Información Importante

Evaluación de la transmisión de flujos de video codificados de acuerdo con el estándar HEVC/H.265

Sergio Andrés López Guerrero

Trabajo de grado para optar el título de Magister en Redes y Sistemas de Comunicación

Director

Wilder Eduardo Castellanos Hernández

Doctor en Telecomunicaciones

Codirector:

Tito Raúl Vargas Hernández

Magister en Tecnologías, Sistemas y Redes de Comunicación en la Especialidad de Ingeniería Telemática

Universidad Santo Tomas, Bucaramanga

División de Ingenierías y Arquitectura

Facultad de Ingeniería en Telecomunicaciones

2018

ELABORADO POR:	**REVISADO POR:**	**APROBADO POR:**
_____ *Sergio Andrés López Guerrero*	_____ *Wilder Eduardo Castellanos Hernández.* *Tito Raúl Vargas Hernández*	Comité de posgrado FIT

Dedicatoria

… A mis padres y hermanos.

Agradecimientos

El trabajo descrito en este artículo es el resultado de una cooperación académica en el marco de la Maestría en Redes y Sistemas de Comunicaciones de la Universidad Santo Tomás Bucaramanga, a través de convenios interinstitucionales que fortalecen el potencial académico-investigativo de los autores.

Agradezco a Dios por mostrarme este camino académico, el cual no fue fácil, pero con sus bendiciones, pude encontrar siempre las soluciones para superar los inconvenientes que se presentaron, y que al final me dieron las ideas y argumentos sólidos para sustentar el éxito en cada fase o tarea realizada. Es gracias a esto que hoy tengo más conocimiento y experiencia, que consecuentemente refleja una ganancia o amplitud en mi perfil profesional.

Agradezco ampliamente a mi director de proyecto, el Doctor Wilder Eduardo Castellanos Hernández, quien con su orientación y continuo soporte, me respaldó en la exploración y ejecución de esta investigación, y que sin lugar a dudas, este desarrollo no hubiera tenido el mismo éxito sin su dirección.

Así mismo, me permito agradecer a mi codirector, MSc Tito Raúl Vargas Hernández, quien siempre estuvo atento a emitir comentarios que apuntaban a orientar la investigación, manteniendo un camino o estructura investigativa para alcanzar los objetivos aquí propuestos.

Finalmente, y no menos importante, quiero agradecer orgullosamente a mi familia, mis padres, mis hermanos y mi novia, quienes siempre estuvieron conmigo brindándome apoyo y comprensión en los momentos de tensión y largas jornadas de dedicación, logrando obtener la claridad necesaria para alcanzar tan excelentes resultados.

Tabla de contenido

Lista de Tablas

Lista de Figuras

Lista de Apéndices

Resumen

El presente trabajo de investigación describe el estudio realizado para implementar una metodología, con el fin de evaluar las técnicas de transmisión de flujos de video codificados de acuerdo con el estándar HEVC/H265, y los experimentos llevados a cabo para la validación de dicha metodología.

La investigación inicialmente se basó en el estudio del estado del arte y antecedentes relacionados con la transmisión de flujos de video de ultra alta definición, en donde se logró identificar la necesidad de realizar estudios de video *streaming* con codificación HEVC. Igualmente, se desarrolló un análisis del estándar HEVC, para identificar sus principales características y sus aplicaciones. Para conocer y entender el proceso de *streaming* de video, se realizó un estudio preliminar sobre el proceso de transmisión de videos codificados con H.264. Esto con el objetivo de identificar cada una de las etapas del proceso, y conocer las herramientas, tanto para la transmisión de este tipo de contenidos, como para la evaluación de la *calidad de servicio (QoS)* y calidad del video recibido.

La metodología y plataforma de experimentación propuestas en este trabajo, involucran la integración de múltiples herramientas software para las etapas de codificación, encapsulamiento del video, simulación de la red y análisis de resultados. Una de las principales características de la plataforma planteada es la utilización del software MININET, el cual es un emulador de redes definidas por software que permite agregarle mayor realismo a las simulaciones de las redes que los simuladores de red tradicionales.

Con el fin de comprobar la utilidad de la plataforma planteada, se desarrollaron varios ejercicios de simulación en los cuales se realizó la transmisión de un video sobre diferentes

topologías de red. En dichos ejercicios fueron evaluados parámetros de calidad de servicio como el retardo, *throughput*, porcentaje de paquetes perdidos; y la calidad del video recibido en términos de PSNR y SSIM.

Palabras clave: HEVC, H265, UHD, video coding, video streaming, calidad de servicio, QoS, PSNR, SSIM.

Introducción

Los diferentes servicios de difusión de video digital, tales como los servicios de *streaming*[1] de video, han crecido exponencialmente en los últimos años y a su vez, han provocado un incremento significativo del tráfico multimedia en las redes de comunicaciones. Pero no solamente los servicios de video en "streaming" llaman la atención de los operadores de red. Con la llegada de las arquitecturas inalámbricas de nueva generación, también se abre la posibilidad de transmitir video de mayor calidad y por lo tanto, la implementación de servicios como IPTV, telecirugía [1] y la televisión inmersiva [2]. Uno de los aspectos más relevantes de este tipo de servicios, es el hecho de que el contenido que se transmite es altamente sensible a las variaciones de las condiciones de la red, lo cual supone un reto tecnológico importante, sobre todo si se desea proveer una buena calidad de experiencia (QoE, *Quality of Experience*) a los usuarios. Por otra parte, la alta demanda de los terminales con pantallas de muy alta resolución ha aumentado las expectativas de los usuarios, quienes ya no solo se conforman con visualizar el contenido multimedia, sino que lo desean con una muy alta calidad de imagen. Esto provocará que millones de personas en el mundo estén descargando flujos de video de ultra alta definición simultáneamente por las redes de comunicaciones. En este contexto, es importante el desarrollo y la evaluación de nuevas técnicas de compresión de video que hagan viable la transmisión de este tráfico.

En la actualidad, la popularidad de servicios multimedia emergentes tales como Youtube, Netflix, juegos en línea, videoconferencia y las transmisiones en vivo en las redes sociales (como Facebook Live, Instagram, entre otras), han desarrollado nuevos hábitos en el consumo del contenido audiovisual por parte de la sociedad, en donde las experiencias se comunican a

[1] *Streaming* se define como la técnica para la transmisión de contenido, en un flujo continuo de datos hacia un receptor, y reproducción continua en este último, sin la necesidad de descargar la totalidad del contenido.

través de video streaming, y el usuario demanda cada vez mejor calidad en la imagen, con altas resoluciones en sus videos, generando un fuerte incremento de tráfico en las redes.

En los estudios [3] y [4] se muestra que el tráfico de video en Internet crecerá en 79% y 75% para los años 2020 y 2022 respectivamente, en donde el 16% corresponde al consumo de video UHD *(Ultra High Definition)*. Sectores económicos como universidades, hoteles, transporte, y aquellos que ofrecen sus servicios a través de plataformas de streaming, se verán impactados por el crecimiento de este tipo de tráfico, razón por la cual se busca, mediante soluciones TIC, el uso eficiente de las redes de comunicaciones, ofreciendo al usuario un servicio con excelente calidad en la imagen. El aumento significativo en el uso de terminales UHD, ha provocado un incremento en la expectativa de los usuarios en cuanto a calidad de video, quienes día tras día demandan cada vez más contenido en UHD, traduciéndose esto en una sobrecarga importante en las redes de comunicaciones. En este contexto cobra importancia, el desarrollo de un trabajo investigativo en el área de los sistemas de video de ultra alta definición, teniendo en cuenta las características inherentes a este tipo de tráfico y la evaluación de su impacto sobre las redes, tal como es el objetivo del presente proyecto.

Para abordar los problemas mencionados anteriormente, este estudio se inició con la evaluación preliminar de la transmisión de video, consultando las herramientas existentes para tal fin, encontrando posteriormente que no existen herramientas disponibles para la evaluación de la transmisión de video de ultra alta definición. Una de las soluciones que se propone en este trabajo, es el uso de un emulador de redes definidas por software (SDN, Software Defined Networks). Las SDN hoy representan un nuevo paradigma para las redes de telecomunicaciones, ya que suponen reemplazar los dispositivos de red como conmutadores y enrutadores, por módulos software orquestados por un controlador, quien toma las decisiones

de reenvío de paquetes sobre la red. Esto refleja un impacto significativo en ahorro de despliegue y mantenimiento de red (Capex y Opex, respectivamente).

En el presente trabajo se describe una plataforma de modelado y simulación basada en redes definidas por software que permite la evaluación de la transmisión de video de ultra alta definición. Esta plataforma consiste en la integración de herramientas de codificación, empaquetado, streaming, decodificación y evaluación, junto al emulador de red MININET, para evaluar la transmisión de video de ultra alta definición. Además, se describe la metodología de uso de la plataforma, así como los resultados obtenidos de las evaluaciones realizadas en la transmisión de flujos de video codificados con HEVC/H.265. Estos resultados muestran como las condiciones de red, en términos de retardos y pérdida de paquetes, pueden afectar la calidad del video recibido. La metodología propuesta involucra el uso de scripts que automatizan la ejecución de las etapas de codificación, empaquetado, streaming, decodificación y evaluación de calidad de servicio y calidad del video recibido.

El trabajo desarrollado implicó una metodología propia de una investigación aplicada con enfoque experimental y cuantitativo. Carácter experimental dado que se realizaron diferentes experimentos o simulaciones aleatorias, buscando evaluar el impacto en el desempeño de la transmisión de flujos de video de ultra alta definición, al modificar parámetros de red como retardo y pérdida de paquetes. Se realizó un análisis cuantitativo sobre los resultados en los parámetros de calidad de imagen (PSNR y SSIM).

El presente documento está organizado de la siguiente manera: en el capítulo 1 se resumen las principales generalidades del proyecto, en el capítulo 2 se describe la evaluación de la transmisión de video en términos de antecedentes y pruebas preliminares. El capítulo 3 presenta de forma breve los conceptos fundamentales de las *redes definidas por software (SDN),*

protocolo OpenFlow y las características principales de la herramienta MININET, la cual se definió como la herramienta a utilizar para las simulaciones propuestas en este estudio. Posteriormente, en el capítulo 4 se define la metodología para evaluar la transmisión de flujos de video codificados con el estándar HEVC/H.265 sobre redes de comunicaciones, a través de un emulador de red (MININET) bajo diferentes condiciones de red. El capítulo 5 contiene los resultados de la evaluación de parámetros de calidad de servicio y la calidad de los videos recibidos en diferentes topologías de red. Al finalizar este documento, se presentan algunas conclusiones y sugerencias para trabajo futuro, que invitan a la comunidad académica a continuar estudios enfocados a la optimización de las redes de transporte para la transmisión de contenidos de videos de ultra alta definición.

La importancia de las contribuciones de este trabajo radica en el estudio del estado del arte de la evaluación de la transmisión de video de ultra alta definición, así como también la definición de una metodología e integración de herramientas en una sola plataforma para la evaluación extremo a extremo de la transmisión de este tipo de contenidos.

1 Generalidades del proyecto

1.1 Planteamiento del Problema

Para responder a la necesidad de transmitir contenidos de video con excelente calidad en la imagen, se ha incrementado la implementación de terminales y aplicaciones que utilizan el formato UHD, lo cual llevará a la masificación de dicho formato de video, y su transmisión demandará un mayor consumo de los recursos de red. Por ejemplo, algunos de los parámetros que se verán impactados serán: el ancho de banda, el tiempo de procesamiento, la capacidad de almacenamiento, entre otros. Adicionalmente, teniendo en cuenta el nivel de exigencia en la calidad de imagen por parte del usuario, quien cada vez demanda altas resoluciones en sus videos, y una buena calidad en la reproducción de los mismos, se requiere contar con herramientas y metodologías de evaluación que permitan analizar el impacto que pueden ocasionar factores como el retardo y la pérdida de paquetes sobre la calidad del video durante la transmisión de este tráfico. La evaluación de estos factores corresponde a un tema de creciente interés a nivel internacional. Sin embargo, no existen las herramientas adecuadas que permitan integrar los procesos de codificación junto con el análisis del tráfico de video cuando este se transmite sobre las redes de comunicaciones. Además, los estudios realizados hasta el momento no contemplan escenarios realistas donde las pérdidas de paquetes se originen tanto por las limitaciones de las aplicaciones de red, como por las interferencias transitorias.

Para responder los problemas planteados anteriormente durante la transmisión de contenido de video, este proyecto tiene como objeto evaluar la transmisión de flujos de video con codificación HEVC/H.265 sobre redes de comunicaciones, a través de herramientas de

modelado y simulación, con el fin de determinar las principales ventajas de dicha codificación y su comportamiento sobre diferentes condiciones de red.

Con el estudio del comportamiento de los flujos de video bajo condiciones variables de red, se promueve el desarrollo de técnicas de optimización en la transmisión de este tipo de tráfico, que garanticen y mantengan aceptables los niveles de calidad de servicio y calidad del video recibido.

1.2 Formulación del Problema

¿Cómo simular el comportamiento de flujos de tráfico de video con codificación HEVC/H.265 bajo diferentes condiciones de red?

1.3 Sistematización del Problema

¿En qué consiste la codificación HEVC/H.265 y cuáles son sus ventajas?

¿Qué herramientas existen actualmente para simular la transmisión de flujos HEVC/H.265?

¿Cuál es la herramienta de simulación adecuada para evaluar la transmisión de flujos HEVC/H.265?

¿Cuáles métricas deben ser evaluadas para cuantificar la calidad de videos transmitidos y el impacto sobre el tráfico de las redes?

¿Cuál es la metodología que se debe utilizar para simular la transmisión de video sobre redes de comunicaciones y cómo se deben interpretar los resultados de dichas simulaciones?

1.4 Objetivo general

Diseñar una metodología para evaluar la transmisión de flujos de video codificados con el estándar HEVC/H.265 sobre redes de comunicaciones, a través de herramientas de modelado y simulación bajo diferentes condiciones de red.

1.5 Objetivos específicos

- Identificar las principales características de la codificación de video en ultra alta definición, así como sus principales aplicaciones, para determinar los requerimientos técnicos necesarios en la transmisión de dicha información.
- Determinar las principales herramientas software utilizadas en la simulación de la transmisión de video, para verificar su posible uso en el estudio del video de ultra alta definición.
- Evaluar la calidad de los videos de ultra alta definición durante la transmisión sobre diferentes escenarios de red, mediante la implementación y ejecución de simulaciones exhaustivas, calculando parámetros de QoS y de desempeño de red.

