Plataforma Web 2.0 para la Sincronización Distribuida de Contenidos Multimedia e Interacción Social

Jordi Belda[1], Mario Montagud[1,2], Fernando Boronat[1], Javier Pastor[1]

[1] Immersive Interactive Media (IIM) R&D Group

[1] Campus de Gandia - Universitat Politècnica de València (UPV), Spain

[2] Centrum Wiskunde & Informatica (CWI), Amsterdam (The Netherlands)

{jorbelva@, mamontor@, fboronat@dcom., fjpastor@dib.}upv.es

Keywords: IDMS, Interactive Media, Social Interaction, Social TV, Synchronization, Web TV.

Abstract. Shared media experiences between geographically distributed users are gaining momentum. Relevant examples are Social TV, synchronous e-learning and multi-player online games. This paper presents a first release of Wersync, an adaptive web-based platform that provides distributed media synchronization and social interaction (via shared navigation control commands and text chat channels) across remote users. By using Wersync, users can create or join on-going sessions for concurrently consuming the same media content with other remote users in a synchronized manner. Additionally, Wersync provides two social presence mechanisms to encourage the participation of external users in on-going sessions and two privacy mechanisms. Wersync has been developed by exclusively relying on standard web-based technologies, which ensures cross-network, cross-platform and cross-device support. The evaluation results and a link to a demo video prove the satisfactory performance of Wersync, and its functionalities, respectively.

Introducción

Tradicionalmente, familiares y amigos se han reunido en lugares físicos comunes para consumir contenidos multimedia (p.ej., contenidos de TV) juntos. Esto les posibilita interactuar, socializar, compartir impresiones y emociones, revivir momentos pasados, etc. Quizás, el ejemplo más ilustrativo sea el de un grupo de amigos quedando en la casa de uno de ellos o en un bar para ver juntos un partido de fútbol transcendente.

Sin embargo, actualmente se está gestando un cambio de paradigma en cuanto al consumo de contenidos multimedia. Nos encontramos ante una sociedad globalizada en la que miembros de una misma familia y amigos viven en diferentes ciudades o países, por motivos de trabajo, estudios, etc. Esta segregación geográfica imposibilita muchas veces el poder disfrutar de las situaciones previamente descritas.

Gracias a los avances tecnológicos, como son las tecnologías de acceso y distribución de contenidos, la amplia conectividad de los dispositivos de consumo, y el *boom* del Social Media, estas experiencias multimedia compartidas se pueden recrear aunque los usuarios no estén físicamente en el mismo sitio. En este nuevo paradigma, varios usuarios remotos pueden estar consumiendo simultáneamente el mismo contenido mientras interactúan en tiempo real mediante servicios de chat (p.ej., WhatsApp, Twitter) o audio/video conferencia (p.ej., Skype). Así pues, cada usuario podría ver el partido de fútbol desde un lugar diferente (incluso en diferentes ciudades o países), pero podría seguir comentando el partido y celebrando los goles juntos con sus amigos. Cuando se trata de consumo de contenidos de TV, esto se conoce comúnmente como la TV Social.

Aparte de la TV Social, se pueden destacar otros tipos de experiencias multimedia compartidas en red muy comunes, como la educación a distancia en tiempo real, los juegos en red multi-jugador y los servicios de multi-conferencia [1].

Sin embargo, proporcionar de manera satisfactoria este tipo de experiencias compartidas presenta múltiples desafíos [2]. En particular, varios componentes tecnológicos deben ser desarrollados e implantados para que sean realmente inmersivas e interactivas. Un factor clave que va a determinar su éxito es la sincronización multimedia. En primer lugar, se van a necesitar mecanismos de *sincronización intra-flujo* para que la reproducción de cada componente multimedia (audio, video, texto…) sea fluida y natural, a pesar de los retardos y el jitter. En segundo lugar, se van a necesitar mecanismos de *sincronización inter-flujo* para que, en cada receptor, todos los componentes multimedia relacionados estén alineados en el tiempo en el momento de su reproducción. Por ejemplo, el audio que se escucha debe corresponderse con las escenas que se visualizan. En tercer lugar, van a ser necesarios mecanismos de Sincronización Multimedia Inter-Destinatario (IDMS, *Inter-Destination Media Synchronization*) para que los procesos de reproducción de todos los receptores estén sincronizados y, por tanto, todos los usuarios en la sesión compartida perciban al mismo tiempo los mismos eventos. En ausencia de IDMS, las interacciones entre los usuarios serían incoherentes y podrían ocurrir situaciones frustrantes, como sería el caso de enterarse de un gol al leer o escuchar las celebraciones de los amigos a través de un canal de chat, antes de verlo o escucharlo a través del contenido multimedia consumido de forma compartida (véase la Figura 1). Esto conllevaría probablemente a que el usuario afectado abandonase la sesión multimedia compartida.

