Exploring WebSocket

This blog post was organized with the assistance of ChatGPT-4o.

Hello, I' m Li Zhiwei. As the founder and CTO of CodeReview platform and a former engineer at LeanCloud, I have extensive experience with WebSocket, especially through the development of the IM SDK.

The Relevance of WebSocket

WebSocket is a protocol providing full-duplex communication channels over a single TCP connection. It' s widely used in modern applications requiring real-time interaction such as instant messaging, real-time comments, multiplayer games, collaborative editing, and live stock prices.

Modern Applications of WebSocket

WebSocket is extensively applied in: - **Instant Messaging (IM)** - **Real-time Comments** - **Multiplayer Games** - **Collaborative Editing** - **Real-time Stock Prices**

Evolution of WebSocket

Polling: The client frequently requests updates from the server. **Long Polling:** The server holds the request open until new information is available. **HTTP Bi-directional Connections:** Requires multiple connections for sending and receiving, and HTTP headers with each request. **Single TCP Connection (WebSocket):** Overcomes limitations of HTTP bi-directional connections, offering higher real-time capabilities and lower latency.

Implementing WebSocket on iOS

Popular iOS WebSocket Libraries: - **SocketRocket (Objective-C, 4910 Stars)** - **Starscream (Swift, 1714 Stars)** - **SwiftWebSocket (Swift, 435 Stars)**

Using SRWebSocket

1. **Initialization and Connection:**

```
SRWebSocket *webSocket = [[SRWebSocket alloc] initWithURLRequest:[NSURLRequest requestWithURL:[NSUR
webSocket.delegate = self;
[webSocket open];
```

2. **Sending Messages:**

```
[webSocket send:@"Hello, World!"];
```

3. **Receiving Messages:** Implement the `SRWebSocketDelegate` methods to handle incoming messages and events.

4. **Error Handling and Event Notifications:** Properly handle errors and notify users of connection issues.

Detailed WebSocket Protocol Explanation

WebSocket operates on top of TCP and introduces several enhancements: - **Security Model:** Adds a browser-based origin security model. - **Address and Protocol Naming:** Supports multiple services on a single port and multiple domains on a single IP address. - **Frame Mechanism:** Enhances TCP with an IP packet-like frame mechanism without length limitations. - **Closing Handshake:** Ensures a clean closure of connections.

WebSocket Protocol Core

1. Handshake: The WebSocket handshake uses the HTTP Upgrade mechanism: - **Client Request:** `http GET /chat HTTP/1.1 Host: server.example.com Upgrade: websocket Connection: Upgrade Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ== Origin: http://example.com Sec-WebSocket-Protocol: chat, superchat Sec-WebSocket-Version: 13`

- **Server Response:** `http HTTP/1.1 101 Switching Protocols Upgrade: websocket Connection: Upgrade Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo= Sec-WebSocket-Prot chat`

2. Data Transmission: WebSocket frames can contain UTF-8 text, binary data, and control frames like close, ping, and pong.

3. Security: Browsers automatically add the Origin header, which cannot be forged by other clients.

WebSocket URIs

- **ws-URI:** `ws://host:port/path?query`
- **wss-URI:** `wss://host:port/path?query`

WebSocket Frame Protocol

Frame Structure: - **FIN (1 bit):** Indicates if this is the final fragment in a message. - **RSV1, RSV2, RSV3 (1 bit each):** Reserved for future use. - **Opcode (4 bits):** Defines the interpretation of the payload data. - `0x0`: Continuation frame - `0x1`: Text frame - `0x2`: Binary frame - `0x8`: Connection close - `0x9`: Ping - `0xA`: Pong - **Mask (1 bit):** Indicates if the payload data is masked. - **Payload Length (7 bits):** Length of the payload data.

Masking Key: Used to prevent man-in-the-middle attacks by masking frames from the client.

Closing Handshake

Close Frame: - Can include a body indicating the reason for closure. - Both sides must send and respond to close frames.

Examples

Example 1: Single-frame Unmasked Text Message

```
0x81 0x05 0x48 0x65 0x6c 0x6c 0x6f
```

Contains "Hello"

Example 2: Single-frame Masked Text Message

```
0x81 0x85 0x37 0xfa 0x21 0x3d 0x7f 0x9f 0x4d 0x51 0x58
```

Contains "Hello" with a masking key

Example 3: Fragmented Unmasked Text Message

```
0x01 0x03 0x48 0x65 0x6c
0x80 0x02 0x6c 0x6f
```

Contains "Hel" and "lo" in two frames

Advanced Topics

Masking and Unmasking: - Masking is used to prevent man-in-the-middle attacks. - Each frame from the client must be masked. - The masking key is chosen randomly for each frame.

Fragmentation: - Used to send messages of unknown length. - A fragmented message starts with a frame where FIN is 0 and ends with a frame where FIN is 1.

Control Frames: - Control frames (like close, ping, and pong) have specific opcodes. - These frames help manage the state of the WebSocket connection.

Augmented Backus-Naur Form

- WebSocket uses ABNF (Augmented Backus-Naur Form) for defining message frames and data structures.

Implementing WebSocket in Different Scenarios

1. Real-time Chat Applications: - WebSocket can maintain an open connection to deliver messages instantly.

2. Live Updates: - Stock prices, sports scores, and other real-time data can be pushed to clients as soon as they change.

3. Multiplayer Games: - Synchronizes game state across multiple players with minimal latency.

Conclusion

WebSocket significantly enhances real-time communication capabilities for web and mobile applications by providing a robust and efficient protocol built on top of TCP. By understanding and implementing WebSocket, developers can create more responsive and interactive applications. Whether you are developing for IM, real-time commenting, or multiplayer gaming, WebSocket is a powerful tool in your development arsenal.

For further details, refer to the WebSocket RFC and explore libraries like SocketRocket on GitHub.

Acknowledgements

Thank you for following along. Feel free to reach out on GitHub or Weibo for more insights and discussions.