ISEN | école d'ingénieurs
ALL IS DIGITAL!

Communications Web

WebSocket

ISEN Ouest

thibault.napoleon@isen-ouest.yncrea.fr

Sommaire I

ISEN OUEST

Sommaire I

ISEN OUEST

WebSocket

ISEN OUEST

Définition

Définition

WebSocket est un protocole de communications informatique permettant de créer des canaux de communication *full-duplex* au-dessus d'une connexion *TCP*.

Même si ce protocole est défini, au départ, pour les applications *Web*, il peut être utilisé dans n'importe quelle application client/serveur.

Normalisation :

Le protocole *WebSocket*, créé en 2008, a été normalisé en 2011 à travers la *RFC* 6455.

L'interface de programmation est toujours en cours de standardisation par le *W3C*.

WebSocket

ISEN OUEST

Schéma de fonctionnement

Trois étapes :

- **Connexion (*Handshake*).**

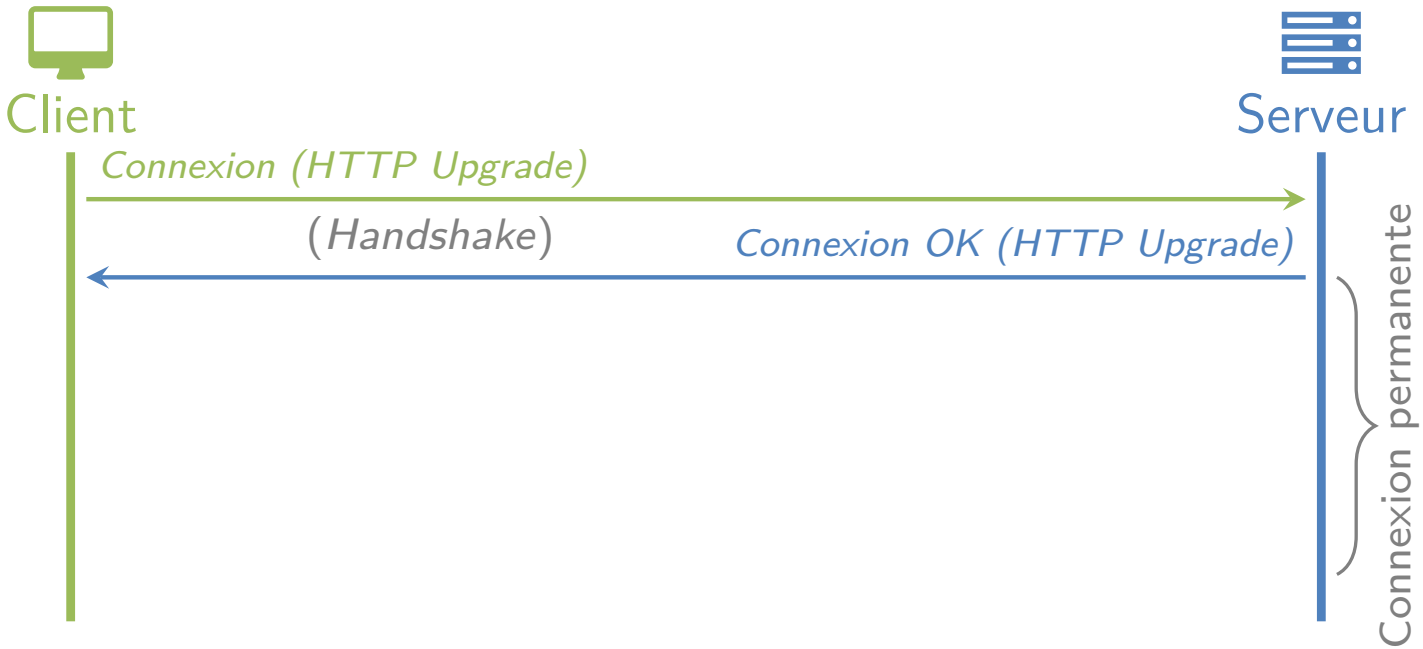

Schéma présentant les trois grandes parties du protocole WebSockets.

WebSocket

ISEN OUEST

Schéma de fonctionnement

Trois étapes :

