Servidor de corrents d'àudio i vídeo basat en programari i formats lliures

Alumna: Angela Abad Román

Tutor: Xavier Giró

Departament: Teoria del Senyal i Comunicacions

Enginyeria Tècnica de Telecomunicacions especialitat en Imatge i So

Escola Universitària d'Enginyeria Tècnica Industrial de Terrassa

UNIVERSITAT POLITÈCNICA DE CATALUNYA

ÍNDEX DE CONTINGUTS

Capítol 1.
INTRODUCCIÓ

El projecte fi de carrera que es presenta a continuació, dels estudis d'Enginyeria Tècnica de Telecomunicacions especialitat en Imatge i So, ha estat realitzat de Gener a Juny de 2007, al Laboratori del departament de Teoria del Senyal i Comunicacions (TSC), de l'Escola d'Enginyeria Tècnica Industrial de Terrassa (EUETIT) de la Universitat Politècnica de Catalunya (UPC), sota la supervisió del professor Xavier Giró.

Tenint en compte que tot el sector audiovisual tendeix a la digitalització de tots els continguts, i a la personalització i interacció amb aquests. Existeixen dues línies de difusió, el contingut en directe (temps real) i el contingut sota demanda.

La idea inicial d'aquest projecte era ampliar el treball previ fet per Stefano Mosca al seu projecte "*Live streaming and peer to peer television*", on tractava temes de descàrrega de vídeo i de difusió de corrents en directe, però centrant-lo en les emissions de corrents en directe. El projecte havia d'utilitzar solucions de programari lliure i formats lliures, on es buscava connectar el reproductor multimèdia VideoLan, amb Internet, servint com a font emissora del corrent.

Finalment, s'ha utilitzat un ordinador de proves del laboratori, que conté tot el programari necessari, per assolir els objectius del projecte. La descripció del projecte i els passos a seguir es detallen en aquesta memòria.

Voldria agrair en els resultats d'aquest projecte a totes les persones que han estat al meu costat. Als amics, per saber-me distreure en moments d'estres. Al tutor del projecte, en Xavier Giró, per implicar-se tant en el tema. A l'Albert Márquez, per l'ajuda prestada al laboratori. I sobretot a la meva família, que encara que no l'he tinguda aprop, sempre han estat amb mi.

Capítol 2.
ESTAT DE L'ART

TIPUS DE PROGRAMARI

El programari, és la part lògica de l'ordinador, és a dir, el conjunt de programes i instruccions que regeixen el funcionament del maquinari.

PROGRAMARI LLIURE

El programari lliure és el programari que, un cop obtingut, pot ser utilitzat, copiat, estudiat, modificat i redistribuït lliurement. El programari lliure acostuma a estar disponible gratuïtament a Internet, o a un baix preu si l'adquirim per mitjà d'altres medis (CD-Rom, DVD, disquets...); tot i així, no és obligatori que sempre passi així i, encara que conservi la seva filosofia lliure, pot ser venut comercialment.

D'altra banda, el programari gratuït (en anglès *freeware*) inclou en algunes ocasions el seu codi font, encara que no sigui lliure, a diferència del programari lliure, ja que no se'ns asseguren els drets a la modificació i redistribució del programa.

En general, es pot dir que un programa és lliure si permet les quatre llibertats definides per la Free Software Foundation[1]:

- La llibertat d'executar el programa per qualsevol propòsit *(llibertat 0).*
- La llibertat de veure com funciona el programa i adaptar-lo a les necessitats pròpies *(llibertat 1)*. L'accés al codi font és un requisit.
- La llibertat de redistribuir còpies *(llibertat 2)*
- La llibertat de millorar el programa i de distribuir-lo de nou amb les millores realitzades, per tal que tota la comunitat se'n pugui beneficiar *(llibertat 3)*. Igual que a la llibertat 1, l'accés al codi font és un requisit.

La GPL[2], Llicència Pública General, és un tipus de llicència per programari que permet la copia, distribució (comercial o no) i modificació del codi, sempre que qualsevol modificació es continuï distribuint amb la mateixa llicència GPL. La llicencia GPL no permet la distribució de programes executables sense el codi corresponent.

Codi Obert

El codi obert és el programari que el codi font està públicament disponible. Permet que qualsevol pugui mirar el codi, però no fa necessàriament que el codi segueixi essent obert en futures redistribucions, propietat que si que té el programari lliure.

1. http://www.fsf.org/
2. General Public License

PROGRAMARI PRIVATIU

El programari privatiu (també conegut com a programari de propietat o programari no lliure), en contraposició al programari lliure, es refereix a aquell programari que limita les seves possibilitats d'ús, modificació i/o redistribució (amb o sense modificacions). Normalment el seu codi font no és disponible, o bé, sota restriccions.

La Free Software Foundation s'hi refereix a tot aquell programari que és "no lliure" o semi-lliure.

Encara que el codi font pugui fer-se públic, el programari pot ésser privatiu si es mantenen les restriccions abans esmentades.

FORMATS DE CODIFICACIÓ

Avui en dia, es comprimeixen les dades de vídeo o d'àudio, amb els respectius còdecs, ja que la forma original d'aquestes dades son massa grans per als sistemes de comunicació actuals. Aquest còdecs, poden ser lliures o privats.

ÀUDIO

Els còdecs d'àudio s'encarreguen de realitzar les transformacions necessàries en un senyal d'àudio per tal de reduir el seu volum, comprimint les dades. A continuació farem una comparació entre el MP3 i el Vorbis.

Mp3

L' MP3 o MPEG-1 Layer 3 és un format de so propietari famós perquè ha tingut un gran èxit en l'intercanvi d'arxius de so a través d'Internet, amb l'aparició d'aplicacions de compartició de fitxers. Permet una àmplia gamma de qualitats de so (taxes de bits) amb variades freqüències de mostra. Actualment és el format de compressió de so més estès per Internet.

Vorbis

L'Ogg Vorbis és totalment obert, sense propietari, lliure de patent i d'atribucions, desenvolupat per Xiph.org com a part del projecte Ogg. Adequat per a àudio de mitjana a alta qualitat a taxes de transferència fixes i variables. Ofereix una qualitat bastant superior a MP3, WMA, RealAudio i VQF ocupant el mateix espai.

VÍDEO

La majoria de còdecs comprimeixen la informació, per a que pugui ser emmagatzemada o transmesa amb el mínim espai possible, per aconseguir-ho s'aprofita que les seqüències de vídeo presenten redundància en les dimensions espacial i temporal. Per tant eliminant informació redundant s'aconsegueix codificar la informació més optimitzadament.

MPEG

El Grup d'experts en imatges en moviment (Motion Picture Expert Group) era originàriament un grup encarregat del desenvolupament de normes de codificació per a àudio i vídeo, de l'ISO[3].

MPEG-1

Estàndard inicial per a la compressió d'àudio i vídeo. Esdevingué norma pels CD de vídeo i inclou el popular format de compressió d'àudio MP3.

MPEG-4

Conté moltes de les característiques de MPEG-1 i MPEG-2, i altres estàndard relacionats, com el suport de VRML[4], estès per a Visualització en 3D, arxius compostos en orientació a objectes, suport per a la gestió de Drets Digitals externs i alguns tipus d'interactivitat.

Theora

Theora és un còdec de vídeo lliure desenvolupat per la fundació Xiph.org com a part del projecte Ogg. Està basat en el còdec VP3, el codi del qual la companyia On2 Tecnologies va donar-los-hi sota una llicència BSD. La fundació l'ha refinat i ha ampliat donant-li el mateix abast futur que el còdec d'àudio Vorbis.

3. Organització Internacional per a l'Estandardització
4. VRLM (Virtual Reality Modeling Language)

CONTENIDORS

Els formats d'arxiu poden contenir diferent informació codificada amb diversos còdecs, només son contenidors d'informació.

Mp4

Extensió oficial per a la nova generació d'arxius MPEG-4. Emmagatzemaran diferents tipus de dades, des de música a imatges, i la idea és intentar ser un format únic, en el que es podria inclús emmagatzemar dades de diferents tipus en un mateix arxiu. Els formats que componen un MP4 estàndard són:

- **So:** MP3, AAC i Apple Lossless com a principals
- **Vídeo:** MPEG-4, MPEG-3 i MPEG
- **Imatge:** JPG i PNG
- **Subtítols:** XMT i BT

Ogg

Es un format de contenidor multimèdia desenvolupat per la Fundació Xiph.org i és el format nadiu per als còdecs multimèdia que també desenvolupa la mateixa fundació.

El format és lliure de patents i obert, igual que tota la tecnologia de Xiph.org. Fou dissenyat per a donar un alt grau d'eficiència en la reproducció en temps real i en la compressió de fitxers.

Mètodes de Difusió

Actualment existeixen diferents tipus de difusió de les dades per Internet que és necessari conèixer-les, ja que ens són de molta utilitat per a detallar millor després

UNICAST

Els paquets s'envien d'una màquina a una altra. S'estableix una connexió directa i única, entre el servidor i el client que demana les dades. El servidor dedica l'amplada de banda, que dependrà de les dades a transmetre, si és una quantitat elevada de dades com pot ser la difusió de vídeo, serà necessària una gran amplada de banda.

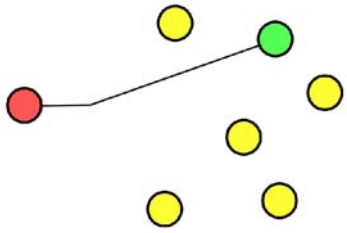

Figura 1: Mètode unicast

MULTICAST

És un protocol que funciona mitjançant IP. La transmissió per Multicast permet a una única màquina, la comunicació amb un conjunt especific de màquines, que no estan definits amb una combinació d'adreça IP i màscara estàndard. És obligatori obtenir una adreça IP multicast de la IANA per poder assegurar un canal estable, o la comunicació estarà limitada a una LAN local. Aquest sistema podem tenir problemes amb l'amplada de banda del servidor, degut que, a més usuaris necessitem més amplada.

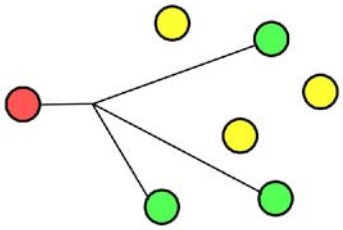

Figura 2: Mètode multicast

PEER TO PEER

La transmissió de manera Peer to Peer, o P2P, és un mètode molt nou, que encara esta en desenvolupament en algunes àrees. Aquest mètode consisteix en que un servidor es comunica amb altres màquines, i aquestes a la vegada s'ofereixen per a difondre les dades, d'aquesta manera la xarxa augmenta molt ràpidament, a cada nou usuari que rep informació es converteix en servidor. És un mètode que no ens exigeix una amplada de banda excessiva. Encara que sempre que s'il·lustra el funcionament, no es te en compte, que hi ha una connexió continua amb el servidor, ja que es qui guarda les dades de cadascun dels usuaris connectats, i per tant ha d'informar als nous usuaris amb quina màquina es poden connectar per a obtenir les dades transmeses.

