AYADI Marwen, BEN ABDELJELIL Mohamed
MAGROUN Hamza, RASAMIZANANY Kune

TER

Outil de vision conférence / chat basé web HTML5/WebRTC/NodeJS

Encadré par : Monsieur Michel BUFFA

AYADI Marwen, BEN ABDELJELIL Mohamed
MAGROUN Hamza, RASAMIZANANY Kune

Sommaire

Introduction

Ce document présente le TER dont le groupe est composé de AYADI Marwen, BEN ABDELJELIL Mohamed, MAGROUN Hamza et RASAMIZANANY Kune. Il a été réalisé sous l'encadrement de Monsieur Michel Buffa.

Le but de ce projet est de développer une application complète de type Adobe Connect, incluant chat et vidéo, mais aussi de la géolocalisation HTML5, pour faire le support en ligne du cours HTML5 que Monsieur Michel Buffa donne pour le W3C.

Les technologies utilisées sont le JavaScript, JQuery, NodeJS car il faut un serveur pour initier les connexions et gérer les sessions, HTML5 et CSS3, tandis que pour la partie chat, nous utilisons la librairie socket.io pour NodeJS ainsi que les WebSockets de HTML5.

1. WebRTC

1.1 Présentation

Le webRTC qui est l'acronyme de "Web Real-Time Communications" ou encore "communication web en temps réel" en français, est une nouvelle technologie basée sur le standard HTML5 qui permet d'échanger des communications audio, vidéo et data directement au travers des navigateurs Internet. Sans installation d'aucune sorte, sans plug-in à télécharger, sans serveurs d'infrastructure propriétaires… et le tout en mode Peer-to-Peer.

Aujourd'hui ce n'est pas le cas pour de nombreux services de chat ou de vidéo, certains d'entre-eux qui semblent être intégrés au navigateur comme ceux de Google, de Facebook, de Skype ou de Microsoft requièrent en réalité des plugins dédiés et propriétaires. Il est par exemple impossible de faire du chat vidéo entre Skype et Google, car il n'y a aucune interopérabilité entre les carnets d'adresses, préférences et états de communication, limitation des outils pour

transmettre tel ou tel flux ou communiquer à plusieurs sur la même session de communication… bref, le web d'aujourd'hui est encore très perfectible concernant la communication en générale.

