INPT
المعهد الوطني للبريد والمواصلات
ⵜⴰⵙⴷⴰⵡⵜ ⵏⵜⵉⵎⵓⵔⴰⵙⵜ ⴰⵎⴰⵡⴰⵍⵏ ⴷ ⵜⵉⵎⵓⵔⴰⵙⵉⵏ
Institut National des Postes et Télécommunications

TP2 : La communication multimédia en temps réel sur le web : WEBRTC

SUD341: Systèmes temps réel et embarqués

Réalisé par :

EL MEKKAOUI Fayssal
HAJJAJI Ayyoub
ELAMMARI Souhail

Encadré par :

EN-NOUAARY Abdeslam

WebRTC

Filière SUD - CLOUD & IOT - INE2

Introduction:

Dans ce TP nous allons essayer de développer une application P2P (Peer-to-Peer) en se basant sur les technologies WEBRTC et HTML5 conformément aux discussions menées dans le cadre du cours. L'application doit permettre à deux utilisateurs, devant leurs navigateurs, de télécharger une page web et d'ouvrir une session P2P pour échanger des flux media (audio et vidéo) et de données (généralement de texte). Les flux media (audio et vidéo) sont capturés à partir des périphériques multimédia de chaque utilisateur. Le fluxde données, quant à lui, est saisi à travers le clavier moyennant un forum de discussion.

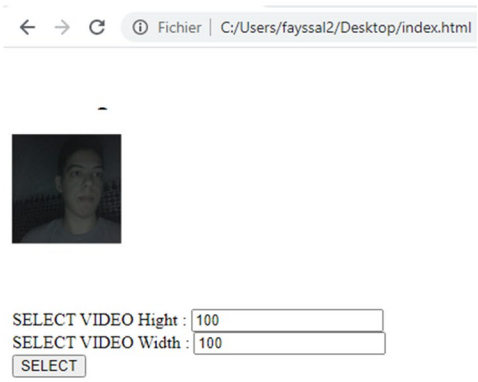
Capture des flux audio et vidéo du pair local :

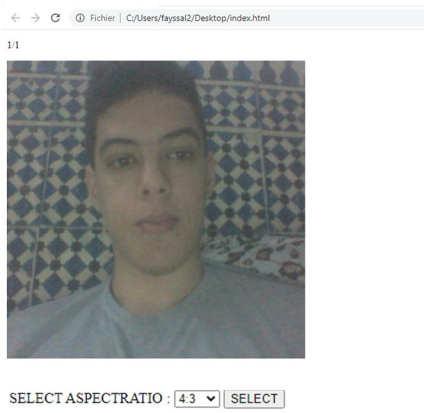
Pour capturer les flux audio et vidéo d'un utilisateur, on doit utiliser l'API MediaStream du WEBRTC. La visualisation des flux obtenus est faite en utilisant HTML5 et JavaScript.

Nous avons ajouté deux buttons (Unmute, Stop video), ils nous permettent de couper le son et d'arréter la capture de la vidéo.


```
const muteUnmute = () => {
    const enabled = myVideoStream.getAudioTracks()[0].enabled;
    if (enabled) {
      myVideoStream.getAudioTracks()[0].enabled = false;
      setUnmuteButton();
    } else {
      setMuteButton();
      myVideoStream.getAudioTracks()[0].enabled = true;
    }
  }

  const playStop = () => {
    console.log('object')
    let enabled = myVideoStream.getVideoTracks()[0].enabled;
    if (enabled) {
      myVideoStream.getVideoTracks()[0].enabled = false;
      setPlayVideo()
    } else {
      setStopVideo()
      myVideoStream.getVideoTracks()[0].enabled = true;
    }
```

Nous allons modifier le code source de l'application en ajoutant des contraintes de résolution sur les medias capturés.
Le client peut choisir la taiile de la video en donnant la hauteur et le largeur en pixel , ou bien choisir un Aspect-Ratio proposé (1:1, 3:4, 16:9).