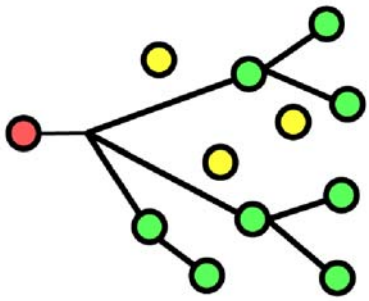

Figura 3: Mètode Peer to Peer

Corrents en Temps Real

El fet d'enviar les dades en Streaming, no es més que crear un flux continuu de dades, d'aquesta manera el client pot veure el fitxer, com vídeo o àudio, mentre s'està transmetent.

A continuació presentarem el programari existent referent al mètode Streaming, diferenciant els mètodes de difusió Unicast i Multicast, que els considerarem el mateix, i el mètode Peer to Peer.

DIFUSIÓ AMB UNICAST/MULTICAST

IceCast

IceCast és un projecte de programari lliure, de Xiph.org Foundation, primerament dedicat a la difusió d'àudio, que a partir de la versió 2, permet la difusió de vídeo.

Part del projecte es un servidor, capaç de difondre continguts com Vorbis, per a àudio, i Theora, per a vídeo, sobre el protocol HTTP. Aquest servidor munta una pàgina web per a consultar l'estat del servidor, gestionar les connexions, etc.

Es necessita un programa extern, com a font del flux de vídeo, com el FreeJ o el VideoLan, en el cas de voler crear un flux d'àudio hi han molts programes externs disponibles, depenent del Sistema Operatiu que fem servir.

ShoutCast

ShoutCast és una tecnologia desenvolupada per Nullsoft per a l'enviament d'àudio en temps real. Utilitza el protocol HTTP o multicast per a difondre ràdio per Internet, amb codificació d'mp3.

El format de sortida que li arriba al client, es pot llegir des de programes com el propi Nullsoft WinAmp, Apple iTunes o Windows Media Player.

Actualment s'està utilitzant com a radio, a la pàgina web del FIB, el Festival Internacional de Benicàssim[5]

5. http://radio.fiberfib.com

VLC

VideoLan és un projecte de programari lliure, que produeix programari per a la recepció i enviament de vídeo, alliberat sota el GNU, General Public License , funciona en moltes plataformes. Inclou característiques molt importants, que ens permeten crear corrents multicast, des d'un arxiu, un disc o també des d'un webcam (en aquest cas és necessari utilitzar Linux i haver instal·lat Vídeo 4 Linux). De fet l'última versió (0.86) dóna suport a la distribució de corrents pel mètode Shout cap a un servidor, com per exemple el IceCast.

DIFUSIÓ DEL SENYAL PEL MÈTODE PEER TO PEER

Octoshape

Octoshape es un connector, de codi tancat, que afegim al nostre ordinador per obtenir amb una connexió punt a punt, àudio, vídeo i dades.

Està basat en el llenguatge de programació Java, funciona de manera estable amb Windows XP, i existeixen versions beta per a GNU/Linux i Mac OS X. La instal·lació amb GNU/Linux és molt complexa, i no ens assegura un funcionament correcte. Amb Mac OS X, és més senzill, però existeix un requeriment indispensable, només funciona amb Windows Media Player per a Mac.

Aquest programa fa de gestor del flux, i l'envia cap al reproductor predeterminat, per a la seva previsualització

Sopcast

Sopcast utilitza la tecnologia Peer to Peer, per a difondre àudio i vídeo, el cor del projecte és el protocol de comunicacions produït per el Grup de SopCast, que és anomenat Sop://. Es tracta d'una aplicació que només funciona amb el Sistema Operatiu de Windows, que incorpora la implementació d'una guia dels canals disponibles, agrupats en diferents grups.

Flumotion

Flumotion es un servidor d'enviament de vídeo en temps real, creat per l'empresa Fluendo, especialitzada en la creació de productes i serveis de consulta per als continguts multimèdia de Unix.

Presenta eines d'administració gràfiques intuïtives, fent fàcil la tasca d'establir i manipular àudio i corrents de vídeo per a fins i tot administradors de sistema de novicis. Flumotion s'allibera sota el GPL.

Actualment, la TV3 està utilitzant aquest servei, per a la transmissió en directe.

Figura 4: La TV3 per Internet amb Flumotion

PeerCast

PeerCast és una eina de difusió d'àudio i vídeo, de codi obert. Transmet àudio (Ogg Vorbis, MP3, WMA) i vídeo (Ogg Theora, Nullsoft Streming Video, WMV) per Internet.

L'aplicació està disponible per a els tres Sistemes Operatius més populars, Windows, Mac OSX, GNU/Linux.

El programari client té l'habilitat de servir pàgines web als navegadors normals com Mozilla i Internet Explorer. Aquest recurs fa que la gent dintre d'un entorn LAN pot buscar i escoltar canals sense haver d'instal·lar el programari de client en el seu PC. Les oficines poden tenir un client de PeerCast que proporciona corrents d'àudio al LAN sencer. O es pot crear una xarxa privada amb els amics en Internet per escoltar música.

CoopCast

És un projecte nou, que encara està en desenvolupament, que permet enviar i rebre àudio o vídeo en una xarxa peer to peer. Consisteix de dos programes:

CoopCastPeer és un programa simple que els clients han de tenir, per a poder participar en la xarxa peer to peer i poder rebre el flux.

CoopCastTracker és un programa per controlar i gestionar les connexions entre les màquines S'ha d'instal·lar en un ordinador constantment connectat a la Internet i s'ha de mantenir constantment actiu.

IceShare

IceShare és una biblioteca que distribueix corrents Ogg en una xarxa pseudo P2P. Es basa fortament en BitTorrent, però funciona a nivell d'Ogg, i a diferència de PeerCast funciona tant amb arxius, com amb corrents continus.

Ha estat dissenyat per a que l'accés als corrents sigui mitjançant un enllaç web, a icet://. No per a la cerca en xarxes P2P, com Gnutella, Kazaa o eDonkey. Encara que les webs contindran un cercador de continguts als servidors IceTracker.

És un projecte de Xiph.org Foundation, encara en desenvolupament, que pretén ser com l'IceCast, en multicast, però amb el mètode de difusió P2P.

TAULES COMPARATIVES

En aquest apartat analitzarem les característiques tècniques de cadascuna de les opcions presentades abans.

DIFUSIÓ AMB UNICAST/MULTICAST

	Cod.Àudio	Cod.Video	Plataforma			Codi	Llicència
			Windows	GNU/Linux	Mac OSX		
IceCast	Ogg Vorbis, MP3,	Ogg Theora	✔	✔	✘	Tancat	GPL
ShoutCast	MP3	-	✔	✔	✔	Tancat	Freeware
VLC	AAC, AC3, DV Audio, FLAC, MP3, Speex, Vorbis	H.263, H.264, MJPEG, MPEG-1, MPEG-2, MPEG-4 P2, VP5, VP6, Theora, RealVideo, DV	✔	✔	✔	Obert	GPL

DIFUSIÓ DEL SENYAL PEL MÈTODE PEER TO PEER

	Cod.Àudio	Cod.Video	Plataforma			Codi	Llicència
			Windows	GNU/Linux	Mac OSX		
Octoshape	Ogg Vorbis, MP3, WMA	MPEG-4, Nullsoft Video, or WMV and other codecs	✔	✔	✔	Tancat	Propietari
Sopcast	MP3	asf, wmv, rm, rmvb	✔	✔	✘	Tancat	Propietari
Flumotion	mp3, Ogg Vorbis	wmv, flv, Ogg Theora	✔	✔	✘	Obert	Lliure
PeerCast	Ogg Vorbis, MP3, WMA	Ogg Theora, Nullsoft Streming Video, WMV	✔	✔	✔	Tancat	GPL
CoopCast	-	-	✔	✘	✘	Obert	GPL
IceShare	Ogg Vorbis	Ogg Theora	-	-	-	Obert	GPL

Capítol 3.
REQUERIMENTS

El professor Xavier Giró, va voler investigar en els temes tractats per Stefano Mosca, pel que fa a la distribució de dades multimèdia en temps real per Internet. Aquest fet podria ser aprofitable per VilaWeb TV, per fer emissions en directe des del portal web, ja que avui en dia, utilitzen un sistema de podcast.

Per tant els requisits que es plantegen son els següents:

- Un sistema basat en programari i formats lliures, d'aquesta manera esta a l'abast de tothom.

- La solució ha de ser reproduïble des de qualsevol dels sistemes operatius més destacats, ja sigui un sistema basant en GNU/Linux, un MacOS o un Windows.

- Emissió d'un corrent continu en temps real del que passa al Laboratori, amb una Webcam, al que tothom pugui tenir accés,

- Integració dels corrents en un portal web, on la seva reproducció pugui ser possible amb la instal·lació d'algun connector, i d'aquesta manera sigui de fàcil utilitat per l'usuari.

- La distribució de les corrents hauria de ser possible per un mètode unicast/multicast, com per un mètode peer to peer, per tal de no sobrecarregar la xarxa.

Capítol 4.
INSTAL·LACIÓ I CONFIGURACIÓ

Sistema Operatiu

El Sistema Operatiu (SO), que s'ha fet servir per a la instal·lació i configuració del programari, ha estat una de les distribucions de GNU/Linux, concretament Kubuntu 7.04 Feisty Fawn.

En canvi, la recepció del corrent es possible des de qualsevol altre Sistema Operatiu, ja sigui Windows, MacOS o GNU/Linux. Degut a que l'aplicació que s'utilitza per rebre'l, és el VideoLan, que actualment es pot instal·lar a qualsevol dels SO abans mencionats.

Totes les ordres que es donen a continuació s'executen des del Terminal de Kubuntu.

IceCast

Utilitzarem IceCast com a servidor de distribució dels corrents d'àudio i de vídeo. A continuació s'explicarà el procediment per a la correcta instal·lació i configuració del mateix.

INSTAL·LACIÓ

Abans de començar la instal·lació de l'IceCast, haurem de tenir instal·lats altres llibreries i paquets:

- Instal·lació del paquet: *libxml2*

```
sudo apt-get install libxml2-dev
```

- Instal·lació del paquet: *libxslt*

```
sudo apt-get install libxslt-dev
```

- Instal·lació del paquet: *oog/vorbis*

```
sudo apt-get install libalogg-dev
```

- Instal·lació del servidor d'Icecast.

```
sudo apt-get install icecast-server
```

- Procediu amb la instal·lació de l'Icecast2.

```
sudo apt-get install icecast2
```

- Canvieu els permisos de la carpeta, per que l'usuari pugui editar l'interior. *Les xxxx s'han de substituir pel nom d'usuari*.

```
sudo chown -R xxxx:users /etc/icecast2/
```

- Feu una còpia del fitxer `icecast.xml`, en un directori creat per l'usuari.

```
mkdir /home/.../icecast/
cp /etc/icecast2/icecast.xml /home/.../icecast.xml
cd /home/.../
```

- Editeu el fitxer, segons les instruccions de configuració que s'expliquen més endavant. *(saltar a l'apartat de configuració)*

```
kate icecast.xml
```

Haureu de crear dos fitxers buits, per a que IceCast generi els seus logs:

- Canvieu els permisos de la carpeta dels logs, per que l'usuari tingui accés.

```
sudo chown -R xxxx:users /var/log/icecast2/
```

- Aneu al directori

```
cd /var/log/icecast2/
```

- Creeu els dos arxius en blanc.

```
touch error.log
touch access.log
```

- L´últim pas es muntar el servidor IceCast:

```
icecast2 -c /home/.../icecast.xml
```

- Comprovació del funcionament del servidor

```
http://IP_servidor:8000
```

- Es carregarà la següent plana web:

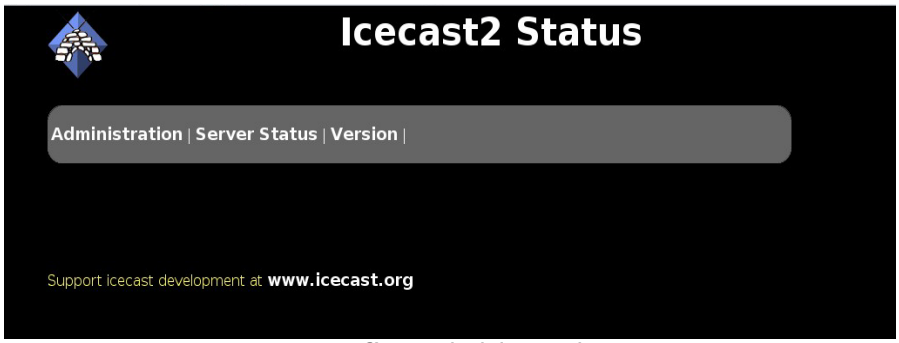

Figura 5: Intefície web del servidor IceCast

CONFIGURACIÓ

L'icecast.xml és la principal eina de configuració del servidor i conté diferents apartats que cal especificar, segons els nostres requisits. Es tracta d'un fitxer XML que es pot modificar amb qualsevol editor de text. A continuació s'expliquen cadascuna de les parts més rellevants. El fitxer complet es troba a l'Annex I.

