Ejecutando Peers P2PSP en Google Chromecast

Cristóbal Medina-López,[1] Vicente González-Ruiz,[1] L. G. Casado, [1] J.A.M. Naranjo [2] y Juan Pablo García Ortiz [3]

Resumen— **Este trabajo muestra que es posible ejecutar un peer que implementa el protocolo P2PSP (Peer-to-Peer Straightforward Protocol) en un dispositivo Google Chromecast (GC) con el objetivo de realizar streaming. Actualmente GC puede realizar una descarga usando el modelo C/S (cliente/servidor) para el visionado de streaming multimedia mediante aplicaciones Web como son Youtube o NetFlix, entre otras. Para ello GC ejecuta el navegador Web Google Chrome, que es capaz de interpretar código HTML5 y JavaScript, además de incorporar el protocolo WebRTC (una tecnología que permite que varias aplicaciones en distintos dispositivos puedan comunicarse entre sí). Por otra parte, P2PSP es un protocolo específicamente diseñado para reducir la latencia en sistemas de transmisión multimedia. Para conseguirlo, los peers P2PSP (como ocurre también en otros sistemas P2P) utilizan su ancho de banda de subida para retransmitir el stream, de forma que el sistema es mucho más escalable que el modelo C/S. El P2PSP tiene además algunas características exclusivas que lo hacen especialmente adecuado para ser ejecutado en dispositivos con potencia limitada. Los experimentos realizados demuestran que un peer P2PSP ofrece una buena calidad de experiencia al ser ejecutado sobre un GC, un dispositivo de bajo coste y bajo consumo energético.**

Palabras clave— **P2PSP, chromecast, WebRTC, live streaming, peer-to-peer, HTML5.**

I. Introducción

Actualmente, los sistemas de streaming de vídeo son servicios de uso generalizado. Según Cisco Systems, más de la mitad de los datos que se transmiten son multimedia (audio y vídeo) [1]. Esto ha motivado que aparezcan multitud de arquitecturas de streaming que básicamente tratan de transmitir varias señales multimedia a los usuarios, o sistemas finales, con la máxima calidad posible, es decir, baja latencia, baja pérdida de datos, alta interactividad, alta resolución, con garantías de seguridad, etc.

La solución para streaming de vídeo en directo más usada hoy en día es el modelo C/S (Cliente/Servidor), que se basa en la idea de que el stream es generado en un servidor y, bajo demanda (pull model) o de forma automática (push model), el stream es transmitido tratando de minimizar la latencia al conjunto de clientes. Sin embargo, los sistemas puros C/S están limitados por el ancho de banda del servidor. Por este motivo, la mayoría de los portales de streaming C/S utilizan una red de entrega de contenidos (Content Delivery Network, CDN, en inglés) en la que el contenido multimedia se encuentra replicado en un conjunto de servidores. Podría afirmarse que dicha arquitectura es la que actualmente tiene más éxito ya que es usada por los sistemas de distribución multimedia más populares, como por ejemplo YouTube.

En este contexto, el modelo P2P (Peer-to-Peer) ha surgido como una clara alternativa al modelo C/S. En las redes P2P, la fuente que "inyecta" el stream en la red puede ser única y envía sólo una o unas pocas copias del stream. Los peers, que actúan como clientes y servidores, usan su ancho de banda de subida para compartir el stream recibido con otros peers de la red. Por tanto, la clave de esta tecnología radica en la solidaridad de los usuarios, que aportan su ancho de banda de subida para el bien de la comunidad de peers.

Por otra parte, actualmente, se usan una gran variedad de dispositivos móviles (smartphone, tablet, netbook, etc.), para la recepción y visionado de contenidos multimedia. El principal inconveniente del visionado en estos dispositivos es su reducido tamaño. Para solucionar este problema, Google lanzó al mercado el dispositivo Chromecast (GC). GC se conecta a un televisor o pantalla con una entrada HDMI y puede ser controlado por un dispositivo auxiliar (un smartphone, tablet o pc, por ejemplo) para que reciba un stream multimedia que finalmente se reproduce en el televisor o pantalla. Un dispositivo GC puede recibir el stream del dispositivo auxiliar o también lo puede descargar directamente de otra fuente que puede estar en Internet. En este trabajo se muestra que aunque el GC dispone de unos recursos limitados y por lo tanto tiene un bajo consumo energético, es capaz de ejecutar un peer del protocolo P2PSP.

El resto de este documento tiene la siguiente estructura. En la sección II se hace una breve revisión de los trabajos relacionados. En la Sección III se muestran las características generales del protocolo P2PSP. La Sección IV introduce el dispositivo Chromecast. La Sección V presenta las tecnologías que incorpora el dispositivo y que son necesarias para la implementación de un peer P2PSP. Las adaptaciones necesarias de las entidades P2PSP y su implementación se muestran en la Sección VI. Las limitaciones actuales del desarrollo se muestran en la Sección VII. Por último, la Sección VIII muestra las conclusiones.

