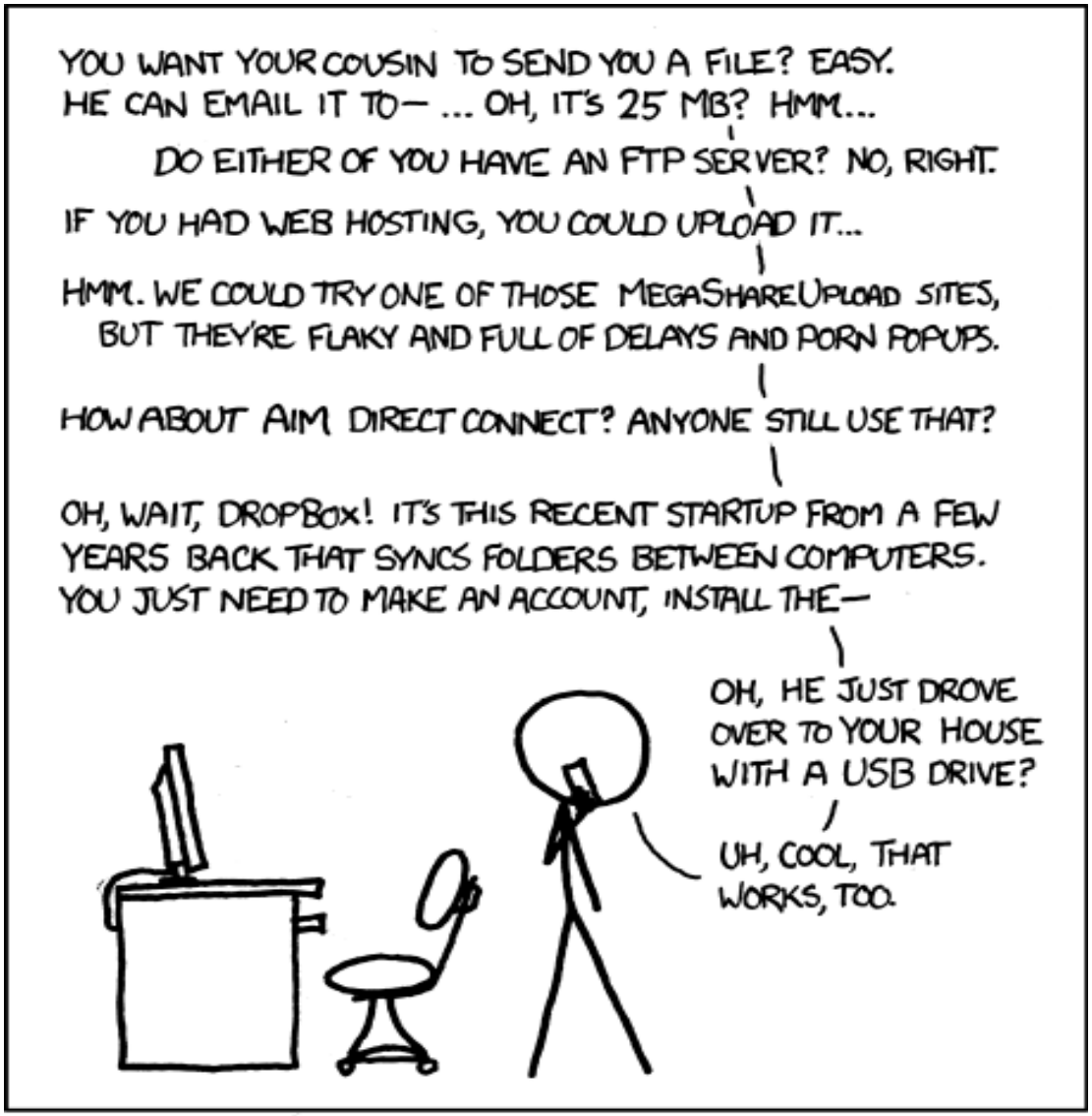
Webmission

p2p via web

Envío de ficheros. Caso de uso:



I LIKE HOW WE'VE HAD THE INTERNET FOR DECADES, YET "SENDING FILES" IS SOMETHING EARLY ADOPTERS ARE STILL FIGURING OUT HOW TO DO.

