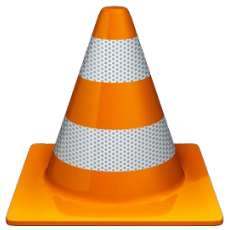
Proyecto de streaming de vídeo



Autor: Alejandro Romero Abadía

Índice

Introducción....3
El servidor Icecast (Streaming en directo)....4
Introducción....4
Instalación de Icecast2....5
Interfaz web de administración....6
Una prueba sencilla....8
El reproductor VLC....10
Introducción....10
Prueba con VLC (modo texto)....10
Prueba con VLC (modo gráfico)....11
Emitir vídeo de una webcam....12
Servidores de Vídeo bajo demanda (VoD)....13
El servidor GNUMP3D....13
Instalación de GNUMP3D....14
Interfaz web de GNUMP3D....14
Configuración de GNUMP3D....15
Instalación de varios servicios....15
Instalación de los servicios....16
Codificador FFMPEG....17
Instalación de FFMPEG....18
Instalación del códec de vídeo VP8 (WebM)....19
Opciones de FFmpeg....21
Streaming con FFserver....23
Ejemplo de streaming con FFserver....24
Usar FFMPEG con Icecast....26
Usar servidor web Apache como servicio de streaming bajo demanda en HTML5....26
El servidor Stream-m (experimental)....27
Servidor Flumotion....29
Instalación....30
Interfaz gráfico....31
Vídeo en directo (live)....34
Ficheros de configuración....42
Vídeo bajo demanda (VOD)....43
Vídeo en directo (live)....45
El reproductor de vídeo de HTML5....59
Referencias....60

Introducción

En este proyecto veremos como montar servicios de vídeo por la red así como algunas de sus posibilidades.

Existen dos formas de transmitir vídeo por la red, tanto local como por Internet, el vídeo en directo y el vídeo bajo demanda. A ambos se les llama comúnmente streaming de vídeo, que quiere decir flujo de vídeo.

El vídeo en directo consiste en emitir un flujo de datos, que pueden ser vídeo, audio o ambos, en tiempo real. Un cliente que esté utilizando un servicio de streaming en directo estará visualizando lo que el servidor emita en ese momento sin posibilidad de volver a visualizar partes anteriores del vídeo.

El vídeo bajo demanda, como su nombre indica, consiste en la emisión del flujo de datos de vídeo y audio pero en el momento en que un cliente lo solicite. Esto implica que para que un vídeo pueda ser servido bajo demanda el vídeo tiene que haber sido grabado anteriormente, por ejemplo un fichero de vídeo.

En ambos casos los clientes, que son reproductores de contenidos de audio y vídeo, utilizan buffers que van almacenando el flujo de datos antes de empezar a reproducirlo. De esta forma se intenta evitar la latencia que puede producirse durante la recepción de los datos.

Otra parte importante de la emisión de datos multimedia por la red es la codificación. El audio o el vídeo deberá ir codificado en un determinado formato para facilitar la emisión de los datos. Según que codificación tenga, podrá reducirse el ancho de banda consumido durante la emisión, y así el servicio será mas eficiente. Sin embargo cuanto mas comprimido esté el flujo de datos, menor será su calidad aunque tendrá menos problemas a la hora de servirlos en la red reduciendo la latencia. En este caso es necesario configurarlo para que haya un equilibrio entre la calidad de imagen o audio, y la forma de transmisión de éste.

Podemos configurar aspectos como el bitrate, o bits por segundo que son enviados a la red, pudiendo manejarlos para administrar la carga de la red. Cuanto más bits por segundo, más carga para la red, lo cual dificultaría la reproducción. Sin embargo, si el bitrate es muy bajos, puede ocasionar errores en la imagen, como una imagen pixelada, aunque el flujo se envíe correctamente.

En este proyecto veremos varios servicios de streaming, donde se instalarán en un equipo con Linux Debian Squeeze (Debian 6) de 64 bits.

El servidor Icecast (Streaming en directo)

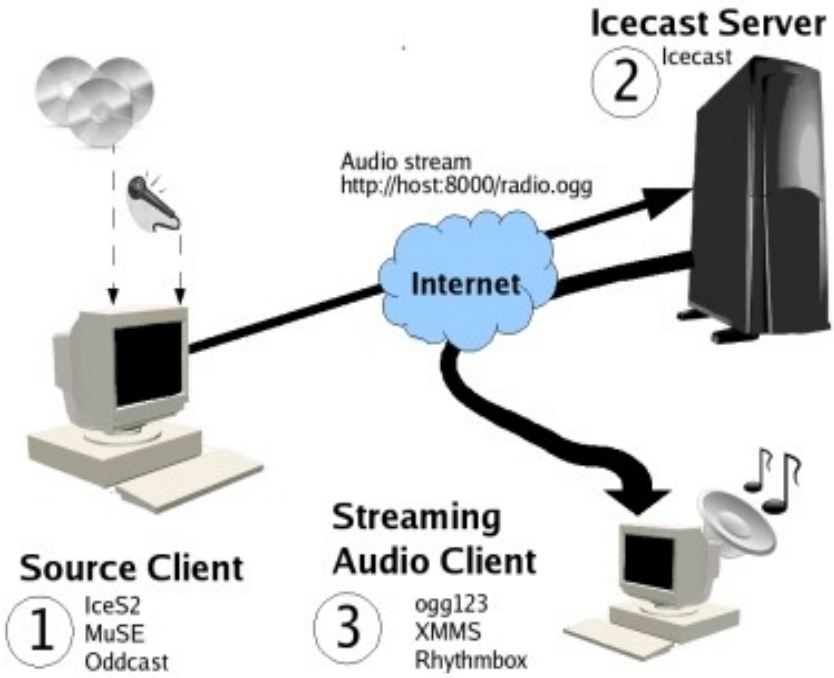
Introducción

Los servidores de streaming de vídeo y audio suelen trabajar de la misma forma, una fuente emisora de datos, que pueden ser vídeo o audio, envía el flujo de datos hacia un servidor de streaming, que se encarga de distribuir ese flujo de datos entre los clientes. Los clientes pueden ser reproductores multimedia instalado en el sistema operativo o en la web. El servidor de streaming que vamos a usar es Icecast, en su versión 2, que distribuirá el flujo de datos a los clientes que lo soliciten.

Éste servidor es un proyecto de software libre mantenido por Xiph.org. Es capaz de distribuir contenido tanto de audio como de vídeo, aunque al principio fue exclusivamente de audio, siendo muy utilizado para crear radios en Internet.

Actualmente Icecast soporta los streams Ogg Vorbis, MP3, Ogg Speex, Ogg FLAC, Ogg Theora y AAC. Podemos ver mas detalles del proyecto y su documentación en su sitio oficial.

Los servidores de streaming distribuyen el flujo en puntos de montaje, que son configurados en el mismo fichero de configuración.



La fuente emisora de datos puede ser otro reproductor multimedia, una tarjeta de TV, una webcam, etc.

Instalación de Icecast2

En esta pequeña práctica vamos a usar un programa que tomará los datos ofrecidos por los dispositivos y se los enviará al servidor de streaming Icecast2. Para empezar instalamos Icecast2:

#aptitude install icecast2

Una vez instalado, accedemos al fichero de configuración de Icecast2, que se encuentra en /etc/icecast2/icecast.xml. Nos encontramos con un fichero XML con multitud de opciones para configurar Icecast. Se comentan algunas interesantes:

<limits></limits>

Entre estas dos etiquetas podemos configurar una serie de límites que podemos establecer en nuestro servidor:

<clients>nº_clientes</clients>

Aquí definimos el número máximo de clientes que van a usar el servidor para descargar el contenido multimedia.

<sources>nº_de_conexiones</sources>

Se define el número máximo de fuentes. Son el máximo número de flujos de datos desde el origen (desde un dispositivo hacia Icecast).

<header-timeout>segundos</header-timeout>

Aquí se indica el tiempo máximo en segundos que un cliente debe esperar para poder conectarse al servidor.

<authentication></authentication>

Usuarios y contraseñas para la configuración del servidor. También se configuran usuario y contraseña para iniciar las conexiones con las fuentes de datos (para que se puedan crear las conexiones entre los dispositivos y el servidor).

<source-password>contraseña</source-password>

Aquí se indica la contraseña que deberán usar las fuentes de datos para enviar el flujo de datos al servidor. “source” es el nombre de usuario. Para mandar los datos al servidor, se debe usar el usuario “source” y la contraseña que se indique aquí.

<relay-password>contraseña</relay-password>

Los servidores Icecast pueden enviar los datos a otro servidor Icecast, actuando este como maestro o relay. Esta opción se define en el servidor maestro, donde se establece la contraseña que deberán usar los servidores esclavos. Como usuario se usa “relay”.

<admin-user>usuario</admin-user>
Usuario administrador. Se puede usar en el interfaz web.
<admin-password>contraseña</admin-password>
Contraseña del administrador. Se puede usar en la interfaz web.

<hostname>servidor</hostname>
Nombre o IP del servidor.

<port>nº_puerto</port>
Puerto usado para que los usuarios puedan descargar el flujo de datos, por ejemplo, un reproductor multimedia.

<mount>
<mount-name>nombre_del_punto_de_montaje</mount-name>
</mount>
Aquí se establece un punto de montaje donde se va a volcar el flujo de datos desde la fuente.

Existen más opciones para incluir dentro del bloque <mount>. Estas opciones serían solo para este punto de acceso, a diferencia de las opciones de fuera del bloque que serían globales. Algunas aquí:

<username>nombre</username>
Nombre de usuario para este punto de montaje específico.

<password>pass</password>
Contraseña para este punto de montaje específico.

<max-listeners>nº</max-listeners>
Número de usuarios para acceder al punto de montaje.

Se pueden ver más opciones aquí.

Es importante activar un parámentro en /etc/default/icecast2. El parámetro a modificar es "ENABLE" y lo tenemos que activar para que el servidor pueda ejecutar el script de inicio. Lo dejamos así:

ENABLE=true

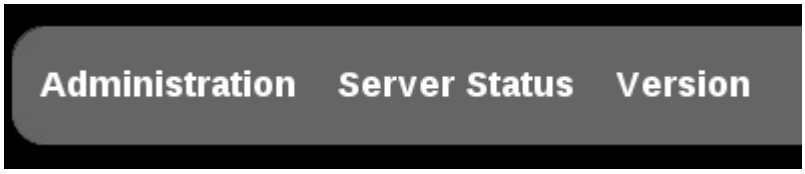
Existen muchas mas opciones para configurar el servidor a las necesidades del administrador.

Interfaz web de administración

Icecast posee una interfaz web de administración donde podemos ver estadísticas del uso del servidor por clientes, los puntos de montaje que se están retransmitiendo, eliminar una conexión de un cliente concreto o un flujo de datos concreto. Para entrar, dependerá de como tengamos configurado el fichero de configuración. En él tendremos que indicar un nombre de usuario y

contraseña para administrador. Eso se indica entre las etiquetas <admin-user> y <admin-password>, siendo la primera para el nombre del usuario y la segunda para la contraseña.

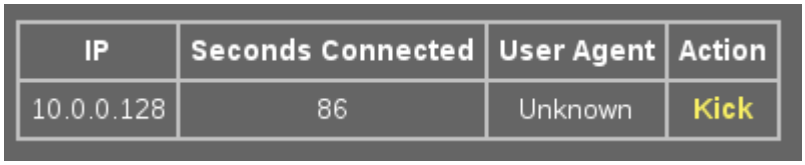
Para entrar vía web al interfaz de administración, escribimos en el navegador la URL referente a la dirección IP del servidor y el puerto que tengamos configurado en Icecast (direcciónIP:puerto). En este ejemplo, usamos 10.0.0.1:8000, siendo el puerto 8000 el usado por defecto del servidor. Accederemos a una pantalla con estas opciones:



Al pulsar en alguno de los enlaces nos pedirá el usuario y contraseña que indicamos en el fichero de configuración (<admin-user> y <admin-password>). Al entrar, en "Administration" nos aparecerá unas estadísticas globales del servidor, indicando por ejemplo el número de clientes, que serían los flujos de datos en uso, los usuarios que están usando el servicio (listeners), dirección IP del servidor, etc, además de mostrar otras estadísticas de los puntos de montaje en uso.

Podemos ver un vídeo que está en un punto de montaje desde el navegador poniendo la URL del punto de montaje directamente en el navegador, como por ejemplo http://10.0.0.1:8000/prueba.ogg.

Por cada punto de montaje en uso encontraremos unas opciones útiles como "List Clients" que lista los usuarios que se encuentran utilizando el punto de montaje. Desde ahí, podemos "echar" a un usuario pulsando en "Kick".



IP	Seconds Connected	User Agent	Action
10.0.0.128	86	Unknown	Kick

Podemos ver algunos datos de los usuarios conectados como su IP o los segundos que lleva conectado.

En “Move mountpoints” tenemos la posibilidad de pasar los usuarios que estén conectador en un punto de montaje a otro punto de montaje.

“Update Metadata” actualiza los metadatos del stream.
Por último “Kill source” desactiva el punto de montaje, haciendo que los usuarios que lo estén utilizando dejen de usarlo.

Una prueba sencilla

Primero realizamos una pequeña práctica sencilla, usando los valores por defecto que trae Icecast. Lo único que se añadirá al fichero de configuración será el punto de montaje. Añadimos lo siguiente:

```
<mount>
        <mount-name>prueba.ogg</mount-name>
</mount>
```

El punto de montaje se llamará prueba.ogg.

No nos olvidemos de activar el servicio en /etc/default/icecast2.

Reiniciamos el servicio con:

#/etc/init.d/icecast2 restart

Ahora instalamos un par de aplicaciones, que son ffmpeg2theora y oggfwd:

#aptitude install ffmpeg2theora

#aptitude install oggfwd

Una vez instalados, y con un vídeo de prueba alojado en el mismo equipo donde está el Icecast, ejecutamos este comando:

ffmpeg2theora prueba.avi -o /dev/stdout | oggfwd localhost 8000 hackme /prueba.ogg

Con esto, ffmpeg2theora reproduce el vídeo, lo convierte a formato theora y lo envía al dispositivo estándar /dev/stdout. El programa oggfwd toma el flujo de datos de /dev/stdout, lo envía a localhost por el puerto 8000, con la contraseña “hackme” al punto de montaje prueba.ogg. El pueto 8000 está establecido en el fichero de configuración de Icecast, así como la contraseña “hackme” del usuario “source” y el punto de montaje prueba.ogg.

Ahora podemos reproducir el stream desde un reproductor cualquiera como por ejemplo el reproductor de películas que viene instalado por defecto en Debian.
Abrimos el programa y seleccionamos Película/Abrir dirección. Después introducimos la URL del punto de montaje del Icecast, que sería http://10.0.0.1:8000/prueba.ogg:

Una vez que se cargue el buffer, podremos ver el vídeo en streaming desde la red local.

El reproductor VLC

Introducción

El reproductor VLC es un reproductor multimedia de código abierto y mantenido por el proyecto VideoLAN. Puede reproducir multitud de codecs de audio y vídeo. Además le han añadido la posibilidad de hacer streaming de vídeo, y además poder enviarlo a un servidor Icecast. Puede ser manejado tanto por interfaz gráfica como por comandos.

Prueba con VLC (modo texto)

Instalamos el paquete vlc:

#aptitude install vlc

Con el servidor Icecast ya configurado, no necesitaremos hacer cambios en el, solo configuraremos la salida del programa VLC. Para lanzar el flujo de datos hacia Icecast desde VLC por línea de comandos hay que indicar una serie de parámetros:

vlc prueba.avi --sout '#transcode{vcodec=theora,vb=800,acodec=vorb,ab=128}:standard {access=shout,mux=ogg,dst=source:hackme@10.0.0.1:8000/prueba.ogg}'

Con este comando indicamos que VLC reproduzca el vídeo prueba.avi con una serie de parámetros:

--sout → Manipulará la salida del vídeo.

Los parámetros de ponen entre comillas simples.

#transcode → Se codificará la salida del vídeo, y se pasará al codec de vídeo theora con tasa de bits a 800, y el audio a vorb (ogg) a 128 bits por segundo.

:standard → Se usará salida standard y se pasarán los siguientes parámetros:

access=shout → Se enviará a un servidor Icecast.

mux=ogg → Se pasará encapsulado con ogg.

dst → Es donde se va a enviar el flujo. En este caso se va a enviar a la dirección 10.0.0.1 en el puerto 8000 al punto de montaje prueba.ogg con el usuario source y contraseña hackme.