II. Trabajos relacionados

En este trabajo se integran varias tecnologías en el desarrollo de un sistema P2P de transmisión de vídeo en tiempo real en dispositivos con pocos recursos y pocos requerimientos energéticos como es Google Chromecast. Algunas de esas tecnologías ya

[1]Universidad de Almería (CeiA3), España, e-mail: {cristobalmedina, leo, vruiz}@ual.es
[2]Scality, Francia, e-mail: juan.munoz@scality.com
[3]Luxunda, España, e-mail: info@luxunda.es

están desarrolladas, como es el caso de HTML5 y JavaScript y otras como P2PSP, WebRTC y MSE, están en desarrollo.

En la literatura existen estudios que integran protocolos P2P en decodificadores o receptores de televisión [2]. Debido a las limitaciones de los recursos en los dispositivos, algunos autores desplazan la red P2P de distribución al cloud, en el que cada dispositivo tiene un punto diferente de acceso al contenido [3]. En [4] se hace un análisis de los recursos necesarios para realizar streaming de vídeo en directo en dispositivos móviles y en [5] se presenta una aplicación de streaming de vídeo en directo P2P para dispositivos compatibles con Android [5]. Existe una gran heterogeneidad en cuanto a las capacidades de terminales/redes y los requerimientos de los usuarios para consumir contenido multimedia, siendo HTTP el protocolo más usado en Internet para tales propósitos [6]. En este sentido, nuestros estudios se pueden aplicar a dispositivos capaces de ejecutar un navegador que incluya la tecnología WebRTC, como Firefox o Chrome. Este estudio se centra en Chromecast debido a su bajo coste y recursos limitados.

III. P2PSP

P2PSP (Peer-to-Peer Straightforward Protocol) [7] es un protocolo de comunicación de la capa de aplicación para el streaming de contenido multimedia sobre Internet. El protocolo P2PSP puede usarse para una gran variedad de servicios de transmisión en directo que va desde pequeñas reuniones hasta grandes sistemas de IPTV (Internet Protocol TeleVision). A diferencia del tradicional modelo C/S (Cliente/Servidor) y su extensión usando CDNs (Content Delivery Networks) para el streaming de vídeo, en el modelo P2P cada usuario contribuye con su ancho de banda de subida al sistema. Por esta razón, los sistemas P2P son en general mucho más escalables y robustos que las arquitecturas basadas en C/S.

El protocolo P2PSP se centra en dos aspectos fundamentales:

1. A pequeña escala, todos los peers se comunican entre sí, todos con todos, utilizando mensajes punto a punto (unicast).
2. En principio y para que el sistema escale suficientemente, un peer debe ser tan solidario con el resto de peers como éstos lo son con él. De lo contrario, el peer no podrá formar parte de la red P2P ("team" en la jerga P2PSP). Dicha solidaridad se expresa en términos de ancho de banda. Por lo tanto, para que un peer forme parte del team, debe de enviar al team, en promedio durante cada cierto intervalo de tiempo, tanto como recibe del team.

A. Características del P2PSP

Las principales propiedades que definen el protocolo son:

- **Baja latencia:** El protocolo P2PSP ha sido especialmente definido para realizar streaming minimizando el retardo de transmisión, lo que puede ser útil para emitir vídeo (y su audio asociado) en directo. Permite transmitir a través de una red de comunicaciones basada en la conmutación de paquetes, un evento que se está produciendo en ese momento así como eventos ya creados o grabados.
- **Independencia del contenido:** El protocolo no interpreta, ni trata de analizar, qué se está transmitiendo.
- **Arquitectura modular:** El número de módulos depende de los requisitos finales.
- **Simplicidad:** El módulo más básico es muy simple, lo que permite que pueda ser ejecutado en sistemas con recursos muy limitados.
- **Multicasting real:** P2PSP puede usar IP multicast en caso de estar disponible.
- **Soporte para peers en redes privadas:** Los peers pueden estar en redes privadas, incluso detrás de NAT simétricos.
- **Soporte para modelos de transmisión de contenido escalable:** El protocolo es totalmente compatible con multiresolución y las técnicas de streaming adaptativo [8].
- **Alta integración:** Es compatible con el modelo cliente/servidor, convirtiéndose en un modelo híbrido P2P/CS.

