探索 WebSocket

這篇博文是由 ChatGPT-4o 協助整理的。

介紹

大家好，我是李智維。作為 CodeReview 平台的創始人兼 CTO，以及前 LeanCloud 工程師，我在 WebSocket 方面有豐富的經驗，尤其是在 IM SDK 的開發過程中。

WebSocket 的重要性

WebSocket 是一種在單一 TCP 連接上提供全雙工通信信道的協議。它被廣泛應用於需要實時交互的現代應用中，如即時通訊、實時評論、多玩家遊戲、協作編輯和實時股票價格。

WebSocket 的現代應用

WebSocket 廣泛應用於以下領域：- **即時通訊（IM）** - **實時評論** - **多玩家遊戲** - **協作編輯** - **實時股票價格**

WebSocket 的演變

輪詢：客戶端頻繁請求服務器獲取更新。**長輪詢：**服務器保持請求打開，直到有新信息可用。**HTTP 雙向連接：**需要多個連接進行發送和接收，並且每個請求都包含 HTTP 頭。**單一 TCP 連接（WebSocket）：**克服了 HTTP 雙向連接的局限性，提供了更高的實時能力和更低的延遲。

在 iOS 上實現 WebSocket

流行的 iOS WebSocket 庫： - **SocketRocket（Objective-C，4910 Stars）** - **Starscream（Swift，1714 Stars）** - **SwiftWebSocket（Swift，435 Stars）**

使用 SRWebSocket

1. **初始化和連接：**

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

2. **發送消息：**

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

3. **接收消息：**實現 `SRWebSocketDelegate` 方法來處理傳入的消息和事件。
4. **錯誤處理和事件通知：**適當處理錯誤並通知用戶連接問題。

詳細的 WebSocket 協議解釋

WebSocket 運行在 TCP 之上，並引入了幾個增強功能：- **安全模型：**增加了基於瀏覽器的源安全驗證模型。- **地址和協議命名：**支持單個端口上的多個服務和單個 IP 地址上的多個域名。- **幀機制：**通過 IP 包類似的幀機制增強了 TCP，沒有長度限制。- **關閉握手：**確保連接的乾淨關閉。

WebSocket 協議核心

1. 握手： WebSocket 握手使用 HTTP 升級機制：- **客戶端請求：** `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`

- **服務器響應：** `http HTTP/1.1 101 Switching Protocols Upgrade: websocket Connection: Upgrade Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo= Sec-WebSocket-Prot chat`

2. 數據傳輸： WebSocket 幀可以包含 UTF-8 文本、二進制數據和控制幀，如關閉、ping 和 pong。

3. 安全：瀏覽器自動添加 Origin 頭，這無法被其他客戶端偽造。

WebSocket URI

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

WebSocket 幀協議

幀結構： - **FIN（1 位）：**表示這是消息的最後一個片段。- **RSV1, RSV2, RSV3（各 1 位）：**保留用於未來使用。- **Opcode（4 位）：**定義有效載荷數據的解析方式。- `0x0`：繼續幀 - `0x1`：文本幀 - `0x2`：二進制幀 - `0x8`：連接關閉 - `0x9`： ping - `0xA`： pong - **Mask（1 位）：**表示有效載荷數據是否被遮罩。- **有效載荷長度（7 位）：**有效載荷數據的長度。

遮罩鍵：用於通過遮罩客戶端的幀來防止中間人攻擊。

關閉握手

關閉幀： - 可以包含表示關閉原因的主體。- 雙方必須發送和響應關閉幀。

示例

示例 1：單幀未遮罩文本消息

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

包含 “Hello”

示例 2：單幀遮罩文本消息

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

包含 “Hello” ，帶遮罩鍵

示例 3：分片未遮罩文本消息

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

分片包含 “Hel” 和 “lo” 兩幀

高級主題

遮罩和解遮罩： - 遮罩用於防止中間人攻擊。- 每個來自客戶端的幀都必須被遮罩。- 每幀的遮罩鍵是隨機選擇的。

分片： - 用於發送未知長度的數據。- 分片消息從 FIN 為 0 的幀開始，到 FIN 為 1 的幀結束。

控制幀： - 控制幀（如關閉、ping 和 pong）有特定的操作碼。- 這些幀用於管理 WebSocket 連接的狀態。

擴展性

擴展數據可以放在消息體的應用數據前： - 保留位可以控制每個幀。- 保留一些操作碼供未來定義。- 如果需要更多操作碼，可以使用保留位。

發送： - 必須確保連接是 OPEN 狀態。- 數據封裝在幀中，數據過大時可以選擇分片發送。- 第一幀的值必須正確，告知接收端數據類型（文本或二進制）。- 最後一幀的 FIN 必須設為 1。

關閉握手： - 雙方都可以發送關閉幀。- 發送關閉幀後，不再發送任何數據。- 接收到關閉幀後，丟棄以後收到的任何數據。

關閉連接： - 關閉 WebSocket 連接，即關閉底下的 TCP 連接。- 發送或收到關閉幀後，WebSocket 連接狀態為正在關閉。- 當底下的 TCP 連接關閉後，WebSocket 連接狀態為已關閉。

參考資料

- WebSocket RFC : RFC6455
- 知乎《WebSocket 是什麼原理 ?》：知乎鏈接
- SocketRocket: GitHub 鏈接

致謝

感謝大家的關注。如果有更多問題或討論，歡迎在 GitHub 或微博上與我交流。