Trabajo Fin de Grado

Administración y monitorización de una red para contribución de vídeo SRT entre dos centros de producción de televisión.

Autor: José Manuel Gómez Líndez

Tutor: Miguel Martín Mateo

Grado de Ingeniería de Tecnologías y Servicios de Telecomunicación

Administración de Redes y Sistemas Operativos

Agradecimientos

A mi dos estrellas, Ana y Dani. Gracias por alumbrarme el camino.

Resumen

Las redes IP y sus protocolos de comunicación han evolucionado haciendo habitual las conexiones de banda ancha, convirtiéndose en una alternativa real para la distribución de vídeo en los centros de producción de televisión.

Gracias a Internet es posible sustituir redes dedicadas de vídeo por redes IP para contribuciones de vídeo por *streaming* en tiempo real. Esto implica una reducción en los costes y en el tiempo de despliegue. Por otro lado, el reto está en garantizar la integridad del servicio manteniendo un alto ancho de banda constante en el tiempo.

Teniendo esto en cuenta, se tomará como supuesto para este TFG definir la topología de red, conexión y configuración necesarias, que permita a un centro de producción de televisión realizar una contribución de vídeo en tiempo real hacia otro(s) centro(s) de producción a través de Internet.

El entorno de aplicación para este TFG será simulado utilizando *software* de virtualización.

Se creará una herramienta de monitorización que mostrará las métricas relevantes de los equipos. Permitirá definir los posibles fallos que sufra el sistema y el punto donde se deberá intervenir para solucionarlo.

Abstract

IP networks and their communication protocols have evolved to make broadband connections commonplace, becoming a real alternative for video distribution in TV production facilities.

Thanks to the Internet, it is possible to replace dedicated video networks with IP networks for real-time streaming video contributions. This implies a reduction in costs and deployment time. On the other hand, the challenge is to guarantee the integrity of the service while maintaining a constant high bandwidth over time.

With this in mind, the assumption for this TFG is to define the network topology, connection and configuration necessary to allow a broadcaster centre to make a real-time video contribution to other production centre(s) over the Internet.

The application environment for this dissertation will be simulated using virtualization software.

A monitoring tool will be created that will show the relevant metrics of the equipment. It will allow to define the possible failures suffered by the system and the point where it will be necessary to intervene to solve them.

Índice.

Índice de Figuras y Tablas.

1. Introducción.

1.1 Descripción.

En este TFG se va a hacer en primer lugar una descripción técnica de los elementos de la televisión que van a permitir su transporte por redes, centrándose en las redes de datos y, como objetivo, la posibilidad de hacerlo a través de Internet.

Una vez definida la señal que se quiere transmitir, se define el tipo red de datos por la que se quiere transmitir, explicando conceptos relevantes que servirán para apoyar la posterior implementación.

Se engloban algunos de los protocolos de *streaming* de vídeo más importantes y utilizados actualmente en el entorno de la televisión profesional y se destacan sus características más relevantes. En este TFG se abordan los protocolos de *streaming* ideados para la televisión en tiempo real o IPTV, descartando los desarrollados para aplicaciones como vídeo bajo demanda.

La red de contribución en televisión, como se verá, establece unos determinados criterios de calidad, de ahí el protocolo de vídeo *streaming* elegido y criterios de calidad en el servicio.

En la implementación se va a definir una electrónica de red que satisfaga los requisitos establecidos para la aplicación multimedia descrita. Se analiza la tolerancia a fallos del sistema y se crea un interfaz de usuario que pueda indicar posibles problemas en la conectividad de los elementos que lo componen.

1.2 Justificación.

Los últimos trece años he desarrollado mi actividad profesional en el entorno de la producción de televisión y en este tiempo su distribución ha ido cambiando conforme la tecnología asociada a las redes de datos e Internet lo han hecho.

Como ejemplo, al inicio de este período, la contribución de televisión por satélite era la más habitual en mi trabajo diario. Sin embargo, con el avance de la tecnología y el despliegue de las redes de telefonía móvil 4G, LTE (ahora en desarrollo 5G), las señales de televisión se reciben en los centros de producción en su gran mayoría utilizando estas redes. Su uso ha proliferado gracias a que su despliegue y mantenimiento suponen un coste menor y además han evolucionado en fiabilidad y reducido la latencia en las conexiones.

Además, en el último año, se han ido sustituyendo la utilización de circuitos de vídeo punto a punto por fibra óptica sin compresión, por circuitos como el que se aborda en este TFG, a través de Internet. De ahí mi interés por profundizar en su funcionamiento.

1.3 Objetivos.

El objetivo principal es presentar, de forma simulada, la electrónica de red, su conexión y la configuración necesarias, que definan y garanticen la transmisión de vídeo por *streaming* entre dos centros de producción de televisión. Para ello:

- Establecer las características del servicio de streaming y seleccionar el equipamiento adecuado para la red entre los servidores de vídeo y los clientes.
- Definir los requisitos de seguridad y consistencia.
- Gestionar la tolerancia a fallos y dotarlo de redundancia.
- Agregar calidad de servicio (QoS) a la red para fiabilidad y rendimiento.
- Realizar una emulación del funcionamiento del sistema mediante *software* de virtualización.
- A partir de la simulación, extraer métricas de funcionamiento de los equipos.
- Implementar una aplicación web con dichos datos que sirva como herramienta de monitorización y habilite la intervención por parte de personal cualificado ante algún fallo detectado en el sistema.

1.4 Alcance.

El alcance de este TFG sería el de poder monitorizar elementos de una red de datos para habilitar una intervención correctiva en caso de fallo.

Para ello se definirá la red y el protocolo de *streaming* de vídeo en tiempo real que favorezca el servicio a través de una red tan heterogénea como Internet.

Se analizará la tolerancia a fallos del sistema y se definirán posibles escenarios de fallo y su posible resolución

Por último, se implementará una interfaz de usuario para que un operador de un centro de monitorización de red pueda intervenir.

1.5 Planificación.

Se detalla a continuación la planificación de trabajos y contenidos para el presente TFG.

1.5.1 Cronograma.

El plan de trabajo se detalla a continuación en la **Tabla 1.**, estructurado en semanas para cada uno de los hitos definidos con la entrega de las diferentes PECs:

Nombre	Duración	Fecha de Inicio	Fecha de Fin
Definición del Tema del Proyecto	3 días	27/09/2023	29/09/2023
PEC 1: Propuesta del Plan de Trabajo	16 días	30/09/2023	15/10/2023
Recopilación de información	4 días	30/09/2023	3/10/2023
Definición de los trabajos a realizar y los objetivos	4 días	4/10/2023	7/10/2023
Planificación. Diagrama de Gantt	3 días	8/10/2023	10/10/2023
Repaso y redacción del plan de trabajo	5 días	11/10/2023	15/10/2023
PEC 2: Entrega del 40% - 60% de la memoria	35 días	16/10/2023	19/11/2023
Recopilación de información sobre el estado del arte.	7 días	16/10/2023	22/10/2023
Instalación del *software* para simulación. Definición de red y conexión.	7 días	23/10/2023	29/10/2023
Redacción de capítulo 2 sobre estado del arte. Contexto de redes en televisión, codificación de vídeo y generación de *streaming* en SRT.	7 días	30/10/2023	5/11/2023
Configuración de los elementos de red. Primera simulación en GNS3.	7 días	6/11/2023	12/11/2023
Redacción de la primera parte del capítulo 3 sobre la implementación. Equipos de red, conexión.	7 días	13/11/2023	19/11/2023
PEC 3 - Entrega 80% - 90% de la memoria	35 días	20/11/2023	24/12/2023
Simulación de diferentes escenarios de trabajo. Análisis de posibles fallos y obtención de datos que permitan su monitorización.	7 días	20/11/2023	26/11/2023
Recopilación de los resultados en la memoria en capítulo 3	7 días	27/11/2023	3/12/2023
Desarrollo de la aplicación *web*. Instalación de *framework.* Configuración del servidor *web* Apache. Comunicación con electrónica de red.	7 días	4/12/2023	10/12/2023
Finalizar aplicación *web* y recoger resultados en documentación.	7 días	11/12/2023	17/12/2023
Revisar y finalizar la memoria.	7 días	18/12/2023	24/12/2023
Entrega Final	21 días	25/12/2023	14/01/2023
Preparar presentación	7 días	25/12/2023	31/12/2023
Ensayo y preparación de la defensa.	7 días	01/01/2024	07/01/2024
Revisión de memoria y presentación. Entrega.	7 días	08/01/2024	14/01/2024

Tabla 1. Cronograma del plan de trabajo

1.5.2 Diagrama de Gantt.

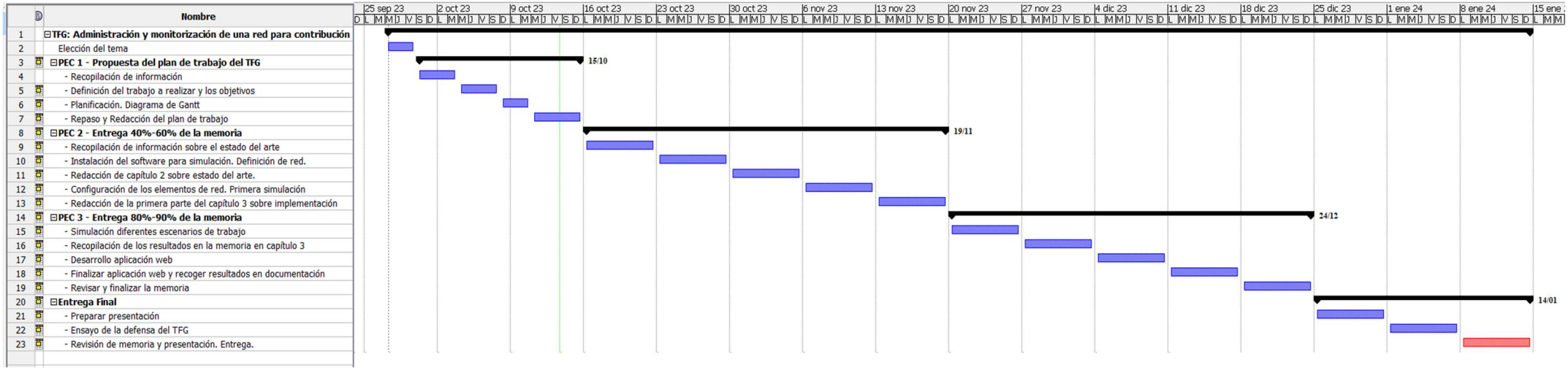

Figura 1. Diagrama de Gantt de la planificación de trabajo

1.6 Recursos.

El entorno de trabajo será simulado y para ello se utilizará el siguiente *software*:

- Para la simulación de la electrónica de red y el servicio de *streaming* de vídeo:
    - **GNS3** (*Graphical Network Simulation*): *Software* de simulación y emulación de redes de datos. *Software* de código abierto y gratuito desarrollado por *SolarWinds Worldwide*.

      Permite incluir en los proyectos imágenes virtualizadas de los diferentes elementos que conforman la red. Supone una herramienta muy potente a la hora de diseñar y administrar redes. Debido a su característica de emulación, permite realizar pruebas fuera del entorno de producción, evitando sus riesgos, y garantizando el buen funcionamiento en la implementación real.
    - **VirtualBox**: Plataforma de gestión de equipos virtualizados que pueden correr cualquier versión de sistemas operativos como Windows, Linux, Solaris, OS/2 y OpenBSD. En este trabajo se utilizará para simular los equipos finales (servidores y clientes). Es un *software* de código abierto y gratuito de Oracle.
    - **VMWare Workstation Player**: Versión gratuita de este *software* de virtualización que se utiliza para instalar la máquina virtual de GNS3 que actúa como servidor de equipos de red virtualizados.
    - **SRT-Live-Transmit**: Software libre alojado y desarrollado por Haivision en *GitHub* que permite la creación de un flujo de salida SRT a partir de otro de entrada. Para este TFG el flujo de entrada será un *MPEG-Transport Stream* generado a partir de un archivo de vídeo.
    - **FFmpeg**: Herramienta con gran versatilidad para procesado de archivos multimedia. Software libre y gratuito que se utilizará para crear un *stream* a partir de un archivo.
    - **Imágenes de la electrónica de red de Cisco**: Ya que Cisco es uno de los fabricantes de equipos más extendido en redes de datos, se utilizarán imágenes virtuales de este fabricante para los equipos de red necesarios.
- Para la implementación del servidor web para la monitorización del sistema:
    - **Python:** Lenguaje de programación de alto nivel multiparadigma (admite programación orientada a objetos, imperativa y funcional). Es un lenguaje interpretado (permite ejecutar otros programas), dinámico (una variable puede tomar diferentes tipos) y multiplataforma (programas que se pueden ejecutar en arquitecturas x86 de Windows, Linux o Mac). Se utiliza la versión 3.10.
    - **Flask de Python:** *Framework* para Python orientado al desarrollo de aplicaciones web.

- **Apache Web Server:** *Software* gratuito y colaborativo que permite habilitar un servicio web para el interfaz de usuario que se va a desarrollar.
- **MySQL:** Gestor de base de datos de lenguaje SQL. Es gratuito y propietario de Oracle.
- **VS Code (Visual Studio Code):** IDE de desarrollo de *software* gratuito y propietario de Microsoft.

Todo este *software* correrá sobre un ordenador portátil con las siguientes características:

- Procesador AMD Ryzen 7 con 8 núcleos de CPU a 2,9 GHz
- 16 GB de memoria RAM DDR4
- Sistema operativo Windows 10 con arquitectura de 64 bits.

1.7 Previsión de problemas.

Se detallan algunos problemas que se podrían encontrar en el desarrollo del trabajo:

- La virtualización de equipos supone un consumo de recursos de procesamiento y memoria que podría limitar las características de la red definida y modificar el alcance de este trabajo.
- El rendimiento en la red conformada con los equipos virtualizados es posible que no se corresponda con el que tendría la red con equipamiento real, afectando a las premisas expuestas en este TFG.
- Los elementos de red virtualizados de Cisco tienen una licencia de uso propietaria, por lo que habría que pagar para utilizarlas.
- Con anterioridad no he trabajado con GNS3 ni conozco en profundidad el interfaz de línea de comandos (CLI) de Cisco, lo que puede suponer una demora con respecto a lo planificado.

2. Estado del arte.

En este capítulo se definirán brevemente las características de la señal de televisión que se pretende transmitir a través de la red definida. Para ello se darán nociones sobre la necesidad de compresión de la señal de televisión y sus características.

Se definirán posibles redes por las que se distribuye la señal de televisión poniendo el foco en las redes de datos IP y sus características.

Así mismo, se darán las nociones suficientes para contextualizar la evolución de las redes de comunicaciones en el ámbito de la televisión y se definirán los objetivos a conseguir con este TFG.

2.1 La señal de televisión (HDTV).

Par a este TFG no se va a entrar en detalle en la señal de televisión ni en su evolución a lo largo de tiempo, simplemente se van a definir unas características técnicas relevantes para el desarrollo de este trabajo. En el ***Anexo 1. Conceptos sobre la señal de televisión.*** se van a especificar ciertos conceptos sobre televisión a los que se irá haciendo referencia en este capítulo.

En general, para el ámbito de este TFG, la señal de vídeo o de televisión hace referencia a una señal de televisión digital conformada por vídeo, audio y datos auxiliares.

La señal de vídeo concretamente contará con una resolución de cuadro HD (*High Definition*) de 1920x1080 píxeles de exploración entrelazada a una frecuencia de 25 imágenes por segundo. Este perfil utiliza la nomenclatura estandarizada de 1080i@50Hz, uno de los perfiles de televisión HD utilizado en Europa.

No se va a tener en cuenta para este trabajo, pero podrían incluirse datos auxiliares como los necesarios para teletexto, subtitulado, descripción de formato, etc., …

Este perfil de televisión HDTV es uno de lo que son distribuidos actualmente en la TDT (Televisión Digital Terrestre) por muchos *broadcasters*.

2.2 Los caminos por donde viaja la televisión.

En el proceso, desde que se genera la señal de televisión en un centro de producción de televisión o *broadcaster*, hasta que llega al cliente final, existen diferentes caminos en los que la señal de televisión tendrá características diferentes.

En la imagen de la **Figura 2**. se ilustran sintetizados los tramos y posibles tipos de redes por los que se distribuye la señal de televisión:

- **Centro de producción o *Broadcaster***: En el estándar ITU-R BT.709, cuya primera edición es de 1990, se definen las características comunes de la señal digital de televisión. Así mismo, en el estándar SMPTE 292M su distribución en el centro de producción, sin comprimir, y por redes de video SDI (*Serial Digital Interface*). Para la señal descrita en el apartado ***2.1 La señal de televisión***, se especifica un régimen binario 1,5 Gbps para las características más exigentes de calidad.

Distribuir esta señal supone una red compleja de conmutadores, sincronizadores, mezcladores, etc., … conectados por cable coaxial. Esta red es exclusiva para este servicio y tiene un elevado coste.

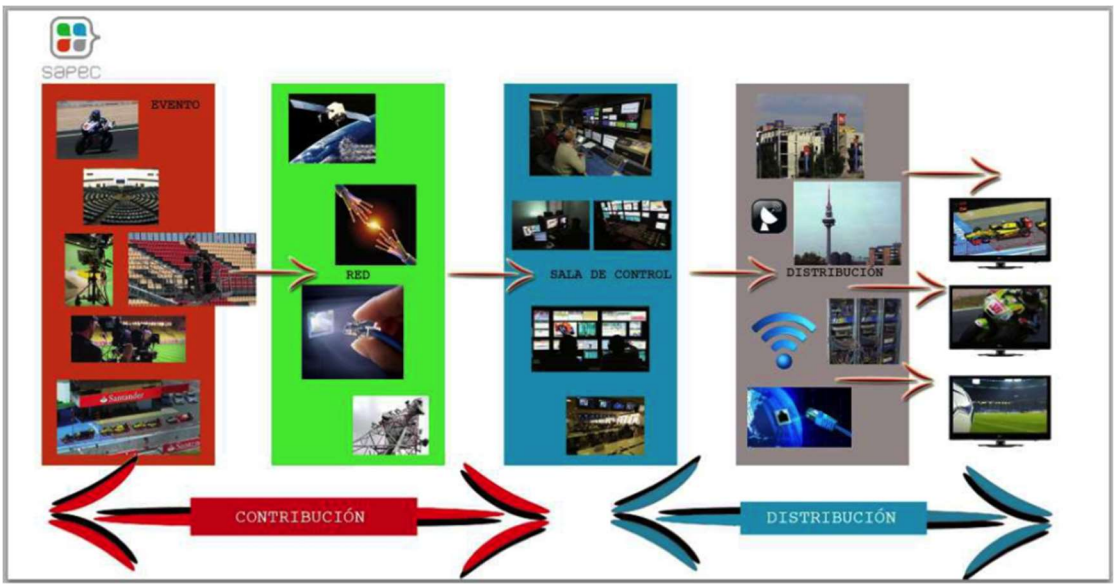

Figura 2. Caminos de la televisión. Fuente: [1]

Actualmente, hay una migración hacia redes IP que consigue las mismas prestaciones que la red HD-SDI y permiten distribuir señales de mayor definición como 4K y 8K (UHDTV). Los conjuntos de estándares que lo especifican son SMPTE 2110 y SMPTE 2022 publicados en 2017.

Como se puede entender, la señal generada en los centros de los proveedores de contenidos tiene un ancho de banda tan elevado que para poder transportarla es necesario realizar una compresión, tema que se aborda en detalle en el apartado ***2.3. Compresión de vídeo y audio.***

- **Redes de contribución (ámbito de este TFG)**: Son las que engloban todo el envío y recepción de señales que servirán para conformar el producto que se entrega al cliente final.

  En uno de los puntos de estas redes se pueden encontrar, por ejemplo, delegaciones del centro principal, estudios de grabación externos, instalaciones de eventos deportivos, etc., …

  Estas señales son susceptibles de ser editadas, procesadas y/o regeneradas (codificación-decodificación) varias veces, por lo que es necesario que, aunque estén comprimidas, tengan una alta calidad. Se suele partir de señales con muestreo 4:2:2 con muestras de 8 o 10 bits. Dependiendo del tipo de compresión, para señales 1080i@50 se parte de un régimen binario de 20-50 Mbps.

- **Redes de distribución**: Comprende la entrega de la señal de televisión definida como producto hacia las cabeceras de las redes de difusión.

  Esta señal, a diferencia con la transportada por las redes de contribución, ya incluye todos los datos auxiliares que conforman información de teletexto, descripción de relación de aspecto, subtitulado, etc., …

Como no se va a realizar regeneración de la señal se reduce el perfil de muestreo a 4:2:0 que permite utilizar compresiones mayores manteniendo una calidad adecuada para la entrega final, con regímenes binarios de 15-20 Mbps.

