# Anexo I. Puesta a punto del equipo

## 1.Instalación Ubuntu 14.04 LTS.

Se decide realizar todo el proceso de construcción de la App en el entorno del sistema operativo Ubuntu ya que pertenece a software libre y está totalmente apartado de cualquier tipo de copyright. La versión de Ubuntu que ha sido instalada en el mini-pc es la siguiente: UBUNTU 14.04 LTS


Imagen 1: captura de inicio de ejecución de S.O. Ubuntu

La versión está totalmente de forma gratuita para su descarga y libre de software malicioso en la siguiente dirección url: http://cdimage.ubuntu.com/netboot/14.04/?_ga=1.222350681.314541382.1

469442846

Una vez descargada la imagen de este software, bastará con ejecutarlo y seguir los pasos de configuración.

Tener en cuenta que si se tiene instalado otro S.O en el mismo disco donde se va a realizar esta instalación, da la opción o de formatear y ocupar todo el disco, o compartir los dos sistemas instalados. Al iniciar el dispositivo, da la opción de elegir con que sistema operativo arranca el terminal.

Para terminar, si se tiene dudas sobre la instalación, puede valerse de la información que se obtiene en el siguiente link:

http://www.ubuntu-guia.com/2012/04/como-instalar-ubuntu.html

Aquí se encuentra detallado, paso a paso la instalación con mucho detalle incluso aporta vídeo tutorial.

## 2.Descarga e instalación de Gstreamer en Ubuntu.

Para comenzar hay que dirigirse a la siguiente página https://gstreamer.freedesktop.org

Aquí habrá que situarse en el apartado de descargas y seleccionar el sistema operativo sobre el que se va a trabajar. En este caso, se accede al directorio de fuentes de descargas y posteriormente se descargarán los siguientes plugins que finalmente serán necesarios para una buen uso de Gstreamer. Se escoge la versión 1.6.4 por motivos de compatibilidad y solución de errores para su uso en Android Studio para dispositivos Android. En la actulidad existen versiones superiores donde cabe la posibilidad de que se encuentren errores sin resolver en sus funciones.

Por cuestiones de organización, se crea una carpeta específica llamada gstreamer para almacenar estos archivos para su posterior localización e instalación. Los archivos a descargar son los siguientes y siempre referenciados a la versión 1.6.4 con extensión de archivo .tar.gz:

- gst-libav/
- gst-plugins-bad/
- gst-plugins-base/
- gst-plugins-good/
- gst-plugins-ugly/
- gstreamer/

Al poseer este tipo de extensión de archivo, una vez realizada la descarga se procede a la descompresión de los archivos para su instalación.

El proceso de instalación para cada archivo contiene la misma secuencia de órdenes con el fin de una correcta instalación. Para ello, se abre un terminal nuevo y mediante la orden cd, se posiciona en la carpeta que almacena los plugins de Gstreamer previamente descargados.

A continuación, se posiciona otra vez con la ayuda del comando cd, en el primer plugin y se realizan las siguientes operaciones:

```
.configure –prefix=/usr

make

sudo make install
```


Al realizar cualquiera de estos comandos puede darse el caso de que el pc pregunte si desea instalar complementos asociados al archivo, habrá que indicarle en todos los casos que sí.

Una vez instalado el primer archivo, habrá que posicionarse otra vez en la carpeta de Gstreamer y volver a realizar los pasos anteriores para los archivos restantes de instalar.

Para comprobar la instalación, se abre una nueva terminal introduciendo los siguientes export:

```
export LD_LIBRARY_PATH=/usr/lib

export PKG_CONFIG_PATH=/usr/lib/pkgconfig

export GST_PLUGIN_PATH=/usr/llib/gstreamer-1.0

export PATH=$PATH:/usr/lib:/usr
```

y a continuación introducir el siguiente comando:

```
gst-launch-1.0  videotestsrc ! ximagesink
```

Si todo ha sido instalado correctamente, se mostrará la siguiente ventana, de lo contrario, habrá que volver a realizar la instalación de los archivos.

# 3.Pruebas realizadas con Gstreamer.

## 3.1. Reproducción mediante el uso de Gstreamer.

Con toda la información asimilada anteriormente citada, se puede proceder a la construcción de las pipelines que permitirán enviar el flujo a las secondary screens.

Para ello, se realizan varias pruebas mediante scripts lanzados por terminal de consola para posteriormente implementarlo en lenguaje c en la plataforma de eclipse. Cada elemento irá separado entre exclamaciones y espacios.

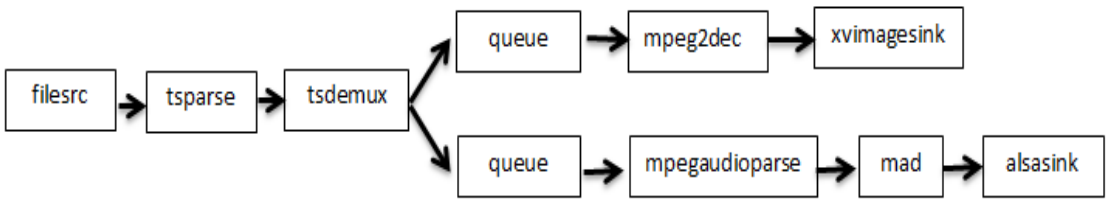
Cabe mencionar que la instalación de este framework incluye una herramienta que se puede acceder a ella a través de comandos en un terminal, que informa sobre las propiedades de cualquier elemento que se haga en una consulta con dichos comandos. Esta herramienta es *gst inspect*. Imprime por consola la descripción de cualquier pluggin previamente instalado que se consulta o de cualquier elemento perteneciente a una pipeline.

### 3.1.1. Reproducción archivo con extensión .ts almacenado en disco.

#### *3.1.1.1. Codificación SD*

El script que permite reproducir un programa en calidad SD, es decir todos los canales excepto los HD, es el siguiente:

```
gst-launch-1.0 filesrc location=/home/carlos/Vídeos/atresmedia.ts ! tsparse !
tsdemux program-number=152 emit-stats=TRUE name=demux demux.  ! queue !
mpeg2dec ! xvimagesink demux. ! queue ! mpegaudioparse ! mad  ! alsasink
```




Se va a identificar cada elemento:

- gst-launch-1.0: comando que permite generar pipelines desde la terminal.

- filesrc location=/home/carlos/Vídeos/atresmedia.ts: elemento que indica de qué tipo de fuente se va a obtener el flujo multimedia y el subelemento location, indica donde se encuentra el archivo TS a usar.

- tsparse: analiza el flujo de transporte MPEG-2, permite comprobar que no hay errores.

- tsdemux: decir que las subclases se ha puesto por investigación y para que pudiera funcionar. Se debe poner el program number correspondiente al flujo a visualizar, y además, se escoge que este elemento vaya compartiendo variables como el PTS con la aplicación desarrollada. Además se le da nombre a este elemento para poder tener una bifurcación en la pipeline, una rama para vídeo y la otra para audio.

- queue: cola simple de datos.

- mpeg2dec: Decodificador MPEG2 que pertenece a la librería LibMpeg2.
- xvimagesink: se encarga de reproducir el vídeo, es el elemento que representa la pantalla.

- mpegaudioparse: se encarga de filtrar los archivos del TS relacionados con el audio MPEG 1 Audio (niveles 1-3).