Límits

Aquesta part ens permet configurar les dades d'accés al nostre server IceCast,

```
<limits>
    <clients>10</clients>
    <sources>2</sources>
    <threadpool>5</threadpool>
    <queue-size>524288</queue-size>
    <client-timeout>30</client-timeout>
    <header-timeout>15</header-timeout>
    <source-timeout>10</source-timeout>
    <burst-on-connect>1</burst-on-connect>
    <burst-size>65535</burst-size>
</limits>
```

Clients

Nombre màxim de clients als que permetem la connexió al nostre servidor. Considerarem a qualsevol que estigui escoltant el nostre corrent d'àudio o vídeo com a client, fins i tot als que estiguin connectats a la plana web. Aquest nombre l'haurem de modificar segons les nostres limitacions d'amplada de banda.

Sources

Màxim nombre de fonts connectades al servidor. Les fonts son els corrents que envia el client cap al servidor, per a que siguin accessibles, cada corrent es considera una font.

Threadpool

És el nombre de fils que s'engeguen per suportar les connexions dels clients. El valor que hi ha per defecte es per al suport d'un trànsit del servidor petit o mitjà, si es vol tenir un trànsit alt al servidor, haurem d'augmentar aquest valor.

Queue-size

És el mida màxima (en bytes) d'un client (oient) que esta en cua. Un oient es pot temporalment endarrerir a causa de congestió de xarxa, en aquest cas una cua interna es manté per a cada oient. Si la cua es fa més gran que aquest valor de configuració, llavors el servidor donarà de baixa l'oient del corrent que escoltava.

Client-timeout

És el màxim temps (en segons) que pot estar un client sense escoltar cap font, fins que es consideri com a no-client.

Header-timeout

El temps màxim (en segons) d'espera a l'entrada d'una petició, una vegada el client ha fet la connexió amb el servidor.

Source-timeout

Si una font (?) no envia cap mena de dades en un període de temps (en segons), la connexió de la font es traurà del servidor.

Autenticació

En el següent apartat configurarem les dades d'accés de tots els usuaris i contrasenyes, ja sigui per tasques administratives o per a l'accés a les fonts i les transmissions.

```
<authentication>
    <source-password>hackme</source-password>
    <relay-user>relay</relay-user>
    <relay-password>********</relay-password>
    <admin-user>admin</admin-user>
    <admin-password>********</admin-password>
</authentication>
```

Source-password

Contrasenya sense xifrar que utilitzen les fonts per a connectar-se a l'Icecast2. Per defecte, el nom d'usuari per a totes les connexions ha de ser `source`.

Relay-user i Relay-password

Suposant que es crea una xarxa de dos maquines, una d'elles es el servidor mestre i l'altre actua com a esclau. Aquestes dades s'utilitzen en el servidor mestre com a part de l'autenticació quan un esclau demana la llista de corrents de dades a transmetre. El nom d'usuari per defecte es `relay`.

Admin-user i admin-password

El nom d'usuari i la contrasenya que s'emprarà per a tasques administratives. Tenint accés a les estadístiques del servidor, i a les pantalles d'administració de la web.

Configuració Del Servidor

Aquest apartat el dedicarem a la configuració del servidor, adreça IP, ports d'accés, ...

```
<hostname>XXX.XXX.XXX.XXX</hostname>

<!-- You can use these two if you only want a single
listener
  <port>8000</port>
  <bind-address>127.0.0.1</bind-address>-->

<!-- You may have multiple <listener> elements -->
  <!--<listen-socket> -->
       <port>8000</port>
       <!--bind-address>127.0.0.1</bind-address> -->
  <!--</listen-socket>-->
  <!--
     <listen-socket>
       <port>8001</port>
     </listen-socket>
   -->
```

Hostname

Aquest és el nom de la màquina o adreça IP que s'utilitzarà per connectar amb el nostre servidor.

Port

El port TCP que permet establir les connexions dels clients amb el servidor. Es poden establir múltiples connexions on s'haurà d'especificar els ports TCP d'accés.

Bind-address

Una adreça IP opcional que es pot utilitzar per vincular a una targeta de xarxa específica.

PeerCast

Utilitzarem el PeerCast com a servidor de distribució dels corrents d'àudio en una xarxa P2P i repetidor dels corrents. A continuació s'explicarà el procediment d'instal·lació. S'instal·larà sobre un dels sistemes operatius GNU/Linux, concretament, la distribució Kubuntu 7.04 Feisty Fawn, sobre Windows i MacOS

GNU/Linux

- Instal·leu el paquet *peercast*

```
apt-get install peercast
```

Windows o MacOS

- Aneu a la pàgina web de PeerCast[6] i descarregueu l'instal·lador.
- Seguiu les instruccions.

Ices

En un principi utilitzarem l'Ices com a programa de font client de corrents d'àudio per a IceCast. Nomes funciona amb GNU/Linux.

- Instal·leu el paquet

```
apt-get install ices2
```

- Creeu directori per a guardar els logs.

```
sudo mkdir /var/log/ices
```

- Canvieu els permisos del directori creat.

```
sudo chown -R xxxx:users /var/log/ices/
```

- Creeu fitxer de log.

```
touch /var/log/ices/ices.log
```

- Moveu els fitxers de configuració

```
mkdir /home/.../ices
cp /usr/share/doc/ices2/examples/ices-alsa.xml /home/.../ices/
cp /usr/share/doc/ices2/examples/ices-oss.xml /home/.../ices/
cp /usr/share/doc/ices2/examples/ices-playlist.xml /home/.../ices/
```

6. http://www.peercast.org/download.php

VideoLan

Tindrem dues instal·lacions possibles de VLC, com a client reproductor (com una aplicació independent o com un connector del Firefox) o com a font de vídeo i àudio.

Windows

- Aneu a la pagina web de VideoLan[7] i descarregueu-lo
- Seguiu les instruccions d'instal·lació
- En cas de voler el connector pel Firefox, marqueu la casella que ho indica.

GNU/Linux

- Instal·lació VLC com a client reproductor

```
sudo apt-get install vlc
```

- Instal·lació VLC amb connector

```
sudo apt-get install vlc vlc-plugin-* mozilla-plugin-vlc
```

Aquesta instal·lació no conté en les opcions per defecte, el mòdul que necessitem per fer servir VLC com a client font d'IceCast. El mòdul `access_output_shout` ha d'estar activat.

- Per veure si el mòdul Shout està activat

```
vlc -l | grep shout
```

- ens apareix a continuació

```
VLC media player 0.8.6 Janus
  playlist       New winamp 5.2 shoutcast import
  shout          Shoutcast radio listings
  shout          Shoutcast TV listings
```

Com que no ens apareix, haurem d'instal·lar VLC a partir d'una recompilació del codi font. Per tal de no estar pendent d'actualitzacions de programari, farem una instal·lació totalment separada i independent de la que ja tenim.

7. http://www.videolan.org

- Descarregueu el codi font de l'última versió fins al moment (0.8.6b)

```
wget http://download.videolan.org/pub/videolan/vlc/0.8.6b/vlc-0.8.6b.tar.gz
mv vlc-0.8.6b.tar.gz /usr/local/src/.
cd /usr/local/src
tar xvfz vlc-0.8.6b.tar.gz
```

- Col·loqueu-vos al directori on l'heu descomprimit

```
cd /usr/local/src/vlc-0.8.6b/
```

- Instal·leu els paquets necessaris per compilar el Vlc:

```
apt-get build-dep vlc;
```

Existeixen múltiples opcions, mòduls i codecs disponibles a l'hora de compilar VideoLan, podeu veure'ls fent: `./configure -help` . En aquest projecte es van utilitzar els següents mòduls: códec theora, el modul v4l per accedir a dispositius de captura, el mòdul dvb en el cas que es volgués accedir a la tarja de TV, i mòdul shout, que es el que ens permet servir de client font a IceCast.

- Compileu l'aplicació amb els mòduls descrits.

```
./configure  --prefix=/opt/vlc-0.8.6b/  --exec-prefix=/opt/vlc-0.8.6b/ --enable-theora --enable-shout --enable-v4l --enable-dvb
make
sudo make install
```

- Comproveu ara que el mòdul Shoutcast està instal·lat

```
./vlc -l | grep shout
VLC media player 0.8.6a Janus
  access_output_shout    IceCAST output
  playlis                New winamp 5.2 shoutcast import
  shout                  Shoutcast radio listings
  shout                  Shoutcast TV listings
```

- Executeu vlc

```
./vlc
```

- S'obrirà la seva interfície gràfica.
- Feu clic a 'Paràmetres'-> 'Preferències'
- Al menú de l'esquerra seleccioneu '*Corrent de Sortida*' -> '*Sortida d'Accés*' -> '*Shoutcast*'. Aquí es pot configurar el nom del corrent i la seva descripció.

Altres Instal·lacions

FFMPEG2THEORA

Aquest paquet es necessari a l'hora de codificar els fitxers MPEG o AVI a OGG Theora. Serà utilitzat al Script que s'ha generat per la conversió de formats[8].

```
sudo apt-get install ffmpeg2theora
```

MENCODER

Aquest paquet, al igual que l'anterior, s'utilitza al Script de conversió de formats, en aquest cas convertirà un arxiu en MP4, a MPEG

```
sudo apt-get install mencoder
```

Logitech QuickCam Express

Per a tenir un bon funcionament de la càmera web, serà necessari instal·lar els controladors.

GNU/Linux

- Instal·leu el paquet: *spce5xx*

```
sudo apt-get spce5xx-source
```

Windows

- Inseriu el CD que ve amb la webcam, on es troba el controlador.
- Connecteu la webcam, i seguiu les instruccions de l'instal·lador.

