PRÀCTIQUES VIDEOCONFERÈNCIA AMB WEBRTC – P3

Objectius

- Heu de construir una aplicació de videoconferència entre dos, basada en el navegador, fent servir WebRTC, però de moment amb vosaltres mateixos
  - us enviareu el vídeo a vosaltres mateixos, amb un sol navegador
  - en un requadre ("*local video*") es mostrarà el vídeo local i en l'altre requadre ("*remote video*") es mostrarà el mateix vídeo local

WebRTC (i)

- What is WebRTC? *
  - "WebRTC is an open framework for the web that enables Real Time Communications in the browser"
  - "It includes the fundamental building blocks for high-quality communications on the web, such as network, audio and video components used in voice and video chat applications"
  - "These components, when implemented in a browser, can be accessed through a JavaScript API (Application Programming Interface), enabling developers to easily implement their own RTC web application"
  - "The WebRTC effort is being standardized on an API level at the W3C and at the protocol level at the IETF"

* https://webrtc.org/faq/#what-is-webrtc

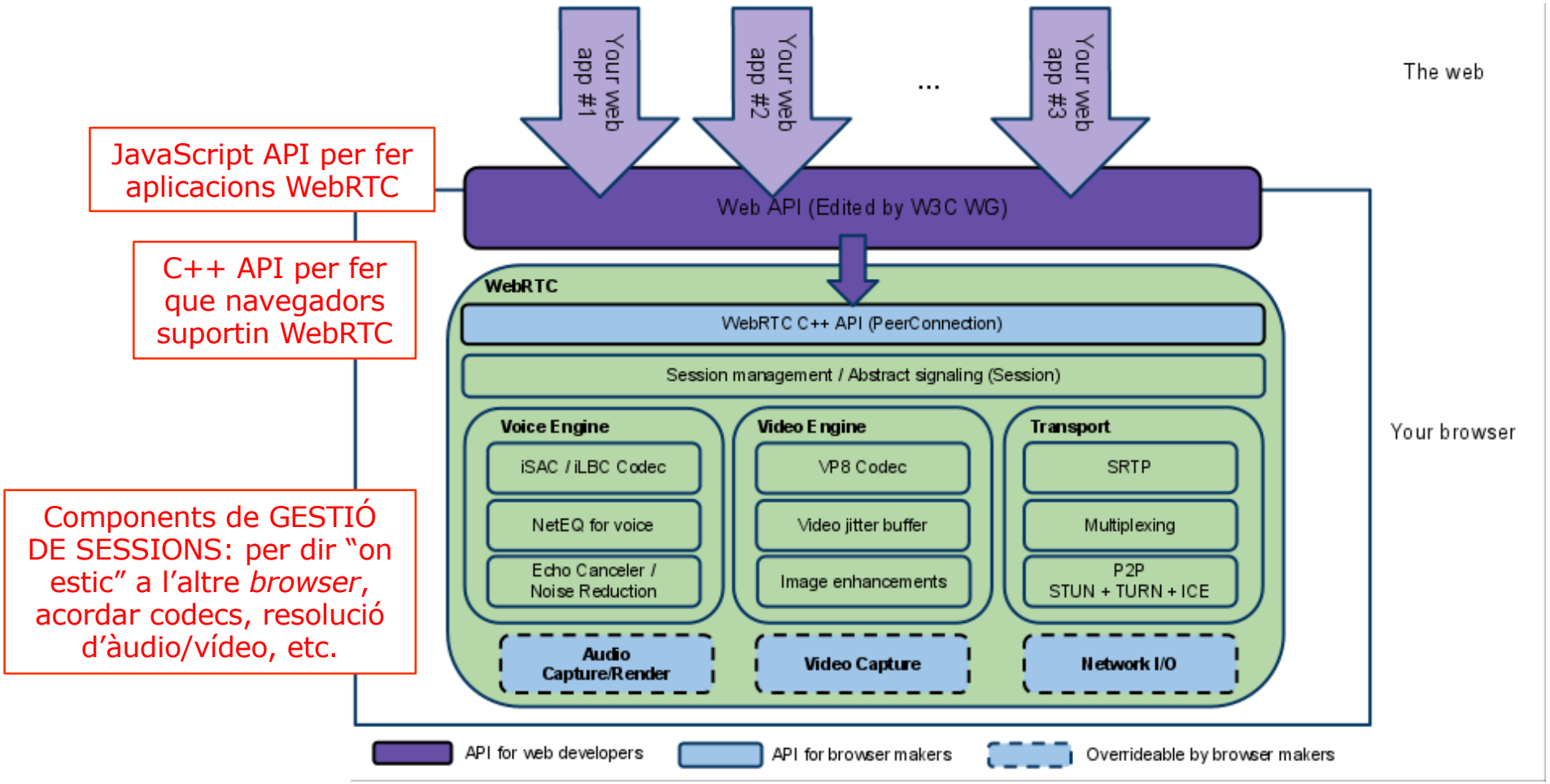
WebRTC (ii)