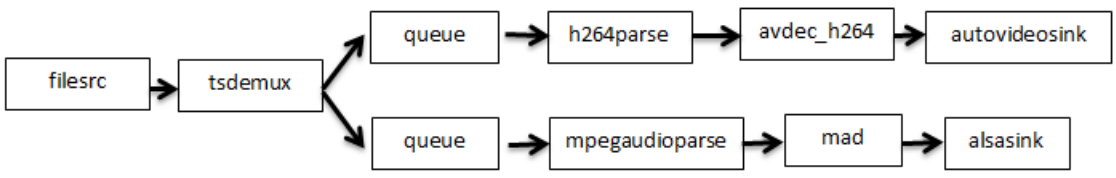
- mad: Utiliza el código mad para decodificar trenes de mp3.

- alsasink: Salida a través de una tarjeta de sonido ALSA (Arquitectura de Sonido Avanzada para Linux).

#### *3.1.1.2. Codificación HD*

Para seguir, el script encargado de reproducir los programas en HD es el siguiente:

```
gst-launch-1.0 filesrc location=/home/carlos/Vídeos/atresmedia.ts ! tsdemux
name=demux program-number=151 demux. ! queue max-size-buffers=0 max-size-
time=0 ! h264parse ! avdec_h264 ! autovideosink demux. ! queue !
mpegaudioparse ! mad ! alsasink
```




Como se puede observar, el único cambio que se aprecia está en la ramificación correspondiente a la decodificación del vídeo, donde se emplean los dos siguientes elementos:

- h264parse: Analiza flujos H.264.

- avdec_h264: decodifica los flujos H.264.

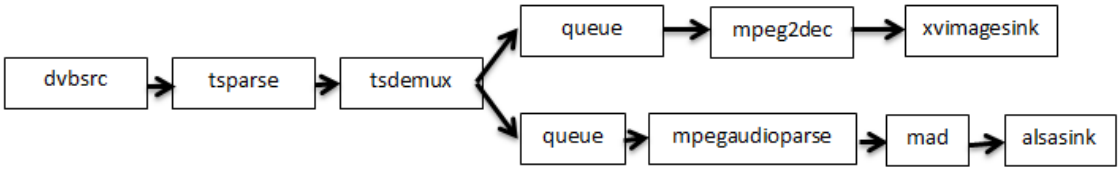
### 3.1.2. Reproducción de señal de TV (DVB-T) en directo.

#### *3.1.2.1. Reproducción de un programa en formato SD de la señal de TV (DVB-T).*

Respecto a las pipelines descritas anteriormente, el único cambio que tendrán será el de la elección del archivo fuente a reproducir. Se sustituye el elemento filesrc por dvbsrc definiendo las características empleadas por el TDT en España para la codificación de cualquier multiplex.

La pipeline resultante para reproducir canales en formato SD tiene la siguiente forma:

```
gst-launch-1.0 dvbsrc adapter=0 inversion=AUTO modulation=AUTO trans-mode=AUTO bandwidth=8 frequency=626000000 code-rate-lp=AUTO code-rate-hp=AUTO guard=AUTO hierarchy=AUTO ! tsparse ! tsdemux program-number=152 emit-stats=TRUE name=demux demux. ! queue ! mpeg2dec ! xvimagesink demux.! queue ! mpegaudioparse ! mad ! alsasink
```




Como se puede observar, el elemento dvbsrc contiene los siguientes subelementos:

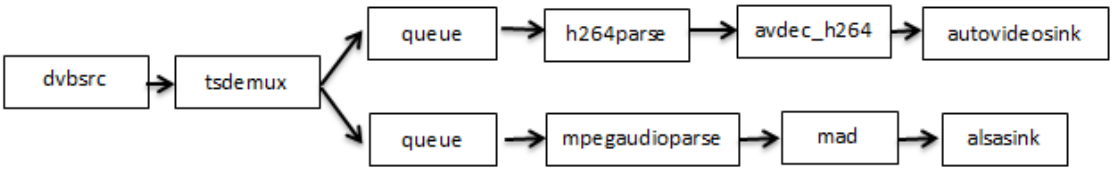
- adapter=0: se indica que numero contiene de adaptador en el pc la sintonizadora USB.
- inversion=AUTO: se indica de forma automática la información de la inversión.
- modulation=AUTO: se establece la modulación de forma automática.
- trans-mode=AUTO: se establece el modo de transmisión de forma automática.
- bandwidth=8: indica el ancho de banda del multiplex en MHz.
- frequency=626000000: indica la frecuencia en Hz del multiplex.
- code-rate-lp=AUTO :se establece de forma automática la tasa de codificación de baja prioridad
- code-rate-hp=AUTO: se establece de forma automática la tasa de codificación de alta prioridad.
- guard=AUTO: se establece de forma automática el intervalo de guarda.

- hierarchy=AUTO: se establece de forma automática la información de jerarquía.

### *3.2.2. Reproducción de un programa en formato HD de la señal de TV (DVB-T).*

Por otro lado, el script para los canales en formato HD, el elemento dvbsrc será del mismo tipo, ya que solo difieren con el de SD en la forma de decodificar el programa. Este sería el resultado del script para los canales HD:

```
gst-launch-1.0 dvbsrc adapter=0 inversion=AUTO modulation=AUTO trans-mode=AUTO bandwidth=8 frequency=626000000 code-rate-lp=AUTO code-rate-hp=AUTO guard=AUTO hierarchy=AUTO ! tsdemux name=demux program-number=151 demux. ! queue max-size-buffers=0 max-size-time=0 ! h264parse ! avdec_h264 ! autovideosink demux. ! queue ! mpegaudioparse ! mad ! alsasink
```