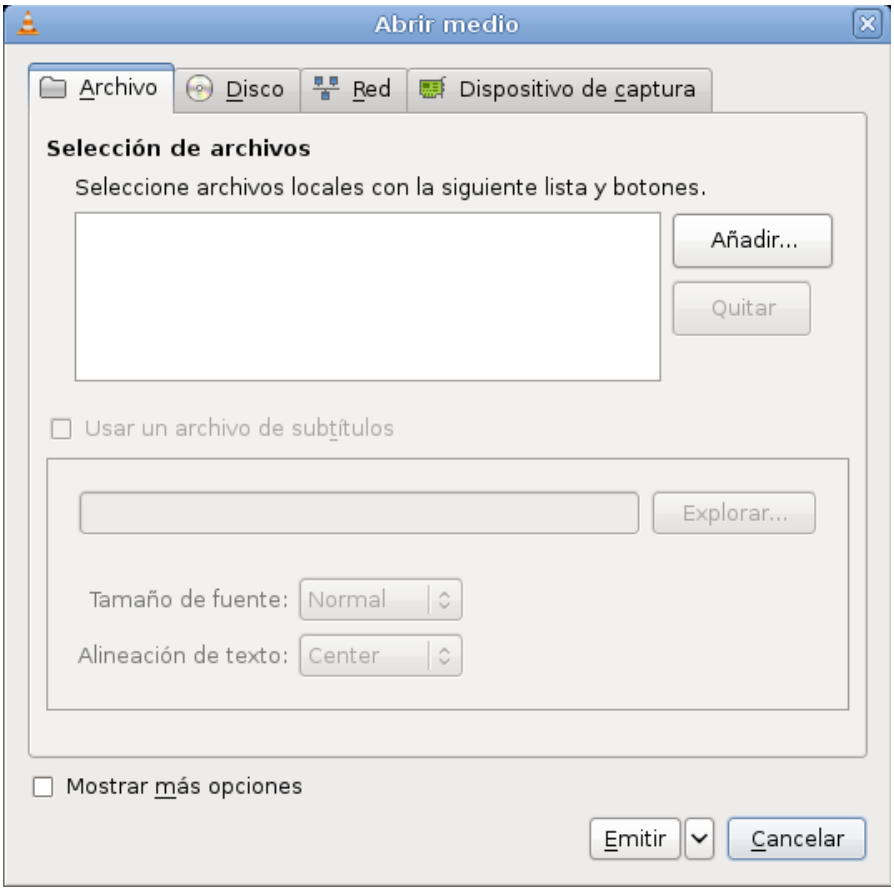
Prueba con VLC (modo gráfico)

También podemos hacer lo mismo desde su interfaz gráfica simplemente abriendo el programa y realizar estos pasos:

Desplegar “Medio” y pulsar sobre “Emitir”.

Podemos seleccionar qué elemento queremos emitir, si desde un archivo, desde le lector DVD, desde otro flujo de datos de la red o desde un dispositivo como por ejemplo una webcam.

En el caso de que sea un archivo, seleccionamos el archivo a reproducir:



Después pulsamos en “Emitir”. Esto nos leva a otra ventana donde tendremos que indicar la fuente, que aparecerá ya indicada, y pulsando en siguiente configuraremos otros aspectos:

Podemos indicar en “Nuevo destino” si lo queremos enviar a otro fichero para que se guarde en disco o al Icecast, entre otras opciones. Seleccionamos Icecast y pulsamos en “Añadir”. Aquí instroducimos los datos del servidor Icecast como su dirección IP, el puerto, el punto de montaje, en este caso prueba.ogg, y el usuario y contraseña, en este caso source:hackme. Es importante que entre el usuario y la contraseña aparezca los dos puntos (:).

Dejamos “Habilitar transcodificar”, que se encargará de realizar la conversión. Esta conversión se indica en “Perfil” donde seleccionaremos “Video – Theora + Vorbis (OGG)”. Pulsamos sobre siguiente.

En la última ventana activamos “Emitir todas las emisiones elementales”. Veremos un cuadro de texto donde podemos ver una cadena de salida con los parámetros establecidos, que podemos modificar.
En cuanto pulsemos sobre Emitir, comenzará la emisión del flujo de datos hacia Icecast.

Ya solo quedaría que los clientes se conectaran al servidor bien por un reproductor multimedia o un reproductor de vídeo en la web.

Emitir vídeo de una webcam

También podemos capturar el vídeo de una webcam USB y enviarlo a Icecast para que lo distribuya. Usando VLC, el comando que tendríamos que ejecutar sería:

```
vlc v4l2:///dev/video0 --sout
'#transcode{vcodec=theora,vb=1000,scale=1,acodec=none}:std{access=shout,mux=ogg,dst=source
:hackme@10.0.0.1:8000/prueba.ogg}'
```

VLC toma el vídeo del dispositivo /dev/video0, que es donde se trata la información de vídeo de una webcam u otro dispositivo de vídeo.
V4l2 (Video4Linux2) es una API de captura de vídeo que VLC usa para extraer el flujo de datos del dispositivo. Se encuentra integrado en el núcleo de Linux.

Existen webcams IP que se conectan directamente a la red. Estas tienen su propia dirección IP y pueden enviar directamente el flujo de datos hacia Icecast, sin tener que usar ningún codificador que envíe el flujo como puede ser VLC.

Una de las opciones mas interesantes del streaming con VLC es la capacidad de duplicar la salida del stream, pudiendo así enviar el flujo a Icecast por un lado, mientras se graba en disco en un fichero de vídeo por otro. Podemos hacerlo con el video procedente de una webcam, se toma el flujo de datos de la webcam, se envía por un lado al servidor Icecast para que los clientes lo reproduzcan, y por otro de graba en disco, que puede ser utilizado por otro servidor de video bajo demanda (VoD) para reproducirlo en cualquier momento.

Ejemplo:

Enviar audio, duplicarlo hacia Icecast y al disco:

```
vlc prueba_musica.mp3 --sout
'#transcode{acodec=vorb,ab=128}:duplicate{dst=std{access=shout,mux=ogg,dst=source:hackme@
192.168.77.184:8000/prueba.ogg},dst=std{access=file,mux=ogg,dst=/home/alejandro/Escritorio/m
usica.ogg}}'
```

Con “duplicate” podemos enviar dos veces la señal hacia dos sitios distintos, poniendo además dos veces la opción de destino (dst).

Para poder ver estos vídeos en el navegador web, podemos incrustar un reproductor incluyendo este código en la web:

```
<embed src="http://192.168.1.180:8000/prueba.ogg" type="video/ogg"
   controller="true" autoplay="true" height="400" width="400">
</embed>
```

En este código le indicamos al navegador que incruste un reproductor de vídeo con formato ogg y que tome el stream en la dirección indicada, que es el punto de montaje prueba.ogg del servidor Icecast.

Servidores de Vídeo bajo demanda (VoD)

Los servidores de vídeo bajo demanda ofrecen servicio de streaming de vídeo o audio, pero a diferencia del streaming en directo, el contenido multimedia puede ser reproducido en cualquier momento, ya que el contenido se encuentra alojado en el servidor. Un claro ejemplo puede ser Youtube. También tiene los mismos problemas que el servicio en directo ya que funciona de la misma forma. Pueden cortarse o que la calidad de imagen sea mala debido a la calidad de la conexión. También se usan buffers durante la reproducción y el usuario puede avanzar, retroceder o pausar el vídeo.

El servidor GNUMP3D

Uno de los servicios que funcionan con VoD es gnump3d. Este servicio se podía instalar desde los repositorios de Debian, pero actualmente hay que descargarse el paquete desde la web oficial. Se puede descargar desde aquí.

Instalación de GNUMP3D

Una vez descargado el paquete tar.gz (desde aquí), lo descomprimimos con tar:

```
tar xvf gnump3d-3.0.tar.gz _
```

Una vez descomprimido el paquete, entramos en el directorio creado y ejecutamos el comando "make install":

```
make install_
```

Ahora que está instalado podremos hacer uso de él. El fichero de configuración se encuentra en "/etc/gnump3d".
Si ejecutamos el comando "gnump3d-top - -manual" veremos algunas de las opciones que dispone GNUMP3D.

Con la configuración por defecto que trae gnump3d, podemos crear un directorio, concretamente /home/mp3, y depositar ahí cualquier fichero de música en mp3. Después ejecutamos el comando "gnump3d-index" para que pueda indexar los ficheros que se encuentren en el directorio. Por último lanzamos el servicio con el comando "gnump3d". Ahora podemos ver una web de administración del servicio introduciendo en el navegador la URL "direccionIP:8888", siendo el puerto 8888 el puerto por defecto de gnump3d. Ahí podemos ver los ficheros mp3 contenidos en el directorio /home/mp3. Este directorio se puede cambiar en el fichero de configuración, en "root".

Interfaz web de GNUMP3D

Desde el interfaz web del servicio podemos ver algunas estadísticas y configuraciones. Accedemos como está descrito anteriormente (direccionIP:8888), y accederemos a una página como esta:

En la pestaña "Music" nos aparecerán los elementos que tengamos en el directorio /home/mp3. También podemos tener ficheros de vídeo, con lo cual también tendremos la posibilidad de reproducir vídeo bajo demanda.
En la pestaña "Browse by Tag" tendremos los ficheros catalogados en diferentes secciones.

La pestaña “Custom Playlist” aparecerán las listas de reproducción personalizadas que tengamos.
La pestaña “Random Selection” consiste en mostrar todos los ficheros de audio que existen en el directorio, incluido los que estén en otros directorios dentro del directorio principal.

La pestaña de “Preferences” consiste en cómo ordenar el contenido, el idioma y el tema del interfaz web.
Las de “Search” sirve para buscar entre los elementos del directorio y “Statistics” son estadísticas del uso del servidor.

Configuración de GNUMP3D

Como muchos servicios de Debian, GNUMP3D también tiene un fichero de configuración para personalizar el servicio a nuestras necesidades, según el uso que le vayamos a dar. El fichero se encuentra en “/etc/gnump3d/gnump3d.conf”.

Algunas de las opciones mas importantes son:

user → Usuario que va a utilizar el servicio.
port → Puerto donde va a escuchar el servicio. Por defecto 8888
root → Es el directorio principal donde se van a depositar los ficheros multimedia.
logfile → Fichero de logs. Por defecto en “/var/log/gnump3d/access.log”.
errorlog → Fichero de logs para errores. Por defecto en “/var/log/gnump3d/error.log”.
allowed_clients → Permitir clientes por su IP. También se pueden permitir rangos IP.
denied_clients → Denegar clientes por su IP. Se pueden poner rangos IP.

Todas estas opciones y el resto tienen en el mismo fichero de configuración una descripción de su uso. También aquí, en la documentación oficial, se explica el uso de las opciones.

Instalación de varios servicios.

Se va a proceder con la instalación de los servicios de forma que una webcam graben contenido en el disco duro y lo envíen a Icecast, que a su vez lo distribuirá en directo. El servidor de VoD tendrá en su directorio principal los ficheros de vídeo grabados por la webcam, así que el servidor enviará el vídeo a los clientes que lo soliciten.

Los objetivos son:

- Establecer una cámara que puede ser USB (que está conectada al servidor) o IP (que VLC tomará el contenido de ella indicando la URL de la cámara).
- VLC tomará el flujo de datos de la cámara y, por un lado, enviará el contenido a Icecast, y por otro, guardará el contenido en disco, concretamente en el directorio principal de GNUMP3D.
- GNUMP3D tendrá los ficheros de vídeo en su directorio principal y servirá el contenido.
- Se podrán visualizar abriendo un reproductor e indicando la URL para reproducir los vídeos, o desde la web incrustando un reproductor.

Instalación de los servicios

Instalaremos los servicios, empezando por Icecast:

#aptitude install icecast2

Instalamos el reproductor VLC:

#aptitude install vlc

Se instalarán multitud de dependencias.

Ahora procedemos con la instalación de GNUMP3D:

Descargamos el paquete aquí. Nos descargamos el “tar.gz”.

Lo descomprimimos:

#tar xvf gnump3d-3.0.tar.gz

Entramos en el directorio y ejecutamos esto:

#make install

Ya tenemos los servicios instalados. Ahora procederemos a configurarlos:

Activamos icecast en "/etc/default/icecast2" en "ENABLE" cambiando false por true.

Creamos primero el punto de montaje en el fichero de configuración de icecast. Para eso editamos "/etc/icecast2/icecast.xml" y añadimos el punto de montaje:

<mount>

<mount-name>cam1.ogg</mount-name>

</mount>

Ahora si queremos podemos personalizarlo a nuestro gusto cambiando el puerto (por defecto 8888), el nombre y password del admin(<admin-user><admin-password>), u otras opciones como las anteriormente vistas en la sección de Icecast.

Configuraremos el dervidor GNUMP3D, editando el fichero "/etc/gnump3d/gnump3d.conf" y modificamos "root" poniendo un directorio dedicado a los videos grabados, por ejemplo, "/videos".

A continuación ejecutamos el comando de VLC apropiado, y usando "duplicate" para que los vídeo se guarden en el directorio que hemos configurado en GNUMP3D, en "/videos".

Codificador FFMPEG

FFMPEG es una serie de herramientas de codificación y streaming de vídeo y audio que forma parte de la comunidad de software libre. El proyecto contiene herramientas como pueden ser el codificador *ffmpeg*, el servidor de streaming *ffserver*, el reproductor *ffplay*, la biblioteca *libavcodec* que contiene los códecs que ffmpeg puede usar, etc.

La sintaxis de los comandos tiene esta estructura:

ffmpeg [opciones] [opciones de entrada] [entrada] … [opciones de salida] [salida] …

Un ejemplo del comando puede ser este:

ffmpeg -i /home/usuario/Escritorio/video.avi /home/usuario/Escritorio/video2.ogg

Este comando convierte un vídeo con extensión .avi a otro con extensión .ogg.

ffmpeg -f video4linux2 -s 320x240 -r 24 -i /dev/video0 /home/usuario/Escritorio/prueba.avi

Este comando se encarga de capturar vídeo de una webcam USB y guardarlo en la ruta indicada.

Instalación de FFMPEG

FFmpeg puede instalarse en Debian bien desde los repositorios o bien compilando el código que ofrece el proyecto en su web.

- Para instalarlo desde los repositorios solo basta ejecutar el siguiente comando en un terminal:

aptitude install ffmpeg

Nota: De esta forma se instalará la versión estable para Debian, la r0.5.5-4, que es del 2009. No es la versión mas actualizada.

- Para instalarlo desde el código fuente debemos seguir las indicaciones que ofrecen en la web del proyecto, aquí. Se puede descargar la última versión usando *git*, o descargando paquetes de versiones estables (FFmpeg Releases). Si deseamos instalar la última versión, primero instalamos git:

aptitude install git

Lo siguiente será dirigirnos a un directorio donde depositaremos el proyecto y ejecutamos *git clone [URL]ffmpeg*. Por ejemplo:

git clone git://git.videolan.org/ffmpeg.git ffmpeg

Entramos en el directorio que se ha creado y procederemos al compilado. Tenemos la oportunidad de añadir opciones a la hora de compilar, como la compatibilidad con theora, vorbis o incluso VP8. Podemos ver las opciones disponibles con el comando *./configure --help*.

Debemos instalar una serie de dependencias para poder instalar ffmpeg con éxito:

- build-essential → Compilador C.
- yasm → Ensamblador.
- libasound2-dev → Librería necesaria para poder usar ALSA en ffmpeg.
- libvorbis-dev → Codec de vídeo Theora.
- libvorbis-dev → Codec de audio Vorbis.
- libvpx → Codec de vídeo VP8 (Para incluirlo, seguir Instalación del codec de vídeo VP8)

FFmpeg lleva instalado internamente multitud de codificadores y demás componentes, sin embargo podemos añadir mas instalando la respectiva dependencia e indicándolo en las opciones del compilado. Por ejemplo podemos también instalar el codec de vídeo VP8.

Instalación del códec de vídeo VP8 (WebM)

El códec de vídeo VP8 fue liberado por Google en 2010 para ser implementado como estándar de vídeo en HTML5, dentro del contenedor WebM, y así rivalizar con el códec H.264.