2 Evaluación de la transmisión de video: antecedentes y pruebas preliminares

2.1 Antecedentes y herramientas para la transmisión de video

Como parte de las pruebas y consultas previas en esta investigación, se tienen los antecedentes detallados en el apéndice A (anteproyecto). Sin embargo, a continuación se describen los trabajos más relevantes, debido principalmente a que presentan herramientas para la transmisión y evaluación de flujos de video codificado con el estándar HEVC/H265.

En [5] se describe un estudio sobre las barreras prácticas del estándar HEVC en ambientes reales, y se propuso un *framework* de evaluación para contenido codificado con HEVC, a este framework se le conoce como HEVStream. Permite la evaluación y pruebas de flujos de video codificado con HEVC bajo un rango de pérdida de paquetes, restricciones de ancho de banda y retardos de red. Los resultados mostraron una percepción de reducción en la calidad de la imagen, medida bajo PSNR, que puede ser esperado bajo un rango amplio de limitaciones de red y condiciones de pérdida de paquetes. No hay una versión pública disponible de esta herramienta.

En [6] se evaluó el desempeño de la transmisión de contenido HEVC sobre ambientes de red simulados, para áreas afectadas por desastres y calamidades. La simulación de esta transmisión estuvo sujeta a varios modelos de error en el simulador NS3 (Network Simulator 3). Se evaluaron los efectos de la velocidad y número de hosts en el jitter y el retardo característicos de una red bajo capas, mientras se transmiten flujos de HEVC basados en contenidos. Se midieron los efectos de los errores de red en la calidad de los flujos de video HEVC en términos de PSNR. Los resultados mostraron que HEVC tiene mejor desempeño hasta con 0.001% de

errores de red, y hasta 30 nodos transmitiendo simultáneamente, y con velocidades hasta de 100 m/s.

Por otra parte, en [7] se propuso un framework altamente flexible para transmitir videos usando mecanismos de transporte poco confiables, combinando MPEG-2 TS, RTP y UDP. En este trabajo, también fueron simuladas diferentes condiciones de red tales como pérdidas de paquetes, retardo y Jitter, tanto en la transmisión como en offline, a través de la implementación de varios modelos ajustables de distribuciones de probabilidad de error tales como el modelo de Gilbert-Elliot. En [7] se menciona la herramienta *Sirannon* como software para la simulación de streaming en redes, sin embargo, no ha sido actualizada desde varios años, y por lo tanto no soporta el estándar HEVC.

En [8] fue desarrollada la herramienta HEVC-SIM para la codificación y transmisión de video, basada en implementaciones de código abierto de codificadores y decodificadores HEVC. Esta herramienta utiliza dos métodos de cubrimiento de error para ayudar al decodificador a recuperar el video luego de pérdida de paquetes, es decir, la simulación de la pérdida de paquetes se realiza mediante dos métodos: aleatoriamente usando Gilbert-Elliott (cadena de Markov de dos estados), o por sustracción manual de paquetes.

Finalmente, en [9] se propone un set de configuración de herramientas para evaluar streaming de HEVC, basados en el uso de herramientas como HM (H.265 Reference Software) para la codificación/decodificación del video, y herramientas de NCTuns para la generación de tráfico (EvalSVC).

2.2 Herramientas para evaluar la transmisión de video H264

Después de consultar algunos antecedentes y las características propias del estándar, se realizaron pruebas de transmisión de video codificado en H264, con el objetivo de conocer la metodología para la simulación de la transmisión de video usando las herramientas existentes, explorando sus características y funcionalidades. Los resultados y conclusiones de estas pruebas son mencionados en el apéndice B (artículo), en donde se detallan los parámetros de la simulación, características del video transmitido y las herramientas utilizadas en cada una de las etapas del proceso. Esta fase del proyecto permitió adquirir habilidades en la codificación de video, la simulación de streaming, captura y manipulación de trazas, y la evaluación tanto de la calidad del video como de parámetros de QoS. Otro de los aportes de este estudio preliminar, fue la identificación y utilización de herramientas ampliamente utilizadas en investigaciones sobre video streaming, tal como FFMPEG [10], GPAC-MP4BOX [11], Evalvid [12] y la integración de estas con el simulador de redes NS-2 [13]. La interacción de estas herramientas ha sido ampliamente utilizada en la construcción de plataformas de estudio de diferentes servicios de video streaming, tal como se evidencia en los trabajos descritos en las referencias [14], [15], [16], [17].

La metodología de uso de este tipo de herramientas se resume en la Figura 1. Esta metodología consiste básicamente en codificar el video, luego se obtiene una traza que caracteriza el video (mediante la herramienta *Evalvid-mp4trace*), dicha traza posteriormente se pasa a la herramienta de simulación, donde se hace la transmisión y recepción. Al finalizar la simulación, se tiene una etapa de reconstrucción del video a partir de las trazas de envío, recepción y la traza original del video (mediante la herramienta *Evalvid-etmp4*). Una vez se reconstruye el video en lado del receptor, este se decodifica para posteriormente realizar la

evaluación de la calidad del video recibido, a partir de herramientas como "*psnr.exe*" de Evalvid.

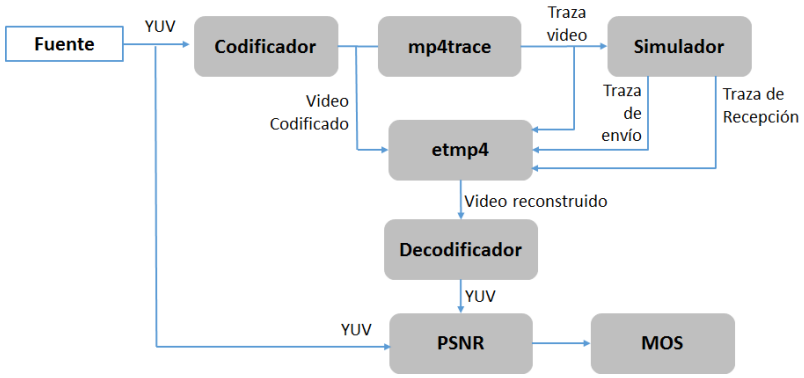


Figura 1. Metodología para la evaluación de la transmisión de video.

El conjunto de herramientas descrito anteriormente fue inicialmente pensado como la plataforma idónea para realizar las adaptaciones necesarias para establecer, no solo estudios de transmisión de video H.264, sino que además permitiera experimentar la transmisión de videos codificados con HEVC/H265. Sin embargo, no fue posible la adaptación de todos los algoritmos que componen la herramienta llamada *Evalvid-mp4trace*, la cual se encarga de generar un archivo de texto con la descripción del video para que el simulador de redes pueda generar el tráfico por la red. En la Figura 2 se muestra una captura de pantalla del error generado al intentar procesar un video H.265 con *mp4trace*.

```
salopezgu@proyectomrsc:~/Simulaciones/Vidyo1/TXvidyo1/H265$ mp4trace -f -s 192.168.1.2 12346 vidyo1_hevc.mp4 > ns2trace.tr
Track 1: Video (Unsupported)     - 1280x720 pixel, 100 samples, 00:00:01.666
Track 2: Hint (RTP) for track 1  - 100 samples, 00:00:01.666
salopezgu@proyectomrsc:~/Simulaciones/Vidyo1/TXvidyo1/H265$
```

Figura 2. Error de mp4trace al enviar un video H265.

La traza que caracteriza el video, obtenida en el paso anterior, no fue correcta, por lo tanto, el proceso de transmisión de video HEVC/H265 con este set de herramientas no fue exitoso. A partir de lo anterior, fue necesario consultar y probar otras herramientas encontradas a partir del análisis del estado del arte.

2.3 StreamSIM: plataforma para la simulación de la transmisión de video HEVC/H265

StreamSIM es una herramienta que utiliza un proceso en cadena para simular el *streaming* de video con varios codecs, implementaciones de codificadores y decodificadores, configuraciones de red y protocolos de transmisión [7]. La metodología en cadena por etapas utilizada por esta herramienta se muestra en la Figura 3.

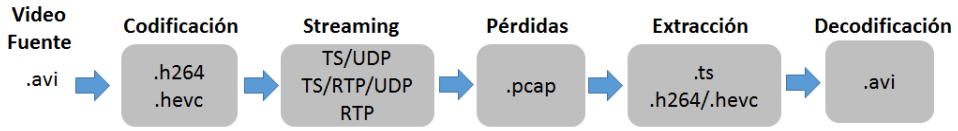


Figura 3. Pasos de procesamiento en StreaSim y sus formatos de archivos.

Adaptado de [7].

La documentación completa para su instalación y pruebas se encuentra disponible en https://gitlab.com/vqeg/streamsim [7].

La herramienta consta de una serie de carpetas en donde se alojan los videos fuentes, archivos de configuración, logs y los resultados de cada etapa, como lo muestra la Figura 4.

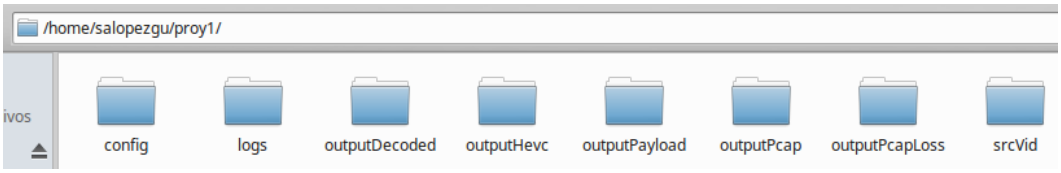


Figura 4. Estructura de carpetas StreamSim.

La carpeta *config* contiene todo el árbol de configuración de la cadena de procesos, y su estructura se muestra en la Figura 5.

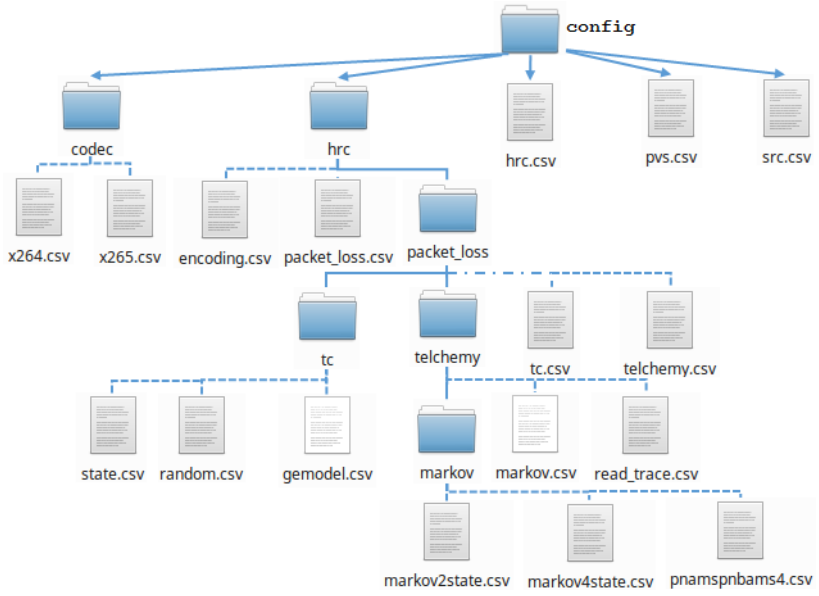


Figura 5. Árbol de archivos de configuración de StreamSim.

El archivo *"src.csv"* contiene la identificación de los videos *.avi* a transmitir. La carpeta *HRC (Hypothetical Reference Circuits)* contiene los archivos donde se establece los parámetros utilizados para la codificación del video y el modelo de pérdidas de paquetes. Estos parámetros son leídos desde los archivos "*encoding.csv"* y "*packetloss.csv"*. El archivo "*pvs.csv" (Processed Video Sequences)* relaciona las configuraciones descritas en el archivo *"hrc.csv"* con los videos registrados en el archivo *"src.csv"*. La herramienta permite configurar códec x264 y x265, los cuales son leídos en el archivo *"encoding.csv"*. En la etapa de pérdida de paquetes se utilizan dos herramientas, para el caso de modo online, la herramienta *TC (Traffic Control)* permite utilizar modelos de distribución de pérdidas como *Gilbert-Elliot* y cadenas de *Markov*. Para el caso de modo offline, la herramienta *Telchemy* [18] permite insertar pérdidas de acuerdo con un modelo de Markov o mediante la lectura de una traza de pérdidas definida por el usuario en el archivo *"read_trace.csv"*. Cada una de estas opciones de pérdidas de paquetes es configurada en sus archivos correspondientes (ver Figura 5.), por ejemplo, en el archivo *"markov2state.csv"* se definen los parámetros que describen un modelo de Markov de

dos estados, que permite insertar pérdidas según las probabilidades de transición de estados y probabilidades de pérdidas, como se describe en [18].

2.3.1 Características de StreamSim.

Video streaming simulation toolchain (StreamSIM) es una herramienta implementada sobre Linux y programada en *Python*. Su funcionamiento consiste en la ejecución de un script llamado "*chain.py"*, en el cual se ejecutan los 5 pasos mostrados en la Figura 3. En la Figura 6 se muestra un fragmento del código *Python* del script *"chain,py"*, el cual hace el llamado a las funciones que ejecutan los 5 pasos del proceso. El código fuente completo se muestra en el apéndice D.

chain.py

```
"""
This application transmits a collection of raw video material through configurable networks and stores the
obtained results.
"""

__author__ = 'Alexander Dethof'

from chainApp.processingChain import ProcessingChain

# True if the error warnings should be displayed more detailed; False if the error messages should be just shown and
# the application should exit by displaying it. By default the debug mode is set to false!
DEBUG_MODE = False


def __execute_chain():
    """
    Creates a new processing chain and executes it
    """

    processing_chain = ProcessingChain()

    try:
        from atexit import register
        register(processing_chain.cleanup)

        processing_chain.execute()
    except KeyboardInterrupt:
```

Figura 6. Fragmento del código fuente en python del script "chain.py".

Es importante mencionar que el script *"chain.py"* no configura la simulación, este sólo hace el llamado a las funciones que ejecutan los 5 pasos del proceso. La simulación se configura en cada una de las carpetas y archivos mostrados en la Figura 5, y explicados en la sección anterior.

StreamSim tiene tres modos de ejecución: el modo *full* permite ejecutar todos los pasos de la cadena continuamente hasta un paso de parada definido por el usuario; el modo *single* permite ejecutar un solo paso de la cadena, y el modo *continuo* permite definir el paso desde el cual se inicia el proceso hasta el paso final. La herramienta tiene la habilidad de codificar en H264 y

H265, enviar los *streams* de video sobre diferentes ambientes de red, soporta tres opciones de protocolos para realizar el *streaming*: MPEGTS/UDP/IP, MPEG-TS/RTP/UDP/IP y RTP/IP, y permite insertar modelos de pérdidas de paquetes (Markov) cuando se utiliza en modo offline, simulando el *streaming* de video a través de la dirección de *localhost*. En modo online (redes establecidas físicamente), se define la dirección destino del host receptor, y se utiliza la herramienta TC/NETEM *(Traffic control)* [19] para administrar el envío de los paquetes. StreamSim utiliza las siguientes herramientas para cada paso del proceso, y su modo de funcionamiento es descrito en la siguiente sección: FFMPEG [10] para la codificación, decodificación y streaming del video codificado, TCPdump [20] para la captura de trazas, TC/NETEM *(Traffic control)* [19] para administrar el envío de paquetes en modo online, y la extracción del *payload* o video recibido se realiza mediante la herramienta *scapy* [21].