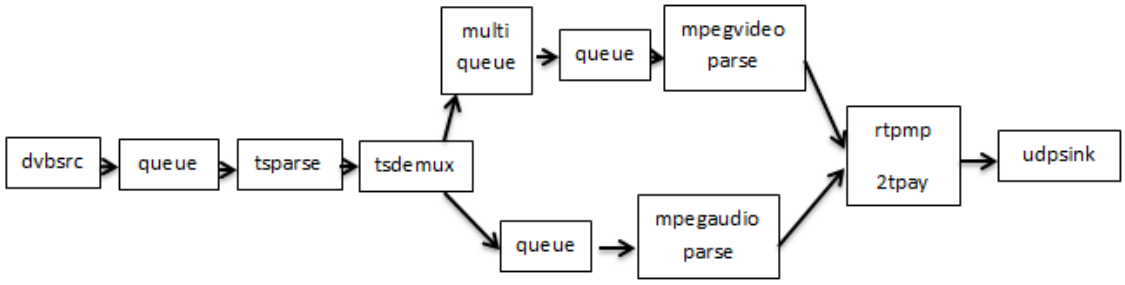



### 3.1.3. Streaming de contenido audiovisual.

#### *3.1.3.1. Streaming de contenido audiovisual en formato SD*

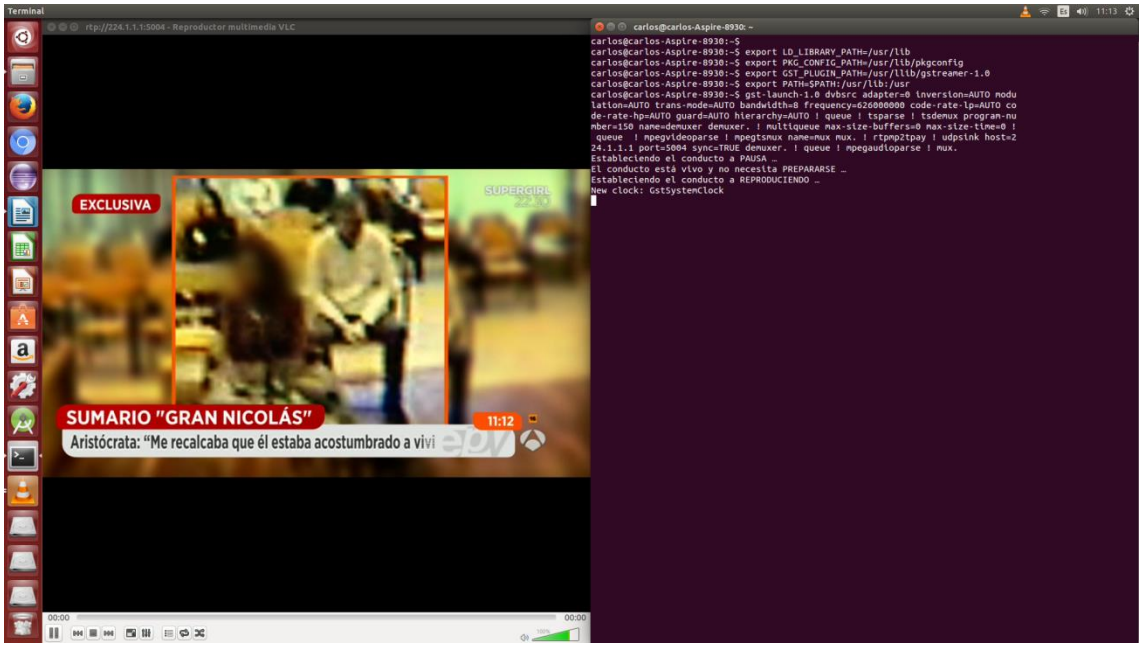
Para emitir desde el servidor, como se ha visto anteriormente, primero se decodifica el material multimedia que se desea. Como se quiere usar para difundir dicho material, se debe de volver a codificar para poder ser enviado mediante RTP. El script tendría la siguiente forma:

```
gst-launch-1.0 dvbsrc adapter=0 inversion=AUTO modulation=AUTO trans-mode=AUTO bandwidth=8 frequency=626000000 code-rate-lp=AUTO code-rate-hp=AUTO guard=AUTO hierarchy=AUTO ! queue ! tsparse ! tsdemux program-number=150 name=demuxer demuxer. ! multiqueue max-size-buffers=0 max-size-time=0 ! queue  ! mpegvideoparse ! mpegtsmux name=mux mux. ! rtpmp2tpay ! udpsink host=224.1.1.1 port=5004 sync=TRUE demuxer. ! queue ! mpegaudioparse ! mux.
```



- mpegtsmux: encargado de multiplexar el flujo multimedia de un programa en un TS.
- name=mux mux. : se encarga de bifurcar en este punto la pipeline.
- rtpmp2tpay: Carga útil MPEG 2 TS codificada en paquetes RTP (RFC 2250).
- Udpsink: enviar datos a través de la red a través de UDP.
- host=224.1.1.1 dirección donde enviar los paquetes.
- port=5004: puerto a emplear para enviar los paquetes.
- sync=TRUE: propiedad para mantener una sincronización.

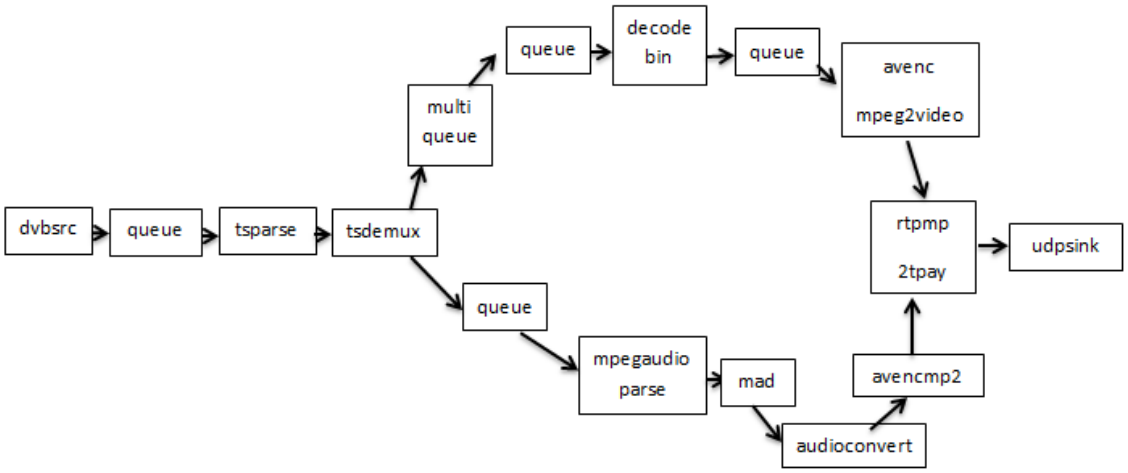
Para comprobar el funcionamiento, se hace uso del reproductor VLC media player para poder visualizar su contenido.




### *3.1.3.2. Streaming de contenido audiovisual en formato HD.*

El script para el servidor de contenido en HD sería el siguiente:

```
gst-launch-1.0 dvbsrc adapter=0 inversion=AUTO modulation=AUTO trans-mode=AUTO bandwidth=8 frequency=626000000 code-rate-lp=AUTO code-rate-hp=AUTO guard=AUTO hierarchy=AUTO ! queue ! tsparse ! tsdemux program-number=149 name=demuxer demuxer. ! multiqueue max-size-buffers=0 max-size-time=0 name=mq ! decodebin ! queue  ! avenc_mpeg2video ! mpegtsmux name=mux ! rtpmp2tpay ! udpsink host=224.1.1.1 port=5004 sync=TRUE demuxer. ! mq.  mq. ! mpegaudioparse ! mad ! audioconvert ! avenc_mp2  ! mux.
```




Respecto al de formato SD, en este se puede observar que al emplear otro tipo codificación en el multiplex debe modificarse la parte del video de la pipeline. Además, cabe mencionar que para poder enviar canales en formato HD es necesario convertirlo a formato SD.

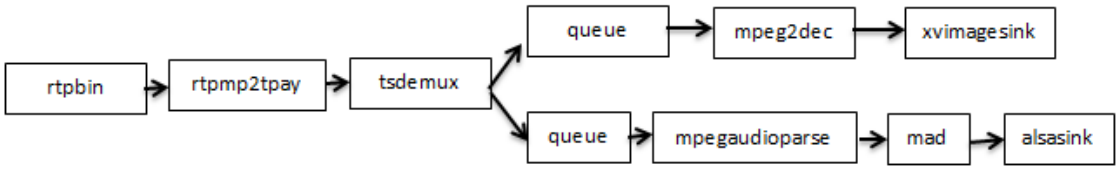
- decodebin: permite el arranque automático de decodificación a RAW[1] .
- avenc_mpeg2video : encargado de utilizar la librería libav mpeg2video para codificarlo y ser multiplexado.