- **Redes de difusión**: Comprenden el transporte de señal a los clientes finales.

  La señal alcanza el nivel de compresión más elevado. Como ejemplo, un canal de televisión HDTV que se recibe por TDT se recibe con un *bitrate* variable de unos 5 Mbps

Antes de entrar en más profundidad a definir posibles tipos de redes de contribución, se va a dar una visión de los tipos de compresión de vídeo en tiempo real más importantes que se utilizan en el contexto de la televisión y sus principales características.

2.3 Compresión de vídeo y audio.

La compresión en la señal de vídeo comúnmente se basa en el análisis de la información espacial y temporal de la imagen para eliminar la redundancia existente. Las técnicas que se aplican aprovechando la redundancia de información en píxeles próximos se denominan *intra-frame* y las que hacen lo mismo, pero entre imágenes consecutivas, se denominan *inter-frame*

Del mismo modo, para la señal de audio, se aprovecha las limitaciones del oído humano para reducir la información y no incluir la señal no audible.

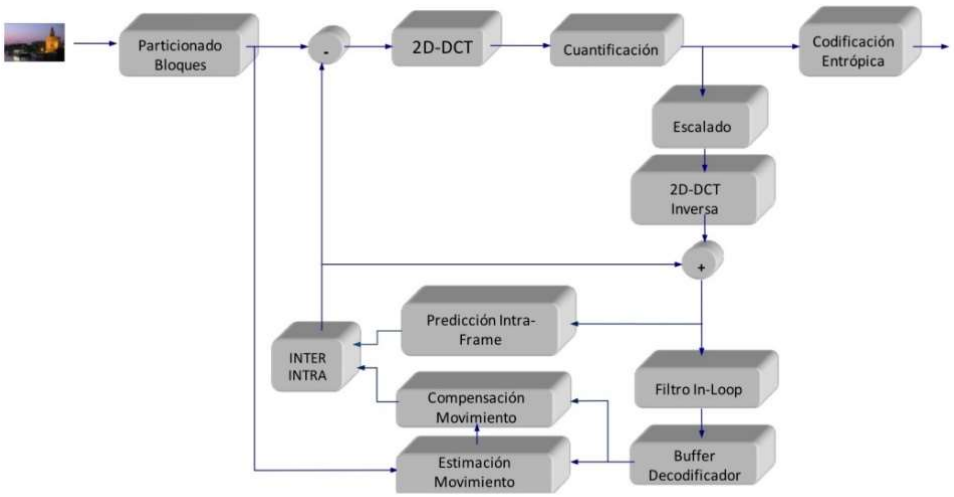

Figura 3. Diagrama de bloques del codificador MPEG.Fuente [2]

Desde la aparición a principios de los noventa, la codificación MPEG (*Motion Picture Experts Group*) en tiempo real mantiene en sus diferentes versiones el esquema que se representa en la **Figura 3.**, donde básicamente la señal original se divide en bloques que son procesados y codificados aprovechando las propiedades inter e intra de las imágenes.

2.3.1 MPEG-2.

El protocolo de compresión MPEG-2 surge en 1994 para mejorar las características de su antecesor (MPEG-1) que realizaba codificación *inter-frame* de dos imágenes, actual y anterior, utilizando por primera vez técnicas de predicción y compensación temporal (ME – *Motion Estimat*ion; MC – *Motion Compesation*).

Permite compresión de señales de vídeo progresivas y entrelazadas y se definen diferentes perfiles y niveles de compresión, permitiendo flexibilidad en su uso y en los regímenes binarios obtenidos. También estandariza la compresión de la señal de audio asociada.

En la codificación *inter-frame* se definen imágenes que, a partir de las técnicas de compresión se definen como:

- ***I- Frames*** *(Intra Frame)*: Son las imágenes que se obtienen de la señal original utilizando codificación intra-frame.

  No se conseguiría una ratio de compresión suficiente si la señal obtenida fuera una consecución de *I-Frames,* por lo que habrá incluir imágenes más comprimidas con técnicas de predicción de movimiento y diferencia.

- ***P-Frames*** *(Predictive Frames):* Se genera a partir de imágenes I o de la última imagen P ya obtenida (predicción hacia delante). Se parte de bloques de 16x16 píxeles y se realiza una estimación de movimiento con respecto a la imagen de referencia. De estos bloques se codifica la información de los píxeles que son diferentes, estimando el movimiento de la imagen en horizontal y/o vertical. Por lo tanto, su peso dependerá del movimiento en la secuencia de vídeo.

- ***B-Frames*** *(Bidirectional Predictive Frames)*: Las imágenes B se generan a partir de imágenes tipo P o I anteriores y P o I posteriores (predicción bidireccional). Son las imágenes con mayor compresión

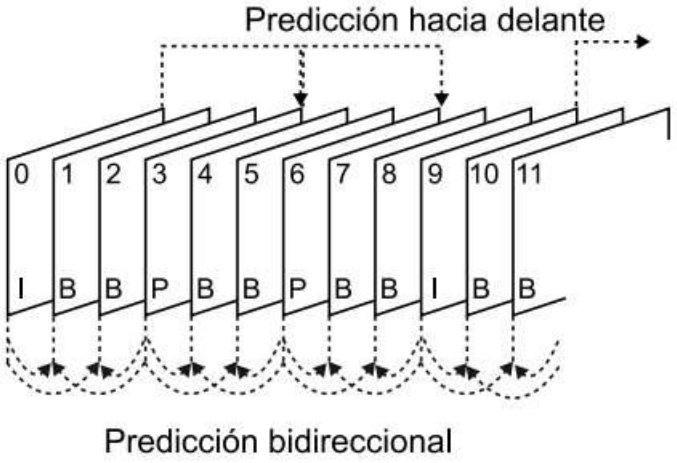

Figura 4. GOP del estándar de compresión MPEG. Fuente [3]

En la imagen de la **Figura 4**. se esquematiza el proceso de combinación de los tres tipos de imágenes formando lo que se denomina GOP (*Group Of Pictures*). Como se puede observar hay un compromiso entre compresión y calidad (mayor número de *B-*

frames mayor compresión y menor calidad – mayor número de imágenes I y P menor compresión y mayor calidad).

Hay que tener en cuenta que los errores de codificación o transmisión se encadenan debido a la relación explicada en la generación de las imágenes que conforman el GOP.

Las características de la señal comprimida final serán establecidas por los perfiles y niveles (especificado como *perfil@nivel*) de compresión.

Perfil @ nivel de resolución	Hz	Muestreo	Mbps	Ejemplo de aplicación
SP@LL	176×144	15 4:2:0	0.096	tablets, móviles
SP@ML	352×288 320×240	15 4:2:0 24	0.384	PDA
MP@LL	352×288	30 4:2:0	4	descodificadores
MP@ML	720×480 720×576	30 4:2:0 25	15 (DVD: 9.8)	DVD, TDT, televisión por cable y satélite
422P@ML	720×480 720×576	30 4:2:2 25	50	Sony IMX
422P@H-14	1440×1080 1280×720	30 4:2:2 60	80	reservado
422P@HL	1920×1080 1280×720	30 4:2:2 60	300	reservado

Tabla 2. Combinación de perfiles y niveles más usuales. Fuente [3]

En la **Tabla 2.** se esquematizan los perfiles@niveles más comunes en MPEG-2. Como se puede observar, MPEG-2, aunque comprende la señal HDTV entre sus posibilidades, se utiliza para señales de definición standard SDTV con resoluciones de cuadro para Europa de 720x576 píxeles.

En redes de contribución se parte de un perfil alto de codificación y se suele utilizar *422p@ML* (*422 Profile @ Main Level*) con regímenes binarios de 50 Mbps. En redes de distribución se suele utilizar *MP@ML* (*Main Profile @ Main Level*) con un ancho de banda de 15 Mbps.

Existen varios estándares de compresión de audio que pueden acompañar a la señal de vídeo en MPEG-2. En televisión profesional se suele utilizar MP2 (MPEG Layer 2). Se especifican diferentes frecuencias de muestreo (en audio profesional suele ser 48KHz) y dependiendo cual se seleccione se consiguen ratios de compresión de 32Kbps hasta 384Kbps (de nuevo compromiso entre calidad y compresión).

Las técnicas que se utilizan para codificar el audio tienen en cuenta las características del oído humano como:

- **Sensibilidad**: El oído es más sensible a los sonidos en el rango de frecuencia de la voz humana (300 Hz – 3400 Hz)

- **Enmascaramiento frecuencial**: Hay frecuencias que, si se reciben a niveles altos, hacen que sonidos próximos en frecuencia captados con menos nivel no sean apreciados.
- **Enmascaramiento temporal**: Sonidos que se reciben a un alto nivel hacen que, durante un intervalo de tiempo, otros sonidos de menor nivel no sean escuchados.

En la imagen de la **Figura 5.** se muestra el diagrama de bloques de esta codificación de audio y de forma general se observa que el espectro frecuencial de la señal se divide en bandas de frecuencia (hasta 32) y, atendiendo a los criterios definidos anteriormente, se utilizará una cuantificación (uso de bits) mayor o menor.

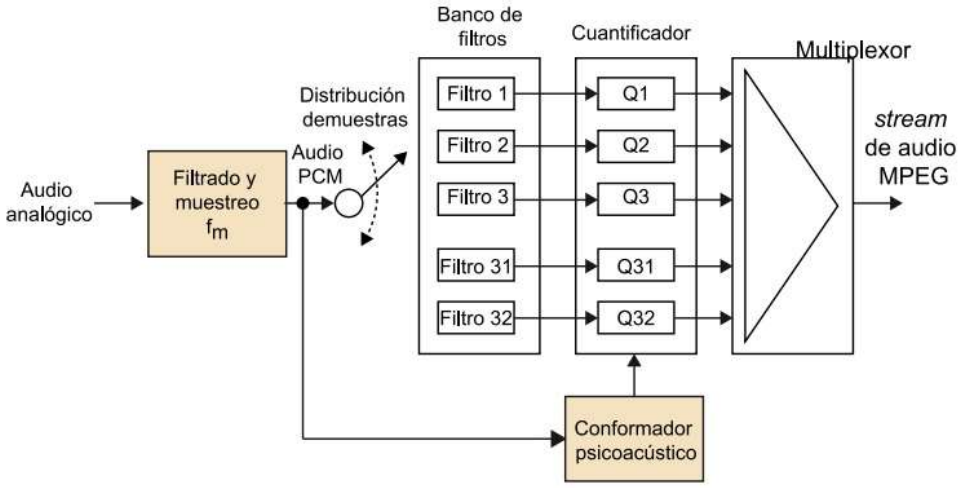

Figura 5. Diagrama de bloques del codificador MPEG. Fuente [4]

Otros estándares de compresión de audio que pueden acompañar a la señal de vídeo pueden ser MP3 (*MPEG Layer 3*), LPCM (*Linear Pulse Code Modulation*), AAC (*Advanced Audio Coding*) y Dolby Digital, entre otros.

2.3.2 MPEG-4 Part 10 (H.264/AVC).

Aparece una década posterior al MPEG-2 y mantiene la estructura básica de funcionamiento de su antecesor. Se convertirá en el estándar más usado para el almacenamiento y transporte de señales de televisión HDTV.

En MPEG-2, cuando se produce una conmutación entre flujos de vídeo diferentes, el descodificador presenta una pérdida total de calidad que se acaba solucionando tras un intervalo de tiempo. Para solucionar este problema, a las imágenes que forman el GOP (I, P y B), se añade en los perfiles más altos:

- SP-*Frames* (*Switching P-Frames*): Cuenta con macrobloques P o I
- SI-*Frames (Switching I-Frames)*: Cuenta con macrobloques I.

El tamaño de los bloques que se procesan cambia dependiendo de si la información intra-cuadro cuenta con pocos cambios (macrobloques de 16x16) o si cuentan con una mayor entropía (bloque de 8x8). De forma gráfica se muestra la diferente conformación de bloques y su aplicación sobre la imagen en la **Figura 6**.

Además, se añaden filtros *anti-blocking* en la codificación que suavizan el efecto de bloque debido a esta división en el procesado.

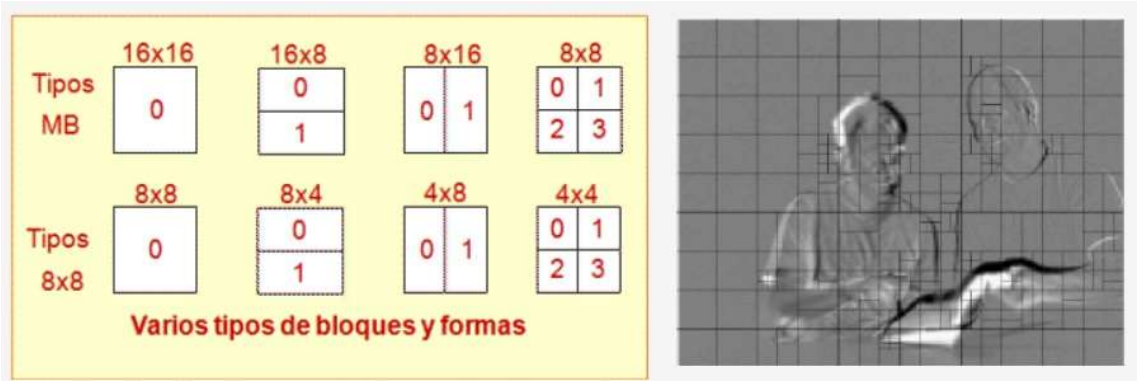

Figura 6. Tipos de bloques intra-cuadro y su utilización según la información de la imagen.

Al igual que MPEG-2 permite conseguir diferentes grados de calidad y compresión gracias a la definición de perfiles y niveles. En este caso se definen hasta siete perfiles de los que los más utilizados son:

- BP (*Baseline Profile*): Utilizado para aplicaciones donde el equipamiento no es profesional, tales como teléfonos móviles o reproductores portátiles.
- MP (*Main Profile*): Dirigido a transporte de señales HDTV en redes de difusión como TDT. Parte de señales con patrón de codificación 4:2:0
- HP (*High Profile*): Definido para entornos profesionales. Es el que se utilizará en redes de contribución a partir de señales de vídeo 4:2:2 o en centros de producción con señales 4:4:4

En este caso, los niveles se representan con numeraciones que van del 1.0 al 5.1 y especifican, de forma creciente, el régimen binario que se va a obtener. Con nivel 1.0 se hablaría de 64 kbps y en 5.1 de 240 Mbps).

Para señales HDTV, utilizando una combinación HP@4.0 se conseguirá un régimen binario máximo de 15-20 Mbps.

De nuevo, el flujo de audio asociado a la señal de vídeo podrá ir codificado en formatos como los que se especificaba para MPEG-2

En general, y como diferencia principal con su antecesor, permite conseguir los mismos criterios de calidad subjetiva utilizando mucho menos ancho de banda.

Se convierte en el formato de compresión precursor del transporte de vídeo por redes de datos IP.

2.3.3 HEVC (H.265).

Una nueva actualización del estándar MPEG en 2013 supone un nuevo protocolo de compresión que mantiene las características de codificación espaciotemporal de sus antecesores, pero mejora los procesos de estimación y compensación de movimiento.

Las dos mejoras más importantes que introducen son:

- Los bloques en los que se divide la imagen se definen con diferentes tamaños, se adaptan dinámicamente al contenido de la imagen.
- Existe un filtrado en el dominio espacial que minimiza los artefactos que pueden aparecer ante errores introducidos por una alta compresión. Mayor esfuerzo en aumentar la calidad subjetiva que la calidad objetiva por falta de información.

Gracias a estas características permite una alta precisión por píxel debido al tamaño dinámico de los bloques procesados, pudiendo procesar macrobloques de tamaño 32x32 píxeles. Se convierte en el estándar ideal para comprimir señales de ultra alta definición UHDTV (*Ultra High Definition*) 4K y 8K, consiguiendo unas ratios de compresión de 500:1 (señales 4K con regímenes binarios de 10-15 Mbps).

2.4 MPEG Transport Stream.

Una vez definida, de forma resumida, la evolución de los codificadores MPEG y cuáles son sus características más importantes, se puede observar que, realizada la codificación, se obtienen dos flujos de datos asociados a la señal de vídeo y audio respectivamente. Surge la necesidad de almacenar y/o transportar estos flujos o *streams* MPEG y para ello se define el *MPEG Transport Stream*.

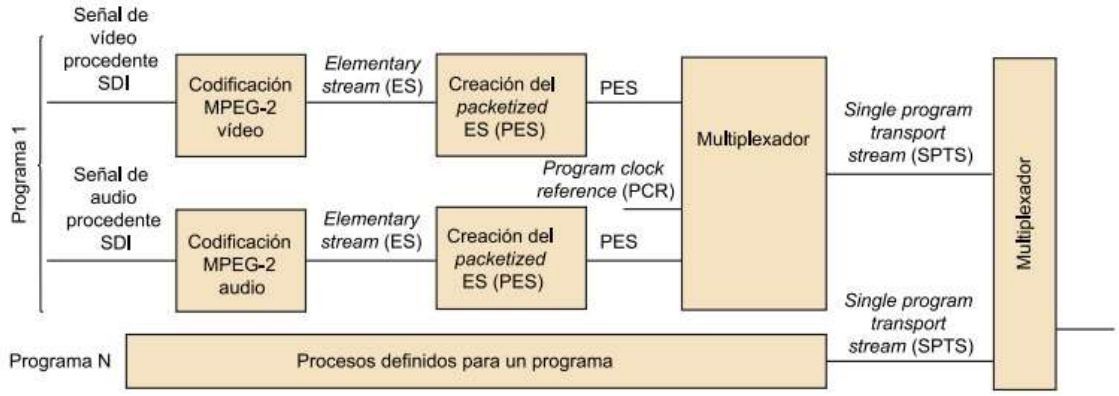

Figura 7. Diagrama de bloques de la conformación del MPEG *Transport Stream*

El transporte de señales MPEG se definió asociado a la versión MPEG-2, por lo que tambien se denomina MPEG-2 TS.

En el MPEG TS surgen los siguientes conceptos:

- **PES** (*Packetized Elementary Stream*): Consiste en la construcción de paquetes de datos a partir del flujo de datos MPEG elemental (**ES** – *Elementary Stream*) que se obtiene a la salida del codificador MPEG.
- **SPTS** (*Single Program Transport Stream*): Fruto de la multiplexación de los PES asociados a un programa, vídeo por un lado y audio por otro.

Los paquetes PES se conforman con una cabecera y un fragmento (**AU – Access Unit**) del ES de vídeo, audio o datos del programa correspondiente. La cabecera contiene la siguiente información:

- Identificador de ES.
- Referencias temporales para la sincronización de los ES en reproducción.
- Número de secuencia. Permite la detección de pérdidas de codificación o transmisión en el decodificador.

Con estos PES se conforma los paquetes del flujo de transporte TSP (*Transport Stream Packets*) tal y como se esquematiza en la imagen de la

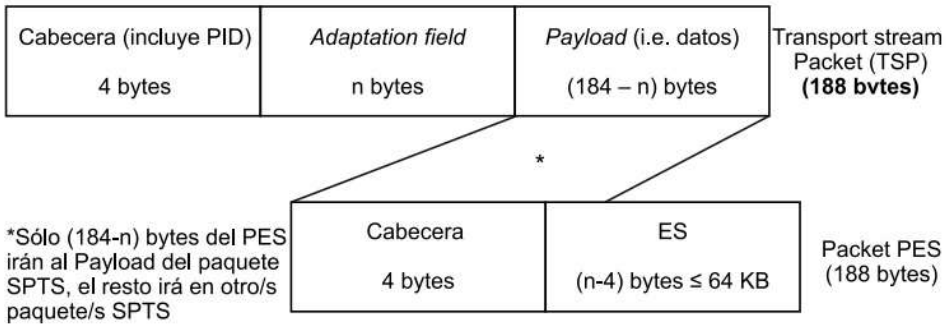

Figura 8. Estructura de los paquetes que conforman el MPEG - *Transport Stream*

El TS (*Transport Stream*) final lo podrá conformar un único programa y su SPTS asociado, o se podrán multiplexar N programas con sus SPTS asociados.

2.5 Redes IP.

En el diagrama de bloques de la **Figura 9.** se esquematiza los diferentes tipos y algunos ejemplos de las redes de comunicaciones.

