

Web Socket

Web Socket

In this lesson, let's dig deeper into the application level protocols. There are multiple application level protocols such as HTTP, WebSocket, FTP, SMTP, SSH, and WebRTC. Every single one of these protocols makes use of TCP, except WebRTC, which uses UDP.

The problem

Let's consider the scenario of building a chat application. In this context, HTTP works best when dealing with stateless data. It follows a request-response protocol, where the client requests a page and the server responds with the requested page. However, this approach is unidirectional lacks real-time communication capabilities. To stay updated, the client must continually make requests to the server, resulting in significant overhead. This situation is far from ideal, especially in a group chat setting where receiving messages in real time is crucial. Imagine a situation where the client fetches messages from the server every minute. This would be highly suboptimal.

Therefore, while we can leverage HTTP for certain aspects, we encounter a challenge in implementing polling. Polling involves checking the server's status at regular intervals, potentially every minute, to retrieve new messages. Determining the appropriate intervals for checking these messages poses a dilemma. A one-minute gap might be too long, leading to delayed message delivery. On the other hand, checking every second would generate an excessive number of requests, placing a heavy load on the server and consuming unnecessary resources.

In summary, while HTTP serves its purpose, it presents limitations when it comes to real-time chat applications. The reliance on polling raises questions about choosing the appropriate intervals for message checks, balancing efficiency and responsiveness without overwhelming the system.

WebSocket as the solution

WebSocket on the other hand, is a distinct protocol that facilitates two-way communication between a client and a server, enabling seamless back-and-forth data

transmission. This capability proves highly valuable in scenarios that demand real-time updates, such as chat applications, live streaming apps, or real-time gaming apps.

Establishing a WebSocket connection

Indeed, WebSocket can be seen as an “upgrade” from HTTP in the sense that it starts with a standard HTTP request and then transitions to a WebSocket connection if both the client and server support the protocol. This allows for continuous, bidirectional communication between the client and server.

Major web browsers like Google Chrome, Firefox, and Edge have built-in support for the WebSocket protocol. Additionally, popular server-side web application frameworks such as Node.js, Django, and ASP.NET also offer support for WebSocket. This widespread support enables developers to leverage the benefits of real-time, two-way communication in their web applications.

Below is an overview for establishing a WebSocket connection.

1. **Client Sends a WebSocket Handshake Request:** The client will establish a WebSocket connection by initiating a WebSocket handshake request. This is an HTTP Upgrade request with a few special headers.
2. **Server Response (Handshake Response):** If the server supports the WebSocket protocol and is willing to accept the connection under the conditions the client has requested, it will return with a status code 101, which indicates the server understands the upgrade header field request and indicates which protocol it is switching to.
3. **Data Transfer:** After the handshake, the client and server send data between each other in real time. This is more efficient than constantly opening and closing new HTTP connections. WebSockets are truly bi-directional, and this way the client will not have to keep checking the server.

By default:

- WebSocket (WS) connections use port 80, similar to HTTP.
- WebSocket Secure (WSS) connections use port 443, similar to HTTPS.

It is important to note that while devices and browsers may support WebSocket, the network they're connected to must also allow WebSocket connections. Some restrictive firewalls might block WebSocket connections but because it is so ubiquitous and compatible with existing web infrastructure, it is generally well-supported.

Recall that there are several versions of HTTP, including HTTP/2. The introduction of HTTP/2 allows for multiplexing, which means that multiple requests in parallel can be initiated over a single TCP connection. However, this is not a perfect replacement to WebSocket as they are still very prevalent to this day.

Example from Twitch.com

If we look at a live streaming website where chat is being updated in real-time, the protocol that it is highly likely using is WebSocket, and not HTTP. We can dive a little deeper and prove this. We can use the network tab in developer tools for this. In the network tab, filtering for `ws` reveals that the protocol being used is `websocket`, as WebSocket serve this exact purpose.