### 3.1.4. Recepción de contenido audiovisual

#### *3.1.4.1. Recepción de contenido audiovisual en formato SD.*

El script que genera la posibilidad de recepción (cliente) de material audiovisual en formato SD a través de un servidor RTP contiene la siguiente forma:

```
gst-launch-1.0 –m rtpbin latency=200 udpsrc uri=udp://224.1.1.1
caps="application/x-rtp"  port=5004 ! rtpmp2tdepay ! tsdemux emit-stats=TRUE
name=demux demux. ! queue !  mpeg2dec ! xvimagesink demux.! queue !
mpegaudioparse ! mad ! alsasink
```




[1] formato de archivo digital de imágenes que contiene la totalidad de los datos de la imagen tal y como ha sido captada por el sensor digital de la cámara.

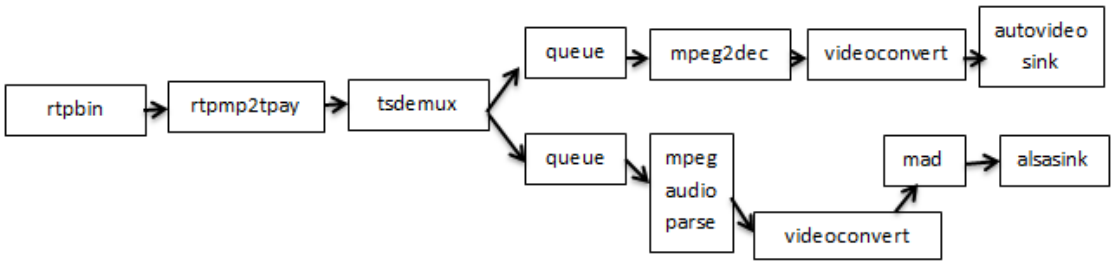
Se definen los elementos aun no empleados para explicar su función:

- rtpbin: indica uso de la librería de plug-in de gestión de sesiones RTP.
- latency=200: latencia permitida en la conexión. (valor por defecto).
- udpsrc: indica que la información se va a enviar vía UDP.
- uri=udp://224.1.1.1 : indica la dirección a la que se quiere conectar para recibir el flujo de datos.
- caps="application/x-rtp"
- port=5004 : puerto que se emplea para la conexión.

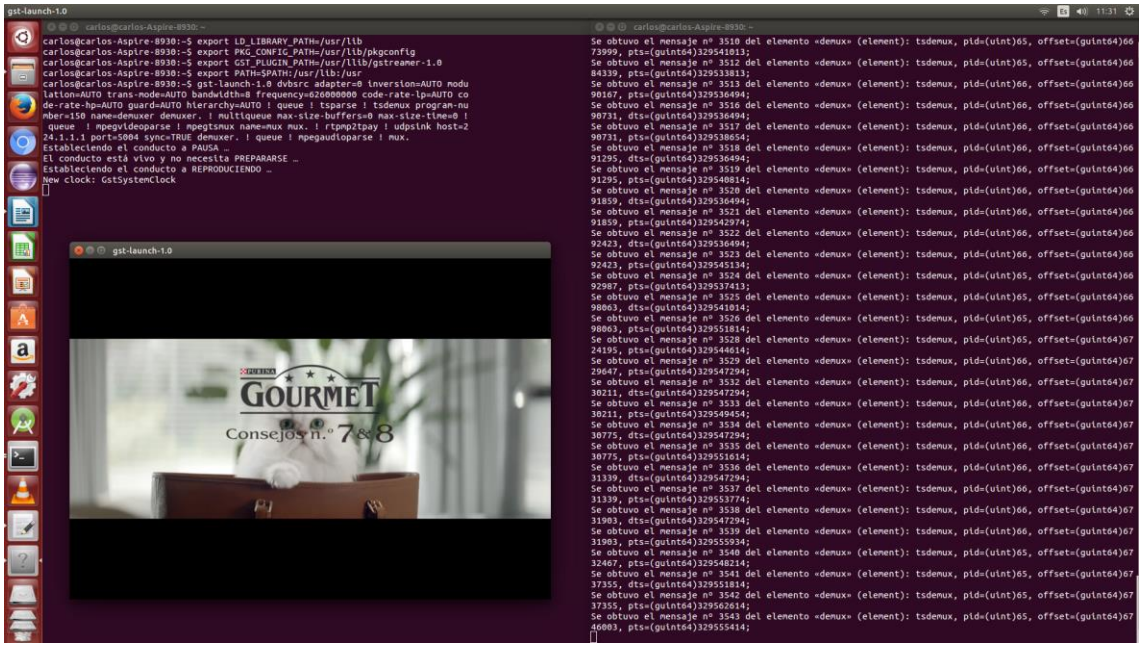
### *3.1.4.2. Recepción de contenido audiovisual en Formato HD*

En el caso del cliente para reproducir canales en formato HD, el script tendría la siguiente forma:

```
gst-launch-1.0 –m rtpbin latency=200 udpsrc uri=udp://224.1.1.1
caps="application/x-rtp"  port=5004 ! rtpmp2tdepay ! tsdemux name=demux
demux. ! queue ! mpeg2dec ! videoconvert ! autovideosink demux. ! queue !
mpegaudioparse ! mad ! audioconvert ! alsasink
```




Para comprobar el funcionamiento, en un terminal se iniciaría el script de servidor y en otro terminal se iniciará el cliente correspondiente.




## 3.5. Generacion contenido HLS.

En este apartado se presenta una pipeline que genera contenido HLS usando Gstreamer.

```
dvbsrc adapter=0 inversion=AUTO modulation=AUTO trans-mode=AUTO bandwidth=8
frequency=770000000 code-rate-lp=AUTO code-rate-hp=AUTO guard=AUTO hierarchy=AUTO ! queue !
tee name=tee0 tee0. ! queue ! tsparse ! tsdemux program-number=570 name=demuxer21 ! queue !
mpegvideoparse ! mpegtsmux name=muxer21 ! hlssink max-files=6 playlist-
location=/var/www/html/HLStest/playlistSDLa1.m3u8 target-duration=4
location=/var/www/html/HLStest/segmentoSDLa1%05d.ts demuxer21.audio_00cb ! queue !
mpegaudioparse ! muxer21. demuxer21.audio_00cc ! queue ! mpegaudioparse ! muxer21.
```

Como se puede observar, el elemento *dvbsrc* contiene los siguientes subelementos:

- *adapter=0*: se indica que numero contiene de adaptador en el pc la sintonizadora USB.
- *inversion=AUTO*: se indica de forma automática la información de la inversión.
- *modulation=AUTO*: se establece la modulación de forma automática.
- *trans-mode=AUTO*: se establece el modo de transmisión de forma automática.
- *bandwidth=8*: indica el ancho de banda del multiplex en MHz.
- *frequency=626000000*: indica la frecuencia en Hz del multiplex.

- *code-rate-lp=AUTO* :se establece de forma automática la tasa de codificación de baja prioridad

- *code-rate-hp=AUTO*: se establece de forma automática la tasa de codificación de alta prioridad.

- *guard=AUTO*: se establece de forma automática el intervalo de guarda.

- *hierarchy=AUTO*: se establece de forma automática la información de jerarquía.

*Queue* : cola simple de datos.

*Tsparse* : analiza el flujo de transporte MPEG-2, permite comprobar que no hay errores.

*Mpegvideoparse :* identifica y comprueba que el flujo sea del tipo mpeg.