2.3.2 Análisis del funcionamiento de StreamSim.

Con el fin de evaluar el funcionamiento de la plataforma StreamSim, se simuló la transmisión de un video en el modo offline, es decir, haciendo un loop hacia la dirección de *localhost*. El video utilizado en este experimento fue la secuencia de prueba conocida como "*akiyo"* [22]. Este video está en formato YUV, es decir, sin compresión y está compuesto por 300 fotogramas con tamaño de 352x288 píxeles. A continuación, se describen los pasos realizados durante la ejecución de la simulación y la obtención de los resultados:

1. Codificación: El codificador utilizado fue FFMPEG [10], el códec x265, con un *bitrate* de 2500 kbps. El siguiente comando ejecuta la etapa en mención, y sus resultados se muestran en la Figura 7.

```
sudo python chain.py encode_videos -s -p /home/salopezgu/proy1/ -l -y
```

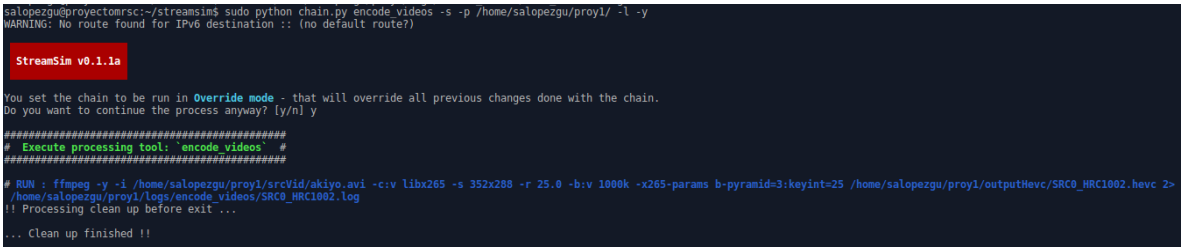


Figura 7. Etapa de codificación en StreamSim.

El comando mencionado anteriormente muestra el llamado del script *"chain.py"* para que ejecute solamente el paso denominado *encode_videos*, el cual corresponde a la codificación de los videos. La opción *–s* corresponde al modo de ejecución sencillo, en el cual se ejecuta solamente el paso indicado en el comando. La opción *–p* direcciona el *path* o ruta en donde se encuentran la carpeta de configuración *config* y las carpetas de resultados y logs de cada paso del proceso. La opción *–l* permite guardar un log de la ejecución de las herramientas propias del paso en ejecución. La opción *–y* permite sobrescribir los resultados del paso en ejecución, sobre resultados ya existentes.

2. Streaming: Como se mencionó anteriormente, el streaming se realizó en el modo offline, es decir, haciendo un loop hacia la dirección de localhost, utilizando la herramienta FFMPEG [10]. Para este experimento se utilizó el formato MPEGTS/UDP/IP debido a que es uno de los más utilizados en la actualidad. El tráfico recibido es capturado con la herramienta TCPdump [20], la cual permite obtener trazas del tráfico de paquetes, en un archivo ".pcap" como lo muestra la Figura 8. Este archivo es utilizado en la siguiente etapa (inserción de pérdidas).

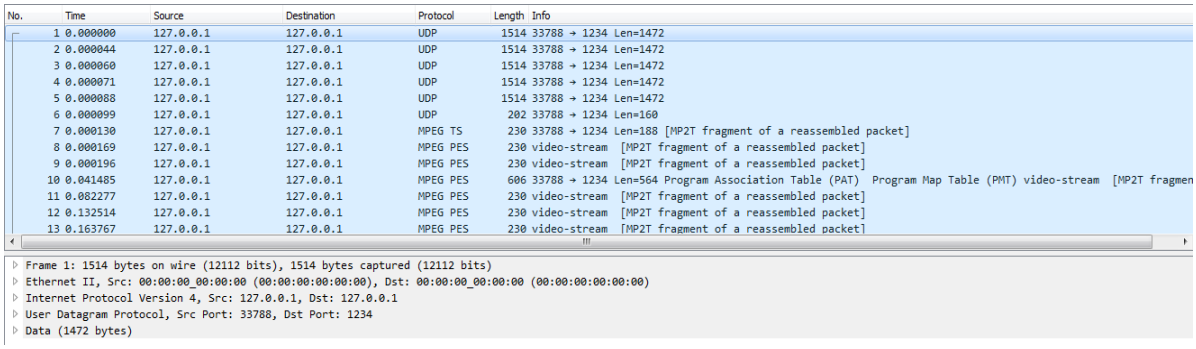


Figura 8. Fragmento de la traza *.pcap* generada en el paso *streaming*.

El siguiente comando ejecuta la etapa en mención, y sus resultados se muestran en la Figura 9. A diferencia del anterior, aquí se escribe la opción *stream_video*, la cual corresponde al paso en ejecución.

```
sudo python chain.py stream_videos -s -p /home/salopezgu/proy1/ -l -y
```

Figura 9. Etapa de streaming en StreamSim.

En la Figura 9 se aprecian los comandos que se ejecutan en esta etapa, en los cuales se evidencia que la captura de trazas con TCPdump se realiza en la interfaz *lo* (localhost), y el *streaming* de FFMPEG se realiza hacia la dirección de *localhost* (127.0.0.1). El resultado de esta etapa corresponde al video transmitido con extensión *.ts*, la traza *.pcap*, y los logs de FFMPEG y TCPdump. Estos archivos se almacenan en las respectivas carpetas de resultados, como se muestra en el apéndice E.

3. Inserción de pérdidas: Dado que el experimento realizado fue en modo offline, en esta etapa se utilizó la herramienta *Telchemy* [18] para la inserción de pérdida de paquetes. Las pérdidas insertadas se generaron de acuerdo a un modelo de Markov de dos estados, en el

periodo comprendido desde 1 segundo después de iniciar el *streaming* hasta 1 segundo antes de finalizarlo. El siguiente comando ejecuta la etapa en mención, y sus resultados se muestran en la Figura 10.

```
sudo python chain.py insert_loss -s -p /home/salopezgu/proy1/ -l -y
```

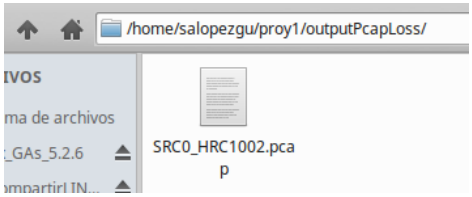


Figura 10. Traza .pcap resultante luego de la etapa de inserción de pérdidas.

4. Extracción de *payload:* A partir de la traza *.pcap* resultante en la etapa de inserción de pérdidas, se utiliza la herramienta *scapy* [21] para extraer el *payload* de la traza y reconstruir el video recibido. Dado que el formato utilizado en el streaming fue MPEG-2 TS *(.ts),* el payload o video reconstruido es almacenado en este mismo formato. El siguiente comando ejecuta la etapa en mención, y sus resultados se muestran en la Figura 11.

```
sudo python chain.py extract_payload -s -p /home/salopezgu/proy1/ -l -y
```

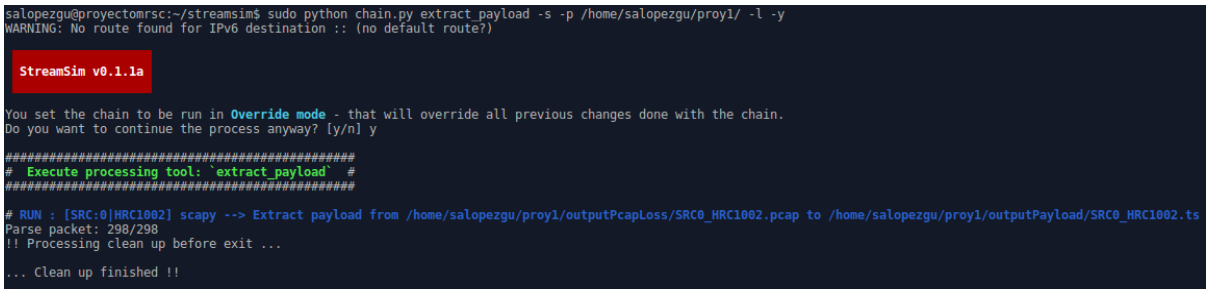


Figura 11. Extracción de payload con Scapy en StreamSim

5. Decodificación: Se decodifica el video recibido en un contenedor .avi, utilizando FFMPEG y el mismo códec usado en la etapa de codificación. El siguiente comando ejecuta la etapa en mención, y sus resultados se muestran en la Figura 12.

```
sudo python chain.py decode_loss -s -p /home/salopezgu/proy1/ -l -y
```

Figura 12. Decodificación en StreamSim.

En este punto finaliza el uso de la herramienta StreamSim, con lo cual se puede apreciar que sólo permite realizar sreaming de video y reconstruir un video recibido, pero no cuenta con herramientas adicionales para evaluar la calidad del video reconstruido, por ejemplo a través de métricas como el PSNR, lo cual implica usar herramientas adicionales como Evalvid para generar estos resultados. Adicionalmente, no contempla los cálculos de parámetros de calidad de servicio como *throughput*, *delay*, *jitter*, entre otros, en los cuales se analicen las trazas de envío y recepción de paquetes. Al tratarse de una simulación de streaming hacia la dirección de *localhost*, solo se tiene una traza capturada por TCPdump, a la cual posteriormente se le insertan pérdidas de paquetes, generando una nueva traza, por lo tanto en este punto debería existir una etapa de cálculo de parámetros de calidad de servicio entre las dos trazas mencionadas anteriormente.

2.4 Conclusiones

Una vez revisado el estudio de antecedentes y herramientas para la transmisión de video, y luego de la prueba realizada con la herramienta StreamSim para la transmisión de video HEVC/H265, se tienen las siguientes conclusiones:

1. No fue posible la adaptación de todos los algoritmos que componen la herramienta llamada *Evalvid-mp4trace* para lograr transmitir y evaluar video codificado con HEVC/H265.

2. La herramienta StreamSim tiene la ventaja de tener carpetas y archivos de configuración por separado, que permite relacionar diferentes configuraciones registradas en el archivo *"hrc.csv"* con los videos registrados en el archivo *"src.csv"*, obteniendo varias transmisiones en un solo lanzamiento.

3. StreamSim, en el modo offline, sólo permite simular transmisión de video hacia la dirección de *localhost*, con lo cual no se pueden tener dos extremos de transmisión (emisor y receptor) en los cuales se capturen trazas que permitan realizar cálculos de calidad de servicio QoS como retardos, porcentaje de paquetes perdidos, *throughput*, entre otros.

4. Igualmente, en el modo offline, no se puede configurar diferentes topologías de red que permitan realizar estudios de transmisión de video con diferentes condiciones de red.

5. Las pérdidas son introducidas de manera manual y no son consecuencia de un proceso aleatorio propio de la interacción de los diferentes elementos de red, tales como movilidad, congestión en los nodos, capacidad de procesamiento de los nodos, el modelo de propagación, la interacción de los diferentes componentes software y los fenómenos físicos propios de la propagación de las señales, entre otros.

6. Otros parámetros de red, así como aspectos propios de los protocolos y/o tecnologías de red no pueden ser introducidos en las simulaciones con StreamSim. Las razones mencionadas anteriormente, reflejan falta de realismo en las simulaciones.

Teniendo en cuenta los puntos mencionados anteriormente, fue necesario explorar nuevas alternativas que permitan resolver algunas de las desventajas presentadas en StreamSim. En el capítulo 4 se presenta la alternativa propuesta, la cual está basada en el uso del software de emulación de redes definidas por software MININET, la cual si permite tener un ambiente completo de red *end to end* para el análisis de QoS. Por lo tanto, en el siguiente capítulo, y antes

de describir la plataforma propuesta, se va a presentar una breve descripción de los conceptos fundamentales de las redes definidas por software (SDN, *Software Defined Networks*), aspectos básicos del protocolo *openflow* y los fundamentos de uso de la herramienta MININET.

3 Redes Definidas Por Software: Openflow Y Mininet

En este capítulo se describen brevemente los conceptos fundamentales de las redes definidas por software (SDN, *Software Defined Networks*), así como los aspectos básicos del protocolo openflow y los fundamentos de uso de la herramienta MININET. Estos conceptos se relacionan con el desarrollo de este proyecto, debido a que la metodología propuesta para la evaluación de la transmisión de video HEVC, incluye el uso de la herramienta MININET como emulador de redes definidas por software, en la cual se definirán las topologías de red para el desarrollo de los estudios propuestos.

3.1 Redes definidas por software (SDN, Software Defined Networks)

Las redes de transporte se ubican en el centro de las grandes tendencias tecnológicas como la nube, la movilidad y el video. Para lograr la implementación de estas nuevas tecnologías, se requiere una evolución de la arquitectura de red. Las redes definidas por software se han convertido en una de las principales novedades para conseguir esta evolución en la red [23].

SDN representa un nuevo paradigma capaz de volver las redes más eficientes, escalables, ágiles y dinámicas, mediante el aumento de programación y automatización [24]. De acuerdo con el análisis de la firma *Frost & Sullivan*, "*Software Defined Networking: en busca de la automatización de la red"*, los principales beneficios de esta tecnología en la infraestructura son cuatro: crear un punto de control único en la red, permitir mayor agilidad y rápido aprovisionamiento, reducir los costos (Apex y Opex), ofrecer un ambiente programable [23].

SDN está captando la atención de investigadores en el campo académico y en la industria, con el objetivo de producir soluciones que generen reducción de costos de red y aumento de ingresos en los proveedores de servicio gracias a las funcionalidades que esta tecnología ofrece en *networking* [25]. El paradigma SDN plantea retos multidimensionales en el ámbito técnico, financieros y de negocios: Desafíos técnicos de escalabilidad, tolerancia a fallos, centralización de garantías y seguridad (al tener un solo controlador, esto puede ser atractivo para ataques de red); los desafíos financieros corresponden a las restricciones presupuestales, la no disponibilidad de un modelo de transición por etapas; y los desafíos empresariales como la aceptabilidad y la construcción de confianza entre los operadores de red [26].