Ventajas de Webmission:

- No precisa instalación en el cliente. No plugin en el navegador.
- No existe límite de tamaño.
- Los datos no se guardan en servidor.
- No se requieren cuentas de usuario ni login.
- Es inmediato.

Webmission usa:

Node.js / Socket.io

WebRTC

Filereader

Filesystem

WebRTC

Compuesta fundamentalmente por 3 APIs:

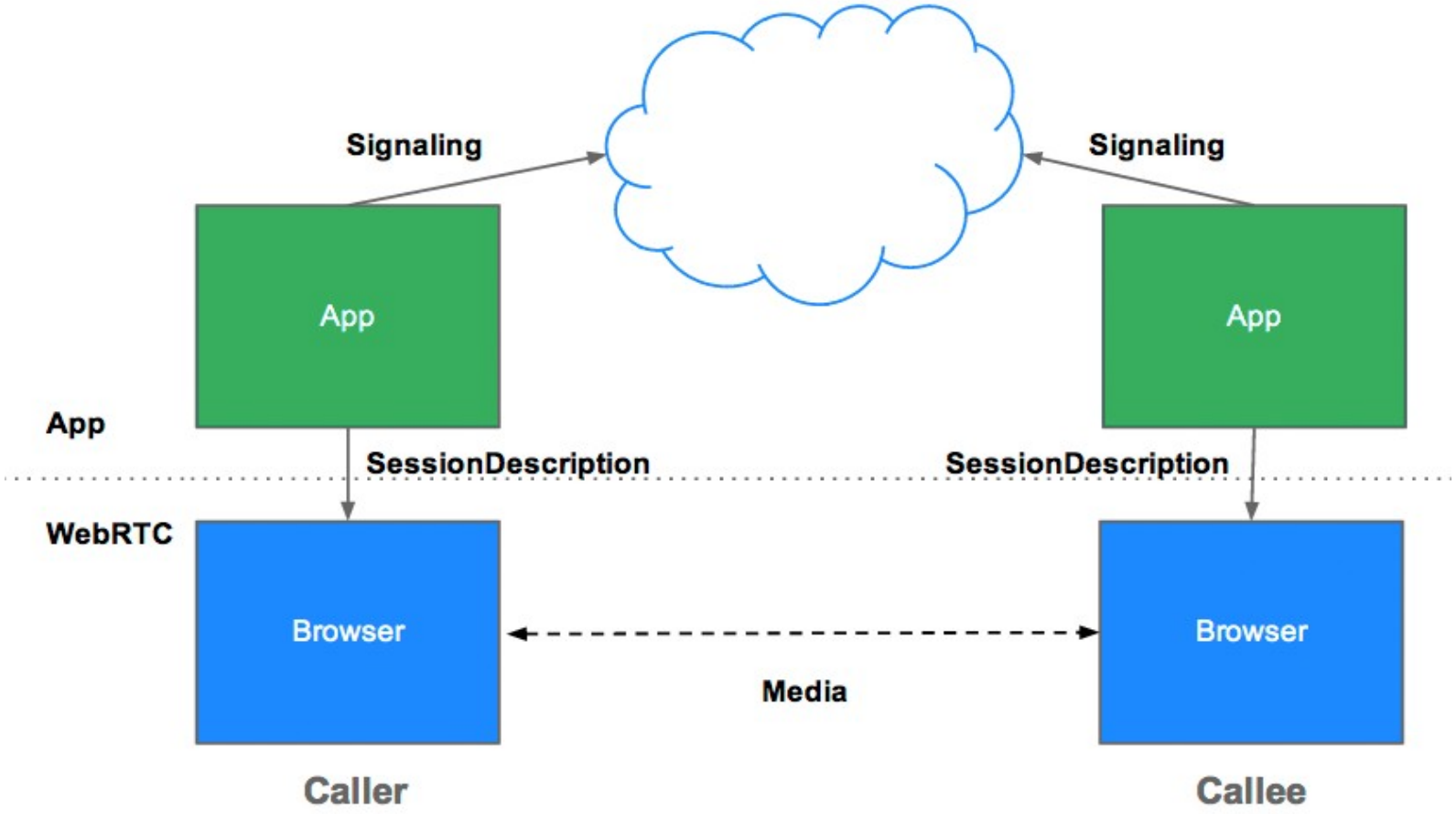
- MediaStream (transmisión de video y audio)

- RTCPeerConnection (establece y mantiene una conexión estable entre pares)

- RTCDataChannel (canal de comunicación bidireccional que admite todo tipo de datos)

No funciona sólo: necesitamos...

un mecanismo de señalización



Señalización

¿Para qué?