B. Entidades

El protocolo P2PSP define los siguientes tipos de entidades:

- **Source (O):** El nodo source, que en realidad no pertenece al team P2PSP, se encarga de producir el stream que es transmitido en la red P2PSP. Normalmente es un servidor de streaming que usa el protocolo HTTP. Un ejemplo de source puede ser el servidor Icecast. El source controla el bit-rate de la transmisión en la red. Recuérdese que el protocolo P2PSP no analiza lo que transmite.
- **Player (L):** Un player, que técnicamente tampoco forma parte del team P2PSP, se encarga de solicitar y reproducir el stream.
- **Splitter (S):** Esta entidad recibe el stream desde el source, lo divide en trozos del mismo tamaño y los envía a los nodos peers. En un team P2PSP hay al menos un splitter.
- **Peer (P):** Un peer recibe los trozos desde el splitter y desde otros peers, los concatena en orden para crear el stream original y lo envía normalmente a un player. Algunos trozos son también enviados a otros peers. En un team P2PSP pueden existir cientos de peers.

La Figura 1 muestra un team P2PSP simple con 3 peers. El source envía el stream de vídeo al splitter, el splitter lo divide en trozos (chunks) del mismo tamaño y envía un chunk diferente a cada peer. Los peers se encargan de reenviarse los chunks recibidos del splitter entre ellos con el objetivo de que todos tengan el stream completo. Finalmente, cada peer

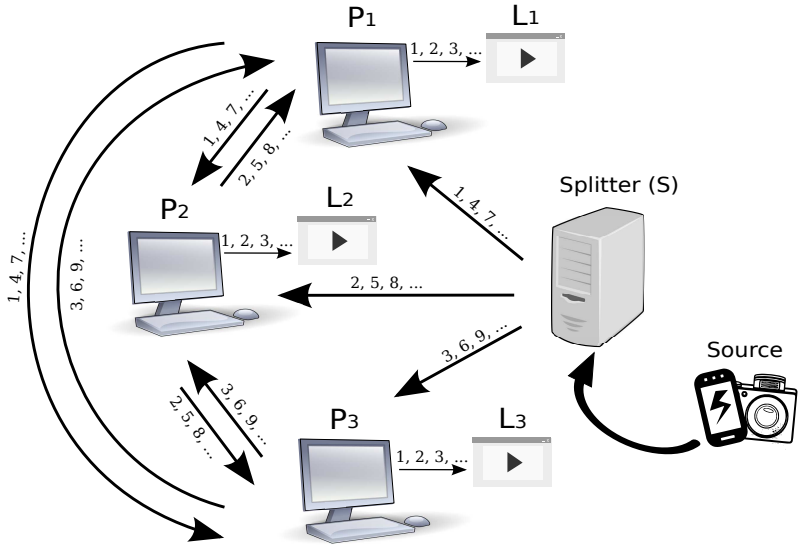

Fig. 1

Ejemplo de team P2PSP. Las flechas y sus etiquetas indican la retransmisión de chunks. S envía un chunk diferente a cada peer usando un esquema Round-Robin. Los peers envían cada chunk recibido de S al resto de peers del team.

reordena los chunks y envía el stream al player.

IV. Google Chromecast

Google Chromecast (GC) es un pequeño dispositivo de bajo coste y consumo, que conectado a una pantalla o a un televisor a través del puerto HDMI puede ampliar sus capacidades con conexión a Internet y reproducción de contenido multimedia en alta definición [9]. GC se ha convertido en un dispositivo muy popular debido a su reducido precio, con millones de usuarios en todo el mundo.

Un GC cuenta en su interior con un chip Marvell DE3005-A1, una memoria RAM de 512MB y una memoria de almacenamiento de 2GB, incluyendo también codecs para decodificación por hardware de vídeos con los formatos de compresión VP8 y H264.

En lo referente al software, Chromecast ejecuta una versión especialmente adaptada para este dispositivo de ChromeOS. Chromecast es capaz de ejecutar una instancia del navegador web Google Chrome, permitiendo la realización de este estudio al ser compatible con HTML5, JavaScript y WebRTC.

Chromecast puede usarse como un dispositivo de recepción y reproducción de contenido multimedia de dos maneras:

- Recibiendo el contenido multimedia enviado desde otro dispositivo auxiliar, como puede ser un smartphone, un ordenador o una tablet.
- Descargando directamente el contenido multimedia de Internet usando su navegador Google Chrome, siguiendo las indicaciones realizadas desde el dispositivo auxiliar.

Nosotros nos basaremos en la última opción, que es posible gracias a la simplicidad del protocolo P2PSP y al conjunto de tecnologías que incorpora el navegador Google Chrome que se ejecuta en el GC.

V. Tecnologías necesarias para el peer P2PSP en Google Chromecast

Uno de los aspectos más interesantes del GC es que ejecuta una versión de ChromeOS y el navegador Web Google Chrome. Google Chrome es capaz de ejecutar aplicaciones complejas gracias a la evolución del lenguaje HTML junto con otras APIs que incorpora o puede incorporar, permitiendo al usuario disfrutar de una experiencia completa en el navegador sin la necesidad de tener la correspondiente aplicación de escritorio, como podría ser un reproductor multimedia. En la actualidad es posible visualizar tanto contenido multimedia como complejos sistemas de finanzas o herramientas de modelado, por poner algunos ejemplos.