Para entender el concepto de las Redes Definidas por Software, es preciso recordar las tareas que realizan los enrutadores en las redes de comunicaciones:

- En primer lugar, los enrutadores deben conocer y actualizar la topología de red en cada instante, almacenando información de enrutamiento como las rutas que deben seguir los paquetes transmitidos. Esto se realiza en el plano de control [27].
- En segundo lugar, los enrutadores, teniendo en cuenta las decisiones en el plano de control en cuanto a las rutas de reenvío, se encargan de reenviar paquetes desde un origen a un destino. Esto se realiza en el plano de datos [27].

Teniendo en cuenta lo anterior, las SDNs surgen de las funcionalidades programables de la red y la separación de los planos de control y datos en los dispositivos de red. El principio fundamental del funcionamiento de las SDN consiste en trasladar el plano de control de los dispositivos de red (conmutadores y enrutadores), a un solo dispositivo externo a la red que cumpla dichas funciones, y este es el controlador. El controlador es el elemento software que toma las decisiones de reenvío de paquetes y creación de las rutas de reenvío para los dispositivo

de red, logrando de esta manera configurar todo tipo de dispositivo como conmutadores, enrutadores, traductores NAT, o *firewalls* [27].

SDN ha surgido de la necesidad de superar las deficiencias de las redes actuales en cuanto a escalabilidad, reduciendo entonces el hardware necesario para su montaje (disminución de *Capex – CAPital EXpenditures*), facilitando la gestión de los elementos de red, disminuyendo el tiempo de administración de la red (disminución de *Opex - OPerational EXpenditures),* agilizando el despliegue de aplicaciones y servicios sobre una red unificada bajo un controlador [28].

La implementación de SDN tiene la ventaja de tener una visión completa de la red, toda vez que permite intercambiar información entre las capas de la arquitectura: aplicación, control e infraestructura, como lo muestra la Figura 13.

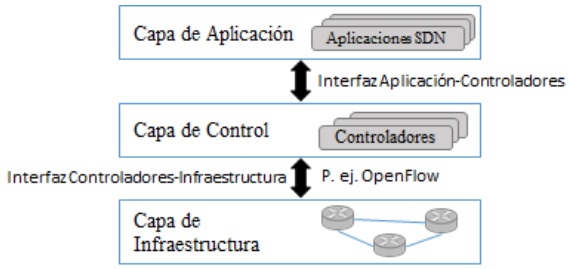


Figura 13. Modelo de arquitectura SDN.

Adaptado de [27].

Dentro de la capa de aplicación de SDN se tienen funciones como enrutamiento adaptativo, itinerancia sin interrupciones, seguridad y mantenimiento de la red, virtualización de red y *cloud computing*. En la arquitectura SDN, el controlador es el elemento más importante (capa de control), el cual gestiona la capa de infraestructura y la capa de aplicación mediante interfaces. Desde la capa de infraestructura, se recibe el estado de la red, y se configura de acuerdo con las necesidades de la capa de aplicación. Finalmente, la capa de infraestructura consta de los dispositivos de conmutación que tienen dos funciones (ver Figura 14): control (topología o

estadísticas de tráfico) que obtiene el estado de la red para comunicarlo al controlador y recibir de él, las rutas de reenvío; la función de datos que consiste en reenviar paquetes según las decisiones en el plano de control [27].

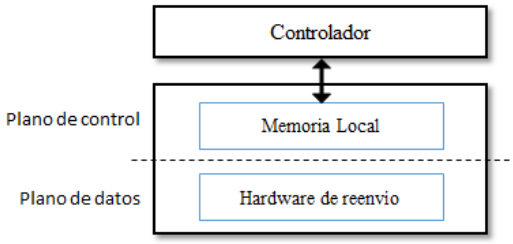


Figura 14. Dispositivo de red en la capa de infraestructura.

Adaptado de [27].

3.2 OpenFlow

OpenFlow es un protocolo de estándar abierto utilizado para la gestión de las redes SDN, que centraliza las decisiones de los dispositivos de red, permitiendo su programación y funcionamiento [28].

Un dispositivo OpenFlow dispone de tres partes:

- Tabla de flujos: Cada entrada de la tabla dispone de campos con información de paquetes entrantes, contadores (estadísticas), e instrucciones [27].
- Canal seguro: Conecta el dispositivo al controlador, permitiendo el envío de comandos y paquetes a través de protocolo OpenFlow [27].
- Protocolo OpenFlow: Protocolo estándar abierto para comunicación entre controlador y conmutador, permitiendo al controlador insertar, eliminar, modificar y buscar entradas en las tablas de flujo [27].

3.3 MININET

MININET es un emulador de redes conformado por dispositivos de red tales como host, conmutadores, enlaces y controladores, unificados en un sistema LINUX, y permite experimentar con SDN y OpenFlow. Es de fácil manejo, ya que a través de la línea de comandos (CLI) se puede interactuar con los dispositivos de red y realizar pruebas de desempeño de red [27].

Algunas características importantes de MININET son:

- Sencillez, debido a que cuenta con topología sencilla programada para desplegar y hacer pruebas [27].
- Se pueden crear diferentes tipos de topologías a través de lenguaje Python, para simular distintos escenarios de red reales, evitando costos de despliegue real de red [27].
- Se emula el envío de paquetes a través de interfaces Ethernet, con velocidad de enlace y retardo [27].
- Cuenta con herramientas de desempeño como ping y tcpdump, iperf para realizar pruebas de red entre host [27].
- El envío de paquetes se puede personalizar a través de la programación con protocolo OpenFlow [29].
- MININET es un proyecto de open source, es decir, se puede examinar su código fuente y reportar errores, funcionalidades, parches, entre otros [29].
- Capacidad para simular redes inalámbricas a través de la instalación de un paquete adicional (mininet-wifi), el cual, además, permite emular el movimiento de los nodos y la propagación de la señal [28].

- MININET presenta una ventaja importante comparada con otros emuladores y simuladores, permite emular la red con dispositivos reales. Los dispositivos que emula, generan tráfico y permiten la ejecución de código real para realizar experimentos y pruebas en escenarios reales [28].
- MININET funciona sobre Linux, y por tanto permite el uso de su stack de protocolos y herramientas como iperf, ping, entre otros, para desarrollar pruebas de desempeño de red [30].
- Permite la interacción del usuario, ya que se puede introducir código con funciones de red en los dispositivos de red directamente [30].
- Es rápidamente reconfigurable y de inicio rápido [30].
- Provee hasta 2Gbps de ancho de banda [30].

4 Evaluación de la transmisión de video HEVC: metodología y plataforma propuestas

En este capítulo se presenta la metodología y plataforma que finalmente fueron utilizadas para la evaluación de la transmisión de videos codificados con H265. Igualmente, se presenta un ejemplo de funcionamiento de la plataforma, y los scripts que automatizan la ejecución de las etapas del proceso.

4.1 Metodología

En la Figura 15 se presenta la metodología propuesta para la simulación de la transmisión de video codificado con H265, la cual corresponde a la integración de algunas herramientas de software libre conocidas para las etapas de codificación, empaquetado, encapsulado, decodificación y evaluación de la calidad del video recibido, con la herramienta MININET de emulación de SDN.

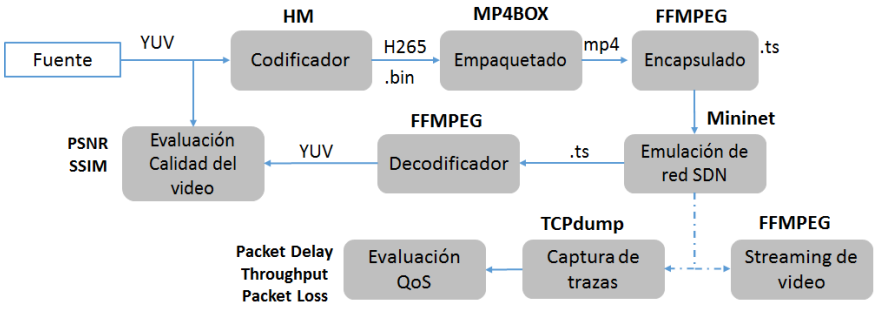


Figura 15. Metodología para evaluar la transmisión de flujos de video codificados con el estándar HEVC/H.265 sobre redes de comunicaciones, a través de un emulador de red (MININET) bajo diferentes condiciones de red.

4.2 Descripción de la plataforma de simulación

Como se observa en la Figura 15, la plataforma se compone de diferentes etapas con funciones propias de un proceso de simulación de la transmisión de video sobre redes de comunicaciones, las cuales son descritas a continuación y su funcionamiento se presentará en la siguiente sección con un ejemplo de aplicación.

Etapa 1 - Codificación: El video original en formato YUV es codificado con el códec H265, utilizando el codificador *HM (HEVC test Model)* [31], el cual es el software de referencia para el estándar HEVC/H265. La salida en esta etapa es el video codificado con extensión *".bin"*.

Etapa 2 - Empaquetado: Una vez codificado el video, este debe ser empaquetado en un contenedor *".mp4"*, para lo cual se utiliza la herramienta MP4BOX de GPAC [11].

Etapa 3 - Encapsulado: El video empaquetado en formato *".mp4"* debe ser encapsulado en formato *transport stream (ts)*, el cual permite la transmisión o streaming de este tipo de contenido. Este formato divide el video en pequeños paquetes o streams, los cuales son transmitidos en intervalos de tiempo dentro de un flujo de transporte conocido como *transport stream*. Esta etapa es ejecutada con la herramienta FFMPEG [10], y su salida es un archivo de

video con extensión *".ts"*, el cual corresponde al video a ser transmitido por la red emulada en la siguiente etapa.

Etapa 4 - Emulación de red SDN. En esta etapa se define la topología de red a emular con la herramienta MININET [30], la cual consiste en un emulador de SDN. La topología de red es creada mediante un script *Python*, en el cual se configuran los host, conmutadores, controladores, así como también las condiciones de red (retardo de los enlaces y el porcentaje de paquetes perdidos en los enlaces).

Etapas 5 y 6 - Captura de trazas y Streaming de video: Una vez iniciada la red SDN en la herramienta MININET, se procede a realizar el streaming de video desde el host transmisor hacia el host receptor, capturando al mismo tiempo las trazas de envío y recepción en cada extremo. Para esto, se abren instancias *xterm* (terminal) en el host transmisor y receptor, y se lanzan comandos de la herramienta TCPdump [20] para la captura de trazas, y FFMPEG para el streaming de video. El video encapsulado en formato *".ts"* (video transmitido) es la entrada a esta etapa, y en la salida se obtiene el video recibido en el host receptor (en formato *".ts"*) y las trazas de envío y recepción en formato de texto. Estas trazas describen todos los paquetes enviados y recibidos en ambos extremos del proceso, y son utilizadas en la siguiente etapa para la evaluación de calidad de servicio QoS.

Etapa 7 - Evaluación QoS. A partir de las trazas obtenidas en la etapa anterior, se realiza el cálculo de algunos parámetros de QoS tales como: retardo extremo a extremo, *throughput*, y porcentaje de paquetes perdidos. En esta etapa se desarrollaron scripts en lenguaje *Perl*, AWK (***A**lfred Aho, Peter **W**einberger, y Brian **K**ernighan*) y otras funciones de procesamiento de datos de Linux, para poder organizar las trazas y calcular las métricas de QoS. El código fuente de estos scripts se encuentra en el apéndice D.

Etapa 8 - Decodificador: El video recibido luego de la transmisión en formato *".ts"*, es decodificado y convertido a formato YUV para la posterior evaluación de calidad con respecto al video original. El proceso de decodificación se realiza con la herramienta FFMPEG.

Etapa 9 - Evaluación de calidad del video recibido: Una vez obtenido el video reconstruido con formato YUV en el extremo del receptor, este se compara fotograma por fotograma con el video original mediante cálculos de PSNR (Relación pico señal a ruido, *Peak Signal-to-Noise Ratio*) y SSIM (structural similarity index), como métricas de calidad de video, a través de herramientas de *Evalvid*. El PSNR define la relación entre la máxima energía posible de una señal y el ruido que afecta a su representación fidedigna [15]. Su definición matemática se encuentra en [15]. El SSIM es una métrica basada en distorsión, que considera la degradación de la calidad de la imagen como los cambios percibidos en la información estructural [32]. Su definición matemática se encuentra en [32].

4.3 Ejemplo de utilización del framework o plataforma

En esta sección se presenta un ejemplo de aplicación para mostrar cómo se desarrolla una simulación de la transmisión de video H265 utilizando la plataforma propuesta. En la Tabla 1 se describen las características y parámetros del video utilizado, y una imagen de un fotograma del mismo.

Tabla 1. *Parámetros del video transmitido en el ejemplo de aplicación.*

	Parámetro	Valor
 Fotograma del video 4k transmitido	Nombre	ShakeNDry_3840x2160_120fps_420_10bit_YUV.yuv
	Tamaño en disco	7.3 GB
	Formato	YUV
	Resolución	4k (3840x2160)
	Cantidad de fotogramas	300
	Tasa de fotogramas	120 fps
	Formato de muestreo	4:2:0
	Profundidad de bit	10
	Duración	2.5 s
	URL	http://ultravideo.cs.tut.fi/#testsequences [33]
	Propietario	Digiturk

Nota: Características y parámetros del video transmitido en el ejemplo de aplicación.

La topología de red emulada con la herramienta MININET para este ejemplo, se presenta en la Figura 16, la cual se compone de un *host* transmisor (h1), un *host* receptor (h2), un conmutador (s1), y un controlador (c0). El código fuente en *Python* para la para la creación de esta topología se encuentra en el apéndice C, con el nombre de *"H1S1H2.py"*.

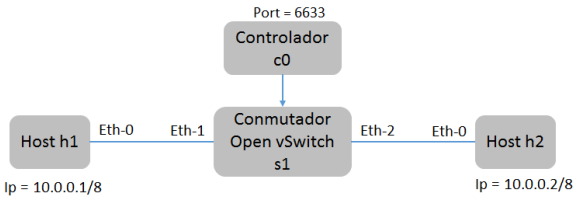


Figura 16. Topología SDN sencilla H1S1H2.py en MININET.

A continuación, se presenta la ejecución de cada una de las etapas de la plataforma a través de comandos, sin embargo, es importante mencionar que la ejecución de las mismas fue automatizada mediante scripts *".bash"*, los cuales serán explicados en la siguiente sección.

Etapa 1 - Codificación: Como se mencionó anteriormente, la codificación se realiza con el software referencia *HM (HEVC test Model)*, el cual permite codificar el video con el códec

HEVC/H265. Algunos parámetros de codificación son presentados en la Tabla 2, y se encuentran de forma completa en el apéndice D. Estos parámetros se deben registrar en el archivo *"encoding_config.cfg"*. Es importante mencionar que los parámetros de codificación que no se registran en este archivo, la herramienta HM toma sus valores por defecto.

