TP3 : La Plateforme WebRTC

Rapport Détaillé d'Implémentation et Analyse

Asmaa BAGRI
Youssef DADDA
Akram EL MAMOUN

21 novembre 2025

Table des matières

1 Introduction et Objectifs du TP

L'objectif de ce TP est de développer une application P2P en temps réel basée sur **WebRTC** et **HTML5** afin d'échanger des flux média et de données. Le défi principal est de gérer le franchissement des **NATs** et des **Pare-feux** via un **canal de signalisation** (**ICE/STUN/TURN**).

2 Partie 1 : Flux Média et Canal de Données

2.1 Capture des Flux Audio et Vidéo (API `MediaStream`)

La capture des flux est réalisée en utilisant l'API `MediaStream` (Question 1) et en ajoutant des contraintes de résolution (Question 2). Cette étape est validée par la demande d'autorisation du navigateur.

2.2 Échange des Données avec l'API `RTCDataChannel`

L'API `RTCDataChannel` permet d'établir un canal de données arbitraires entre les pairs. Le code de base simule cet échange pour valider le transfert bidirectionnel des messages. La comparaison avec `Websocket` souligne la nature faible latence et P2P du `RTCDataChannel`.

3 Partie 2 : Signalisation et Négociation

3.1 Signalisation avec NodeJS/Socket.io

Le serveur de signalisation `Socket.io` est mis en œuvre pour relayer les messages `SDP` (Offre/Réponse) et `ICE` (Candidats), assurant la logique de création/jointure des canaux.

3.2 Échange SDP et ICE

Le succès de la connexion P2P dépend de l'échange des descripteurs SDP et des candidats ICE. La négociation réussie de ces paramètres, gérée par le canal de signalisation, permet l'établissement du chemin de communication final entre les deux navigateurs.

4 Partie 3 : Extensions et Intégration Finale

4.1 Extension 1 : Intégration Complète du Flux Média

L'intégration complète lie la capture média, le `RTCDataChannel`, le flux audio/vidéo et la signalisation, menant à une session P2P fonctionnelle.

4.2 Extension 2 : Technologies de Signalisation Alternatives

Le service de signalisation a été implémenté avec des alternatives à `Socket.io` : **Websocket Natif** et **XHR (Long Polling)**.

4.3 Extension 3 : Implémentation Générique et Configurabilité

Une interface générique a été développée pour permettre à l'utilisateur de choisir la technologie de signalisation et le mode de données souhaités au moment de l'usage.

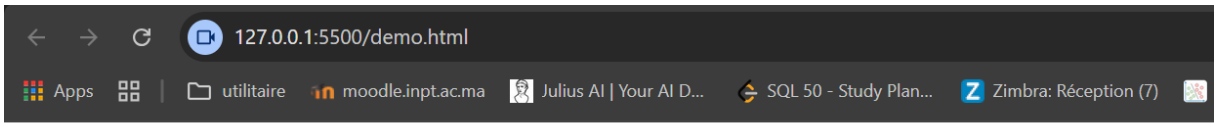
4.4 Extension 4 : Mode Vidéoconférence/Multipartites (Concept)

L'étude conceptuelle de l'Extension 4 conclut que l'architecture **SFU (Selective Forwarding Unit)** est la solution la plus viable pour la scalabilité des flux dans un mode vidéoconférence.

5 Conclusion

Ce TP a permis une exploration exhaustive de la pile WebRTC. Nous avons réussi l'intégration complète des flux média, du canal de données, et du service de signalisation. Les extensions ont validé la flexibilité de l'architecture et son potentiel d'évolution vers des topologies de vidéoconférence avancées.

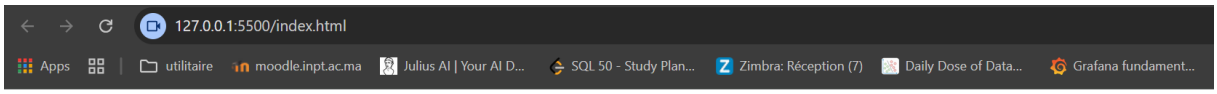
6 Annexes : Figures de l'Exécution



Exemple de `getUserMedia()`

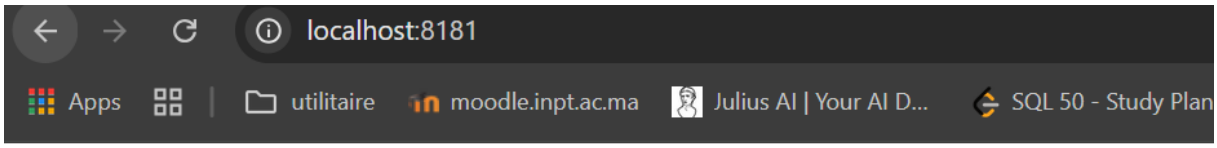
Il s'agit d'appeler `getUserMedia()` et d'afficher le resultat en utilisant la balise HTML5

Examiner les codes sources HTML et JavaScript ...