Para incluirlo en FFmpeg debemos instalarlo antes de la compilación. Existe una versión en los repositorios de Debian, el paquete libvpx, sin embargo a la hora de la compilación de FFmpeg, este no la acepta porque requiere una versión superior, así que descargaremos el código del proyecto de VP8 y lo compilaremos.

Primero, nos dirigimos a la web del proyecto y descargamos el paquete *libvpx-v0.9.7-p1.tar.bz2* que se encuentra aquí.

Una vez descargado, lo llevamos a un directorio limpio y lo descomprimimos:

tar xvf libvpx-v0.9.7-p1.tar.bz2

Entramos en el directorio creado. Podemos usar *./configure --help* si queremos ver las opciones de compilación. En la práctica no indicaremos ninguna, por lo que se instalarán las opciones por defecto. Ejecutamos:

./configure

Lo compilamos:

make -j2

Usamos la opción -j y el número de núcleos de nuestro procesador para que la compilación sea mas rápida.

A continuación lo instalamos con:

make install

Ya tendremos instalado la versión mas actualizada del códec VP8, así que ya podremos incluirlo en FFmpeg añadiéndolo en las opciones de compilación.

Con VP8 instalado, procedemos a configurar la compilación de FFmpeg:

./configure --enable-libtheora --enable-libvorbis --enable-libvpx

Lo compilamos:

make -j2

Y lo instalamos:

make install

- Otra forma de instalar FFmpeg desde el código que proporciona la web oficial es descargando el programa en versiones recientes pero estables. Son las descargas compriidas en "*bzip2 tarball*" o "*gzip tarball*", que están en la sección "*FFmpeg Releases*" de la página de descargas.

Descomprimimos el fichero con el comando *tar*.

El procedimiento de compilado e instalación es el mismo que el que usamos con *git*. Simplemente hay que configurar el compilado, compilarlo e instalarlo:

./configure --enable-libtheora --enable-libvorbis --enable-libvpx

make -j2

make install

Opciones de FFmpeg

Este programa contiene muchas opciones, desde como capturar el origen de los datos, la salida, cambiar de codec, formato contenedor, etc. Aquí algunas de las opciones:

-i → Indica el fichero u origen del flujo de datos. También indica la ruta de salida, dependiendo del orden que se coloque. Por ejemplo, se puede usar para indicar el origen, y después en el mismo comando, indicar la salida. Al indicar la salida es opcional, puesto que se puede indicar la ruta sin poner la opcion -i.

-f → Indica el dispositivo donde se captura el flujo. Puede ser de una webcam, un microfono o una tarjeta capturadora de vídeo. Ejemplo:

ffmpeg -f video4linux2 -s 320x240 -r 25 -i /dev/video1 /home/usuario/Escritorio/prueba.ogg

Este comando usa video4linux2 para capturar la imagen de la webcam, que es el dispositivo /dev/video1.

Es comúnmente usado *video4linux2* para recoger el flujo de datos de las webcams puesto que es una API de captura de vídeo para Linux, que puede usarse también con otros dispositivos de vídeo.

También puede usarse *alsa* (Advanced Linux Sound Architecture) para poder recoger el flujo de datos del micrófono.

ffmpeg -f alsa -i plughw:0,0 /home/usuario/Escritorio/audio.ogg

Este ejemplo consiste en capturar el audio del micrófono usando el componente ALSA. El dispositivo *plughw:0,0* es el dispositivo de captura de audio por defecto del componente ALSA. Para poder capturar el audio es necesario activarlo. Para ello ejecutamos en un terminal *alsamixer,* presionamos en F4 para filtrar por los dispositivos de captura, seleccionamos *capture* con las teclas de dirección y pulsamos la tecla espacio para activarla. Después subimos el volumen con la tecla de dirección arriba.

Otra forma de capturar el audio es con *oss* (Open Sound System), que es el antiguo componente de audio del núcleo de Linux, que hoy en día está obsoleto. Aquí se pueden indicar dispositivos de */dev*.

-s → Establece la resolución de la imagen. Ej: 320x240. También puede indicarse abreviaciones como *vga*, que corresponde 640x480. Una lista de estas abreviaciones puede comprobarse en la documentación oficial.

-async [nº] → Sincroniza el audio con el vídeo. Se indica en segundos y pueden ser números negativos o positivos.

-r [nº] → Establece el número de frames por segundo (fps). Por defecto es 25.

-acodec [nombre códec] → Selecciona el códec de audio a usar.

-vcodec [nombre códec] → Selecciona el códec de vídeo a usar.

Para ver los códecs disponibles, ejecutar "*ffmpeg -formats"*.

Estas son algunas de las opciones más básicas de FFMPEG, aunque existen infinidad de opciones para controlar el flujo.

Streaming con FFserver

Uno de los componentes que se instalan con FFmpeg es el servidor de streaming de audio y vídeo FFserver. Usando FFmpeg como fuente de datos y FFserver como servidor que reciba las peticiones de los clientes, se puede montar un servicio de streaming en directo. Hay que decir que este servidor no está lo suficientemente desarrollado y puede ser inestable a veces. Otra desventaja es el menor número de códecs que soporta, entre ellos mpeg o avi. Podemos encontrar en la web oficial el fichero de configuración por defecto aquí.

La documentación oficial se puede encontrar también aquí.

El funcionamiento de Ffserver es muy simple: recibe una fuente de datos, sea audio, vídeo o ambos, y a partir de la configuración, ofrece los datos con una extensión y codecs determinado a los clientes. Estas fuentes son llamados “feeds”. Cada feed tiene su propia configuración y cada uno puede recibir un flujo de datos distinto.

El fichero de configuración se encuentra en “/etc/ffserver.conf” y desde aquí de puede configurar el servidor. Cada opción viene acompañada por una descripción de su función. Algunas de ellas:

- Port → Establece el puerto de escucha del servidor. Los clientes pueden conectarse a través de este puerto.
- MaxClients → Número máximo de clientes que soportará el servidor.
- <Feed [nombre].ffm> → Nombre del feed. Al usar FFmpeg para enviar el flujo a FFserver abrá que incluir el nombre del feed en la URL:

  ffmpeg -i /home/usuario/Escritorio/video.mpg http://localhost:8090/nombrefeed.ffm

- File [ruta] → Aquí se almacenará temporalmente el feed.
- FileMaxSize [nº] → Tamaño máximo del feed temporal en disco.
- ACL allow [IP] → Direcciones IP aceptadas para reproducir el flujo.
- Stream [nombre].[extensión] → Nombre del stream. Los clientes se conectarán a través de este nombre en una URL:

  http://192.168.1.10:8090/nombrestream.mpg

- Feed [nombrefeed].ffm → Feed que usará el stream. Un stream estará asociado a un feed. De esta forma, FFmpeg enviará el flujo a un feed, este feed se asociará con un stream, y los

clientes se conectarán al stream.

FFmpeg --------> Feed --------> Stream -------->Cliente

- Format [formato contenedor] → Aquí se indica el formato contenedor del stream. En el mismo fichero de configuración se indican los formatos soportados.
- AudioBitRate [nº] → Tasa de bits por segundo. A mayor número mayor es el tráfico en la conexión. Depende del codec usado.
- AudioSampleRate [nº] → Frecuencia del audio. Normalmente es 44100. Depende también del codec usado.
- VideoBitRate [nº] → Tasa de bits de vídeo.
- VideoBufferSize → Tamaño del buffer de vídeo. Con un tamaño inadecuado el flujo puede interrumpirse.
- VideoFrameRate [nº] → Frames por segundo del vídeo. Al ejecutar el comando de FFmpeg, hay que indicar el mismo número con la opción "*-r*". Deben coincidir, si no, mostrará un error.
- VideoSize [nº] → Resolución del vídeo. También puede usarse abreviaturas como "*vga*".
- AudioCodec [nombrecodecaudio] → Nombre del codec de audio que se va a utilizar. Por ejemplo "*libvorbis*" o simplemente "*vorbis*". Si está comentada, FFserver eligirá el codec con referencia al formato contenedor. Por ejemplo, si "*Format*" contiene "*ogg*", FFserver usará vorbis, si contiene "*mpeg*", usará mp2.
- VideoCodec [nombrecodecvideo] → Nombre del codec de vídeo. Al igual que "*AudioCodec*", se indica el codec de vídeo que se usará. También FFserver usará codecs por defecto si está comentada.

Ejemplo de streaming con FFserver

Con FFmpeg instalado, Configuraremos /etc/ffserver.conf. Para este ejemplo se dejarán la mayoría de las opciones por defecto. Solo se modificarán algunas para que funcione:

- Cambiamos "*VideoFrameRate*" por 25, que es el que usaremos con el comando de FFmpeg.
- Configuramos "*VideoSize*" con 320x240.
- Comentamos todas las líneas del ejemplo "*ASF compatible*" para que no envíe dos flujos al mismo stream.

A continuación, ejecutamos en un terminal el suguiente comando para arrancar FFserver con el fichero de configuración:

ffserver -f /etc/ffserver.conf

Ahora FFserver se quedará escuchando. Lo que aparezca en pantalla es el log del servicio.

Ejecutamos en un terminal aparte el comando de FFmpeg. Algo como esto:

ffmpeg -r 25 -s 320x240 -f video4linux2 -i /dev/video0 -f alsa -i plughw:0,0 http://localhost:8090/feed1.ffm

Con esto empezará a recoger la imagen de la webcam USB, que en el sistema está en /dev/video0, y el sonido del micrófono. Podemos reproducir el vídeo con un reproductor que pueda reproducir streaming, por ejemplo VLC, que puede leer multitud de codecs.

Si usamos VLC para reproducir el ejemplo anterior, tenemos que ir a “Medio > Abrir volcado de red” y añadimos la siguiente URL:

http://localhost:8090/test1.mpg

Pulsamos en “Reproducir” y veremos como se carga el buffer de VLC y aparece la imagen de la webcam y el sonido del micrófono.

Usar FFMPEG con Icecast

Aunque FFmpeg no pueda enviar el flujo a Icecast directamente, si que podemos usar oggfwd como intermediario para hacer llegar el vídeo, al igual que con ffmpeg2theora. Para ello, debemos tener instalado Icecast, FFmpeg y oggfwd.

Primero activamos Icecast en el fichero */etc/default/icecast2* modificando la línea de ENABLE en *true*. Despues nos dirigimos a */etc/icecast2/icecast.xml* e incluimos un punto de montaje y con sus opciones. Para hacer la prueba, algo básico como poner solo el nombre del punto de montaje:

<mount>

<mount-name>prueba.ogg</mount-name>

</mount>

Una vez editado el fichero, iniciamos el servicio de Icecast, y lanzamos el siguiente comando de FFmpeg y oggfwd:

ffmpeg -f video4linux2 -r 25 -s 160x124 -i /dev/video0 -f alsa -i plughw:0,0 -f ogg /dev/stdout | oggfwd localhost 8000 hackme /prueba.ogg

Debemos poner la opción *-f ogg* al final para que surja un error de formato. Si el vídeo y el audio no están sincronizados, podemos usar la opción *-async [nº segundos]* para sincronizarlo. Este comando enviará el vídeo en codec Theora y el audio en codec Flac.

Usar servidor web Apache como servicio de streaming bajo demanda en HTML5

El servidor web Apache también incluye la función de servir vídeo de un fichero en HTML5, con el codec WebM por ejemplo. Esto simplifica mucho la tarea a la hora de subir un vídeo a la web. Para realizar esto debemos hacer algunos cambios en un fichero e incluir en la web código en HTML5 para que funcione.

Primero tenemos que modificar el fichero que hace referencia a los tipos mime del sistema, que es donde Apache identifica la extensión de los ficheros. Esto se encuentra en */etc/mime.types*.

Añadimos en este fichero estas dos líneas:

audio/webm weba
video/webm webm

De esta forma Apache podrá identificar los ficheros WebM y ofrecerlo en la web sin problemas.

Debemos tener un fichero en formato WebM para hacer la prueba. Podemos obtenerlo grabando con la webcam y FFmpeg con el siguiente comando:

ffmpeg -f video4linux2 -r 25 -s 360x240 -i /dev/video0 -f alsa -i plughw:0,0 -f webm /var/www/video.webm

Este comando grabará vídeo con el codec VP8 y el audio en Vorbis. Ahora incorporamos el código HTML5 en la web con la etiqueta <*video*>:

<video src=”video.webm” controls></video>

por supuesto, para poder visualizarlo, tenemos que usar un navegador que sea compatible con HTML5, que a día de hoy, son la mayoría.

El servidor Stream-m (experimental)

Stream-m es un servidor de vídeo basado en WebM y que está en un estado experimental, por lo tanto sus cualidades son muy limitadas y sus opciones escasas. Aun así es bastante estable para la versión en la que se encuentra. Al igual que Icecast, este servidor solo se encarga de distribuir el contenido de a los clientes, dejando la codificación a otro programa, por ejemplo a FFmpeg. Este servidor podemos descargarlo de su web aquí. Nos descargamos el fichero comprimido “stream-m-r19-20110927.zip”.

Realicemos un ejemplo:

Una vez descargado el fichero, creamos un directorio para descomprimirlo. Lo hacemos de esta forma:

unzip stream-m-r19-20110927.zip

Se descomprimirán todos los ficheros necesarios para el funcionamiento del servidor. Este servidor, a diferencia de otros, no se instala, sino que se hace funcionar mediante un comando.

Copiamos el fichero *server.conf.sample* y lo pegamos en el mismo directorio con el nombre de *server.conf* para usarlo o hacer modificaciones en el sin modificar el original. Una vez creado el fichero, ejecutamos el siguiente comando:

java -cp lib/stream-m.jar StreamingServer server.conf

Se ejecutará el servidor y mostrará el log del servicio.

A continuación, usaremos FFmpeg para ofrecerle el vídeo al servidor:

ffmpeg -f video4linux2 -r 25 -s 360x240 -i /dev/video0 -f alsa -i plughw:0,0 -f webm http://localhost:8080/publish/first?password=secret

Con esto se enviará al servidor el vídeo de la webcam y el audio del micrófono en formato WebM. Todos estos pasos vienen descritos en el fichero README del directorio del servidor como puede ser el nombre del stream (“first”) o la contraseña de dicho stream (“secret”).

Para ver el vídeo, podemos incluir una línea en la página web que está ofreciendo Apache. Al igual que la prueba del servidor bajo demanda con Apache, podemos incluir la misma línea, pero en vez de indicar el fichero de vídeo, indicamos una URL, que es donde está emitiendo Stream-m. Por ejemplo:

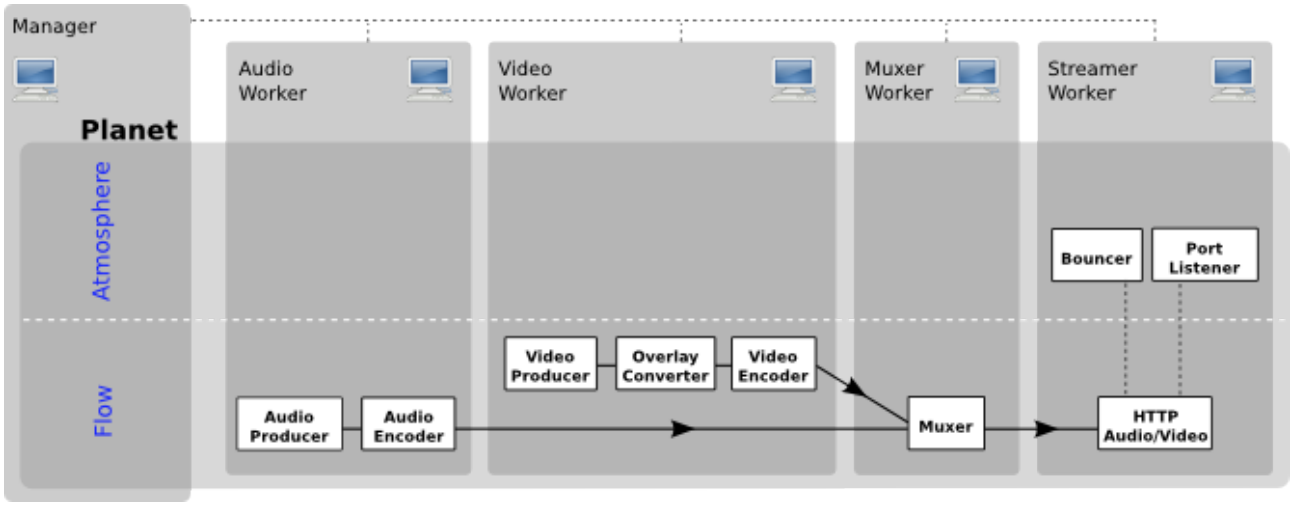
<video src=”http://localhost:8080/custome/first” controls></video>

Se incluye la palabra “publish” en la URL para enviar al servidor el vídeo, y la palabra “custome” en la URL para ver el vídeo.