*Tsdemux* : decir que las subclases se ha puesto por investigación y para que pudiera funcionar. Se debe poner el program number correspondiente al flujo a visualizar, y además, se escoge que este elemento vaya compartiendo variables como el PTS con la aplicación desarrollada. Además se le da nombre a este elemento para poder tener una bifurcación en la pipeline, una rama para vídeo y la otra para audio.

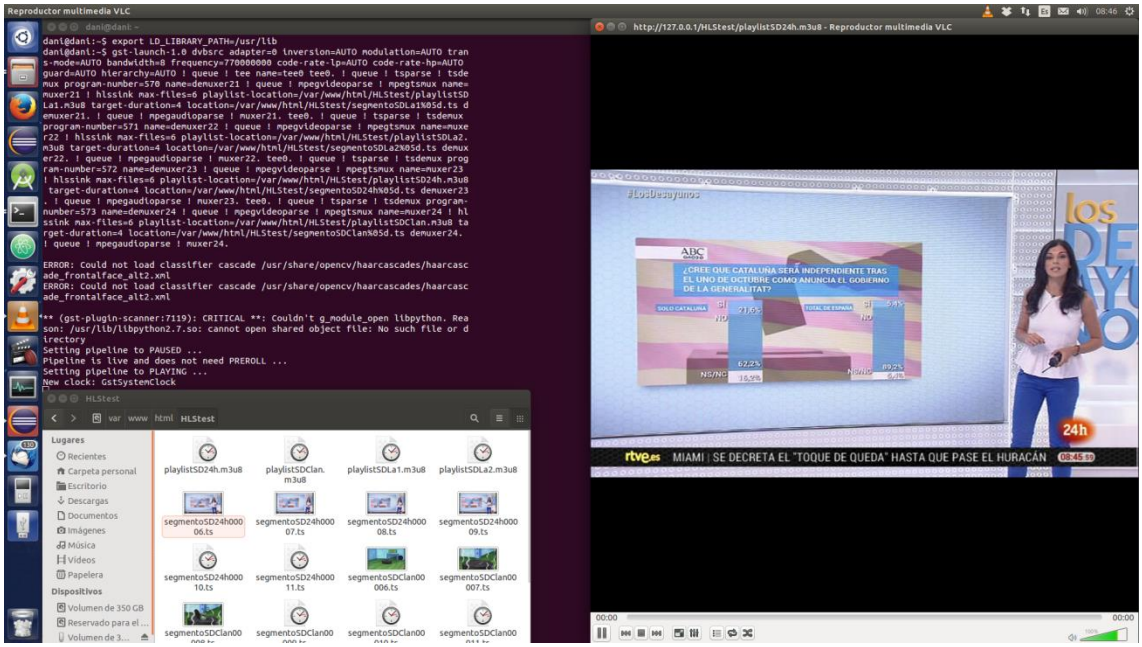
*Mpegtsmux* : encargado de multiplexar el flujo multimedia de un programa en un TS.

*Hlssink max-files=6* : indica que el número máximo de los archivos generados por *HLS.* En este caso son 6

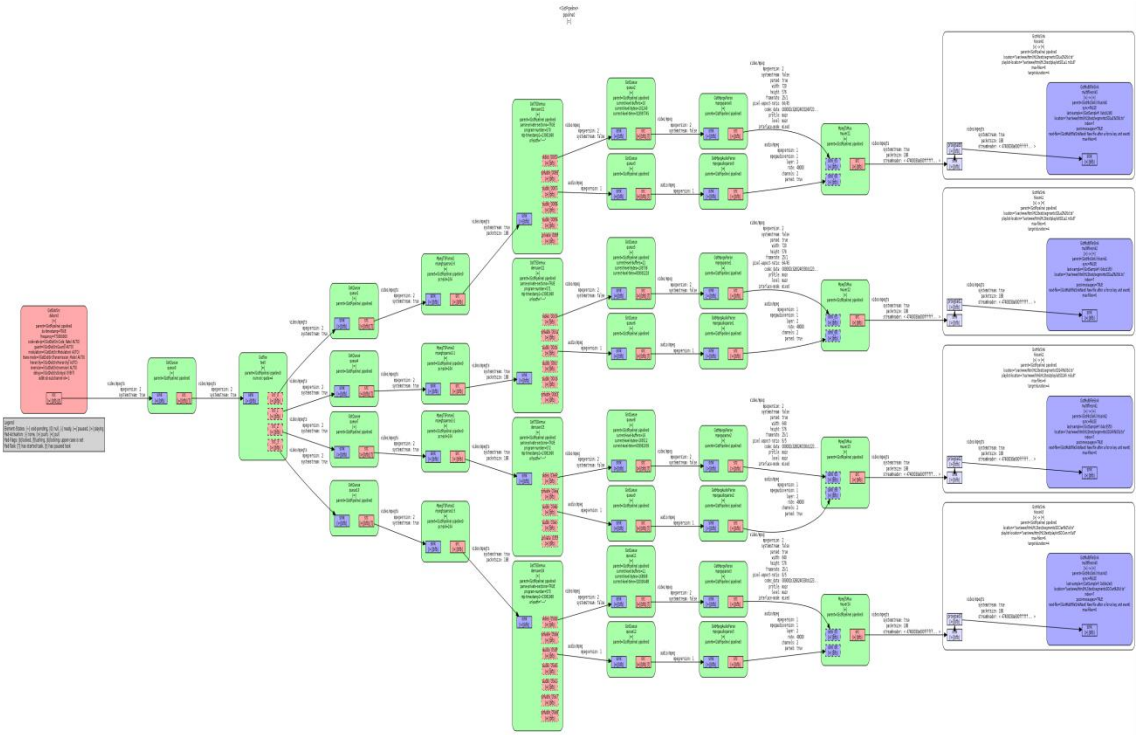
*Playlistlocation=/var/www/html/HLStest/playlistSDLa1.m3u8*: indica en que raíz del directorio del servidor se encuentra el archivo con extensión .m3u8

*target duration=4* : indica que cada segmento es de 4 segundos de duración

*location=/var/www/html/HLStest/segmentoSDLa1%05d.ts*:indica donde se almacenan los segmentos ts .El %05d indica que cada segmento tendrá 5 enteros en el nombre es decir que ira generando segmentos tipo antena300000.ts, antena300001.ts, antena300002.ts etc hasta conformar 5.




Este ejemplo corresponde con el canal de La 1 de TVE (Television Española). Puesto que este tipo de pipeline es la que se va utilizar en el proceso de desarrollo del trabajo, se generá un esquema para observar sus elementos reales. En este caso es decodificado todo el multiplex de RTVE (Radio Televisión Española).



## 4.Instalación sintonizadora TDT por USB.

Una vez obtenidas las pipelines encargadas de reproducir el material multimedia tanto en formato de calidad SD y HD, el siguiente paso será obtener dicho archivo pero en tiempo real, es decir, ya no se trata con archivo almacenado sino que se retransmitirá en directo. Para ello, primero se debe instalar la sintonizadora conectada por USB.