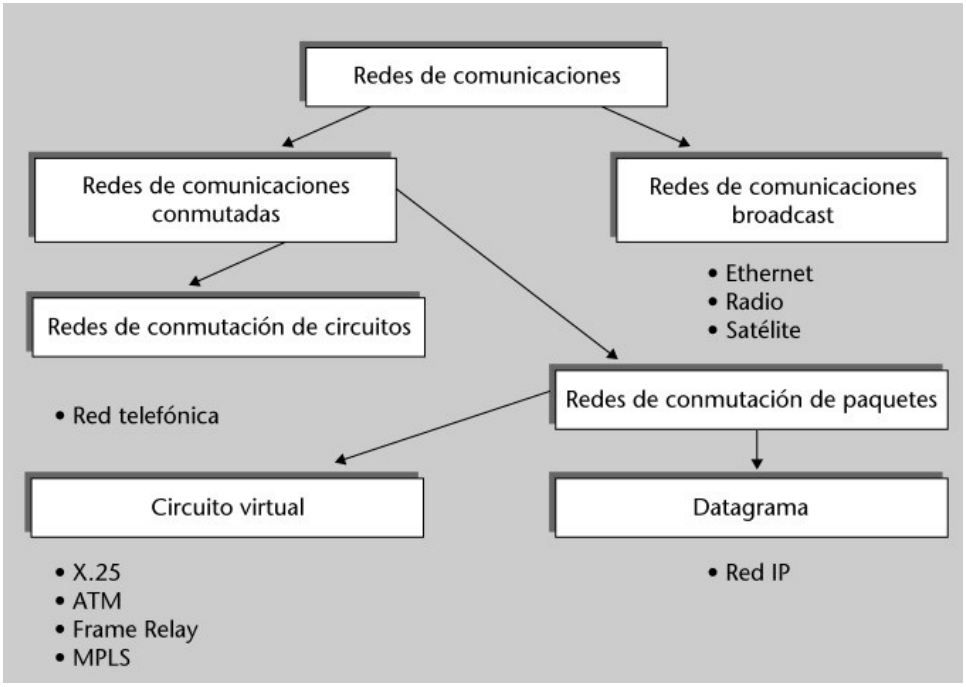

Figura 9. Clasificación de las redes de comunicaciones. Fuente: [6]

Como se ha comentado en el capítulo introductorio de este TFG, la televisión pasó de ser transportada únicamente por redes de difusión (*broadcast*) de radiofrecuencia como satélite o TDT, para converger a redes de datos o de conmutación de paquetes.

En el ámbito de las redes de contribución se van a describir dos de las redes de circuito virtual que se han utilizado (o utilizan actualmente) para contribución de vídeo, como son ATM y MPLS. Más extensamente de describen conceptos relacionados con las redes IP ya que suponen el contexto de este trabajo.

Se van a definir en este apartado los protocolos y servicios en los que se basan las redes IP, englobados en capas y en comparación con el modelo de referencia OSI.

De las redes IP, los niveles más interesantes para el desarrollo de este trabajo serán los comprendidos a nivel de red y a nivel de transporte. Tambien se analizan técnicas de difusión de contenidos en esta red y sus características.

2.5.1 Modelos de capas OSI y TCP/IP

El modelo OSI (*Open Systems Interconnection*) es un modelo de protocolos de red de referencia. Fue estandarizado por ISO (International Organization for Standarization) y creado en 1980. El modelo TCP/IP se definió con anterioridad, en la década de los setenta, para definir la red de gran alcance (WAN – *Wide Area Network*) ARPANET, precursora de Internet.

En la imagen de la **Figura 10.** se comparan ambos modelos.

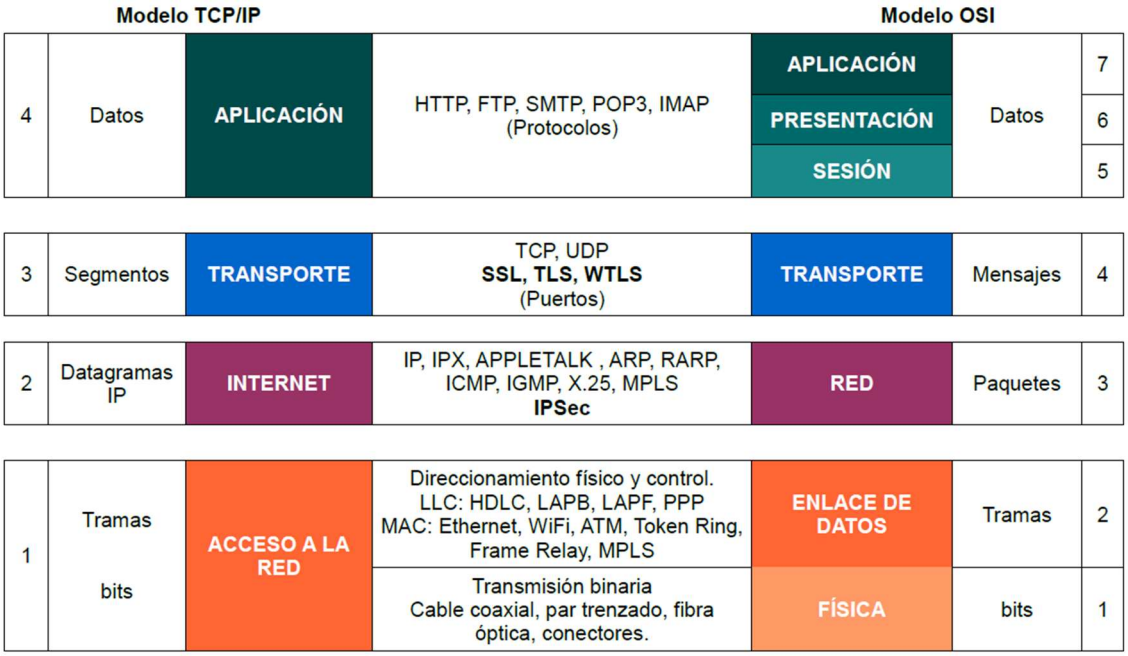

Figura 10. Comparativa entre los modelos OSI y TCP. Fuente [7]

Como se puede observar, en el modelo OSI se establecen siete capas o niveles, mientras que en TCP/IP tienen su correspondencia en sólo cuatro.

Cada capa recibe un servicio de la capa inferior y ofrece otro u otros servicios a la capa superior. Para ello se definen en TCP/IP una serie de reglas o protocolos.

- **Capa de acceso a la red**: Abarca los niveles 1 y 2 del modelo OSI. Características físicas (conectores, cableado, etc., …) que se utilizará para la conexión a la red. Dentro del nivel 2 de la capa OSI se establecen dos servicios como son el control de acceso al medio (MAC) y la capa de control lógico de enlace lógico (LLC). Establecen las características para iniciar y finalizar la conexión entre dos dispositivos en la red.
- **Capa de red o internet:** Tiene correspondencia con el nivel 3 del modelo OSI. Tiene como finalidad el empaquetamiento y direccionamiento de información dentro de la red.
- **Capa de transporte:** Se corresponde con el nivel 4 del modelo OSI. Sus protocolos definen las características de conexión a nivel de sesión.
- **Capa de aplicación**: Abarca los niveles 5,6 y 7 del modelo OSI. Los protocolos que se engloban en este nivel definen las características de las aplicaciones para intercambiar datos entre dispositivos. Algunas de las aplicaciones más utilizadas son HTTP, FTP, SMTP, etc., …

Del conjunto de niveles definidos en la suite de protocolos TCP/IP, se van a describir más ampliamente protocolos pertenecientes a las capas de red (IP) y de transporte (TCP y UDP), ya que definen el direccionamiento de paquetes dentro de la red, así como los mecanismos que permiten conseguir que éstos lleguen correctamente.

2.5.2 Protocolo IP (*Internet Protocol*)

Este protocolo permite la comunicación en redes de conmutación de paquetes como es Internet (enrutamiento). Es un protocolo no orientado a conexión, es decir, no se define un camino estable entre origen y destino, es más, ni siquiera se puede asegurar que el destinatario esté disponible antes del envío.

Para el uso de este protocolo se pueden utilizar dos versiones distintas que se definen a continuación

2.5.2.1 IPv4.

La versión 4 del protocolo IP o IPv4 se define en la RFC. 791 de la IETF de 1981. Los paquetes que se intercambian los dispositivos en este nivel se denominan datagramas IP, que, para esta versión, su cabecera es la que se muestra en la imagen de la **Figura 11.**

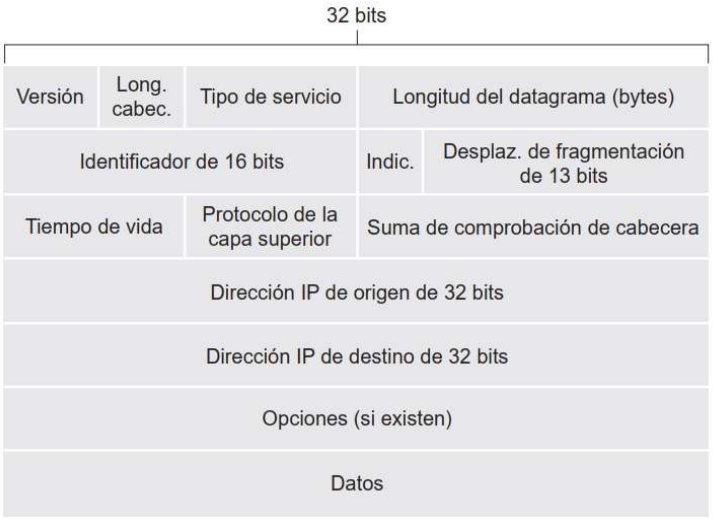

Figura 11. Datagrama IPv4. Fuente [8]

Algunos de sus campos más importantes:

- **Tiempo de Vida (TTL)**: Campo de 8 bits que define el *tiempo de vida* del datagrama en la red. Si el destino es inaccesible, para evitar que esté circulando indefinidamente, cada encaminador por el que pasa lo decrementa en una unidad. Al llegar a 0 el *router* en cuestión lo elimina.
- **Protocolo de la capa superior:** Campo de 8 bits que indica el protocolo de la capa de transporte. Según su valor indicará si es TCP, UDP, etc., …
- **Suma de comprobación de cabecera (*Head Checksum*):** Campo de 16 bits que permite a los *routers* detectar si hay algún error en el datagrama recibido. En el envío se suman los valores de la cabecera IP en parejas de 2 bytes utilizando aritmética complemento a 1. El resultado se almacena en este campo. El router que recibe el datagrama hace esta misma operación y compara el valor obtenido con el de este campo. Normalmente, el router que detecta una diferencia, detecta el error y descarta el datagrama.

- **Direcciones de fuente y destino:** En estos campos de 32 bits se almacenan las direcciones IP de los equipos fuente y destino respectivamente. Sirven, por tanto, para direccionar los paquetes.

  Una dirección IP se expresa con una notación decimal con puntos donde el valor de cada byte se expresa en decimal separado por un punto.

  Por ejemplo: 192.168.1.1 es la dirección IP equivalente a los 32 bits:

  11000000 10101000 00000001 00000001.

- **Datos:** Acompañando a esta cabecera irán los datos procedentes de la capa superior, la capa de transporte.

2.5.2.2 IPv6.

La versión 6 del protocolo IP surge ante la necesidad de un sucesor para la versión 4, ya que con el crecimiento de Internet se observó que se convertía en un recurso limitado. Se define la última versión del protocolo IPv6 en la RFC. 8200 de la IETF de 2017. El datagrama de esta versión contará con los campos de la **Figura 12.**:

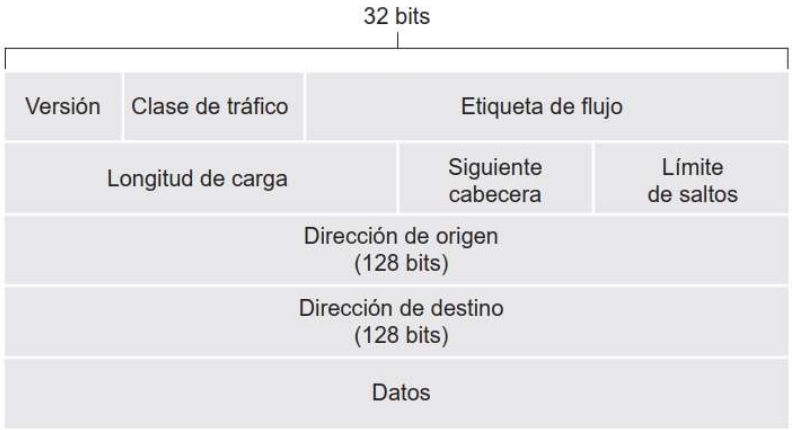

Figura 12. Datagrama IPv6. Fuente [8]

Algunos de sus campos más importantes son:

- **Siguiente cabecera:** Indica cual es la siguiente cabecera del siguiente protocolo que se encuentra incluido en los datos. Similar al campo *protocolo de la capa superior* de IPv4
- **Límite de saltos:** Su valor se va decrementando cada vez que pasa por un router hasta llegar a cero, momento que se retira de la circulación. Parecido a *TTL* de IPv4.
- **Direcciones de origen y destino:** La gran diferencia con IPv4, ya que se utilizan 128 bits para las direcciones IP.

  En este caso la dirección IPv6 se expresan 16 bits en formato hexadecimal entre dos puntos. Además, si hay ceros consecutivos, éstos se pueden simplificar.

  Por ejemplo: La dirección 1080:0:0:0:4:300:100c:44aa es equivalente a 1080::4:300:100c:44aa

El cambio de una versión a otro debe ser adaptativo, ya que hay muchos dispositivos que no están preparados para recibir datagramas IPv6, por lo que se utilizan técnicas en las que redes IPv6 se comunican a través de redes IPv4 por tunelación, tal y como se recoge en la RFC. 4213. Esta técnica lo que hace es que a los datagramas IPv6 se les añaden las cabeceras IPv4 necesarias para que puedan viajar y ser enrutados en redes IPv4. Los datagramas IPv6 viajan como datos añadidos a dichas cabeceras.

En este TFG se ha decidido utilizar la versión 4 del protocolo por lo que, en adelante, siempre que se haga referencia a este protocolo se hará a la versión IPv4.

2.5.3 Protocolos a nivel de transporte

La capa de transporte permite una comunicación lógica entre aplicaciones de dispositivos diferentes, mientras que la capa de red crea dicha comunicación lógica entre dispositivos.

Los segmentos, que es el nombre que reciben los paquetes de la capa de transporte, incluyen dos campos muy relevantes: los puertos de origen y destino. Estos campos identifican de manera unívoca a los sockets de las aplicaciones. Cada puerto es un número de 16 bits (0-65535). El rango del 0 a 1023 contienen valores reservados para aplicaciones conocidas (como el puerto 80 para HTTP).

Los protocolos de transporte más importantes que se utilizan en redes públicas como Internet se detallan a continuación.

2.5.3.1 TCP (*Transmission Control Protocol*)

Este protocolo ofrece un servicio a la capa superior (de aplicación) orientado a la conexión fiable. Para ello utiliza ciertas técnicas de control asegurando la llegada de todos los paquetes, correctamente y en orden a destino. Realiza una regulación en la velocidad de los datos procurando adaptarla al ancho de banda disponible en los nodos que atraviesa.

Se utiliza para *streaming* en redes públicas donde no se pueden implementar técnicas de calidad de servicio ya que, ante pérdidas esporádicas de paquetes repercuten en la calidad subjetiva de la señal de una manera poco significativa. Como inconveniente, cuando hay pérdida o errores en los paquetes se solicita a origen que los vuelva a enviar. Esto supone un aumento en al ancho de banda y en el retardo.

2.5.3.2 UDP (*User Data Protocol*)

Este protocolo ofrece un servicio a la capa superior que no está orientado a la conexión y no ofrece fiabilidad en la entrega. No presenta mecanismos de control por lo que ante pérdidas o errores en los paquetes no se solicita su reenvío. Tampoco se puede asegurar que lleguen en orden.

Sin embargo, debido a estas características, es un protocolo ligero al no contar con los métodos de control. Es idóneo para aplicaciones en tiempo real ya que, al no garantizar la conexión, no hay ese tiempo de retardo en iniciar la transmisión.

Estas características hacen a UDP el protocolo idóneo para *streaming* de audio/vídeo en tiempo real siempre que se pueda asegurar calidad en el servicio.

2.5.4 Modos de multidifusión en la red.

Dentro de las redes IP existen diferentes técnicas para difundir contenido multimedia. Se enumeran y describen a continuación.

2.5.4.1 *Unicast.*

En este tipo de difusión la comunicación es punto a punto. Cada usuario recibe su propio contenido, aumentando el ancho de banda en el servidor. Por este motivo permitirá un número máximo de clientes. En la imagen de la **Figura 13.** se representa esta situación.

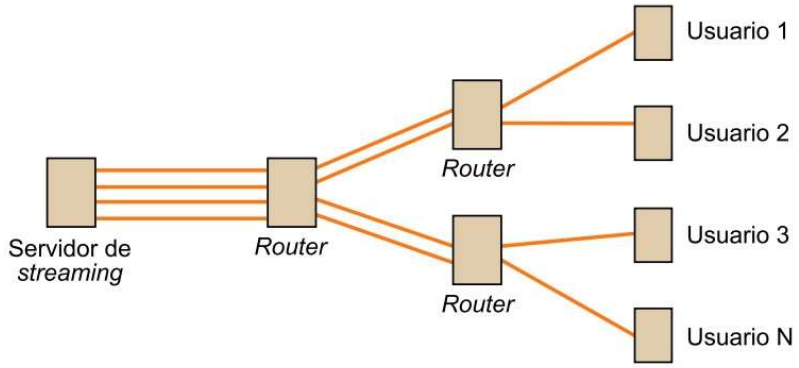

Figura 13. Representación de una sesión de *streaming unicast.* Fuente: [9]

2.5.4.2 *Multicast*

Este es el tipo de difusión más usual entre los proveedores de IPTV ya que permite llegar a un gran número de usuarios con un dimensionamiento moderado de la red. El servidor de contenidos genera un único flujo y serán los *routers* que componen la red, configurados para ello, los que duplicarán el envío hacia los clientes que lo soliciten. En la imagen de la **Figura 14.** se esquematiza este modelo.

2.5.4.3 *Broadcast*

En este tipo de difusión se envía el contenido simultáneamente a todos los nodos de la red, no hay petición previa por parte de los clientes. Su ámbito está en el de las redes de área local LAN (*Local Área Network*) en un contexto muy controlado, ya que la inyección de este tipo de tráfico podría suponer una ralentización generalizada en la red.

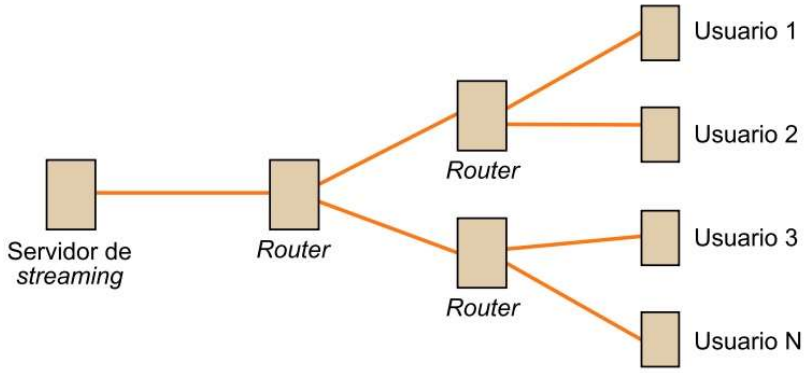

Figura 14. Esquema de difusión *multicast*. Fuente: [9]

2.5.5 Elementos de la red IP

Como se ha detallado anteriormente, las redes IP son redes de conmutación de paquetes. Por lo que, a parte de los equipos finales como servidores y hosts, básicamente están conformadas por conmutadores (*switches*) y enrutadores (*routers*), tal y como se representa en la imagen de la **Figura 15**. Se describen sus funciones básicas a continuación.

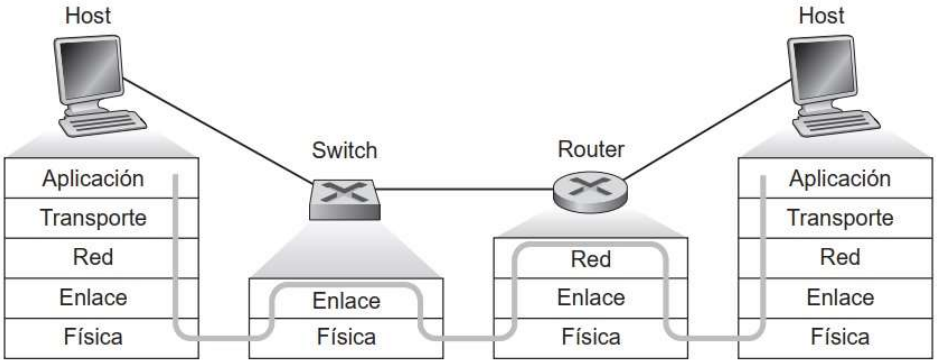

Figura 15. Elementos de la red IP. Fuente: [8]

2.5.5.1 Encaminadores (*Routers*)

Un encaminador o *router* trabaja en el nivel 3, en la capa de red, y realiza funciones de reenvío de paquetes y de conmutación. En la imagen de la **Figura 16.** se esquematiza el funcionamiento de un *router*.