- WebRTC a la Wikipedia *
  - "WebRTC (Web Real-Time Communication) is a standard that defines a collection of communications protocols and APIs (Application Programming Interfaces) that enable real-time communication over peer-to-peer connections"
  - "This allows web browsers to not only request resources from backend servers, but also real-time information from browsers of other users"
  - "This enables applications like video conferencing, file transfer, chat, or desktop sharing without the need of either internal or external plugins"
  - "WebRTC was standardized by the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF). The reference implementation is released as free software under the terms of a BSD license"

* https://en.wikipedia.org/wiki/WebRTC

WebRTC (iii)

L'arquitectura de WebRTC (extret de https://webrtc.org/architecture/)



JavaScript API per fer aplicacions WebRTC

C++ API per fer que navegadors suportin WebRTC

Components de GESTIÓ DE SESSIONS: per dir "on estic" a l'altre *browser*, acordar codecs, resolució d'àudio/vídeo, etc.

Components d'ÀUDIO i VÍDEO: accés i control al HW, codecs, cancel·lació d'eco, reducció soroll, reproducció, etc.

Components de XARXA: protocols de transport, *buffering*, *rate adaptation*, control d'error, *NAT traversal techniques*, etc.

WebRTC (iv)

- WebRTC disposa de diverses APIs en JS
  - *getUserMedia* (o *MediaStream*), que permet al navegador accedir p.e., a la càmera i al micròfon, i obtenir-ne fluxos (*streams*)
  - *RTCPeerConnection*, que permet establir trucades d'àudio i vídeo, enviar els fluxos, etc., entre navegadors
  - *RTCDataChannel*, que permet enviar dades entre navegadors
  - *getStats*, que permet obtenir estadístiques sobre l'enviament dels fluxos (p.e., paquets perduts, retard, etc.)

- WebRTC no està implementat en tots els navegadors, ni tampoc exactament igual
  - farem servir *Firefox* i/o *Chrome*, i si mòbil, serà Android

- WebRTC requereix un servidor *web* segur
  - haurem de veure com fer-lo amb Apache o Node.js

Servidors *web* segurs

- Per fer un servidor *web* segur caldria disposar d'un certificat SSL
  - emès per un centre de confiança o autoritat certificadora (CA)

- Opcions que tenim
  - "Pagar" a una CA per un certificat, fer els tràmits, etc., car i lent; caldria fer-ho per un sistema en funcionament ("en producció")
  - *Let's Encrypt* és una CA que emet certificats gratuïts, que donen confidencialitat i integritat però una molt pobre autenticació
  - … "jo" podria fer de CA, generar-vos els certificats, i que vosaltres m'afegíssiu a la llista de CAs del vostre navegador
  - certificats autosignats, que permeten encriptar però no són de confiança (navegador ho dirà); adequats per fer desenvolupament

- Utilitzarem *self-signed certificates*
  - la meitat ho fareu amb Apache2 i l'altra meitat amb Node.js, així veiem diferents maneres de fer-ho

Què heu de fer?