Servidor Flumotion

Flumotion es un servidor de vídeo tanto en directo como bajo demanda escrito en python y desarrollado por la empresa española Flumotion Services. El servidor puede adquirirse de forma gratuita y en forma de pago, la cual la de pago ofrece mas opciones y herramientas. En este proyecto se usará el servidor básico gratuito. Podemos ver su documentación oficial aquí.

La arquitectura de Flumotion es distinta a la de los demás servidores de streaming. Se basa en separar el flujo en componentes y organizarlos según su función.



Como podemos ver en la imagen, el flujo son separados por componentes, que tienen una función específica:

- Producer: Se encargan de tomar la información que dan los dispositivos, ficheros u otros streaming desde la Red. Sería como el origen del flujo. Aquí toman la información de una webcam, micrófono, dispositivo de TV, etc.
- Encoder: Se encargan de codificar la información ofrecida por el Producer. Pueden codificar a theora, VP8, etc en el caso de vídeo, o vorbis, mp3, etc en el caso de audio.
- Converter: Convierte la imagen a distintos formatos. También contiene numerosas opciones para manejar la imagen.
- Muxer: Encapsula el audio y el vídeo resultante a un formato contenedor como pueden ser ogg, avi, webm, etc.
- HTTP Audio/Video: Servidor HTTP de streaming. Sirve las peticiones de los clientes.
- Bouncer: Mecanismos de seguridad entre los componentes.
- Port Listener: Escucha las peticiones en el puerto indicado en el servidor.

Flow es donde se encuentra el procesado del flujo de datos y la atmósfera es el paso intermedio entre la Red y el flujo. La atmósfera puede contener componentes de seguridad, como pueden ser la restricción de usuarios o verificar la seguridad entre los distintos componentes, ya que dichos componentes pueden estar en distintos equipos.

Por último, todo se engloba en un planeta, que es todo el servicio de Flumotion.

También se puede observar la presencia de manager y worker. Manager es quien controla el proceso de streaming y los worker son los que crean los procesos de los componentes, componentes, como ya se dijo, pueden estar en distintos equipos, por lo tanto en cada uno de ellos habría worker trabajando también.

Instalación

Este servidor puede ser instalado en Ubuntu desde los repositorios:

aptitude install flumotion

Al instalarlo desde los repositorios, el directorio de instalación de Flumotion será "*/etc/flumotion*", si se compila, se instalará donde se haya compilado.

Dado que la prueba se realizará en Debian, tendremos que descargarnos el fichero para compilarlo, que podemos descargarlo desde aquí.

Descargamos la última versión, "flumotion-0-10-0.tar.gz". A continuación lo descomprimimos:

tar xvzf flumotion-0-10-0.tar.gz

Se descomprimirá un directorio con todas las herramientas para compilar, pero antes debemos resolver una serie de dependencias. Podemos leer estos requisitos en el fichero README. Dado que las dependencias de Gstreamer están instaladas por defecto en Debian, instalaremos las siguientes:

- build-essential
- python2.6-dev
- libghc6-gstreamer-dev
- python-zbarpygtk

- python2.6-twisted
- python-kiwi
- python2.6-cairo
- libvorbis-dev
- libtheora-dev
- libogg-dev

Configuramos:

./configure

Compilamos:

make -j2

Y por último lo instalamos:

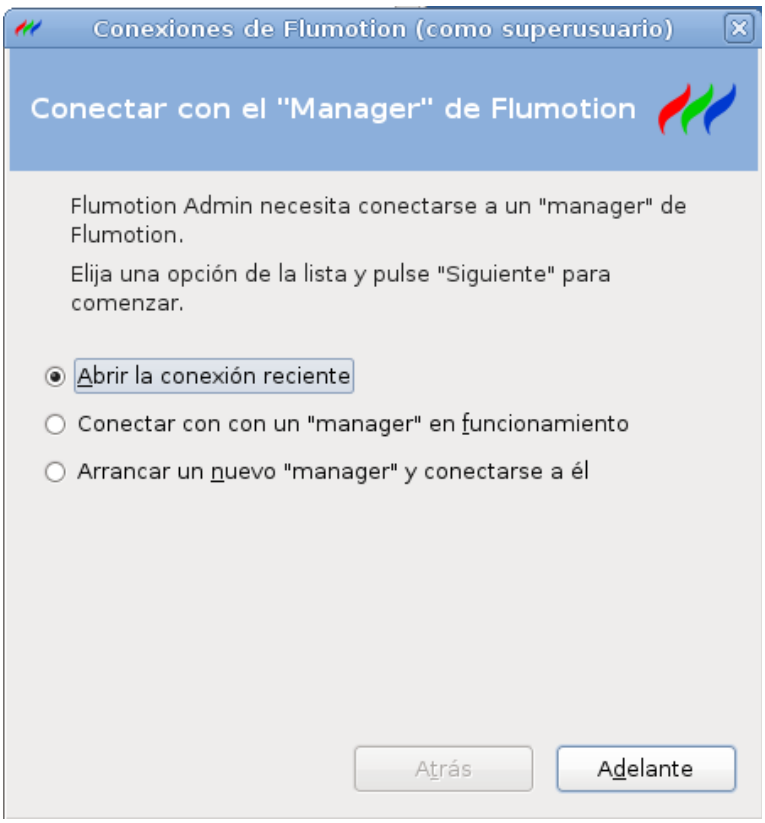
make install

Interfaz gráfico

Flumotion puede configurarse por medio de ficheros de configuración como también por su interfaz gráfico. Para abrirlo, ejecutamos el siguiente comando:

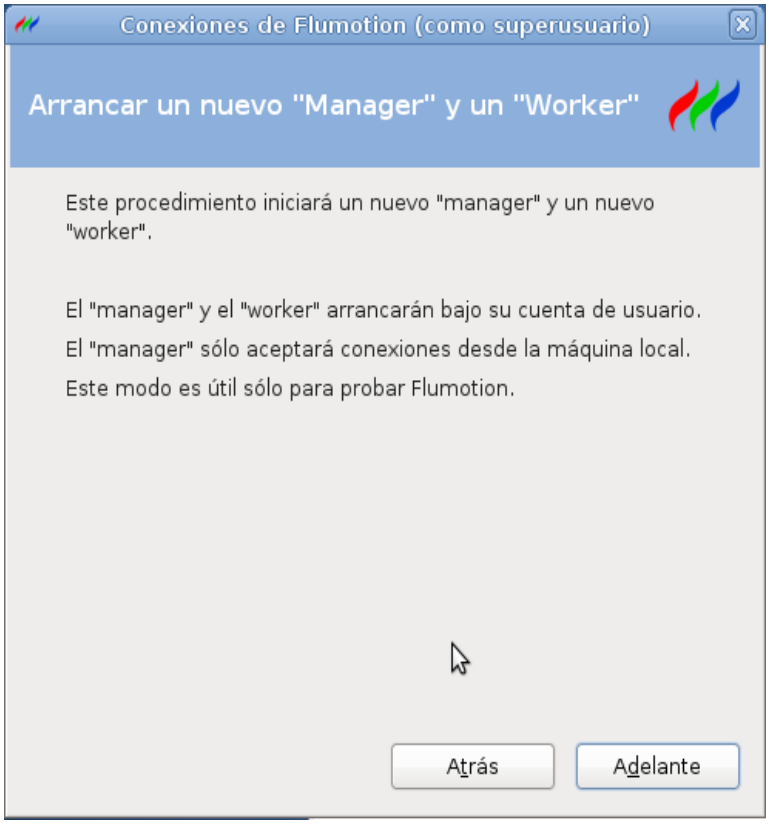
flumotion-admin

Nos aparecerá el administrador gráfico. Con él podemos configurar los manager paso a paso, pero sin poder configurar ciertos detalles que en los ficheros de configuración si podemos. He aquí una captura del asistente:

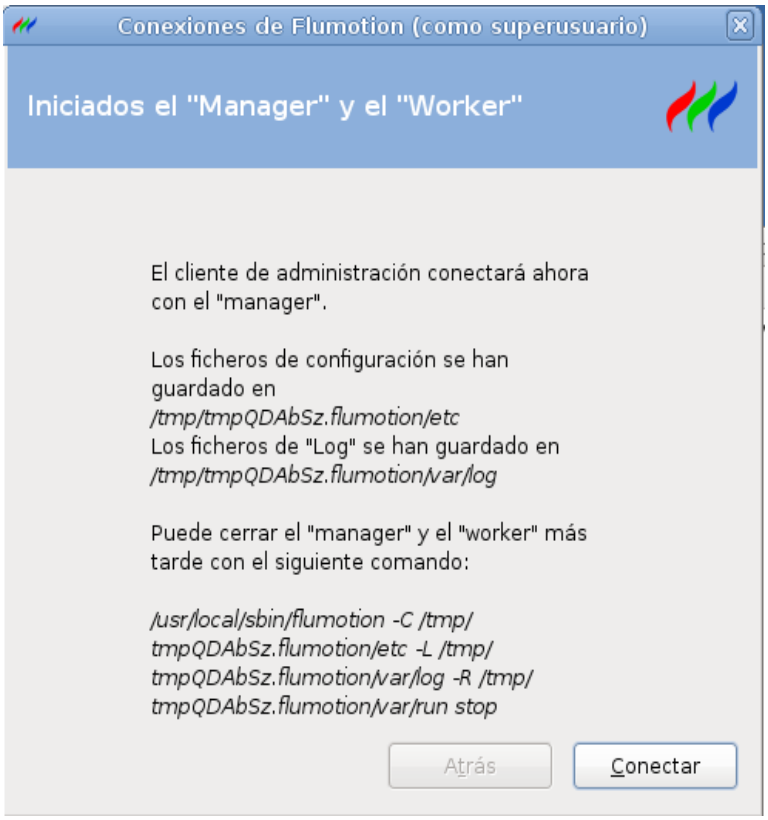


Podemos elegir entre tres opciones. Con la primera podemos abrir una conexión con un manager que hayamos arrancado anteriormente. La segunda consiste en conectarse a un manager que está corriendo ahora mismo. Con la tercera opción podemos crear un nuevo manager desde cero, con el asistente.

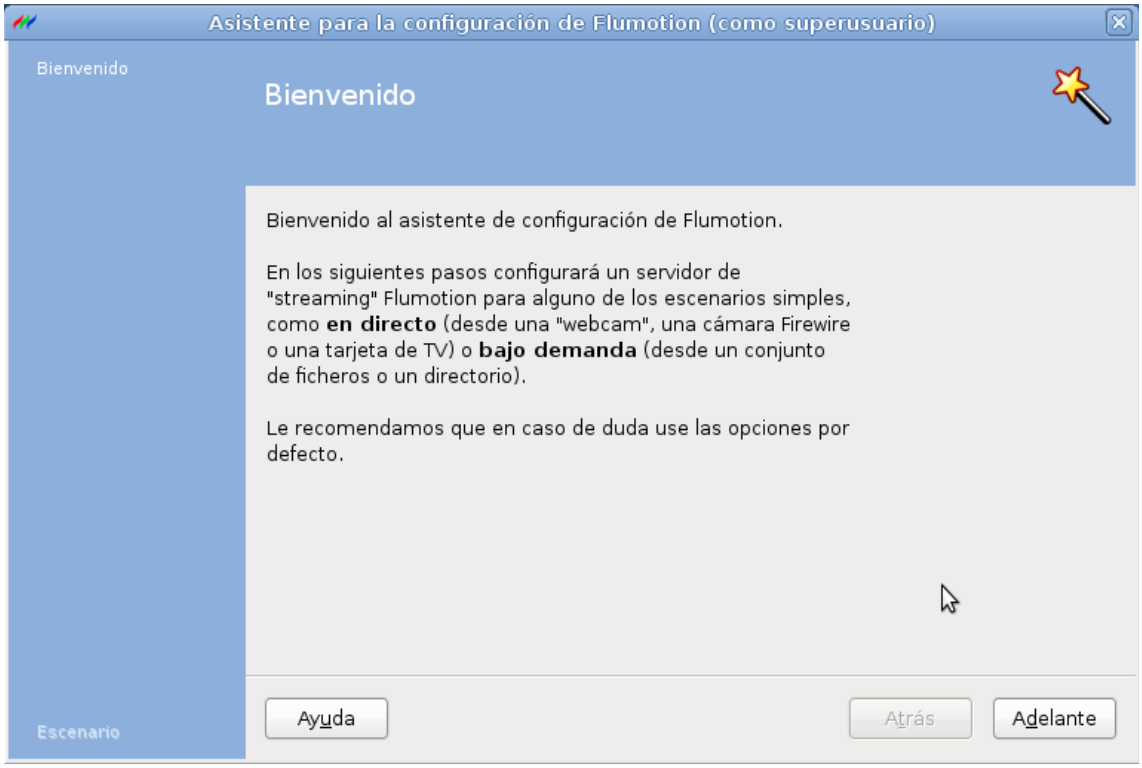
Seleccionamos la tercera opción y comenzará la configuración.



Nos indicará que se creará un manager, un worker y solo se podrá acceder a este manager de forma local. Continuamos.

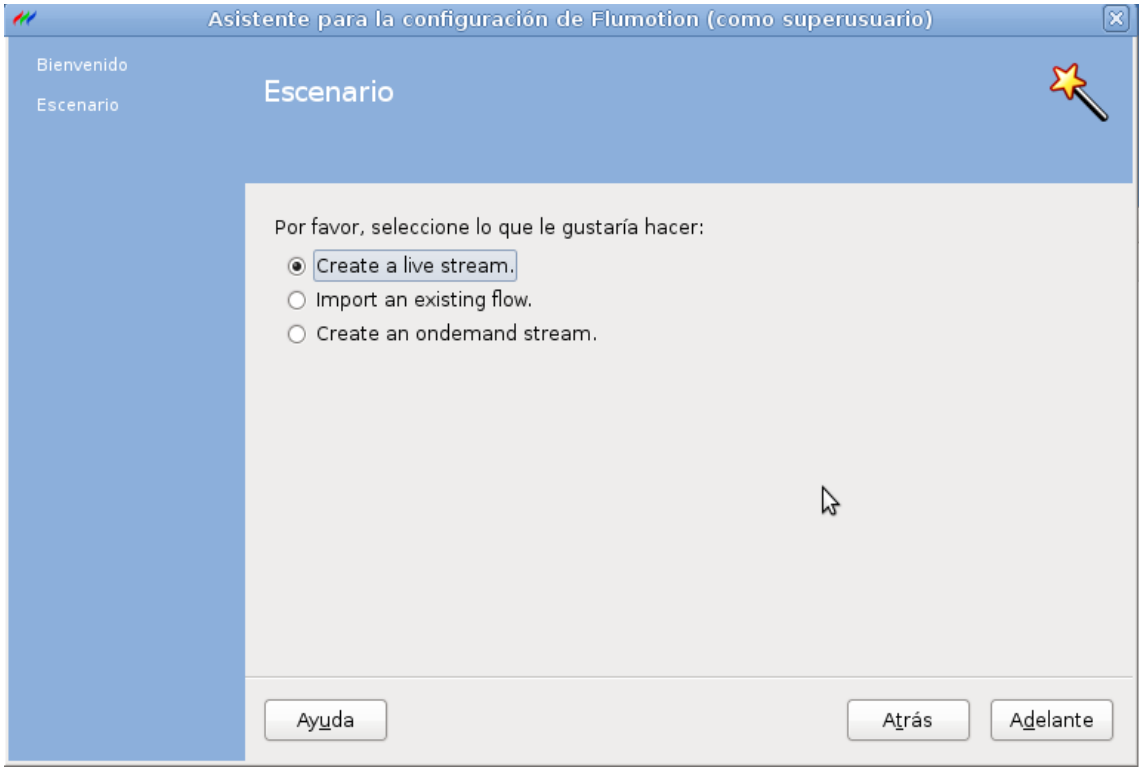


El manager y el worker están creados y ahora el asistente se conectará a ellos, Nos indica también las rutas de los ficheros de configuración y de los log, y el comando para detener el manager y el worker.

