Exploration de WebSocket

Cet article de blog a été rédigé avec l'assistance de ChatGPT-4o.

Introduction

Bonjour à tous, je suis Li Zhiwei. En tant que fondateur et CTO de la plateforme CodeReview, ainsi qu'ancien ingénieur chez LeanCloud, j'ai une expérience approfondie dans le domaine de WebSocket, en particulier dans le développement de SDK IM.

L'importance de WebSocket

WebSocket est un protocole qui fournit un canal de communication full-duplex sur une seule connexion TCP. Il est largement utilisé dans les applications modernes nécessitant des interactions en temps réel, telles que la messagerie instantanée, les commentaires en temps réel, les jeux multijoueurs, l'édition collaborative et les cours boursiers en temps réel.

Applications modernes de WebSocket

WebSocket est largement utilisé dans les domaines suivants : - **Messagerie instantanée (IM)** - **Commentaires en temps réel** - **Jeux multijoueurs** - **Édition collaborative** - **Prix des actions en temps réel**

L'évolution de WebSocket

Polling : Le client demande fréquemment des mises à jour au serveur. **Long polling :** Le serveur maintient la requête ouverte jusqu'à ce que de nouvelles informations soient disponibles. **Connexion bidirectionnelle HTTP :** Nécessite plusieurs connexions pour l'envoi et la réception, et chaque requête contient des en-têtes HTTP. **Connexion TCP unique (WebSocket) :** Surmonte les limitations de la connexion bidirectionnelle HTTP, offrant une meilleure capacité en temps réel et une latence plus faible.

Implémentation de WebSocket sur iOS

Bibliothèques WebSocket populaires pour iOS : - **SocketRocket (Objective-C, 4910 étoiles)** - **Starscream (Swift, 1714 étoiles)** - **SwiftWebSocket (Swift, 435 étoiles)**

Utilisation de SRWebSocket

1. **Initialisation et Connexion :**

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

2. **Envoyer un message :**

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

3. **Réception des messages :** Implémentez les méthodes du protocole `SRWebSocketDelegate` pour gérer les messages entrants et les événements.

4. **Gestion des erreurs et notifications d'événements :** Gérez correctement les erreurs et informez les utilisateurs des problèmes de connexion.

Explication détaillée du protocole WebSocket

Le protocole WebSocket est un protocole de communication bidirectionnelle qui permet une interaction en temps réel entre un client et un serveur. Contrairement à HTTP, qui est basé sur un modèle requête-réponse, WebSocket établit une connexion persistante entre le client et le serveur, permettant ainsi l'échange de données en temps réel sans avoir à renouveler la connexion à chaque requête.

Fonctionnement de WebSocket

1. **Établissement de la connexion (Handshake)**
   La connexion WebSocket commence par un handshake HTTP. Le client envoie une requête HTTP spéciale au serveur, demandant une mise à niveau vers le protocole WebSocket. Cette requête contient des en-têtes spécifiques, tels que `Upgrade: websocket` et `Connection: Upgrade`, ainsi qu'une clé secrète générée par le client.

```
GET /chat HTTP/1.1
Host: example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Sec-WebSocket-Version: 13
```

Le serveur répond avec une réponse HTTP 101 (Switching Protocols) et fournit une clé de réponse calculée à partir de la clé du client.

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
```

2. **Communication bidirectionnelle**
   Une fois la connexion établie, le client et le serveur peuvent échanger des messages en temps réel. Les messages peuvent être de type texte ou binaire, et sont encapsulés dans des trames WebSocket.

```javascript
// Exemple de code client WebSocket en JavaScript
const socket = new WebSocket('ws://example.com/chat');

socket.onopen = function(event) {
    console.log('Connexion WebSocket établie');
    socket.send('Hello Server!');
};

socket.onmessage = function(event) {
    console.log('Message reçu du serveur : ' + event.data);
};

socket.onclose = function(event) {
    console.log('Connexion WebSocket fermée');
};
```

3. **Fermeture de la connexion**
   La connexion WebSocket peut être fermée par l'un des deux côtés (client ou serveur) en

envoyant une trame de fermeture. Une fois la connexion fermée, aucune donnée ne peut plus être échangée.