Exemple de `getUserMedia()` avec contraintes

Capture vidéo avec contraintes de résolution 1280x720



Time: 8.714 --> Channel YOUSSEF has been created!

Time: 8.715 --> This peer is the initiator...

Time: 29.632 --> Message from server: request to join channel YOUSSEF

Time: 29.633 --> Broadcast message from server:

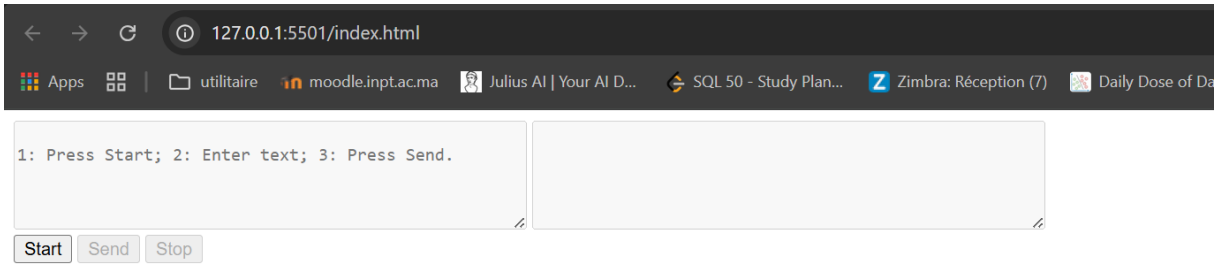
S --> broadcast(): client 382zdMtNCdoXZVTxAAAH joined channel YOUSSEF

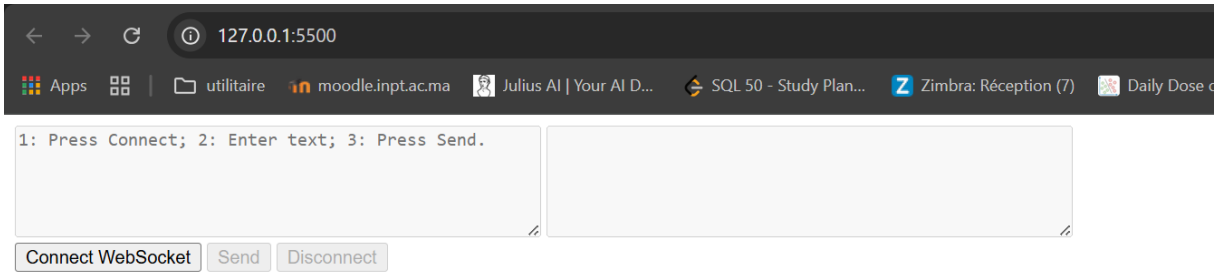
Time: 40.220 --> Got response from other peer:

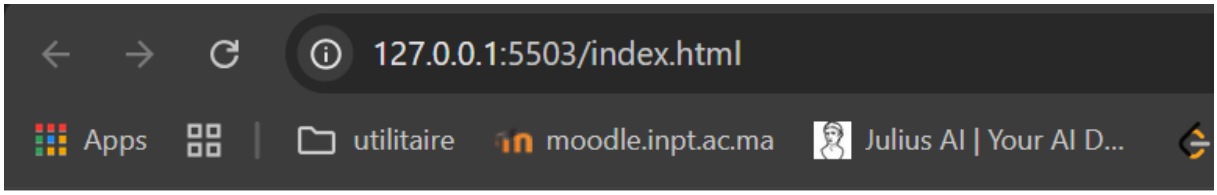
TEST SUCCEEDED

Time: 58.273 --> Got response from other peer:

RETRYING ANOTHER TEXT







WebRTC Signaling Client

Ce client utilise Socket.io pour la signalisation.


WebRTC Application Générique
Configuration de la Session
Signalisation : Socket.io
Canal de Données : RTCDataChannel (P2P)
Nom du Canal : default_channel
Démarrer & Rejoindre