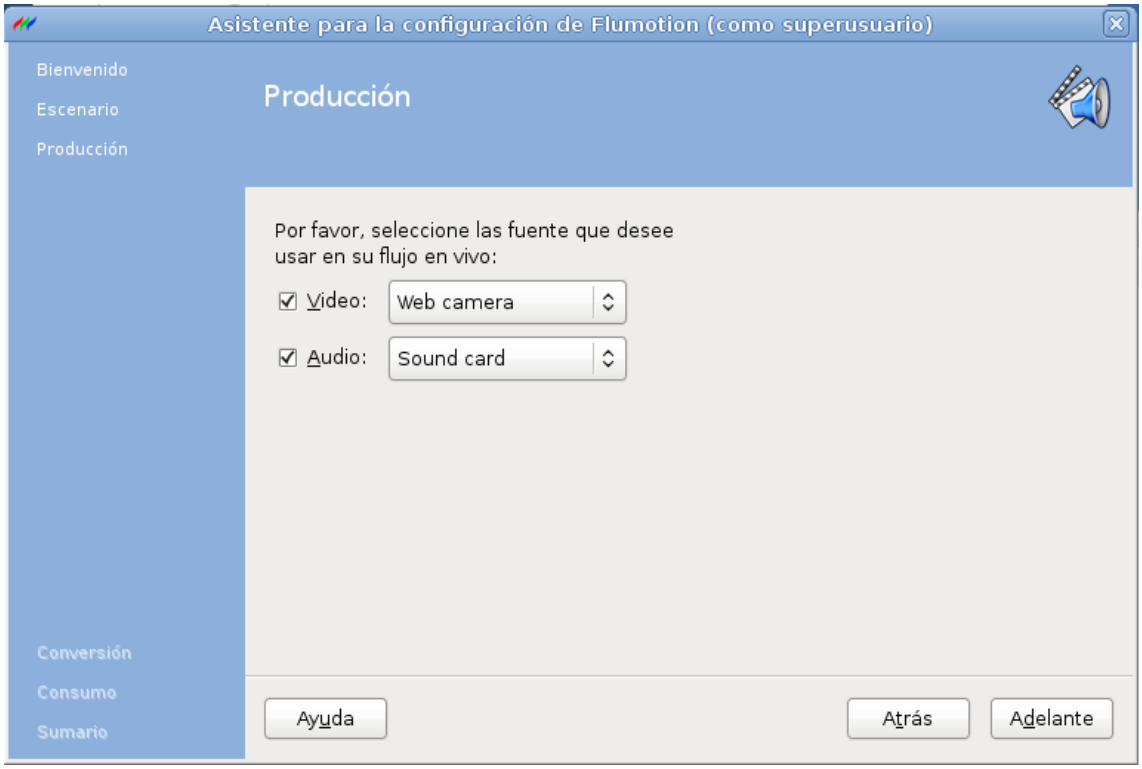


Vídeo en directo (live)

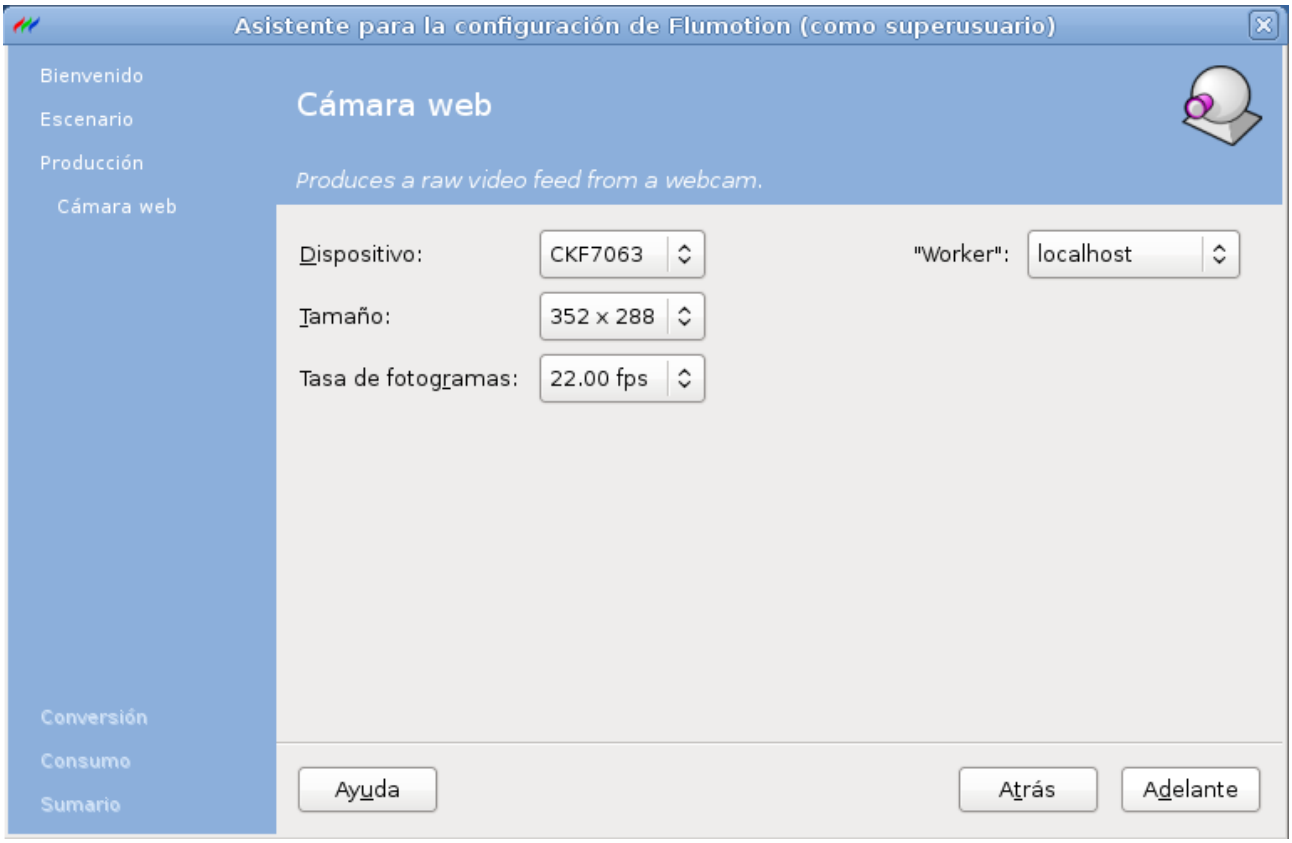
Avanzamos mas, y nos aparecerá una pantalla para seleccionar lo que queremos hacer. Por el momento elegimos crear un streaming en directo (live).



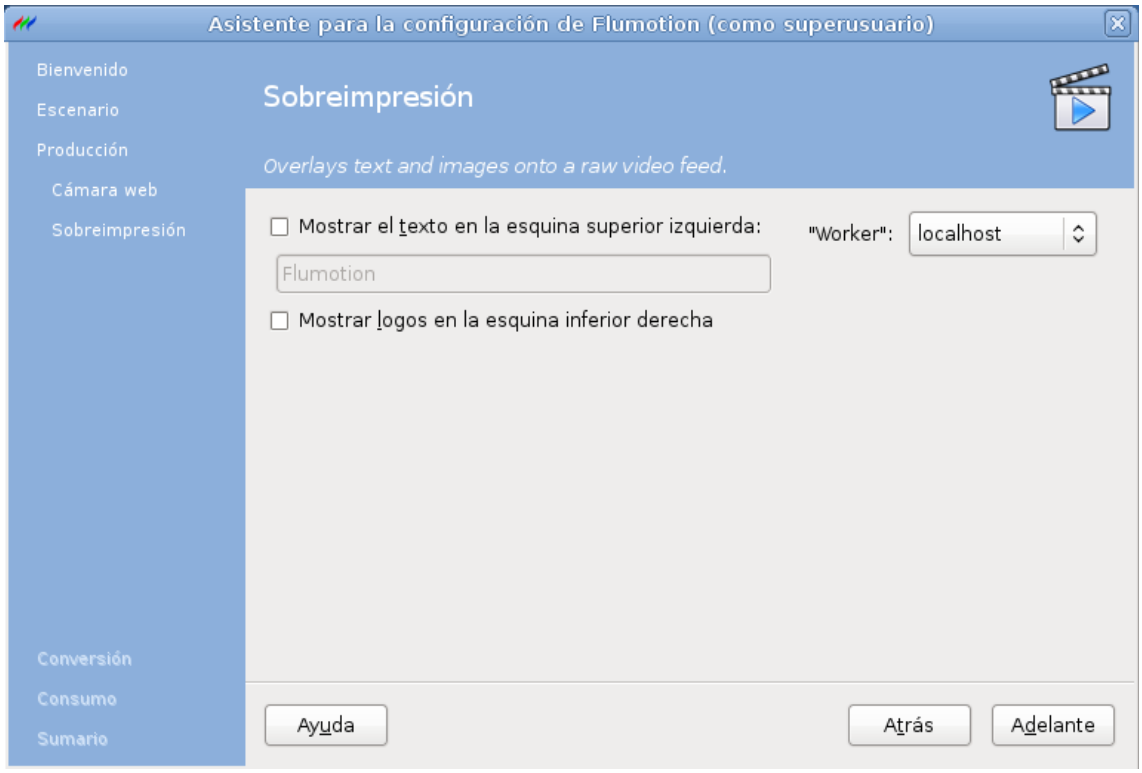
En la siguiente opción, podemos seleccionar el tener vídeo y audio. En ellas podemos elegir distintos dispositivos. Dado que en esta prueba se realizará con una webcam USB/integrada y con el micrófono, elegimos en vídeo “Web camera” y en audio “Sound card”. Continuamos.



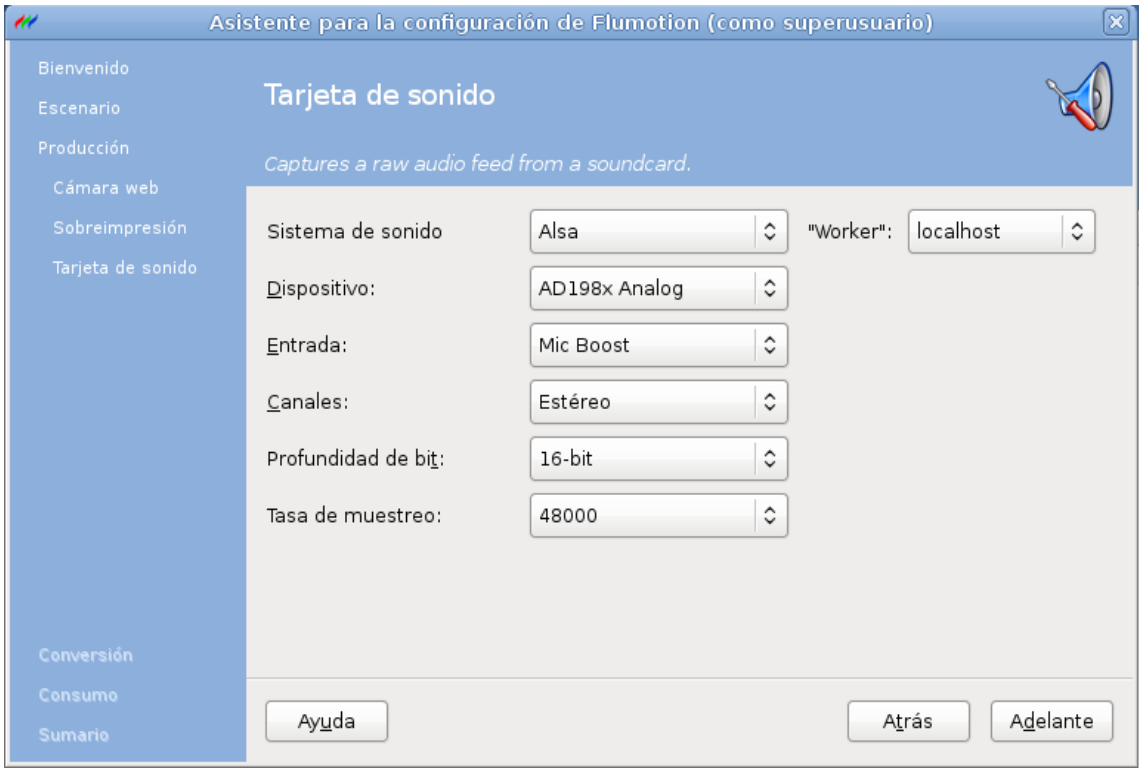
Se detectará la cámara y sus características, y podemos configurarlas a nuestro gusto. Elegimos que el worker está en localhost.



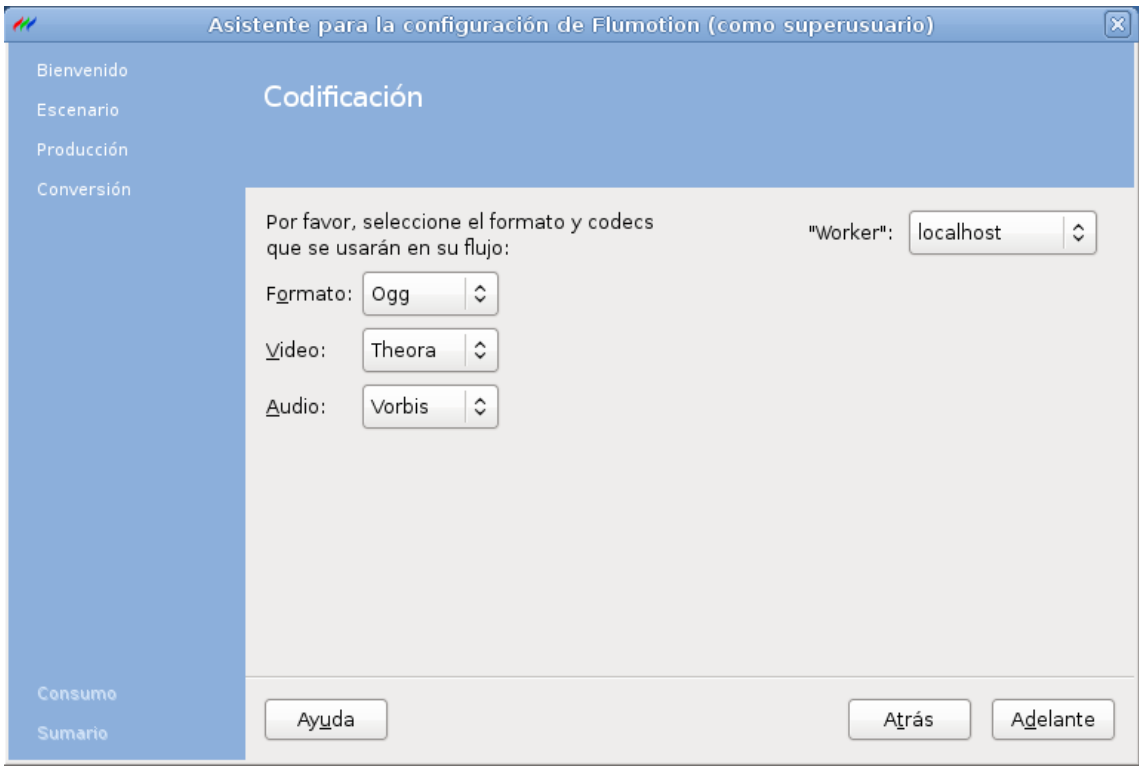
Podemos elegir un texto para que se superponga en la imagen y mostrar un logo. Podemos desactivarlas si queremos.



Ahora configuramos las opciones de la tarjeta de sonido. Elegimos el micrófono como dispositivo de entrada de audio.