8. Consultar Annex II

Configuració de la Xarxa

L'entorn de treball del projecte consta d'un ordinador com a servidor central amb un tallafoc que ens fa la funció de NAT[9]. A aquest servidor es troben connectats tots els PC existents de la LAN, un d'ells amb el que s'ha estat treballant. Cadascun dels PCs tenen accés directe des de l'exterior, és a dir, una adreça IP pública.

Per tal d'obrir els ports s'ha emprat la funció `iptables`:

```
iptables -t filter -A FORWARD -d XXX.XXX.XXX.XXX -p tcp -m multiport
--dport
```

Els ports que s'han obert son els següents:

8000 i 8001 ---> IceCast

7144 i 7144 ---> PeerCast

9 NAT (Traducció d'adreça de xarxa) és un mecanisme utilitzat pels routers IP, que reescriu les adreces de la font i/o destí dels paquets quan passen per ell. La majoria dels sistemes que utilitzen NAT fan això per permetre que múltiples usuaris d'una xarxa privada puguen accedir a Internet utilitzant una única adreça IP pública.

Capítol 5.
CASOS D'ÚS

En aquest projecte es va començar a investigar primerament en l'àmbit de l'enviament de l'àudio, on no ens va aparèixer cap mena de complicacions de maquinari. En canvi, al continuar experimentant amb vídeo no vam obtenir bons resultats degut a la màquina amb la que treballàvem no era suficientment potent, per realitzar les codificacions dels vídeos i després enviar el corrent cap al servidor.

Es va fer un canvi de màquina on s'ha experimentat que el problema el teníem realment en la potència de la màquina.

A continuació fem una descripció del maquinari amb el que s'ha treballat:

CPU	Pentium IV a 3,2 Ghz
RAM	4GB
Disc Dur	160 GB Serial Ata a 7200 rpm
Tarja Gràfica	PCI express de 256 Mb ATI

Enviament d'Àudio

Aquesta secció explica com fer funcionar el programari que hem instal·lat i configurat a l'apartat anterior per l'enviament d'àudio. Sempre es necessitarà una aplicació font, que en el nostre cas, serà l'Ices o el VLC. La font servirà el corrent d'àudio cap al servidor, IceCast per connexions Multicast, o PeerCast per una connexió Peer-to-Peer.

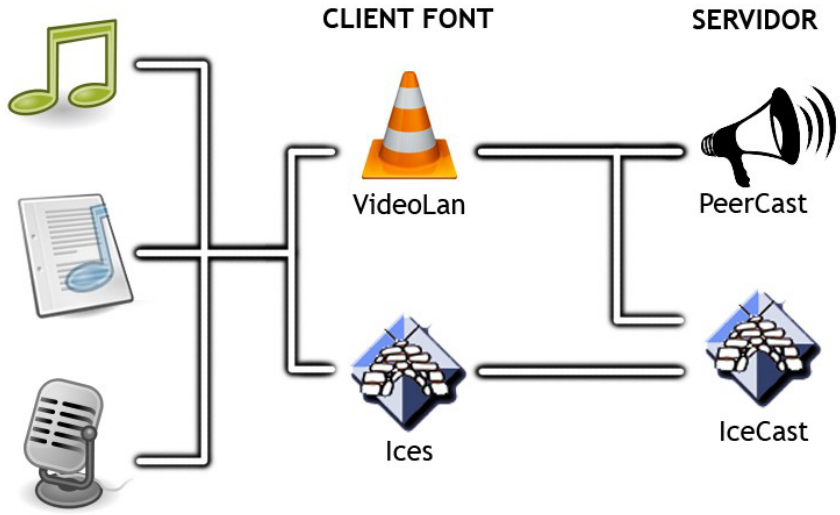

Figura 6: Esquema de corrents d'àudio

ICES + ICECAST

Aquesta combinació, es la més senzilla d'emprar degut a que els dos programes que s'utilitzen són dels mateixos creadors, i estan dissenyats expressament per funcionar conjuntament.

Únicament cal modificar els fitxers de configuració que es van instal·lar amb Ices: `ices-alsa.xml` i `ices-oss.xml` quan es vulgui servir l'àudio adquirit per la tarja de so (un o altre segons la configuració existent al SO Linux); o el fitxer de configuració `ices-playlist.xml`, que el farem servir en el cas de voler emetre fitxers d'àudio, ja sigui des d'una llista de reproducció o directament des d'un fitxer.

- Creeu un fitxer buit amb extensió .txt a la mateixa carpeta on esta `ices-playlist.xml`

```
cd /home/.../ices/ices-playlist.xml
touch playlist.txt
```

- Editeu aquest fitxer `playlist.txt`

```
kate playlist.txt
```

- Incloeu els fitxers a enviar, amb la seva ruta. Si nomes voleu enviar un fitxer, incloeu aquest fitxer únicament.

```
/home/..../canço1.ogg
/home/.../canço2.ogg
```

- Deseu el fitxer `playlist.txt`.

- Arranqueu el servidor IceCast:

```
icecast2 -c /home/.../icecast.xml
```

- Engegueu l'Ices amb el fitxer de configuració adequat, una d'aquestes tres ordres:

```
ices2 /home/.../ices-playlist.xml
ices2 /home/.../ices-oss.xml
ices2 /home/.../ices-alsa.xml
```

- Ja teniu disponible el corrent a la pàgina web del servidor l'Icecast: http://ip_servidor:8000

VLC + ICECAST

Encara que amb la combinació anterior ja s'ha aconseguit transmetre àudio, un dels nostres objectius era poder utilitzar VLC com a font, al igual que com a receptor del corrent, degut a les possibilitats que ens dona.

- Primer haureu d'engegar el servidor:

```
icecast2 -c /home/.../icecast.xml
```

- Utilitzeu l'executable del VLC que s'ha creat anteriorment amb el mòdul shout activat, situat a: `opt/vlc-0.8.6b/`

```
cd /opt/vlc-0.8.6b/bin
```

El format de l'ordre a utilitzar tindrà aquesta estructura com a base:

```
./vlc -vvv TIPUS_FONT --sout '#standard{access=shout,mux=ogg,
url=source:SOURCE-PASSWORD@IP_SERVIDOR:PORT/PUNT_MUNTATGE.ogg}'
```

- El `TIPUS_FONT` varia depenent del que es vulgui enviar, ja sigui un fitxer d'àudio, una llista de reproducció, o des de l'entrada de la tarja de so, com pot ser el micròfon.
- Amb el paràmetre `--sout`, li estem indicant que ha d'utilitzar el mòdul shout que hem instal·lat. L'accés també serà del tipus shout.
- L'àudio que s'envii haurà d'estar encapsulat (`mux`) en un contenidor `ogg`
- La part de la `url`, es la més important per poder connectar correctament amb el servidor IceCast.
    - `source`, es el nom per defecte per a totes les connexions.
    - `SOURCE-PASSWORD`, és la contrasenya font que hem establert al fitxer de configuració icecast.xml.
    - `IP_SERVIDOR`, és l'adreça IP a la que hem de dirigir el corrent que enviem.
    - `PORT`, es el port al que tenim el servidor IceCast, si no s'ha fet cap canvi al fitxer de configuració, el port és el 8000.
    - PUNT_MUNTATGE.ogg, aquest nom l'estableix l'usuari que es disposa a enviar el corrent, per tal de diferenciar les diferents fonts que arriben al servidor IceCast, tal com mostra la Figura 2. Per tant, es pot posar qualsevol nom, però sempre ha d'acabar amb `ogg`.

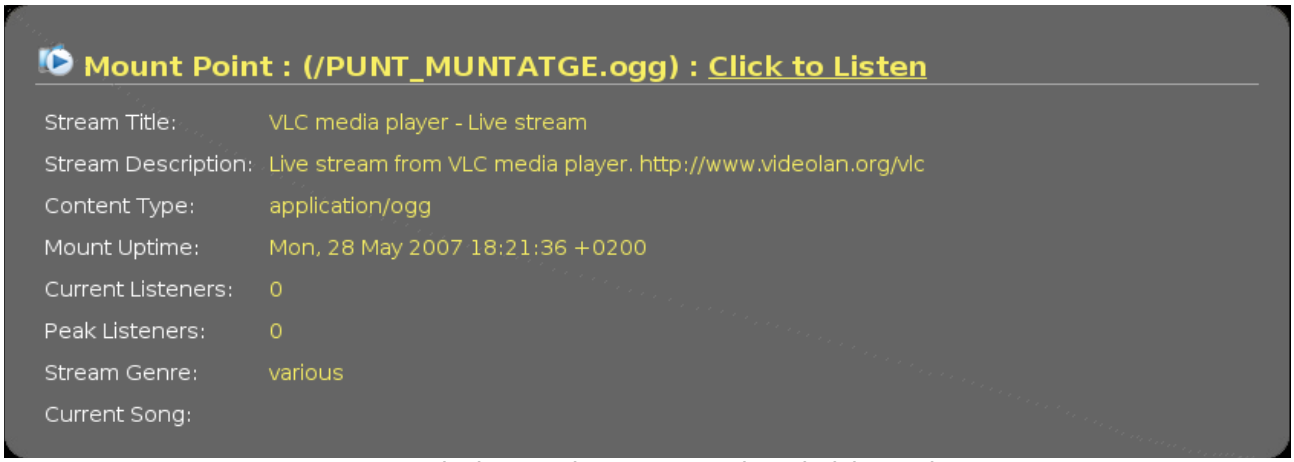

Figura 7: Exemple de punt de muntatge a la web del servidor Icecast

Una vegada explicada l'estructura veurem com tractar-la per cadascun dels casos:

Fitxer En Format OGG Vorbis

En aquest cas només haurem d'utilitzar l'estructura bàsica, degut a que el fitxer ja es troba en el format OGG, no cal codificar-lo, només enviar-lo.

```
./vlc -vvv /home/.../cançò.ogg --sout '#standard{access=shout,mux=ogg,
url=source:****@XXX.XXX.XXX.XXX:8000/vlc_music_ogg.ogg}'
```

Fitxer En Format MP3

En aquest cas es pretén enviar un corrent a partir d'un fitxer en format MP3, sense haver-lo convertit a OGG prèviament. Per tant és el VLC l'encarregat de codificar-lo, utilitzant el còdec vorbis (`acodec=vorbis`) i establint la taxa de bits (`ab=128`). La resta de l'estructura romandrà igual.

```
./vlc -vvv /home/.../cançò.mp3 --sout '#transcode{acodec=vorbis,ab=128}:
standard{access=shout,mux=ogg,url=source:****@XXX.XXX.XXX.XXX:8000/vlc_mu
sic_mp3.ogg}'
```

Llista De Reproducció

En aquest cas, l'estructura de l'ordre depèn del que hi hagi a la llista .m3u. Si son fitxers d'àudio en OGG s'utilitza l'estructura bàsica, però si els fitxers son MP3 caldrà codificar-los, anàlogament al que es feia amb un únic fitxer.

Per crear aquesta llista de reproducció, obriu la interfície gràfica del VLC, ja sigui el que s'instal·la per paquets o el que heu instal·lat seguint els passos del Capítol anterior.