The screenshot shows the Twitch.com interface with a live stream by A_Seagull. The browser's developer tools are open, and the network tab is filtered for 'ws'. The network tab shows a websocket connection to irc-ws.chat.twitch.tv. The table below summarizes the network activity:

Name	Status	Protocol	Type	Initiator	Size	Time	Waterfall
irc-ws.chat.twitch.tv	101	websocket	websocket	vendor-ddcb84a...js:1	0 B	Pending	
CpwwdevB8QjMWXFeGf4sI88v1y9QumsBKuCidBY41Gts_6aifm...	200	http/1.1	fetch	amazon-ivs-wasmworker.min-75...	14.1 kB	90 ms	
spade-uri	CORS error		fetch	main.522b023...js:17201	0 B	182 ms	

The visual above demonstrates that websocket is indeed the protocol being used in real-time chat.

twich.tv/a_seagull

Following Browse

The broadcaster has indicated that this channel is intended for mature audiences.

A_Seagull [DROPS] FORMER RANK 1 HALLOWEEN EVENT (NOW PEASANT-TIER GAMER) **LIVE** Overwatch 2 old boomer washed competitive English DropsEnabled 29,704 3:36:41

Follow Subscribe

STREAM CHAT

Twitch Drops Get rewards by watching a_se...

Send a message

0 Chat

Elements Console Recorder Performance insights Sources **Network** Performance Memory Application Security Lighthouse

Preserve log Disable cache No throttling

ws

5000 ms 10000 ms 15000 ms 20000 ms 25000 ms 30000 ms 35000 ms 40000 ms 45000 ms 50000 ms 55000 ms 60000 ms 65000 ms 70000 ms 75000 ms 80000 ms 85000 ms

Name

- irc-ws.chat.twitch.tv
- CpwFdevB8QjMWXFeGf4sl88v1y9QumsBKuCidB...
- spade-uri

Headers Messages Initiator Timing

General

Request URL: wss://irc-ws.chat.twitch.tv/

Request Method: GET

Status Code: 101 Switching Protocols

The visual above demonstrates 101 switching protocol message discussed earlier.

twich.tv/a_seagull

Following Browse

A_Seagull **LIVE** Overwatch 2 old boomer washed competitive English DropsEnabled 29,704 3:36:41

Follow Subscribe

STREAM CHAT

Twitch Drops

Send a message

0 Chat

Elements Console Recorder Performance insights Sources **Network** Performance Memory Application Security Lighthouse

Preserve log Disable cache No throttling

ws

10000 ms 20000 ms 30000 ms 40000 ms 50000 ms 60000 ms 70000 ms 80000 ms 90000 ms 100000 ms 110000 ms 120000 ms 130000 ms 140000 ms

Name

- irc-ws.chat.twitch.tv
- CpwFdevB8QjMWXFeGf4sl88v1y9QumsBKuCidB...
- spade-uri

Headers Messages Initiator Timing

All Enter regex, for example: (web)?socket

Data

	Length	Time
@badge-info=;badges=;client-nonce=8c85e76ae14cb3f364be4305b8ce1ad1;color=#DAA520;display-name=hires24;emotes=41:0-7;first-msg=0;flags=j...	374	14:47:22...
@badge-info=;badges=no_audio/1;client-nonce=0ba4cf3073783e4ceff88dca8f2165;color=#0000FF;display-name=Wollmilchsau1337;emotes=245:36-...	454	14:47:22...
@badge-info=subscriber/5;badges=subscriber/3,premium/1;client-nonce=a8245831a58b2a6c9d4fd85c2e6861c7;color=#0000FF;display-name=vorod13...	392	14:47:26...
@badge-info=subscriber/24;badges=subscriber/24,bits/10000;color=#5F9EA0;display-name=Childish7;emotes=first-msg=0;flags=jd=0f1d5f0e-4859-4...	405	14:47:29...
@badge-info=;badges=glhf-pledge/1;client-nonce=9d49c0bf8765a3027af0193a2ead3b59;color=#00FF7F;display-name=jackohlantern69;emotes=first...	694	14:47:37...
@badge-info=;badges=premium/1;client-nonce=eda5c5e2bf8bf3f3fec0dc10f11303b3;color=#8A2BE2;display-name=Donatello'sStickThing;emotes=;firs...	462	14:47:38...
@badge-info=subscriber/5;badges=subscriber/3,premium/1;client-nonce=b51e550ec613d26d6e93b6d8dd2fb8c;color=#0000FF;display-name=vorod13...	405	14:47:40...

3 / 222 requests 14.1 kB / 264 kB transferred 20.5 k

The visual above denotes the messages sent in the chat.