important (real-time stock market prices, real-time transmission of patients' vitals). The progression of real-time data transfer has also allowed us to implement services like Video Conferencing and Voice over Internet Protocol (VoIP).












	Bitcoin (BTC)	\$112,258,470,271	\$6,476.90	0.34%
	Ethereum (ETH)	\$21,079,456,279	\$205.28	1.00%
	XRP (XRP)	\$18,300,359,061	\$0.45754	1.12%
	Bitcoin Cash (BCH)	\$7,685,056,759	\$441.35	1.30%
	EOS (EOS)	\$4,872,438,666	\$5.38	0.89%
	Stellar (XLM)	\$4,647,208,028	\$0.24597	3.75%
	Litecoin (LTC)	\$3,117,595,231	\$53.02	0.30%
	Tether (USD)	\$2,034,815,704	\$0.97996	-0.27%
	Cardano (ADA)	\$1,988,528,820	\$0.07670	2.54%
	Monero (XMR)	\$1,707,956,640	\$103.53	-0.35%
	TRON (TRX)	\$1,586,203,247	\$0.02413	0.56%

Fig-1 Cryptocurrency exchange website which requires web socket for real-time exchange rate update

Over HTTP connection, when a client requires a data, it opens a port, requests for the data, the server sends data on the same port, and then the port on the client side is closed. HTTP is connection-less, which means when a client sends a request to the server it is connected to the server but after the request is sent, no connection exists. And even more, a

server cannot listen to another client until it processes the request of the connected client [2]. HTTP is also stateless, because every request which is made to receive or send data is independent and no data is used from the previous request by the HTTP to make a new request. But in order for the server to respond to the request, it needs some data about the client and the data required, which is sent in the form of HTTP headers of request [2].

2. ACHIEVING REAL-TIME COMMUNICATION

Polling: The earliest and the easiest way to implement real-time communication. The client sends a request to the server. When a server receives this request, it responds with a new message, if there is one, or with an empty response if no new message is available. Again after a short time, the client resends the request to the server again to see if any new messages are available [1]. This short time Δ is called the polling interval.

A shortcoming to this way is if there are no new messages for the client, there are still requests from the client which contains the headers. This increases the load on the server and also consumes bandwidth of the network.

Long Polling: Long Polling came up as a solution to the unnecessary request sent by the client when there is no new data on the server. With long polling, the server doesn't send an empty response when there are no new messages available for a client. Instead, the server holds the request until a new message is available or a timeout expires. This reduces the number of client requests when no new messages are available [1].

A shortcoming to this way is, to keep a connection alive the connection has to be saved locally on the server which requires extra computation and space.

Web Sockets: Sockets come up with the solution to both the problems. With web sockets, we can reduce the metadata (HTTP headers) that are sent in every request (a shortcoming of Polling) and we can also provide full-duplex communication through a single socket (a shortcoming of Long Polling).

3. WEBSOCKETS

Web sockets act like proxies over HTTP, meaning it tunnels a TCP connection over HTTP. It is considered the most ideal option when compared to HTTP Polling and HTTP Long Polling. Web sockets have their own set of protocols, ws:// (web socket) and wss:// (web socket secure). These protocols were standardized by IETF in 2011.

Currently, there are many websites which deploy web sockets to gain real-time communication. But across the internet we never see a website address with wss:// or ws:// but majorly http:// and https://. This is due to the fact that web sockets are deployed as a proxy over http, so it basically uses the same ports as used by http i.e. port 80(HTTP) and port 443(HTTPS). The data transfer through websockets can be analyzed using the network analyzer in a web browser.

A request to upgrade to Websockets is sent and applied during the initial handshake between the client and server. The HTTPS headers contain all the information required by the server to process the request. Once the request is processed, websockets protocol is mapped over the HTTP/HTTPS protocol.

From client (browser) to server:

```
GET /text HTTP/1.1
Upgrade: WebSocket
Connection: Upgrade
```

From server to client (browser):

```
HTTP/1.1 101 WebSocket Protocol Handshake
Upgrade: WebSocket
Connection: Upgrade
```

Fig-2 Standard HTTP header for upgrading to Websockets

Once the websockets is implemented the HTTP header reduces to a size of an additional 2 bytes as compared to normal polling headers which have headers of size minimum of 871 bytes [3]. This massively reduces the server load and also reduces the bandwidth clogging.

