UNIVERSIDAD MIGUEL HERNÁNDEZ

Programa de Doctorado en

Tecnologías Industriales y de Telecomunicación

Aplicación de técnicas para la mejora de la transmisión de contenido multimedia en entornos de redes vehiculares

Autor: Pedro Pablo Garrido Abenza

Director: Dr. Manuel José Pérez Malumbres

Codirector: Dr. Pablo José Piñol Peral

Tesis Doctoral presentada en la Universidad Miguel Hernández

para la obtención del título de Doctor del Programa de Doctorado en

Tecnologías Industriales y de Telecomunicación

2019

La presente memoria de Tesis Doctoral, desarrollada en el Programa de Doctorado en Tecnologías Industriales y de Telecomunicación de la Universidad Miguel Hernández de Elche, se presenta en la modalidad de compendio de publicaciones acorde con la normativa de la Universidad Miguel Hernández de Elche.

A continuación se indican las publicaciones que conforman la Tesis Doctoral, incluyendo también los indicios de calidad.

- **Garrido Abenza, P.P.**, Malumbres, M.P., Piñol, P., López-Granado, Otoniel, *Source Coding Options to Improve HEVC Video Streaming in Vehicular Networks*, Sensors Journal (ISSN: 1424-8220) - Special Issue "Advances on Vehicular Networks: From Sensing to Autonomous Driving", Vol. 18, Issue 9, 2018. DOI: https://doi.org/10.3390/s18093107.
  Factor de impacto: 3.031, JCR: 15/61 (Q1) en la categoría "Instruments & Instrumentation"

En el Anexo B se incluye copia completa del artículo mencionado.

Autorización de Presentación de Tesis Doctoral por un Conjunto de Publicaciones

Director: Dr. D. Manuel José Pérez Malumbres

Codirector: Dr. D. Pablo José Piñol Peral

Título de la tesis: ***Aplicación de técnicas para la mejora de la transmisión de contenido multimedia en entornos de redes vehiculares***

Autor: D. Pedro Pablo Garrido Abenza

El director y codirector de la tesis reseñada AUTORIZAN SU PRESENTACIÓN EN LA MODALIDAD DE CONJUNTO DE PUBLICACIONES.

En Elche, a _____ de _____ de _____

Fdo. Dr. D. Manuel José Pérez Malumbres Fdo. Dr. D. Pablo José Piñol Peral

Informe del Coordinador de la Comisión Académica del Programa de Doctorado

Dr. D. OSCAR REINOSO GARCÍA, coordinador del Programa de Doctorado en Tecnologías Industriales y de Telecomunicación (TECNIT) de la Universidad Miguel Hernández de Elche,

CERTIFICA

Que el trabajo realizado por D. PEDRO PABLO GARRIDO ABENZA titulado ***Aplicación de técnicas para la mejora de la transmisión de contenido multimedia en entornos de redes vehiculares*** ha sido dirigido por el Dr. D. MANUEL JOSÉ PÉREZ MALUMBRES y codirigido por el Dr. D. PABLO PIÑOL PERAL, y se encuentra en condiciones de ser leído y defendido como Tesis Doctoral ante el correspondiente tribunal en la Universidad Miguel Hernández de Elche.

Lo que firmo para los efectos oportunos.

En Elche, a _____ de _____ de _____

Fdo. Dr. D. Oscar Reinoso García
Coordinador del Programa de Doctorado en
Tecnologías Industriales y de Telecomunicación (TECNIT)

Financiación

Este trabajo ha sido financiado parcialmente por el Ministerio de Economía y Competitividad mediante el proyecto TIN2015-66972-C5-4-R, así como con el proyecto RTI2018-098156-B-C54 del Ministerio de Ciencia, Innovación y Universidades, ambos cofinanciados con fondos FEDER (MINECO/FEDER/UE).

Además, también se utilizó financiación procedente de una subvención para la realización de proyectos de I+D+i desarrollados por grupos de investigación emergentes de la Conselleria de Innovación, Universidades, Ciencia y Sociedad Digital de la Generalitat Valenciana (GV/2019/020).

Por último, al doctorando se le concedió una ayuda económica en la Convocatoria de Ayudas y Bolsas de viaje para la difusión de resultados de investigación en el marco del programa de doctorado en TECNologías Industriales y de Telecomunicación (TECNIT) de la Universidad Miguel Hernández de Elche (RR 2426/18 UMH).

Agradecimientos

En primer lugar, expresar mi agradecimiento a mis directores, por su enorme paciencia e inestimable ayuda a lo largo de todo este proceso.