- Connexion (*Handshake*).
- **Communications bidirectionnelles.**

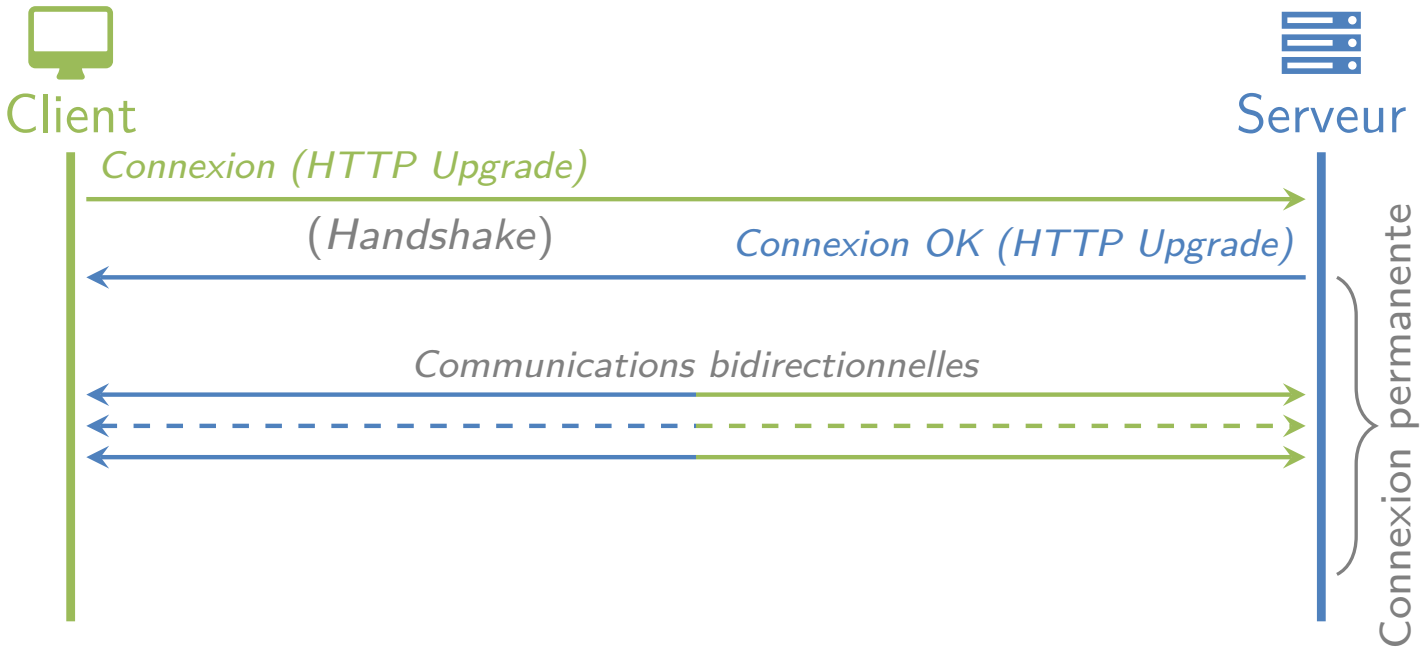

Schéma présentant les trois grandes parties du protocole WebSockets.

WebSocket

ISEN OUEST

Schéma de fonctionnement

Trois étapes :

- Connexion (*Handshake*).
- Communications bidirectionnelles.
- **Déconnexion.**

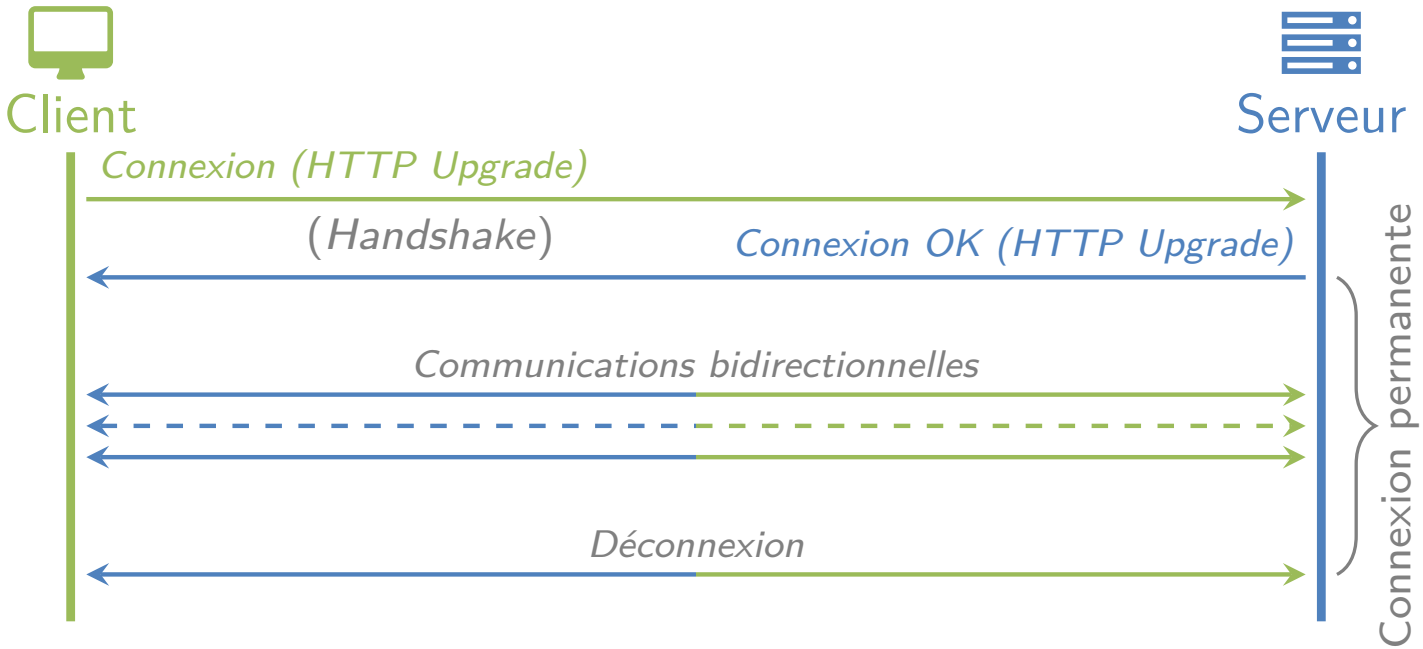

Schéma présentant les trois grandes parties du protocole WebSockets.