Figura 1. Impacto de la variabilidad de retardos en experiencias multimedia compartidas.

El estudio en [3] refleja la relevancia actual y creciente de este tipo de experiencias multimedia compartidas y la necesidad de mejores soluciones tecnológicas para poder proporcionarlas. En concreto, se preguntó a más de 1000 usuarios sobre sus hábitos de uso, necesidades, preferencias y expectativas en este tipo de escenarios. Un 21% de los usuarios afirmó haber participado previamente en este tipo de experiencias, aunque más de un 71% mostró interés en las mismas. Un 69% opinó que este tipo de experiencias proporciona una sensación de compañerismo (*togetherness*), a pesar de las barreras geográficas. De hecho, un 31% comentó que el hecho de poder compartir experiencias de TV con amigos y/o familiares es muchas veces el motivo por el que ven la televisión. Con respecto a las limitaciones existentes, un 32% afirmó que, según su experiencia, los retardos y las diferencias de los mismos entre usuarios en este tipo de escenarios son perceptibles, mientras que un 59% opinó que realmente son una barrera que impide disfrutar de manera satisfactoria este tipo de experiencias. En consecuencia, un 66% opinó que es interesante y necesario el diseño e implantación de mejores soluciones tecnológicas para que dichas experiencias multimedia compartidas sean naturales y satisfactorias.

En este contexto, este artículo presenta una primera versión de Wersync, una plataforma Web 2.0 adaptativa y escalable que proporciona sincronización multimedia distribuida precisa e interacción social entre diferentes grupos de usuarios remotos. La interacción

social se proporciona mediante la compartición de los controles de navegación (p.ej., play, pause, cambios de posición del vídeo…) y un chat de texto (se proporcionarán canales de chat audiovisuales en una versión futura). Además, Wersync proporciona dos mecanismos de presencia para estimular la participación de usuarios externos. El primero de ellos es un menú interno con listas que indican las sesiones activas, sus miembros y el clip que está siendo visualizado. El segundo de ellos es el envío automatizado de tweets, incluyendo la información apropiada (p.ej. hashtags, nicks, clip siendo visualizado, URLs…), cada vez que un participante se une a una sesión en Wersync.

Con respecto a su aplicabilidad, Wersync no está pensada únicamente para fines de entretenimiento, como es la TV Social, sino también para otros escenarios distribuidos relevantes, como el aprendizaje a distancia en tiempo real o el tele-trabajo.

Nuestro objetivo no ha sido únicamente desarrollar otra plataforma de consumo distribuido y sincronizado de contenidos multimedia, sino que nuestra plataforma mejore significativamente a las demás plataformas existentes (resumidas en la Sección 2), y así pueda tener muy relevante en el paradigma actual de consumo de contenidos multimedia.

Una ventaja clave de Wersync es el uso exclusivo de tecnologías web estándar, como HTML5 y Javascript. Por un lado, esto garantiza soporte multi-red, multi-plataforma y multi-dispositivo, además de un despliegue e implantación ubicuo/a. Por otro lado, el uso de HTML5 permite seleccionar el formato más apropiado (p.ej., códec, resolución…) de los contenidos multimedia, en base a las capacidades de los dispositivos y/o a las condiciones de red. Además, tan sólo se necesita un navegador para utilizar Wersync, sin necesidad de instalar ningún software ni hardware de terceros. Otra característica diferenciadora de Wersync es que proporciona mecanismos de IDMS precisos para diferentes grupos de usuarios, combinándolos con mecanismos de interacción social, presencia y privacidad. Ninguna otra plataforma existente proporciona todas estas funcionalidades.

Aunque el diseño de Wersync no esté finalizado, los componentes clave para conseguir las funcionalidades de consumo multimedia, sincronización e interacción social ya están disponibles, y se presentan en este artículo. Las pruebas de evaluación demuestran el rendimiento satisfactorio de Wersync. Además, se añade un enlace a un video que demuestra sus funcionalidades. Finalmente, se anuncian algunas extensiones futuras.

Estructura del Texto

La estructura de este artículo es la siguiente. En la Sección 2 se presentan algunos trabajos relacionados. Los componentes de Wersync y sus funcionalidades se describen en la Sección 3. En la Sección 4 se presentan algunos resultados de evaluación. Finalmente, en la Sección 5 se presentan algunas líneas de trabajo futuro.

Trabajos Relacionados

Hasta el momento, se han propuesto numerosas soluciones de sincronización multimedia inter-flujo e IDMS [4]. En esta sección, se revisan las soluciones de IDMS y las plataformas multimedia relacionadas existentes, enfatizando las ventajas de Wersync.