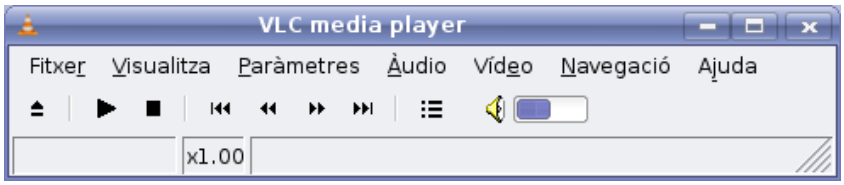

Figura 8: Interfície gràfica del VLC

- Aneu a Visualitza>Llista de reproducció (`Ctrl+L`).

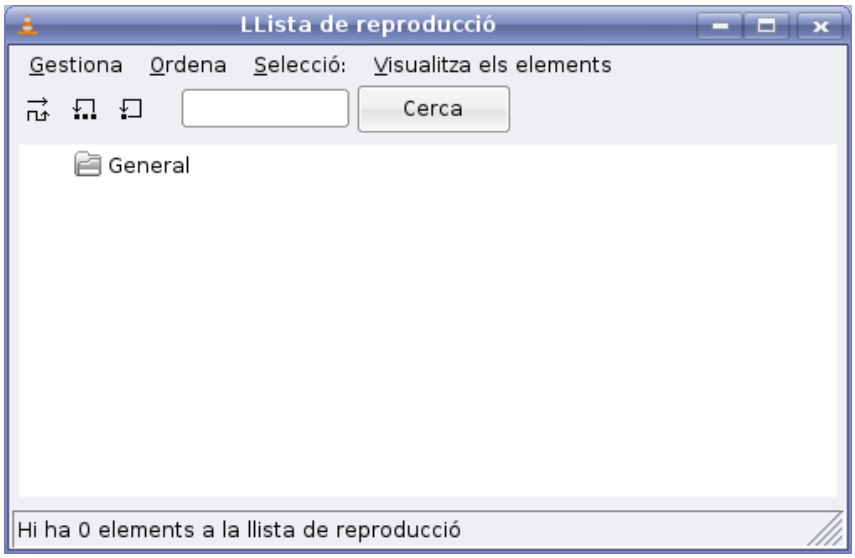

Figura 9: Llista de reproducció del VLC

- Inseriu els fitxers que formaran part de la llista.
- Per desar la llista, aneu a: Gestiona>Desa la llista de reproducció

Un cop creada i desada la llista, tanqueu aquesta interfície gràfica del VLC i continueu executant les ordres des del terminal.

- En el cas que la llista contengui només fitxers OGG, s'utilitza l'estructura bàsica, però no s'ha d'incloure l'extensió al fitxer de llista.

```
./vlc -vvv /home/.../llista_ogg --sout '#standard{access=shout,mux=ogg,
url=source:****@XXX.XXX.XXX.XXX:8000/vlc_llista_ogg.ogg}'
```

- En el cas d'utilitzar fitxers MP3, cal codificar-los a OGG abans d'enviar-los.

```
./vlc -vvv /home/.../llista_mp3 --sout '#transcode{acodec=vorbis,ab=128}:
standard{access=shout,mux=ogg,url=source:****@XXX.XXX.XXX.XXX:8000/vlc_ll
ista_mp3.ogg}'
```

Un cop ja s'està enviant el corrent, podem fer que aquesta llista no s'aturi mai, i es reprodueixi en forma de bucle, prement aquest botó: 

Entrada De La Tarja De So

Si voleu que el corrent sigui l'entrada de la tarja de so, per exemple un micròfon o la sortida d'un receptor de ràdio FM, s'utilitza el mòdul video 4Linux (`v4l`).

```
./vlc -vvv v4l:/dev/dsp --sout '#transcode{acodec=vorbis,ab=128}:
standard{access=shout,mux=ogg,url=source:****@XXX.XXX.XXX.XXX:8000/vlc_mi
cro.ogg}'
```

VLC + PEERCAST

La combinació de VLC i PeerCast és un cas diferent a les configuracions presentades fins ara, ja que per utilitzar-la no necessitem cap mena d'instal·lació extra al VLC, es pot fer servir directament la instal·lació per defecte dels paquets.

- Obriu la interfície gràfica del VLC.

- Aneu a Obrir fitxer:

Figura 10: Obrir fitxer al VLC

- Navegueu per trobar el fitxer a enviar.
- Un cop triat, aneu a opcions avançades i trieu Stream/Save,>Paràmetres
- Escolliu l'opció HTTP, on l'adreça serà la local: 127.0.0.1 i el port: 8080.

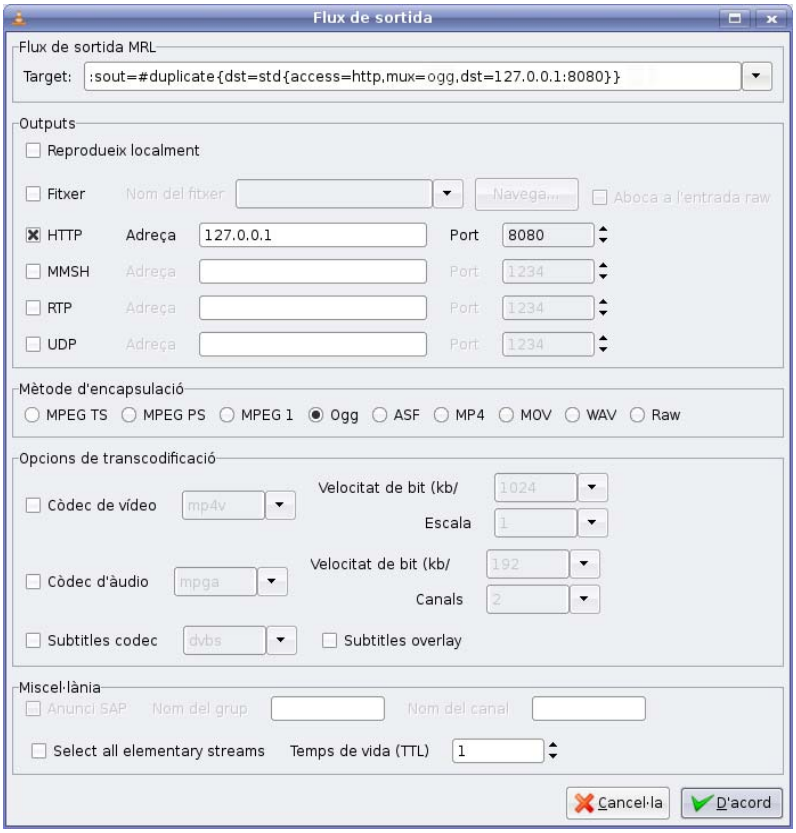

Figura 11: Paràmetres a configurar del corrent de sortida

Ara ja el VLC ja esta enviant el corrent, haureu de configurar PeerCast des de la interfície web per que VLC i PeerCast s'entenguin.

- Aneu a la secció Broadcast

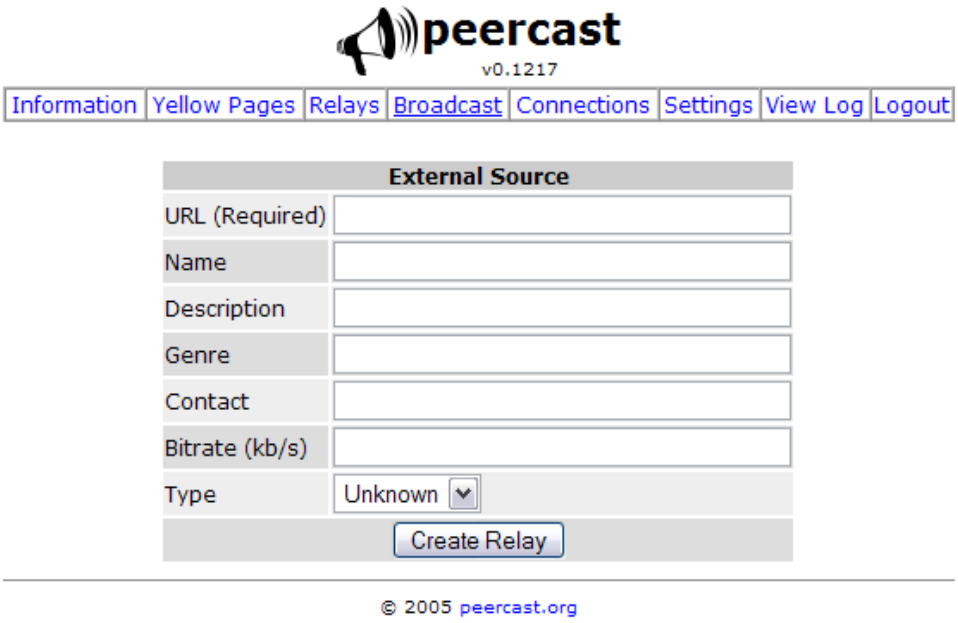

Figura 12: Aspecte de la secció Broadcast

- Introduïu la URL especificada al VLC, http://127.0.0.1:8080
- Trieu l'OGG com a tipus de corrent.
- Poseu un nom al corrent i, si voleu, una descripció.
- Un cop hageu acceptat, ja estarà disponible el corrent a la secció de Relay.

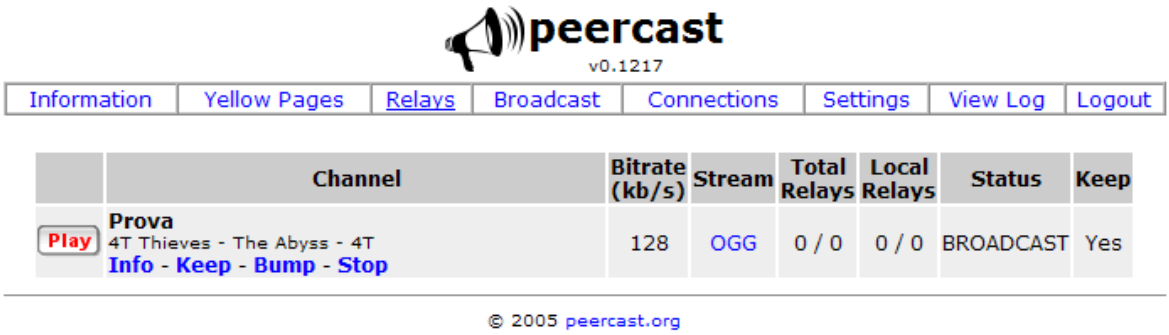

Figura 13: Aspecte de la secció Relay, amb un canal publicat

Enviament de Vídeo

Seguint el mateix esquema que amb l'àudio, ens hem decantat pel VLC com a font i reproductor i utilitzarem com a servidor únicament el IceCast, ja que les proves realitzades amb el PeerCast no han respost tant bé com amb l'àudio.