1.2 Historique

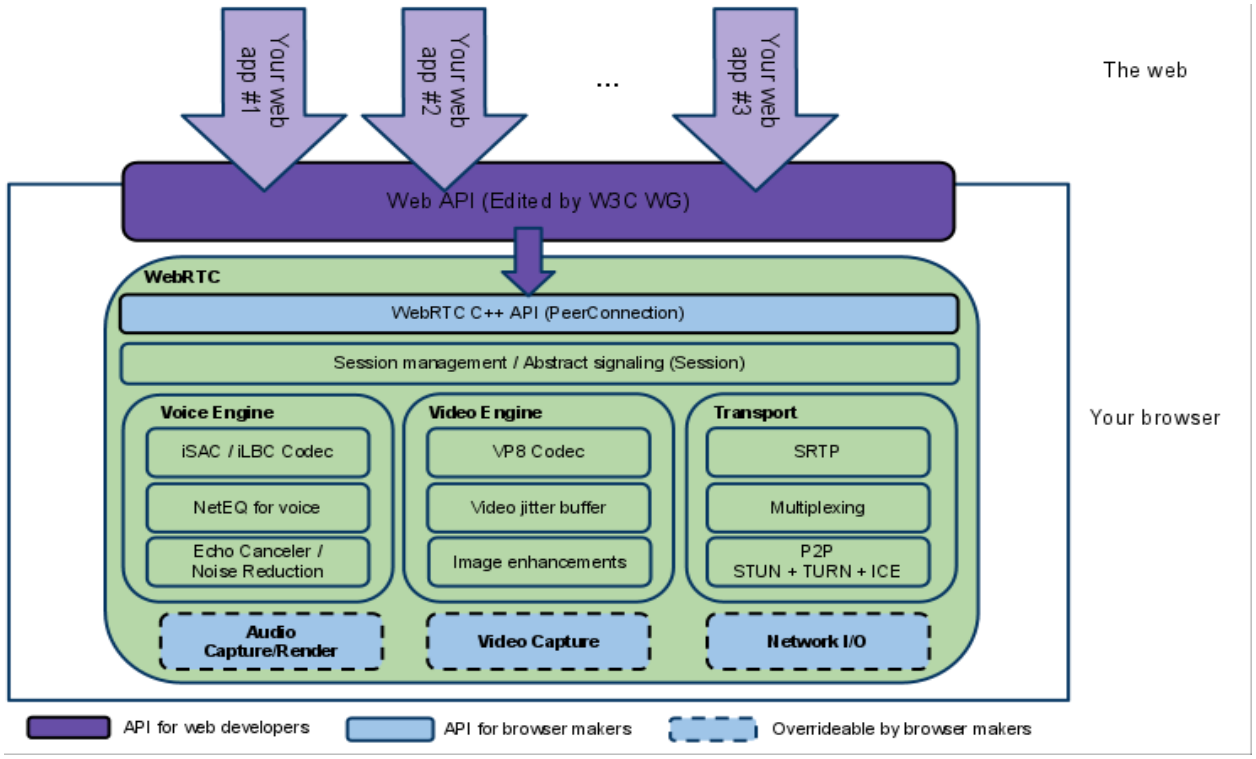
La communication en temps réel entre navigateurs existait déjà et ce n'est pas une nouveauté qui a été introduite par le W3C et l'IETF, mais cela avait besoin de logiciel ou de plugin propriétaire comme par exemple Adobe flash ou encore Microsoft ActiveX.
Ces plugins sont à l'origine de nombreux problèmes, de bugs, de difficultés d'interopérabilité et de mise à jour pour les applications utilisant ces extensions.
Ces extensions exigent aussi des licences et l'intégration de technologies à la fois coûteuses et complexes.

1.3 Les avantages et les objectifs

Les avantages du WebRTC sont tout d'abord la communication en peer-to-peer, c'est à dire que les flux audio, vidéo et de données seront envoyés et reçus directement entre les utilisateurs rien ne sera conservé sur le serveur et les données seront cryptées par le protocole de sécurisation SSL, ainsi les communications sont coordonnées par l'intermédiaire d'un canal de signalisation.
Le WebRTC est disponible sur plusieurs plateformes : Chrome , FireFox , Opéra , Chrome pour Android, elle est également facile à manipuler : pas besoins de plugin ou de quelconque application comme par exemple Flash, Ces API doivent être libres , normalisées , intégrées dans les navigateurs Web et efficaces. Cette nouvelle technologie offre une haute qualité audio et vidéo, c'est à dire que les fonctionnalités du navigateur seront utilisées afin de compresser les sons pour faciliter et améliorer l'écoute ainsi que la conversation entre les différents utilisateurs.

Par conséquent on peut en conclure que son principal objectif est d'offrir aux utilisateurs des options de communication plus simple, moins coûteux et de manière instantanée.

1.4 L'architecture



Comme le montre l'architecture sur la figure ci-dessus, on peut constater que WebRTC se charge du transport(SRTP) et formatage (codecs audio , vidéo) des paquets média et n'implémente pas la gestion de la signalisation .