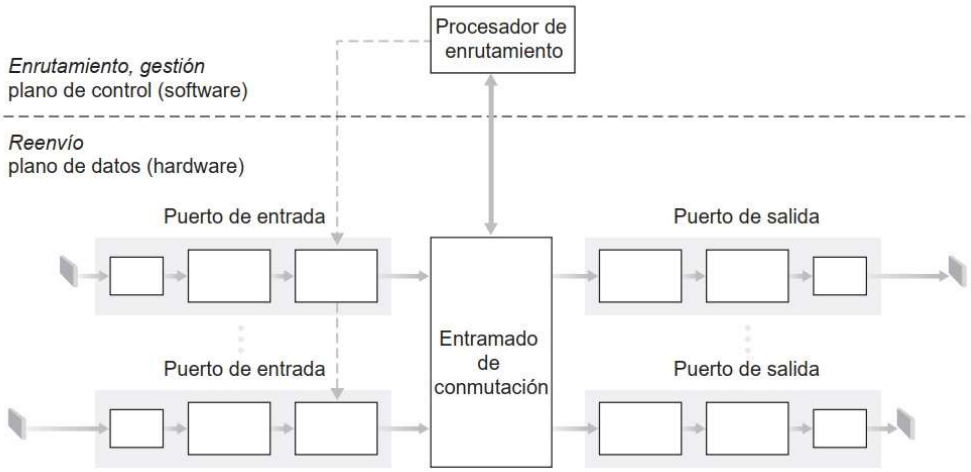

Figura 16. Arquitectura de un *router*. Fuente: [8]

- El reenvío se define como la transferencia de datos desde un enlace de entrada al enlace(s) de salida correspondientes. Físicamente, en el *router*, los enlaces de comunicación se denominan puertos de entrada y salida y serán identificados con una dirección IP. (No tienen nada que ver con la definición de puertos de la capa de transporte). Aquí se define un pipeline de proceso con una velocidad de procesamiento muy elevada. En la imagen de la **Figura** 17**.** se representa el proceso de reenvío llevado a cabo en el puerto de entrada.

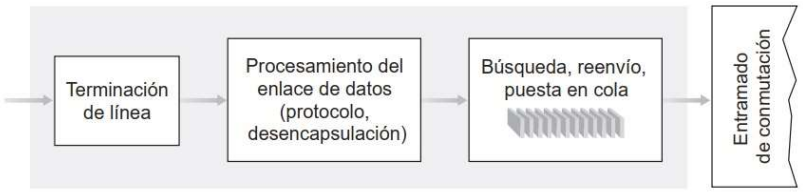

Figura 17. Procesado en el puerto de entrada. Fuente: [8]

- El entramado de conmutación a nivel hardware conecta los puertos de entrada y salida. Las conmutaciones físicas suponen operaciones en el orden de los nanosegundos
- El procesador de enrutamiento realiza las operaciones en el plano de control dentro de la capa de red. La gestión se hace a nivel software. Aquí es donde se ubican las tablas de enrutamiento y la información del estado de los enlaces de comunicación. Estás operaciones son algo más lentas, en el orden de los milisegundos.

En la imagen de la **Figura 18.** se muestra un ejemplo de direccionamiento IP que se realiza en un *router.*

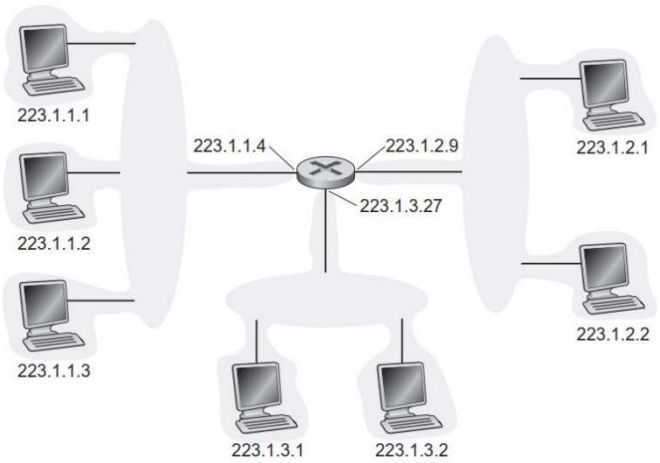

Figura 18. Conexión de subredes a través de un *router*. Fuente: [8]

En este caso existen tres subredes conectadas a los tres interfaces del *router* que tendrán una dirección IP coherente con la red a la que están conectados. La subred de la izquierda, por ejemplo, se podrá identificar como 223.1.1.0/24, donde /24 indica el número de bits, de los 32 posible, que se utilizan para identificar la subred (se denomina máscara de subred).

Los *routers* conectados por alguna de sus interfaces a Internet tendrán que contar con una dirección global o pública que no puede estar duplicada y que suele ser gestionada y facilitada por los ISP (*Internet Service Providers*). El resto de las interfaces del *router* tendrán un ámbito privado.

La técnica que realizan los *routers* para comunicar diferentes equipos de la red local o privada con diferentes equipos en Internet es lo que se denomina traducción de direcciones de red o NAT (*Network Address Translation*). Se ejemplariza el funcionamiento de un *router* NAT en la **Figura 19.**

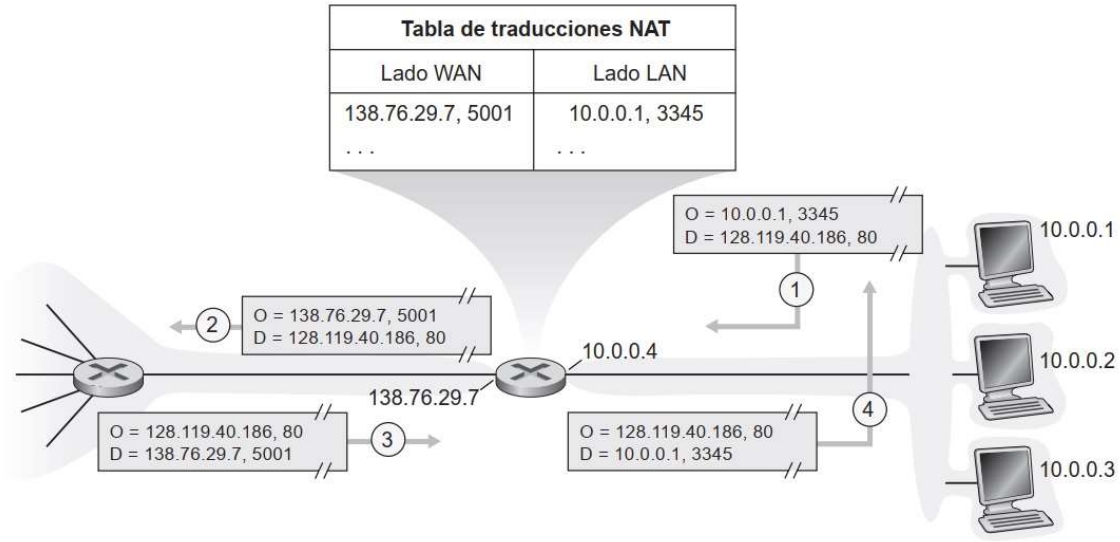

Figura 19. Ejemplo de comunicación por NAT. Fuente: [8]

Como se puede observar, en la parte privada se encuentra la subred 10.0.0.0/24 y en la parte pública el interfaz del *router* conectado tiene la dirección 138.76.29.7. El equipo conectado a la red local con dirección 10.0.0.1 se conecta

con el servidor 128.119.40.186 no con su IP sino con la IP pública del *router* (138.76.29.7) realizando el proceso de NAT.

2.5.5.2 Conmutadores (*Switches*)

Como se observa en la imagen de la **Figura 15.** los *routers* procesan los datagramas hasta en nivel 3, capa de datos, mientras que los *switches* lo hacen hasta la capa 2, de enlace. Los switches realizan conmutación y reenvío como los *routers* pero éstos se basan en las direcciones MAC (*Media Access Control*) de los equipos. Éstas tienen 6 bytes de longitud que se suelen representar en valor hexadecimal separados por dos puntos (cada byte lo representan dos números hexadecimales). Este valor será inherente al interfaz de red del *host* conectado a la red y su valor será único.

El switch realiza dos operaciones básicas:

- Filtrado: Detecta si alguna trama tiene errores y la descarta.
- Reenvío: Determina por qué interfaces se debe enviar la trama y la envía.

Estas dos funciones las realiza el *switch* siguiendo la tabla de conmutación donde asocia interfaces con direcciones MAC. Además, esta tabla se crea automáticamente. Cada vez que el *switch* recibe trama por una de sus interfaces, extrae la dirección MAC del campo de dirección MAC de origen y la registra asociada a dicho interfaz. Además, si cambia la conexión en la trama o no se conecta nada en un tiempo, la tabla es capaz de adaptar sus valores a esta nueva situación.

Otra propiedad interesante de los *switches* es que son capaces de aislar el tráfico entre sus enlaces, definiendo lo que se denomina una red local virtual VLAN (*Virtual Local Area Network*). En la imagen de la **Figura 20.** se ejemplariza esta situación, donde se define una VLAN en los interfaces del 2 al 8 y otra VLAN con los interfaces del 9 al 15. Estas redes, aunque conectadas al mismo *switch*, trabajarán totalmente aisladas una de la otra.

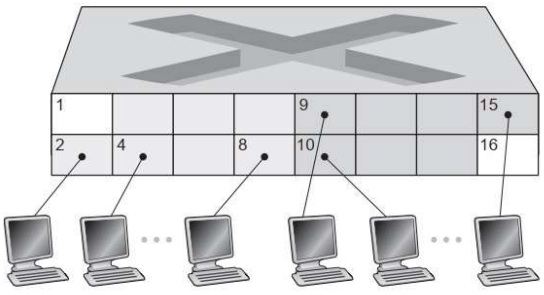

Figura 20. Definición de VLANs en un *switch*. Fuente: [8]

2.6 *Streaming* de audio/vídeo.

El *streaming* consiste en la entrega y transmisión de audio y vídeo a través de redes IP. Para este TFG el tipo que se va a analizar es el que se realiza en tiempo real (*live streaming*), es decir, el contenido se está produciendo simultáneamente a su transmisión.

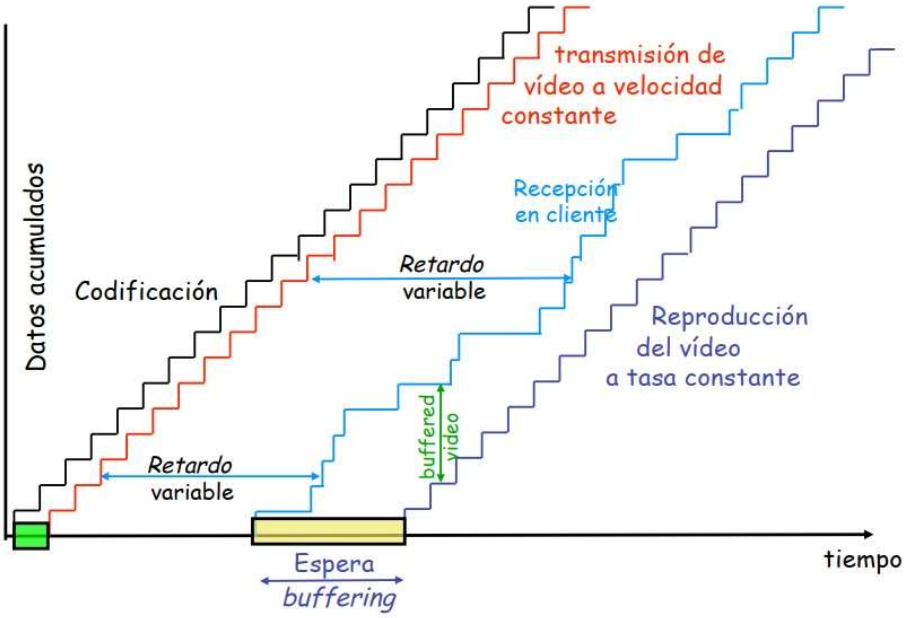

Figura 21. Proceso de *live streaming* respecto al tiempo.
Fuente:https://www.tlm.unavarra.es/~daniel/docencia/doctorado/2008-09/slides/Streaming.pdf

En la imagen de la **Figura 21.** se representa el proceso de *live streaming* desde que se genera, tras un proceso de codificación de la señal en directo, hasta que es reproducida por el cliente. Como se puede observar, la red introduce un retardo variable que es compensado con un proceso de *buffering* en la recepción. Se acumulan datos durante este tiempo de *buffering* y, una vez transcurrido, empieza la reproducción a *bitrate* constante. Siempre que el retardo acumulado no supere el del *buffer,* la reproducción será satisfactoria.

A continuación, se describen dos de los protocolos más usado para este tipo de *streaming*.

2.6.1 RTSP (*Real Time Streaming Protocol*).

Es un protocolo cuya primera versión se recoge en la RFC 2326 de 1998 mientras que su segunda versión es de 2016 recogida en la RFC 7826.

Es un protocolo que lleva bastante tiempo utilizándose en Internet para *streaming*, pero a pesar de que aparecen nuevos protocolos, sigue vigente.

Se basa en las siguientes características para eventos en vivo:

- Es un protocolo de comunicación bidireccional en el que se separa el canal de control de conexión del canal de transmisión del *stream*, tal y como se muestra en la imagen de la **Figura 22.**

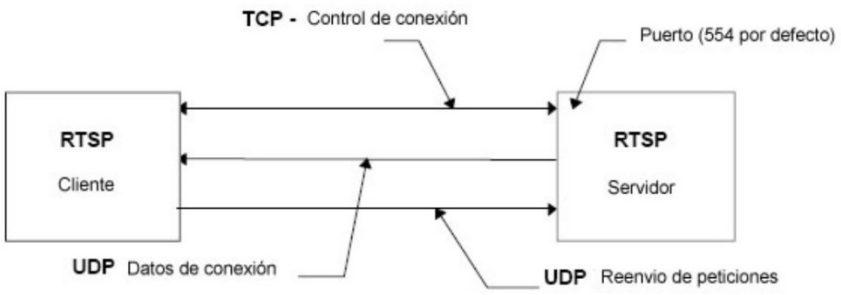

Figura 22. Comunicación por protocolo RTSP. Fuente [11]

- Es un protocolo seguro gracias a que utiliza los protocolos de transporte seguro como TLS (*Transport Layer Security*)
- Permite multidifusión tanto *unicast* como *multicast.*
- Es independiente del protocolo de transporte pudiendo utilizar UDP o TCP, pero para transmisiones en vivo utiliza el protocolo de transporte UDP.
- Para establecer un canal de comunicación fiable, utiliza mecanismo de control que, ante pérdida o errores de paquetes, se solicite reenvío al servidor. Las tramas incluyen una marca de tiempo (*timestamp*) para evitar ambigüedades en los reenvíos.

2.6.2 SRT (*Secure Realible Transport*).

Este protocolo para transmisión de vídeo en tiempo real se definió para ser usado en redes públicas como Internet. En un primer momento fue desarrollado por la compañía Haivision en 2013. En 2017 se convierte en un protocolo de código abierto, lo que ha permitido crear la SRT Alliance, consorcio donde compañías de diferente ámbito contribuyen ofreciéndolo en su equipamiento y colaboran en su desarrollo. Pretende así convertirse en el estándar de facto de la industria audiovisual.

Sus características más importantes:

- SRT se basa en el protocolo de transporte UDT (*UDP-based Data Transfer*) que permite envío rápido de archivos a través de redes públicas pero que no está pensado para vídeo. SRT introduce ciertas modificaciones sobre este protocolo para que permita que, en un *streaming* de vídeo en tiempo real, haya un mecanismo de recuperación de paquetes. De esta forma se consigue una comunicación confiable.
- Se definen tres perfiles en las comunicaciones SRT:
    - *Caller* o Iniciador: Dispositivo que inicia la conexión SRT. Este dispositivo debe conocer la IP pública y puerto del dispositivo receptor.
    - *Listener* o Receptor: Dispositivo que espera la petición para iniciar la conexión
    - *Rendezvous* o Encuentro: En este caso tanto transmisor como receptor deben estar en este modo. Los dispositivos negocian la conexión SRT en un puerto acordado.
- Se utilizan buffers para controlar la latencia del orden de los milisegundos.
- Al igual que RTSP, utiliza marcas temporales en los paquetes.
- Establece un canal seguro mediante encriptación AES de 256 bits.

El proceso de comunicación se resume en el diagrama de comunicación entre transmisor y receptor de la **Figura 23.**

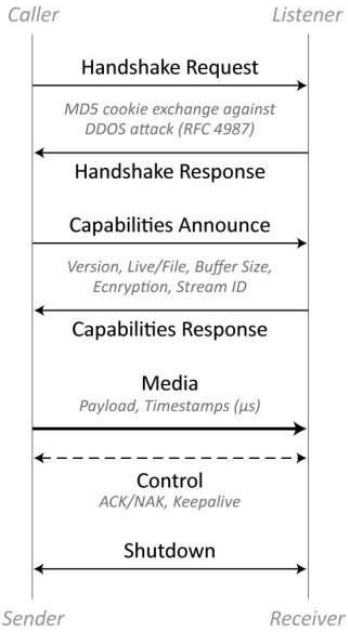

Figura 23. Diagrama de comunicación en el protocolo SRT. Fuente [12]

2.7 Redes de datos utilizadas para contribución de vídeo

En este apartado se van a enumerar y definir algunas de las redes de datos que se han ido utilizando en el entorno profesional de televisión para la distribución de audio/vídeo.

2.7.1 Redes ATM

El protocolo ATM (*Asynchronous Transfer Mode*) se desarrolla para comunicación en redes de distribución de servicios integrados de banda ancha (B-ISDN) y sus características se definen en la recomendación I.150 de la ITU-T de 1999.

Lo que se pretende es transmitir el MPEG Transport Stream generado (visto en el apartado ***2.4 MPEG Transport Stream***) a partir de la compresión de audio y vídeo, por las redes ATM, aprovechando que permitían un canal de transmisión adecuado.

Las redes ATM son redes de gran alcance WAN que conjugan la conmutación de celdas (*cells*) con la definición de circuitos virtuales (redes orientadas a la conexión). Estas características hacen que el *ATM Forum* defina como transmitir video comprimido en MPEG-2 sobre ATM:

- Las celdas llegan a destino con retardos variables por lo que, para paliar su efecto en la recepción, se incluyen buffers que garanticen la reproducción continua.
- Cuando hay errores en el *stream*, ATM no permite la retransmisión de datos, por lo que para detectar y corregir errores se utilizan técnicas en la trama de

vídeo como FEC (*Fordward Error Correction*) y CRC (*Cyclic Redundancy Check*).

- Al ser orientado a conexión permite tanto difusión *unicast* como *multicast.*
- En la definición de capas para las redes ATM, existe una denominada capa de adaptación a ATM (AAL – *Adaptation ATM Layer*). Se describen diferentes niveles de adaptación y se decide adoptar AAL5. Con este nivel de adaptación se consigue gran eficiencia para transportar hasta dos flujos MPEG con *bitrate* constante de 5Mbps con una tasa de errores BER (*Bit Error Rate*) $\leq 10^{-10}$

Este tipo de redes están en desuso debido a que son difícilmente escalables y a que presentan problemas de compatibilidad con las redes IP.

2.7.2 Redes MPLS

Las redes de conmutación de etiquetas multiprotocolo MPLS (*Multi-Protocol Label Switching*) definen circuitos y caminos virtuales.

Las cabeceras MPLS se encuentran entre las de enlace y red y, como se muestra en la imagen de la **Figura 24**. cuentan con un campo denominado *etiqueta*.

Figura 24. Cabecera MPLS. Fuente []

Los *routers* de estas redes no necesitan extraer la dirección IP para realizar la conmutación, ésta se hace a partir del valor de la etiqueta. Como ventajas, este tipo de conmutación es más rápida que la de los datagramas IP y los enrutamientos se realizar por rutas que con los protocolos de enrutamiento IP no serían posibles.

MPLS, además, se utiliza para crear redes privadas virtuales VPN (*Virtual Private Networks*), y como definen redes de gran alcance se les denomina *MacroLAN*.