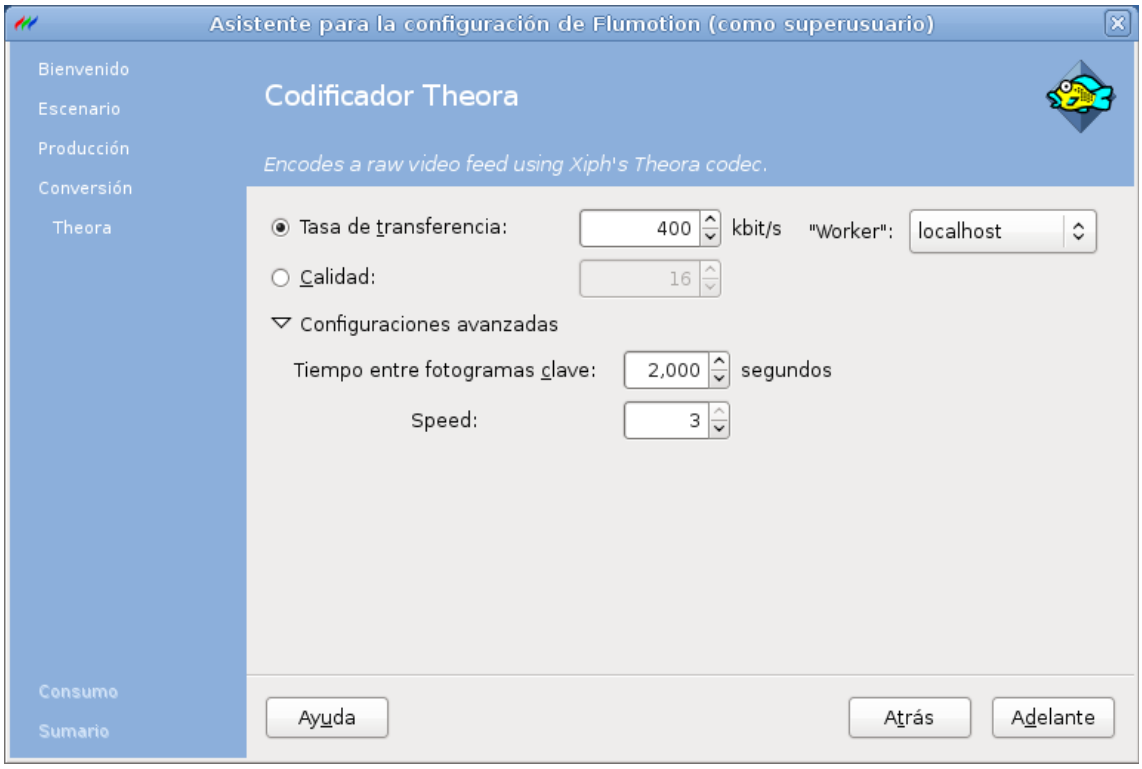


En la siguiente pantalla podemos ver que tenemos la opción de cambiar los codecs de vídeo y audio, además de elegir el formato contenedor. Aparecerán los que están instalados en Flumotion. Elegimos Theora como vídeo, Vorbis como audio, y Ogg como formato contenedor.

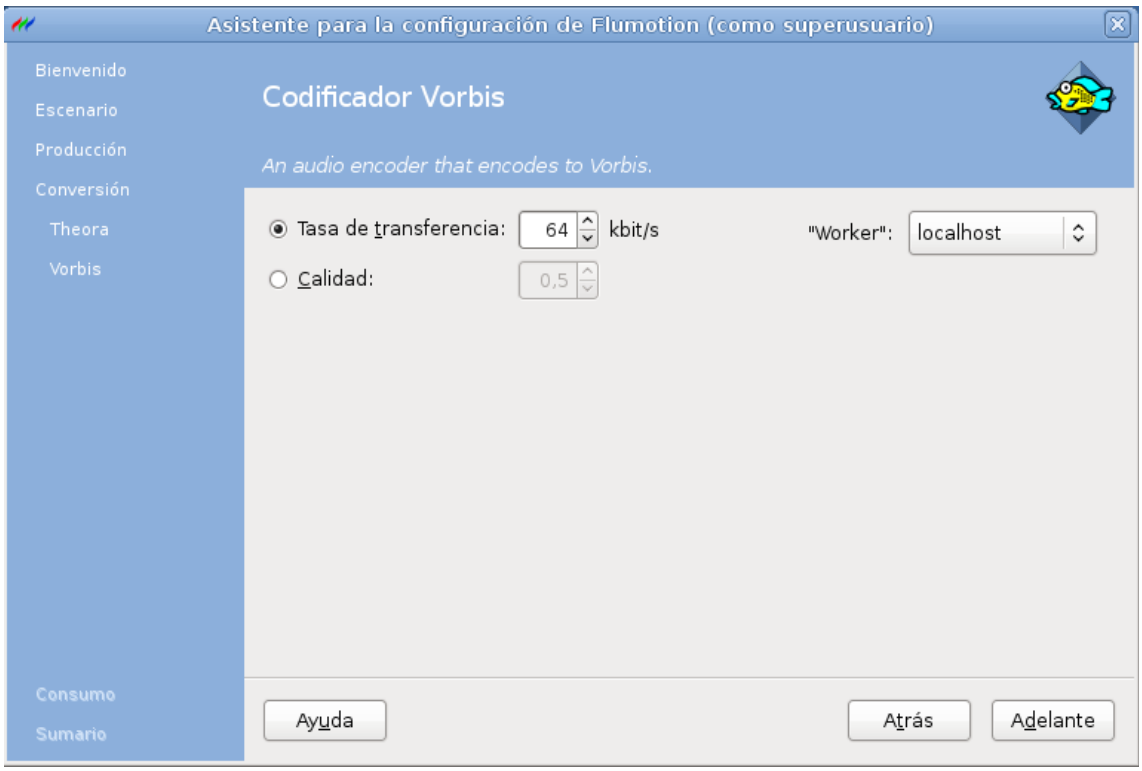


Después nos encontraremos con una pantalla que nos muestra la configuración de vídeo Theora. Podemos cambiar entre la tasa de bits o calidad. Si mantenemos la tasa de bits, podemos configurarlo de forma que el ancho de banda sea constante, pero la calidad del vídeo puede variar en algún momento. Si seleccionamos calidad, Flumotion intentará que la calidad del vídeo sea la misma durante la reproducción, variando la tasa de bits y por tanto el ancho de banda. La calidad puede tomar un valor entre 0 y 63.

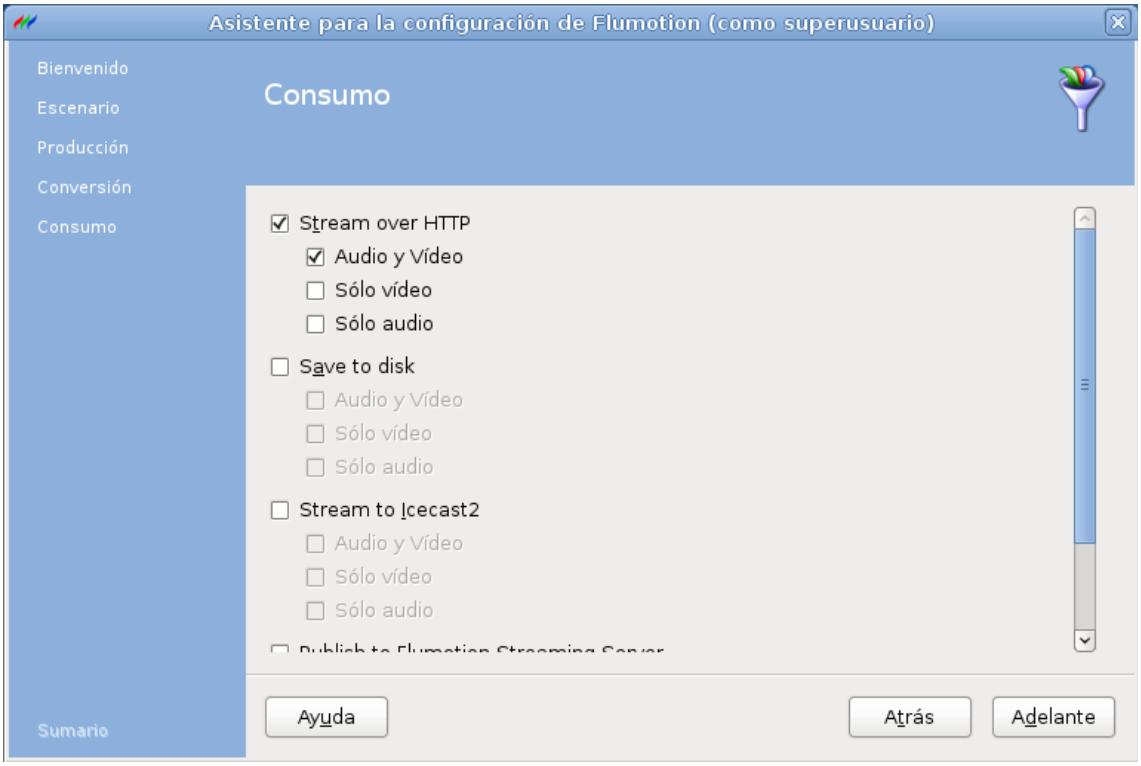
Las opciones avanzadas son opciones mas concretas del codec Theora. Podemos variar el tiempo de fotogramas clave y la velocidad.



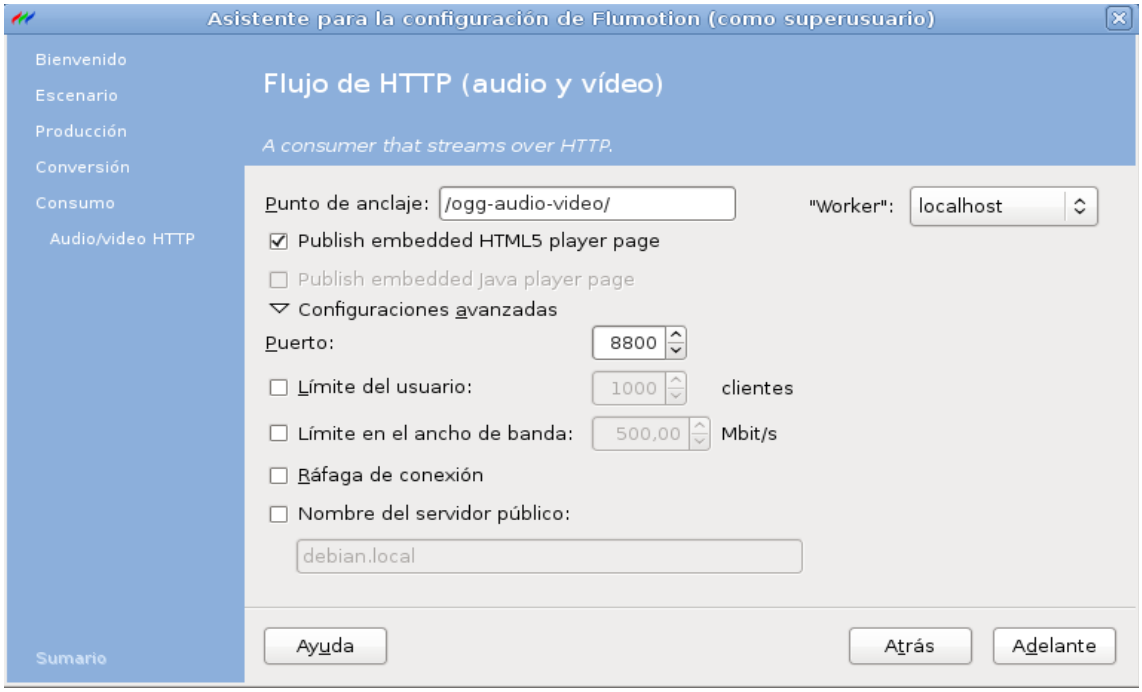
A continuación se procede a configurar un par de opciones de Vorbis. Al igual que Theora, podemos cambiar entre la tasa de bits y la calidad.



Continuamos. Lo siguiente será configurar unas opciones sobre la forma de servir el contenido. Podemos servirlo por HTTP, por medio del servidor de streaming, guardarlo en disco o enviarlo a un servidor de streaming Icecast. Elegimos por ahora la primera opción.

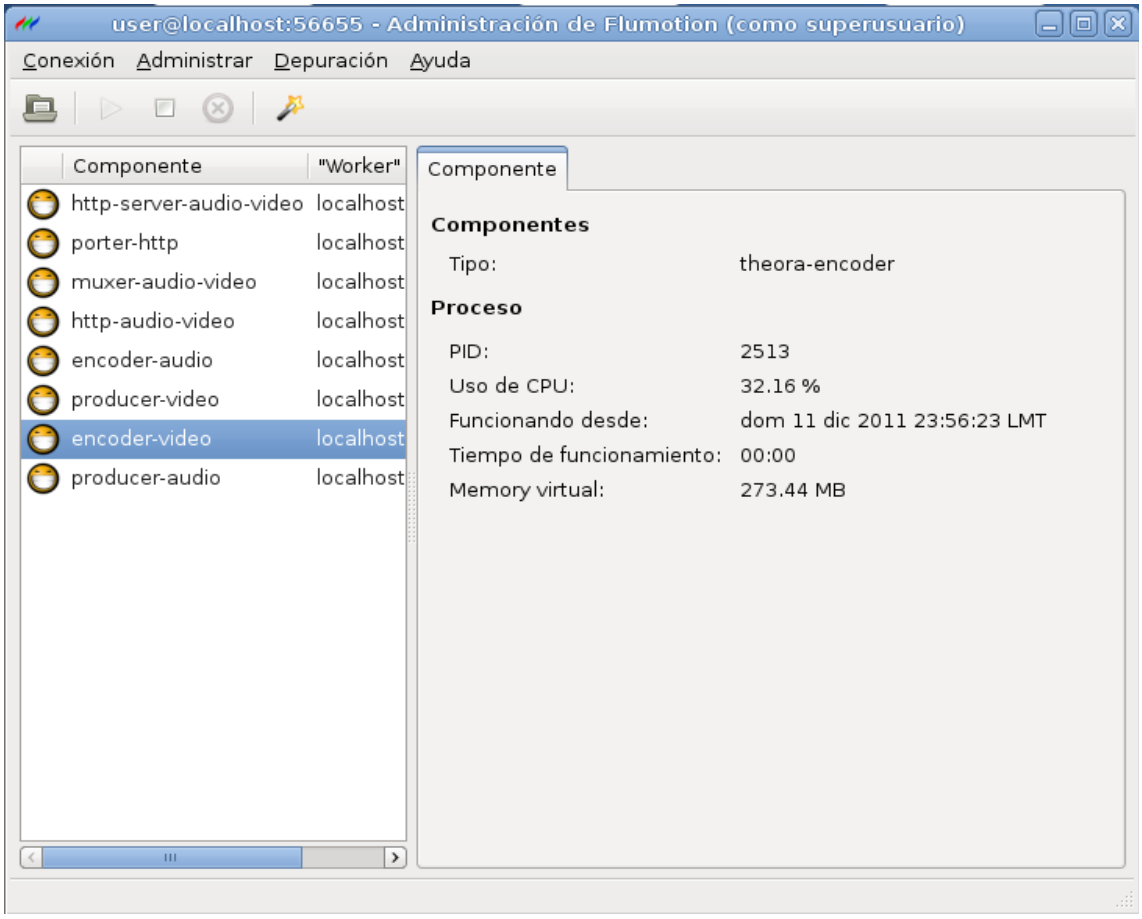


El siguiente paso es configurar opciones como el punto de anclaje, o punto de montaje. Aquí se define la URL para acceder al contenido. Podemos cambiar el puerto, limitar el ancho de banda, limitar el número de clientes, etc. Podemos activar la ráfaga de conexión (burst), que consiste en reducir el tiempo en que se llena el buffer en el cliente antes de comenzar con la reproducción. Solo sería al comienzo, después el flujo se enviaría a la velocidad configurada. También se puede indicar el nombre del servidor. Si activamos el publicar un reproductor embebido en HTML5, los clientes que accedan a la URL desde el navegador podrán ver el vídeo con un reproductor HTML5.



Ya terminamos de configurar el manager.

A continuación nos aparecerá una ventana mostrando el estado de los componentes. Podemos desde aquí detenerlos o arrancarlos. Además podemos cambiar en tiempo real algunas opciones de vídeo y ver las estadísticas del servidor.



Entre las opciones de “Conexión” podemos hacer algunas de las opciones que nos ofrecía el asistente, como conectarse a otro stream en ejecución, conectarse a uno reciente, etc. También nos da la posibilidad de exportar e importar la configuración que tengamos del stream. Al exportarla, nos dará un fichero XML con la configuración del planeta. En él podemos ver cómo están configurados los componentes. En la pestaña “Administrar” podemos manejar los componentes, pararlos, ejecutarlos, o incluso crear un nuevo stream a partir del que tenemos, así como ejecutar de nuevo el asistente. En “Depuración” podemos activarlo para ver con más detalle la configuración de cada componente, mostrándonos más pestañas en cada componente.

