REPOBLIKAN'I MADAGASIKARA
FITIAVANA - TANINDRAZANA - FANDROSOANA

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE SUPÉRIEURE POLYTECHNIQUE D'ANTSIRANANA

Mention STIC

PROJET DE FIN DE SEMESTRE

Parcours Télécommunication et Réseau

Réalisation d'une application de communication en temps réel (WebRTC)

Par :

ANDRIANARISOA Daniel

Encadreurs :

Mme. RAOELIVOLOLONA Tefy

Mr. RAKOATOARIJAONA Raonirivo

⋆ Année Universitaire : 2019 - 2020 ⋆

REPOBLIKAN'I MADAGASIKARA
FITIAVANA - TANINDRAZANA - FANDROSOANA

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE SUPÉRIEURE POLYTECHNIQUE D'ANTSIRANANA

Mention STIC

PROJET DE FIN DE SEMESTRE

Parcours Télécommunication et Réseau

Réalisation d'une application de communication en temps réel (WebRTC)

Par :

ANDRIANARISOA Daniel

Encadreurs :

Mme. RAOELIVOLOLONA Tefy
Mr. RAKOATOARIJAONA Raonirivo

Membre de jury :

Pr./Dr./Mme/Mr. NOM Prénoms	Grade	Titre
Pr./Dr./Mme/Mr. NOM Prénoms,	Grade	Titre
Pr./Dr./Mme/Mr. NOM Prénoms,	Grade	Titre
Pr./Dr./Mme/Mr. NOM Prénoms,	Grade	Titre

⋆ Année Universitaire : 2019 - 2020 ⋆

Remerciements

Je tiens à remercier spécialement Madame RAOELIVOLOLONA Tefy et Monsieur RAKOTOARIJAONA Rivo pour leur encadrement, orientation, conseils et disponibilité qu'ils nous ont témoignés pour me permettre de mener à bien ce projet.

Je souhaite aussi à exprimer mes remerciements aux membres du jury de soutenance, pour leur précieux jugement de ce projet. Qu'ils trouvent ici l'expression d'une reconnaissance profonde et sincère.

Sans oublier de remercier mes collègues étudiants de L'ESP Antsiranana et les gens de mon entourage, pour l'environnement serrein qui'ls ont apportés, propice à la completion de ce projet.

Un grand merci à toutes les personnes qui ont contribué au succès de ce projet et qui m'ont aidée lors de la rédaction de ce rapport.

Table des matières

Liste des figures

Liste des acronymes et symboles

Acronymes

API Application Programming Interface

CSS Cascading Style Sheet

HTML HyperText Markup Language

HTTP HyperText Transfert Protocol

ICE Interactive Connectivity Establishment

IETF Internet Engineering Task Force

IP Internet Protocol

RTC Real Time Communication

SDP Session Description Protocol

SIP Session Initiation Protocol

UDP User Datagram Protocol

W3C World Wide Web Consortium

WebRTC Web Real-Time Communications

Chapitre 1

Introduction au WebRTC

Sommaire

1.1 Introduction

La conception structurelle web classique est basée sur un modèle client-serveur, dans lequel les navigateurs envoient une demande de contenu HTTP au serveur web, qui répond en leur envoyant une réponse contenant les informations demandées.

Le décodage de la réponse par les navigateurs est possible car elles contiennent les composants pour se conformer aux normes du W3C, ici en l'occurrence, de balises HTML, styles CSS et code Javascript.

De même, en ce qui concerne le modèle de navigateurs, afin de supporter WebRTC, chaque navigateur doit ajouter un autre composant pour pouvoir la prendre en charge. La plupart des principaux navigateurs ont ajouté le composant RTC depuis 2014.

À partir du diagramme 1.1, nous pouvons clairement voir le composant WebRTC intégré dans le navigateur. Il se charge d'appeler les API audio et vidéo natives du système d'exploitation.

Javascript joue un rôle important dans WebRTC, car la technologie s'exécute sur le navigateur. Javascript est le choix idéal en tant que langage de programmation pour assurer la médiation entre le WebRTC et l'application via un ensemble d'API exposées.