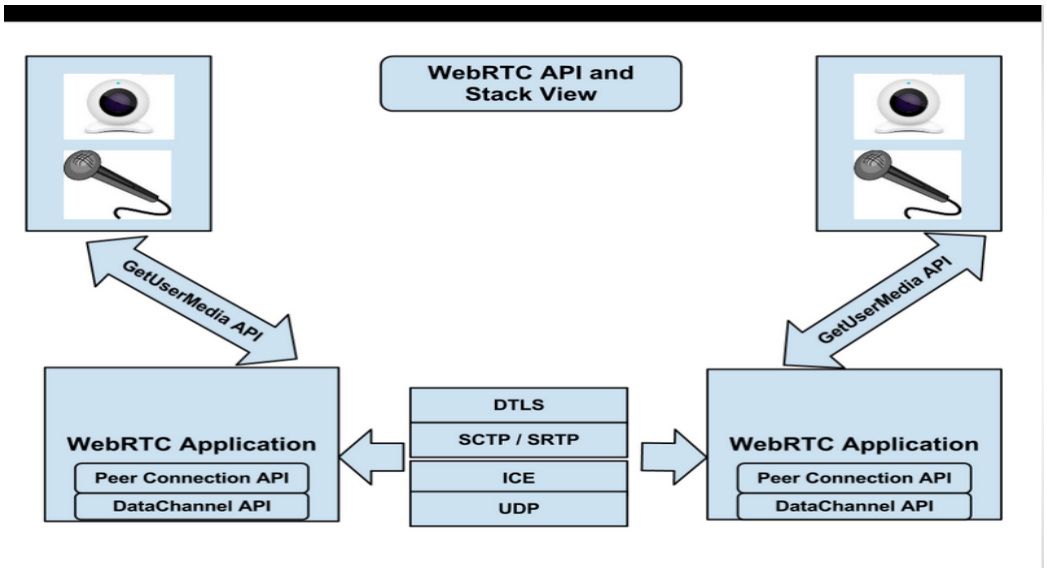
On peut distinguer 2 couches distinctes :

- La couche WebRTC C++ API : c'est une couche API qui permet aux développeurs de

navigateurs d'implémenter facilement l'API Web.

- La couche Web API : c'est une couche API faite pour être utilisée par les développeurs pour développer des applications Web comme les vidéo-conférences.

2. Les API de WebRTC



Comme on peut le voir dans le schéma ci-dessus, l'API WebRTC fonctionne grâce aux standards comme STUN, ICE, TURN, DTLS ou encore SRTP:

L'objet **PeerConnection** utilise les protocoles UDP, STUN et ICE dans le but d'assurer la continuité de la connexion en cas de translation d'adresse par NAT et éviter le blocage par les parefeux (notamment en entreprise).

-Le protocole **UDP** est utilisé par l'API peerConnection dans le but de représenter le lien établi avec un navigateur distant.