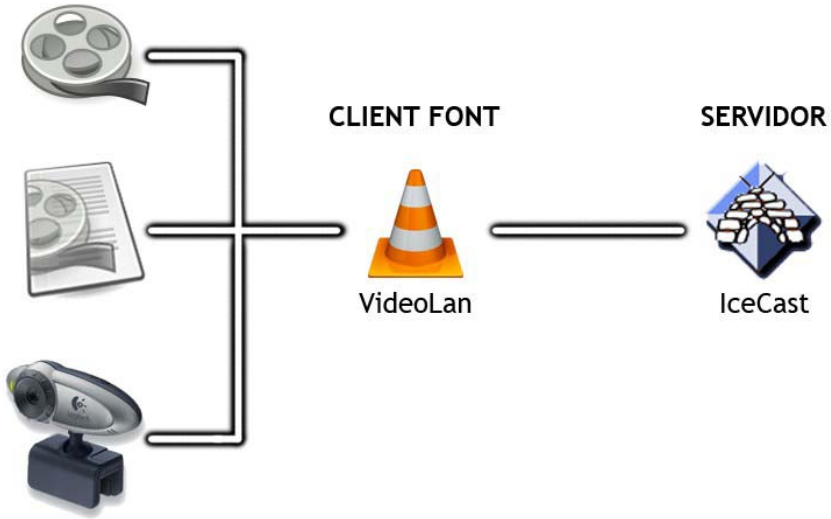

Figura 14: Esquema de corrents de vídeo

VLC + ICECAST

Per repetir aquesta combinació però amb vídeo, només cal replicar la mateixa estructura bàsica que amb àudio.

Per codificar fitxers en MPG o AVI a OGG s'ha utilitzat el programari ffmpeg2theora. En canvi, per als fitxer MP4 no s'ha pogut fer una codificació directa i abans s'han codificat a MPG amb un script, i d'aquesta manera tenir-ho automatitzat. (*especificat a l'Annex II*)

Fitxer En Format OGG Theora

Els fitxers ja els tenim codificats amb OGG, per tant no cal cap modificació a l'estructura bàsica.

```
./vlc -vvv /home/.../video.ogg --sout '#standard{access=shout,mux=ogg,
url=source:****@XXX.XXX.XXX.XXX:8000/vlc_video_ogg.ogg}'
```

Llista De Reproducció De Fitxers OGG

La llista la creareu de la mateixa manera que en el cas d'àudio, però s'afegiran els fitxers de vídeo en OGG. Tampoc posarem l'extensió de la llista.

```
./vlc -vvv /home/.../llista_ogg --sout '#standard{access=shout,mux=ogg,
url=source:****@XXX.XXX.XXX.XXX:8000/vlc_video_llista_ogg.ogg}'
```

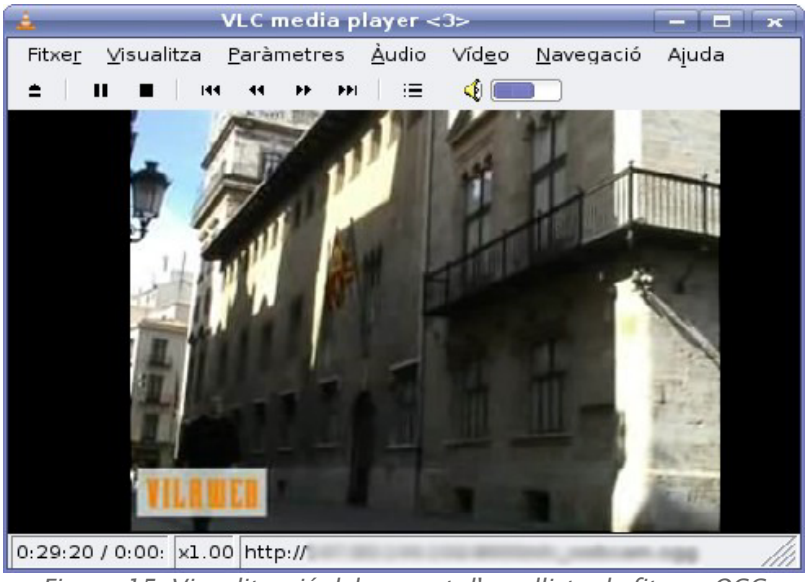

Figura 15: Visualització del corrent d'una llista de fitxers OGG

WebCam

En aquest cas, el format de vídeo que ens dona la càmera prové del video4linux instal·lat al sistema, per tant abans d'enviar-lo cal codificar el senyal a theora (vcodec=theo) i establir la taxa de bits (vb=1024). Aquesta taxa de bits no es fixa i es pot canviar depenent de les necessitats.

```
./vlc -vvv v4l:/dev/video0 --sout '#transcode{vcodec=theo,vb=1024}:
standard{access=shout,mux=ogg,url=source:****@XXX.XXX.XXX.XXX:8000/vlc_we
bcam.ogg}}'
```

- Si volem incloure també l'àudio del micròfon, cal codificar ambdues fonts.

```
./vlc -vvv v4l://:v4l-vdev="/dev/video0" :v4l-adev="/dev/dsp" --sout
'#transcode{vcodec=theo,vb=1024,acodec=vorbis,ab=128}:standard{access=sho
ut,mux=ogg, url=source:****@XXX.XXX.XXX.XXX:8000/vlc_webcam_micro.ogg}}'
```

Figura 16: Visualització del corrent de la Webcam

Mentre es feien les proves d'aquest projecte, s'ha observat que la Webcam no sempre connecta satisfactòriament, a vegades s'ha de fer més d'un intent de connexió, executant el mateix codi fins que la Webcam comença a enviar el corrent. Aquesta errada, no depèn de la taxa de bits de la codificació, el més segur sigui un problema amb els controladors de la càmera.

VLC + PEERCAST

En aquesta combinació per l'enviament de vídeo, s'ha intentat seguir el mateix procediment que amb l'àudio. Els resultats no han sigut satisfactoris, degut a que el Peercast no era capaç d'interpretat el que el VLC enviava.

OGGCAP VIDEO BROADCASTER + PEERCAST

Aquesta combinació només es possible amb SO Windows. El OggCap Video Broadcaster s'instal·la per defecte fer la instal·lació del PeerCast, per tant la comunicació entre els dos es gairebé immediata.

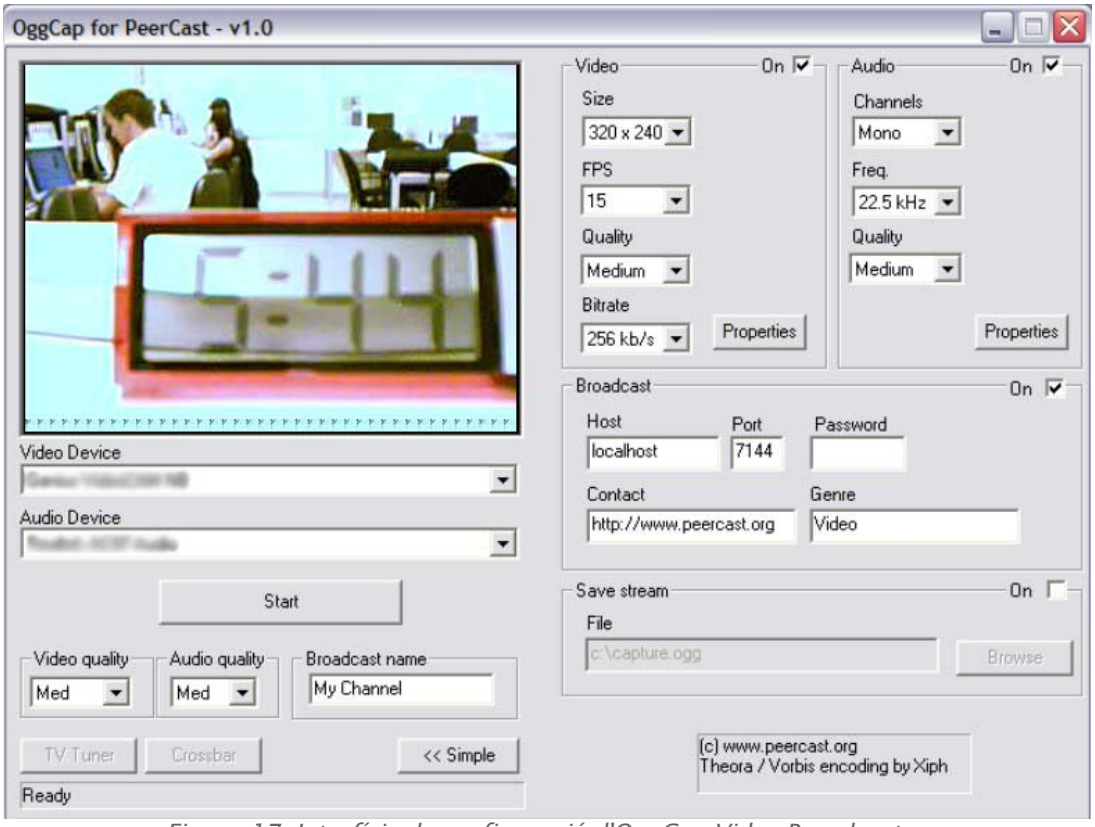

Figura 17: Interfície de configuració d'OggCap Video Broadcaster

Aquest programa ens permet connectar directament una Webcam o un sintonitzador de TV, només cal seleccionar el dispositiu de vídeo i d'àudio desitjat.

En la part de configuració simple del corrent, podem canviar la qualitat del vídeo i de l'àudio, i a *Broadcast name* s'especifica el nom del corrent. A les configuracions avançades, podrem controlar alguns paràmetres més, com la mida del vídeo, la qualitat, els fotogrames per segon, la taxa de bits, els canals d'àudio i la seva freqüència. Fins i tot, es pot emmagatzemar el corrent que s'envia, en disc.

Un cop tots els paràmetres estan configurats, enviem el corrent cap al PeerCast, prement el boto Start, i ja estarà llest per que pugui ser escoltat.

Reproducció dels Corrents

Per a la reproducció dels corrents, utilitzarem el VLC, com a client reproductor.

Per a les fonts que ens proporciona el servidor IceCast tenim dues formes d'accedir al corrent: accedint a la pàgina web del servidor [http://XXX.XXX.XXX.XXX:8000] i així visualitzem totes les fonts (punts muntatge) existents, o si sabem el nom del punt de muntatge, podem accedir directament [http://XXX.XXX.XXX.XXX:8000/punt_muntatge.ogg.m3u]

Integració dels Corrents en Web

Mitjançant el connector de VideoLan[10] per a Firefox, ens serà molt fàcil poder visualitzar el corrent de vídeo integrat en un document HTML, amb la utilització del següent codi JavaScript:

```
<embed type="application/x-vlc-plugin"
       name="corrent"
       pluginspage="http://www.videolan.org"
       id="vlc"
       target="http://XXX.XXX.XXX.XXX:8000/punt_muntatge.ogg.m3u">
  <br>
  <a href="javascript:;" onclick='document.corrent.play()'>Play video</a>
  <a href="javascript:;" onclick='document.corrent.stop()'>Stop video</a>
</embed>
```

10. En el cas de no tenir el connector instal·lat, revisar el *Capítol 4. Instal·lació i Configuració*, en l'apartat de VideoLan

Les ordres de JavaScript poden variar depenent de les necessitats de l'usuari. A continuació s'especifiquen algunes:

```
play() : Inicia el corrent d'entrada
pause() : Pausa el corrent d'entrada
stop() : Atura el corrent d'entrada
fullscreen() : Canvia el vídeo a pantalla completa
set_volume(vol) : Canvia el volum (rang 0 a 200)
get_volume() : Mostra el valor del volum
mute() : Silencia
next() : Següent
previous() : Previ
isplaying() : retorna TRUE, si el connector esta reproduint
get_length() : Llargada del corrent en segons
get_position() : Posició en el corrent en percentatge
get_time() :  Posició en el corrent en segons
```

Al departament de Teoria del Senyal i Comunicacions de l'EUETIT, s'ha creat un portal web, dedicat als continguts multimèdia que es generen al departament. Entre els continguts disponibles, es troben els corrents generats amb aquest projecte, en aquest cas la Webcam del Laboatori i la llista de vídeos de VilawebTV. A continuació veïem l'aspecte de les webs:

Figura 18: Webcam del laboratori al TSC

Figura 19: Canal de Vilaweb al TSC

Capítol 6.
FUTURES LÍNIES DE TREBALL

Tenint en compte que la durada d'aquest projecte, ha sigut de quatre mesos, han quedat temes que es podrien resoldre si hagués tingut un període més llarg per treballar. També s'ha de tenir en compte que s'ha estat treballant amb unes tecnologies que evolucionen cada dia, i que actualment no son estables.