Para instalarla, basta con descargarse los controladores correspondientes al dispositivo receptor de la señal TDT y luego instalarlos. El proceso empieza abriendo un nuevo terminal e insertar los siguientes comandos.

```
sudo apt-add-repository ppa:chrisfu/rt2832u-dkms

sudo apt-get update

sudo apt-get install rtl2832u-dkms
```

Para ver la tv con el pc se necesita dvb-tools para poder escanear la red TDT española con ayuda del sintonizador conectado por puerto USB. Su instalación se realiza con los siguientes comandos

```
sudo apt-get install dvb-tools
scan -o zap /usr/share/dvb/dvb-t/es-Valencia > channels.conf
```

Para más información, visitar estos links de interés:

http://blog.hardc0re.org.uk/2012/09/realtek-rtl2832u-dvb-t-on-ubuntu-1204.html

http://blog.desdelinux.net/como-ver-la-tv-digital-abierta-en-linux/

## 5.Instalación Eclipse en Ubuntu.

En este anexo se describe el proceso de descarga e instalación de Eclipse en el S.O de ubuntu. Los pasos a seguir son los siguientes:

Paso 1: descargar el IDE de Eclipse

Existen diferentes versiones del IDE de Eclipse y diferentes paquetes de distribución dependiendo del procesador de nuestro equipo, en este caso descargaré la versión de 64 bits para Linux.

Paso 2: descomprimir el fichero descargado

Por lo general, el fichero descargado se almacena en la carpeta /Downloads del directorio /home. Movemos el fichero descargado al directorio /opt con el siguiente comando desde un terminal

```
sudo mv Download/eclipse-* /opt/
```

Y ahora descomprimimos el fichero con el siguiente comando

```
cd /opt

sudo tar -xvf eclipse-java-luna-R-linux-gtk-x86_64.tar.gz
```

Paso 3: creamos un fichero Desktop

Ahora vamos a crear un fichero .desktop para tener un acceso directo al IDE de Eclipse en la barra de accesos directos. Nos movemos al directorio /usr/share/applications

```
cd /usr/share/applications
```

y creamos el fichero eclipse.desktop

```
sudo gedit /usr/share/applications/eclipse.desktop
```

lo editamos con las siguientes líneas y lo guardamos

```
[Desktop Entry]

Name=Eclipse

Type=Application

Exec=env UBUNTU_MENUPROXY= /opt/eclipse/eclipse

Terminal=false

Icon=/opt/eclipse/icon.xpm

Comment=Integrated Development Environment

NoDisplay=false

Categories=Development;IDE;

Name[en]=eclipse.desktop

X-Desktop-File-Install-Version=0.22
```

Paso 4: creamos un enlace simbólico

Con el siguiente comando instalamos el eclipse.desktop en el Unity

```
sudo desktop-file-install /usr/share/applications/eclipse.desktop
```

Creamos un enlace simbólico al IDE de Eclipse con los siguientes comandos

```
cd /usr/local/bin/

sudo ln -s /opt/eclipse/eclipse
```

Y para que el icono de Eclipse aparezca en el buscador de recursos de Ubuntu, ejecutamos el siguiente comando

```
sudo cp /opt/eclipse/icon.xpm /usr/share/pixmaps/eclipse.xpm
```

Por último, lanzamos el IDE de Eclipse para comprobar que todo funciona correctamente.

## 6.Instalación Android Studio en Ubuntu.

**Paso 1** : Abrir un terminal presionando ***Ctrl + Alt + T .***

**Paso 2** : Agregar el repositorio con el siguiente comando:

```
sudo add-apt-repository ppa: paolorotolo / android-studio
```

**Paso 3** : Actualizamos el APT con el comando:

```
sudo apt-get update
```

**Paso 4** : Ahora instalamos el programa con el comando:

```
sudo apt-get install android-studio
```

**Paso 5** : Una vez instalado, ejecute el programa escribiendo en el terminal:

```
studio
```

# Anexo II. Flujos de programa y de transporte de MPEG2.

En cada estándar se definen los esquemas de codificación de canal y modulación para el medio de transmisión que se trate, pero, en cualquiera de los casos, la codificación de fuente es adaptación del estándar MPEG-2. La seña de material de entrada y de salida audiovisual se denomina *MPEG-2 Transport Stream* (TS) o flujo de transporte. Para emplearse en DVB, se debe complementar haciendo uso de *Información del Servicio* (SI). Se multiplexan varios programas audiovisuales formando un múltiplex donde cada programa está compuesto por uno o varios flujos elementales (ES) paquetizados (PES) y encapsulados en paquetes de transporte (transport packets). Cada uno de estos paquetes, al mismo tiempo, está marcado con PIDS que indica a qué tipo de flujo elemental pertenecen.

Para una mejor aclaración, un Programa, empleando la tecnología MPEG, es un servicio o canal simple de radiodifusión que cuando se comprime, cada componente simple del programa recibe el nombre de Elementary Stream (ES). Un programa puede estar compuesto por varios ES (vídeo, audio, subtítulos, teletexto...). Cada ES se estructura en paquetes que reciben el nombre de Packetised Elementary Stream(PES), es decir, por cada PES habrá un ES original.

Para que un decodificador pueda recuperar por completo un programa a través de la ayuda que le ofrecen los PIDS, es necesario incluir información adicional dentro del flujo de transporte para que relacione los valores de PIDS con los programas a los que pertenecen. Esta información recibe el nombre de Program Service Information (PSI) o Información Específica de los Programas.

La tabla que representa PSI definida por MPEG-2 está compuesta por cuatro tipos:

- PAT (*Program Association Table*). Tabla de inclusión obligatoria que contiene lista de todos los programas disponibles en el TS.

- CAT (*Condicional Acces Table*).Esta tabla aparecerá únicamente si algún canal del multiplex es de acceso condicional. Esta información no está especificada en MPEG-2 ya que depende del tipo de sistema de cifrado que se haya empleado.

- PMT (*Program Map Table*). Esta tabla proporciona información acerca del programa y de los flujos que se asocian a el.

- NIT (*Network Information Table*). Incluye información de red.

También hay que mencionar *MPEG-2 Program Stream* (PS) o también denominado flujo de programa, ya que este flujo de señal se usa para almacenar y recuperar información digital en entornos libres de cualquier error. A diferencia de TS, este sólo puede multiplexar un solo programa.