En [2] se presentó una plataforma para el consumo distribuido y sincronizado de contenidos multimedia. IDMS se consigue mediante la sincronización de relojes de todas las entidades involucradas y adoptando un esquema de control centralizado, en el que se envían mensajes periódicos para estimar el mayor retardo de red y aplicarlo a todos los receptores. Dicha plataforma también permite compartir los controles de navegación e integra canales de chat basados en texto y voz. Las pruebas de evaluación realizadas mostraron que los errores de sincronización en dicha plataforma son del orden de 150ms en entornos LAN y de 300ms en entornos WAN. Asimismo, dicha plataforma se utilizó en [5] para determinar los niveles de diferencias entre tiempos de reproducción (es decir, asincronías) que son tolerables para los usuarios en un entorno Social TV. Se concluyó que

asincronías del orden de 1s ya pueden percibirse, mientras que asincronías del orden de 2s ya son molestas para la mayoría de usuarios, independientemente de si utilizan texto o voz como canal de chat. Es por ello que uno de los objetivos de este trabajo ha sido diseñar una plataforma basada en tecnologías web (con la ventajas que ello supone), que proporcione mejores prestaciones de sincronización, pero que también posibilite seleccionar el clip a visualizar, crear y unirse a sesiones, y proporcione mecanismos de presencia y privacidad.

En [6] se presentó una plataforma web que permite la reproducción sincronizada de fotos y video clips entre usuarios remotos. Dicha plataforma incluye mecanismos de integración con Facebook y con una herramienta de audio conferencia. De esta manera, se estimula la interacción entre los usuarios y la sensación de (co-)presencia. IDMS se consigue adoptando un esquema de control centralizado, incluyendo una base de datos que es accedida periódicamente por los receptores para solicitar o actualizar la información sobre sincronización. Sin embargo, a pesar que los autores de dicha plataforma reconocen la relevancia de IDMS para proporcionar experiencias multimedia compartidas satisfactorias, en dicha plataforma pueden ocurrir errores de sincronización de varios segundos, lo que es inaceptable en aplicaciones multimedia distribuidas en tiempo real, como la TV Social o multi-conferencia. Como ventajas, Wersync está basada en el uso de componentes web más escalables y apropiados para las comunicaciones interactivas, así como permite conseguir un rendimiento mucho mejor con respecto a IDMS.

En [7] se presentó una herramienta de video conferencia en grupo que integra un plugin para el reproductor AMBULANT SMIL (Synchronized Multimedia Integration Language) capaz de proporcionar IDMS. La funcionalidad IDMS se consigue a través de la sincronización de los relojes de cada instancia del reproductor SMIL en la sesión compartida. Dicha plataforma también permite compartir los controles de navegación. La ventaja de Wersync es que está basada en componentes web estándar y no se necesita la instalación de ningún software ni plugin, por lo que puede utilizarse en cualquier sistema operativo y dispositivo. Además, los mecanismos de streaming utilizados en las plataformas en [2] y en [7] pueden comportar problemas con firewalls y NAT, aspectos que quedan resueltos con Wersync. Por último, Wersync no depende exclusivamente de la sincronización de los relojes de los receptores para conseguir IDMS, sino que puede adoptar otros mecanismos alternativos, tal y como se describirá más adelante.

En [8] se examinó la idoneidad de varios componentes tecnológicos (p.ej., esquemas de control, algoritmos y técnicas de ajuste) cuando se utiliza una solución de IDMS estándar (RFC 7272 [9]). Dicha solución de IDMS fue extendida en [10] para proporcionar sincronización dinámica basada en eventos. Asimismo, en [11] se propuso una solución de IDMS para MPEG DASH (*Dynamic Adaptive Streaming over HTTP*). Esta solución se implementa en los clientes y adopta un esquema de control distribuido para gestionar diferentes grupos de usuarios y para negociar la referencia temporal a la que ajustarse. Las soluciones de IDMS propuestas en [8], [10] y [11] son muy prometedoras, pero se han centrado exclusivamente en el diseño y evaluación de componentes para conseguir IDMS. Wersync adopta los mecanismos más adecuados de dichas soluciones para el tipo de escenarios bajo estudio, mejorando algunos de ellos. Además, a diferencia de dichos trabajos, Wersync proporciona funcionalidades para la selección del contenido multimedia a visualizar, gestión de sesiones, interacción social, presencia y privacidad. Por otra parte, su evaluación no se realiza en entornos simulados, sino en entornos reales heterogéneos.