Para que se produzca el handshake entre los usuarios previamente al establecimiento de la conexión P2P

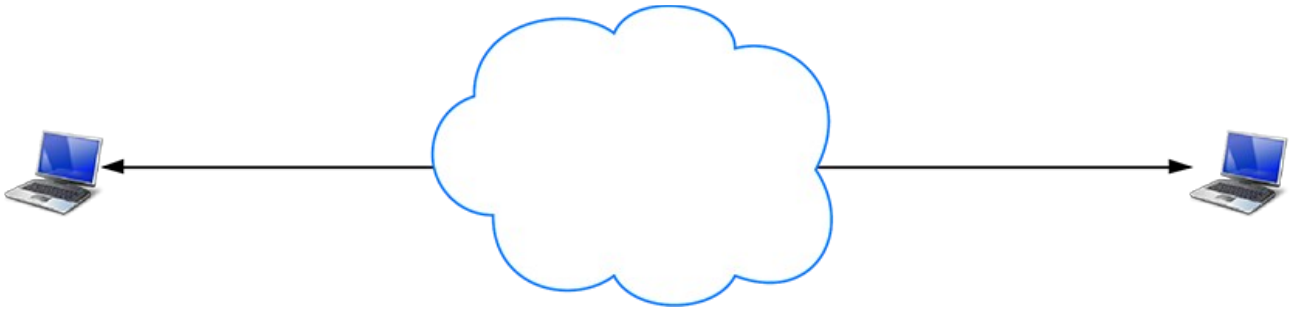
¿Qué se necesita para el handshake?

- Mensajes de control de sesión
- Configuración de red: mensajes SDP (Session Description Protocol)
- Mensajes de tipo ICE candidate (IP pública y puertos disponibles)

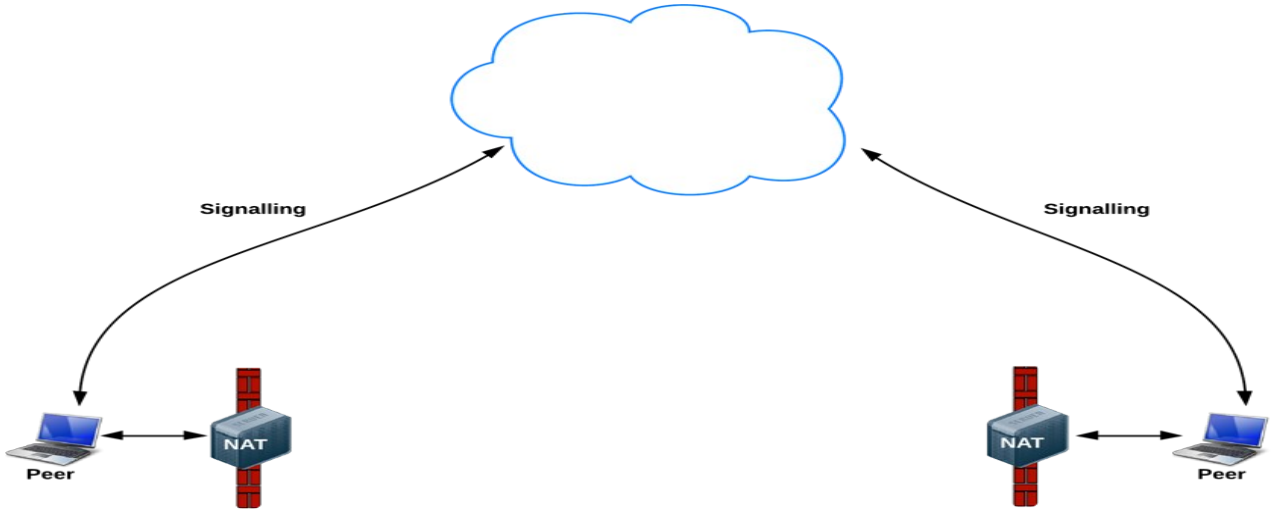
¿Cómo está hecho en Webmission?