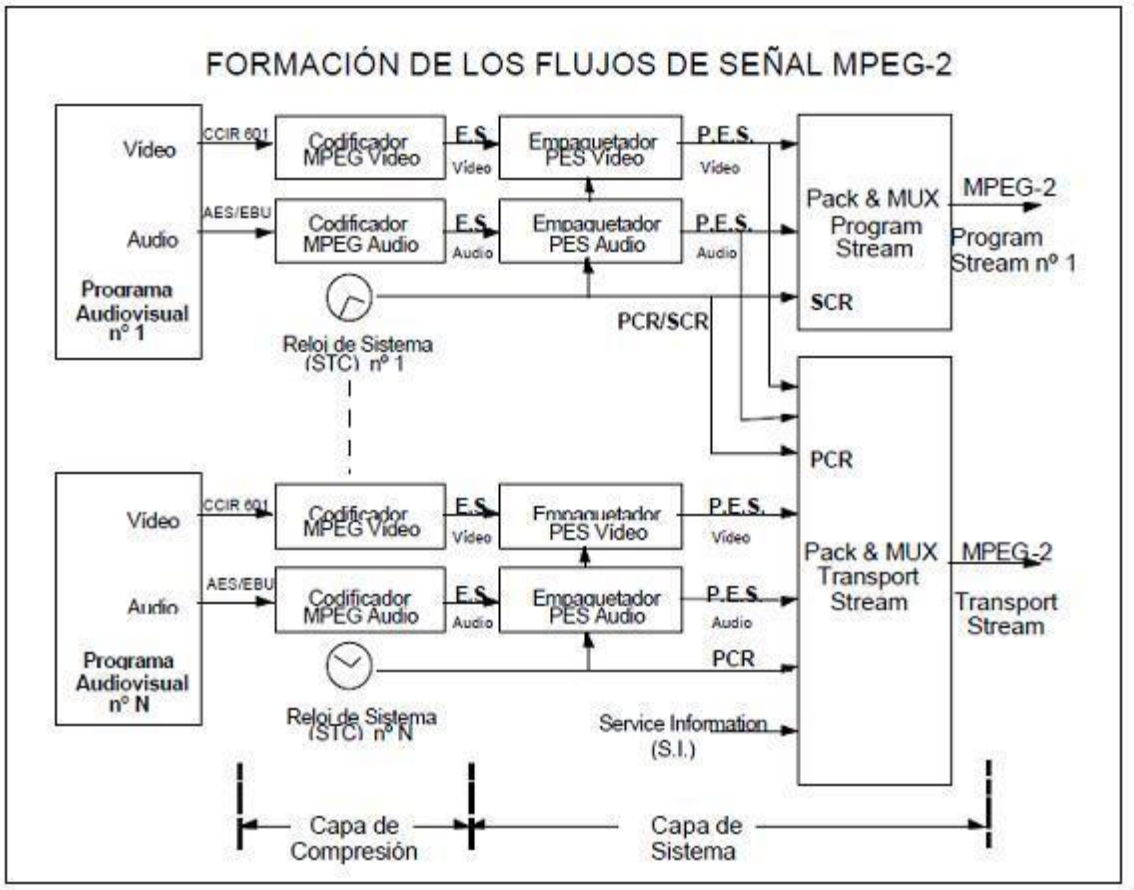



**Imagen 1: Esquema resultante para la formación de flujos de señal MPEG-2. [3]**

En el caso de PS, se multiplexan el audio, vídeo, datos, etc... y se le añade un reloj del sistema. Sólo se transmite la información perteneciente a un único programa en el que se debe de compartir, de forma obligatoria, un mismo reloj de referencia.

El PS está formado por PACKS que a su vez están formados por cabecera del pack (pack-header), otra opcional (system-header) y numerosos PES-packets pertenecientes a los ES del programa audiovisual.

Estos packs no tienen definida una longitud de paquete.

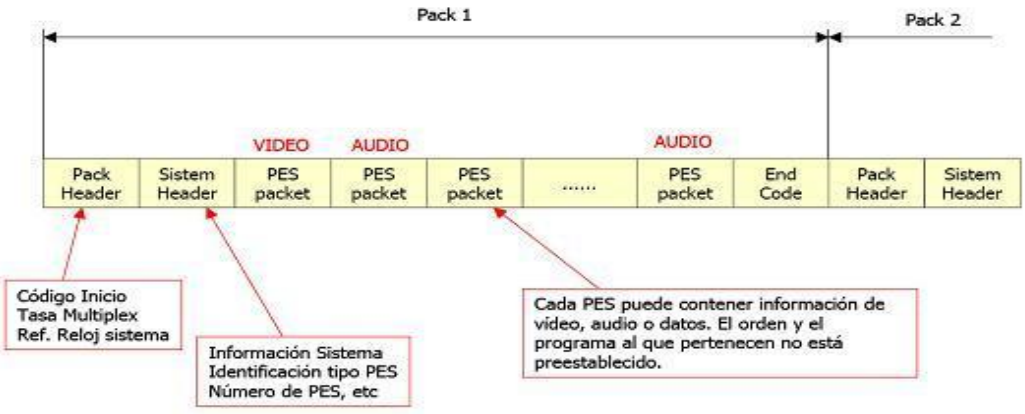



**Imagen 2: Estructura que conforma un pack de Program Stream. [4]**

Por otro lado, un múltiplex TS contiene varios programas (PS) y, además, incluye más información relacionada con el servicio, como pueden ser la Tabla de Asociación de Programas(PAT), la Tabla con Información para Acceso Condicional(CAT), y la Tabla de Mapa de cada Programa(PMT)...

Cada paquete de transporte de TS siempre tiene una longitud fija de 188 bytes (cabecera de 4 bytes a veces seguida de otro campo denominado de adaptación, que en caso de no estar lleno se rellena el exceso de espacio disponible, y finalmente un campo llamado de carga útil o Payload).

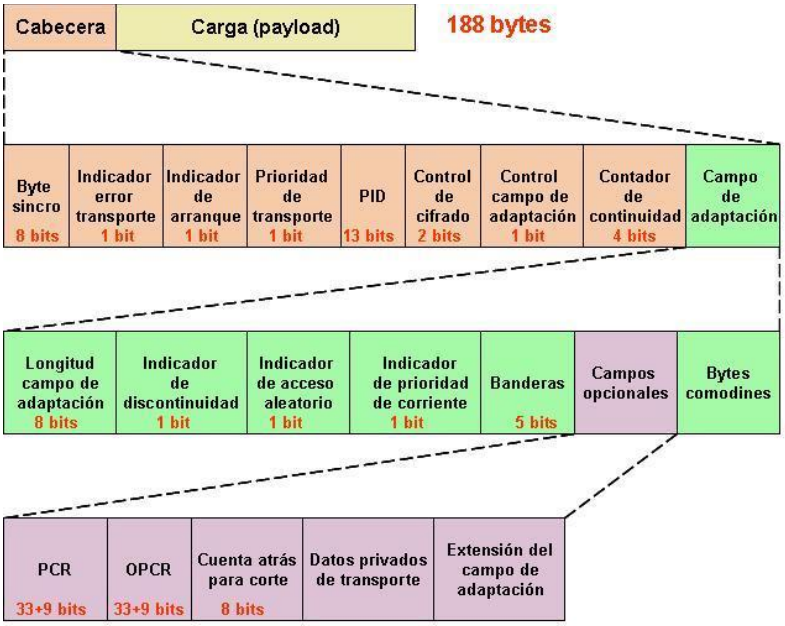



**Imagen 3: Estructura de un paquete de Transport Stream. [5]**

La construcción de estos paquetes de transporte, debe cumplir las siguientes condiciones:

- El primer byte de cada PES packet tiene que ser el primer byte del payload del transport packet.
- El transport packet contiene únicamente datos obtenidos de un PES packet.

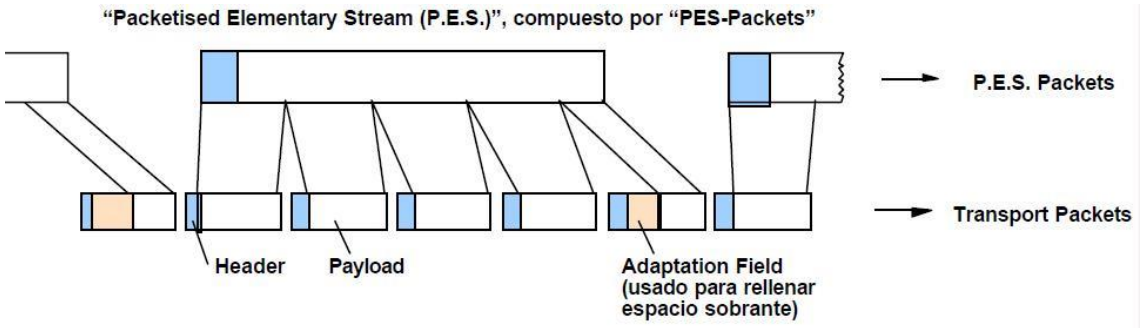



**Imagen 4: En esta imagen se puede observar el desglose de P.E.S. en una serie de Transports Packets. [6]**

Es improbable que un PES packet pueda rellenar la carga útil con un numero entero de paquetes de transporte de manera precisa. En ocasiones, el espacio sobrante del último paquete de transporte correspondiente a un PES packet, se rellena de forma deliberada mediante un campo de adaptación de longitud apropiada.