Gracias a todas las personas y cosas que en algún momento me hayan hecho la tarea un poco más fácil, y a aquellas que hayan hecho lo contrario, también.

Quisiera agradecer especialmente a mi familia, mujer e hijos, por estar siempre ahí, y también, pedirles disculpas por lo contrario.

Por último, gracias a todo el que lo lea, y al que no, también.

Tocar una nota equivocada, es insignificante.
Tocar sin pasión, es imperdonable.
Ludwig van Beethoven

Resumen

La transmisión de contenido multimedia en entornos de redes vehiculares, o Vehicular Ad-hoc NETworks (VANETs), puede tener un gran número de aplicaciones. Sin embargo, el uso de un canal inalámbrico compartido por todos los nodos de la red y el alto requerimiento de ancho de banda para el envío de vídeo pueden hacer que la calidad del vídeo recibido no sea aceptable, sobre todo en aplicaciones en tiempo real (p. ej., *video streaming*), donde se requiere, además, un bajo retardo y una variación de retardo acotada. A todos estos problemas se les añaden los continuos cambios en la topología debido a la movilidad de los nodos de la red, especialmente en este tipo de redes, donde la alta velocidad de los vehículos limita drásticamente el tiempo de comunicación entre ellos o con la infraestructura fija. Mediante el uso de técnicas de codificación o compresión de vídeo, como el estándar High Efficiency Video Coding (HEVC), se reduce la cantidad de datos necesaria para su almacenamiento, así como el ancho de banda requerido para su transmisión. Sin embargo, la calidad de vídeo percibida por el receptor puede verse muy afectada por los problemas ocurridos durante la transmisión, por lo que se requieren otros mecanismos que aumenten la robustez de las secuencias de vídeo transmitidas.

Esta Tesis Doctoral está enmarcada en la investigación de técnicas que permitan mejorar significativamente la calidad con la que un usuario percibe la recepción de un flujo de vídeo HEVC en entornos de redes vehiculares. Para la consecución de este objetivo, se comenzó con el desarrollo de ciertas herramientas *software* que permitieran la realización de los diversos experimentos necesarios. Como primer resultado se obtuvo la aplicación *GatcomSUMO*, que permite la generación de escenarios y movilidad de los vehículos para el simulador OMNeT++, el *framework* Veins y el simulador de tráfico SUMO. Posteriormente se desarrolló el entorno denominado *Video*

Delivery Simulation Framework over Vehicular Networks (VDSF-VN), con objeto de realizar la codificación de secuencias de vídeo mediante el codificador HEVC, la ejecución de simulaciones con los simuladores mencionados, la decodificación de las secuencias de vídeo recibidas, y la generación de gráficos con las estadísticas recogidas de forma automatizada, entre otras utilidades. Con todo lo anterior se llevaron a cabo diversos experimentos mediante simulación para evaluar la calidad de secuencias de vídeo transmitidas bajo diferentes condiciones de tráfico en la red. En primer lugar, se actuó sobre los parámetros del codificador de vídeo HEVC, utilizando modos de codificación con distintos patrones de refresco Intra y variando el número de fragmentos (*tiles*) de cada imagen de vídeo (*frame*). Los modos de codificación con mayor refresco Intra (p. ej., All Intra) requieren un mayor *bitrate* para su transmisión, sufren una mayor pérdida de paquetes, un mayor retardo (*delay*) y una alta variación de retardo (*jitter*). Sin embargo, en presencia de otro tipo de tráfico en la red (tráfico de fondo), se evidenció que son más robustos que otros modos con menor refresco Intra, en los que la pérdida de un cuadro (*frame*) puede tener un mayor efecto negativo en la calidad final del vídeo recibido debido a las interdependencias existentes entre *frames*. En cuanto al uso de *tiles*, se utilizaron distintos patrones uniformes junto con varios modos de codificación. Cuando se incrementa el número de *tiles* por *frame*, aunque se pierde algo de eficiencia de codificación, se consigue mayor robustez en presencia de tráfico de fondo, es decir, una mejor calidad del vídeo reconstruido. Sin embargo, el uso de *tiles* introduce una sobrecarga en el *bitstream* resultante y, puesto que a partir de un determinado valor el incremento de calidad no es significativo, con objeto de no saturar la red se determinó que lo óptimo es el uso de valores intermedios (entre 4 y 8 *tiles* por *frame*). En segundo lugar, y en combinación con los mecanismos anteriores, se utilizó la calidad de servicio (QoS) definida en el IEEE Std. 802.11p-2010, mediante la asignación de una mayor prioridad a los paquetes correspondientes a los *frames* de tipo I, y a los de todos los *frames* de vídeo. Los mejores resultados se obtuvieron en este último caso. En comparación a cuando no se utiliza QoS, las pérdidas correspondientes a los *frames* prioritarios se reducen considerablemente, sin perjudicar demasiado al resto de tráfico de menor prioridad existente en la red (tráfico de fondo), obteniéndose un mejor aprovechamiento del canal inalámbrico.