Cada vez más usuarios prefieren el uso de las aplicaciones en el navegador en lugar de sus equivalentes de escritorio. Este tipo de aplicaciones ofrecen varias ventajas, como la independencia de la plataforma, el acceso remoto (no es necesario tener instalada la aplicación de forma local), la facilidad de uso, etc. No obstante, aún existen algunas limitaciones, como por ejemplo que la aplicación no puede ser multihebrada.

A continuación se describirán algunas de las tecnologías necesarias para una implementación del protocolo P2PSP en la Web, lo que es necesario para su uso en Google Chromecast.

A. HTML5 y JavaScript

HTML5 (HyperText Markup Language 5) es la quinta revisión del lenguaje básico de la Web [10]. El código HTML es interpretado por el navegador Web, permitiendo que el usuario interaccione con el contenido de forma sencilla. En la versión 5 se establecen nuevos elementos y atributos. Por ejemplo, se introducen elementos multimedia como la etiqueta `<video>` y `<audio>` para proporcionar funcionalidades multimedia directamente en el navegador, me-

diante una interfaz estandarizada, haciendo uso de codecs para mostrar los contenidos. Además, se incluyen elementos de comunicación en forma de APIs como son los WebSockets, permitiendo una comunicación bidireccional entre un servidor y el navegador Web.

Otro elemento importante en este desarrollo es JavaScript. JavaScript es un lenguaje de programación interpretado y se define como orientado a objetos. La forma de uso más habitual de JavaScript es en el lado del cliente, permitiendo mejoras en la interfaz de usuario y aplicaciones Web dinámicas. No obstante, puede ser usado en el lado del servidor así como en aplicaciones externas a la Web.

El uso habitual de JavaScript en el navegador Web es embebido en páginas HTML que interaccionan con el modelo de objetos del documento (DOM: Document Object Model). El DOM es una API que proporciona una interfaz que permite acceder a objetos para la representación de documentos HTML y XML, así como para su modificación.

B. WebSocket

WebSocket es un protocolo que provee al navegador Web de un canal de comunicación bidireccional con un servidor sobre una única conexión TCP [11] [12]. Su diseño se basa en las infraestructuras HTTP y HTTPS existentes, de modo que se ha diseñado para funcionar sobre los puertos 80 y 443, además de sobre proxies HTTP. Existe la posibilidad de establecer otro puerto mediante el intercambio de mensajes HTTP. El protocolo WebSocket se estandarizó en 2011 por el IETF y actualmente está siendo estandarizada su API por el W3C. Gracias a este protocolo es posible mantener conexiones persistentes entre el navegador y el servidor donde ambas partes pueden comenzar el intercambio de información en cualquier momento.

C. WebRTC

(Web Real Time Communications) es un proyecto de software libre mantenido por Google, Mozilla y Opera, que pretende convertirse en un nuevo estándar que extiende las capacidades del navegador Web [13]. WebRTC permite al navegador comunicarse directamente y en tiempo real con otros navegadores utilizando una arquitectura peer-to-peer.

La API de WebRTC está siendo definida conjuntamente por el W3C (World Wide Web Consortium) y por el IETF (Internet Engineering Task Force). La misión de ambas organizaciones es conseguir una API JavaScript que junto con las etiquetas necesarias del estándar HTML5 pueda definir un nuevo protocolo de comunicación con el que sea posible comunicar los navegadores directamente entre sí. Además, la API define también otros elementos que permiten acceder a la webcam y al micrófono (entre otros periféricos) sin necesidad de instalar un plugin.

Recientemente varias aplicaciones han hecho uso de tecnología WebRTC. Por ejemplo, para sistemas de vídeo conferencias de ámbito general [14], para docencia [15] o en sistemas de streaming de vídeo bajo demanda (VoD) [16]. En [17] proponemos la ejecución de un peer P2PSP en el navegador web usando WebRTC. WebRTC también se ha usado en sistemas de distribución de contenidos mediante P2P [18].

C.1 PeerConnection

PeerConnection es un componente de WebRTC que permite establecer una comunicación estable y eficiente de streaming de datos entre peers, es decir, de navegador a navegador. Ambos navegadores (o peers) ejecutan una instancia de la misma aplicación JavaScript. Antes de conseguir la conexión entre los peers es necesario un intercambio previo de mensajes para acordar los parámetros de la comunicación, lo que se hace a través de un servidor de señalización (signaling server, véase la sección V-C.3). Una vez establecida la conexión, los navegadores pueden intercambiar mensajes directamente, sin la necesidad de pasar por ningún servidor intermedio.