Para la formación del Transport Stream, se le aplica a cada flujo elemental de señal, ya sea vídeo, audio o datos, se le aplica el proceso descrito anteriormente. Estos paquetes no siguen ningún tipo de orden a la hora de ser multiplexados. Solo se respeta el orden de paquetes pertenecientes a cada flujo elemental. Además de estos paquetes, también se multiplexan dos tipos de paquete de transporte, el primero serían los paquetes de información de servicio (SI) y el segundo, paquetes de transporte nulos que se utilizan para reservas de capacidad del multiplex.

Para continuar, seguidamente se describe en que consiste un Transport packet.

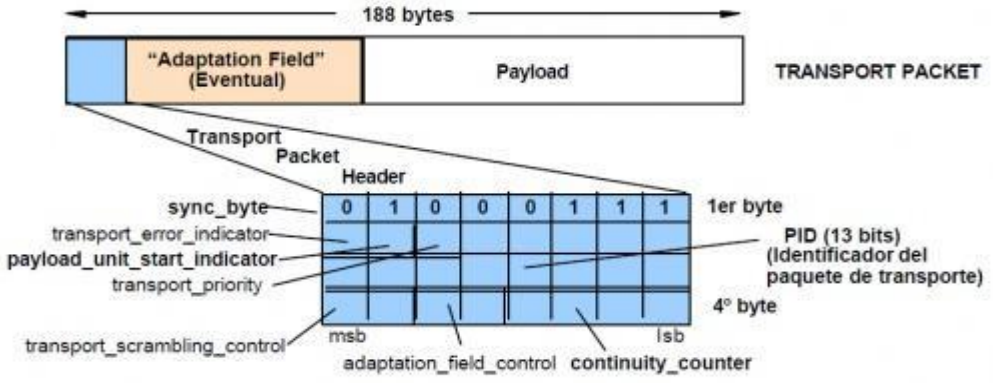



**Imagen 5: Desglose de un Transport Packet.**

Se divide en tres partes, Transport Packet Header que se describe a continuación, el campo de adaptación y carga útil descritos previamente.

Transport Packet Header tiene una longitud fija de cabecera de 4 bytes a diferencia de la cabecera de los PES packets que era variable. De esta nueva cabecera podemos destacar 4 elementos de los más importantes.

- Sync Byte. Este campo pertenece al primer byte de la cabecera y siempre tiene el valor 47 en hexadecimal. Este byte aparece cada 188 bytes con el fin de que su localización de comienzo de cada paquete de transporte en los decodificadores sea más fácil.

- Packet Identifier (PID). Como bien se ha dicho anteriormente, un TS puede contener diferentes programas cada uno con sus respectivos flujos elementales distribuidos en paquetes de transporte. Pues bien, este campo de 13 bits se encarga de identificar los distintos paquetes de transporte asociados a un determinado flujo elemental de entre todos los demás.
  De sus posibles valores que puede poseer, $2^{13}$ valores, se reservan 17 para usos especiales, por tanto solo quedan 8175 posibles valores que determina el valor máximo que puede contener un TS.
- Payload Unit Start Indicator. Este campo solo contiene un bit que se pone a 1 para poder indicar que el primer byte del payload del paquete de transporte es también el primer byte de un PES packet.
- Continuity count field. Contiene 4 bits. Su función es incrementarse entre sucesivos paquetes de transporte pertenecientes al mismo flujo elemento. Con esto se consigue que el decodificador pueda detectar la ganancia o pérdida de un paquete y de esta forma poder ocultar errores que podrían aparecer.

Respecto al valor que contiene los PID, para que un decodificador pueda recuperar por completo los paquetes de flujo correspondientes a cada programa, es necesario incluir información adicional dentro del flujo de transporte que relacione esos valores de PID con sus respectivos programas. Esta información recibe el nombre de Información Específica de los Programas o Program Specific Information (PSI).

La información PSI viene definida por MPEG-2 y establece la inclusión dentro del flujo de transporte de cuatro tipos de tablas como la Program Association Table (PAT), Conditional Acces Table (CAT), Program Map Table (PMT) y Private.

Para sistemas DVB, la Información de servicio o Service Information (SI), incluye otras 4 tipos de tablas obligatorias dentro del TS y 3 de tipo opcional.

Las tablas que son obligatorias son las siguientes:

-Network Information Table (NIT). Si esta tabla se encuentra presente, por definición constituye el programa nº 0 del multiplex y además está considerada como tabla de datos privados que pueden contener información como tipo de red física utilizada para transmitir el TS, frecuencias de canal, características de modulación… definida por el radiodifusor y no por MPEG.

Ejemplo PID=0x0010

-Service Description Table (SDT).Esta tabla contiene datos que describen los servicios del sistema como nombres del servicio, nombre del proveedor y más parámetros asociados a cada servicio de un mismo multiplex.

Ejemplo PID=0x0011

-Event Information Table (EIT). Se usa para transmitir información relativa a los acontecimientos en curso o futuros en el multiplex, es decir, hora de comienzo, duración…

Ejemplo PID=0x0012

- Time & Date Table (TDT). En esta tabla se proporciona información relativa a la hora y fecha del momento para usarse para poner en hora el reloj interno del receptor de la señal.

Ejemplo PID=0x0014

Y por otro lado, están las tablas opcionales de DVB-SI descritas a continuación:

-*Bouquet Association Table* (BAT). El término bouquet se usa para referirse a una colección de servicios comercializados como entidad única. Además de proporcionar información esta información relativa a los bouquets, también informa del nombre del bouquet y la lista de servicios disponibles que puede ofrecer cada bouquet.

Ejemplo PID=0x0011

-*Running Status Table* (RST).Esta tabla se encarga de actualizar de la forma más rápida información relativa a la situación de un acontecimiento, es decir que está o no está sucediendo. Solo se transmite una vez.

Ejemplo PID=0x0013

-*Time Offset Table* (TOT). Esta tabla ofrece información sobre la fecha y hora real.

Ejemplo PID=0x0014

-*Stuffing Tables* (ST).Son tablas de relleno que se utilizan para invalidar tablas que ya no sirven.

Ejemplos PID=0x0010=0x0011=0x0012=0x0013=0x0014

Volviendo a la información PSI, seguidamente se describirán sus tres tablas más significativas.

- *Program Association Table* (PAT). Contiene una lista con todos los programas disponibles en el TS. Cada programa contiene la tabla con los datos que identifican a dicho programa(PMT).
  Ejemplos PID=0x0000
- *Conditional Acces Table* (CAT). Se encarga de proporcionar detalles de sistema de cifrado empleado y también proporciona el valor de los PID de los paquetes de transporte que contienen información de control de acceso condicional.
- *Program Map Table* (PMT). Para cada programa audiovisual incluido en un TS hay una tabla PMT asociado a dicho programa. Esta tabla proporciona información sobre el programa y el flujo elemental que lo compone.