Com a futurs actes de treball es proposa:

- Treballar en l'àmbit del Peer to Peer, per alliberar amplada de banda, tenint en compte la càrrega que els vídeos suposen per a gestió de connexions d'Internet. Provar el programari encara en desenvolupament, IceShare, possiblement es molt similar al IceCast, i ens permetrà generar una xarxa peer to peer que treballi amb el format contenidor OGG.

- La possibilitat de poder emetre un corrent des de les targes DVB, que actualment no estan correctament instal·lades al Laboratori. Possiblement un cop estiguin ben instal·lades i no entren en conflicte amb la Webcam, serà molt senzill enviar aquest corrent cap al servidor IceCast, per a difusió unicast/multicast.

- Una millora del connector de VideoLan per a Firefox, seria mot satisfactori degut a que alguns cops, quan es connecta al corrent, el Firefox ha finalitzat de manera sobtada, degut a un mal comportament del connector.

Capítol 7.
CONCLUSIONS

Una vegada finalitzat el projecte, s'han assolit gairebé tots els objectius i requeriments inicials que es plantejaven.

- Tot el sistema proposat funciona sota programari lliure i formats lliures, i s'ha treballat, en les instal·lacions i configuracions amb sistemes operatius lliures, GNU/Linux, concretament amb Ubuntu Edgy i Kubuntu Feisty.

- La solució proposada es manté estable, després de tenir el sistema un temps prolongat funcionant.

- S'ha aconseguit transmetre corrents en temps real d'àudio i vídeo des de tota mena de fonts possibles, arxius, llistes de reproducció, entrada de la tarja de so i webcam. Utilitzant el mètode de difusió unicast/multicast, que no es actualment el mes òptim en l'àmbit de les comunicacions per Internet, però si que es acceptable a un nombre baix de connexions.

- La solució proposada per al mètode de difusió peer to peer, ha funcionat només per a corrents d'àudio en temps real, però no s'han obtingut resultats en l'àmbit de les corrents de vídeo.

- La integració dels corrents en un portal web, es molt senzilla, al fer servir el connector del VideoLan per a Firefox, que utilitzar llenguatge JavaScript, aixo ens permet accedir des de qualsevol Sistema Operatiu en el que es pugui instal·lar el navegador web Firefox.

- Per a la reproducció dels corrents es proposa el VideoLan, que es programari lliure, la seva instal·lació es possible en la majoria de Sistemes Operatius vigents i llengües.

Amb la realització d'aquest projecte s'han assolit nous coneixements informàtics, en l'àmbit del sistemes operatius, i el tractament de GNU/Linux, i el mon del programari lliure i les seves possibilitats. També s'han obtinguts coneixements sobre els mètodes de difusió de continguts multimèdia actuals, de gestió de xarxes, i formats d'arxius i les seves conversions.

ANNEX I
FITXERS DE CONFIGURACIÓ

ICECAST.XML

```
<icecast>
    <limits>
        <clients>100</clients>
        <sources>2</sources>
        <threadpool>5</threadpool>
        <queue-size>524288</queue-size>
        <client-timeout>30</client-timeout>
        <header-timeout>15</header-timeout>
        <source-timeout>10</source-timeout>
        <!-- If enabled, this will provide a burst of data when a client
             first connects, thereby significantly reducing the startup
             time for listeners that do substantial buffering. However,
             it also significantly increases latency between the source
             client and listening client.  For low-latency setups, you
             might want to disable this. -->
        <burst-on-connect>1</burst-on-connect>
        <!-- same as burst-on-connect, but this allows for being more
             specific on how much to burst. Most people won't need to
             change from the default 64k. Applies to all mountpoints  -->
        <burst-size>65535</burst-size>
    </limits>

    <authentication>
        <!-- Sources log in with username 'source' -->
        <source-password>hackme</source-password>
        <!-- Relays log in username 'relay' -->
        <relay-password>hackme</relay-password>

        <!-- Admin logs in with the username given below -->
        <admin-user>admin</admin-user>
        <admin-password>hackme</admin-password>
    </authentication>

    <!-- Uncomment this if you want directory listings -->
    <!--
    <directory>
        <yp-url-timeout>15</yp-url-timeout>
        <yp-url>http://dir.xiph.org/cgi-bin/yp-cgi</yp-url>
    </directory>
     -->

     <!-- This is the hostname other people will use to connect to your
server.
    It affects mainly the urls generated by Icecast for playlists and yp
    listings. -->
    <hostname>localhost</hostname>

    <!-- You can use these two if you only want a single listener -->
    <!--<port>8000</port> -->
    <!--<bind-address>127.0.0.1</bind-address>-->

    <!-- You may have multiple <listener> elements -->
```

```
<listen-socket>
    <port>8000</port>
    <!-- <bind-address>127.0.0.1</bind-address> -->
</listen-socket>
<!--
<listen-socket>
    <port>8001</port>
</listen-socket>
-->

<!--<master-server>127.0.0.1</master-server>-->
<!--<master-server-port>8001</master-server-port>-->
<!--<master-update-interval>120</master-update-interval>-->
<!--<master-password>hackme</master-password>-->

<!-- setting this makes all relays on-demand unless overridden, this
is useful for master relays which do not have <relay> definitions
here.The default is 0 -->
<!--<relays-on-demand>1</relays-on-demand>-->

<!--
<relay>
    <server>127.0.0.1</server>
    <port>8001</port>
    <mount>/example.ogg</mount>
    <local-mount>/different.ogg</local-mount>
    <on-demand>0</on-demand>

    <relay-shoutcast-metadata>0</relay-shoutcast-metadata>
</relay>
-->

<!-- Only define a <mount> section if you want to use advanced
options,like alternative usernames or passwords
<mount>
    <mount-name>/example-complex.ogg</mount-name>

    <username>othersource</username>
    <password>hackmemore</password>

    <max-listeners>1</max-listeners>
    <dump-file>/tmp/dump-example1.ogg</dump-file>
    <burst-size>65536</burst-size>
    <fallback-mount>/example2.ogg</fallback-mount>
    <fallback-override>1</fallback-override>
    <fallback-when-full>1</fallback-when-full>
    <intro>/example_intro.ogg</intro>
    <hidden>1</hidden>
    <no-yp>1</no-yp>
    <authentication type="htpasswd">
            <option name="filename" value="myauth"/>
            <option name="allow_duplicate_users" value="0"/>
    </authentication>
    <on-connect>/home/icecast/bin/stream-start</on-connect>
    <on-disconnect>/home/icecast/bin/stream-stop</on-disconnect>
</mount>

<mount>
```

```
    <mount-name>/auth_example.ogg</mount-name>
    <authentication type="url">
        <option name="mount_add"
              value="http://myauthserver.net/notify_mount.php"/>
        <option name="mount_remove"
              value="http://myauthserver.net/notify_mount.php"/>
        <option name="listener_add"
              value="http://myauthserver.net/notify_listener.php"/>
        <option name="listener_remove"
              value="http://myauthserver.net/notify_listener.php"/>
    </authentication>
</mount>
-->

<fileserve>1</fileserve>

<!-- set the mountpoint for a shoutcast source to use, the default if
not specified is /stream but you can change it here if an alternative
is         wanted or an extension is required
<shoutcast-mount>/live.nsv</shoutcast-mount>
-->
<paths>
  <!-- basedir is only used if chroot is enabled -->
    <basedir>/usr/share/icecast2</basedir>

     <!-- Note that if <chroot> is turned on below, these paths must
     both be relative to the new root, not the original root -->
    <logdir>/var/log/icecast2</logdir>
    <webroot>/usr/share/icecast2/web</webroot>
    <adminroot>/usr/share/icecast2/admin</adminroot>
    <!-- <pidfile>/usr/share/icecast2/icecast.pid</pidfile> -->

      <!-- Aliases: treat requests for 'source' path as being for
      'dest' path May be made specific to a port or bound address
      using the "port" and "bind-address" attributes.
      -->
    <!--
    <alias source="/foo" dest="/bar"/>
      -->
      <!-- Aliases: can also be used for simple redirections as
      well,this   example   will   redirect   all   requests   for
      http://server:port/ to the status page
      -->
    <alias source="/" dest="/status.xsl"/>
</paths>

<logging>
    <accesslog>access.log</accesslog>
    <errorlog>error.log</errorlog>
    <!-- <playlistlog>playlist.log</playlistlog> -->
        <loglevel>4</loglevel> <!-- 4 Debug, 3 Info, 2 Warn, 1 Error
    -->
        <logsize>10000</logsize> <!-- Max size of a logfile -->
    <!-- If logarchive is enabled (1), then when logsize is reached
    the     logfile     will     be     moved     to     [error|access|
    playlist].log.DATESTAMP, otherwise it will be moved to [error|
    access|playlist].log.old.  Default  is  non-archive  mode  (i.e.
    overwrite)
```

```
        -->
        <!-- <logarchive>1</logarchive> -->
    </logging>

    <security>
        <chroot>0</chroot>
        <!--
        <changeowner>
            <user>nobody</user>
            <group>nogroup</group>
        </changeowner>
        -->
    </security>
</icecast>
```

IceCast.xml Versió Mínima

Aquesta versió del fitxer, conté els paràmetres mínims de configuració, que son els que s'han de canviar.

```
<icecast>
    <limits>
        <sources>100</sources>
    </limits>
    <authentication>
        <source-password>hackme</source-password>
        <relay-password>hackme</relay-password>
        <admin-user>admin</admin-user>
        <admin-password>hackme</admin-password>
    </authentication>

    <!--<directory>
        <yp-url-timeout>50</yp-url-timeout>
        <yp-url>http://dir.xiph.org/cgi-bin/yp-cgi</yp-url>
    </directory> -->

    <hostname>localhost</hostname>
    <listen-socket>
        <port>8000</port>
    </listen-socket>
    <fileserve>1</fileserve>
    <paths>
        <logdir>/var/log/icecast2</logdir>
        <webroot>/usr/share/icecast2/web</webroot>
        <adminroot>/usr/share/icecast2/admin</adminroot>
        <alias source="/" dest="/status.xsl"/>
    </paths>
    <logging>
        <accesslog>access.log</accesslog>
        <errorlog>error.log</errorlog>
            <loglevel>3</loglevel> <!-- 4 Debug, 3 Info, 2 Warn, 1 Error -->
    </logging>
</icecast>
```