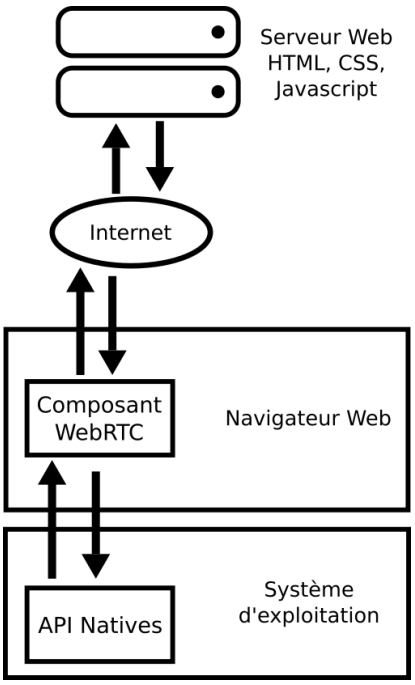

FIGURE 1.1 – composant WebRTC

1.2 Normes WebRTC

Les normes WebRTC sont actuellement développées conjointement par le W3C et l'IETF. Le W3C travaille sur la définition des API nécessaires aux applications web Javascript pour interagir avec la fonction RTC du navigateur. L'IETF développe les protocoles utilisés par la fonction RTC du navigateur pour communiquer avec un autre navigateur ou point de terminaison de communications Internet.

Le travail du W3C est centré sur le groupe de travail WebRTC et le travail de l'IETF est centré sur le groupe de travail RTCWeb (Real-Time Communications Web). Les deux groupes sont indépendants, mais se coordonnent étroitement.

1.3 Architecture WebRTC

Le navigateur cache la grande partie de l'implémentation WebRTC, tout le travail de capture des médias, de la description de session et des transmissions réseau, tout est abstrait de la surface et n'expose que de simples API.

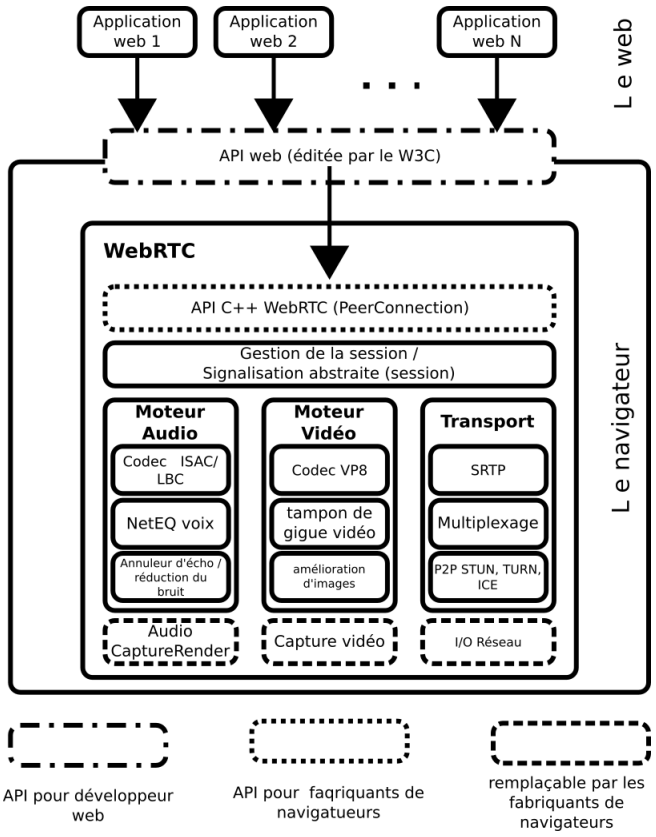

FIGURE 1.2 – architecture WebRTC

L'application ne peut appeler que l'API web qui est normalisée par le W3C pour les développeurs web à consommer.

1.3.1 Le triangle WebRTC

Le scénario le plus répandu est probablement celui où les deux navigateurs exécutent la même application web WebRTC, téléchargée à partir de la même page web. Cela produit le «triangle» WebRTC illustré à la figure 1.3. Cet arrangement est appelé triangle en raison de la forme de la signalisation (côtés du triangle) et des flux médiatiques ou de données (base du triangle) entre les trois éléments. Une connexion peer établit le transport pour les médias vocaux et vidéo et les flux de canaux de données directement entre les navigateurs.