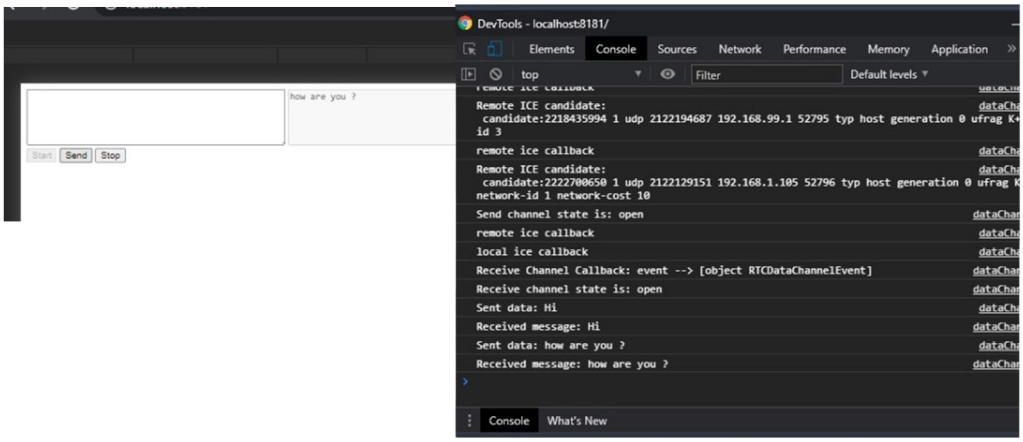
Voici la partie du code qui nous permet de changer les contraintes sur la resolution des médias capturés.

```
z=document.getElementById("in3").value;
document.getElementById("demo").innerHTML = z;

var video_con = {
  mandatory: {

    maxAspectRatio: z,
    },
  optional: []
};
navigator.mediaDevices.getUserMedia({ audio: true, video: video_con })
```

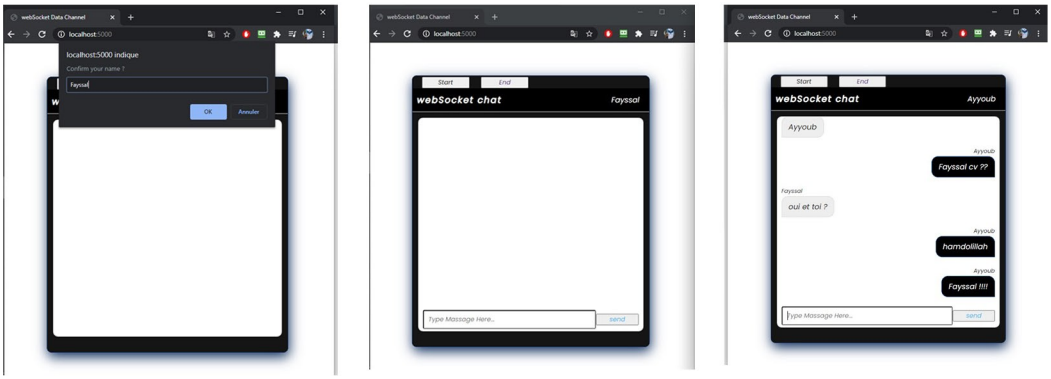
2. Échange des données entre les pairs en utilisant l'API RTC-DataChannel :

L'interface RTCDataChannel représente un canal réseau qui peut être utilisé pour les transferts bidirectionnels peer-to-peer de données arbitraires. Chaque canal de données est associé à un RTCPeerConnection, et chaque connexion homologue peut avoir jusqu'à un maximum théorique de 534 65 canaux de données (la limite réelle peut varier d'un navigateur à l'autre).

Il s'agit de trois buttons, le button start nous permet de commencer la connexion, le buttons Send permet d'envoyer les données enregistrée, et Stop arréte la connexion

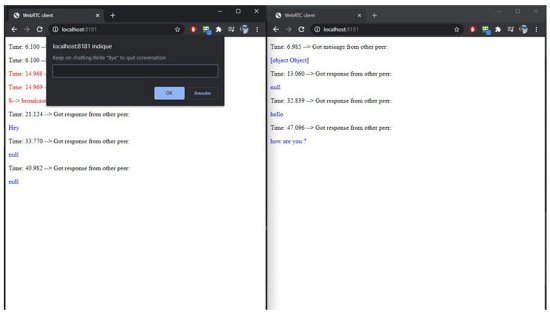


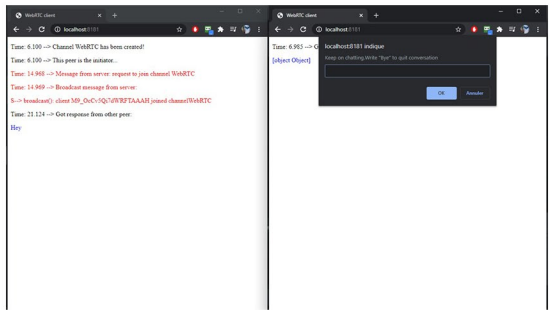
Par la suite nous allons modifier le code source de l'application pour remplacer l'API RTC-DataChannel par Websocket.



