Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Enginyeria Informàtica

Títol: Estudi i disseny d'un sistema de comunicació directa amb WebRTC .

Document: Resum

Alumne: Andreu Mañosa Crous

Tutor: Jose Luís Marzo Lázaro

Departament: Arquitectura i tecnologia de computadors

Àrea: Arquitectura i tecnologia de computadors

Convocatòria (mes/any): Setembre 2016

INTRODUCCIÓ

L'objectiu del treball és dissenyar i implementar un sistema que permeti la comunicació entre usuaris a través de diferents medis (text, àudio i vídeo).

El sistema forma una xarxa entre tots els usuaris que i pertanyen, per això gestiona el registre i autentificació pels usuaris, la seva llista de contactes, les converses entre usuaris autentificats i permet establir videoconferències amb vídeo i/o àudio entre ells.

Quan un usuari es registra a l'aplicació se li crea una identitat amb la que és identificat respecte els demés i té la possibilitat de contactar amb els altres usuaris que formen la xarxa, podent-los-hi enviar peticions per a obrir converses de dos o més usuaris. Quan es reuneixen els usuaris i es crea una conversa aquesta permet que es puguin enviar missatges de text, fitxers i imatges entre ells, a més de la possibilitat d'establir multi conferències en temps real.

La gestió de videoconferències s'ha dissenyat i implementat fent ús de la tecnologia ***WebRTC***. A partir de la API que WebRTC ofereix per a *JavaScript* s'ha construït a sobre, una API personal (ASWRTC) que ofereix la possibilitat de realitzar multi conferències directes amb control de les velocitats de transmissió (VDT) i fluxos multimèdia intercanviats sense la necessitat de conèixer la tecnologia de WebRTC.

En la implantació del sistema primer s'ha construït la API ASWRT que ofereix les següents funcionalitats:

- Comunicar a tots els usuaris que accedeixen a la mateixa videoconferència de manera directa (P2P) a través dels medis d'àudio, vídeo i text.
- Gestió dels fluxos multimèdia locals (engegar/parar l'àudio i vídeo amb els que participem a la videoconferència. També permet la gravació dels fluxos multimèdia locals que enviem als demés usuaris).
- Gestió dels fluxos multimèdia remots (permet gravar, aturar i reprendre l'àudio i vídeo provinent dels altres usuaris que participen en la videoconferència).
- Intercanvi de missatges de text, imatges i fitxers a través d'un xat de manera directe.
- Ajust de les VDT a les restriccions marcades per qui usa la API (dona al programador el control de les VDT usades per cada usuari perquè no se'ls sobrecarregui la xarxa).
- Obtenció i oferiment d'estadístiques sobre l'estabilitat de les videoconferències.
- Reducció de les VDT en temps real en el cas de que la videoconferència presenti inestabilitats en l'intercanvi de fluxos multimèdia.

A partir del conjunt de mètodes que proporciona la interfície de ASWRTC s'ha construir la part *front-end* amb el suport de l'entorn de treball *AngularJS*.

L'aplicació consta de tres pàgines web, una per el registre i autentificació d'usuaris, una pàgina pel perfil de cada usuari identificat (aquesta és la pàgina que s'ofereix als usuaris per que gestionin la seva identitat i converses) i una per a la realització de videoconferències.

Raons que justificaven el desenvolupament

El projecte va sortir amb l'idea de donar un millor suport a la part de videoconferències directes de VITAM, un dels projectes desenvolupats al BCDS. VITAM (Videoconference Teleassistance and Monitoring) és una altre aplicació únicament de multi conferències Web basada en WebRTC i construïda sobre la plataforma Licode.

L'ús de *Licode* a nivell de servidor soluciona el problema d'escalabilitat que presenta la realització de videoconferències en P2P (per cada usuari de més que participa en una multi conferència és necessari doblar els recursos de xarxa que l'usuari necessita per enviar i rebre tot els fluxos de dades en temps real als demés usuaris). Per solucionar el problema d'escalabilitat *Licode* permet que els recursos de xarxa que un usuari necessita per a realitzar una videoconferència siguin sempre els mateixos independentment del nombre d'usuaris que hi ha a la conversa.

Malgrat ser l'ús de la plataforma *Licode* una molt bona solució a nivell de clients, l'alternativa que ofereix trasllada els problemes d'escalabilitat del client al servidor, fent necessari que s'hagin d'augmentar les capacitats de hardware i xarxa del servidor depenent del nombre total d'usuaris que poden estar realitzant videoconferències en un moment determinat.

L'alternativa que el projecte ofereix és establir canals directes (P2P) entre navegadors regulant les velocitats de transmissió que s'usa per cada canal. L'augment i disminució de VDT sobre els fluxos multimèdia intercanviats permet regular la quantitat d'informació que s'envia i es rep per a cada medi de cada usuari de la multi conferència.

D'aquesta manera, amb una correcte gestió dels recursos usats es poden realitzar multi conferències de varis usuaris sense la necessitat de preocupar-se d'exhaurir la capacitat de xarxa que es necessita.