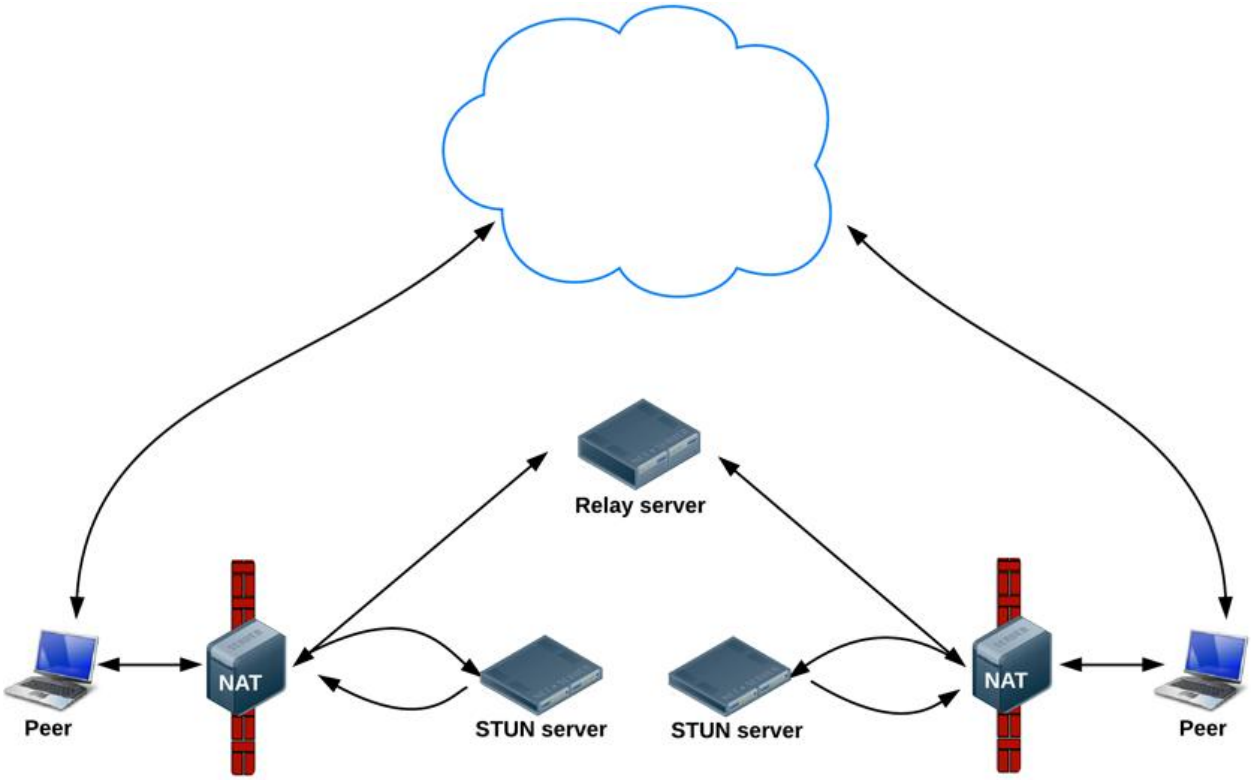
-**ICE** est l'acronyme Anglais de Interactive Connectivity Establishment, c'est un protocole qui utilise des paquets de tests afin de de pouvoir déterminer les règles de filtrages du pare-feu, mais aussi pour traverser un NAT.

Il fait référence aux procédés utilisés pour trouver des interfaces Web et des ports (trouver des candidats).

Ice essaie de connecter les pairs directement, avec la plus faible latence, via UDP. Dans ce processus,

Les serveurs STUN ont une seule tâche: permettre le peer derrière le NAT pour trouver l'adresse publique et le port.

Si UDP échoue, ICE essaie TCP. Si la connexion directe échoue, en particulier à cause du parcours du NAT et des pare-feu, ICE utilise un relai intermédiaire(TURN server). En d'autres mots, ICE utilisera en premier STUN avec UDP pour connecter directement les pairs, si ça échoue, il utilisera comme relai le serveur TURN.



2.1 L'API MediaStream

Le travail de l'API MediaStream est de demander à l'utilisateur la permission d'accéder à la caméra et au microphone, puis de créer une vidéo synchronisée et un flux audio. Il le fait avec une méthode JavaScript appelée getUserMedia ().

Le code de base pour créer un flux et l'affichage de la vidéo en JavaScript est la suivante :

```
var constraints = {video: true};

function successCallback(stream) {
  var video = document.querySelector("video");
  video.src = window.URL.createObjectURL(stream);
}

function errorCallback(error) {
  console.log("navigator.getUserMedia error: ", error);
}

navigator.getUserMedia(constraints, successCallback, errorCallback);
```

La fonction getUserMedia déclenchée par un bouton, autorise l'accès aux différents équipements (webcam , microphone).

En cas de succès la fonction successCallback sera appelée afin de construire effectivement le flux media, sinon en cas de refus c'est la fonction errorCallback qui sera appelée en affichant un message d'erreur.

2.2 L'API RTCPeerConnection

RTCPeerConnection est le composant du WebRTC qui gère la communication stable et efficace des flux de données entre pairs, en revanche il offre les API nécessaires pour l'établissement de la session audio/vidéo .