Quand le client clique sur start le navigateur vas lui demender de choisir un nom, puis il peut envoyer les donnés enregitrées et communiquer avec un autre client.

Établissement d'un canal de signalisation entre les deux pairs :
La signalisation est un processus qui permet de découvrir les pairs et de négocier les paramètres communs de session à établir entre eux . Dans ce qui suit, on implémente un canal de signalisation en utilisant NodeJS/Socket.io :

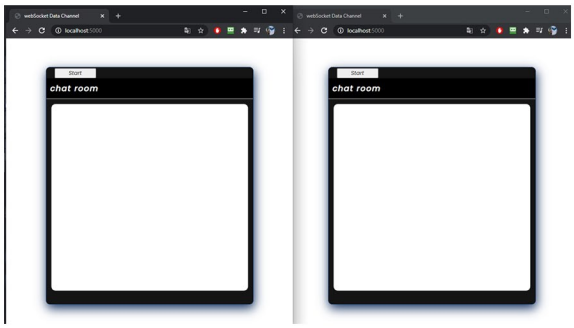


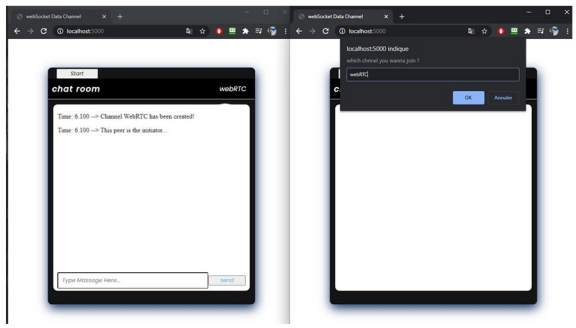


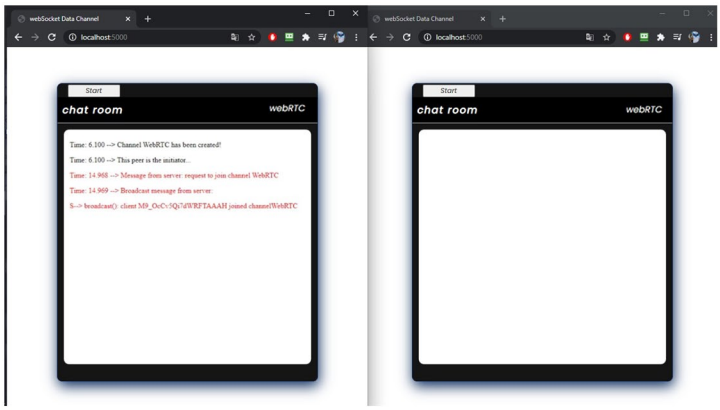
D'abord le client choisit le nom d'un canal, si le nom existe déja le client se dérige vers ce canal, sinon un nouveau canale sera créer.
Le canal sera bloqué si deux clients existent déja.

Le programme de cette application nous pose un petit problème, quand un client envoi un message il reçoit un autre message de valeur "NULL".

Essai des autres technologies pour fournir le service de signalisation aux deux pairs communicants.







Le fonctionnement de l'applicaion est amélioré de tel sort qu'on ne reçoit plus les messages "NULL".

Conclusion:

À travers ce TP nous étions capable de créer des applications RTC, qui nous permet de capturer des flux audio et vidéo à l'aide de l'API MediaStream du WEBRTC, et aussi d'échanger des données arbitraires (texte, binaire ...) entre deux pairs en utilisant L'API RTCDataChannel, et finalement d'établir un canal de signalisation en utilisant NodeJS/Socket.io.
On trouve ses mécanismes souvent dans le développement des applications réseaux sociaux comme Facebook, messanger .. ou bien des plateformes de visio-conférence comme BBB.