Mediante Socket.io corriendo sobre un servidor Node.

Esto no ocurre:



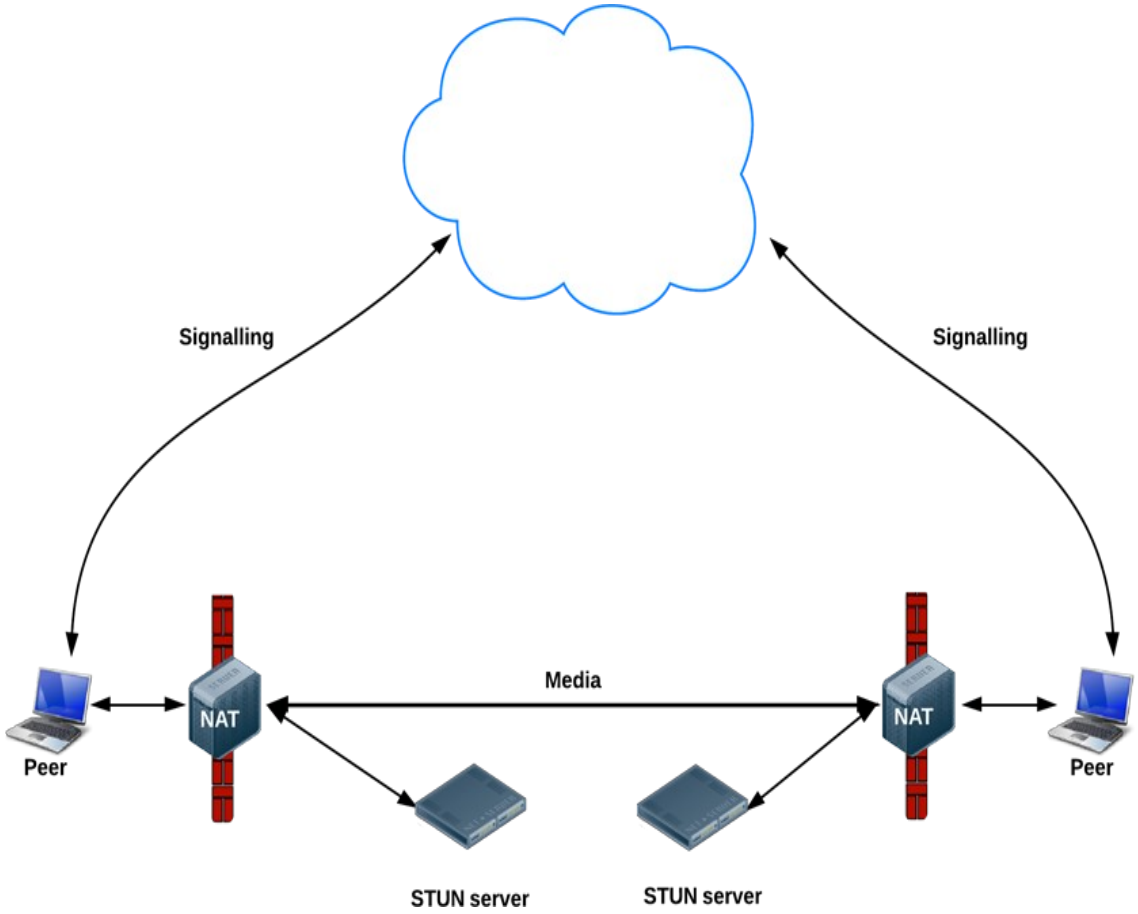
Encontramos esto:



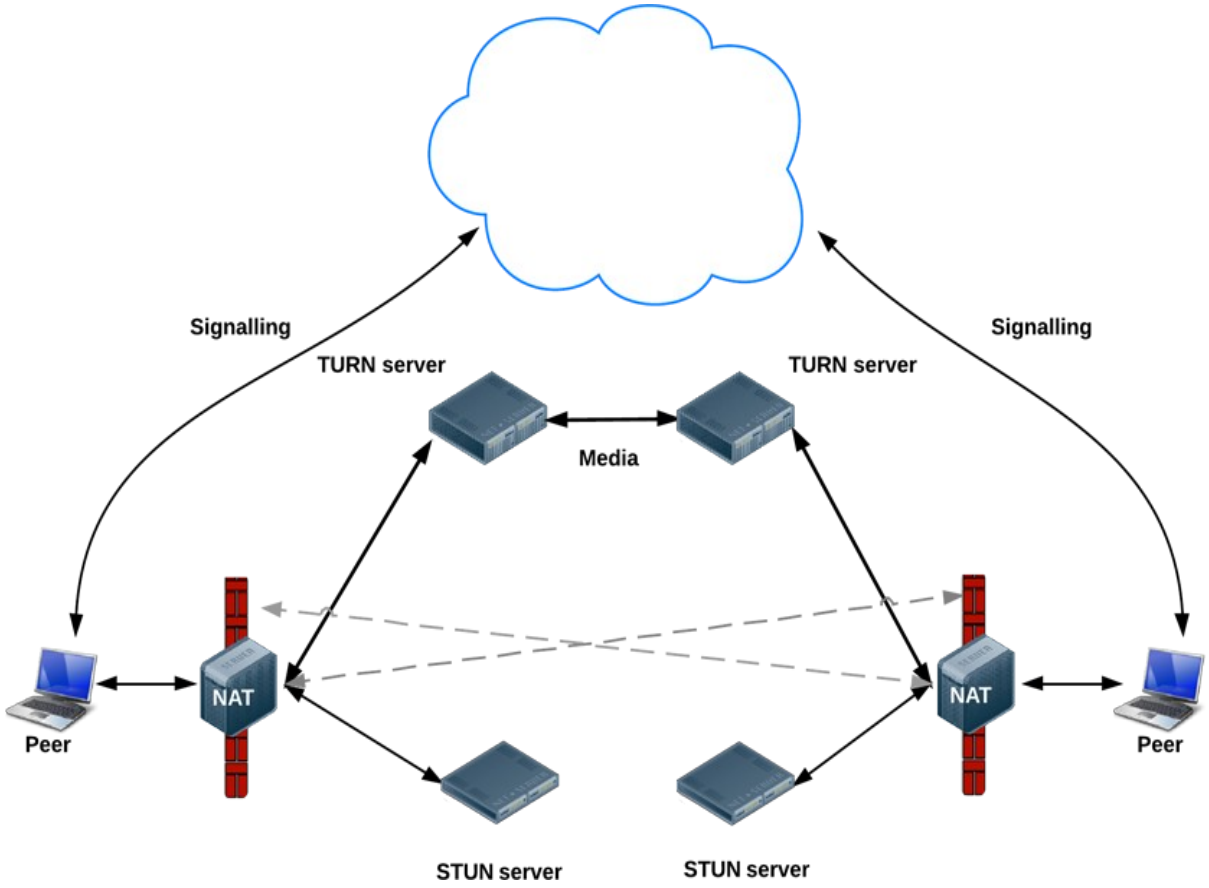
ICE (Interactive Connectivity Establishment) es usado en el mecanismo de señalización