Aunque a un elevado precio, las características de estas redes permiten que sean utilizadas como redes de contribución para televisión:

- Se garantiza una banda ancha constante entre puntos finales
- Al establecerse circuitos virtuales, permite calidad de servicio y se pueden definir transmisión *unicast* y *multicast*.

Ésta supone una solución muy adecuada para crear redes de contribución con delegaciones del centro de producción principal. Al ser un servicio de banda ancha, además de los circuitos para televisión, se puede dotar de conexión a Internet y administrar las redes de las delegaciones dentro de la red local.

2.7.3 Contribución a través de Redes IP (Internet). (Ámbito de la implementación)

Hasta no hace mucho tiempo se pensaba que era imposible el uso de redes a través de Internet para contribución de vídeo. Internet es una red heterogénea, de extensión amplia, que implica los siguientes inconvenientes para el transporte de vídeo de alta calidad.

- ***Jitter***: Consiste en una fluctuación en el retardo que se produce en la recepción de los paquetes transmitidos. Éstos pueden ser debidos a diferentes causas como cambio en el enrutamiento, congestión en el tráfico, etc., …
- **Retardo punto a punto**: Tiempo que tarda el flujo de paquetes en llegar desde origen a destino.
- Internet no es una red orientada a la conexión. No permite realizar sesiones ni *broadcast* ni *multicast.*
- Al no ser una red orientada a la conexión es complicado garantizar la calidad del servicio.

Sin embargo, es necesario cumplir unos requerimientos de calidad para las redes de contribución como los que se indican en la **Tabla 3.**

Tasa de video streaming (Mbps)	Latencia	Jitter	Periodo de pérdida	Intervalo de pérdida	Promedio de paquetes perdidos
15	<200 ms	<50 ms	1 paquete	1 error c/4 horas	9,14e-8
17	<200 ms	<50 ms	1 paquete	1 error c/4 horas	7,31e-8
18.1	<200 ms	<50 ms	1 paquete	1 error c/4 horas	6,09e-8

Tabla 2. *Requerimientos mínimos para brindar HDTV con QoE MPEG-4. Elaborado por:*

Santiago Mina G. Fuente (FORUM, 2006)

Tabla 3. Requerimientos de calidad para una red de contribución.

3. Implementación.

3.1 Escenario 1. Topología de Red.

La primera topología de red definida es la que se muestra en la imagen de la **Figura 25.**

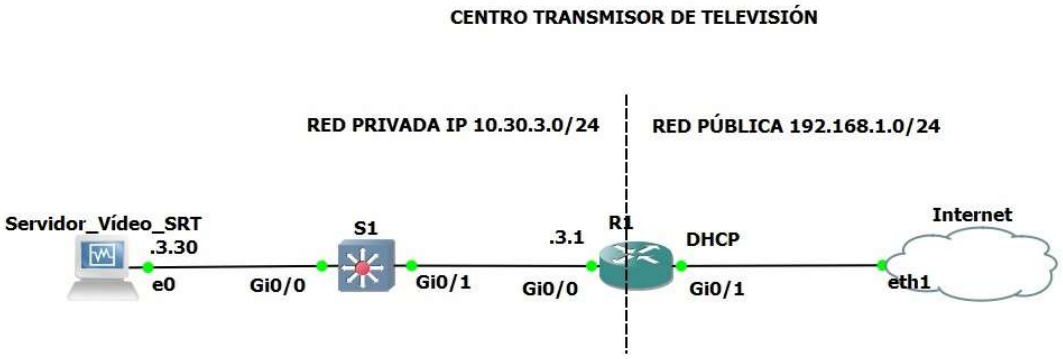

Figura 25. Modelo inicial para centro de producción de televisión transmisor

En la **Tabla 4.** se define el direccionamiento IP de los equipos que conforman la red.

Equipo	Interfaz	Dirección IP	Máscara de subred	Gateway
R1	Gi0/1	DHCP		N/A
	Gi0/0	10.30.3.1	255.255.255.0	N/A
Servidor_Vídeo_SRT	Ethernet	10.30.3.30	255.255.255.0	10.30.3.1

Tabla 4. Direccionamiento IP del equipamiento de red

Se ha implementado el equipamiento con que contaría el centro transmisor. El codificador de vídeo se conectaría a un *switch* (**S1**) de capa 2, y éste a un *router* (**R1**).

Configurando el dispositivo ***Internet,*** tal y como se especifica en *3.1.4 Conexión a Internet.*, se permite desde la máquina anfitrión reproducir el *streaming,* verificando que la configuración de red es la adecuada y haciendo de centro receptor.

3.1.1 Servidor de Vídeo. (SRT-Server)

Para la codificación de vídeo y generación de *streaming* en tiempo real existe una serie de equipos en el mercado denominados DCM (*Digital Content Manager*) que reciben la señal directamente en HD-SDI por cable coaxial y se pueden configurar para generar flujos de red con diferentes protocolos. Un ejemplo de este tipo de equipos es el que ofrece Haivision, el *Makito Encoder*, dónde se puede ver su trasera en la imagen de la **Figura 26.**

Figura 26. Codificador y transmisor de vídeo de Haivision. Fuente: https://www.haivision.com/products/video-encoder-solutions/

Permite hasta cuatro entradas de vídeo SDI y su conexión a la red se puede hacer por cable con pares trenzados o fibra (SFP+).

De forma complementaria, se tendrá que utilizar un equipo decodificador de características parecidas en el otro punto de la comunicación.

Para este TFG no se cuenta con un equipo de estas características, por lo que se va a simular de forma virtual. Para ello, se va a utilizar una imagen de un ordenador con sistema operativo Ubuntu 20.04 LTS. Se ha dotado de una RAM de 2048 MB y un procesador de 2 núcleos.

```
#!/bin/bash

gnome-terminal -e "/home/usuario/pfg/srt/./srt-live-transmit srt://10000 'srt://:5200?mode=listener&passphrase=123456789101112'"
sleep 1
gnome-terminal -e "/home/usuario/pfg/ffmpeg/./ffmpeg -re -i /home/usuario/Vídeos/CRHD25.mp4 -f mpegts 'srt://127.0.0.1:10000'"

exit
```

Figura 27. Script para simulación de codificador de vídeo: *ffmpeg* + *srt-live-transmit*

Se van a utilizar dos softwares diferentes para realizar lo propuesto:

- El codificador de vídeo ***FFMPEG***: En la imagen de la **Figura 27.** se muestra cómo se toma un archivo de entrada y con unos parámetros se crea un *stream* que se dirige a un socket en el puerto 10000:
    - ***-re*:** Con este modificador se indica que la reproducción del archivo sea a reproducción normal (1x).
    - ***-i video.mp4***: Indica el archivo de entrada
    - ***-f mpegts***: La salida será un flujo mpeg-ts en el puerto 10000
- El generador de *streaming* ***SRT-Live-Transmit*** que básicamente lo que hace es recibir como entrada un flujo de vídeo (Mpeg-TS) por el puerto 10000 y lo reenvía como *streaming* SRT al puerto "de escucha" 5200. Como se ve en la imagen de la **Figura 27.** se añade el parámetro ***passphrase=123456789101112*** que será la clave con la que se cifrará el contenido con AES de 256 bits y que necesitará el cliente para desencriptarlo.

3.1.2 Switch

Para emular el switch se utiliza la versión 15.2 de IOS para *switches* que trabajan en la capa 2 o capa de enlace. Se definirá un equipo de 8 puertos *GigabitEthernet* que se podría extrapolar al equipo comercial de Cisco de la serie Catalyst C2960-L.

Figura 28. Switch Catalyst C2960-L. Fuente:
https://tech-distributor.com/products/ws-c2960l-16ps-ll/

3.1.3 Router.

El *router* utilizado para la simulación también es una imagen de Cisco de la versión IOSv 15.6. Cuenta con cuatro interfaces *GigabitEthernet*. Este equipo podría equipararse con el equipo comercial de la serie 2900 que se muestra en la imagen de la **Figura 29.**

Figura 29. Router Cisco 2900 Series. Fuente:
https://www.priceblaze.com/cisco2901-k9-rf-Cisco-Network-Routers

3.1.4 Conexión a Internet.

GNS3 cuenta con un dispositivo denominado *Cloud*, representado con una nube, que permite a través de su máquina virtual, conectar el sistema a Internet. Éste actúa como un servidor de DHCP que proporciona una dirección del rango al que pertenece la máquina virtual.

En este caso, para poder acceder a la red a través de R1 y permitir la monitorización y reproducción del *streaming*, se necesita que la dirección IP pertenezca a la subred de la máquina anfitrión (simulando la IP pública).

Para ello, la máquina virtual de GNS3 se ha definido en VMWare con dos interfaces de red. Como se muestra en la imagen de la **Figura 30.**, una se configura en NAT, con lo que la máquina virtual tomará una IP virtual, la que toman los equipos definidos en ella (R1 y S1) que, por diferentes puertos, están accesibles por telnet.

Como se puede ver, el otro interfaz se define como *Bridged* para conseguir que la red sea accesible desde la máquina anfitrión.

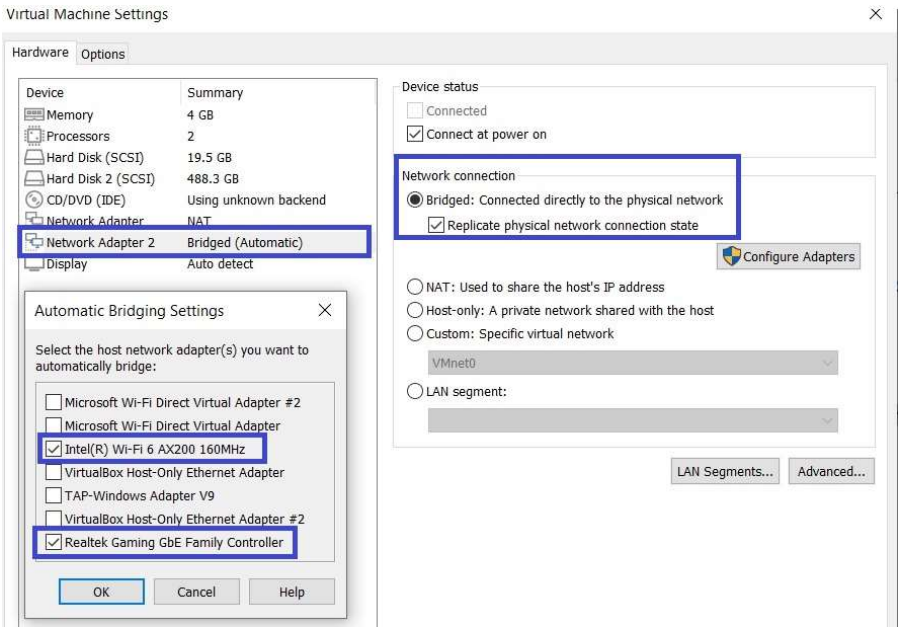

Figura 30. Configuración de red de la máquina virtual de GNS3

El dispositivo *Cloud* denominado ***Internet***, con esta configuración, permite la conexión a través de la máquina virtual por estas dos interfaces de red, y para el objetivo descrito, se utiliza la segunda.

En la imagen de la **Figura 31.** se representa la conexión de todos los elementos involucrados, tanto virtualizados con el *sofware* GNS3, como dispositivos reales.

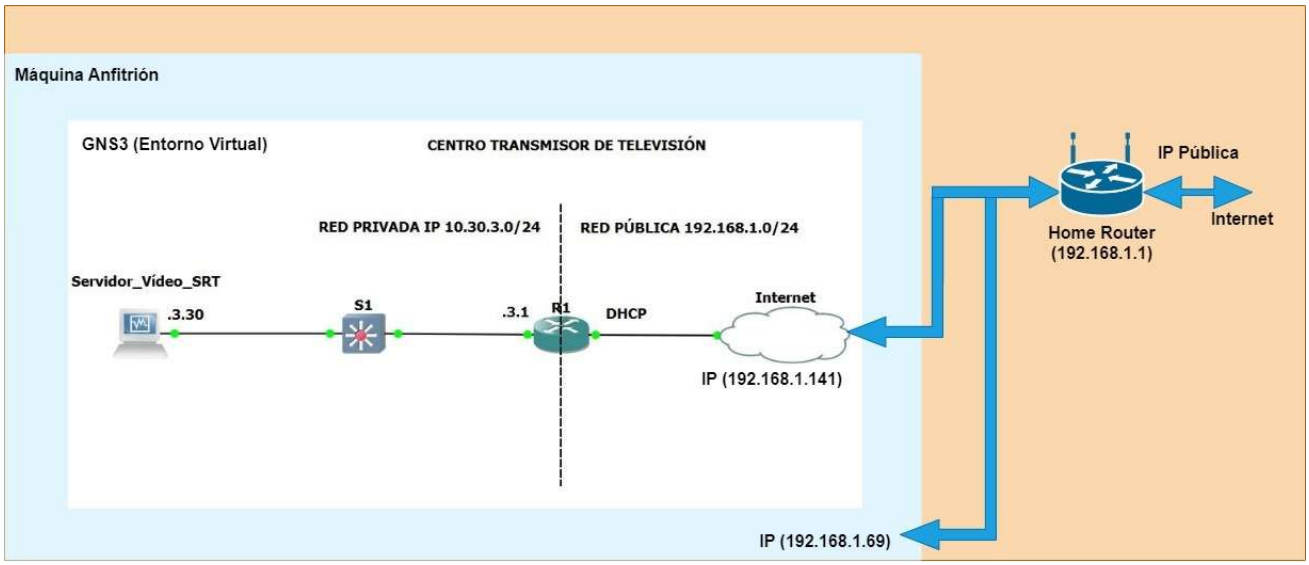

Figura 31. Conexión entre equipamiento virtual y equipamiento físico

3.2 Configuración

Una vez definida la topología de red se aborda la configuración de los equipos definidos en el apartado anterior.

3.2.1 Direccionamiento IP

Atendiendo a la **Tabla 4.** se define una red privada en el rango de direcciones 10.30.3.0/24. El servidor de *streaming* de vídeo SRT se configura con la dirección 10.30.3.30, tal y como se muestra en la imagen de la **Figura 32.**

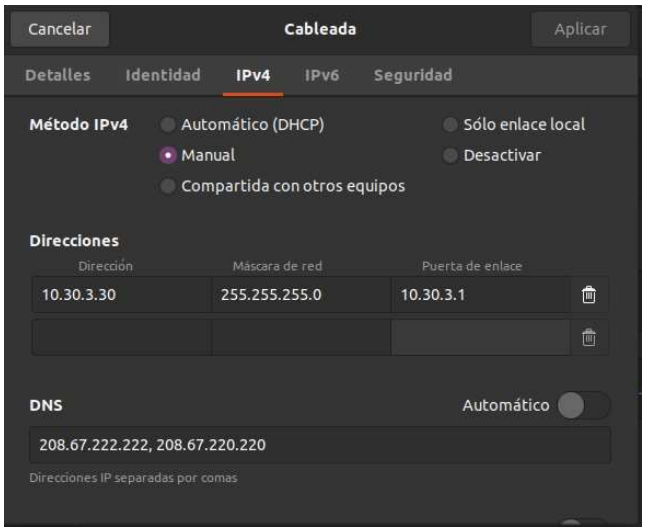

Figura 32. Configuración manual de la tarjeta de red del servidor de vídeo.

La puerta de enlace o *gateway* será el router R1 y tendrá la dirección IP 10.30.3.1. Para configurar esta interfaz de red (Gi0/0 de R1) habrá que utilizar los comandos CLI (*Command Line Interface*) de Cisco de la **Figura 33.**:

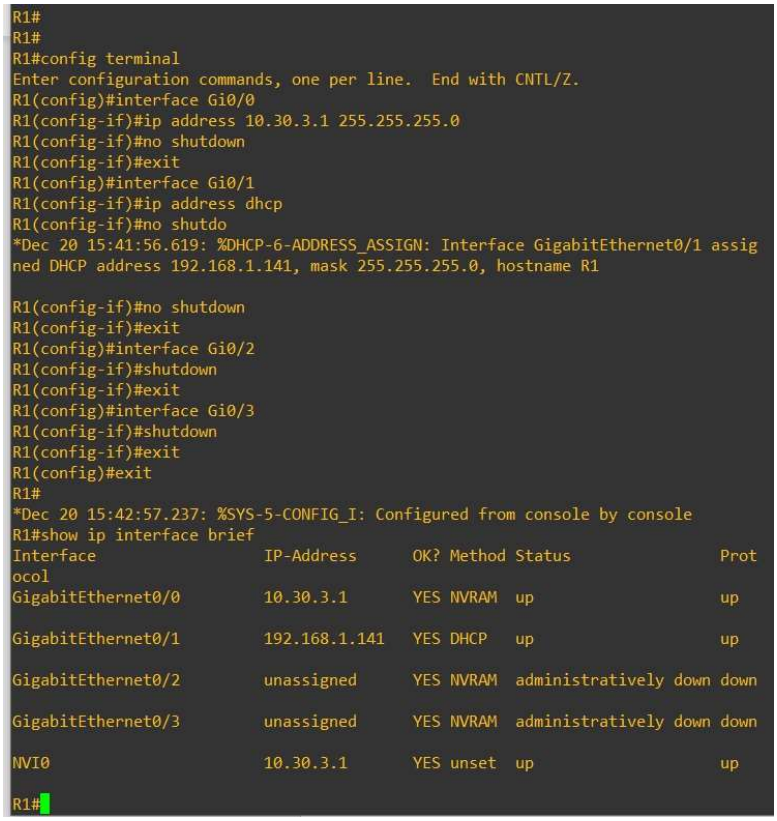

Figura 33. Configuración de los interfaces de red del *router* R1

El interfaz Gi0/1 emula la salida a Internet con una dirección pública. La recibe R1 por DHCP siendo 192.168.1.141, en la subred de la maquina anfitrión con IP 192.168.1.69. Los interfaces Gi0/2 y Gi0/3 se desactivan con el comando *shutdown.*

El switch S1 muestra la tabla de direcciones MAC y VLAN de la **Figura 34.**

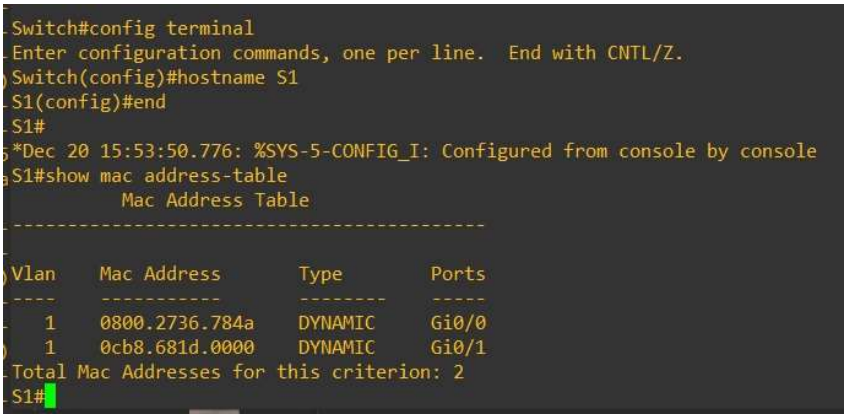

Figura 34. Tabla de direcciones MAC del *switch* S1

Una vez definidas las direcciones IP, se envían comandos PING desde R1 hacia el servidor de SRT y hacia Internet (8.8.8.8 dirección IP de un servidor DNS de Google) y

se observa que la configuración es correcta. Habrá que configurar el enrutamiento para que el servidor tenga conexión a Internet.

3.2.2 Enrutamiento.

En un primer momento se quiere permitir que todos los equipos de la red local tengan conexión a Internet. Para ello, tal y como se explicó en el apartado ***2.5.5.1 Encaminadores (Routers)***, habrá que realizar una conexión por NAT. Esto se define en R1 con los comandos CLI de la **Figura 35.**

```
R1#config terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface G0/0
R1(config-if)#ip nat inside
R1(config-if)#acces-list 1 permit 10.30.3.0 0.0.0.255
                 ^
% Invalid input detected at '^' marker.

R1(config-if)#access-list 1 permit 10.30.3.0 0.0.0.255
R1(config)#exit
R1#
*Nov 26 18:40:13.965: %SYS-5-CONFIG_I: Configured from console by console
R1#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface Gi0/1
R1(config-if)#ip nat outside
R1(config-if)#exit
R1(config)#ip nat inside source list 1 interface Gi0/1 overload
R1(config)#exit
R1#
*Nov 26 18:41:35.206: %SYS-5-CONFIG_I: Configured from console by console
R1#
```

Figura 35. Configuración de conexión NAT

En este caso hay una única dirección pública por lo que el NAT se denomina por sobrecarga (*overload*).