Une fois le flux multimédia créé, WebRTC utilise l'API RTCPeerConnection pour communiquer les données en continu entre pairs, ce dernier emploie une interface très simple.

Afin d'établir cette connexion pair à pair, il est nécessaire de s'appuyer sur un canal de communication établi par un serveur Web et utilisant par exemple un objet WebSocket.

Pour obtenir une connexion, l'un des pairs doit obtenir les informations locales.

RTCPeerConnection doit gérer les tâches suivantes:

- Traitement du signal (y compris l'annulation d'écho et réduction du bruit)
- Communication peer-to-peer
- Chiffrement
- Gestion de la bande passante

2.3 L'API RTCDataChannel

L'API RTCDataChannel offre la possibilité d'échange peer-to-peer des données telles que des images ou du texte entre les pairs. La communication se fait directement entre les navigateurs, donc RTCDataChannel peut être beaucoup plus rapide que WebSocket .

2.4 L'API RTCRecord

Il s'agit d'une technologie de WebRTC qui permet d'enregistrer de l'audio au format ogg/wav et de la vidéo au format webm .
Il est à noter que si on veut enregistrer de la vidéo, le navigateur compatible est Firefox Nightly.

3. Le signaling

Le signaling est le mécanisme par lequel les pairs envoient des messages de contrôle à chacun dans la but d'établir le protocole de communication, le canal et la méthode. Ceux-ci ne sont pas spécifiés dans les standard WebRTC. En fait, le développeur peut choisir n'importe quel protocole de message(comme SIP ou XMPP), et n'importe quel canal de communication duplex (comme

WebSocket ou XMLHttpRequest) en tandem avec une API de connexion persistante à un serveur pour AppEngine.

Il est utilisé pour échanger 3 types d'information :

- Les messages de contrôle de session qui permettent d'initialiser et de fermer la communication ainsi que de reporter les erreurs
- Configuration: L'adresse IP et le port de la machine
- Les informations sur le média : Les codecs et les résolutions supportées par le navigateur local et le navigateur avec lequel il veut communiquer

L'échange d'information via signaling doit être complet avant que le streaming peer-to-peer ne commence.

4. Tâches réalisées

4.1 Comment ça marche

La première étape de ce TER fut d'observer le code source des sites proposant la conversation en webRTC, l'étude des outils, framework ainsi que des applications utilisant le WebRTC, tout cela dans le but de mieux comprendre le fonctionnement de la communication en temps réel.

4.1.1 Outils/framework utiles au développement

simpleWebRTC

Cette librairie alimente le site Talky.io (gratuit, communication peer-to-peer, possible de se connecter à plusieurs, partage d'écran (chrome seulement), session privée, fonctionne

uniquement sur firefox et chrome).

L'avantage est qu'il est composé de plusieurs petits modules indépendants, au cas où il faudrait ne pas tout sélectionner comme par exemple:

-Signalmaster : node.js/socket .io alimente le signaling du serveur.

-webrtc.js: rend Compatible la communication webrtc entre les navigateurs, gestion des connexions en peer to peer.

-RTCPeerConnection: permet la communication en peer to peer.

-getUserMedia : permet l'accès aux utilisateurs de webcam, micro.

-attachMediaStream : permet l'ajout des flux vidéo et réserve les fonctionnalités de coupure de son et miroiter votre écran dans une simple API.

-webrtcsupport : vérifie la capacité des navigateurs, extrait les constructeurs nécessaires au fonctionnement comme par exemple PeerConnection, sessionDescription et IceCandidate.

-getScreenMedia : extension de getUserMedia mais pour l'accés aux écrans lors du partage de l'écran.