ICES-ALSA.XML

```
<?xml version="1.0"?>
<ices>

    <!-- run in background  -->
    <background>0</background>
    <!-- where logs go. -->
    <logpath>/var/log/ices</logpath>
    <logfile>ices.log</logfile>
    <!-- size in kilobytes -->
    <logsize>2048</logsize>
    <!-- 1=error, 2=warn, 3=infoa ,4=debug -->
    <loglevel>4</loglevel>
    <!-- logfile is ignored if this is set to 1 -->
    <consolelog>0</consolelog>

    <!-- optional filename to write process id to -->
    <!-- <pidfile>/home/ices/ices.pid</pidfile> -->

    <stream>
        <!-- metadata used for stream listing -->
        <metadata>
            <name>Example stream name</name>
            <genre>Example genre</genre>
            <description>A short description of your stream</description>
            <url>http://mysite.org</url>
        </metadata>

        <!--    Input module.
            This example uses the 'oss' module. It takes input from the
            OSS audio device (e.g. line-in), and processes it for live
            encoding.  -->
        <input>
            <module>alsa</module>
            <param name="rate">44100</param>
            <param name="channels">2</param>
            <param name="device">hw:0,0</param>
            <!-- Read metadata (from stdin by default, or -->
            <!-- filename defined below (if the latter, only on SIGUSR1) -->
            <param name="metadata">1</param>
            <param name="metadatafilename">test</param>
        </input>

        <!--    Stream instance.
            You may have one or more instances here.  This allows you to
            send the same input data to one or more servers (or to
            differentmountpoints on the same server). Each of them can
            have different parameters. This is primarily useful for a)
            relaying to multiple independent servers, and b)
            encoding/reencoding to multiple bitrates.

            If one instance fails (for example, the associated server
            goes down, etc), the others will continue to function
            correctly. This example defines a single instance doing live
```

```
            encoding at low bitrate.  -->

        <instance>
            <!--    Server details.
                You define hostname and port for the server here, along
                with the source password and mountpoint.  -->

            <hostname>localhost</hostname>
            <port>8000</port>
            <password>hackme</password>
            <mount>/example1.ogg</mount>
                <yp>1</yp>    <!-- allow stream to be advertised on YP,
default 0 -->

            <!--    Live encoding/reencoding:
                channels and samplerate currently MUST match the
                channels and samplerate given in the parameters to the
                oss input module above or the remsaple/downmix section
                below.  -->

            <encode>
                <quality>0</quality>
                <samplerate>22050</samplerate>
                <channels>1</channels>
            </encode>

            <!-- stereo->mono downmixing, enabled by setting this to 1 -->
            <downmix>1</downmix>

            <!-- resampling.
                Set to the frequency (in Hz) you wish to resample to, -->

            <resample>
                <in-rate>44100</in-rate>
                <out-rate>22050</out-rate>
            </resample>
        </instance>

    </stream>
</ices>
```

ICES-OSS.XML

```
<?xml version="1.0"?>
<ices>

    <!-- run in background  -->
    <background>0</background>
    <!-- where logs go. -->
    <logpath>/var/log/ices</logpath>
    <logfile>ices.log</logfile>
    <!-- size in kilobytes -->
    <logsize>2048</logsize>
    <!-- 1=error, 2=warn, 3=infoa ,4=debug -->
    <loglevel>4</loglevel>
    <!-- logfile is ignored if this is set to 1 -->
    <consolelog>0</consolelog>

    <!-- optional filename to write process id to -->
    <!-- <pidfile>/home/ices/ices.pid</pidfile> -->

    <stream>
        <!-- metadata used for stream listing -->
        <metadata>
            <name>OSS</name>
            <genre>Example genre</genre>
            <description>A short description of your stream</description>
            <url>http://mysite.org</url>
        </metadata>

        <!--    Input module.
            This example uses the 'oss' module. It takes input from the
            OSS audio device (e.g. line-in), and processes it for live
            encoding.  -->
        <input>
            <module>oss</module>
            <param name="rate">44100</param>
            <param name="channels">2</param>
            <param name="device">/dev/dsp</param>
            <!-- Read metadata (from stdin by default, or -->
            <!-- filename defined below (if the latter, only on SIGUSR1) -->
            <param name="metadata">1</param>
            <param name="metadatafilename">test</param>
        </input>

        <!--    Stream instance.
            You may have one or more instances here. This allows you to
            send the same input data to one or more servers (or to
            different mountpoints on the same server). Each of them can
            have different parameters. This is primarily useful for a)
            relaying to multiple independent servers, and b)
            encoding/reencoding to multiple bitrates.

            If one instance fails (for example, the associated server
            goes down, etc), the others will continue to function
```

```
            correctly. This example defines a single instance doing live
            encoding at low bitrate.  -->

        <instance>
            <!--    Server details.
                You define hostname and port for the server here, along
                with the source password and mountpoint.  -->

            <hostname>localhost</hostname>
            <port>8000</port>
            <password>hackme</password>
            <mount>/example1.ogg</mount>
                <yp>1</yp>    <!-- allow stream to be advertised on YP,
default 0 -->

            <!--    Live encoding/reencoding:
                channels and samplerate currently MUST match the channels
                and samplerate given in the parameters to the oss input
                module above or the remsaple/downmix section below.  -->

            <encode>
                <quality>0</quality>
                <samplerate>22050</samplerate>
                <channels>1</channels>
            </encode>

            <!-- stereo->mono downmixing, enabled by setting this to 1 -->
            <downmix>1</downmix>

            <!-- resampling.
                Set to the frequency (in Hz) you wish to resample to, -->

            <resample>
                <in-rate>44100</in-rate>
                <out-rate>22050</out-rate>
            </resample>
        </instance>

    </stream>
</ices>
```

ICES-PLAYLIST.XML

```
<?xml version="1.0"?>
<ices>
    <!-- run in background -->
    <background>0</background>
    <!-- where logs, etc go. -->
    <logpath>/var/log/ices</logpath>
    <logfile>ices.log</logfile>
    <!-- 1=error,2=warn,3=info,4=debug -->
    <loglevel>4</loglevel>
    <!-- set this to 1 to log to the console instead of to the file above -->
    <consolelog>0</consolelog>

    <!-- optional filename to write process id to -->
    <!-- <pidfile>/home/ices/ices.pid</pidfile> -->

    <stream>
        <!-- metadata used for stream listing (not currently used) -->
        <metadata>
            <name>Example stream name</name>
            <genre>Example genre</genre>
            <description>A short description of your stream</description>
        </metadata>

        <!-- input module
            The module used here is the playlist module - it has
            'submodules' for different types of playlist. There are two
            currently implemented, 'basic', which is a simple file-based
            playlist, and 'script' which invokes a command to returns a
            filename to start playing. -->

        <input>
            <module>playlist</module>
            <param name="type">basic</param>
            <param name="file">playlist.txt</param>
            <!-- random play -->
            <param name="random">0</param>
            <!-- if the playlist get updated that start at the beginning -->
            <param name="restart-after-reread">0</param>
            <!-- if set to 1 , plays once through, then exits. -->
            <param name="once">0</param>
        </input>

      <!-- Stream instance
            You may have one or more instances here. This allows you to
            send the same input data to one or more servers (or to
            different mountpoints on the same server). Each of them can
            have different parameters. This is primarily useful for a)
            relaying to multiple independent servers, and b)
            encoding/reencoding to multiple bitrates. If one instance
            fails (for example, the associated server goes down, etc),
            the others will continue to function correctly. This example
            defines two instances as two mountpoints on the same server.
            -->
```

```
        <instance>
            <!-- Server details:
                 You define hostname and port for the server here, along
                 with the source password and mountpoint.  -->
            <hostname>localhost</hostname>
            <port>8000</port>
            <password>hackme</password>
            <mount>/example.ogg</mount>

            <!-- Reconnect parameters:
                 When something goes wrong (e.g. the server crashes, or
                 the network drops) and ices disconnects from the
                 server, these control how often it tries to reconnect,
                 and how many times it tries to reconnect. Delay is in
                 seconds. If you set reconnectattempts to -1, it will
                 continue  indefinately. Suggest setting reconnectdelay
                 to a large value if you do this.
            -->
            <reconnectdelay>2</reconnectdelay>
            <reconnectattempts>5</reconnectattempts>

            <!-- maxqueuelength:
                 This describes how long the internal data queues may be.
                 This basically lets you control how much data gets
                 buffered before ices decides it can't send to the server
                 fast enough, and  either shuts down or flushes the queue
                 (dropping the data) and continues.  For advanced users
                 only.
            -->
            <maxqueuelength>80</maxqueuelength>

            <!-- Live encoding/reencoding:
                Currrently, the parameters given here for encoding MUST
                match the input data for channels and sample rate. That
                restriction will be relaxed in the future.
            -->
            <encode>
                <!-- bps. e.g. 64000 for 64 kbps -->
                <nominal-bitrate>64000</nominal-bitrate>
                <samplerate>44100</samplerate>
                <channels>2</channels>
            </encode>
        </instance>

    </stream>
</ices>
```

ANNEX II
SCRIPT DE CONVERSIÓ MP4 A OGG

En aquest annex, s'explica el Script utilitzat per la conversió dels arxius de vídeo en mp4 a ogg, de forma automàtica. S'utilitzaran dos programes diferents, Mencoder transformarà els vídeos de MP4 a MPG, i el Ffmpeg2theora transformarà els MPG a arxius OGG.

- Creeu un directori general: `videos`

```
mkdir videos
```

- Poseu el Script: `rus.sh` a dintre
- Canvieu els permisos per que sigui executable

```
cd videos
chmod u+x run.sh
```

- Creeu 3 directoris, `mp4`, `mpg` i `ogg`

```
mkdir mp4
mkdir mpg
mkdir ogg
```

- Al directori `mp4`, col·loqueu els vídeos en format .mp4
- Executeu el Script

```
run.sh
```

Automàticament, començarà a codificar els arxius, i els col·locarà cadascun als seus respectius directoris. El procés pot trigar, depèn de la quantitat de vídeos a codificar, i de la llargada.

Al codificar s'ha definit un format de sortida de 352x288 píxels, per tal de no haver d'emetre uns arxius extremadament pesats.

Els noms dels arxius es mantenen al llarg de tota la codificació, els arxius tindran el mateix nom d'origen però amb extensió i codificació canviada.

RUN.SH

```
#!/bin/sh

# This script transcodes all MP4 files from the input folder
# to MPEG-1 and then to the output folder to OGG
# Written by Xavier Giro & Angela Abad
# 18th May 2007

MP4_MPG=/usr/bin/mencoder
MPG_OGG=/usr/bin/ffmpeg2theora
MP4_DIR=./mp4
MPG_DIR=./mpg
OGG_DIR=./ogg

# for each file in the MP4 directory
for MP4_FILE in $MP4_DIR/*.mp4
  do
  MPG_FILE=$MPG_DIR/`basename $MP4_FILE .mp4`.mpg

  # transcode in MPEG-1 to MPEG directory
  echo
  echo 'Transcoding: '$MP4_FILE' to '$MPG_FILE' '

  $MP4_MPG  $MP4_FILE  -ovc  lavc  -vf  scale=352:288  -oac  lavc  -o
$MPG_FILE

  OGG_FILE=$OGG_DIR/`basename $MPG_FILE .mpg`.ogg

  # transcode in OGG to output directory
  echo
  echo 'Transcoding: '$MPG_FILE' to '$OGG_FILE' '

  $MPG_OGG $MPG_FILE -x 352 -y 288 -o $OGG_FILE

done
```

Índex de Figures