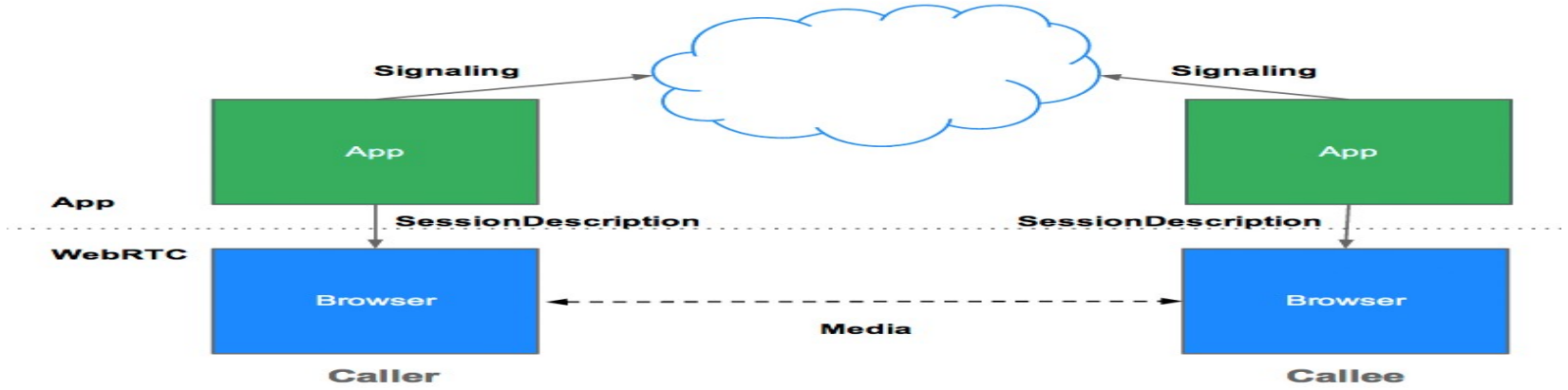
1. Intenta conexión directa, mediante UDP.
   El servidor STUN permite que cada máquina detrás de un NAT
   averigüe su propia IP pública.

2. Si no funciona lo anterior, mediante TCP



3. Si aún así no establece conexión, conexión indirecta
mediante servidor TURN (no conexión P2P).





Resumen:

- A crea una instancia de RTCPeerConnection, con manejador onIceCandidate. Establece su sesión local.