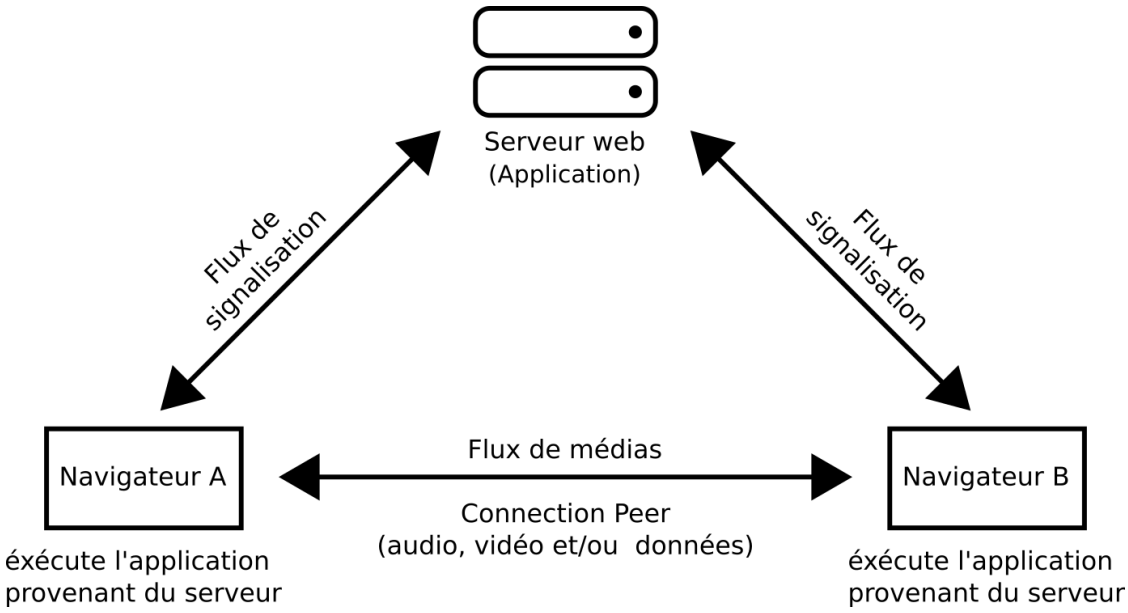

FIGURE 1.3 – triangle WebRTC

1.4 Composants Importtants du WebRTC

1.4.1 Flux de médias

Les appareils d'aujourd'hui sont dotés de plusieurs capacités matérielles multimédias. Dans les appareils mobiles, vous obtenez une caméra avant, une caméra arrière et un microphone audio. Sur les ordinateurs de bureau et les tablettes, vous obtenez le même ensemble de sources multimédias. Chacune de ces sources multimédia peut générer un flux multimédia.

Ce flux multimédia peut être capturé par l'API du navigateur WebRTC et l'API fournira ce flux en tant que variable au langage Javascript et que les développeurs d'applications pourront afficher sur une page comme n'importe quel flux vidéo normal.

1.4.2 Peer to peer

L'un des avantages distinctifs du WebRTC est le fait qu'il s'agit d'une communication peer to peer. Cela signifie évidemment que la vidéo passera directement de l'ordinateur d'un utilisateur à un autre.

1.4.3 Description de la session

En général, pour qu'une communication ait lieu entre deux périphériques quelconques, il existe un mécanisme de prise de contact qui doit être effectué en premier, pour établir la session entre les deux périphériques, puis le flux des messages ou des paquets peut démarrer.

C'est la même chose avec WebRTC. Il existe un protocole de déscription de session : SDP. Il s'agit d'un ensemble de messages qui doivent être échangés entre les deux navigateurs avant qu'ils ne commencent à envoyer des flux multimédias.

Pour échanger les données de description de session, nous avons besoin d'une machine intermédiaire. C'est là que le concept de signalisation entre en jeu.

1.4.4 Signalisation

Supposons que le navigateur A essaie de communiquer avec le navigateur B. Le navigateur A enverra ses données de description de session au serveur de signalisation S, et le même processus pour le navigateur B également.

Une fois ce processus terminé, chaque navigateur a maintenant à sa disposition tout le nécessaire pour établir une connexion et envoyer le flux multimédia à l'autre navigateur. La figure 1.3 aidera à expliquer ce concept.

1.4.4.1 Rôle de la signalisation

La signalisation a un rôle important dans les communications en temps réel :

— Négociation des capacités et des paramètres des médias ntre les appareils dans le même appel

— Identification et authentification des participants à une session

— Contrôler la session multimédia, indiquer la progression, modifier et terminer la session

— Résolution de l'éblouissement, lorsque les deux côtés d'une session essaient d'établir ou de modifier une session en même temps

Pourquoi la signalisation n'a pas été standardisé ?