At the time of writing, web sockets are supported by almost every mainstream browser.

IE	Edge	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini
		2-3.6		3.1-4			
		4-5	4-14	5-5.1	10.1	3.2-4.1	
6-9		6-10	15	6-6.1	11.5	4.2-5.1	
10	12-16	11-61	16-68	7-11.1	12.1-54	6-11.2	
11	17	62	69	12	55	11.4	all
	18	63-64	70-72	TP		12	

Fig-3 Compatibility of web sockets in browsers [4]

According to a study, WebSocket network consumes 50% less network bandwidth than an AJAX server. Websockets can send up to 215.44% more data samples when consuming the same amount of network bandwidth as AJAX. [5]

WebSocket is also compatible with IoT devices. It can work on MicroPy and Arduino. Also, there are many websockets library made for the clients and servers due to its huge popularity.

4. ANALYSING NETWORK FOR WEBSOCKETS

We will be analyzing network of a cryptocurrency price tracking website https://coins.live which uses websockets for real time price updating. For analyzing the network, we will be using Google Chrome' build 69.0.3497.100

Step 1: Open Google Chrome, and press ctrl+shift+I to open developer tools

Step 2: Switch to network tab to capture data

Step 3: Navigate to https://coins.live and you will see traffic being captured in real time.

Step 4: Click on any packet to analyze it.

4.1 ANALYSING HEADERS OF FIRST PACKET CAPTURED

```
▼ Request Headers
:authority: coins.live
:method: GET
:path: /
:scheme: https
accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
accept-encoding: gzip, deflate, br
accept-language: en-GB,en-US;q=0.9,en;q=0.8
cache-control: no-cache
cookie: __cfduid=de82dddbff7c7669c546cb24f01f362851540041004; __ga=GA1.2.1927890955.1540041009; __omappvp=zFZ28Cx6k94aMoDr8u67CAzB4PD1w0ZPw8K2JAIFZxG1HibCq80z3eEAkyzoF3LB1UezMRq0hZwJ0wJMausD1QtY6J79yYpn; __smVID=f435bd18f9a1c22beb420bcb133f45f7e88b9ad925c584f0244f7e5f528bb08; __z1cmid=oyhjKN2gTyLbhR; __gid=GA1.2.1666706590.1540274256; __smToken=6YpExrWE1Jvw5YMsx4NCMZES: easvlogin session=99aa0963e872106094a5746ednt: 1
pragma: no-cache
upgrade-insecure-requests: 1
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36
```

Fig-4 Request Headers of HTTP request packet

```
▼ General
Request URL: https://coins.live/
Request Method: GET
Status Code: 200
Remote Address: 104.18.35.79:443
Referrer Policy: no-referrer-when-downgrade

▼ Response Headers
cache-control: no-store, no-cache, must-revalidate
cf-cache-status: MISS
cf-ray: 46e99e42ed56aa5c-SIN
content-encoding: br
content-type: text/html; charset=UTF-8
date: Wed, 24 Oct 2018 04:14:44 GMT
expect-ct: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
expires: Thu, 19 Nov 1981 08:52:00 GMT
pragma: no-cache
server: cloudflare
status: 200
vary: Accept-Encoding
via: 1.1 google
```

Fig-5 Response Header of HTTP response packet

In the captured headers we can see it contains all the information about the client and network that the server needs to provide the data like scheme, user-agent, accept-encoding, accept-language and in response server sends the data specifying the content-type, timestamp, status of request, server type, etc.

Step 5: To filter out websocket packets and frames, switch to 'ws' tab from 'all' tab.

We can now see the captures websocket handshake packet.

4.2 ANALYZING HEADER OF WEBSOCKET HANDSHAKE PACKET:

Any network that wants to use websocket protocol starts with a handshake. The handshake is a GET request from client to server for upgrading to websocket protocol and the servers' response.