Abstract

The transmission of multimedia content in vehicular network environments, or Vehicular Ad-hoc NETworks (VANETs), is gaining high relevance in the automotive industry due to the attractive applications and services that may be developed. However, the use of a wireless channel shared by all the vehicles in a particular area, joined with the constrained requirements in terms of bandwidth and delay of video delivery applications, can make the quality of the received video not acceptable, especially in real time applications (e.g., video streaming), where low delay and bounded jitter are required. Furthermore, the topology of the network suffers frequent changes due to the mobility of the network nodes, and the high speed of the vehicles drastically limits the communication time between them or with the fixed infrastructure. By using video coding or compression techniques such as the High Efficiency Video Coding (HEVC) standard, the amount of data required for storage and transmission bandwidth are significantly reduced. However, the video quality perceived by the receiver can still be greatly affected due to transmission impairments, so other mechanisms are required to increase the robustness of the transmitted video sequences.

This Doctoral Thesis is focused in the research of techniques that allow to significantly improve the video quality perceived by a user receiving an HEVC video stream in a vehicular network environment. In order to achieve this objective, we began with the development of some software tools to properly carry out the required experiments. As a first result, the *GatcomSUMO* application was developed. This application allows the generation of network scenarios and vehicles mobility for the OMNeT++ simulator, the Veins framework and the SUMO traffic simulator. Subsequently, the environment called *Video Delivery Simulation Framework over*

Vehicular Networks (VDSF-VN) was also developed with the aim of creating a high detailed simulation environment including all the steps of a video delivery service: encoding of source video using the HEVC encoder, packetization of the encoded bitstream, simulation of network packets delivery, reassembling of the received packets, and finally, decoding of the reconstructed bitstream to play the video by the user. Also, several auxiliar utilities were developed like the ones focused in the automatization and statistics graphs generation, among others. With all of the above mentioned tools, various experiments were carried out by simulation to evaluate the quality of video sequences transmitted under different traffic conditions on the network. First, the parameters of the HEVC video encoder were tunned, in particular, using encoding modes with different Intra refresh patterns and varying the number of fragments (*tiles*) of each video image (*frame*). Encoding modes with higher Intra refresh rates (e.g., All Intra) require a higher bitrate for transmission, suffer a greater number of lost packets, a greater delay and delay variation (*jitter*) than other modes with less Intra refresh rates. However, in the presence of other traffic on the network (*background traffic*), it was evidenced that they are more robust than the other modes, as the loss of a frame can have a greater negative effect on the final quality of the received video due to the interdependencies between frames. Regarding the use of *tiles*, different uniform patterns were used together with various coding modes. As the number of *tiles* per *frame* increases, even though some encoding efficiency is lost, greater robustness is achieved in the presence of background traffic, that is, a better quality of the reconstructed video is achieved. However, the use of *tiles* introduces an overload in the resulting bitstream, and as from a certain value upwards the quality increase is not significant, in order to not saturate the network it was determined that the optimum is the use of intermediate values (between 4 and 8 *tiles* per *frame*). Secondly, and combined with the previous mechanisms, the Quality of Service (QoS) defined in IEEE Std. 802.11p-2010 was used, by assigning a higher priority to the packages corresponding to I frames only, and all the video frames. The best results were obtained in the latter case. In comparison to an scenario where QoS is not used, the losses corresponding to the priority frames are considerably reduced, without harming the rest of the lower priority traffic existing in the network (*background traffic*), obtaining a better use of the wireless channel.