Se realiza un enrutamiento estático que permita el tráfico de entrada al servidor de vídeo por el puerto 5200, por donde se habilitará al *streaming*. Se permite el transporte por los dos protocolos de transporte: UDP y TCP. En la imagen de la **Figura 36.** se observa dicha configuración en comandos CLI:

```
R1#config terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#ip nat inside source static tcp 10.30.3.30 5200 interface GigabitEthernet0/1 5200
R1(config)#ip nat inside source static udp 10.30.3.30 5200 interface GigabitEthernet0/1 5200
```

Figura 36. Acceso al puerto 5200 del servidor de vídeo desde Internet

3.2.3 Análisis de primeros resultados

Se parte de un archivo de vídeo siguiendo las especificaciones descritas en el apartado ***2.3.2 MPEG-4 Part 10 (H.264/AVC)***.:

- Compresión de vídeo: H.264 HighProfile@4.0
- Resolución de Vídeo: 1920x1080 @ 25 fps.
- Compresión de audio: AAC
- Tasa de bits: 15-20 Mbps

Al intentar reproducir este archivo se observa que la reproducción del *streaming* SRT en la máquina anfitrión no es posible. Se establece la conexión, pero el vídeo llega prácticamente congelado la mayoría del tiempo y con gran cantidad de artefactos.

Se confirma con Wireshark que hay una gran cantidad de pérdida de paquetes. Se ilustra en la imagen de la **Figura 37.** Además, se puede observar que los paquetes van encriptados.

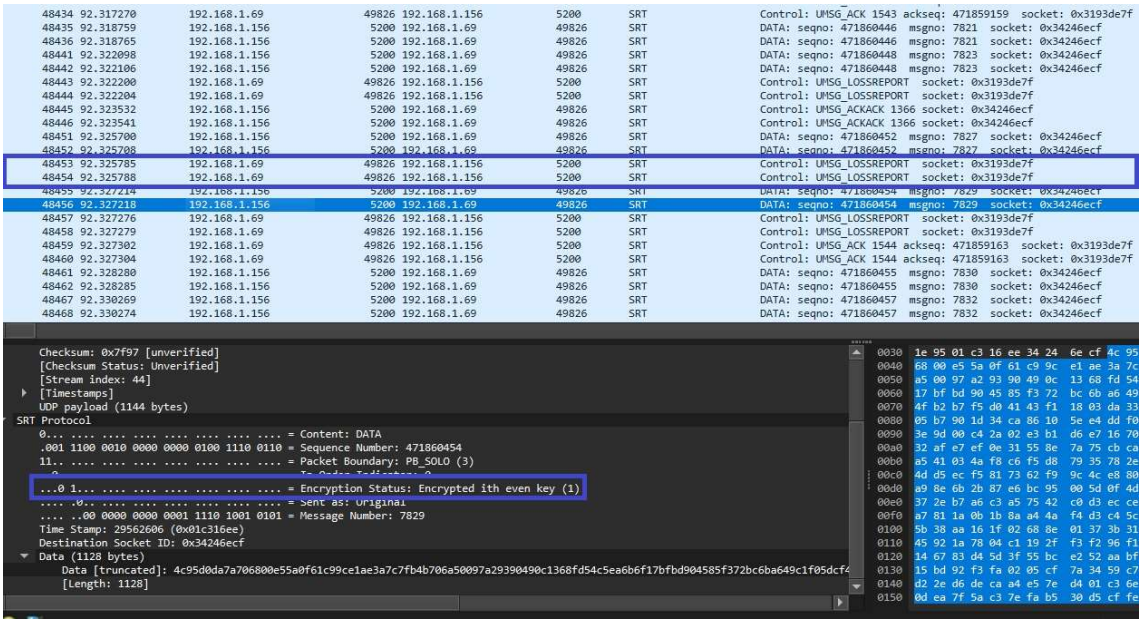

Figura 37. Captura de flujo SRT con pérdidas

Mientras que la máquina anfitrión cuenta con una conexión simétrica a Internet de 150 Mbps, el servidor de vídeo no supera los 1.5 Mbps generándose un cuello de botella que hace imposible alcanzar los parámetros de calidad necesarios para una contribución de televisión en el ámbito profesional.

Se verifica que este problema viene derivado del uso de dispositivos virtuales. GNS3 no es una herramienta adecuada para analizar la tasa de transferencia efectiva (*throughput*) en los equipos de red ya que, para gestionar los recursos consumidos del equipo anfitrión, limita el ancho de banda.

Atendiendo a esta situación se decide partir de un archivo con una tasa que se adecúe al *throughput* de la red:

- Compresión de vídeo: H.265 MainProfile@4.0
- Resolución de Vídeo: 720x576 @ 25 fps.
- Compresión de audio: AAC
- Tasa de bits: 1-1.5 Mbps

Con este archivo el flujo SRT que se genera no tiene pérdidas de paquetes y su reproducción es fluida, pero sin la calidad predefinida en los apartados anteriores

3.2.4 Tolerancia a fallos

Como se puede observar en la red definida en la **Figura 25.** cada uno de los elementos que la conforman sería un punto único de fallo (s*ingle point of failure*): un fallo en cualquiera de los componentes supondría una interrupción del servicio.

La solución que otorgaría a la red una alta disponibilidad sería la redundancia de todos los equipos. Se dispondría de dos servidores de vídeo independientes con la misma señal (*main* y *backup*) y síncronos. Cada uno de estos equipos estarían conectados a *switchers* independientes y a su vez a *routers* diferentes para salir a Internet (dos IPs públicas distintas). Este modelo se implementaría tanto en el centro emisor como en el transmisor. En el caso en que un elemento de la cadena falle se perdería un servicio, pero se dispondría del otro en su lugar, sólo habría que sustituir la señal en la cadena de emisión. El hecho de que fallen dos elementos simultáneamente en una cadena de emisión sería bastante improbable, por eso se hablaría de un servicio de alta disponibilidad.

A nivel de implementación para este TFG no supondría ningún cambio en la configuración de los equipos descritos, básicamente se duplicarían. En el centro receptor se estarían recibiendo dos flujos SRT por dos IP públicas diferentes.

3.3 Monitorización de red

Para la monitorización de los flujos que están presentes en la red y, por tanto, para monitorizar el estado del *streaming* de vídeo que se genera en la red definida, se habilita el protocolo de monitorización de red NetFlow de Cisco es su versión 9.

La estructura de datos que se envía desde los equipos está basada en plantillas que son configurables y definidas por el administrador en el equipo que los envía. Se pueden recibir tres tipos de paquetes de datos: Plantillas del formato del conjunto de flujos, opciones de formato de las plantillas y los registros de datos del conjunto de flujos de red (los datos y las plantillas pueden mandarse juntos en el mismo datagrama)

Esta configuración se realiza en el router R1 y básicamente se realiza siguiendo los siguientes pasos:

- Creación de un registro: Se definen en este paso las características o campos de los diferentes flujos de datos que se desea recolectar y que definirán la plantilla del formato de los datos. Entre otros, se registrarán las direcciones IP origen y destino, el protocolo de transporte, los puertos de origen y destino, los bytes y la duración de la trama con los *timestamps* de los paquetes primero y último.
- Definir a dónde exportar los datos: Los datos se envían a la máquina anfitrión por el puerto 2055.
- Crear un monitor de los datos: Con los datos configurados para ser enviados y definido el socket donde se van a exportar, se crea un monitor donde se indica un *timeout* para el envío de paquetes. Para este caso se enviarán cada 10 segundos.
- Aplicar el monitor de datos a un interfaz: Se especifica en que interfaz del elemento configurado (en este caso R1) se quiere monitorizar el flujo de

datos y en qué sentido. Para este caso, se ha decidido capturar el flujo en el interfaz público (Gi0/1) y de salida.

El proceso de configuración de estos pasos en el *router* R1 se detalla en la imagen de la **Figura 38.**

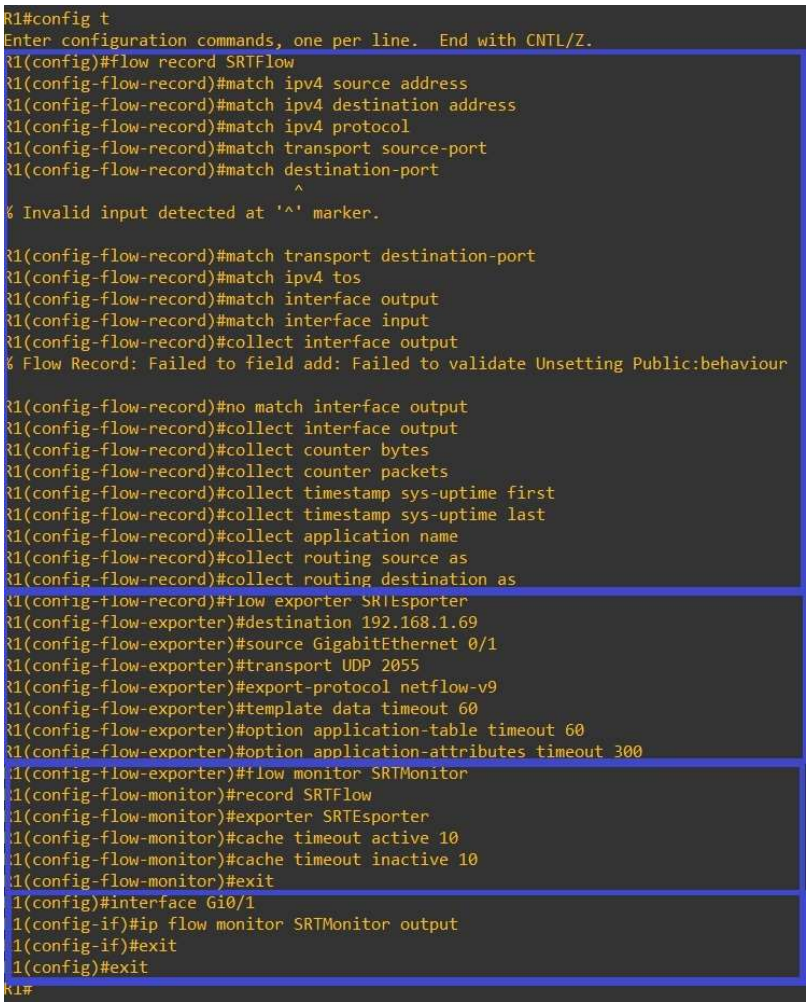

Figura 38. Configuración del protocolo NetFlow V9 en el router R1

Así mismo, realizando una captura de tráfico con Wireshark, se muestra en la imagen de la **Figura 39.** un ejemplo de los posibles paquetes que se reciben por este protocolo. En este caso se trata de un paquete donde se incluyen, en primer lugar, los campos de la plantilla, que coinciden con los definidos en R1 al definir el registro de datos. Cada uno de ellos indica el tipo de dato y su longitud. En la segunda parte (***FlowSet 2***) se reciben los datos de un flujo en concreto formateados con la plantilla. Para este caso justamente se recibe información sobre el flujo del *streaming* de vídeo SRT.

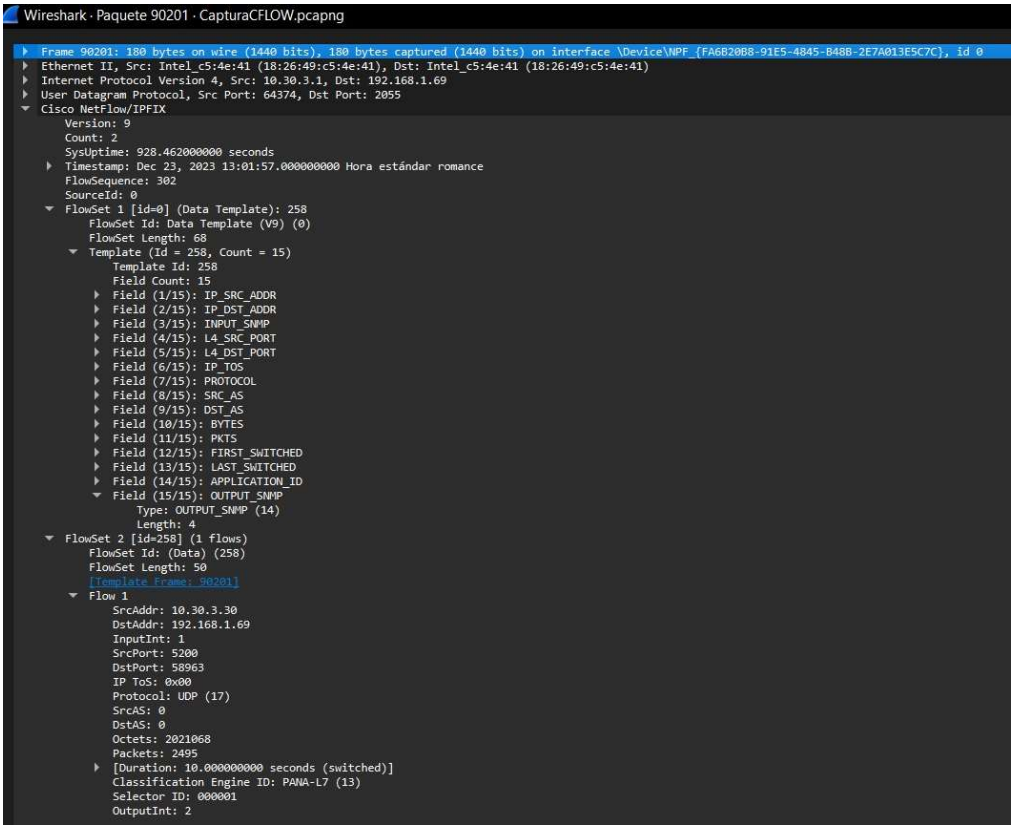

Figura 39. Paquete NetFlow capturado con Wireshark

3.3.1 Parámetros de monitorización

El tráfico NetFlow que recibe el equipo anfitrión se va a analizar y se van a obtener básicamente dos parámetros de monitorización:

- Estado del *streaming*: Teniendo en cuenta el *timeout* definido para el envío de datos, se va a determinar si el servicio de *streaming* está activo o no.
- El ancho de banda: Se va a calcular el ancho de banda del *streaming* utilizando tres campos recibidos con el protocolo NetFlow: LAST_SWITCHED, FIRST_SWITCHED y BYTES. Se calculará como (los valores de LAST_SWITCHED y FIRST_SWITCHED se reciben en milisegundos):

$$Bandwidth(Kbps) = \frac{8 \times BYTES}{(LAST\_SWITCHED - FIRST\_SWITCHED) \times 10^{-3} \times 1024}$$

3.4 Interfaz de usuario

El interfaz de usuario consiste en una aplicación web donde, por un lado, se refleja el ancho de banda del *stream* de vídeo en el transcurso del tiempo y, por otro, en el caso de que se detecte un corte, se dará la posibilidad de realizar un diagnóstico que intentará ofrecer un acotamiento en el error. En la imagen de la **Figura 40.** se muestra su apariencia:

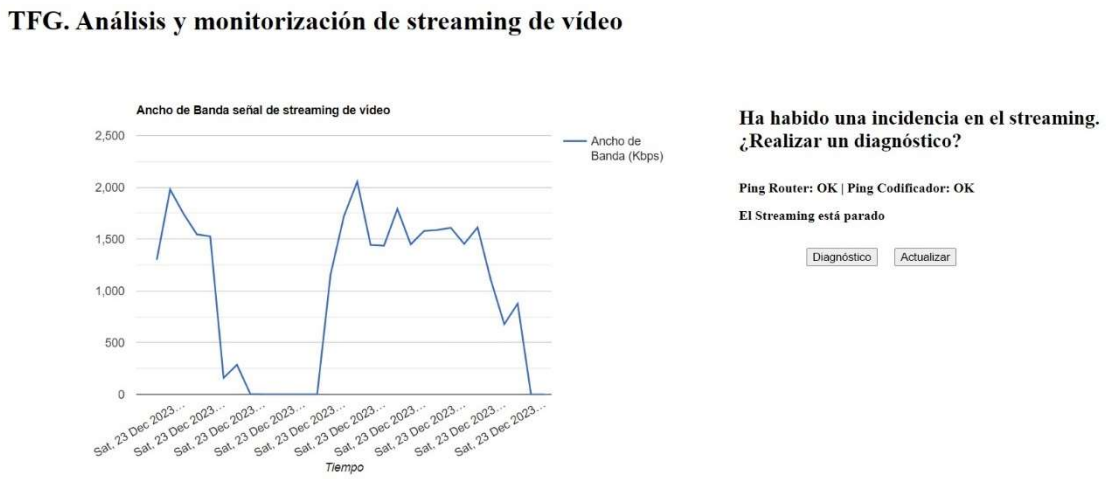

Figura 40. Interfaz gráfico basado en aplicación web

3.4.1 Definición del *front-end*

En la parte de representación de la aplicación web se ha utilizado un script para representación gráfica de datos denominada *Google Charts*. Dentro de las posibilidades de representación de datos se utiliza *LineChart*. En su función principal *DrawChart()*, que se muestra en la imagen de la **Figura 41.**, se reciben los 30 últimos valores de ancho de banda almacenados en la base de datos (***bw***) y se almacenan en un array ***yVal*** que se representarán en el eje de ordenadas. En el eje de abscisas se representan los valores temporales almacenados en ***xVal***, también extraídos de la base de datos en el *back-end.*

Se hace una actualización periódica de esta página cada 10 segundos para ir mostrando los últimos valores almacenados en la base de datos.

```
function drawChart() {

  let yVal = {{ bw|tojson }};
  let xVal = {{ fechas|tojson }}
  // Set Data
  const data = google.visualization.arrayToDataTable([
    ['Fecha', 'Ancho de Banda (Kbps)'],
    [xVal[30],Number(yVal[30])],[xVal[29],Number(yVal[29])],[xVal[28],Number(yVal[28])],
    [xVal[27],Number(yVal[27])],[xVal[26],Number(yVal[26])],
    [xVal[25],Number(yVal[25])],[xVal[24],Number(yVal[24])],[xVal[23],Number(yVal[23])],
    [xVal[22],Number(yVal[22])],[xVal[21],Number(yVal[21])],[xVal[20],Number(yVal[20])],[xVal[19],Number(yVal[19])],
    [xVal[18],Number(yVal[18])],[xVal[17],Number(yVal[17])],[xVal[16],Number(yVal[16])],
    [xVal[15],Number(yVal[15])],[xVal[14],Number(yVal[14])],
    [xVal[13],Number(yVal[13])],[xVal[12],Number(yVal[12])],[xVal[11],Number(yVal[11])],
    [xVal[10],Number(yVal[10])],[xVal[9],Number(yVal[9])],[xVal[8],Number(yVal[8])],
    [xVal[7],Number(yVal[7])],[xVal[6],Number(yVal[6])],[xVal[5],Number(yVal[5])],
    [xVal[4],Number(yVal[4])],[xVal[3],Number(yVal[3])],
    [xVal[2],Number(yVal[2])],[xVal[1],Number(yVal[1])],[xVal[0],Number(yVal[0])]
  ]);

  // Set Options
  const options = {
    title: 'Ancho de Banda señal de streaming de vídeo',
    hAxis: {title: 'Tiempo'},
    vAxis: {title: 'Ancho de Banda (Kbps)'},
    legend: 'None',
    bar: {groupWidth: '95%'},
    vAxis: {minValue:0},
    vAxis: {minValue:2000}
  };

  // Draw
  const chart = new google.visualization.LineChart(document.getElementById('myChart'));
  chart.draw(data, options);

}
```

Figura 41. Script que permite la visualización gráfica de los datos.

Figura 42**.** se muestra como en el momento en que se detecta

Una vez que desde el back-end se detecta que el servicio de *streaming* se ha parado, se habilitan dos formularios asociados a los botones ***Actualizar*** y ***Diagnóstico.*** Utilizan los métodos GET y POST de HTTP respectivamente para acceder a la página principal.

A continuación, en la definición del *back-end* se describirá como se procesa cada una de estas llamadas.

```
<h1>TFG. Análisis y monitorización de streaming de vídeo</h1>
<div style="width: 1400px; overflow: hidden; align-items: center;">
  <div id="myChart" style="width:100%; max-width:800px; height:500px;float: left;"></div>

  {% if estado=="nonok" %}
  <div style="padding-top: 3%; padding-left: 5%; overflow: hidden;"><p><h2>Ha habido una incidencia en el streaming. ¿Re
  {% endif %}
  {% with messages = get_flashed_messages() %}
    {% if messages %}
      {% for message in messages %}
        <div style="padding-top: 1%; padding-left: 5%; overflow: hidden;">
          <strong>{{ message }}</strong>
        </div>
      {% endfor %}
    {% endif %}
  {% endwith %}
  {% if estado=="nonok" %}
    <div style="width: 400px; overflow: hidden; align-items: center;margin-left: 200px;">
      <div style="float: left;margin-left: 150px;margin-top: 30px;">
        <form action="/" method="post">
          <input type="submit" value="Diagnóstico" name="Actualizar"></input>
        </form>
      </div>
      <div style="padding-left: 5%; overflow: hidden;margin-top: 30px;">
        <form action="/" method="get">
          <input type="submit" value="Actualizar" name="" ></input>
        </form>
      </div>
    </div>
  {%endif%}
  {%if estado=="ok"%}
  {% endif %}
</div>
{% endblock %}
```

Figura 42. Página de bienvenida de la aplicación web.