PeerConnection hace uso del protocolo ICE junto con los servidores STUN [19] y TURN [20] para permitir a las transmisiones de stream basadas en el protocolo UDP [21] atravesar NATs y firewalls.

C.2 DataChannel

Hasta la aparición de WebRTC, enviar datos entre varios navegadores era un proceso ineficiente, poco escalable y falto de privacidad, porque era necesario enviar todos los datos a un servidor intermedio que era el encargado de reenviar la información al resto de los navegadores involucrados en la comunicación.

La API DataChannel de WebRTC permite que el navegador Web realice una comunicación de datos bidireccional con otro navegador (o peer), ofreciendo un conjunto flexible de tipos de datos a transmitir. Entre los tipos de datos soportados están Blob y ArrayBuffer, específicamente diseñados para trabajar con objetos binarios de gran tamaño.

Además, DataChannel cuenta con dos modos de funcionamiento: el modo no confiable (similar a UDP) y el modo confiable (similar a TCP). El modo no confiable permite un funcionamiento más rápido y por lo tanto más adecuado para transmisión de stream en directo, pero presenta la desventaja de que no garantiza ni la entrega ni la llegada en orden de los mensajes, que sí es garantizada en el modo confiable.

C.3 Signaling Server

La especificación de WebRTC establece que debe existir un servidor en Internet el cual, mediante algún protocolo de comunicación, presente un nuevo peer al resto de ellos. La señalización se describe de forma abstracta ya que WebRTC no especifica cómo ha de llevarse a cabo.

Cualquier sistema conectado a Internet puede realizar las tareas del Signaling Server. Existen protocolos estándar que se usan para fines similares, como SIP o XMPP, aunque se puede usar cualquier protocolo que permita una comunicación bidireccional,

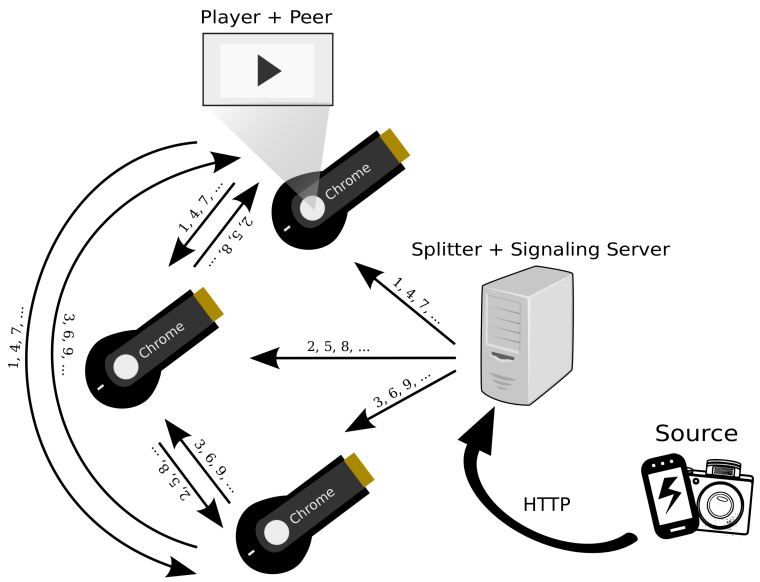

Fig. 2

Un escenario equivalente al de la Figura 1 pero usando una implementación de los peers para Chromecast. Se han fusionado algunas entidades como los peer+player y splitter+signaling server.

tal y como se verá en la sección VI.

D. Media Source Extensions

Media Source Extensions, también conocido como MSE, es una especificación elaborada por el HTML Working Group del W3C con el objetivo de extender las funcionalidades del objeto HTMLMediaElement [22]. Las MSE permiten a los desarrolladores construir streams multimedia dinámicos en JavaScript con las etiquetas `<audio>` y `<video>`. Adicionalmente, se define un modelo de buffering que describe cómo los agentes de usuario deben actuar cuando se añaden diferentes medios.

El objeto MediaSource representa una fuente de datos multimedia para un HTMLMediaElement. Se usa el objeto SourceBuffer para añadir datos multimedia al buffer en tiempo real al mismo tiempo que el elemento vídeo está reproduciendo el contenido.

Las MSE aún están en proceso de estandarización por el W3C, siendo actualmente una *Candidate Recommendation*. Algunas de las ventajas que incorpora y/o extiende del elemento Media Source son:

- Permite construir streams multimedia mediante JavaScript, independientemente de cómo el elemento multimedia sea incrustado en la aplicación Web.
- Define un modelo de buffering que facilita casos de uso como streaming adaptativo, inserción de publicidad, saltos de tiempo y edición de vídeo.
- Aprovecha al máximo la caché del navegador.
- No requiere soporte para un codec particular.