- Cal instal·lar un servidor *web* segur, amb Apache o Node.js

- De l'aplicació de videoconferència us donem un esquelet del codi mig fet que heu de completar
  - és la pàgina *web* (amb HTML i JS) que implementa l'aplicació de videoconferència entre dos

- Heu de lliurar un informe a través de "La meva UdG", en un únic fitxer en format ZIP, el nom del qual tingui els noms dels membres del grup (Cognom1Nom1-Cognom2Nom2-P3.zip, p.e., VilaPere-FabregaLluis-P3.zip), i que contingui el següent:
  - una presentació (en forma de transparències) per fer una exposició oral de la feina feta (expliqueu el codi que heu fet, feu una descripció de les funcions de llibreria utilitzades, etc.)
  - el fitxers de codi

Esquelet del codi

- index.html
  - no cal modificar-lo
  - interfície *web* senzilla per reproduir el vídeo i un xat
  1. Amb el botó *Start* s'obté l'*stream* que es reprodueix a "*Local Video*"
  2. Un cop obtingut s'habilita el botó *Call* per trucar a l'altre usuari
  3. Un cop establerta la connexió, es mostra el vídeo a l'àrea "*Remote Video*", i a més s'activa la possibilitat del xat i d'acabar la trucada
  4. Per últim, amb el botó *Hangup* es finalitza la trucada

- localPeerConnection-lib.js
  - no cal modificar-lo
  - llibreria de variables globals i funcions

- localPeerConnection.js
  - SÍ cal modificar-lo
  - fa la inicialització de l'aplicació i les accions

Entorn de treball

- Cal el següent
  - una 1a màquina amb un servidor *web* segur
  - una 2a i 3a màquines amb un navegador* i una càmera *web*
  - les tres màquines unides per una xarxa

- Feu-ho com us sembli millor, però us proposem el següent:
  - la 1a màquina (amb el servidor *web* segur), que sigui una màquina virtual (VM) de *VirtualBox* amb el S.O. *Ubuntu Server*
  - la 2a i 3a màquines (amb el navegador* i càmera *web*), que siguin la màquina "real" on corre la virtual i un mòbil*, o bé la màquina "real" i un altre ordinador, o bé dos ordinadors, o bé dos mòbils, etc
  - unides per una xarxa? escolliu el mode de xarxa *bridged* de la VM; després feu la configuració de xarxa IP de l'Ubuntu de manera que la VM tingui connectivitat amb la 2a i 3a màquines

    (si hi ha un *router* NAT entre les màquines, poseu a la xarxa interna les dels navegadors, o si al revés, "obriu-hi" el #port del servidor)

* per WebRTC farem servir *Firefox* i/o *Chrome,* i si mòbil, Android

Més informació, p.e., a

- Sobre certificats autosignats, per Apache i per Node.js

  https://en.wikipedia.org/wiki/Self-signed_certificate/

  https://www.digitalocean.com/community/tutorials/how-to-create-a-ssl-certificate-on-apache-for-ubuntu-12-04

  http://tecadmin.net/setup-ssl-certificate-with-node-js-in-linux/

- Sobre WebRTC i sobre la seva API en JS

  https://webrtc.org/

  https://webrtc.org/faq/#what-is-webrtc

  https://en.wikipedia.org/wiki/WebRTC

  https://www.w3.org/TR/webrtc/

  https://www.w3.org/TR/mediacapture-streams/#dom-navigator-getusermedia

  https://www.w3.org/TR/webrtc/#rtcpeerconnection-interface

  https://www.w3.org/TR/webrtc/#rtcdatachannel

  https://w3c.github.io/webrtc-stats/webrtc-stats.html

  https://codelabs.developers.google.com/codelabs/webrtc-web/#0

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