```
// Fermeture de la connexion côté client
socket.close();
```

Avantages de WebSocket

- **Communication en temps réel** : WebSocket permet une communication en temps réel entre le client et le serveur, ce qui est idéal pour les applications comme les chats, les jeux en ligne, ou les tableaux de bord en temps réel.
- **Réduction de la latence** : Contrairement à HTTP, où chaque requête nécessite une nouvelle connexion, WebSocket maintient une connexion persistante, réduisant ainsi la latence.
- **Efficacité** : WebSocket utilise moins de bande passante et de ressources système par rapport à des techniques comme le polling ou le long polling.

Cas d'utilisation courants

- **Chats en ligne** : Les applications de chat utilisent souvent WebSocket pour permettre aux utilisateurs de communiquer en temps réel.
- **Jeux en ligne** : Les jeux multijoueurs en temps réel peuvent utiliser WebSocket pour synchroniser les actions des joueurs.
- **Notifications en temps réel** : Les applications qui nécessitent des notifications instantanées, comme les réseaux sociaux, peuvent utiliser WebSocket pour informer les utilisateurs de nouveaux messages ou événements.

En résumé, WebSocket est un protocole puissant pour les applications nécessitant une communication en temps réel entre le client et le serveur. Sa capacité à maintenir une connexion persistante et à réduire la latence en fait un choix privilégié pour de nombreuses applications modernes.

WebSocket fonctionne sur TCP et introduit plusieurs améliorations : - **Modèle de sécurité :** Ajoute un modèle de vérification de sécurité basé sur l'origine du navigateur. - **Adressage et nommage des protocoles :** Prend en charge plusieurs services sur un seul port et plusieurs domaines sur une seule adresse IP. - **Mécanisme de trames :** Améliore TCP avec un mécanisme de trames similaire aux paquets IP, sans limite de longueur. - **Poignée de main de fermeture :** Assure une fermeture propre de la connexion.

Le cœur du protocole WebSocket

1. Poignée de main : La poignée de main WebSocket utilise le mécanisme de mise à niveau HTTP : - **Requête du client :** `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`

- **Réponse du serveur :** `http HTTP/1.1 101 Switching Protocols Upgrade: websocket Connection: Upgrade Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo= Sec-WebSocket-Protocol: chat`

2. Transfert de données :
Les trames WebSocket peuvent contenir du texte UTF-8, des données binaires et des trames de contrôle telles que la fermeture, le ping et le pong.

3. Sécurité : Le navigateur ajoute automatiquement l'en-tête Origin, qui ne peut pas être falsifié par d'autres clients.

URI WebSocket

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

Protocole de trames WebSocket

Structure de la trame : - **FIN (1 bit) :** Indique qu'il s'agit du dernier fragment du message. - **RSV1, RSV2, RSV3 (1 bit chacun) :** Réservés pour une utilisation future. - **Opcode (4 bits) :** Définit la manière dont les données de la charge utile doivent être interprétées. - `0x0` : Trame de continuation - `0x1` : Trame de texte - `0x2` : Trame binaire - `0x8` : Fermeture de la connexion - `0x9` : Ping - `0xA` : Pong - **Mask (1 bit) :** Indique si les données de la charge utile sont masquées. - **Longueur de la charge utile (7 bits) :** Longueur des données de la charge utile.

Clé de masquage : Utilisée pour empêcher les attaques de type "man-in-the-middle" en masquant les trames du client.

Fermeture de la poignée de main

Fermeture de la trame : - Peut inclure un corps indiquant la raison de la fermeture. - Les deux parties doivent envoyer et répondre à la trame de fermeture.

Exemple

Exemple 1 : Message texte non masqué sur une seule trame

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

Contient “Hello”

Exemple 2 : Message texte masqué sur une seule trame

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

Contient “Hello”, avec une clé de masquage

Exemple 3 : Message texte fragmenté non masqué

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

Les fragments contiennent deux trames : “Hel” et “lo”.

Sujets avancés

Masquage et Démasquage : - Le masquage est utilisé pour prévenir les attaques de type “man-in-the-middle”. - Chaque trame provenant du client doit être masquée. - La clé de masquage pour chaque trame est choisie de manière aléatoire.

Fragmentation : - Utilisé pour envoyer des données de longueur inconnue. - Un message fragmenté commence par une trame avec FIN à 0 et se termine par une trame avec FIN à 1.

Frames de contrôle : - Les frames de contrôle (comme close, ping et pong) ont des codes d’opération spécifiques. - Ces frames sont utilisées pour gérer l’état de la connexion Web-Socket.

Extensibilité

Les données d'extension peuvent être placées avant les données d'application dans le corps du message : - Les bits réservés peuvent contrôler chaque frame. - Certains codes d'opération sont réservés pour des définitions futures. - Si davantage de codes d'opération sont nécessaires, les bits réservés peuvent être utilisés.

Envoi : - Il est impératif de s'assurer que la connexion est en état OPEN. - Les données sont encapsulées dans des trames, et en cas de données trop volumineuses, il est possible de choisir de les envoyer en fragments. - La valeur de la première trame doit être correcte, informant le récepteur du type de données (texte ou binaire). - Le FIN de la dernière trame doit être défini à 1.

Fermeture de la poignée de main : - Les deux parties peuvent envoyer une trame de fermeture. - Après l'envoi de la trame de fermeture, aucune donnée supplémentaire ne doit être envoyée. - Après avoir reçu une trame de fermeture, toute donnée reçue par la suite doit être ignorée.

Fermeture de la connexion : - Fermer la connexion WebSocket signifie fermer la connexion TCP sous-jacente. - Après l'envoi ou la réception d'une trame de fermeture, l'état de la connexion WebSocket passe à "en cours de fermeture". - Lorsque la connexion TCP sous-jacente est fermée, l'état de la connexion WebSocket devient "fermée".

Références

- WebSocket RFC : RFC6455
- Zhihu□WebSocket □□□□□□□□Lien Zhihu
- SocketRocket: Lien GitHub

Remerciements

Merci à tous pour votre attention. Si vous avez d'autres questions ou souhaitez discuter, n'hésitez pas à me contacter sur GitHub ou Weibo.