Finalmente, se citan otras dos plataformas que proporcionan consumo distribuido de contenidos multimedia. Por un lado, *Yahoo! Zync* [12] permite compartir videos indicando su URL en la herramienta de chat de *Yahoo! Messenger*. Por otro lado, *Watchitoo* [13] es otra aplicación web que proporciona servicios de chat de texto, así como de audio y video conferencia, cuando varios usuarios consumen el mismo contenido multimedia. Sin embargo, estas herramientas se basan esencialmente en compartir los controles de navegación, pero no proporcionan mecanismos de sincronización continuos y precisos.

Plataforma Wersync

En esta sección se presentan y describen los componentes tecnológicos utilizados para diseñar e implementar Wersync, así como las funcionalidades proporcionadas.

Tecnologías Web Utilizadas

Wersync se ha desarrollado mediante el uso exclusivo de tecnologías web estándar, como HTML5 y Javascript, lo que garantiza soporte multi-red, multi-dispositivo, multi-plataforma y multi-navegador. En concreto, se han utilizado cuatro componentes tecnológicos principales para conseguir las funcionalidades buscadas. El primero de ellos es el elemento *video* de HTML5, que permite insertar videos en páginas web, especificando su dirección y su formato (p.ej., códec, resolución…). El segundo componente es *Node.js*, un entorno de desarrollo de código abierto, multi-plataforma, desarrollado en Javascript, para aplicaciones web cliente-servidor. *Node.js* proporciona un modelo de comunicación bidireccional basado en eventos muy apropiado para desarrollar aplicaciones distribuidas eficientes y escalables. El tercer componente es *Socket.IO*, una librería Javascript que permite comunicaciones bidireccionales basadas en eventos entre clientes y un servidor web (*Node.js*). Mediante el uso de *Socket.IO*, se pueden enviar diferentes tipos de mensajes, con diferentes tipos de datos, a través de un canal de comunicaciones único. El cuarto componente consiste en un mecanismo de sincronización de relojes para asegurar que todas las entidades de la sesión compartidas dispongan bien de una base de tiempos común o, al menos, de una noción coherente del tiempo (como se explica a continuación). Mediante el uso combinado de estos componentes tecnológicos, Wersync es capaz de proporcionar las funcionalidades deseadas (descritas en las sub-secciones posteriores).

Una visión general de un cliente Wersync y de las funcionalidades proporcionadas se puede ver en la Figura 2, mientras que un diagrama de las entidades involucradas y de los mensajes intercambiados en Wersync se puede ver en la Figura 3. Aunque para una mejor claridad se hayan representado cuatro servidores diferentes, todos ellos se pueden implementar en la misma entidad.

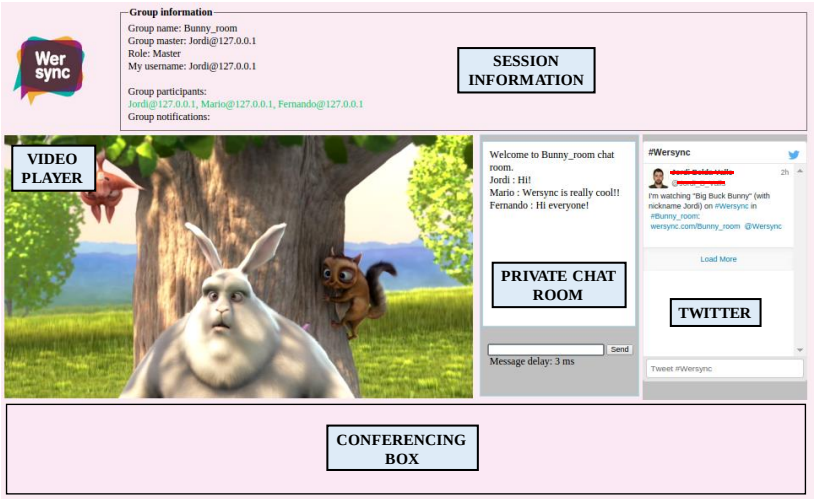

Figura 2. Visión General y Funcionalidades de la Plataforma Wersync.

Sincronización de Relojes

Para conseguir IDMS, se necesita un conocimiento global y coherente del tiempo en la sesión compartida. Se han considerado tres opciones para ello en Wersync, todas ellas basadas en mecanismos de sincronización de relojes. La primera opción consiste en la sincronización de los relojes del sistema de todas las entidades involucradas (p.ej., mediante el uso de clientes *Network Time Protocol* or NTP). Sin embargo, esta opción

podría no ser soportada globalmente en entornos multi-dispositivo y/o multi-plataforma. La segunda opción consiste en la sincronización de los relojes a nivel de aplicación, pero esto podría implicar la instalación de módulos adicionales (p.ej., clientes NTP Javascript). Además, en ambas opciones, podría suceder que no todas las entidades puedan acceder al mismo servidor NTP o que directamente no soporten dicha tecnología. Por ello, como tercera opción, se ha diseñado un mecanismo de sincronización de relojes virtual. Básicamente consiste en utilizar un reloj de referencia (p.ej., el del servidor *Node.js*), e ir midiendo los retardos y las desviaciones entre los relojes involucrados mediante el envío de mensajes bidireccionales periódicos, de una manera similar a como lo hace NTP. Con esto se consigue alinear en el tiempo los relojes de las entidades involucradas, incluso si no utilicen la misma tecnología para la sincronización de relojes.