Tabla 2. *Parámetros de codificación del video 4k transmitido.*

Parámetro	Valor
InputBitDepth	10
InputChromaFormat	420
FrameRate	120
SourceWidth	3840
SourceHeight	2160
FramesToBeEncoded	300
Profile	main10
QP	25

Nota: Algunos parámetros de codificación configurados en el codificador HM.

La codificación se ejecuta lanzando el siguiente comando en la línea de comandos:

```
TAppEncoderStatic -c encoding_config.cfg
```

En donde *TAppEnconderStatic* corresponde a la función de codificación en el codificador HM, *-c* hace el llamado al archivo de configuración *".cfg"* en el cual se registran los parámetros de codificación. La salida en esta etapa corresponde al video codificado *"outencoded.bin"*.

Etapa 2 - Empaquetado: El empaquetado en un contenedor mp4 del video codificado, se realiza a través del comando:

```
MP4Box -add outencoded.bin:fps=120:FMT=HEVC -new outencoded_hevc.mp4
```

Donde *outencoded.bin* corresponde al video codificado en H265, *fps* especifica la tasa de fotogramas por segundo, *FMT* el formato del video codificado y *outencoded_hevc.mp4* corresponde al video empaquetado en formato mp4, el cual se genera al finalizar la ejecución del comando.

Etapa 3 - Encapsulado: A través del comando mostrado a continuación, el video empaquetado se encapsula en formato *transport stream (ts)*, el cual divide el video en pequeños paquetes o streams para poder realizar posteriormente la transmisión.

```
ffmpeg -i outencoded_hevc.mp4 -map 0:v -c copy forTX.ts
```

La salida en esta etapa es *"forTX.ts"*, el cual corresponde al video listo para transmitir.

Antes de continuar con la siguiente etapa, es necesario iniciar el servidor y base de datos de *openvswitch*, el cual contiene la información de inicio de los conmutadores virtuales utilizados por la herramienta MININET en la siguiente etapa. Esto se realiza a través del siguiente comando, y su resultado se muestra en la Figura 17.

```
sudo /usr/local/share/openvswitch/scripts/ovs-ctl start
```

```
salopezgu@proyectomrsc:~/Vídeos/4K/ShakeNDry$ sudo /usr/local/share/openvswitch/scripts/ovs-ctl start
[sudo] password for salopezgu:
 * Starting ovsdb-server
 * system ID not configured, please use --system-id
 * Configuring Open vSwitch system IDs
 * Inserting openvswitch module
 * Starting ovs-vswitchd
 * Enabling remote OVSDB managers
salopezgu@proyectomrsc:~/Vídeos/4K/ShakeNDry$
```

Figura 17. Inicio de servidores y base de datos openvswitch.

Etapa 4 - Emulación de red SDN. En esta etapa se inicia la emulación de la topología de red sobre la cual se realizará la transmisión del video, utilizando la herramienta MININET.

Para dar inicio a la red, se ejecuta el script Python correspondiente a la topología de red creada y configurada, como lo muestra Figura 18.

```
salopezgu@proyectomrsc:~/Vídeos/4K/ShakeNDry$ sudo python H1S1H2.py
*** Creating Nodes ***
*** Adding Link ***
*** Starting Network ***
*** Configuring hosts
h1 h2
*** Runnign CLI ***
*** Starting CLI:
mininet>
```

Figura 18. Inicio de topología de red con MININET.

En este punto, ya se encuentra la red lista para iniciar la etapa de captura de trazas y streaming de video.

Etapas 5 y 6 - Captura de trazas y Streaming de video: En esta etapa, se abren instancias *xterm* (terminal) de cada host, y se lanzan los siguientes comandos de la herramienta TCPdump para la captura de trazas, y de FFMPEG para realizar el streaming:

En el host receptor (h2): Se habilita la captura del tráfico que recibe el nodo mediante TCPdump:

```
tcpdump –i h2-eth0 –nn –v –tt “src host 10.0.0.1 and dst host 10.0.0.2” > received
```

En el comando anterior se aplica el filtro *(“src host 10.0.0.1 and dst host 10.0.0.2”)* para que sólo se registre el tráfico proveniente del host 1. El tráfico registrado por TCPdump es almacenado en el archivo de texto (traza de tráfico) llamado *“received”*.

Figura 19. Comando para la captura de traza "received" en el host receptor.

En el host transmisor (h1): Se habilita la captura del tráfico que envía el nodo mediante TCPdump:

```
    tcpdump –i h1-eth0 –nn –v –tt “src host 10.0.0.1 and dst host 10.0.0.2” >  sent
```

En el comando anterior se aplica el filtro *(“src host 10.0.0.1 and dst host 10.0.0.2”)* para que sólo se registre el tráfico dirigido hacia el host 2. El tráfico registrado por TCPdump es almacenado en el archivo de texto (traza de tráfico) llamado *“sent”*.

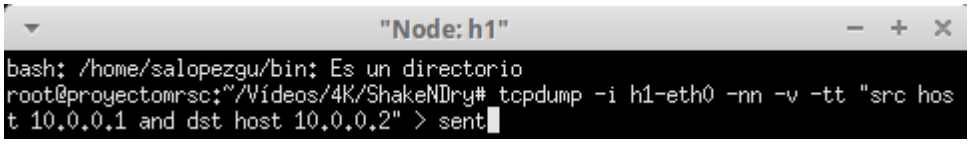


Figura 20. Comando para la captura de traza "sent" en el host transmisor.

Una vez se ha activado la captura de tráfico con TCPdump en ambos extremos de la comunicación, se inicia la transmisión del video en modo streaming con la herramienta FFMPEG.

En el host receptor (h2): se inicia la escucha del puerto 5003.

```
ffmpeg -i rtp://10.0.0.2:5003 -c copy rec_rtp.ts
```

donde *"rec_rtp.ts"* corresponde al video recibido en el extremo del receptor al finalizar el streaming.

Figura 21. Comando para recibir el streaming del video.

En el host transmisor (h1): se inicia el streaming del video.

```
ffmpeg -re -i forTX.ts -f rtp_mpegts rtp://10.0.0.2:5003
```

donde *"forTX.ts"* es el video a transmitir, el formato del streaming es *rtp_mpegts*, el protocolo de transmisión es RTP, y la dirección y puerto destino corresponde al host *h2*.

Figura 22. Comando para iniciar el streaming del video en host transmisor.

Es importante mencionar que la captura de trazas y el streaming de video se deben realizar en el orden mencionado anteriormente, de esta forma se garantiza que los archivos resultantes se generen correctamente.

Etapa 7 Evaluación QoS. En la Figura 23 se muestra la estructura de las trazas "sent" y "received" obtenidas en la etapa anterior. Estas trazas son procesadas con scripts con funciones de procesamiento de datos, para finalmente realizar el cálculo de algunos parámetros de QoS

tales como: retardo extremo a extremo, *throughput*, y porcentaje de paquetes perdidos. El código fuente de estos scripts se encuentra en el apéndice D.

```
1529115402.638142 IP (tos 0x0, ttl 64, id 21716, offset 0, flags [DF], proto UDP (17), length 1498)
    10.0.0.1.44038 > 10.0.0.2.5001: UDP, length 1470
1529115402.662148 IP (tos 0x0, ttl 64, id 21727, offset 0, flags [DF], proto UDP (17), length 1498)
    10.0.0.1.44038 > 10.0.0.2.5001: UDP, length 1470
1529115402.897740 IP (tos 0x0, ttl 64, id 21758, offset 0, flags [DF], proto UDP (17), length 1498)
    10.0.0.1.44038 > 10.0.0.2.5001: UDP, length 1470
1529115403.015295 IP (tos 0x0, ttl 64, id 21777, offset 0, flags [DF], proto UDP (17), length 1498)
    10.0.0.1.44038 > 10.0.0.2.5001: UDP, length 1470
1529115403.132815 IP (tos 0x0, ttl 64, id 21790, offset 0, flags [DF], proto UDP (17), length 1498)
    10.0.0.1.44038 > 10.0.0.2.5001: UDP, length 1470
1529115403.250816 IP (tos 0x0, ttl 64, id 21794, offset 0, flags [DF], proto UDP (17), length 1498)
    10.0.0.1.44038 > 10.0.0.2.5001: UDP, length 1470
```

Figura 23. Estructura de las trazas "sent" y "received".

La Figura 24 presenta el diagrama de flujo con las acciones ejecutadas por los scripts para el cálculo de los parámetros de QoS mencionados anteriormente.

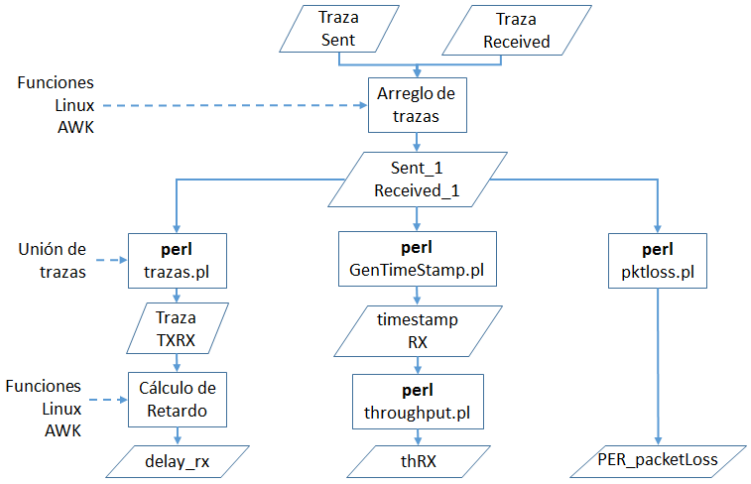


Figura 24. Diagrama de flujo para cálculo de QoS.

Etapa 8 - Decodificación: El proceso de decodificación se realiza con la herramienta FFMPEG, a través del siguiente comando:

```
ffmpeg -i rec_rtp.ts vid_rec.yuv
```

Donde *"vid_rec.yuv"* es la salida, y corresponde al video reconstruido en el extremo del receptor con formato YUV.

Etapa 9 - Evaluación de calidad del video recibido: Para evaluar la calidad del video recibido, se utiliza la herramienta PSNR de Evalvid, mediante el siguiente comando:

```
psnr 3840 2160 420 ShakeNDry_3840x2160_120fps_420_10bit_YUV.yuv
vid_rec.yuv > psnr_final
```

Con este comando se compara el video original transmitido, con el video recibido, ambos en formato YUV, y el resultado es almacenado en el archivo *"psnr_final"*, el cual es graficado con la herramienta GNUplot, mediante los comandos mostrados a continuación.

```
plot "psnr_final" w l linecolor rgb "blue" title 'PSNR'
set grid
set xlabel 'FRAME'
set ylabel 'PSNR [dB]'
set title 'PSNR Loss5 Delay5'
replot
```

En la Figura 25 se muestra un ejemplo de un fotograma para el video transmitido y recibido, en el cual se evidencia la degradación de la imagen que sufre el video, luego del proceso de transmisión y reconstrucción en el receptor.

(a) (b)

Figura 25. Fotograma del video: (a) Transmitido, (b) Recibido.

En este punto, se ha finalizado el ejemplo de aplicación donde se da a conocer el funcionamiento de cada una de las etapas de la plataforma propuesta.

4.4 Empaquetamiento de las tareas

Con el objetivo de automatizar, mediante el uso de scripts, la ejecución de cada una de las etapas de la plataforma propuesta, la metodología fue dividida en tres fases así:

- Una primera fase que incluye las etapas 1, 2 y 3, la cual es ejecutada mediante el script *"Encoding.sh"*, cuyo diagrama de flujo se presenta en la Figura 26, y su código fuente se presenta en el apéndice D.

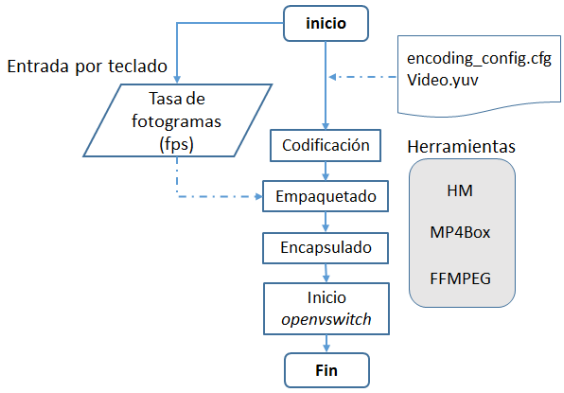


Figura 26. Diagrama de flujo del script "Encoding.sh".

- La segunda fase corresponde al uso de la herramienta MININET, la cual se ejecuta de forma manual lanzando los comandos mencionados en las etapas 4, 5 y 6. Estas no se incluyeron en un script dado que los comandos se digitan directamente en diferentes instancias virtuales (terminales *xterm*) de los hosts, requiriendo la entrada por teclado por parte del usuario, y permitiendo el uso de comandos como ping y otros para realizar test sobre la red.

- La tercera fase incluye lo correspondiente al procesamiento de trazas, evaluación de QoS, decodificación, y evaluación de calidad del video recibido, es decir, las etapas 7, 8 y 9. Esta fase se realiza mediante la ejecución del script QoS_Qvid.sh, cuyo diagrama de flujo se presenta en la Figura 27, y su código fuente se presenta en el apéndice D.

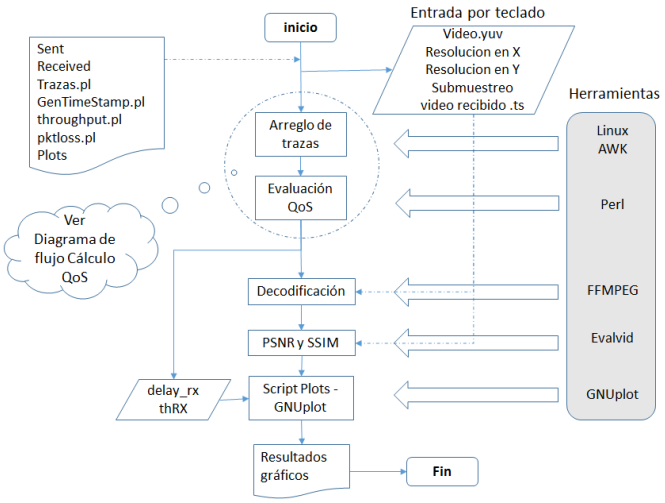


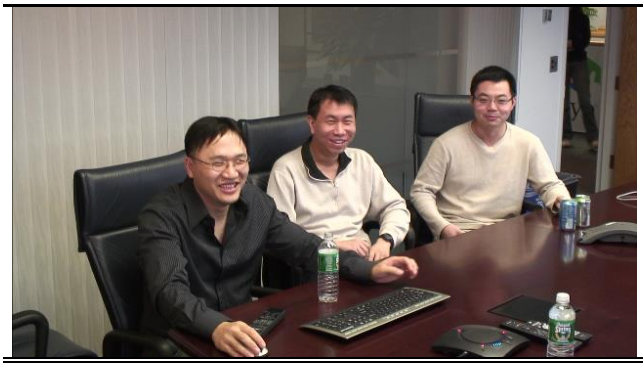
Figura 27. Diagrama de flujo del script "QoS_Qvid.sh".