Ficheros de configuración

Podemos tener una referencia del fichero de configuración exportando la configuración de un stream desde el administrador gráfico. En el podemos ver cómo están dispuestos los componentes y de que forma configurarlos, aunque en la documentación oficial vienen indicadas más opciones para los componentes y su explicación.

En la prueba usamos los directorios “manager” y “worker” que están en el directorio de instalación de Flumotion, *flumotion-0.10.0/conf/*. En caso de que hubieramos instalado por repositorios, estaría en */etc/flumotion*.

Vídeo bajo demanda (VOD)

Este es un ejemplo de fichero de configuración en XML usado para ofrecer vídeo bajo demanda. Se explicarán los componentes y sus propiedades:

```
<planet>


<manager name="planet">

        <host>127.0.0.1</host>
<!--
   <host></host>
   <port></port>
   <transport></transport>
   <certificate></certificate>
-->
<!--
FIXME: would be nice if we find a way to have this be overridden by either
env var or cmd line option
   <debug>5</debug>
-->

   <component name="manager-bouncer" type="htpasswdcrypt-bouncer">
     <property name="data"><![CDATA[
user:PSfNpHTkpTx1M
]]></property>
   </component>
  </manager>

<atmosphere>
        <component name="http-server-ondemand"
                type="http-server"
                label="http-server-ondemand"
                worker="localhost"
                project="flumotion"
                version="0.10.0">

        <property name="mount-point">/</property>
        <property name="port">8800</property>
        <property name="path">/var/streaming</property>
        <plugs>
        </plugs>
        </component>
</atmosphere>

</planet>
```

Como se puede apreciar, el fichero lo abre la etiqueta <planet>, que engloba toda la configuración. Lo siguiente será poner un nombre al manager:

<manager name="planet">

Dentro de esta etiqueta podemos poner algunas opciones como el nombre o dirección IP del host. Aquí también se incluye estas líneas:

```
<component name="manager-bouncer" type="htpasswdcrypt-bouncer">
    <property name="data"><![CDATA[
user:PSfNpHTkpTx1M
]]></property>
  </component>
```

Cabe destacar que todo el fichero se basa mayoritariamente en incluir un componente con un nombre y añadirle propiedades y plugs. Esta línea es necesaria para poder usar el manager y poder ejecutar el worker.

Ahora comienza el bloque referido a la atmósfera. Ya que solo se trata de un servidor bajo demanda, solo se servirán los ficheros alojados en disco, no sería necesario crear un flujo como en el vídeo en directo.

```
<atmosphere>
        <component name="http-server-ondemand"
                type="http-server"
                label="http-server-ondemand"
                worker="localhost"
                project="flumotion"
                version="0.10.0">

        <property name="mount-point">/</property>
        <property name="port">8800</property>
        <property name="path">/var/streaming</property>
        <plugs>
        </plugs>
        </component>
</atmosphere>
```

Solo será necesario un componente, en este caso el servidor bajo demanda. Indicamos el nombre del componente servidor bajo demanda, “http-server-ondemand”. Lo siguiente será indicar algunos aspectos que va a tener el servidor. Debemos indicar el tipo de servidor para que Flumotion entienda de que se trata de un servidor HTTP. Ponemos “http-server”. Podemos poner una descripción, que iría en “label”. Ponemos que el worker está en la máquina local. Las opciones de “project” y “versión” son opcionales, podemos incluir información de la versión o del proyecto.

Lo siguiente será configurar las propiedades que tendrá el servidor. Para indicarlas, siempre usaremos “property name=[nombre de la propiedad]”. Esto lo pondremos en forma de etiquetas, y entre la apertura y el cierre de dichas etiquetas, pondremos el valor que nos interese.

La primera será “mount-point”. Esta propiedad indica cuál será el punto de montaje para acceder al contenido. Ponemos “/”, de esta forma accederemos al contenido poniendo en un reproductor o en un reproductor incrustado en la web con la URL “http://localhost:8800/[nombredelfichero.extensión]”. Indicamos el puerto en la propiedad “port”, que será el 8800. Por último, indicamos la ruta del directorio donde se encuentran los ficheros de vídeo y/o audio. También es posible incluir plugs, de los cuales se explicarán mas adelante.

Vídeo en directo (live)

Ejemplo de configuración de un servicio de vídeo en directo:

```
<?xml version="1.0" encoding="UTF-8"?>

<planet name="planet">

<manager name="planet">
        <host>127.0.0.1</host>

<component name="manager-bouncer" type="htpasswdcrypt-bouncer">
    <property name="data"><![CDATA[
user:PSfNpHTkpTx1M
]]></property>
  </component>
 </manager>


 <atmosphere>
  <component name="http-server-audio-video"
        type="http-server"
        label="http-server-audio-video"
        worker="localhost"
        project="flumotion"
        version="0.10.0">
   <property name="porter-username">oXNIfTxYNVwI</property>
   <property name="mount-point">/ogg-audio-video/html5/</property>
   <property name="hostname">debian.local</property>
   <property name="porter-password">urCEpMwNRsHC</property>
   <property name="type">slave</property>
   <property name="porter-socket-path">flu-PrQjmw.socket</property>
   <property name="port">8800</property>
   <plugs>
    <plug type="component-html5">
     <property name="width">640</property>
     <property name="stream-url">http://debian.local:8800/ogg-audio-video/</property>
     <property name="codecs">vorbis,theora</property>
     <property name="mime-type">video/ogg</property>
     <property name="height">480</property>
    </plug>
   </plugs>
  </component>
```

```
  <component name="porter-http"
             type="porter"
             label="porter-http"
             worker="localhost"
             project="flumotion"
             version="0.10.0">
    <property name="username">oXNIfTxYNVwI</property>
    <property name="socket-path">flu-PrQjmw.socket</property>
    <property name="password">urCEpMwNRsHC</property>
    <property name="port">8800</property>
  </component>

</atmosphere>

<flow name="default">
  <component name="producer-audio"
             type="soundcard-producer"
             label="producer-audio"
             worker="localhost"
             project="flumotion"
             version="0.10.0">
    <property name="device">hw:0</property>
    <property name="channels">2</property>
    <property name="depth">16</property>
    <property name="input-track">Capturar</property>
    <property name="samplerate">44100</property>
    <clock-master>true</clock-master>
  </component>

  <component name="producer-video"
             type="webcam-producer"
             label="producer-video"
             worker="localhost"
             project="flumotion"
             version="0.10.0">
    <property name="format">YUY2</property>
    <property name="framerate">22/1</property>
    <property name="height">480</property>
    <property name="width">640</property>
    <property name="mime">video/x-raw-yuv</property>
    <property name="element-factory">v4l2src</property>
    <property name="device">/dev/video0</property>
    <clock-master>false</clock-master>
  </component>

  <component name="encoder-video"
             type="theora-encoder"
             label="encoder-video"
             worker="localhost"
             project="flumotion"
             version="0.10.0">
    <eater name="default">
      <feed alias="default">producer-video:default</feed>
    </eater>
    <property name="keyframe-maxdistance">44</property>
    <property name="speed">3</property>
    <property name="bitrate">400000</property>
    <clock-master>false</clock-master>
```

```
  </component>

  <component name="encoder-audio"
         type="vorbis-encoder"
         label="encoder-audio"
         worker="localhost"
         project="flumotion"
         version="0.10.0">
   <eater name="default">
    <feed alias="default">producer-audio:default</feed>
   </eater>
   <property name="bitrate">64000</property>
   <clock-master>false</clock-master>
  </component>

  <component name="muxer-audio-video"
         type="ogg-muxer"
         label="muxer-audio-video"
         worker="localhost"
         project="flumotion"
         version="0.10.0">
   <eater name="default">
    <feed alias="default">encoder-audio:default</feed>
    <feed alias="default-bis">encoder-video:default</feed>
   </eater>
   <clock-master>false</clock-master>
  </component>

  <component name="http-audio-video"
         type="http-streamer"
         label="http-audio-video"
         worker="localhost"
         project="flumotion"
         version="0.10.0">
   <eater name="default">
    <feed alias="default">muxer-audio-video:default</feed>
   </eater>
   <property name="burst-on-connect">False</property>
   <property name="porter-username">oXNIfTxYNVwI</property>
   <property name="mount-point">/ogg-audio-video/</property>
   <property name="port">8800</property>
   <property name="type">slave</property>
   <property name="porter-socket-path">flu-PrQjmw.socket</property>
   <property name="porter-password">urCEpMwNRsHC</property>
   <clock-master>false</clock-master>
   <plugs>
   </plugs>
  </component>

 </flow>

</planet>
```

Nota: debian.local puede cambiarse por localhost o la IP actual del sistema para que sea accesible desde la red.

En este fichero, al igual que el de bajo demanda, Ponemos la línea referente a usuario y contraseña del manager para que el worker pueda funcionar. Al configurar la atmósfera, comenzamos a incluir componentes, que en este caso, serán mas numerosos.

El primer componente del ejemplo consiste en que Flumotion ofrecerá un reproductor HTML5 si accedemos desde el navegador a la URL del contenido (si no queremos que se muestre en el navegador con un reproductor incrustado podemos omitir el componente). Primero se configuraría el servidor HTTP y después por la etiqueta plug, las cualidades del reproductor.
En las opciones del servidor, indicamos el tipo, en este caso “http-server”, y las demás opciones, al igual que en la prueba bajo demanda. En todo el fichero se usará la misma información excepto en el tipo, que será diferente en cada componente, para que Flumotion pueda distinguir entre ellos.

A continuación, pondremos algunos datos del componente “porter”. Este componente se encarga de escuchar las peticiones de los clientes en un determinado puerto y pasárselas al servidor de streaming (http-streamer)y al de HTTP (http-server). Podemos poner en “porter-username” cualquier nombre de usuario. En el ejemplo está escrito el que configuró el administrador gráfico, aunque podemos poner el que queramos. Lo mismo con la contraseña, “porter-password”. En “porter-socket.path” establecemos el nombre que tendrá el socket donde se comunicarán el porter con los servidores de streaming y HTTP.
Pondremos el punto de montaje donde se servirá el reproductor HTML5, en este caso será /ogg-audio-video/html5. El el ejemplo, la propiedad de “hostname” viene como “debian.local”, configurado también con el administrador gráfico, aunque podemos poner también “localhost” o la dirección IP del sistema. Si ponemos la IP del sistema podemos visualizar el reproductor HTML5 desde otro equipo de la red.
La siguiente opción a indicar es “type”, donde indicamos de qué forma va a escuchar el componente. Las dos opciones son “master” y “slave”. Si no ponemos nada, la opción por defecto será “master”, y por tanto escuchará directamente en el puerto indicado. En el caso de “slave”, escuchará en el puerto pero a través del “porter”, el cuál este sería intermediario entre la red y el servidor HTTP o el servidor de streaming.
Después indicamos el puerto (“port”) como el 8800, que es donde vendrán las peticiones.
En la etiqueta “plug” pondremos las características de un reproductor HTML5. Lo indicamos poniendo como nombre “component-html5”. Configuramos las dimensiones que va a tener el reproductor, que van a ser 640 de “width” (ancho) y 480 de “height” (alto). Definimos la URL para acceder desde el navegador, en este caso “http://debian.local:8800/ogg.audio-video/”.
Ahora configuramos que codecs va a reproducir el reproductor. Como el vídeo estará codificado en theora y vorbis, definimos en “codecs” los codecs “vorbis,theora”. Y el tipo de vídeo en “mime-type” como “video/ogg”.

Cerramos las correspondientes etiquetas y pasamos al siguiente componente:

```
<component name="porter-http"
            type="porter"
            label="porter-http"
            worker="localhost"
            project="flumotion"
            version="0.10.0">
     <property name="username">oXNIfTxYNVwI</property>
     <property name="socket-path">flu-PrQjmw.socket</property>
     <property name="password">urCEpMwNRsHC</property>
     <property name="port">8800</property>
    </component>
```

En este bloque definimos la configuración del porter, necesario para que los componentes del servidor HTTP y de streaming puedan ser accedidos a la red a través de este. Para ello ambos componentes deben de ser tipo esclavo (slave).
Ponemos el como nombre “porter-http” por ejemplo. El tipo lo definimos como “porter” y el resto de atributos son los típicos que vamos poniendo en el fichero. Ponemos el “username” y el “password” como queramos. El “socket-path” también lo definimos como queramos y el “port” lo indicamos como 8800. Todas estas propiedades deben coincidir en los componentes HTTP y streamer.

Cerramos la atmósfera y comenzamos con el flujo (flow). El nombre del flujo está definido como “default” en esta prueba.

```
<component name="producer-audio"
            type="soundcard-producer"
            label="producer-audio"
            worker="localhost"
            project="flumotion"
            version="0.10.0">
     <property name="device">hw:0</property>
     <property name="channels">2</property>
     <property name="depth">16</property>
     <property name="input-track">Capturar</property>
     <property name="samplerate">44100</property>
     <clock-master>true</clock-master>
    </component>
```

Este componente es el que se encarga de tomar el audio del dispositivo de audio. Puede ser la tarjeta de sonido, sonido test que viene con Flumotion, etc. En esta prueba está configurado para que capture el sonido del micrófono.
Definimos el nombre “producer-audio”. El tipo tiene que ser “soundcard-producer” para que tome el sonido de la tarjeta de sonido.
En las propiedades, definimos el dispositivo ALSA “hw:0”, que es el dispositivo de captura de audio. En “channels” (canales) definimos 2 para que sea estereo. Ponemos en “depth” el número 16 que es el número de bits por muestra (sample) que tendrá el audio. Normalmente el sonido de 16

bits por muestra es utilizado en los CD de audio. Configuramos “input-track” como “Capturar”. Esta es la etiqueta del dispositivo de captura. Lo siguiente será configurar el “samplerate”. El samplerate (frecuencia de muestreo) comúnmente usado es 44100 Hz, también muy usado en los CD de audio.

Pasamos al siguiente componente, el “producer-video”:

```
<component name="producer-video"
            type="webcam-producer"
            label="producer-video"
            worker="localhost"
            project="flumotion"
            version="0.10.0">
    <property name="format">YUY2</property>
    <property name="framerate">22/1</property>
    <property name="height">480</property>
    <property name="width">640</property>
    <property name="mime">video/x-raw-yuv</property>
    <property name="element-factory">v4l2src</property>
    <property name="device">/dev/video0</property>
    <clock-master>false</clock-master>
  </component>
```

Como en los demás componentes, ponemos el nombre del componente y sus demás opciones. En tipo establecemos “webcam-producer” ya que el vídeo se tomará de una webcam.
En “format” establecemos el formato de vídeo de origen, en este caso YUY2, que es el formato nativo de la webcam. Otro formato de webcam puede ser I420, aunque existen variedad de formatos más. En “framerate” establecemos los frames por segundo (fps), en este caso 22. Ponemos la resolución, 480 como “height” y 640 como “width”. El tipo mime lo ponemos con la propiedad “mime”, y será “video/x-raw-yuv”, que es el formato nativo de la webcam. Usamos “element-factory” para establecer la API de captura de vídeo, en este caso “v4l2src” (video4linux2). Por último, indicamos el dispositivo que representa a la webcam, “/dev/video0”.

Lo siguiente será configurar los codecs de vídeo y audio. De aquí saldrá la codificación final que se va a ofrecer a los usuarios:

```
<component name="encoder-video"
            type="theora-encoder"
            label="encoder-video"
            worker="localhost"
            project="flumotion"
            version="0.10.0">
    <eater name="default">
      <feed alias="default">producer-video:default</feed>
    </eater>
    <property name="keyframe-maxdistance">44</property>
    <property name="speed">3</property>
    <property name="bitrate">400000</property>
    <clock-master>false</clock-master>
  </component>
```

Configuramos las características del componente, de tipo “theora-encoder”, ya que lo vamos a codificar el vídeo en Theora.