La signalisation n'est pas normalisée dans WebRTC pour permettre l'interopérabilité entre différents navigateurs. Son flux se situe entre les navigateurs web et le serveur web, et non entre deux navigateurs dans le même appel ou session. Dans ce cas, le serveur ou plutôt le developpeur sélectionne le protocole de signalisation et s'assure que les utilisateurs des applications web ou du support du site utilisent le même protocole.

1.4.4.2 Signalisation et négotiation des médias

La fonction la plus importante de la signalisation est l'échange d'informations contenues dans l'objet de protocole de description de session (SDP) entre les navigateurs. SDP contient toutes les informations nécessaires au fonctionnement du RTC, y compris les types de codecs multimédias (audio, vidéo et données) utilisés et des informations sur la bande passante de la connexion.

Un autre rôle de la signalisation est l'ICE. Il s'agit de l'adresse candidate représentant l'adresse IP et les ports UDP où des paquets multimédias potentiels pourraient être reçus par le navigateur. Mais il ne peut pas être démarré tant que l'adresse candidate n'a pas été échangée sur le canal de signalisation.

Identification et authentification

Le canal de signalisation peut également fournir l'identité des participants à un appel. Par exemple, dans une application web, l'application se chargera d'abord d'authentifier les utilisateurs, et lorsqu'un utilisateur doit appeler un autre utilisateur, l'application web présentera le nom d'utilisateur pour le canal de signalisation comme un identifiant authentifié.

WebRTC a d'autres moyens d'authentifier les utilisateurs sans s'appuyer sur des applications externes. En utilisant le canal multimédia, qui ne repose pas sur des applications externes de

confiance, une identité et une authentification de l'appelant peuvent être fournies dans le chemin multimédia sans dépendre du tout du canal de signalisation.

Contrôle de la session de médias

La signalisation est nécessaire pour lancer l'appel ; cependant, il n'est pas nécessaire d'indiquer l'état ou de mettre fin à une session. Il y a une machine d'état ICE dans le navigateur qui peut fournir ces informations.

1.4.4.3 Résolution de l'éclat

La résolution de l'éblouissement intervient lorsque les deux côtés de la communication essaient de changer de session en même temps. C'est une condition de concurrence qui pourrait entraîner un état non déterminé pour la session. Le protocole de signalisation SIP intègre une résolution d'éblouissement.

1.5 Les API WebRTC

WebRTC a trois fonctionnalités principales :

1. Acquérir l'audio et la vidéo
2. Communication audio et vidéo
3. Communication de données arbitraires

Et il existe trois API Javascript majeures exposées pour appeler ces foncionnalités :

1. Flux multimédia (alias getUserMedia)
2. Connexion Peer RTC (RTCPeerConnection)

3. Canal de données RTC (RTCDataChannel)

1.5.1 Flux multimédia

Nous désignons les caméras et les microphones des ordinateurs des utilisateurs sous le nom de média local. Par conséquent, la première étape consiste à obtenir un flux multimédia local. Cela peut être fait de plusieurs manières. Voici un moyen simple de le faire, juste une méthode Javascript que nous pouvons appeler `getUserMedia()`. Cette méthode récupère les médias des périphériques de la machine.

Pour les problèmes de confidentialité, le navigateur doit obtenir l'autorisation de l'utilisateur pour accéder à la caméra et au microphone de l'utilisateur. Le navigateur demandera à l'utilisateur d'autoriser l'accès. Une fois l'accès accordé, l'application recevra un flux de la caméra et du microphone.

1.5.2 Connexion peer RTC

Il est important de noter que cette connexion est une connexion peer. Cela signifie que cela peut être fait avec n'importe quel ordinateur ou appareil sur Internet. Et la connexion est directement établie entre les deux machines sans serveur impliqué.

Les paquets vidéo/audio circuleront d'un navigateur à un autre. Pour établir une connexion peer, il suffit d'appeler cette méthode : `RTCPeerConnection()`.

Un fait très important qui mérite d'être mentionné est le nombre de connexions qui seront créées pour chaque utilisateur ajouté à une conférence téléphonique.