Con la metodología descrita anteriormente, se conforma entonces un *framework* completo para la evaluación de la transmisión de videos de ultra alta definición codificados con el estándar HEVC/H265.

5 Simulaciones y resultados de evaluación calidad de servicio y calidad del video recibido

Utilizando el *framework* o plataforma descrita en el capítulo anterior, a continuación, se presentan los resultados obtenidos a partir de las simulaciones realizadas sobre tres topologías propuestas, transmitiendo el video conocido con el nombre *vidyo1*, cuyas características se describen en la Tabla 3.

Tabla 3. *Características video transmitido Vidyo1.*



Parámetro	Valor
nombre	Vidyo1
Formato de muestro	4:2:0
Tasa de fotogramas	60 fps
Cantidad de fotogramas	300
Formato	YUV

Nota: Características que describen el video transmitido Vidyo1.

En la Tabla 4 se presentan las condiciones de red configuradas en los enlaces para cada escenario simulado:

Tabla 4. *Condiciones de red por experimento.*

id experimento	Delay [ms]	Packet Loss [%]
1	1	0
2	1	1
3	1	2
4	1	3

Nota: Valores de retardo y pérdida de paquetes para los enlaces en cada topología.

Es importante mencionar que los fundamentos para crear las topologías mencionadas a continuación fueron tomados de [34], en donde se tienen algunos ejemplos prácticos para experimentar con SDN.

Las métricas calculadas en cada uno de los experimentos fueron: retardo, *throughput* y pérdidas de paquetes, los cuales son parámetros que permiten evaluar el nivel de QoS de la red. Además se calcularon parámetros para evaluar la calidad del video recibido como: el PSNR y el índice SSIM, los cuales fueron explicados en el capítulo anterior como parte de la descripción de la plataforma de simulación. Su definición matemática se encuentra en [15] y [32], respectivamente. El SSIM es importante porque indica el índice de similitud entre dos

macrobloques, tomando valores entre 0 y 1, en donde el valor de 1 indica que los macrobloques comparados son idénticos [32].

5.1 Escenario 1: Topología sencilla

Esta topología corresponde a la mostrada en el capítulo anterior (ver Figura 16), la cual estaba conformada por un *host* transmisor (h1), un *host* receptor (h2), un conmutador (s1), y un controlador (c0). El código fuente en *Python* la para la creación de esta topología se encuentra en el apéndice C, con el nombre de H1S1H2.py.

Luego de la ejecución de las simulaciones, los resultados obtenidos fueron los siguientes. En primer lugar, en la Figura 28 se muestran los resultados del retardo de los paquetes recibidos, en donde se evidencia que el retardo para los primeros paquetes, se incrementa en los cuatro experimentos, lo cual es debido al proceso de establecimiento de las rutas al inicio de la transmisión, y al procesamiento implícito en los dispositivos de red como conmutador y controlador. Igualmente, se evidencia que a medida que aumentan las pérdidas de paquetes en los enlaces, el retardo medio extremo a extremo aumenta, como se muestra en la Tabla 5.

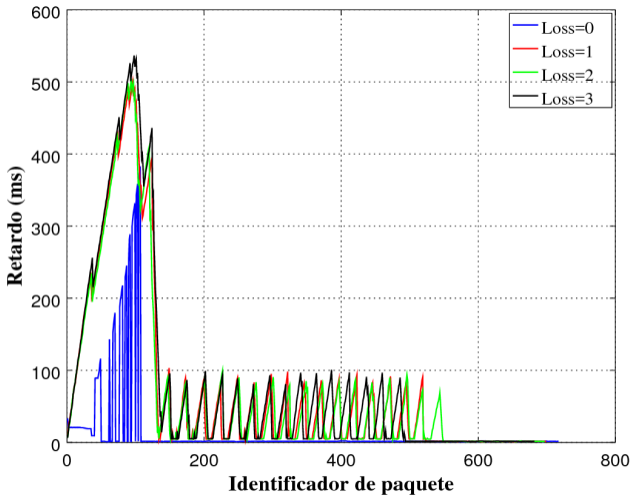


Figura 28. Packet Delay para el escenario 1: topología sencilla.

Tabla 5. *Packet Delay medio para el escenario 1: topología sencilla*.

id Experimento	Retardo medio (ms)
1	13.55
2	74.802
3	73.819
4	80.667

Nota: Retardo medio en milisegundos, obtenido para cada experimento en la topología sencilla.

En la Figura 29 se presenta el *Throughput* medido en el host receptor, en donde se evidencia la disminución de este indicador, a medida que aumentan las pérdidas de paquetes. Lo anterior se sustenta al obtener los valores de *throughput medio* (ver

Tabla 6), en donde se observa que este indicador disminuye a medida que se presentan variaciones en las condiciones de los enlaces, lo cual refleja que las métricas de QoS son coherentes, es decir, en el experimento con más pérdida de paquetes se obtiene un valor de throughput medio menor, debido a la información que se pierde correspondiente a los paquetes perdidos. También se evidencia que, en el experimento 1, al no tener pérdida de paquetes, los paquetes llegan más rápido y la transmisión termina unos instantes antes que los demás experimentos.

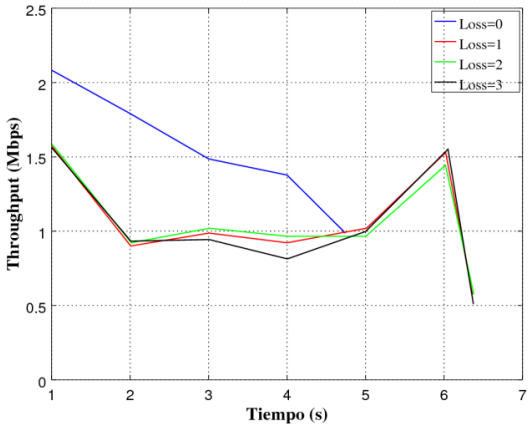


Figura 29. Throughput en el escenario 1: topología sencilla

Tabla 6. *Throughput medio en el escenario 1: topología sencilla*.

id experimento	Throughput medio (Mbps)
1	1.5448
2	1.0725
3	1.0679
4	1.0446

Nota: Throughput medio en Mbps, obtenido para cada experimento en la topología sencilla.

La Tabla 7 resume los valores obtenidos en cuanto a paquetes perdidos para los cuatro experimentos, en la cual se evidencia que este indicador es congruente con los valores configurados en los enlaces. Lo cual, a su vez, demuestra que los scripts programados para el cálculo de estos parámetros a partir de las trazas de tráfico registradas, funcionan correctamente.

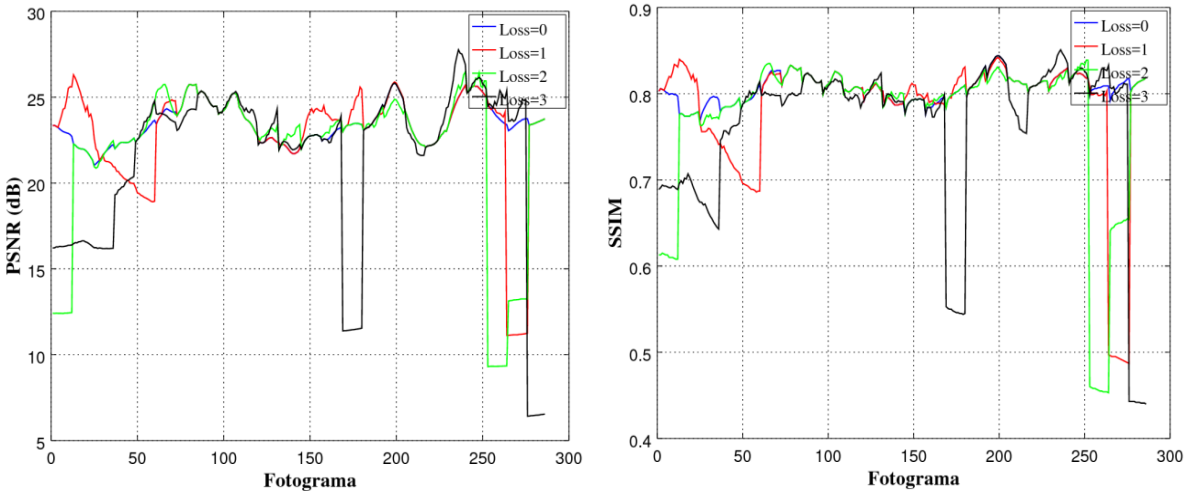
Tabla 7. *Packet Loss en el escenario 1: topología sencilla.*

id experimento	[#] Paquetes enviados	[#] Paquetes recibidos	[#] Paquetes perdidos	[%] Paquetes perdidos
1	717	717	0	0
2	708	699	9	1.2712
3	711	697	14	1.969
4	697	682	15	2.152

Nota: Relación de paquetes perdidos extremo a extremo en los experimentos para la topología sencilla.

Por otra parte, en cuanto a la evaluación de la calidad del video recibido, la Figura 30 muestra el resultado del PSNR y SSIM para los experimentos realizados. De acuerdo a los resultados obtenidos, en el primer experimento (0 pérdidas) el video recibido tiene mejor calidad (psnr promedio de 23.44 dB), un 2% mejor que la calidad del video del experimento 2 (psnr promedio de 22.96 dB), 6% mejor que la calidad del video del experimento 3 (psnr promedio de 22.06 dB), y 8% mejor que la calidad del video del experimento 4 (psnr promedio de 21.6 dB). Esto era previsible ya que al aumentar el porcentaje de paquetes perdidos en los experimentos con

pérdidas, se disminuyó el número de fotogramas correctamente decodificables. Igualmente, la curva de SSIM presenta disminución en los mismos fotogramas en donde el PSNR disminuye, lo cual refleja la correlación que hay entre estas dos métricas, y comparando los valores medios de SSIM (0.8, 0.78, 0.77, 0.76, respectivamente), se evidencia que el índice de similitud disminuye a medida que aumentan las pérdidas de paquetes.



(a) (b)

Figura 30. Escenario 1: Topología sencilla: evaluación de la calidad del video (a) PSNR, (b) SSIM.

5.2 Escenario 2: topología con división de tráfico en dos caminos (Split)

En la Figura 31 se presenta la topología *Split*, en la cual el tráfico recibido en S1 desde h1, es dividido 50% hacia S2 y 50% hacia S3, generando un divisor de tráfico, y mediante modificaciones del enlace S1-S2, se generan retardos y desorden en la llegada de paquetes hacia h3. El streaming de video se realiza desde h1 hacia h3. Dado que se trata de una topología multi-camino, en la cual se requiere tomar de decisiones de reenvío de paquetes (dividir tráfico), se creó un script OpenFlow para definir las rutas de reenvío. El código fuente en *Python* para la creación de esta topología y el script de los flujos, se encuentran en el apéndice C, con el nombre

de *"Split.py"* y *"SplitFlows.sh"*, respectivamente. Esta topología fue tomada de [35] para efectos de aprendizaje y fines académicos.

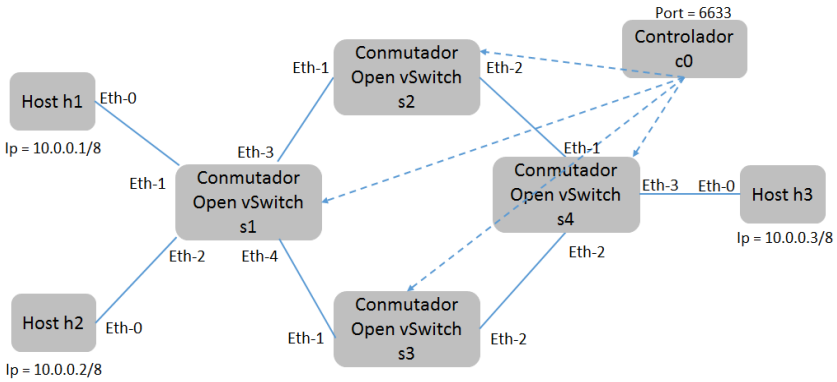


Figura 31. Escenario 2: Topología SDN split (dos caminos) Split.py en Mininet.

Luego de la ejecución de las simulaciones, los resultados obtenidos fueron los siguientes. En la Figura 32 se muestran los resultados del retardo de los paquetes recibidos, en donde, a diferencia que en el escenario anterior, se evidencia que el retardo para los primeros paquetes es mayor, pero en menor proporción que en el escenario 1, esto es debido a que en el escenario 2 se programan las rutas directamente en los conmutadores a través de script de OpenFlow, eliminando el proceso inicial de establecimiento de rutas, presentado en el escenario 1.

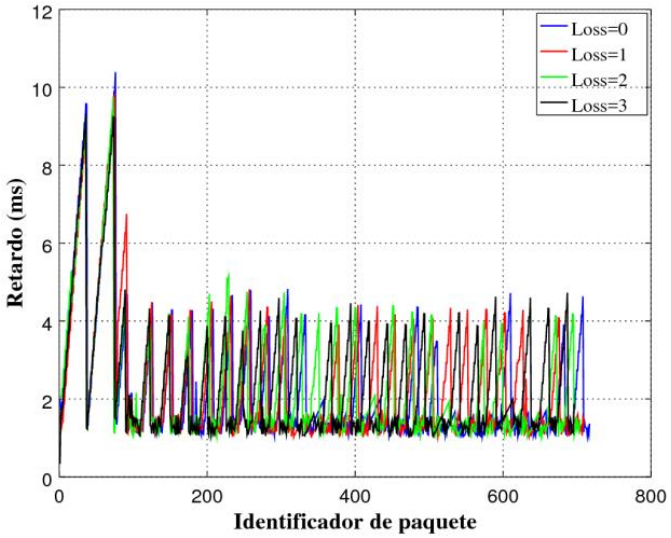


Figura 32. Packet Delay para el escenario 2: topología split

La Figura 33 presenta el *throughput* medido en el host receptor, en donde se evidencia que este indicador sigue igual tendencia para los cuatro experimentos. Al obtener los valores de *throughput medio* (ver Tabla 8), se observa que este indicador disminuye ligeramente a medida que se presentan variaciones en las condiciones del enlace S1-S2, lo cual refleja que las métricas de QoS son coherentes, es decir, en el experimento con más pérdida de paquetes se obtiene un valor de throughput medio menor, debido a la información que se pierde correspondiente a los paquetes perdidos.