De tota aquesta part relacionada amb l'intercanvi de dades a través de WebRTC se n'encarrega la API ASWRTC, que pot ser reutilitzada per a qualsevol aplicació web que vulgui incorporar la possibilitat de comunicar els seus usuaris a través de videoconferències.

Tot el sistema desenvolupat en el projecte podria ser un exemple d'una aplicació que comunica els seus usuaris a través de videoconferències implementades sobre la API ASWRTC.

REQUERIMENTS DEL SISTEMA

Requisits funcionals

Requisits pel registre i identificació

- Registrar un nou usuari a l'aplicació.
- Identificar un usuari i redirigir-lo a la seva pàgina personalitzada.

Requisits per a la gestió del perfil i contactes

- Modificació de la identitat (perfil) de l'usuari.
- Llistar tots els usuaris que formen part de l'aplicació.
- Enviar una petició d'amistat (conversa individual) a un usuari seleccionat.
- Obtenir el llistat de peticions de converses enviades i rebudes.
- Respondre a peticions de conversa rebudes
    - Acceptar petició de conversa
    - Rebutjar petició de conversa
- Llistar la llista d'amistats d'un usuari.

- Enviar peticions de converses múltiples a partir de la llista d'amistats.
- Llistar les converses en les que participa l'actor.
- Seleccionar una conversa i mostrar-la en primer pla.
- Enviar esdeveniments entre converses
    - Enviar trucada
    - Penjar trucada
    - Enviar resposta trucada
    - Enviar imatge
    - Enviar missatge de text
    - Enviar fitxers
- Mostrar les videoconferències que un usuari té obertes.
- Trobar les coordenades (Latitud,Longitud) en les que es troba l'usuari quan accedeix a la plana.
- Realitzar el test de la VDT de pujada i baixada de la que disposa l'usuari.
- Analitzar l'estat de totes les amistats de l'usuari i mostrar si estan o no en línia.

Requisits per a la gestió de videoconferències

- Gestionar fluxos multimèdia de l'usuari local.
    - Parar/engegar el vídeo recollit per la càmera de l'usuari.
    - Parar/engegar l'àudio recollit pel micròfon de l'usuari.
    - Iniciar/finalitzar gravació dels fluxos multimèdia locals actius.
    - Iniciar recuperació de vídeo de la càmera.
    - Iniciar recuperació d'àudio del micròfon.
- Gestionar fluxos multimèdia d'usuaris remots
    - Parar/engegar el vídeo rebut
    - Parar/engegar l'àudio rebut
    - Iniciar/finalitzar gravació dels fluxos multimèdia rebuts de l'usuari remot.
- Gestionar xat
    - Enviar/rebre missatge de text no persistent
    - Enviar/rebre imatge no persistent
    - Enviar/rebre fitxer no persistent
- Gestionar VDT
    - Detectar els canals *WebRTC* inestables
    - Modificar VDT en temps real a través del protocol SDP

Requisits no funcionals

- És una plataforma Web.
- Programat en *JavaScript*, *HTML* i *CSS*.
- Interfície gràfica de fàcil comprensió.
- Servidor amb base de dades.
- El servidor ha de respondre peticions HTTP.
- El servidor ha de permetre executar PHP.
- Canals bidireccionals entre client i servidor.
- Permetre la comunicació entre usuaris d'usuaris des de qualsevol tipus de xarxa.
- Capacitat de processar gran quantitat de peticions.

- Suportar una gran quantitat de connexions TCP.
- Guardar les dades referents als usuaris de manera encriptada.
- Enviar les dades xifrades a través de la xarxa pel compliment de la llei LOPD.
- Es dona suport per Chrome i Firefox.
- En Firefox no es dona suport en el control de les velocitats de transmissió encara.

TECNOLOGIES USADES I DECISIONS PRESES

Part client

Per a la realització de videoconferències s'han utilitzat les API's que ofereix *WebRTC* per a l'intercanvi de fluxos multimèdia en temps real i la llibreria de *Socket.IO* (basada en la tecnologia de *WebSockets*) per a establir un canal bidireccional entre client i servidor per tal de donar suport a la restricció que ofereix WebRTC de tenir un servidor de senyalització per poder intercanviar les descripcions de sessió entre usuaris abans d'establir el canal directe entre ells.

Socket.IO també s'usa en la part de gestió d'usuaris (en la plana del perfil de cada usuari) per tal de comunicar en temps reals als usuaris afectats pels esdeveniments que un usuari genera.

Per la implementació de la part *front-end*, tant en la pàgina de videoconferències com en les demés pàgines s'ha usat el llenguatge d'etiquetes HTML i CSS per a construir les interfícies, i per implementar la lògica del sistema s'ha usat JavaScript juntament amb l'entorn de treball AngularJS i les llibreries JQueri i JQuery-min.