VI. Implementación

En la Sección III hemos visto las entidades que forman parte de un sistema P2PSP y cuál es su arquitectura de red. La Figura 2 muestra cómo sería un escenario equivalente integrando los peers en los dispositivos Chromecast gracias al uso de las tecnologías HTML5, JavaScript, WebRTC y MSE.

A. Splitter+signaling server

Uno de los elementos básicos y necesarios para llevar a cabo una implementación del P2PSP es la entidad splitter, ya que será la encargada de recibir el stream, dividirlo en chunks del mismo tamaño y enviarlo al team. Además, el splitter tiene la función de presentar cada peer entrante al resto del team. Esta última capacidad es similar a la que debe llevar a cabo el Signaling Server en WebRTC durante el proceso de señalización. Los dos procesos son obligatorios, el primero, para cumplir la especificación del protocolo P2PSP y el segundo, para iniciar una PeerConnection en WebRTC. Por tanto, parece lógico que el mismo dispositivo realice ambas funciones. Para realizar una comunicación bidireccional entre el navegador y el Signaling Server se puede usar WebSockets (ver sección V-B), el cual se encuentra disponible en HTML5. Normalmente, los navegadores que soportan WebRTC también soportan WebSocket. Usar JSON (un formato ligero para el intercambio de datos) junto a WebSocket es una buena opción para realizar el intercambio del objeto *session description* del protocolo SDP en WebRTC.

La Figura 3 muestra el proceso completo desde que un peer entra en el team hasta que comienza a recibir chunks. En primer lugar se procede siguiendo el protocolo P2PSP, es decir, el peer que quiere entrar en el team (A en este caso) envía un mensaje *Hello* al splitter, que le contesta enviándole la lista de peers y la cabecera del vídeo. A continuación se inicia el proceso de señalización. Para ello, el peer A pregunta al servidor STUN cuál es su IP_publica:Puerto y STUN responde con la información solicitada. Entonces, A envía un mensaje de tipo *SDP Offer* al

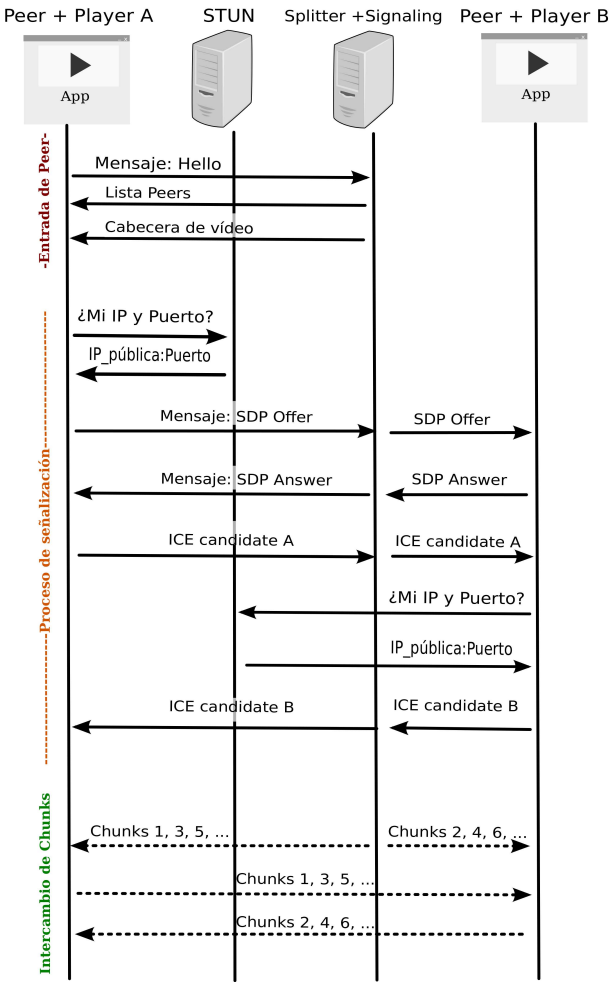

Fig. 3
Secuencia de intercambio de mensajes a la llegada de un peer en un team P2PSP.

que B tiene que dar respuesta con uno de tipo *SDP Answer* para concluir la descripción de las sesiones de comunicación multimedia. Además, A y B comparten todas las IPs por las que podrían ser alcanzados usando mensajes del tipo *ICE candidate*. Hasta este momento, A y B se han comunicado a través del splitter+signaling server. Una vez realizados estos intercambios de mensajes, los peers ya pueden realizar una comunicación directa entre ellos, y por tanto, pueden intercambiar los chunks.

Este proceso es posible si A no se encuentra detrás de un NAT simétrico. En caso contrario, todas las comunicaciones pasarían a través de un servidor TURN.