- El manejador es invocado cuando se encuentra una IP disponible

- A envía su información de red a B

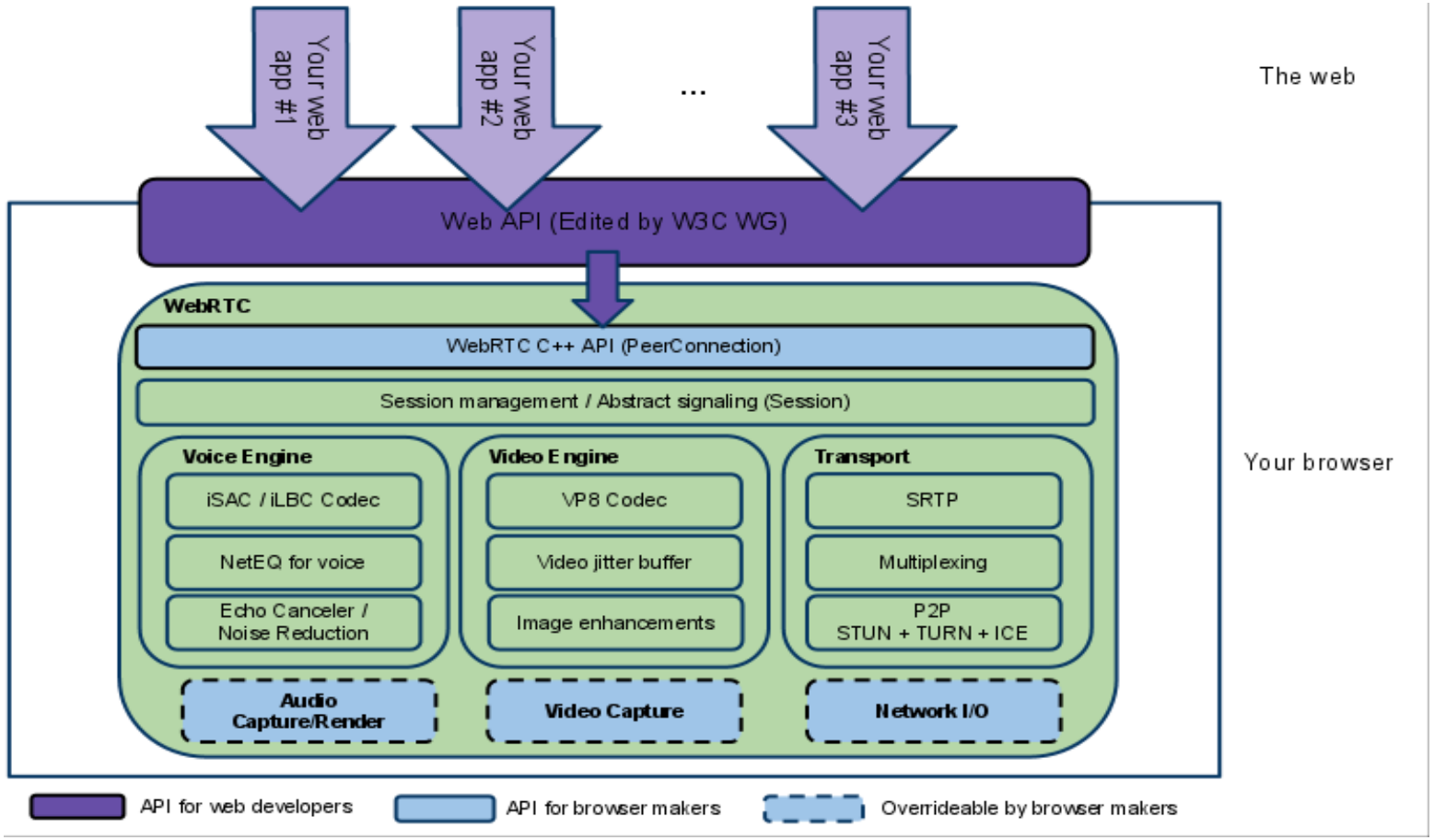
- B recibe el mensaje de A

- B añade la información de candidato que le mandan, y establece su sesión local y remota.

- Se repite el proceso, intercambiando los actores, hasta que que cada uno
haya añadido su IceCandidate y haya establecido su sesión remota con los datos del otro par.

¡Ya tenemos la conexión!

RTCPeerConnection proporciona y mantiene una conexión directa, estable y eficiente.



RTCDataChannel

Canal bidireccional de datos.

Posibilidades infinitas: redes descentralizadas, CDN, escritorio remoto, transferencias de datos...)

Una vez obtenida la conexión, el **canal de datos** mantiene en comunicación a los pares.

Webmission lo usa para:

- Comunicación por chat de texto.

- Envío de ficheros

- Información acerca del estado del envío

- Información sobre estado del canal de datos y de la conexión

Caso de uso típico:

1. Usuario A entra en la página web. La aplicación comprueba la url:
    - Si contiene únicamente el nombre del dominio: puede crear sala.
        Recarga de la página: a la url se añade un código alfanumérico.

2. El usuario indica esta nueva url a otro usuario, B.

3. B entra a través de esa url. Se realiza la conexión.

4. A ofrece un fichero a B.

5. B acepta. La aplicación en A lee el fichero mediante la API Filereader

Usuario A (emisor):

- Se carga en memoria el trozo correspondiente al primer MB (en base64)
- Este trozo se divide a su vez en fragmentos de 13KB
- Cada fragmento de 13KB se envía a través del canal de datos
- Cuando se termina de enviar el primer MB, se envía una señal al usuario B.
- Si se ha llegado a final de fichero, se envía señal de fin de fichero. Si no, vuelve a comenzar el proceso de carga y envío

Usuario B (receptor):

- Recoge cada fragmento de 13KB que va llegando en un array.
- Cuando llega señal de final de chunk (1MB), vuelco del array en el disco duro:
    - Mediante la API Filesystem
- Si no hay señal de fin de fichero, sigue recogiendo y volcando.
- Si señal de fin: creación de enlace para descarga del fichero previa conversión de los datos en base 64 a octal y creación del blob.

API Filesystem

Lectura / Escritura de archivos en zona acotada del disco duro.

Dos tipos:

Temporal:

- El navegador puede borrar datos.
- Límite de uso: De la mitad del espacio libre en disco, cada aplicación no puede superar un 20%. Ejemplo -> 50GB libres: la aplicación no puede superar los 5GB de uso.
- Disponible en navegadores webkit y, extraoficialmente, Firefox.

Persistente:

- Los datos permanecen en disco, hasta que los borre el usuario o la aplicación.
- No existe límite.
- Disponible sólo en navegadores con motor webkit.

¿Decisión? Webmission usa el modo persistente