Part servidora

El servidor del sistema SCS (Signaling Controller Server) consta de:

- Una base de dades relacional gestionada amb MySQL.
- Un servidor Apache ubicat al port 80 i un servidor Apache segur ubicat al 443.
- Un servidor NodeJS ubicat al port 8080. Aquet servidor dona servei a peticions HTTPS (per la API REST) i permet l'establiment de canals bidireccionals entre ell i clients gràcies a la llibreria Socket.IO.

Per a l'ús de *WebRTC* entre usuaris que estan en xarxes privades és necessari un altre servidor que consti de:

- Un servidor que permeti l'intercanvi de missatges a través del protocol STUN. Aquest servidor permet conèixer @IP i ports públics d'usuaris ocultats per routers amb NAT.
- Un servidor que permeti l'intercanvi de missatges a través del protocol TURN. Aquest servidor de reenviament permet fer passar canals WebRTC a través d'ell per tal de donar solució a usuaris que no poden establir comunicacions directes entre els seus navegadors.

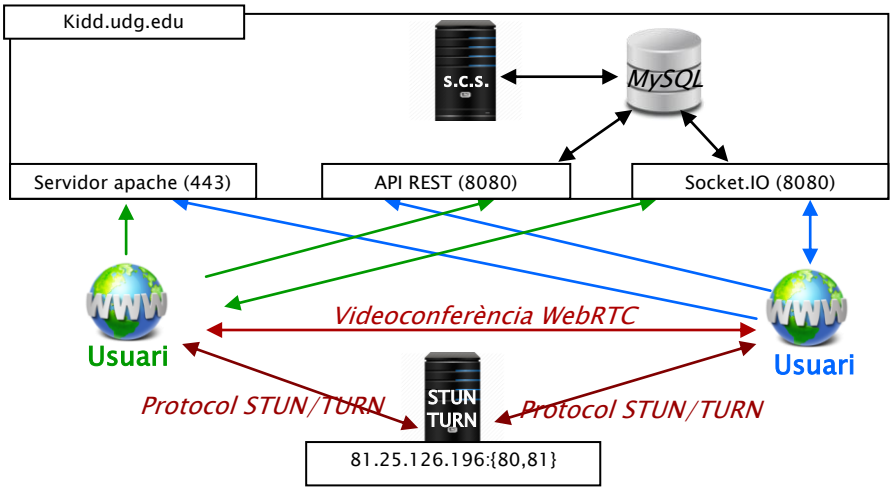
Comunicació client servidor

Per tal de comunicar client i servidor el servidor ofereix una API REST al port 8080 que permet al client interactuar amb la base de dades. Degut a que la API REST segueix una arquitectura *client - servidor* i no permet notificar les dades modificades als altres usuaris afectats en temps real, en el port 8080 també s'ofereix una API en SOCKET.IO que, al igual que la API REST, té accés a la base de dades, però amés permet que els usuaris afectats per un canvi siguin notificats d'aquest en temps real.

Des de la part client es llancen peticions a la API REST a través del mòdul *http* que AngularJS ofereix. I per cridar els mètodes SOCKET.IO del servidor en la part client s'ha dissenyat un conjunt de classes dedicades a aquesta funcionalitat; aquest conjunt de classes s'agrupen amb el nom de API ASSIO i obren un *socket* persistent per a que el servidor pugui notificar en temps real els esdeveniments que van passant i que afecten al client.

ARQUITECTURA DEL SISTEMA

L'arquitectura del sistema és la següent:



En l'arquitectura és mostra com els usuaris poden interaccionar amb els diferents ports que el sistema els i disposa. La part client, un cop l'usuari accedeix al sistema pot interactuar amb el servidor de quatre maneres:

- Ús de peticions HTTPS a Apache per obtenir la pàgina a mostrar a l'usuari.
- Realització de peticions a la API REST del servidor per modificar o consultar la BD.
- Connexió *socket* entre client i el servidor del *namespace* usuari. Aquest canal que s'obre al iniciar la cessió i es tanca al acabar permet tenir les dades del client sempre actualitzades.
- Connexió *socket* entre client i el servidor del *namespace* grup per a cada videoconferència oberta. El canal bidireccional permet realitzar el protocol de senyalització de WebRTC.

CONCLUSIONS I TREBALL FUTUR

El resultats final del treball ha estat un sistema que permet comunicar usuaris des de qualsevol punt de la xarxa d'Internet a través dels medis d'àudio, vídeo i text. Pel intercanvi d'àudio i vídeo s'ofereix un sistema de videoconferències amb canals de comunicació directes entre usuaris de manera segura i adaptable a les qualitats de xarxa dels usuaris (si s'usa navegador *Chrome*) gràcies a la utilització de la tecnologia WebRTC. D'aquesta manera el projecte s'adhereix a l'abast i objectius inicials amb els que es va proposar.

En un futur s'espera:

- Restringir les VDT en *Firefox* a través de la disminució de la qualitat dels fluxos recuperats.
- Construir una interfície que mostri gràficament les estadístiques obtingudes en les conferències.
- Expandir el sistema als dispositius mòbils.