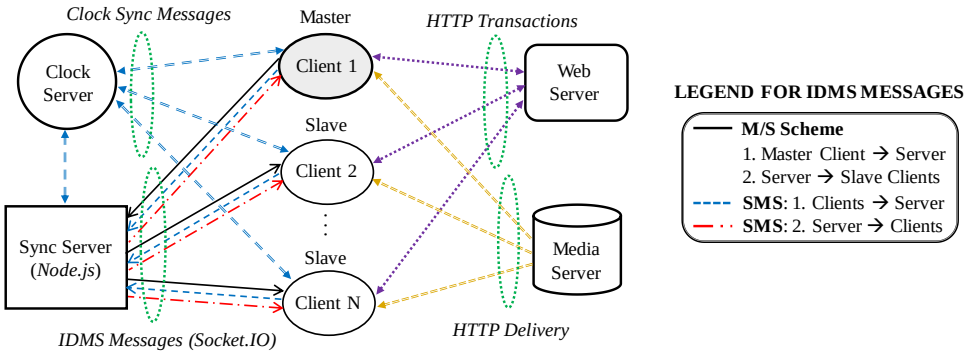

Figure 3. Entidades involucradas y mensajes intercambiados en Wersync.

Protocolo de IDMS

Para conseguir IDMS, clientes específicos envían al servidor de sincronización (colocado junto con el servidor *Node.js*) mensajes de control periódicos (vía el canal *Socket.IO*) incluyendo la temporización de su proceso de reproducción y el tiempo global en el que son enviados. El esquema de control utilizado (descritos en la próxima sub-sección) determinará qué clientes enviarán dichos mensajes IDMS. En todo caso, el servidor re-enviará un único mensaje IDMS a todos los clientes pertenecientes a un grupo específico (ver Figura 3). Al recibir dicho mensaje, cada cliente calculará la asincronía entre su temporización de reproducción y la incluida en el mensaje IDMS, teniendo en cuenta el retardo de tránsito para dicho mensaje. Esto último es posible gracias a la inserción de las marcas de tiempo para cada mensaje IDMS y al mecanismo de sincronización de relojes implementado, y permite conseguir una sincronización precisa. Como resultado, si la asincronía supera un umbral configurado, el cliente deberá ajustar su proceso de reproducción para conseguir IDMS.

Además, Wersync permite compartir los controles de navegación entre todos los clientes, añadiendo también marcas de tiempo de envío en dichos mensajes para conseguir mejor precisión. Con esto se consiguen sesiones interactivas. Por ejemplo, se puede pausar la reproducción del video en todos los clientes para comentar una escena específica, o se puede repetir la visualización de una escena si algo no ha quedada claro.

Esquemas de Control para IDMS

Wersync puede adoptar dos esquemas de control centralizados para conseguir IDMS: 1) el esquema *Master/Slave (M/S)*; y 2) el esquema *Sync Maestro Scheme (SMS)*. Un análisis exhaustivo sobre su idoneidad para IDMS se puede encontrar en [1, 8]. No se ha considerado un esquema de control distribuido porque su uso implica mayor complejidad y capacidad de proceso para los clientes [1, 8], lo que no es conveniente en entornos web.

Si se utiliza el esquema M/S, uno de los clientes será el maestro o administrador, y los demás serán los esclavos. Solo el cliente maestro enviará mensajes IDMS al resto de clientes esclavos, a través del servidor *Node.js* (véase la Figura 3). De esta manera, cada cliente esclavo ajustará su proceso de reproducción para sincronizarse con el maestro.

El cliente maestro se puede seleccionar de manera arbitraria o siguiendo criterios de prioridad específicos (p.ej., podría ser el profesor en un escenario e-learning). En Wersync, por defecto, el (primer) cliente maestro de cada sesión será su creador. Solo el maestro tendrá habilitados los controles de navegación del reproductor para evitar situaciones caóticas. Sin embargo, se ha añadido un mecanismo de conmutación M/S bajo demanda. De esta manera, si un cliente esclavo quiere convertirse en maestro, y así tomar el control de la sesión, puede enviar una petición al maestro (presionando un botón e indicando brevemente el motivo en un cuadro de texto), quien podrá aceptar o no dicha petición.

