# WebSocket の探求

このブログ記事は、ChatGPT-4o の協力を得て整理されました。

---

### 紹介

こんにちは、私は李智維です。CodeReview プラットフォームの創設者兼 CTO、そして元 LeanCloud エンジニアとして、WebSocket に関して豊富な経験を持っています。特に IM SDK の開発プロセスにおいて深い知識を持っています。

### WebSocket の重要性

WebSocket は、単一の TCP 接続上で全二重通信チャネルを提供するプロトコルです。これは、インスタントメッセージング、リアルタイムコメント、マルチプレイヤーゲーム、共同編集、リアルタイム株価など、リアルタイムのインタラクションを必要とする現代のアプリケーションで広く使用されています。

### WebSocket の現代的な応用

WebSocket は以下の分野で広く利用されています：- **インスタントメッセージング（IM）** - **リアルタイムコメント** - **マルチプレイヤーゲーム** - **共同編集** - **リアルタイム株価**

### WebSocket の進化

**ポーリング：**クライアントが頻繁にサーバーにリクエストを送り、更新を取得します。**ロングポーリング：**サーバーはリクエストを開いたままにし、新しい情報が利用可能になるまで待機します。**HTTP 双方向接続：**送信と受信のために複数の接続が必要で、各リクエストには HTTP ヘッダーが含まれます。**単一 TCP 接続（WebSocket）：** HTTP 双方向接続の制限を克服し、より高いリアルタイム性と低遅延を提供します。

### iOS で WebSocket を実装する

**人気の iOS WebSocket ライブラリ：** - **SocketRocket（Objective-C，4910 Stars）** - **Starscream（Swift，1714 Stars）** - **SwiftWebSocket（Swift，435 Stars）**

### SRWebSocket の使用

1. **初期化と接続：**

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

2. **メッセージの送信：**

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

3. **メッセージの受信：** `SRWebSocketDelegate` メソッドを実装して、受信したメッセージやイベントを処理します。
4. **エラー処理とイベント通知：**適切にエラーを処理し、接続問題をユーザーに通知します。

### WebSocket プロトコルの詳細な解説

WebSocket は TCP 上で動作し、いくつかの機能強化が導入されています：- **セキュリティモデル：**ブラウザベースのオリジンセキュリティ検証モデルが追加されました。- **アドレスとプロトコルの命名：**単一のポート上での複数のサービスと、単一の IP アドレス上での複数のドメイン名をサポートします。- **フレームメカニズム：** IP パケットに似たフレームメカニズムにより TCP を強化し、長さの制限がありません。- **クローズハンドシェイク：**接続のクリーンな終了を保証します。

### WebSocket プロトコルの核心

**1. ハンドシェイク：** WebSocket のハンドシェイクは HTTP のアップグレードメカニズムを使用します：- **クライアントリクエスト：** `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`

- **サーバー応答：** `http HTTP/1.1 101 Switching Protocols Upgrade: websocket Connection: Upgrade Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo= Sec-WebSocket-Prot chat`

**2. データ転送：** WebSocket フレームには、UTF-8 テキスト、バイナリデータ、およびクローズ、ping、pong などの制御フレームを含めることができます。

**3. 安全性：**ブラウザは自動的に Origin ヘッダーを追加しますが、これは他のクライアントによって偽造されることはありません。

#### WebSocket URI

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

#### WebSocket フレームプロトコル

**フレーム構造：** - **FIN（1 ビット）：**これがメッセージの最後のフラグメントであることを示します。- **RSV1, RSV2, RSV3（各 1 ビット）：**将来の使用のために予約されています。- **Opcode（4 ビット）：**ペイロードデータの解析方法を定義します。- `0x0`：継続フレーム - `0x1`：テキストフレーム - `0x2`：バイナリフレーム - `0x8`：接続終了 - `0x9`： ping - `0xA`： pong - **Mask（1 ビット）：**ペイロードデータがマスクされているかどうかを示します。- **ペイロード長（7 ビット）：**ペイロードデータの長さ。

**マスキングキー:** クライアントのフレームをマスクすることで、中間者攻撃を防ぐために使用されます。

#### クローズハンドシェイク

**クローズフレーム:** - クローズの理由を示すボディを含むことができます。- 両者はクローズフレームを送信し、応答する必要があります。

#### サンプル

#### 例 1：単一フレームのマスクなしテキストメッセージ

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

「Hello」を含む

#### 例 2：単一フレームのマスク付きテキストメッセージ

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

“Hello” を含み、マスクキーが付いています。

### 例 3：分割された非マスクテキストメッセージ

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

この分割メッセージは、「Hel」と「lo」の 2 つのフレームで構成されています。

### 上級トピック

**マスキングとアンマスキング：** - マスキングは中間者攻撃を防ぐために使用されます。- クライアントからの各フレームはマスクされなければなりません。- 各フレームのマスクキーはランダムに選択されます。

**フラグメンテーション（分片）：** - 未知の長さのデータを送信するために使用されます。- フラグメント化されたメッセージは、FIN が 0 のフレームから始まり、FIN が 1 のフレームで終わります。

**制御フレーム：** - 制御フレーム（例：クローズ、ping、pong）には特定のオペコードがあります。- これらのフレームは、WebSocket 接続の状態を管理するために使用されます。

### 拡張性

**拡張データはメッセージボディのアプリケーションデータの前に配置できます：** - 予約ビットを使用して各フレームを制御できます。- 将来の定義のためにいくつかのオペコードを予約できます。- さらに多くのオペコードが必要な場合、予約ビットを使用できます。

**送信：** - 接続が OPEN 状態であることを確認する必要があります。- データはフレームにカプセル化され、データが大きすぎる場合は分割して送信することができます。- 最初のフレームの値は正しく設定し、受信側にデータのタイプ（テキストまたはバイナリ）を通知する必要があります。- 最後のフレームの FIN は 1 に設定する必要があります。

**クローズハンドシェイク:** - 両方の当事者がクローズフレームを送信できます。- クローズフレームを送信した後、それ以上のデータを送信しません。- クローズフレームを受信した後、それ以降に受信したデータは破棄します。

**接続のクローズ：** - WebSocket 接続を閉じることは、基盤となる TCP 接続を閉じることを意味します。- クローズフレームを送信または受信した後、WebSocket 接続の状態は「クローズ中」になります。- 基盤となる TCP 接続が閉じられると、WebSocket 接続の状態は「クローズ済み」になります。

## 参考資料

- WebSocket RFC : RFC6455
- 知乎《WebSocket 是什么原理 ?》：知乎リンク
- SocketRocket: GitHub リンク

## 謝辞

皆さんのご関心に感謝します。もしさらに質問や議論があれば、GitHub や Weibo で私と交流してください。