Fig-6 Header of websocket upgrade request packet

In the request header,
Connection: Upgrade
Upgrade: websocket

specifies the request to upgrade to websocket protocol
The client side also sends a key along with upgrade request.
Here,

Sec-WebSocket-Key: Olcmuj47cQjClwMLyQXexg==

The server processes this request and sends back the confirmation for the request.

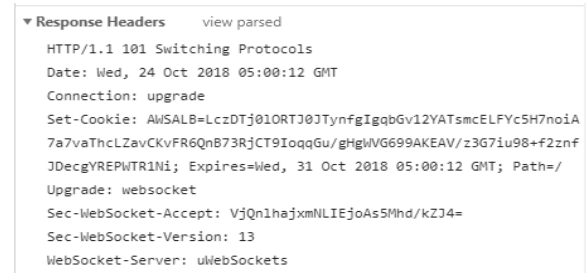


Fig-7 Header of websocket upgrade response packet

The server responds with conformation

101 Switching Protocols

Connection: upgrade

Upgrade: websocket

WebSocket-Server: uWebSockets

i.e. status code 101 for switching protocol and specifying the upgraded protocol

The key sent by the client in the request header is appended to a special globally unique identifier string (GUID) string by server. Then it generates a SHA Hash from the string and sends back to server as

Sec-WebSocket-Accept:

VjQnlhajxmNLIEjoAs5Mhd/kZJ4=

The websocket upgrade request contains all the details about the client needed by server to keep the connection alive and send any further data.

Step-6: To analyze the data frames exchanged through websockets, switch to frames tab to capture real time websocket frames. Click on any frame to see its data.

Data	Length	Time
11540358116687 0 1 0...	467	10:45:16.687
11540358118536 2486...	30	10:45:16.688
11540358118587 2486...	226	10:45:16.730
11540358118763 2484...	4800	10:45:16.914
11540358117105 -184...	4097	10:45:17.104
11540358119061 -184...	302	10:45:19.060
11540358121084 -184...	302	10:45:21.083
11540358122055 -184...	302	10:45:22.054
11540358124054 -184...	302	10:45:24.053
11540358125055 -184...	302	10:45:25.054
11540358127053 -184...	302	10:45:27.052
11540358128062 -184...	302	10:45:28.061
11540358130053 -184...	303	10:45:30.052
11540358131055 -184...	303	10:45:31.055
11540358132055 -184...	303	10:45:32.055

Fig-8 Data frames sent over WebSockets.

Once the handshake has taken place, communication over websockets can start. Since all the data required for sending data to client and keeping the connection alive is exchanged in the handshake, data frames header is reduced to mere

bytes containing opcodes to identify the data contained in the dataframe.

If a connection is to be closed a data frame is sent with close opcode 0x08. After receiving the frame, the client waits for server to shut down its side of connection and then the client shuts the connection.

5. SHORTCOMINGS

Apart from all these advantages, websockets isn't perfect. It is still not compatible with mobile web browsers.

Web sockets are still susceptible to DOS attacks as malicious software can create a large number of web socket connection to the server.

It is also not compatible with various API calls It is also not compatible with REST API.

We often need special configuration for load balancing Websockets introduce a new vulnerability called Cross-Site Websockets Scripting (CRWS).

6. CONCLUSIONS

In a nutshell, Websockets is a revolutionary technology. It is currently the best choice for implementing real-time full-duplex web applications. It reduces the latency and HTTP header load when comparing to alternate technology, and due to its compatibility with other existing services its implementation will keep rising.

It cannot be considered as a replacement to the existing HTTP model but more of as an upgrade to the model.

REFERENCES

- [1] Victoria Pimentel, Bradford G. Nickerson: Communicating and displaying real-time data with WebSocket
- [2] Akshit Grover: Real-time communication with sockets
- [3] Eliot Estep: Mobile HTML5: Efficiency and Performance of WebSockets and Server-Sent Events
- [4] www.caniuse.com
- [5] Darshan G. Puranik, Dennis C. Feiock, James H. Hill: Real-time Monitoring using AJAX and WebSockets