Índice general

Capítulo 1

Introducción

1.1. Motivación

En la actualidad la recepción de TV a través de Internet (IPTV), y otros servicios OTT (Over-The-Top) ya es algo cotidiano, habiendo surgido diversas plataformas de transmisión de vídeo bajo demanda (televisión, películas, series, etc.), tanto a nivel internacional (Netflix, HBO, Amazon Prime Video, YouTube Premium), como a nivel nacional (Movistar+, Vodafone TV, o Filmin). Del mismo modo, también existen plataformas de música y radio a la carta, tales como Spotify, Deezer, Amazon Prime Music o Apple Music. La transmisión de datos multimedia, como puede ser audio y vídeo en *streaming* en entornos de red inalámbricos o Wireless LAN (WLAN) también se está haciendo cada vez más popular. Sin embargo, las redes inalámbricas tienen algunas características que las hacen diferentes de las redes cableadas o Local Area Networks (LAN). El uso de un medio inalámbrico hace que sea muy difícil garantizar una buena calidad de recepción del flujo de vídeo en los dispositivos, debido fundamentalmente a que se trata de un medio compartido por todos los nodos de la red y a la naturaleza cambiante de las características del canal inalámbrico. Se producen diversos fenómenos como la atenuación de la señal con la distancia (*path loss*) o el tiempo (*fading*), la presencia de obstáculos (*shadowing*), y los rebotes debidos a refracción o reflexión (*multipath*). Por otro lado, debido a la movilidad inherente de los nodos de la red, la topología de la red es cambiante, especialmente en el caso de las redes vehiculares o Vehicular Ad-hoc NETworks (VANETs), en donde

la alta velocidad de los vehículos limita el tiempo de comunicación. Una red vehicular está compuesta por vehículos móviles e infraestructura fija como antenas o Road-Side Units (RSUs). Por tanto, pueden darse dos tipos principales de comunicación, entre vehículos o Vehicle-to-Vehicle (V2V), y entre vehículos y la infraestructura o Vehicle-to-Infrastructure (V2I).

Las redes vehiculares tienen un gran potencial puesto que contribuyen a los llamados Intelligent Transportation Systems (ITS), proporcionando una serie de servicios en entornos urbanos e inter-urbanos, para conductores y pasajeros. Entre ellos, podemos mencionar la implementación de sistemas relacionados con la mejora de la seguridad en la conducción, sistemas de información del estado del tráfico o previsión meteorológica, acceso a Internet y aplicaciones de entretenimiento (*infotainment*). La transmisión de vídeo en redes vehiculares puede tener muchas aplicaciones, como la difusión de anuncios o información turística según nuestra posición, transmisión de vídeo en tiempo real o *video streaming*, videovigilancia, visualización del estado del tráfico en alguna zona, etc. Sin embargo, transmitir vídeo con calidad suficiente en estos entornos es muy complicado, puesto que, además de los problemas típicos de las redes inalámbricas, se requiere un gran ancho de banda (*bitrate*), bajo retardo o latencia (*delay*) y una variación de retardo (*jitter*) acotada.

El uso de técnicas de codificación o compresión de vídeo permite reducir la cantidad de datos necesaria para su almacenamiento, así como el ancho de banda requerido para su transmisión. En los últimos años han surgido diversos estándares de codificación de vídeo, como High Efficiency Video Coding (HEVC) [3], que mejora las tasas de compresión de su predecesor H.264/AVC (Advanced Video Coding) [1], manteniendo la misma calidad subjetiva de vídeo percibida por el usuario final. Sin embargo, aunque mediante la codificación de vídeo se reduce de forma considerable la cantidad de información a transmitir, la calidad de vídeo percibida por el receptor puede verse muy afectada por los problemas ocurridos durante la transmisión.

Esta Tesis Doctoral está enmarcada en la investigación de técnicas que permitan mejorar significativamente la calidad con la que un usuario percibe un flujo de vídeo HEVC transmitido en entornos de redes vehiculares, haciendo más robusta su transmisión.

1.2. Objetivos

El objetivo general de la presente Tesis Doctoral es ofrecer una transmisión de vídeo robusta y eficiente en redes vehiculares. En concreto, los objetivos específicos son los siguientes:

1. Desarrollar un entorno de trabajo (*framework*) que permita la simulación y evaluación de la transmisión de vídeo en escenarios de redes vehiculares.

2. Analizar y combinar diferentes técnicas que permitan mantener la calidad de las secuencias de vídeo durante su transmisión a través de las redes vehiculares.

3. Determinar los parámetros óptimos para la codificación de vídeo que permitan incrementar su robustez durante la transmisión, esto es, que no se vea afectado por las interferencias en la transmisión, con una calidad percibida por un usuario final aceptable.

4. Adaptar el sistema de calidad de servicio (QoS) propuesto en el estándar IEEE Std. 802.11p-2010 al entorno de simulación en escenarios VANET, definiendo las estadísticas de red necesarias a nivel de la capa MAC, desglosadas por cada una de las cuatro categorías de servicio (AC).

5. Definir una arquitectura de protección de vídeo combinando todas aquellas técnicas que mejores resultados ofrecen en entornos de red VANET.