WebSocket

ISEN OUEST

Poignée de main (Handshake)

HTTP Upgrade :

Le protocole *WebSocket* est établi à partir du protocole *HTTP*. Pour passer de ce dernier à un protocole *WebSocket*, on utilise le *header* ***HTTP Upgrade*** avec la méthode *GET*. Le serveur fait de même.

Clé :

Afin de garantir l'unicité de la connexion, un échange de clés est fait entre le client C_{key} et le serveur S_{key} :

$$S_{key}=base64(sha1(C_{key}+258EAFA5-E914-47DA-95CA-C5AB0DC85B11))$$

Protocole :

Lors de l'établissement de la communication, il est nécessaire de définir le protocole d'échange de messages. On utilise le plus souvent *chat*.

Sécurité :

Pour éviter les attaques de type *Cross-Site Request Forgery* le serveur doit vérifier l'origine du client.

WebSocket

ISEN OUEST

Poignée de main (Handshake)

Côté client :

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
Origin: http://example.com
```

Côté serveur :

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

Sommaire I

ISEN OUEST

Implémentation côté client (*JavaScript*)

ISEN OUEST

L'objet *WebSocket*

Un constructeur :

- ***WebSocket(url)*** : permet d'établir une connexion de type *WebSocket* à partir d'une *url* :

```
ws://serveur.com:12345
```

Quatre événements :

- ***onopen*** : appelé lors de l'établissement d'une connexion.
- ***onmessage*** : appelé lors de la réception d'un message.
- ***onclose*** : appelé lors de la fermeture de la connexion.
- ***onerror*** : appelé lors d'une erreur sur la connexion.

Deux méthodes :

- ***void send(message)*** : permet d'envoyer un message.
- ***void close()*** : permet de fermer la connexion.

Implémentation côté client (*JavaScript*)

ISEN OUEST

Exemple

```
let websocket;
websocket = new WebSocket('ws://server.com:12345');

websocket.onopen = (event) => {
  console.log('Connexion établie.');
  websocket.send('Mon premier message.');
}

websocket.onmessage = (event) => {
  console.log('Message reçu : ' + event.data);
}

websocket.onclose = (event) => {
  console.log('Communication terminée.');
}
```

Sommaire I

ISEN OUEST

Implémentation côté serveur (*C++/Qt*)

ISEN OUEST

L'objet *QWebSocketServer*

Un constructeur :

- ***QWebSocketServer(url, secureMode)*** : permet d'établir une connexion de type *WebSocket* (*secureMode* permet d'utiliser le protocole sécurisé *wss*).

De nombreux événements dont :

- ***newConnection*** : appelé lorsqu'un client souhaite se connecter. *nextPendingConnection()* permet de récupérer la *WebSocket* associée au client.

De nombreuses méthodes dont :

- ***bool listen(address, port)*** : permet de se mettre en écoute des connexions clientes.
- ***void close()*** : permet de fermer le serveur.

Implémentation côté serveur (*C++/Qt*)

ISEN OUEST

L'objet *QWebSocket*

De nombreux événements dont :

- ***textMessageReceived*** : appelé lorsqu'un message est reçu du client.
- ***disconnected*** : appelé lorsque le client ferme la connexion.

De nombreuses méthodes dont :

- ***qint64 sendTextMessage(message)*** : permet d'envoyer un message au client.

Sommaire I

ISEN OUEST

Conclusion

AJAX vs *WebSocket*

ISEN OUEST

Communications standards :

Lorsque l'utilisateur effectue une action, celle-ci est exécutée et le navigateur attend le résultat. Cela occasionne parfois des délais et une attente pour l'utilisateur.

Communications *AJAX* :

Le mode asynchrone élimine cette attente. Les requêtes au serveur sont lancées sans que soit suspendue l'interaction avec le navigateur et la page est mise à jour lorsque les données requises sont disponibles.

Communications *WebSocket* :

Alors qu'en *AJAX* on opère par une succession de requêtes et réponses alternatives, le protocole *WebSocket* est bidirectionnel : une connexion statique s'établit entre le serveur et le client et les deux parties envoient des données à leur convenance.

Conclusion

ISEN OUEST

Avantages et inconvénients

Avantages :

- Communication bidirectionnelle.
- Simple à mettre en place.
- Nombreuses implémentations.

Inconvénients :

- Nécessite une implémentation du protocole *WebSocket*.
- Ne permet pas les communications pair-à-pair entre deux clients en *JavaScript*.
- Ouvre la porte à des failles de sécurité (*cache poisoning*).

ISEN | école d'ingénieurs
ALL IS DIGITAL!

Avez vous des questions ?