Un problema típico del esquema M/S es que si el maestro abandona la sesión repentinamente, se pierde el control de sincronización (punto único de fallo). Para evitar esto, se han diseñado las siguientes estrategias de re-elección de un nuevo maestro: i) el nuevo maestro se asignará arbitrariamente; ii) el último maestro (si lo hubo), será el nuevo maestro; y iii) el participante de chat más activo será el nuevo maestro. Hasta donde sabemos, estas estrategias no se han considerado en ningún trabajo previo.

Cuando se utiliza el esquema SMS, habrá un servidor de sincronización (que puede implementarse junto con el servidor *Node.js*, con uno de los clientes o ser una entidad independiente) al que todos los clientes enviarán mensajes IDMS (véase la Figura 3). En este caso, a diferencia de cuando se utiliza el esquema M/S en el que el servidor *Node.js* sólo tiene que reenviar los mensajes IDMS del cliente maestro, el servidor también deberá procesar dichos mensajes con tal de evitar un envío masivo a los clientes. Así pues, el servidor recolectará los mensajes IDMS de todos los clientes y calculará la asincronía entre ellos. Si la asincronía supera el umbral, el servidor enviará un nuevo (único) mensaje IDMS, incluyendo los ajustes de reproducción necesarios, a todos los clientes. Así pues, una decisión clave consiste en determinar la referencia temporal a la que sincronizarse. Varias políticas dinámicas se proponen en [8], que consisten en seleccionar como la referencia: i) al cliente más rezagado; ii) al cliente más adelantado; iii) al punto de reproducción medio entre todos los clientes; y iv) a un cliente virtual con una temporización de reproducción ideal (es decir, sin desviaciones). Estas estrategias se han adoptado en Wersync pero, además, se han añadido otras, como seleccionar al participante de chat más activo o a un líder (p.ej., el profesor en e-learning) como la referencia para IDMS. Al recibir el mensaje IDMS, cada cliente deberá ajustar su proceso de reproducción, como en el esquema M/S.

En ambos esquemas de control, el periodo de envío de los mensajes IDMS se adapta dinámicamente en función del número de clientes y de grupos, con tal de no saturar los recursos de red y computacionales, contribuyendo así a una mayor escalabilidad.

IDMS basada en Grupos

Wersync permite la creación de diferentes grupos de usuarios (es decir, sesiones compartidas), que podrán consumir el mismo o diferente contenido multimedia, con procesos de sincronización y con canales de chat independientes. Al entrar en Wersync, un usuario podrá decidir si crear un nuevo grupo o solicitar unirse a un grupo existente.

Técnicas de Ajuste de Reproducción

Cada vez que un cliente recibe un mensaje IDMS, calculará la asincronía entre su proceso de reproducción y la referencia temporal incluida en dicho mensaje (compensando el retardo de tránsito para dicho mensaje). Si la asincronía supera un umbral pre-establecido, se deberán ejecutar técnicas reactivas de ajuste del proceso de

reproducción para conseguir IDMS. Dos tipos de técnicas se han considerado en Wersync. La primera de ellas consiste en ejecutar ajustes agresivos, es decir saltos o pauses, hasta sincronizarse. La segunda de ellas consiste en ajustar de manera suavizada la tasa de reproducción, es decir acelerando o ralentizando, hasta sincronizarse. Este tipo de técnicas se conoce comúnmente como *Adaptive Media Playout* (AMP). Estudios previos han demostrado que saltos y pausas pueden llegar a ser muy molestos para los usuarios, resultando en una calidad de experiencia (*Quality of Experience*, QoE) pobre [14, 15]. Sin embargo, ningún trabajo previo ha comparado su impacto sobre la QoE, por lo que se han añadido ambas técnicas para investigarlo en un trabajo futuro. Además, los saltos y pausas siguen siendo necesarios para cambiar la posición o pausar la reproducción del vídeo.

Chat de Texto Sincronizado

Se ha implementado un chat de texto sincronizado, utilizando el canal *Socket.IO* y el mecanismo de sincronización de relojes. Cada uno de los mensajes de texto incluye una marca de tiempo de envío, que será utilizada para alinear en el tiempo dicho mensaje con la posición de video correspondiente en los receptores (sincronización inter-flujo). También se consideró utilizar Twitter como la herramienta de chat. Sin embargo, el uso de una herramienta de chat personalizada proporciona: i) mayor interactividad (es decir, menores retardos); ii) mayor flexibilidad para añadir e interpretar marcas de tiempo; y iii) canales de chat “privados”, en vez de una canal de chat “público” y abierto cuando se utiliza Twitter.