B. Peer+player

El usuario sólo tiene que lanzar una instancia del peer del protocolo P2PSP, que está compuesto por código JavaScript, HTML5 y WebRTC, en el navegador Chrome del Google Chromecast para disfrutar del contenido multimedia directamente en la televisión, gracias a que el navegador Chrome del GC ya dispone de un player multimedia. La reproducción del stream se lleva a cabo gracias al componente MSE descrito en la sección V-D. Cuando el buffer del peer P2PSP se ha llenado a la mitad de su capacidad, se comienza a enviar su contenido al buffer del reproductor que se encuentra en el objeto *MediaSource* (véase la sección V-D).

Para la comunicación entre peers en distintos GC se usa WebRTC, ya que permite crear un canal de comunicación directo entre los distintos navegadores (peers) usando un protocolo de transporte ligero conocido como SCTP (Stream Control Transmission Protocol) [23].

VII. Limitaciones del desarrollo

No obstante, existen algunas limitaciones que hacen que la implementación P2PSP en GC no tenga las mismas cualidades que la versión de escritorio (p2psp.org). Entre ellas, se pueden destacar las siguientes:

1. **Independencia del contenido:** En la actualidad, existen formatos de vídeo y audio no soportados en Chromecast. Por lo tanto, aunque el P2PSP sea agnóstico respecto al contenido, es necesario acordar un formato de vídeo y audio específico compatible con este tipo de dispositivos.
2. **Estado de la API WebRTC:** La especificación de WebRTC es sólo un borrador y no se sabe cuándo se convertirá en estándar ni qué cambios tendrá la API cuando esto ocurra.
3. **Media Source Extensions (MSE):** Para reproducir el contenido multimedia se usan las MSE. Al igual que la API WebRTC, existen navegadores que no disponen de MSE, aunque no es el caso del Chrome en Chromecast. Sin embargo, esta tecnología está más cerca de convertirse en un estándar ya que es una *Candidate Recommendation* del W3C. Por otra parte, los sistemas que soportan MSE no son compatibles con una gran variedad de codecs. Esta situación obliga a usar dos codec específicos: (1) El formato WebM con codec VP8+Vorbis específicamente codificado para streaming de vídeo o (2) el formato MPEG-DASH (MP4 segmentado).
4. **Limitación en el número de PeerConnection:** Por defecto, y aunque la especificación no dice nada con respecto al número máximo de peers, las implementaciones especificas de la API WebRTC limitan el número de PeerConnection. En concreto, Chrome limita a 256 conexiones por peer [24]. Esta característica hace que el tamaño del team en la implementación del P2PSP para Chromecast no pueda ser superior a 256 peers. Entendemos que esta limitación será eliminada conforme estas tecnologías se popularicen. Por otra parte, construir teams más grandes también es posible diseñando teams jerárquicos.

VIII. Conclusiones

En este trabajo se ha realizado un análisis de las tecnologías de las que dispone GC con el objetivo de ejecutar un peer del protocolo P2PSP completa-

mente en el dispositivo. Nuestra conclusión es que se ha podido realizar una implementación de este tipo en GC. No sólo es posible para los dispositivos GC, sino para todos los que soporten las tecnologías descritas a lo largo de este trabajo. En concreto, todas estas tecnologías están siendo implementadas en navegadores Web comerciales como Mozilla Firefox y Google Chrome, lo que permite desarrollar una red P2PSP heterogénea donde pueden interaccionar distintos tipos de dispositivos para compartir vídeo en tiempo real. Por ejemplo, un smartphone y un PC con navegador Chrome o Firefox podrían formar parte del mismo team que un dispositivo Chromecast.

Existen algunas limitaciones del desarrollo, principalmente el tipo de codecs soportados y el número de conexiones *PeerConnection*. Por tanto, es necesario el avance en el desarrollo e implementación de las tecnologías involucradas, hasta que se conviertan en estándar, para conseguir llegar a una implementación sencilla y eficiente del protocolo P2PSP en la Web y en dispositivos GC.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Ministerio de Ciencia e Innovación (TIN2012-37483), la Junta de Andalucía (P11-TIC-7176) y el Fondo Europeo de Desarrollo Regional (FEDER).

Referencias

[1] Cisco VNI, "Cisco visual networking index: global mobile data traffic forecast update, 2014–2019," *Cisco Public Information*, 2015.

[2] Yu-Heng Lin, Can Yang, and Xin-Xin Chen, "Design and implementation of embedded p2p streaming media system," in *Machine Learning and Cybernetics, 2009 International Conference on*, July 2009, vol. 3, pp. 1364–1369, DOI:10.1109/ICMLC.2009.5212313.

[3] A. Gaeta, S. Kosta, J. Stefa, and A. Mei, "Streamsmart: P2p video streaming for smartphones through the cloud," in *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2013 10th Annual IEEE Communications Society Conference on*, June 2013, pp. 233–235, DOI:10.1109/SAHCN.2013.6644983.