3.4.2 Definición del *back-end*

Previamente a la definición del *back-end* de la aplicación web implementada, se va a describir el funcionamiento del programa que recolecta los datos de monitorización desde la red emulada y que se ejecutará en paralelo.

Se basa en la librería ***netflow*** de *Python* que lee los datos recibidos en el puerto 2055 y los formatea teniendo en cuenta la estructura de datos definida en el protocolo NetFlow de Cisco. Abarca diferentes versiones de este protocolo. Los datos formateados se exportan en objetos de clases según los paquetes recibidos: plantillas de formato de datos, opciones del formato de las plantillas o datos de flujo.

Se ha implementado una clase denominada ***Collector.py*** que genera una conexión a la base de datos de *MySQL* donde se almacenarán los campos de los datos recibidos. Con ***netflow*** se reciben los datos de monitorización formateados procedentes de la red virtual. Ésto se pasa a un objeto de la clase ***Analyzer.py*** que discrimina si el paquete recibido contiene datos de flujo correspondientes al *streaming* de vídeo. Si es así se almacenan en la base de datos. En caso contrario, se compara el *timestamp* recibido con el último almacenado del *streaming* y si han pasado más de quince segundos entre ellos se indica que el *streaming* ha cesado.

Se representa su funcionamiento en el diagrama de la **Figura 43.** Diagrama de funcionamiento del procesado de datos**.**

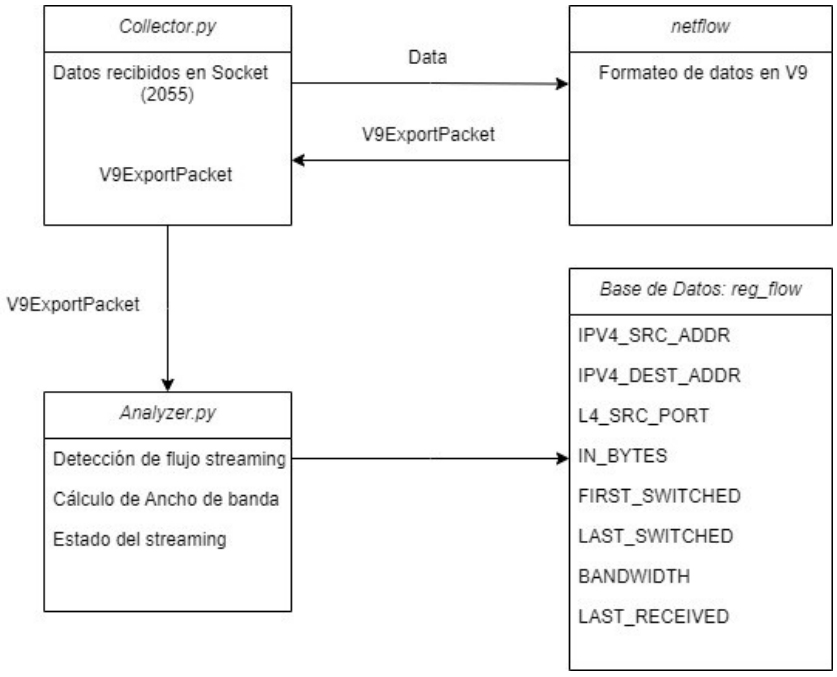

Figura 43. Diagrama de funcionamiento del procesado de datos.

Se utiliza la librería ***PyInstaller*** de *Python* para generar un archivo ejecutable *.exe* (***collector.exe***) que permita su puesta en marcha desde Windows con doble clic.

Como se ha comentado, ***collector.py*** se ejecuta en paralelo a la aplicación web definida con *Flask,* uno de los *frameworks* de *Python* para desarrollo de este tipo de aplicaciones. El archivo que lo implementa se denomina ***monitor.py*** y contempla una única función ***home()*** asociada a la página de bienvenida de la aplicación web. Se discrimina si la llamada a esta página se hace por método GET o método POST:

- *GET*: Se extraen de la base de datos los últimos 30 registros asociados al *streaming* de vídeo. Si el último contine información de flujo se determina que el *streaming* está en marcha y se envían los registros al *front-end* para representar dichos datos.

  Si el último registro no contiene datos se indica que el *streaming* está parado para que el *front-end* muestre esta situación y permita realizar el diagnóstico.

  En la captura de la **Figura 44.** Tratamiento de la aplicación web a peticiones con método GET.

  . se muestra la implementación desarrollada:

```python
@app.route("/", methods=['GET','POST'])
def home():

    if request.method=="GET":
        estado=""
        cursor=mysql.connection.cursor()
        cursor.execute("""SELECT BANDWIDTH, LAST_RECEIVED, IPV4_DST_ADDR
                        FROM videostreaming
                        ORDER BY LAST_RECEIVED
                        DESC LIMIT 30""")

        data = cursor.fetchall()

        bw=[]
        fechas=[]#Si el valor de este registro es nulo, querrá decir que el streaming ha sido interrumpido
        ip_address=[]

        for dato in data:
            bw.append(dato[0])
            fechas.append(dato[1])
            ip_address.append(dato[2])

        if(ip_address[0]==None):
            flash("El Streaming está parado")
            estado = "nonok"
        else:
            session.pop('_flashes', None)
            flash("El Streaming está activo")
            print("bw[0]={}".format(float(bw[0])))
            if( float(bw[0]) < 100 and float(bw[0]>0)):
                flash("Atención, el ancho de banda del streaming es muy bajo")
            estado="ok"

        print("estado= {}".format(estado))
        return render_template("index.html",bw=bw,fechas=fechas,estado=estado)
```

Figura 44. Tratamiento de la aplicación web a peticiones con método GET.

- *POST*: Este método se invoca desde el *front-end* para realizar el diagnóstico una vez se ha detectado la interrupción del *streaming*. Para ello se utiliza una nueva librería de *Python* denominada ***netmiko*** que permite la conexión SSH y envío de comandos CLI a dispositivos Cisco. Se puede detectar si el router no conecta o si el servidor de vídeo no responde al comando PING. También detecta si el interfaz Gi0/0 del router está caído.

  En la imagen de la **Figura 45.** Tratamiento de la aplicación web a peticiones con método POST.

  . se muestra la implementación descrita:

```
if request.method == 'POST':
    try:
        R1 = ConnectHandler(
            host='192.168.1.141',
            port='22',
            username='admin',
            password='admin',
            device_type='cisco_ios'
        )
        session.pop('_flashes', None) #Borra mensajes de advertencia
        if R1.is_alive(): #El router está conectado y operativo
            output = R1.send_command("ping 10.30.3.30",read_timeout=30.0)#Ping a servidor de vídeo
            if "Success rate is 0" in output: #Fallo en el ping
                flash("Ping Router: OK | Ping Codificador: Fault")
                output=R1.send_command("show ip interface g0/0", read_timeout=30.0)
                if "GigabitEthernet0/0 is down" in output: #Interfaz caída
                    flash("Interfaz de conexión con Switch S1 caída")
                R1.cleanup()
                return redirect(url_for("home"))
            else:
                flash("Ping Router: OK | Ping Codificador: OK")
                R1.cleanup()
                return redirect(url_for("home"))
        else:
            flash("Ping Router: Fault")
            R1.cleanup()
            return redirect(url_for("home"))
    except Exception as e:
        print("Error {}".format(e.args))
        flash("Error: Existe un problema para conectar por ssh al router")
```

Figura 45. Tratamiento de la aplicación web a peticiones con método POST.

4. Conclusiones y líneas futuras

Como primera conclusión al finalizar la implementación hay que indicar que no se han podido alcanzar los criterios de calidad exigidos para un enlace de contribución de vídeo por *streaming* SRT. Sin embargo, todo apunta a que, si la configuración se hubiera realizado con equipamiento físico y no virtual, no habría estado presente el problema de cuello de botella por la limitación en el ancho de banda y se hubiera conseguido un enlace con la calidad exigida.

Una vez superado este problema, se observa que el protocolo SRT funciona de manera solvente y el resultado, trabajando dentro de la tasa de transferencia efectiva de

la red es óptimo. En las pruebas realizadas se define una latencia del orden de los 400ms y responde correctamente incluso con encriptación.

De cara a la herramienta de monitorización implementada (con posibilidades de automatización en la configuración de equipos) hay que comentar que podría ser funcional en un entorno de producción real. De manera muy intuitiva, gracias a la representación gráfica, se muestra al usuario el estado del *streaming* de vídeo en todo momento. Del mismo modo, una vez se detecta un corte en la señal, la herramienta puede ayudar a personal cualificado a acotar el error que se está produciendo y solucionarlo.

Creo que en el tiempo disponible para el desarrollo de este trabajo se ha conseguido estudiar los elementos teóricos y descubrir las herramientas necesarias para conseguir lo especificado. Se ha podido implementar una herramienta de monitorización completa que acomete las necesidades principales de cara a poder garantizar una alta disponibilidad en el servicio.

A nivel personal, aunque con el tiempo siempre en contra, he podido tener un acercamiento a lo que sería la administración de redes y poder configurar las características que demandaba el trabajo. Me sirve como punto de partida para un trabajo que me ha parecido atractivo e interesante en todo momento. Además, he podido conocer las posibilidades de control y monitorización que tienen estos equipos con *Python* y llevarlo a una aplicación web que, tras muchas modificaciones y retoques ha quedado funcionando correctamente.

Se detallan a continuación ciertas mejoras o posibilidades de ampliación en un futuro:

- Desplegar el trabajo en producción con equipamiento real: Vista la deficiencia en la gestión del ancho de banda por parte del software de red virtual, lo primero sería configurar y probar lo propuesto en equipos reales. En este entorno, la batería de pruebas y análisis sobre el protocolo SRT serían más interesantes, poniendo en relieve sus verdaderas prestaciones.
- Sería interesante añadir a la herramienta de monitorización un análisis sobre el flujo SRT que indique cuando hay pérdidas de paquetes que tendrán su impacto sobre la señal de vídeo reproducida en la recepción.
- Más allá de la encriptación que hace SRT, no se ha abordado, por falta de tiempo, medidas de seguridad a implementar en el sistema. Se podría haber configurado un firewall en los propios equipos utilizados o añadir un equipo específico para ello (por ejemplo, de Fortinet). Con esto se hubiera limitado el acceso exclusivamente al puerto 5200 desde la WAN.
- La apariencia del *front-end* implementada está basada en la funcionalidad más que en la propia estética. Se podría trabajar en una hoja de estilos que la mejorara.

Glosario.

- ***Back-end*:** Parte del software que implementa la capa de proceso de los datos de para mostrar los resultados en el *front-end*.
- **CLI:** Interfaz de envío de comandos que permite interactuar y enviar comandos de configuración a los equipos de red en este caso.
- **Compresión**: Reducción de la información de la señal de televisión basada en la redundancia espacial y temporal de las imágenes.
- **Contribución**: Proceso al que se somete a la señal de televisión desde su captura en origen hasta que se envía a la cabecera de distribución.
- **Distribución**: Enlaces que llevan la señal de televisión desde los centros de producción de televisión a las cabeceras de que se encargarán de difundir la señal hasta el cliente final.
- ***Framework*:** Entorno de trabajo, conjunto de prácticas estandarizadas que sirven de referencia para realizar un trabajo.
- ***Front-end*:** Parte del software con la que tiene interacción el usuario recogiendo los datos de entrada.
- **HTTP:** Protocolo de transferencia de información a través de archivos en la web.
- **Jitter:** Fluctuación en el retardo que se produce al procesar los paquetes en la red. Puede hacer que ciertos paquetes lleguen demasiado tarde o temprano para ser entregados.
- **LAN:** Red de área local o red privada donde pueden interactuar entre los dispositivos conectados a nivel local.
- **Latencia**: Demora en la transmisión y propagación de los paquetes que circulan por una red de datos.
- **PING:** Utilidad que se utiliza para diagnóstico de conectividad y latencia del equipo anfitrión con otra máquina de la red.
- **SRT:** Protocolo de *streaming* que favorece una comunicación confiable con recuperación de paquetes y ultra baja latencia. Detecta el funcionamiento de la red entre los puntos en comunicación para realizar un *buffering* dinámico.
- ***Streaming***: Transmisión de contenido multimedia a través de una red de ordenadores.
- ***Television Broadcaster*:** Red de telecomunicaciones destinada a la transmisión de televisión donde existe un centro de producción principal.
- ***Throughput*:** Tasa de transferencia efectiva. Volumen neto de datos que fluyen en un sistema, en este caso una red de datos
- ***Timestamp*:** Marca temporal que llevan los paquetes que se propagan por una red para establecer el orden de llega y momento de salida.
- **WAN:** Red de área extensa que interconecta ordenadores en un contexto geográfico mayor que las de área local. Los miembros que interactúan, por tanto, no se encuentran en la misma ubicación.

Anexos.

Anexo 1: Conceptos sobre la señal de televisión digital.

De forma general se va a resumir en este anexo algunos aspectos de la producción de la señal de televisión digitalizada en la cadena de producción.

En los platós de televisión se utilizan sistemas de captación tanto de imagen como de sonido. La señal de vídeo se genera en formato RGB en la cámara y por matrizado se genera una señal con tres componentes: Y (Luminancia), C_B (Diferencia de color (B-Y)) y C_R (Diferencia de color (R-Y)).

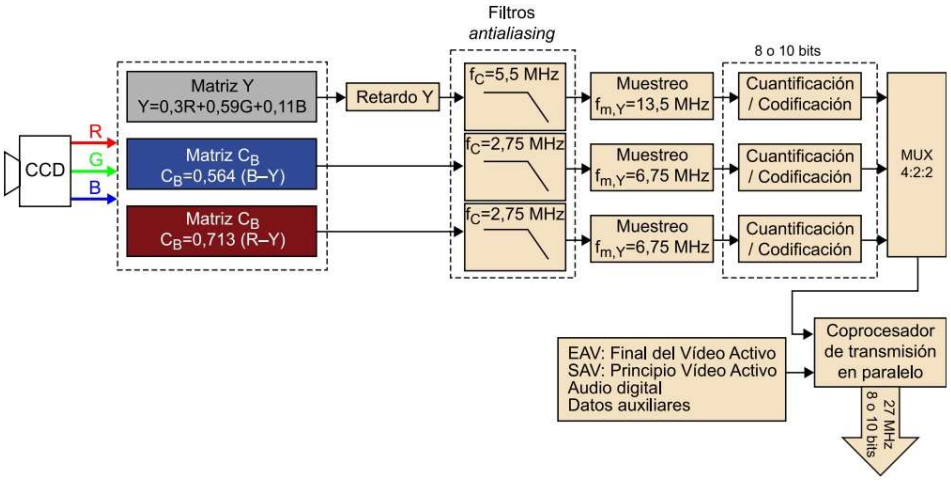

Figura 46. Diagrama de bloques de la señal de televisión digital para transmisión en paralelo. Fuente: [25]

La señal sin pérdidas es la que, en la digitalización, contiene el mismo número de muestras de luminancia que de diferencia de color y se denomina perfil 4:4:4 (por cada 4 muestras de la señal de luminancia se obtienen 4 muestra de cada una de las señales diferencia de color). Sin embargo, debido al alto régimen binario que se obtiene al multiplexar la información de las 3 señales y a que el ojo es mucho más sensible a las variaciones de luminancia que a las de crominancia, se definen diferentes perfiles de submuestreo para crominancia, tal y como se ilustra en la imagen de la **Figura 47.** Perfiles de submuestreo de crominancia: 4:2:2, 4:2:0 y 4:1:1

Fuente: [25].

En el perfil 4:2:2 (diagrama de bloques de la **Figura 46.** Diagrama de bloques de la señal de televisión digital para transmisión en paralelo.

Fuente: [25]

.), cada una de las componentes de color se muestrea a la mitad de frecuencia, obteniéndose 2 muestras de componente de color por cada 4 de la señal de luminancia. Otros perfiles que reducen aún más este régimen binario son 4:1:1 y 4:2:0 donde por cada 4 muestras de luminancia se obtienen 1 de cada componente de color. La diferencia entre ambos es la distribución espacial de las mismas, tal y como se muestra en la imagen de la **Figura 47.** Perfiles de submuestreo de crominancia: 4:2:2, 4:2:0 y 4:1:1

Fuente: [25]. En 4:2:0 se realiza la misma reducción de color en horizontal que en vertical.

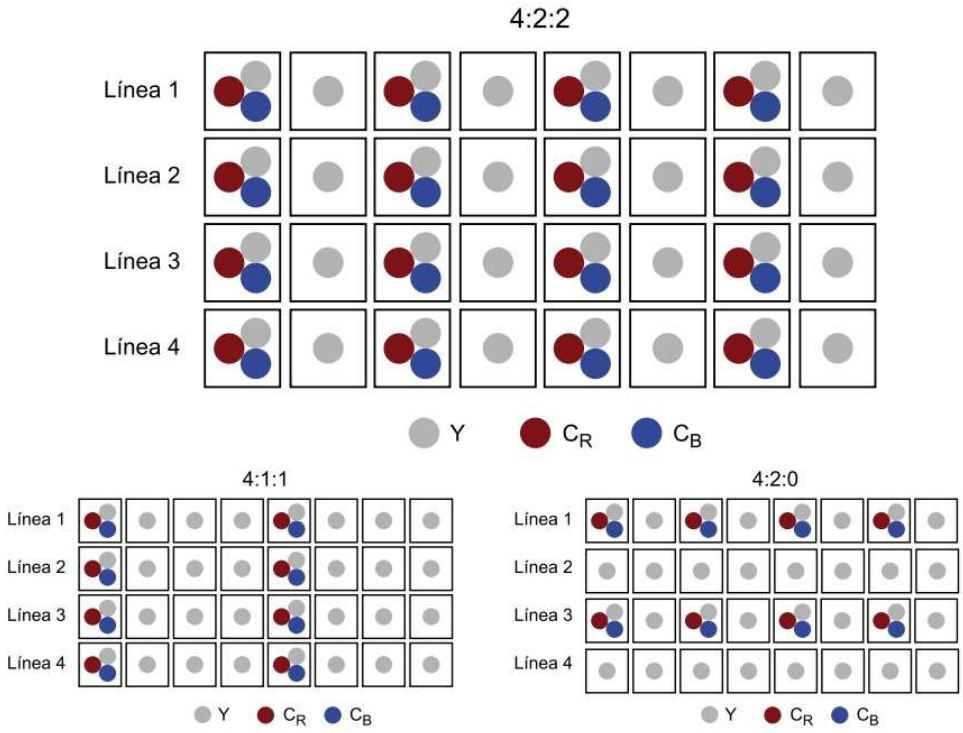

Figura 47. Perfiles de submuestreo de crominancia: 4:2:2, 4:2:0 y 4:1:1
Fuente: [25]

Los valores de crominancia y luminancia representados dependen del número de bits (normalmente 8 o 10 bits) que se utilicen en el muestreo de cada una de las tres componentes. En el caso de la crominancia se habla de profundidad de color y se suelen utilizar 8 o 10 bits.

Otros aspectos que inciden en la calidad de la señal de televisión son la relación de aspecto y la resolución. En señales de definición estándar SD (*Standard Definition*) se establece un total de 576 (o 480 en América) líneas divididas en 720 puntos o píxeles. La alta definición HD (*High Definition*) presenta una relación de aspecto de 16:9 y por norma se utilizan resoluciones de 1920x1080 y 1280x720 píxeles. En la imagen de la **Figura 48.** Resoluciones y relaciones de aspecto más frecuentes en televisión.

Fuente: [25]. se esquematizan algunas de las resoluciones y relaciones de aspecto más utilizadas.