A partir de aquí aparecen un par de nuevas etiquetas, “eater” y “feed”. La etiqueta “eater” consiste en indicar de dónde se va a tomar el flujo de vídeo o audio, en este caso se tomará del componente anterior de vídeo, el “producer-video”. Como nombre del “eater” ponemos default por ejemplo. En el “feed” ponemos como alias default también, y dentro de la etiqueta ponemos el nombre que le dimos al componente que toma el vídeo en el origen, que es “producer-video”. Lo que viene después de los dos puntos es para diferenciar si el componente tiene mas salidas. Como solo tiene una en este ejemplo, podemos “default”.
En las propiedades como “keyframe-maxdistance” ponemos 44, que son la distancia máxima en frames de los frames clave, que son especificaciones técnicas de Theora. A continuación, establecemos al bitrate a 400000 bits por segundo. Esta propiedad se puede manejar dependiendo del ancho de banda.

```
<component name="encoder-audio"
            type="vorbis-encoder"
            label="encoder-audio"
            worker="localhost"
            project="flumotion"
            version="0.10.0">
     <eater name="default">
       <feed alias="default">producer-audio:default</feed>
     </eater>
     <property name="bitrate">64000</property>
     <clock-master>false</clock-master>
   </component>
```

Configuramos ahora la codificación del audio. El tipo lo ponemos como “vorbis-encoder”, ya que en la prueba se utilizará el codec Vorbis para el audio.
Configuramos “eater” y “feed”. En este caso, ponemos los nombres default de nuevo, y ponemos el nombre del componente productor de audio, “producer-audio” y separados por dos puntos, default, ya que solo habrá una salida de audio. En las propiedades solo ponemos que el bitrate será a 64000 bits por segundo. El resto de propiedades, como el samplerate, se dejarán tal y como vienen desde el “producer-audio”.

```
<component name="muxer-audio-video"
            type="ogg-muxer"
            label="muxer-audio-video"
            worker="localhost"
            project="flumotion"
            version="0.10.0">
     <eater name="default">
       <feed alias="default">encoder-audio:default</feed>
       <feed alias="default-bis">encoder-video:default</feed>
     </eater>
     <clock-master>false</clock-master>
   </component>
```

Lo siguiente será configurar el componente que unirá el flujo de vídeo y el flujo de audio y los encapsulará en un formato contenedor, como es el caso de Ogg.
El tipo de componente será “ogg-muxer”. Ahora pondremos dos entradas en “eater”, ya que llegarán dos flujos. Un feed lo llamamos default por ejemplo y definimos el nombre del componente anterior de audio, el codificador Vorbis, que es “encoder-audio” y separados por dos puntos default. El siguiente feed lo ponemos como default-bis para que no se solapen, e indicamos el anterior componente de vídeo en el flujo, “encoder-video”, y separado por dos puntos default.

```
<component name="http-audio-video"
            type="http-streamer"
            label="http-audio-video"
            worker="localhost"
            project="flumotion"
            version="0.10.0">
    <eater name="default">
      <feed alias="default">muxer-audio-video:default</feed>
    </eater>
    <property name="burst-on-connect">False</property>
    <property name="porter-username">oXNIfTxYNVwI</property>
    <property name="mount-point">/ogg-audio-video/</property>
    <property name="port">8800</property>
    <property name="type">slave</property>
    <property name="porter-socket-path">flu-PrQjmw.socket</property>
    <property name="porter-password">urCEpMwNRsHC</property>
    <clock-master>false</clock-master>
    <plugs>
    </plugs>
  </component>
```

El último componente se encargará de servir el flujo a la red. Es similar al componente HTTP “http-server”, solo que este era para mostrar una página web con el reproductor HTML5, y el que vamos a configurar será el servidor de streaming propiamente dicho que servirá el flujo.
El tipo de este componente será “http-streamer”, ya que el streaming se servirá por el protocolo HTTP.
En el “eater” ponemos un nommbre, default por ejemplo, y en el “feed” indicamos tambien el nombre default, además de definirlo como “muxer-audio-video:default”, ya que tomará el flujo del componente anterior, que encapsuló los flujos de vídeo y audio en el formato contenedor.
En las propiedades podemos activar el “burst-on-connect”. Esto se trata de enviar al cliente una cantidad del flujo mas rápido al comienzo de la reproducción. Esto sirve para que el usuario pueda visualizar el contenido sin esperar demasiado que se llene el buffer al comienzo de la reproducción. Después continuará a velocidad normal.
Ya que se usará el componente “porter”, indicamos los datos del porter, como el nombre de usuario, password y socket. Permitimos al componente “http-streamer” que use el porter poniendo como tipo “slave”. El puerto será el 8800.

Con todo esto cerramos las etiquetas del fichero.

Una vez terminado de configurar el fichero de configuración, veremos el fichero de worker. Estos ficheros sirven para poner en marcha los componentes, y no es necesario que estén en el mismo equipo. Ya que las pruebas se harán en una máquina local, todos los ficheros de configuración se encontrarán en la misma. Los ficheros de worker se pondrán dentro de la instalación de Flumotion, en flumotion-0.10.0/conf/workers/. Aquí pondremos los ficheros de worker, que después de instalar Flumotion, aparecerá un fichero default.xml. Podemos usar este, el cual su contenido es el siguiente:

```
<worker>
<!--
You can override the name of the worker, which will typically be
hostname:(xmlfilename)
<worker name="default">
-->

   <manager>
<!--
     This specifies what manager to log in to.
     Compare with command-line options.

     <host></host>
     <port></port>
     <transport></transport>
-->
   </manager>

   <authentication type="plaintext">
<!--
     This specifies what authentication to use to log in.
     Compare with command-line options.
-->
     <username>user</username>
     <password>test</password>
   </authentication>

   <feederports>8650-8669</feederports>
   <debug>*:4</debug>

</worker>
```

Podemos apreciar que en este fichero aparecen algunas opciones. Dentro de la etiqueta “manager” podemos configurarlo de forma que, si el manager se encuentra en otro equipo distinto al que se encuentra el worker, pondremos aquí los datos del servidor del manager, como el nombre de host o su IP, el puerto donde va a escuchar y el protocolo. Mas abajo podemos apreciamos que podemos poner un nombre de usuario y un password. Esto sirve para que al ejecutar el worker se autentique con el manager, por lo que este nombre y password debe estar también en el manager. Hay diferentes tipos de autenticación, en este caso en texto plano (plaintext).

Para hacer andar el servicio, introducimos el siguiente comando:

flumotion-manager -v -T tcp /flumotion-0.10.0/conf/managers/default/planet.xml

Este comando iniciará el manager, que se encuentra en la ruta especificada. La opción -v es para ver el log en pantalla, y la opción -T para que use TCP.

A continuación, ejecutaremos el worker para que los componentes empiecen a trabajar:

flumotion-worker -v -T tcp /flumotion-0.10.0/conf/workers/default.xml

También puede ejecutarse sin el fichero con:

flumotion-worker -v -T tcp -u usuario -p password

Con esto haremos andar el servicio de streaming, el cual podremos ver desde un navegador poniendo http://[IP sel servidor]:8800/[punto de montaje]. Lo mismo si accedemos desde un reproductor. También funcionará si incrustamos un reproductor en una web poniendo esta misma URL.

Otros componentes:

VP8

Ejemplo de fichero del manager con un componente codificador VP8:

```
<?xml version="1.0" encoding="UTF-8"?>
<planet name="admin">

  <atmosphere>
    <component name="http-server-audio-video"
               type="http-server"
               label="http-server-audio-video"
               worker="localhost"
               project="flumotion"
               version="0.10.0">
      <property name="porter-username">MHenvvvmBxaR</property>
      <property name="mount-point">/prueba/html5/</property>
      <property name="hostname">debian.local</property>
      <property name="porter-password">vCslZJcBAUMk</property>
      <property name="type">slave</property>
      <property name="porter-socket-path">flu-pOoSZR.socket</property>
      <property name="port">8800</property>
      <plugs>
        <plug type="component-html5">
```

```
      <property name="width">352</property>
      <property name="stream-url">http://debian.local:8800/prueba/</property>
      <property name="codecs">vorbis,vp8</property>
      <property name="mime-type">video/webm</property>
      <property name="height">288</property>
    </plug>
  </plugs>
 </component>

 <component name="porter-http"
        type="porter"
        label="porter-http"
        worker="localhost"
        project="flumotion"
        version="0.10.0">
  <property name="username">MHenvvvmBxaR</property>
  <property name="socket-path">flu-pOoSZR.socket</property>
  <property name="password">vCslZJcBAUMk</property>
  <property name="port">8800</property>
 </component>

</atmosphere>

<flow name="default">
 <component name="producer-audio"
        type="soundcard-producer"
        label="producer-audio"
        worker="localhost"
        project="flumotion"
        version="0.10.0">
  <property name="device">hw:0</property>
  <property name="channels">2</property>
  <property name="depth">16</property>
  <property name="input-track">Capturar</property>
  <property name="samplerate">48000</property>
  <clock-master>true</clock-master>
 </component>

 <component name="producer-video"
        type="webcam-producer"
        label="producer-video"
        worker="localhost"
        project="flumotion"
        version="0.10.0">
  <property name="format">YUY2</property>
  <property name="framerate">22/1</property>
  <property name="height">288</property>
  <property name="width">352</property>
  <property name="mime">video/x-raw-yuv</property>
  <property name="element-factory">v4l2src</property>
  <property name="device">/dev/video0</property>
  <clock-master>false</clock-master>
 </component>

 <component name="encoder-audio"
        type="vorbis-encoder"
        label="encoder-audio"
        worker="localhost"
```

```
        project="flumotion"
        version="0.10.0">
    <eater name="default">
      <feed alias="default">producer-audio:default</feed>
    </eater>
    <property name="bitrate">64000</property>
    <clock-master>false</clock-master>
  </component>

  <component name="encoder-video"
        type="vp8-encoder"
        label="encoder-video"
        worker="localhost"
        project="flumotion"
        version="0.10.0">
    <eater name="default">
      <feed alias="default">producer-video:default</feed>
    </eater>
    <property name="keyframe-maxdistance">44</property>
    <property name="bitrate">400000</property>
    <clock-master>false</clock-master>
  </component>

  <component name="muxer-audio-video"
        type="webm-muxer"
        label="muxer-audio-video"
        worker="localhost"
        project="flumotion"
        version="0.10.0">
    <eater name="default">
      <feed alias="default">encoder-audio:default</feed>
      <feed alias="default-bis">encoder-video:default</feed>
    </eater>
    <clock-master>false</clock-master>
  </component>

  <component name="http-audio-video"
        type="http-streamer"
        label="http-audio-video"
        worker="localhost"
        project="flumotion"
        version="0.10.0">
    <eater name="default">
      <feed alias="default">muxer-audio-video:default</feed>
    </eater>
    <property name="burst-on-connect">False</property>
    <property name="porter-username">MHenvvvmBxaR</property>
    <property name="mount-point">/prueba/</property>
    <property name="port">8800</property>
    <property name="type">slave</property>
    <property name="porter-socket-path">flu-pOoSZR.socket</property>
    <property name="porter-password">vCslZJcBAUMk</property>
    <clock-master>false</clock-master>
    <plugs>
    </plugs>
  </component>

</flow>
```

```
</planet>
```

En este ejemplo podemos observar algunos cambios respecto al anterior. Aquí, se ha configurado para que el vídeo a emitir esté codificado en el codec VP8, el audio en Vorbis y el formato contenedor WebM. Podemos ver diferencias en " http-server-audio-video", donde se indica que el tipo mime es webm, y se indica al reproductor que los codecs son vorbis y vp8.
Otro componente con cambios es el "encoder-video". Aquí podemos ver que el tipo es "vp8-encoder" ya que vamos a codificar en VP8. Son pocas las diferencias entre una configuración y otra, solo cambian las opciones a lo que el vídeo se refiere.

Conexión con un servidor Icecast

Un ejemplo de componente del tipo "shout2-audio-video", encargado de enviar el flujo a un servidor Icecast.

```
<component name="shout2-audio-video"
            type="shout2-consumer"
            label="shout2-audio-video"
            worker="localhost"
            project="flumotion"
            version="0.10.0">
    <eater name="default">
      <feed alias="default">muxer-audio-video:default</feed>
    </eater>
    <property name="short-name">source</property>
    <property name="description"></property>
    <property name="url">http://localhost/</property>
    <property name="mount-point">/prueba.ogg</property>
    <property name="ip">127.0.0.1</property>
    <property name="password">hackme</property>
    <property name="port">8000</property>
    <clock-master>false</clock-master>
  </component>
```

Entre las propiedades podemos ver "short-name", que es el el nombre de usuario de Icecast, con la contraseña "hackme" en "password". En "ip" ponemos la IP del servidor Icecast, que en este caso está alojado en la máquina local con Flumotion. En "port" ponemos el puerto que tengamos configurado en Icecast, en este caso el 8000. Y en "url" ponemos una URL que se va a asociar al flujo.

Si lo configuramos con el administrador gráfico, debemos activar la casilla correspondiente en "consumo" e introducir los datos del servidor Icecast, tales como el nombre o IP del servidor, puerto, punto de anclaje (mount point), usuario, contraseña entre otras.

Guardar el vídeo en disco a la vez

Podemos guardar lo que estamos emitiendo al mismo tiempo que emitimos con el tipo de componente “disk-consumer”. Podemos ver un ejemplo aquí:

```
<component name="disk-audio-video"
           type="disk-consumer"
           label="disk-audio-video"
           worker="localhost"
           project="flumotion"
           version="0.10.0">
    <eater name="default">
      <feed alias="default">muxer-audio-video:default</feed>
    </eater>
    <property name="directory">/var/streaming</property>
    <property name="rotate-type">none</property>
    <property name="start-recording">False</property>
    <clock-master>false</clock-master>
    <plugs>
    </plugs>
  </component>
```

En este componente, indicamos la ruta donde queremos guardar el vídeo, en este caso */var/streaming*. El vídeo se guardará con la extensión que le hayamos dado en el componente “muxer”.
En la propiedad “rotate-type” podemos poner “size, “time” o “none”. Por defecto es “none”.
La propiedad “start-recording” consiste en comenzar a grabar en el momento en que se activan los componentes. Podemos poner true o false. En el caso de false, comenzará a grabar después de que los componentes hayan arrancado.

Otra utilidad de Flumotion es que podemos ver las propiedades de los componentes desde un terminal de comandos introduciendo el siguiente comando:

flumotion-inspect [tipo de componente]

Por ejemplo, para ver las propiedades del “webcam-producer” introducimos:

flumotion-inspect webcam-producer

El reproductor de vídeo de HTML5

Este reproductor se creó para sustituir al veterano reproductor flash. En HTML5 se puede incluir con la etiqueta <video>. Durante el proyecto hemos comprobado algunos ejemplos del reproductor, aunque se pueden incluir más opciones:

Ejemplo anterior:

<video src=”http://localhost:8080/custome/first” controls></video>

Podemos ver que se pone como src (origen) una URL. Se puede poner cualquier flujo de datos, desde un servidor Icecast, Flumotion, VLC, etc. La opción “controls” nos permite poner los controles del reproductor como play/pause, volumen, etc. También existen otras opciones:

width=”nº”: ancho del vídeo.

height=”nº”: Altura del vídeo.

autoplay: Reproduce automáticamente al cargar la página.

preload: Carga el vídeo durante un corto tiempo antes de comenzar la reproducción.

Referencias

http://es.wikipedia.org/wiki/Streaming
http://es.wikipedia.org/wiki/V%C3%ADdeo_bajo_demanda
http://es.wikipedia.org/wiki/VLC_media_player
http://www.gnu.org/software/gnump3d/
http://jmengual.blogspot.com/2008/11/gnump3d-en-linux.html
http://html5facil.com/tutoriales/uso-basico-de-la-etiqueta-video-en-html5

- Icecast:

http://www.icecast.org/docs/icecast-2.3.2/
http://www.icecast.org/
http://es.wikipedia.org/wiki/Icecast

- FFMPEG:

http://es.wikipedia.org/wiki/FFmpeg
http://ffmpeg.org/

- Video4linux:

http://es.wikipedia.org/wiki/Video4Linux

- ALSA:

http://www.alsa-project.org/main/index.php/Main_Page
http://es.wikipedia.org/wiki/Arquitectura_de_Sonido_Avanzada_para_Linux

- WebM:

http://www.webmproject.org/

- Flumotion:

http://es.wikipedia.org/wiki/Flumotion
http://www.flumotion.net/
http://www.flumotion.net/doc/flumotion/manual/en/trunk/html/
http://blog.zarovich.org/index.php/2011/07/08/flumotion-en-debian-bajo-demanda-con-autenticacion/