Mecanismos de Presencia

Wersync persigue la interacción social entre usuarios. Para ello, se deben proporcionar mecanismos para informar, en todo momento, sobre los miembros activos y los contenidos multimedia siendo visualizados, lo que se conoce como “presencia social” [6]. Dos mecanismos se han añadido para ello. En primer lugar, las lista de sesiones activas, sus miembros, el *nick* del administrador, y una breve descripción del contenido siendo visualizado, se puede comprobar a través de un menú interno con listas desplegables. En segundo lugar, se ha añadido un mecanismo externo de presencia integrando Twitter (utilizando su API Javascript) con Wersync. De esta manera, cada vez que un usuario crea o se une a una sesión, si está registrado en Twitter, se enviará un tweet informando sobre ello (si se desea). Este tweet incluirá los hastags necesarios para identificar unívocamente la sesión (p.ej., *#Wersync*, *#user_nick*, *#session_id*...), una descripción del clip siendo visualizado y una URL para acceder a dicha sesión (véase la Figura 2). Este mecanismo permitirá que usuarios externos conozcan la actividad de sus contactos de Twitter en Wersync, lo que sin duda contribuirá a estimular su participación en sesiones activas.

Aspectos de Privacidad

A pesar de las notificaciones vía Twitter, se puede restringir los usuarios que pueden unirse a cada sesión compartida en Wersync. Cuando un nuevo usuario solicita unirse a una sesión existente, se enviará un mensaje al administrador de dicha sesión, quien podrá aceptar o denegar dicha solicitud. Asimismo, el uso de canales de chat “dedicados” para cada sesión, en vez de utilizar Twitter como canal de chat “público”, contribuye a garantizar la privacidad. Además, si se desea, los mensajes de chat también se pueden encriptar.

Evaluación

El rendimiento de Wersync se ha probado satisfactoriamente, a través de tests iterativos y pruebas de usuarios, para cada una de sus componentes y funcionalidades, utilizando diferentes tipos de dispositivos (PCs, laptops, tablets y smartphones), con diferentes

sistemas operativos (Ubuntu, Windows, MAC y Android) y navegadores (Chrome, Firefox, Internet Explorer y Android), en entornos locales (en y entre los laboratorios de nuestra universidad), regionales (entre ciudades de la misma provincia) y de área amplia (entre ciudades de diferentes provincias y países europeos).

Debido a limitaciones de espacio, solo se proporcionan resultados para un escenario de pequeña escala (entre dos laboratorios de nuestra universidad). En la prueba, cuatro clientes se unieron a una sesión compartida en la que visualizaron 5 minutos de la película "Sintel". El maestro durante toda la sesión fue su creador. Se utilizó el esquema M/S, NTP para la sincronización de relojes y AMP como técnica de ajuste, limitando la variación máxima de la tasa de reproducción a un 25%, como en [8, 14]. El umbral de asincronía permisible se configuró a 50ms con tal de acotar la asincronía entre cualquier par de clientes por debajo de 100ms, que ya puede ser molesto en algunas aplicaciones [1].

Un video mostrando las funcionalidades de Wersync se puede ver en goo.gl/6NjDRf.

Rendimiento de IDMS

La Figura 4 muestra la evolución temporal de la asincronía entre uno de los clientes esclavos y el maestro. En primer lugar, se puede observar que ambos clientes estuvieren sincronizados al inicio de la sesión, debido al envío de un comando "play" incluyendo marcas de tiempo y que, a continuación, la asincronía se mantuvo moderadamente estable. Sin embargo, se ejecutaron varias acciones para forzar situaciones de asincronía (sobre el umbral, representado mediante líneas rojas en la figura) y comprobar si éstas eran corregidas por el protocolo IDMS. Primero, se ejecutaron saltos hacia adelante y hacia atrás en el reproductor del cliente maestro. Debido a ello, se detectaron situaciones de asincronía en el cliente esclavo, pero fueron corregidas rápidamente. En concreto, asincronías positivas y negativas (que significan esclavo atrasado/adelantado con respecto al maestro, respectivamente), ocurrieron debido a los saltos hacia adelante y hacia atrás, respectivamente. Además, se forzó otra situación de asincronía saturando los recursos computacionales del cliente esclavo (ejecutando muchas aplicaciones en paralelo), pero también fue corregida rápidamente. Por último, se puede apreciar en la vista ampliada que la tasa de reproducción se ajustó de manera suavizada (acelerando o ralentizando), mediante la técnica AMP, para corregir situaciones de asincronía.