[4] Jongmyoung Kim and Seungchul Park, "Resource analysis for mobile p2p live video streaming," in *Ubiquitous Information Technologies and Applications*, Young-Sik Jeong, Young-Ho Park, Ching-Hsien (Robert) Hsu, and James J. (Jong Hyuk) Park, Eds., vol. 280 of *Lecture Notes in Electrical Engineering*, pp. 245–252. Springer Berlin Heidelberg, 2014, DOI:10.1007/978-3-642-41671-2_32.

[5] P.M. Eittenberger, M. Herbst, and U.R. Krieger, "Rapidstream: P2p streaming on android," in *Packet Video Workshop (PV), 2012 19th International*, May 2012, pp. 125–130, DOI:10.1109/PV.2012.6229724.

[6] Christian Timmerer, Carsten Griwodz, Ali C. Begen, Thomas Stockhammer, and Bernd Girod, "Guest editorial adaptive media streaming," *Selected Areas in Communications, IEEE Journal on*, vol. 32, no. 4, pp. 681–683, April 2014, DOI:10.1109/JSAC.2014.140401.

[7] Cristobal Medina-López, J.A.M. Naranjo, Juan Pablo García-Ortiz, L. G. Casado, and Vicente González-Ruiz, "Execution of the P2PSP protocol in parallel environments," in *Actas XXIV Jornadas de Paralelismo*, Guillermo Botella y Alberto A. Del Barrio Garcia, Ed., Madrid, Septiembre 2013, pp. 216–221.

[8] M. Ghareeb, A. Ksentini, and C. Viho, "Scalable video coding (svc) for multipath video streaming over video distribution networks (vdn)," in *Information Networking (ICOIN), 2011 International Conference on*, Jan 2011, pp. 206–211.

[9] Google Inc., "Google Chromecast Oficial Website," http://www.google.com/intl/en/chrome/devices/chromecast/, 2013.

[10] W3C, "HyperText Markup Language, versión 5," http://www.w3.org/TR/html5/, October 2014.

[11] W3C, "The WebSocket API," http://www.w3.org/TR/websockets/, September 2012.

[12] Peter Lubbers and Frank Greco, "Html5 web sockets: A quantum leap in scalability for the web," *SOA World Magazine*, 2010.

[13] The WebRTC Team, "Web Real-Time Communication," http://www.webrtc.org/, 2011.

[14] Pedro Rodríguez Pérez, Javier Cerviño Arriba, Irena Trajkovska, and Joaquín Salvachúa Rodríguez, "Advanced videoconferencing based on webrtc," in *IADIS multi conference on computer science and information systems*, 2012.

[15] Jorge Humberto Rubiano, Andrés Felipe Mena, Juan Carlos Hernández, et al., "WebRTC-Una nueva tecnología web al servicio de la educación. Caso en VirtualNet 2.0," in *Cuarta Conferencia de Directores de Tecnología de Información, TICAL2014*, 2014.

[16] J.K. Nurminen, A.J.R. Meyn, E. Jalonen, Y. Raivio, and R. Garcia Marrero, "P2P media streaming with HTML5 and WebRTC," in *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*, April 2013, pp. 63–64.

[17] Cristóbal Medina-López, Juan Pablo García Ortiz, J.A.M Naranjo, L.G. Casado, and Vicente González-Ruiz, "IPTV using P2PSP and HTML5+WebRTC," in *The Fourth W3C Web and TV Workshop (Web & TV Convergence)*, Munchen, Germany, March 2014, IRT (Institut für Rundfunktechnik), Munchen, Germany, p. Paper Submission 5, The World Wide Web Consortium (W3C).

[18] Flávio Ribeiro Nogueira Barbosa and Luiz Fernando Gomes Soares, "Towards the application of webrtc peer-to-peer to scale live video streaming over the internet," in *Simpósio Brasileiro de Redes de Computadores (SBRC)*, 2014.

[19] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing, "Session traversal utilities for nat (stun)," http://tools.ietf.org/html/rfc5389, October 2008.

[20] J. Rosenberg, R. Mahy, and P. Matthews, "Traversal using relays around nat (turn): Relay extensions to session traversal utilities for nat (stun)," http://tools.ietf.org/html/draft-ietf-behave-turn-16, July 2009.

[21] J. Postel, "User datagram protocol," https://www.ietf.org/rfc/rfc768.txt, August 1980.

[22] W3C, "Media Source Extensions," http://www.w3.org/TR/media-source/, March 2015.

[23] Network Working Group, "Stream control transmission protocol," http://tools.ietf.org/html/rfc4960, September 2007.

[24] Vikas Marwaha, "OnIceCandidate not called for more than 10 Peer Connections," https://code.google.com/p/webrtc/issues/detail?id=1343, January 2013.