-mediastream-gain : contrôleur d'entrée volume audio simple

-Hark : Permet de savoir l'identité de la personne qui utilise l'API du web audio

-WildEmitter : version minimale de EventEmitter

RTC

Ensemble de modules open source Node.js qui peuvent être installés avec le npm(node packaged modules) dans le but de pouvoir : accéder à la webcam et au microphone de

l'utilisateur créer des appels audio et vidéo entre les navigateurs, mettre en place des canaux de données entre les navigateurs, ou encore installer un serveur de signalisation.

EasyRTC

EasyRTC est un outil WebRTC open source qui s'adresse principalement aux entreprises pour des communications sécurisées.

Il s'agit plus précisément d'un ensemble d'applications web, des extraits de code, des bibliothèques de composants client et serveur.
EasyRTC a la particularité de fournir une API côté client relativement simple afin que les developpeurs web puissent intégrer ses codes sans être des spécialistes du WebRTC.

CrocodileRTC

CrocodileRTC est un framework servant à la communication temps réel développé en utilisant les technologies WebRTC, SIP ou encore XMPP.
Elle possède une API de haut niveau, ce qui permet donc de pouvoir l'intégrer aux applications web de manière relativement simple.
CrocodileRTC a une multitude de fonctionnalités comme par exemple des séances de communication audio et vidéo, des transferts de fichiers, la gestion des carnets d'adresses, ou encore un service de messagerie.

PeerJS

PeerJS permet de simplifier la communication audio, vidéo et de données en peer-to-peer.
Il enveloppe la mise en œuvre WebRTC du navigateur pour fournir une API complète,

configurable, et facile à utiliser pour la connexion en peer-to-peer.
Les données communiquées entre deux pairs n'affectent pas d'autres serveurs, donc pas de temps de latence d'un serveur intermédiaire, par contre la vitesse de connexion est limitée par le taux d'upload et de download des deux pairs. PeerJS est compatible avec Firefox 23+ et Chrome 26+ seulement.

4.1.2 Présentation des applications utilisant le webRTC

Bistri

Bistri permet d'avoir une conversation audio-visuelle directement dans le navigateur, sans télécharger quoique ce soit, sans installer de plugin, en un simplement un clic. Nul besoin d'avoir un compte d'utilisateur si l'utilisateur rejoint un salon déjà créé mais en revanche l'administrateur du salon lui doit posséder un compte, je pense que c'est l'un des inconvénients qui ralentissent la première utilisation et même les suivantes (juste le temps de se connecter avec le login et password). L'avantage donc c'est que tout se passe dans le navigateur, c'est multi-utilisateurs (4 maximum). Il y a la possibilité d'effectuer des screenshots et des effets sur la vidéo.
Bistri fonctionne sur plusieurs plateformes: PC, Mac, tablette (sous Chrome Android).
Il est utilisé par 1,5 million d'utilisateurs, il est traduit dans 21 langues et il est très populaire.
L'autre inconvénient est que le tchat n'est compatible que de Chrome vers Chrome ou de Firefox vers Firefox. 4 personnes max.

Opentokrtc

Opentokrtc intègre aussi le tchat et la vidéo, l'inconvénient c'est que lors de l'ouverture du salon, des pseudos aléatoires sont attribués aux participants, il faut taper une commande pour s'attribuer un pseudo: /nick "pseudo" il y a aussi d'autres commandes disponibles comme pour voir la liste des participants ou encore voir la liste des commandes disponibles (/help).

Il peut y avoir jusqu'à 20 participants en même temps.

Opentokrtc fonctionne sur:

- iPhone 4S / 5
- iPad 2 / 3 / 4 / mini
- Smartphone Android

L'application Opentok 2.0 pour iphone supporte les connexions en WiFi.