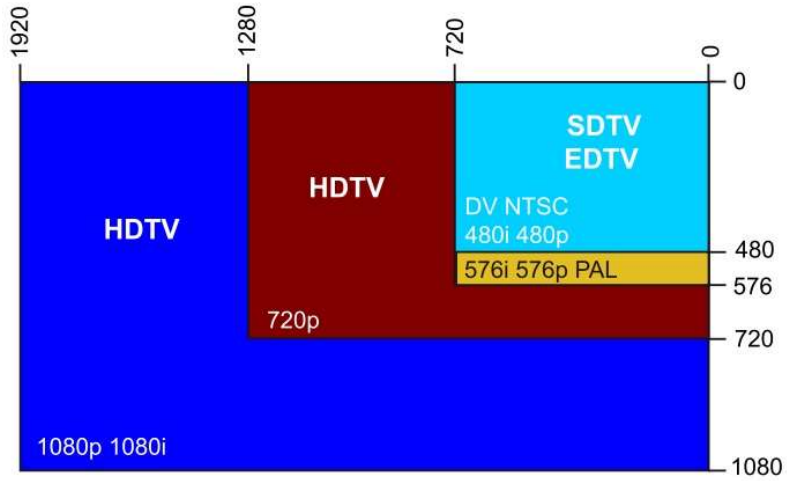

Figura 48. Resoluciones y relaciones de aspecto más frecuentes en televisión.
Fuente: [25]

La velocidad a la que se muestran las imágenes completas se establece en 25 imágenes por segundo (o 30 en el sistema americano) basándose en la capacidad del ojo humano de integrar las imágenes y obtener sensación de movimiento. Esta exploración en los sistemas de captación y representación pueden ser entrelazada o progresiva. En la exploración entrelazada se duplica la frecuencia del sistema, pero verticalmente se muestran la mitad de las líneas que componen un cuadro. Este conjunto de líneas se denomina campo que puede ser par o impar dependiendo si las líneas que lo componen son las pares o impares del cuadro. Por tanto, una imagen la forman 2 campos y la frecuencia será de 25 imágenes (f=25 Hz) o 50 campos (f=50 Hz) por segundo. Como ventaja, la exploración entrelazada presenta en el ámbito digital una reducción de las exigencias de ancho de banda a transmitir. Como desventaja con respecto a la exploración progresiva presenta una deficiencia en la representación haciendo que subjetivamente la señal sea menos nítida, especialmente en imágenes con mucho movimiento.

Por último, el audio se muestrea a una frecuencia de 48 KHz y se utilizan entre 20 y 24 bits por muestra. Se genera una subtrama de 32 bits (una por canal). El flujo se organiza con 192 tramas y cada trama la conforman 2 subtramas. De este modo, para audio estereofónico, se emite el canal izquierdo en una subtrama 1 y el derecho en la subtrama 2.

Este audio ira embebido en el espacio HANC de la trama SDI acompañando al vídeo junto a datos auxiliares y código de tiempo.

Anexo 2: Instalación de GNS3

El software GNS3 es gratuito y es posible su adquisición en https://www.gns3.com/software/download una vez creada una cuenta de usuario. En este caso, el equipo anfitrión donde se va a instalar tiene como sistema operativo Windows 10 de 64 bits.

Para un funcionamiento más optimizado y aprovechar al máximo sus características se recomienda la instalación de la máquina virtual (***GNS3 VM***) que actúa como servidor. Se ofrece tanto imágenes para *VirtualBox* como para *VMWare*, pero se recomienda que se utilice ésta última y es la que se descarga e instala. La configuración de la máquina virtual es la que se muestra en la imagen de la **Figura 49.** Características de la máquina virtual de GNS3 que actúa de servidor**.**

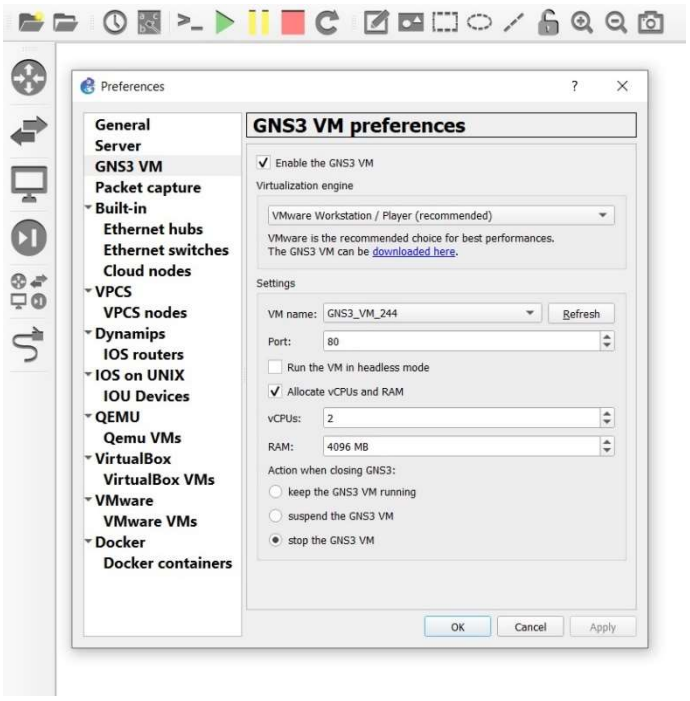

Figura 49. Características de la máquina virtual de GNS3 que actúa de servidor.

Una vez instalado, al ejecutar GNS3 se muestra el interfaz de usuario (GUI – *Graphical User Interface*) de la **Figura 50.** Aspecto de la interfaz gráfica de GNS3**.**, bastante intuitivo a la hora de definir los posibles laboratorios de red.

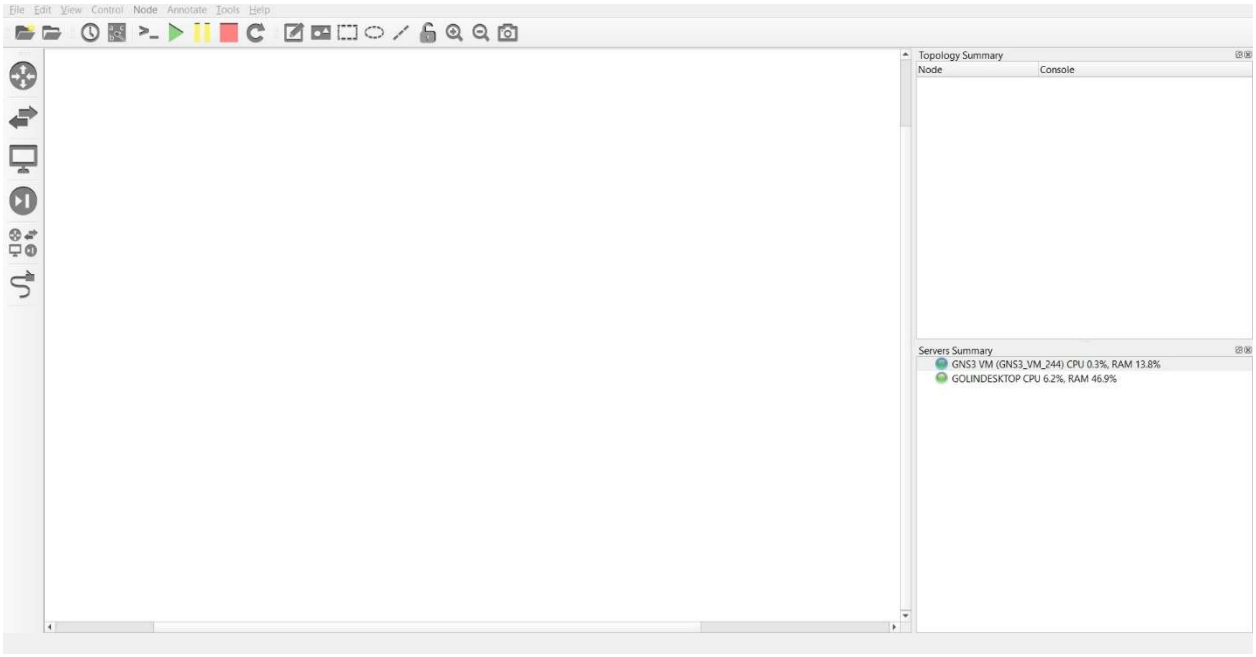

Figura 50. Aspecto de la interfaz gráfica de GNS3.

Para poder definir la topología expuesta en el apartado **3.1 Escenario 1. Topología de Red.** es necesario la instalación previa de los equipos de red virtuales. En este caso se instala la imagen del sistema operativo IOS de Cisco en su versión 15.2 para el *switch* de nivel 2 y la versión 15.16 para el *router*. Siguiendo las recomendaciones de GNS3, estas imágenes virtualizadas se instalan en la máquina virtual ***GNS3 VM*** con el software de virtualización *QEMU*, tal y como se ilustra en la imagen de la **Figura 51.** Definición de los dispositivos virtuales Cisco instalados en GNS3

.

Figura 51. Definición de los dispositivos virtuales Cisco instalados en GNS3

Tal y como se muestra en el apartado ***Servers Summary*** en la imagen de la **Figura 52.** Uso de recursos repartidos entre la máquina anfitrión y la GNS3 VM**.** se muestran los recursos de memoria RAM y CPU repartidos entre los dispositivos virtuales instalados y gestionados en la máquina virtual y los de la máquina anfitrión.

Figura 52. Uso de recursos repartidos entre la máquina anfitrión y la GNS3 VM.

Anexo 3: Configuración del servidor web.

Para la instalación y configuración del servidor *Apache* que va a alojar la aplicación web desarrollada se instala *WAMPServer* que realiza una instalación completa en Windows de todos los elementos para el desarrollo de aplicaciones web. Los elementos que instala son:

- *Apache*: Servidor web que permite al usuario la interacción con la aplicación web.
- *MySQL*: Gestor de base de datos donde se va a crear la base de datos que alberga los datos necesarios para la aplicación web desarrollada.
- *PHP*: La instalación por defecto está configurada para el uso de *PHP* como lenguaje de programación para el desarrollo de aplicaciones web

La elección de este paquete *software* se basa en que, aparte de su fácil instalación y configuración, cuenta con una aplicación web instalada para la administración de bases de datos en *MySQL* denominada *PHPMyAdmin* que, de forma sencilla, permite crear bases de datos, agregar o eliminar tablas dentro de ellas y sus campos asociados. En la imagen de la **Figura 53.** Definición de la base de datos utilizada**.** se muestra la base de datos creada (***reg_flow***) y la tabla (***videostreaming***) en la que se almacenarán los datos asociados al *stream* de vídeo SRT generado.

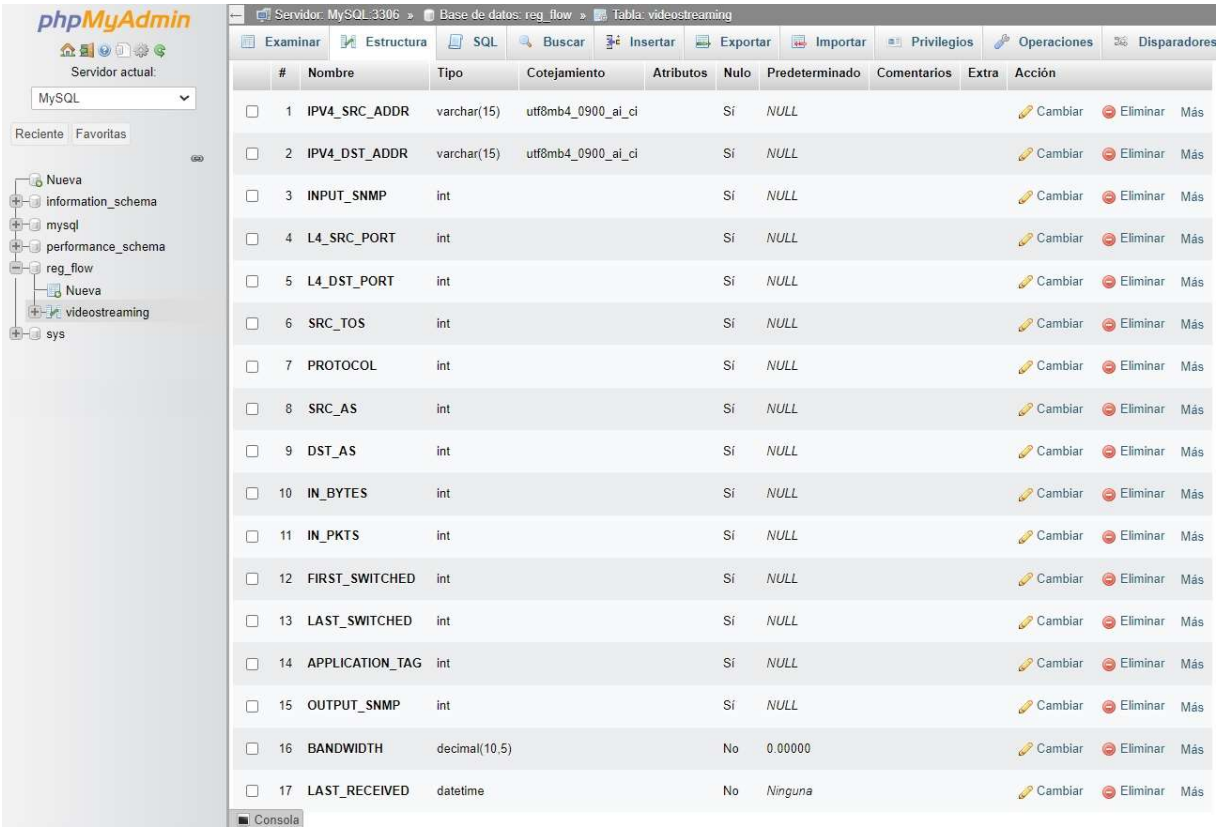

Figura 53. Definición de la base de datos utilizada.

Los campos creados en esta tabla reflejan la estructura de datos asociada al protocolo NetFlow y al registro definido en el *router* R1. Se añaden los campos **BANDWIDTH** y **LAST_RECEIVED** para el control del ancho de banda y estado del *stream* de vídeo SRT.

Aunque esta distribución está pensada para aplicaciones desarrolladas con PHP, se puede cambiar la configuración en el servidor para alojar aplicaciones desarrolladas en *Python*. En concreto los pasos a seguir son los siguientes:

- Instalar la librería de *Python* denominada ***mod-wsgi***: Este módulo implementa un interfaz para alojar aplicaciones web basadas en *Python* en servidores web Apache.
- Crear el archivo de configuración del interfaz *wsgi*: En el directorio donde se encuentra el archivo de la aplicación (***monitor.py***) se crea un archivo de extensión *.wsgi* (***run.wsgi*** en este caso) que sirve al interfaz para indicarle qué archivo contiene la aplicación web y cuál es su ubicación en el equipo.

```
import sys
sys.path.insert(0,'c:/python/tfg/env/src/')
from monitor import app as application
```

Figura 54. Archivo *run.wsgi*

- Modificar la configuración en el servidor *Apache*: En el archivo de configuración del servidor *Apache* ***httpd.conf*** se añade lo necesario para indicar al servidor dónde se encuentra dicho interfaz, la ubicación de la instalación de *Python* en el equipo y dónde se encuentra la aplicación desarrollada.

```
# Flask Project
LoadFile "C:/python310/python310.dll"
LoadModule wsgi_module "C:/python/tfg/env/lib/site-packages/mod_wsgi/server/mod_wsgi.cp310-win_amd64.pyd"
WSGIPythonHome "C:/python/tfg/env"
WSGIPythonPath "C:/python/tfg/env/"
```

Figura 55. Modificación del archivo de configuración *httpd.conf*

Se define el acceso al servidor web como *host* virtual (permitiría alojar varias aplicaciones en el mismo servidor) en el puerto 80 en el archivo de configuración de *Apache* ***httpd-vhosts***.

```
# Virtual Hosts
#

<VirtualHost *:80>
        ServerAdmin admin@admin.com
        ServerName  localhost
        ServerAlias localhost
        WSGIScriptAlias / "C:/python/tfg/env/src/run.wsgi"
        DocumentRoot "C:/python/tfg/env"
        <Directory "C:/python/tfg/env/src/">
                Order deny,allow
                Allow from all
                Require all granted
        </Directory>
        ErrorLog "C:/python/tfg/env/src/logs/error.log"
        CustomLog "C:/python/tfg/env/src/logs/access.log" common
</VirtualHost>
```

Figura 56. Modificación del archivo *httpd-vhosts.conf*

Bibliografía

[1] **Sapec**. *Know How. El camino del vídeo.* Disponible en:

https://sapec.es/el-camino-del-video/

[2] **Sapec** (2015). *White Paper: Tecnologías de compresión de vídeo.* Disponible en:

https://sapec.es/wp-content/uploads/2015/06/white_paper_SAPEC_tecnologia_de_compresion.pdf

[3] **Ribelles García, Alex** (2016). *Digitalización, almacenamiento y transmisión de audio y video.* Fundació per a la Univertitat Oberta de Catalunya (FUOC).

[4] **Mata Díaz, Jorge** (2013). *Codificación de la señal de televisión*. Universitat Oberta de Catalunya (UOC).

[5] **Studio22-Enciclopedia** (2019). *MPEG-4 Part 10*. Disponible en:

https://www.studio-22.com/blog/enciclopedia/mpeg-4-parte10

[6] **López i Rocafiguera, E.; Barberán Agut, P.** (2019). *Introducción a las redes y servicios*. Fundació per a la Univertitat Oberta de Catalunya (FUOC).

[7] **Sheldon** (2021). *¿Cuál es la diferencia entre modelo OSI y modelo TCP/IP?* FS Community. Disponible en:

https://community.fs.com/es/article/tcpip-vs-osi-whats-the-difference-between-the-two-models.html

[8] **Kurose, J. F.; Ross, K. W.** (2017). *Redes de Computadoras. Un enfoque descendente.* Séptima Edición. Pearson Educación, S. A.

[9] **Ribelles García, A.** (2016). *Digitalización, almacenamiento y transmisión de audio y vídeo.* Fundació per a la Univertitat Oberta de Catalunya (FUOC).

[10] **Schulzrinne, H.; Rao, A.; Westerlund, M.; Stiemerling, M.** (2016). *RFC 7826. Real-Time Streaming Protocol Version 2.0*. Internet Engineering Task Force (IETF)

[11] **Wikipedia.** *Protocolo de Transmisión en Tiempo Real*. Disponible en:

https://es.wikipedia.org/wiki/Protocolo_de_transmisi%C3%B3n_en_tiempo_real

[12] **Haivision** (2018). *Technical Overview Secure Reliable Transport Protocol (DRAFT)*

[13] **The ATM Forum** (1999). *Audiovisual Multimedia Services: MPEG-2 Over ATM.* Disponible en:

https://docencia.ac.upc.edu/master/CBA-NGN/ATMB.pdf

[14] **GNS3 Documentation**. Disponible en:

https://docs.gns3.com/

[15] **Cisco.** Disponible en:

https://www.cisco.com/c/es_es/index.html

[16] **MenageEngine.** *Configuring Port Address Translation (PAT) on Cisco Devices*. Disponible en:

https://www.manageengine.com/network-configuration-manager/configlets/configuring-pat-cisco.html#:~:text=With%20Port%20Address%20Translation%20(PAT,NAT%20used%20in%20today's%20networks

[17] **Ffmpeg.** *Documentation.* Disponible en:

https://ffmpeg.org/documentation.html

[18] **SolarWinds Academy.** *How to configure NetFlow for Cisco Routers and switches Running IOS.* Disponible en:

https://www.youtube.com/watch?v=TZUW5lqzZDc

[19] **Cisco.** *GNS3 – Slow Internet.* Disponible en:

https://community.cisco.com/t5/other-network-architecture-subjects/gns3-slow-internet/m-p/3911657#M199380

[20] **Balmelli, S.** (2023). *SRT-Live-Transmit.* Disponible en:

https://github.com/Haivision/srt/blob/master/docs/apps/srt-live-transmit.md

[21] **Hernández E. A.** (2023). *Netmiko- La herramienta de Automatización para Dispositivos de Red.* Disponible en:

https://community.cisco.com/t5/blogs-general/01-netmiko-la-herramienta-de-automatizaci%C3%B3n-para-dispositivos-de/ba-p/4954431

[22] **PyPi.** *Netflow 0.12.2*. Disponibles en:

https://pypi.org/project/netflow/

[23] **Cisco** (2011). *NetFlow Version 9 Flow-Record Format.* Disponible en:

https://www.cisco.com/en/US/technologies/tk648/tk362/technologies_white_paper09186a00800a3db9.html

[24] **Madumal, T.** (2019). *Flask App Deployment in Windows (Apache-Server, mod_wsgi)*. Disponible en:

https://medium.com/@madumalt/flask-app-deployment-in-windows-apache-server-mod-wsgi-82e1cfeeb2ed

[25] **Gago, J.** (2013). *Digitalización de la señal de televisión*. Fundación para la Universitat Oberta de Catalunya (FUOC).

[26] **González C.** *SRT Streaming seguro*. Disponible en:

https://sonidoeiluminacion.com/srt-un-protocolo-abierto-y-seguro-para-streaming/