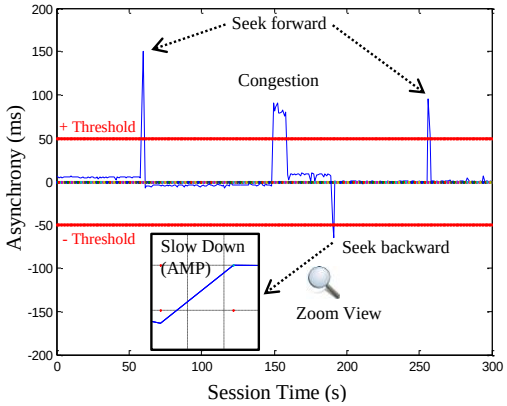

Figura 4. Asincronía utilizando el esquema M/S para IDMS.

Rendimiento de Sincronización Inter-flujo y Retardos de Chat

El rendimiento de la sincronización inter-flujo, entre video y texto se evaluó configurando un bucle en el cliente maestro para enviar un mensaje de chat cada 2s. El retardo de tránsito de dichos mensajes fue bastante uniforme en torno a 5ms (en torno a 40ms en un escenario continental). Estas magnitudes de retardos, que indican el valor de asincronía con respecto a la posición asociada del video, son satisfactorias.

Trabajo Futuro

Varias extensiones hay planificadas para futuras versiones de Wersync. Primero, se extenderá su funcionalidad para permitir el consumo distribuido y sincronizado de aquellos contenidos multimedia para los que los usuarios indiquen su URL, en vez de sólo para aquéllos almacenados en la videoteca del servidor multimedia. Segundo se pretende sincronizar también contenido en vivo. Tercero, se añadirán canales de chat audiovisuales.

Acknowledgement

El presente trabajo ha sido financiado, en parte, por el FEDER y por el MINECO, bajo el programa de apoyo a la I+D+i en el proyecto con referencia TEC2013-45492-R.

References

[1] Montagud, M., et al. 2012. Inter-destination multimedia synchronization: schemes, use cases and standardization. MMSJ, 18(6), 459-482, Nov. 2012.

[2] Vaishnavi, I., Cesar, P., Buterman, D., Friedrich, O., Gunkel, S., Geerts, D. 2011. From IPTV to synchronous shared experiences challenges in design: Distributed media synchronization. Signal Processing: Image Comm., 26(7), 370-377, Aug. 2011.

[3] Boronat, F., Montagud, M., Martínez, M., Marfil, D. 2015. Estudio sobre necesidades, hábitos, preferencias y expectativas de los usuarios con respecto a la sincronización multimedia en escenarios híbridos. JITEL 2015, Mallorca (Spain), Oct. 2015.

[4] Boronat, F., et al. 2009. Multimedia group and inter-stream synchronization techniques: A comparative study. Information Systems, 34(1), 108-131, March 2009.

[5] Geerts, D., et al. 2011. Are we in sync?: synchronization requirements for watching online video together. ACM CHI 2011, Vancouver (Canada), May 2011.

[6] Wijnants, et al. 2012. synchronous MediaSharing: social and communal media consumption for geographically dispersed users. MMSys'12, N.Carolina (USA), Feb. 2012.

[7] Jansen, J., Cesar, P., Bulterman, D. 2013. Multimedia Document Synchronization in a Distributed Social Context. ACM DocEng 2013, Florence (Italy), September 2013.

[8] Montagud, M., Boronat, F., Stokking, H., Cesar, P. 2014. Design, Development and Assessment of Control Schemes for IDMS in a Standardized RTCP-based Solution, COMNET, 70(9), 240-259, Sept. 2014.

[9] van Brandenburg, R., et al. 2014. Inter-destination Media Synchronization using the RTP Control Protocol (RTCP). IETF Standard, RFC 7272, June 2014.

[10] Montagud, M., Boronat, F., Stokking, H. 2013. Early Event-Driven (EED) RTCP Feedback for Rapid IDMS. ACM MM 2013, Barcelona (Spain), Oct. 2013.

[11] Rainer, B., Timmerer, C. 2014. Self-Organized Inter-Destination Multimedia Synchronization for Adaptive Media Streaming, ACM MM'14, Orlando (USA), Nov. 2014.

[12] Yahoo! Zync: http://sandbox.yahoo.com/heres-zync

[13] Watchitoo: http://watchitoo.com/

[14] Su, Y., Yang, Y., Lu, M., Chen, H. 2009. Smooth Control of Adaptive Media Playout for Video Streaming, IEEE TMM, 1(7), 1331-1339, Nov. 2009.

[15] Hossfeld, T., et al. 2011. Quantification of YouTube QoE via Crowdsourcing, IEEE ISM 2011, Dec. 2011.