Elle offre la possibilité de travailler sur des projets, c'est à dire sauvegarder les conversations écrites et vidéos. L'inconvénient majeur c'est que ce n'est pas une application gratuite, on a droit a 30 jours d'essai pour avoir les fonctionnalités cités dernièrement.

apprtc.appspot.com/

Ce site est la démonstration officielle du site www.webrtc.org/demo, il n'y a pas de tchat. Il fonctionne avec les navigateurs Chrome et Firefox. Il n'y a pas d'informations sur le nombre d'utilisateurs maximum sur la même session.

Appear.in

Appear.in utilise la communication peer-to-peer. Cela signifie que tous les flux vidéo et audio sont envoyés directement entre les utilisateurs. Rien n'est conservé sur les serveurs et les communications sont cryptées par SSL.

Ici aussi aucune installation, aucun téléchargement n'est nécessaire.

Cette application fonctionne sur les récentes versions des navigateurs Opera, Chrome, Firefox, il peut y avoir 8 utilisateurs sur la même session.

L'utilisation est simple, il y a l'audio/vidéo et le tchat inclus. Lors de l'envoi d'un texte un screenshot est placé à coté du texte pour savoir qui a envoyé le message et si la vidéo est désactivée alors les messages envoyés sont identifiés par un carré coloré (une couleur différente

pour chaque utilisateur).

Vline

Même principe que pour les autres démos de webrtc, pas d'installation requise, pas de plugin. Vline est encore en développement notamment pour les plateformes iOS et Android.
La qualité de la vidéo est très bonne, meilleure que les autres démo (en tout lors de mon test). En revanche le tchat n'est pas (encore ?) intégré. Il y a possibilité d'activer les notifications sur bureau pour savoir si quelqu'un a rejoint le salon.
Le son n'est pas encore au point sur Firefox, c'est pour cela que les développeurs recommandent l'utilisation du navigateur Chrome pour le moment.

4.1.3 Tableau des fonctionnalités pour chaque application

	Appear.in	**Bistri**	**Vline**	**Apprtc**	**Opentokrtc**
Video/Audio	Oui	Oui	Oui	Oui	Oui
Chat	Oui	Oui	Non	Non	Oui
Personne max	8	4	4		20
iOS	Non	Oui	Bientôt	Non	Oui

Android	Chrome Android	Chrome Android	Bientôt	Chrome Android	Chrome Android
Navigateur	Opera,Chrome & Firefox	Chrome, Firefox	Chrome, Firefox	Chrome, Firefox	Chrome, Firefox
Gratuit	Oui	Oui	Oui	Oui	Oui ou 30 jours (pro)

4.2 Les fonctionnalités réalisées

On indique étape par étape le codage des fonctionnalités :

- Le tchat

- L'utilisation des périphériques audio et vidéo

- Une solution à l'enregistrement des conférences (audio et vidéo)

- La géolocalisation multi-participants

- La Visio conférence à deux

- La Visio conférence à plusieurs

5. Les Problèmes rencontrés et les solutions trouvées

Mis à part le fait que tous les membres du groupe n'étaient pas très fort dans le langage JavaScript, l'un des points majeurs qui nous a causé le plus de difficulté était la mise en œuvre de la Visio conférence à plusieurs. Nos recherches nous ont conduits à les mettre en relation en utilisant un maillage.

En effet, pour mettre en relation trois utilisateurs par exemple, on doit séparer les connexions: u1<-> u2, u1<->u3 et u2<->u3.

En utilisant cette solution, on a été confronté à un problème d'échelle. En effet, pour mettre en relation N utilisateurs, on aurait besoin de N+ (N-1) + (N-2) +.....+1 canaux séparés, ce qui s'avère difficilement réalisable à grande échelle. Pour résoudre ce problème, le spécialiste qui nous aidé dans nos recherches nous a orienté vers l'utilisation d'un serveur, une solution que nous n'avons pas pu explorer et qui selon lui est compliqué à mettre en œuvre.

6. Références

http://fr.wikipedia.org/wiki/WebRTC

http://www.html5rocks.com/en/tutorials/webrtc/basics/

http://www.webrtc.org/