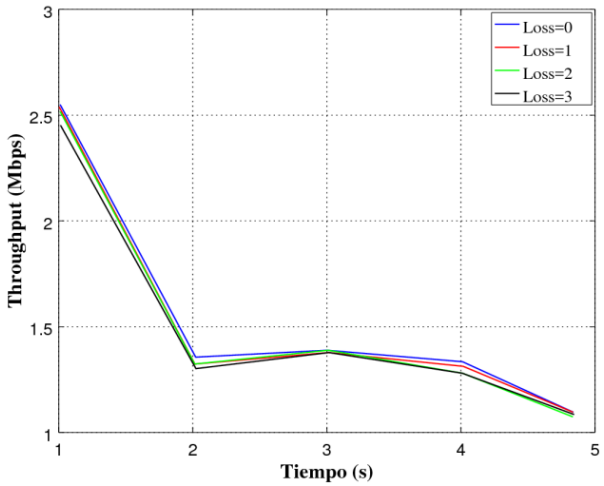


Figura 33. Throughput en el escenario 2: topología Split.

Tabla 8. *Throughput medio en el escenario 2: topología Split.*

id experimento	Throughput medio (Mbps)
1	1.5448
2	1.5297
3	1.5166
4	1.4993

Nota: Throughput medio en Mbps, obtenido para cada experimento en la topología split.

Los valores de *Packet Loss* en la Tabla 9 corresponden a los resultados obtenidos para la topología Split, y en estos se evidencia la congruencia con las pérdidas configuradas en el enlace S1-S2. De esta forma, nuevamente se demuestra el correcto funcionamiento de los scripts

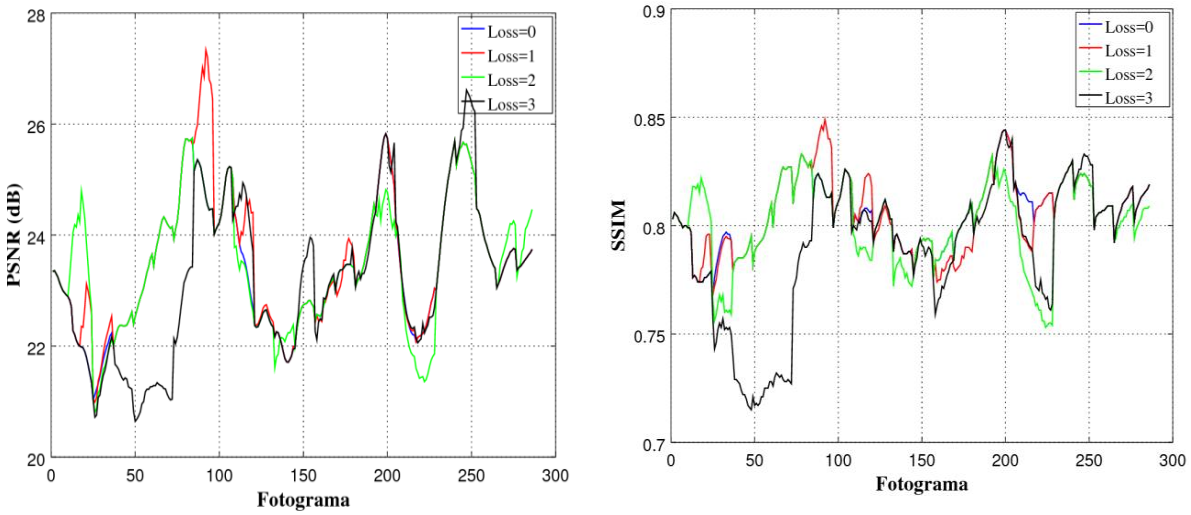
programados para realizar el cálculo de estos parámetros a partir de las trazas de tráfico registradas.

Tabla 9. *Packet Loss en el escenario 2: topología split.*

id experimento	[#] Paquetes enviados	[#] Paquetes recibidos	[#] Paquetes perdidos	[%] Paquetes perdidos
1	717	717	0	0
2	717	710	7	0.976
3	717	704	13	1.813
4	717	696	21	2.928

Nota: Relación de paquetes perdidos extremo a extremo en los experimentos para la topología split.

En cuanto a la calidad del video recibido, la Figura 34 muestra el resultado del PSNR y SSIM para los experimentos realizados, en donde se evidencia que la calidad de la imagen en el video recibido para el experimento 4 (3% de pérdida de paquetes en el enlace S1-S2), entre los fotogramas 25 a 80, se degrada. Al calcular los valores de psnr promedio, no existe mayor diferencia entre los experimentos, teniendo valores cercanos a 23.4 dB con diferentes desviaciones estándar. No existe diferencia significativa entre los valores medios de PSNR, dado que sólo se están variando las condiciones de red en un solo enlace (S1-S2), lo cual no genera tanta afectación como en el escenario 1, en donde todos sus enlaces presentaban variaciones de las condiciones de red (pérdida de paquetes). El SSIM presenta un comportamiento similar, registrando disminución de índice de similitud entre los fotogramas 25 a 80 aproximadamente, lo cual es correlacionado con el comportamiento del PSNR.



(a) (b)

Figura 34. Escenario 2: Topología split: evaluación de la calidad del video (a) PSNR, (b) SSIM

5.3 Escenario 3: topología NSFnet-14

Con el objetivo de implementar una topología más completa, y demostrar la capacidad de la plataforma de simular topologías más realistas y complejas, se desarrolló la topología *NSFnet-14* (*National Science Foundation Network* – NSFNET). NSFNET fue el nombre dado a la topología del *backbone* T1 de la red de computadores en Estados Unidos con 14 nodos y 21 enlaces bidireccionales [36].

Para la investigación en este proyecto, el streaming de video se realiza desde host h1 hacia host h2, a través de una ruta definida en la NSFnet-14. Esta topología se muestra en la Figura 35.

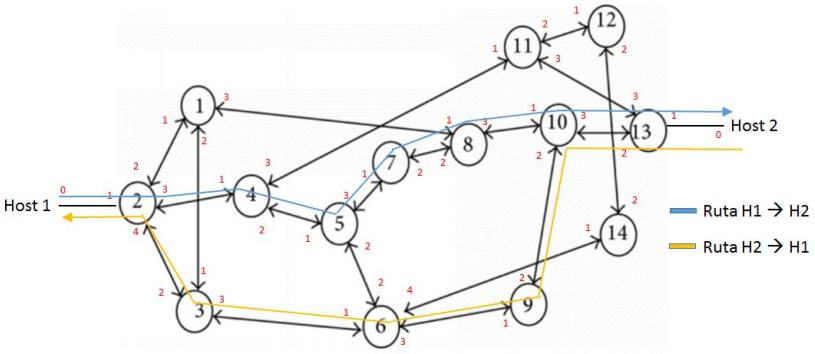


Figura 35. Escenario 3: Topología SDN NSFnet14.py en Mininet.

Los resultados obtenidos en los experimentos para esta topología se muestran a continuación. En la Figura 36 se muestra el comportamiento del retardo en los paquetes recibidos, en los cuales se evidencia que los valores son proporcionales al número de saltos (enlaces) que encuentra la ruta definida, teniendo en cuenta los retardos y pérdidas configurados en cada enlace. Estos valores son mayores a los obtenidos en las topologías anteriores, toda vez que esta red es de mayor dimensión. Igualmente, se evidencia que, debido a las pérdidas de paquetes, la cantidad de fotogramas que llegan al receptor es menor a medida que se cambian las condiciones en los enlaces, es decir, a mayor pérdida de paquetes, menos fotogramas o información llega al receptor.

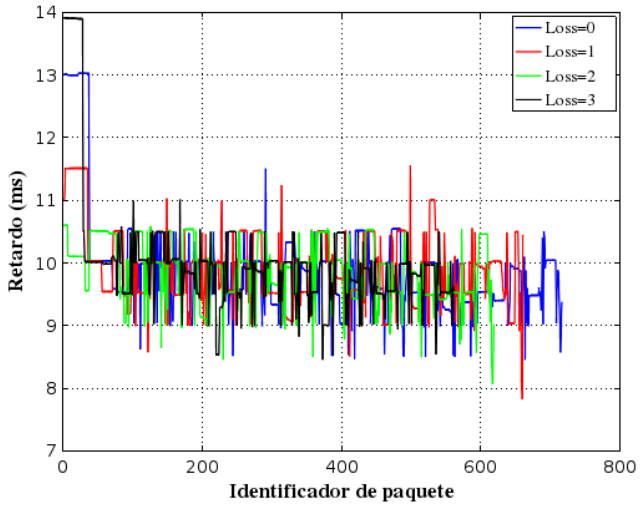


Figura 36. Packet Delay para el escenario 3: topología NSFnet-14

La Figura 37 presenta los valores de *throughput* medidos en el host h2, en donde se evidencia que a medida que aumentan las pérdidas de paquetes, el throughput disminuye debido a la información que se deja de recibir con los paquetes perdidos. Esto se demuestra con los valores de throughput medio mostrados en la Tabla 10. Al igual que en los escenarios presentados anteriormente, en el primer segundo de transmisión se obtiene un pico de throughput en los cuatro experimentos, esto se debe a que durante ese instante se recibe la mayor cantidad de paquetes, acumulando así mayor cantidad de información.

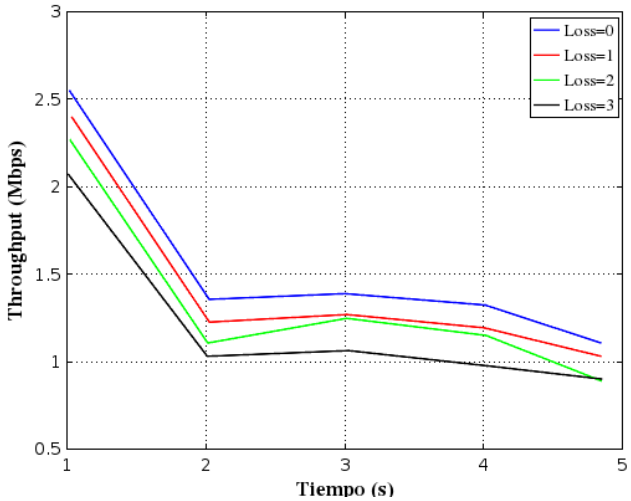


Figura 37. Throughput en el escenario 3: topología NSFnet-14

Tabla 10. *Throughput medio en el escenario 3: topología NSFnet-14.*

id experimento	Throughput medio (Mbps)
1	1.5448
2	1.4233
3	1.3322
4	1.2086

Nota: Throughput medio en Mbps, obtenido para cada experimento en la topología NSFnet-14.

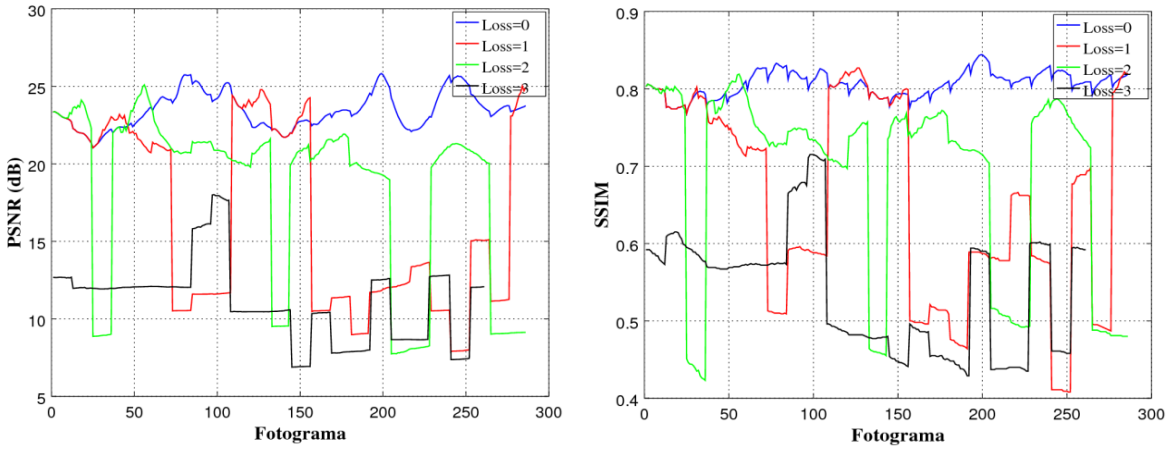
En la Tabla 11 se muestran los valores de *Packet Loss*, en donde igualmente se evidencia el incremento en el porcentaje de paquetes perdidos con respecto a las topologías de red simuladas en los casos anteriores, esto debido a la dimensión mayor que tiene la topología NSFnet-14.

Tabla 11. *Topología NSFnet-14. Packet Loss.*

id experimento	[#] Paquetes enviados	[#] Paquetes recibidos	[#] Paquetes perdidos	[%] Paquetes perdidos
1	717	717	0	0
2	709	661	48	6.77
3	698	619	79	11.32
4	698	562	136	19.484

Nota: Relación de paquetes perdidos extremo a extremo en los experimentos para la topología NSFnet-14.

En cuanto a la calidad del video recibido, la Figura 38 muestra los resultados para las métricas PSNR y SSIM. De acuerdo a los resultados obtenidos, en el primer experimento 1 (0 pérdidas) el video recibido tiene mejor calidad (psnr promedio de 23.44 dB), un 30% mejor que la calidad del video del experimento 2 (psnr promedio de 16.4 dB), 22% mejor que la calidad del video del experimento 3 (psnr promedio de 18.2 dB), y 53% mejor que la calidad del video del experimento 4 (psnr promedio de 11.1 dB). Estos valores son significativamente mayores a los obtenidos en los escenarios anteriores, dada la magnitud que tiene la topología NSFnet-14. Igualmente, la curva de SSIM presenta disminución en los mismos fotogramas en donde el PSNR disminuye, lo cual refleja la correlación que hay entre estas dos métricas, y comparando los valores medios de SSIM (0.8, 0.6, 0.6, 0.5, respectivamente), se evidencia que el índice de similitud disminuye a medida que aumentan las pérdidas de paquetes.



(a) (b)

Figura 38. Escenario 3: Topología NSFnet-14: evaluación de la calidad del video (a) PSNR, (b) SSIM.

Al finalizar las simulaciones con los escenarios mencionados anteriormente, se procedió a realizar simulaciones con las mismas topologías de red, transmitiendo el video descrito en la Tabla 1, y sus resultados son presentados en el apéndice F.