Disons, par exemple, un appel démarré avec deux utilisateurs, tels que l'utilisateur A et l'utilisateur B. Cela signifie qu'une connexion peer sera créée entre les navigateurs des deux utilisateurs, une connexion de A à B.

Si un autre utilisateur rejoint l'appel, disons l'utilisateur C, cela modifiera les connexions pour chaque utilisateur de l'appel. Comme nous l'avons expliqué précédemment, il s'agira d'une connexion peer, ce qui signifie que chaque utilisateur doit être connecté à la machine de chaque utilisateur

lors de l'appel.

Pour comprendre combien de connexions peer seront créées par machine :

$$\begin{aligned} &\mathrm{C}onnexions\,par\,machine = nombre\,d'utilisateurs\,en\,appel - 1 \\ &\mathrm{C}onnexions\,par\,machine = 3 - 1 \end{aligned}$$

Chaque machine aura 2 connexions peer établies si l'appel a 3 utilisateurs.

1.5.3 Canal de données RTC

Les canaux de données prennent en charge les connexions à haut volume et à faible latence ; un canal de données est un canal non multimédia qui prend uniquement en charge le transfert de données.

Il contourne les serveurs et fournit aux développeurs web des canaux configurables pour transférer des données. Les canaux de données dans WebRTC sont construits sur des WebSockets. De cette façon, il obtient une fonctionnalité en temps réel. Des méthodes simples telles que l'envoi sur le gestionnaire de messages peuvent être définies pour obtenir les données. Le canal de données utilise également la même connexion peer que le média, ce qui est une bonne chose, car cela signifie qu'un seul processus de négociation offre/réponse est nécessaire.

Les canaux de données prennent également en charge les deux modes de livraison : livraison garantie (fiable) et rapide (non fiable). Le premier peut être utilisé pour des événements critiques et le second peut être utilisé pour les mises à jour de la position d'un jeu en ligne.

Chapitre 2

Configuration des Interfaces WebRTC

Sommaire

2.1 Introduction

Chapitre 3

Relevé des protocoles mis en jeux dans WebRTC

Sommaire

3.1 Introduction

Références

[1] H.W. BARZ et G.A. BASSETT : *Multimedia Networks : Protocols, Design and Applications*. Wiley, 1 édition, 2016.

[2] A. HAYTHAM : Real-time Communication Using WebRTC. *Special Interest Group on Computer-Human Interaction*, 2018.

[3] A.B. JOHNSTON et D.C. BURNETT : *WebRTC : APIs and RTCWEB Protocols of the HTML5 Real-Time Web*. Smashwords Edition, 2 édition, 2013.

[4] K.G. PATHARE et P.M. CHOURAGADE : WebRTC Implementation and Architecture Analysis. *International Journal of Scientific & Engineering Research*, 7, 2016.

Annexes

A Exemple test de code Javascript dans LaTeX

A.1 Premier snippet

```
(function(){
  var user = getCookie("username");
  document.getElementById("#date-field").innerHTML = new Date();
  document.getElementById("#greetings").innerHTML = "<p>Hello, " + user.name +
    ".</p>";
})()

function getCookie(cname) {
  var name = cname + "=";
  var decodedCookie = decodeURIComponent(document.cookie);
  var ca = decodedCookie.split(';');
  for(var i = 0; i < ca.length; ++i) {
    var c = ca[i];
    while (c.charAt(0) == ' ') {
      c = c.substring(1);
    }
    if (c.indexOf(name) == 0) {
      return c.substring(name.length, c.length);
    }
  }
  return "";
}
```

A.2 Second snippet

```
/* eslint-env es6 */
/* eslint-disable no-unused-vars */
import Axios from 'axios'
import { BASE_URL } from './utils/api'
import { getAPIToken } from './utils/helpers'
```

```
export default class User {
  constructor () {
    this.id = null
    this.username = null
    this.email = ''
    this.isActive = false
    this.lastLogin = ''  // ISO 8601 formatted timestamp.
    this.lastPWChange = ''  // ISO 8601 formatted timestamp.
  }
}

const getUserProfile = async (id) => {
  let user = new User()
  await Axios.get(
    `${BASE_URL}/users/${id}`,
    {
      headers: {
        'Authorization': `Token ${getAPIToken()}`,
      }
    }
  ).then{response => {
    // ...
  }).catch(error => {
    // ...
  }) }
```