探索 WebSocket

这篇博文是由 *ChatGPT-4o* 协助整理的。

介绍

大家好，我是李智维。作为 CodeReview 平台的创始人兼 CTO，以及前 LeanCloud 工程师，我在 WebSocket 方面有丰富的经验，尤其是在 IM SDK 的开发过程中。

WebSocket 的重要性

WebSocket 是一种在单一 TCP 连接上提供全双工通信信道的协议。它被广泛应用于需要实时交互的现代应用中，如即时通讯、实时评论、多玩家游戏、协作编辑和实时股票价格。

WebSocket 的现代应用

WebSocket 广泛应用于以下领域：- **即时通讯（IM）** - **实时评论** - **多玩家游戏** - **协作编辑** - **实时股票价格**

WebSocket 的演变

轮询：客户端频繁请求服务器获取更新。**长轮询**：服务器保持请求打开，直到有新信息可用。**HTTP 双向连接**：需要多个连接进行发送和接收，并且每个请求都包含 HTTP 头。**单一 TCP 连接（WebSocket）**：克服了 HTTP 双向连接的局限性，提供了更高的实时能力和更低的延迟。

在 iOS 上实现 WebSocket

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

使用 SRWebSocket

1. **初始化和连接：**

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

2. **发送消息：**

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

3. **接收消息**：实现 `SRWebSocketDelegate` 方法来处理传入的消息和事件。

4. **错误处理和事件通知**：适当处理错误并通知用户连接问题。

详细的 WebSocket 协议解释

WebSocket 运行在 TCP 之上，并引入了几个增强功能：- **安全模型**：增加了基于浏览器的源安全验证模型。- **地址和协议命名**：支持单个端口上的多个服务和单个 IP 地址上的多个域名。- **帧机制**：通过 IP 包类似的帧机制增强了 TCP，没有长度限制。- **关闭握手**：确保连接的干净关闭。

WebSocket 协议核心

1. 握手：WebSocket 握手使用 HTTP 升级机制：- **客户端请求**：`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`

- **服务器响应**：`http HTTP/1.1 101 Switching Protocols Upgrade: websocket Connection: Upgrade Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo= Sec-WebSocket-Prot chat`

2. 数据传输：WebSocket 帧可以包含 UTF-8 文本、二进制数据和控制帧，如关闭、ping 和 pong。

3. 安全：浏览器自动添加 Origin 头，这无法被其他客户端伪造。

WebSocket URI

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

WebSocket 帧协议

帧结构：- **FIN（1 位）**：表示这是消息的最后一个片段。- **RSV1, RSV2, RSV3（各 1 位）**：保留用于未来使用。- **Opcode（4 位）**：定义有效载荷数据的解析方式。- `0x0`：继续帧 - `0x1`：文本帧 - `0x2`：二进制帧 - `0x8`：连接关闭 - `0x9`：ping - `0xA`：pong - **Mask（1 位）**：表示有效载荷数据是否被遮罩。- **有效载荷长度（7 位）**：有效载荷数据的长度。

遮罩键：用于通过遮罩客户端的帧来防止中间人攻击。

关闭握手

关闭帧：- 可以包含表示关闭原因的主体。- 双方必须发送和响应关闭帧。

示例

示例 1：单帧未遮罩文本消息

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

包含“Hello”

示例 2：单帧遮罩文本消息

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

包含“Hello”，带遮罩键

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”两帧

高级主题

遮罩和解遮罩： - 遮罩用于防止中间人攻击。- 每个来自客户端的帧都必须被遮罩。- 每帧的遮罩键是随机选择的。

分片： - 用于发送未知长度的数据。- 分片消息从 FIN 为 0 的帧开始，到 FIN 为 1 的帧结束。

控制帧： - 控制帧（如关闭、ping 和 pong）有特定的操作码。- 这些帧用于管理 WebSocket 连接的状态。

扩展性

扩展数据可以放在消息体的应用数据前： - 保留位可以控制每个帧。- 保留一些操作码供未来定义。- 如果需要更多操作码，可以使用保留位。

发送： - 必须确保连接是 OPEN 状态。- 数据封装在帧中，数据过大时可以选择分片发送。- 第一帧的值必须正确，告知接收端数据类型（文本或二进制）。- 最后一帧的 FIN 必须设为 1。

关闭握手： - 双方都可以发送关闭帧。- 发送关闭帧后，不再发送任何数据。- 接收到关闭帧后，丢弃以后收到的任何数据。

关闭连接： - 关闭 WebSocket 连接，即关闭底下的 TCP 连接。- 发送或收到关闭帧后，WebSocket 连接状态为正在关闭。- 当底下的 TCP 连接关闭后，WebSocket 连接状态为已关闭。

参考资料

- WebSocket RFC : RFC6455
- 知乎《WebSocket 是什么原理?》：知乎链接
- SocketRocket: GitHub 链接

致谢

感谢大家的关注。如果有更多问题或讨论，欢迎在 GitHub 或微博上与我交流。