6 Conclusiones

Con el desarrollo de este proyecto, se consiguió recorrer un camino investigativo en la línea de comunicaciones multimedia, respecto a la transmisión de flujos de video codificados con el estándar HEVC/H.265. Este proceso incluyó el análisis y consulta de los antecedentes y estado del arte de la evaluación de la transmisión de contenido de video codificado con HEVC/H265, lo cual permitió conocer las generalidades del estándar e identificar las principales características de la codificación de video en ultra alta definición, así como sus principales aplicaciones. Igualmente, el análisis y estudio de antecedentes, permitió identificar las ventajas o ganancias que tiene el estándar HEVC/H.265 sobre su antecesor H264/AVC en cuanto a los requerimientos técnicos de red, en donde principalmente se resalta el ahorro de hasta 50% de ancho de banda, dado que requiere hasta la mitad del *bitrate* utilizado por H264 para obtener la

misma calidad en la imagen. Lo anterior resalta la importancia de la nueva codificación, ya que disminuye significativamente la demanda de recursos por parte de la red. En cuanto a la tolerancia por parte del usuario ante las pérdidas de paquetes, los estudios establecieron una tasa de 3% en pérdida de paquetes, a partir de la cual, el usuario manifiesta mala calidad en la experiencia.

Por otra parte, se logró consultar y probar algunas herramientas software utilizadas en la simulación de la transmisión de video, para verificar su posible uso en el estudio del video de ultra alta definición, encontrando que algunas no están disponibles, otras se probaron con éxito pero no son los suficientemente completas para llevar a cabo esta investigación, con lo cual finalmente se identificó la herramienta MININET, emulador de redes definidas por software (SDN), que permite integrarse junto a otras herramientas conocidas como FFMPEG, mp4Box y Evalvid para conformar un *framework* que permite analizar de extremo a extremo el streaming de video codificado con H265. Lo anterior corresponde a la principal contribución de este proyecto investigativo, en donde se definió una plataforma o *framework* que permite la integración de herramientas de software libre, fácilmente instalables, para evaluar diferentes topologías de red sin la necesidad de implementar un red física, generando resultados reales, dado que utiliza la herramienta de emulación MININET, la cual permite emular la red con dispositivos reales, utilizando los recursos computacionales y de red disponibles en el computador.

La metodología propuesta en este proyecto, permitió evaluar la calidad de los videos de ultra alta definición durante la transmisión sobre diferentes escenarios de red, mediante la implementación y ejecución de simulaciones, calculando, a partir de la obtención de trazas de

envío y recepción (extremo a extremo), parámetros de desempeño de red (QoS) como *packet delay, throughput* y *packet loss*, y parámetros de calidad de video como PSNR y SSIM.

Con los resultados y logros mencionados anteriormente, se alcanzó el objetivo general de esta investigación, el cual estaba dirigido a diseñar una metodología para evaluar la transmisión de flujos de video codificados con el estándar HEVC/H.265 sobre redes de comunicaciones, a través de herramientas de modelado y simulación bajo diferentes condiciones de red. De esta manera, se contribuye a las necesidades académicas actuales en cuanto a la comprensión de video en internet y a la transmisión de este tipo de contenido con resoluciones 4K (ultra alta definición), en donde el tamaño del video y la capacidad de transmisión juegan un papel importante frente a la satisfacción del usuario final, quien cada vez demanda mejor calidad en sus contenidos.

Los resultados obtenidos a partir de los experimentos realizados muestran correspondencia, tanto entre los valores configurados en la topología de red y las métricas calculadas durante la simulación. También hay coherencia entre los parámetros de QoS calculados, y la calidad del video recibido, es decir, la plataforma muestra fielmente la correlación que existente entre la calidad del video recibido con las condiciones de red, en donde se comprobó que, al aumentar los retardos y pérdidas de paquetes en los enlaces para las tres topologías evaluadas, la calidad del video recibido se veía comprometida, obteniendo valores bajos de PSNR y SSIM. Igualmente, los valores de desempeño de red que permiten medir la calidad de servicio QoS, reflejan impacto en sus indicadores de *packet delay, throughput* y *packet loss*, a medida que las condiciones de red en los enlaces se desmejoran. A partir de lo mencionado anteriormente, se demuestra que la plataforma y los cálculos realizados con los algoritmos implementados, reflejan fielmente las configuraciones realizadas en las topologías de red.

Como alcances académicos puntuales durante el desarrollo de esta investigación, se logró afianzar las habilidades y conocimientos en el streaming de video, fortaleciendo las técnicas para la codificación, empaquetado, transmisión y decodificación de este tipo de contenido, logrando definir metodologías o estrategias para el desarrollo de experimentos de video streaming en el marco de las comunicaciones multimedia. Igualmente, permitió conocer y aplicar los conceptos básicos de las SDN, a través de la emulación de este tipo de redes con la herramienta MININET, para la transmisión de contenido de video. Con lo anterior, se incentiva a continuar con la exploración y uso de este tipo de redes, las cuales representan un paradigma en el diseño y despliegue de las redes de comunicaciones.

6.1 Trabajo Futuro

Una vez definida la metodología para la evaluación de la transmisión de flujos de video codificados con el estándar HEVC/H265, tal como lo era el objetivo general de este proyecto, se dejan las pautas iniciales para desarrollar simulaciones exhaustivas con diferentes combinaciones de parámetros de red, que simulen escenarios reales, para definir umbrales mínimos, tanto en la codificación del video como en los parámetros de red, y lograr transmitir este tipo de contenido sin comprometer la calidad del video recibido, garantizando buena experiencia de usuario. Así mismo, definir un set de parámetros de codificación HEVC, con el codificador HM, que permita obtener ventajas en cuanto a tiempo de codificación y tamaño final de comprensión, toda vez que los videos de ultra alta definición son de tamaño significativo aun siendo de corta duración, lo cual representa una limitante importante en la ejecución de múltiples simulaciones que permitan definir umbrales de medición en algún punto del proceso. Así mismo, se propone la integración de algoritmos o herramientas necesarias para

la simulación de este tipo de transmisión sobre redes inalámbricas, ampliando de esta manera el rango de aplicación de la plataforma.

Por otra parte, para contribuir a investigaciones actuales en protocolos de transmisión de video, se recomienda realizar pruebas utilizando protocolo de transporte adaptativo como DASH (*Dynamic Adaptive Streaming over HTTP*), el cual es utilizado en plataformas modernas como Netflix. Igualmente, se propone la inclusión de técnicas de codificación escalable y multivista, que permitan la evaluación de la transmisión adaptativa de video de ultra alta definición, y los flujos de video 3D.

Referencias

[1] W. D. Mattos y P. R. Gondim, «M-Health Solutions Using 5G Networks and M2M Communications,» *IT Professional,* vol. 18(3), pp. 24-29, 2016.

[2] D. R. Bull, Communicating Pictures: A Course in Image and Video Coding, Academic Press, 2014.

[3] CISCO, «Noticias de CISCO,» Lewis, [En línea]. Available: http://cisco.mwnewsroom.com/es/es/release/El-tr%C3%A1fico-IP-global-se-multiplicar%C3%A1-casi-por-tres-entre-2015-y-2020-2367113.

[4] Ericsson, «Ericsson Mobility Report,» [En línea]. Available: https://www.ericsson.com/assets/local/mobility-report/documents/2016/ericsson-mobility-report-november-2016-rlam.pdf.

[5] J. Nightingale, Q. Wang y C. Grecos, «HEVStream: A Framework for Streaming and Evaluation of High Efficiency Video Coding (HEVC) Content in Loss-prone Networks,» *IEEE Transactions on Consumer Electronics,* vol. 58, nº 2, 2012.

[6] S. u. Rehman y G. Raja, «Performance Evaluation of HEVC over Broadband,» 2014.

[7] A. Dethof, W. Robitza y M.-N. Garcia, «StreamSim: A Video Streaming Simulation Toolchain for Unreliable Transport Mechanisms,» de *QoMEX – Eighth International Conference on Quality of Multimedia Experience*, Lisboa, 2016.

[8] M. M. Rashed Khan, HEVC-SIM: A simulation and analysis tool for H.265 packet transmission, Burnaby, Canada: SIMON FRASER UNIVERSITY, 2015.

[9] T. A. Hadhrami, J. Nightingale, Q. Wang, C. Grecos y N. Kehtarnavaz, «A simulator tool set for evaluating HEVC/SHVC streaming,» *SPIE-IS&T Electronic Imaging,* vol. 9400, 2015.

[10] ffmpeg, «ffmpeg,» [En línea]. Available: https://www.ffmpeg.org/about.html. [Último acceso: 25 07 2018].

[11] GPAC, «MP4Box,» [En línea]. Available: https://gpac.wp.imt.fr/mp4box/. [Último acceso: 25 07 2018].

[12] T. -. T. N. Group, «EvalVid - A Video Quality Evaluation Tool-set,» [En línea]. Available: http://www.tkn.tu-berlin.de/research/evalvid/.

[13] ns2, «The Network Simulator - ns-2,» [En línea]. Available: https://www.isi.edu/nsnam/ns/. [Último acceso: 25 07 2018].

[14] W. Castellanos, P. Guzman, P. Arce y J. Guerri, «Mechanisms for Improving the Scalable Video Streaming in Mobile Ad hoc Networks,» *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems,* pp. 33-40, 2015.

[15] W. Castellanos, Quality of Service Routing and Mechanisms for Improving Video Streaming over Mobile Wireless Ad hoc Networks, Valencia: Editorial Universitat Politècnica de València, 2015.

[16] A. Lie y J. Klaue, «Evalvid-RA: trace driven simulation of rate adaptive MPEG-4 VBR video,» *Multimedia Systems,* vol. 14, pp. 33-50, 2008.

[17] T. Le y H. Nguyen, «End-to-end transmission of scalable video contents: performance evaluation over EvalSVC — a new open-source evaluation platform,» *Multimed Tools Appl*, vol. 72, p. 1239 – 1256, 2014.

[18] V. STL, «VQEG - Tools and Subjective Labs Setup,» [En línea]. Available: http://vqegstl.ugent.be/?q=node/27. [Último acceso: 28 07 2018].

[19] M. A. Brown, «Traffic Control HOWTO,» [En línea]. Available: http://tldp.org/HOWTO/Traffic-Control-HOWTO/intro.html. [Último acceso: 28 07 2018].

[20] Tcpdump/Libpcap, «Tcpdump&Libpcap,» [En línea]. Available: http://www.tcpdump.org. [Último acceso: 28 07 2018].

[21] P. Biondi y C. Scapy, «Scapy - Packet crafting for Python2 and Python3,» [En línea]. Available: https://scapy.net/. [Último acceso: 28 07 2018].

[22] xiph, «xiph.org,» [En línea]. Available: https://media.xiph.org/video/derf/. [Último acceso: 29 07 2018].

[23] B. C. Latinoamerica, «Perspectivas de SDN para América Latina,» [En línea]. Available: https://gblogs.cisco.com/la/perspectivas-de-sdn-para-america-latina/. [Último acceso: 29 07 2018].

[24] F. &. Sullivan, «Software Defined Networking: en busca de la automatización de la red,» SlideShare, [En línea]. Available: https://www.slideshare.net/ciscolatinoamerica/wp-premium-content-fssdn-en-busca-de-la-automatizacin-de-la-red. [Último acceso: 29 07 2018].

[25] M. Karakus y A. Durresi, «Economic Viability of Software Defined Networking (SDN),» *Computer Networks,* vol. 135, pp. 81-95, 04 2018.

[26] Sandhya, Y. Sinha y K. Haribabu, «A survey: Hybrid SDN,» *Journal of Network and Computer Applications,* vol. 100, pp. 35-55, 12 2017.

[27] D. A. Serrano Carrera y J. C. Guerri Cebollada, Redes Definidas por Software (SDN): OpenFlow, Valencia: Universitat Politecnica de Valencia, 2015, p. 55.

[28] F. J. Navarro Ojeda y P. Manzoni, Modelado de redes SDN con mininet, Valencia: Universitat Politecnica de Valencia, 2017.

[29] B. Lantz, N. Handigol, B. Heller y V. Jeyakumar, «Mininet GitHub,» GitHub, [En línea]. Available: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet#what.

[30] Mininet, «Mininet Overview,» [En línea]. Available: http://mininet.org/overview/. [Último acceso: 29 07 2018].

[31] F. H. H. Institute, «High Efficiency Video Coding (HEVC),» [En línea]. Available: https://hevc.hhi.fraunhofer.de/. [Último acceso: 29 07 2018].

[32] P. Zhao, Y. Liu, J. Liu, A. Argyriou y S. Ci, «SSIM-basederror-resilientcross-layeroptimization for wirelessvideostreaming,» *Signal Processing: Image Communication,* pp. 36-51, 2015.

[33] U. V. Group, «testsequences,» Tampere University of Techonology, TUT, [En línea]. Available: http://ultravideo.cs.tut.fi/#testsequences. [Último acceso: 16 Julio 2018].

[34] K. Chih-Heng, «Software Defined Network (SDN)---Mininet Learning Guide,» Department of Computer Science and Information Engineering, National Quemoy

University, Kinmen, Taiwan, [En línea]. Available: http://csie.nqu.edu.tw/smallko/sdn/sdn.htm. [Último acceso: 21 07 2018].

[35] K. Chih-Heng, «ffmpeg streaming: udp vs. rtp when there are out of order packets,» Department of Computer Science and Information Engineering, National Quemoy University, Kinmen, Taiwan, [En línea]. Available: http://csie.nqu.edu.tw/smallko/sdn/ffmpeg_streaming_rtp.htm. [Último acceso: 21 07 2018].

[36] P. Lalbakhsh, B. Zaeri y Y.-P. P. Chen, «Using Dead Ants to improve the robustness and adaptability of AntNet routing algorithm,» *Journal of Network and Computer Applications*, vol. 44, pp. 196-211, 2014.

Apéndices

Apéndice A. Anteproyecto “Evaluación de la transmisión de flujos de video codificados de acuerdo con el estándar HEVC/H.265”.

(Ver archivo en medio digital)

Apéndice B. Artículo “Evaluación de la transmisión de flujos de video codificados de acuerdo con el estándar HEVC/H.265”.

(Ver archivo en medio digital)

Apéndice C. Código fuente de las topologías MININET.

(Ver archivo en medio digital)

Apéndice D. Código fuente de scripts para automatización de la plataforma propuesta, y parámetros de codificación en el codificador HM.

(Ver archivo en medio digital)

Apéndice E. StreamSim: Código fuente de script “chain.py” y resultados obtenidos.

(Ver archivo en medio digital)

Apéndice F. Resultados simulaciones para la transmisión del video ShakeNdry.

(Ver archivo en medio digital)