1.3. Publicaciones

Como resultado de este trabajo de Tesis se han obtenido diversas publicaciones en congresos internacionales, congresos nacionales, y revista. De todas ellas, la publicación principal que sustenta esta Tesis presentada en la modalidad de compendio de publicaciones es el siguiente artículo en la revista *Sensors*, la cual está indexada en el Journal Citation Reports (JCR), clasificada en el primer cuartil (Q1) en su año de publicación. Esta publicación se incluye íntegramente en el Anexo B.1, y sus datos bibliográficos completos son los siguientes:

- **Garrido Abenza, P.P.**, Malumbres, M.P., Piñol, P., López-Granado, Otoniel, *Source Coding Options to Improve HEVC Video Streaming in Vehicular Networks*, Sensors Journal (ISSN: 1424-8220) - Special Issue "Advances on Vehicular Networks: From Sensing to Autonomous Driving", Vol. 18, Issue 9, 2018. DOI: https://doi.org/10.3390/s18093107
  Factor de impacto: 3.031, JCR: 15/61 (Q1) en la categoría "Instruments & Instrumentation"

Esta publicación muestra los resultados de experimentar con los distintos parámetros del codificador de vídeo HEVC con objeto de averiguar de qué forma conseguir que la secuencia de vídeo codificada sea más robusta a interferencias con otros tipos de tráfico en la red. En concreto, se experimentó con diversos modos de codificación de refresco Intra (*Intra refresh*), utilizando diferente proporción de *frames* de tipo I. Ello se combinó con el uso de técnicas basadas en fragmentación de los *frames* (*tiles*) utilizando siete patrones distintos (1×1, 1×2, 2×2, 2×3, 2×4, 2×5, 4×4), es decir, utilizando la siguiente secuencia de número de *tiles* por *frame*: 1, 2, 4, 6, 8, 10, y 16. Puesto que con ambas técnicas se incrementa el *bitrate* de la secuencia codificada, el objetivo de este trabajo era averiguar el nivel óptimo de refresco Intra, así como el número de *tiles* por *frame* a utilizar con objeto de mantener una calidad de vídeo aceptable en el envío de vídeo. En concreto, se concluyó que las dos técnicas se complementan satisfactoriamente, aconsejándose el uso de los modos de codificación con mayor refresco Intra, como AI, combinado con el uso de un número de *tiles* por *frame* de 4 ó 6.

Los experimentos de este trabajo, así como el análisis de los resultados, se realizaron utilizando un entorno de trabajo que cubre el proceso completo de simulación de transmisión de vídeo denominado *Video Delivery Simulation Framework over Vehicular Networks* (VDSF-VN), el cual también se describe brevemente.

1.4. Estructura de la Tesis

La presente memoria se estructura de la siguiente forma:

- El Capítulo 2 presenta la metodología utilizada para el desarrollo de la investigación, así como las distintas herramientas *software* empleadas. En concreto, se justifica el uso de la técnica de simulación, y se describen brevemente los diferentes simuladores y codificadores de vídeo empleados.
- El Capítulo 3 se divide en dos partes. Primero se justifica el desarrollo de ciertas aplicaciones *software* y se describe brevemente su funcionalidad. Después, se exponen de manera global los experimentos realizados, así como los resultados obtenidos. Una discusión más detallada sobre los mismos puede encontrarse en la publicación citada en la sección anterior.
- Por último, en el Capítulo 4 se resumen las conclusiones generales, así como las líneas abiertas de investigación.

Capítulo 2

Material y métodos

En este capítulo, tras exponer la metodología empleada para la investigación, se describe el material utilizado en los experimentos, que básicamente consiste en herramientas de simulación y obtención de resultados (sección 2.1), así como *software* para codificación de vídeo (sección 2.2).

Para analizar cualquier problema científico existen tres técnicas: métodos analíticos, métodos experimentales, y simulación [16] [17]. Los métodos analíticos se basan en la creación de un modelo matemático del sistema; sin embargo, se ha demostrado que las redes inalámbricas son matemáticamente intratables debido tanto a la complejidad de las capas de la pila de protocolos, como a su interacción entre ellas [38]. Por otro lado, los métodos experimentales consisten en el despliegue de un sistema y realizar mediciones reales (*testbed*). Aunque se han desarrollado diversos *testbed*, la construcción de estas plataformas es muy compleja y excesivamente cara para el caso de las redes inalámbricas en general, y mucho más para las redes vehiculares debido a sus particularidades (espacio requerido, movilidad, toma de medidas, dificultad para la repetición de experimentos, etc.).

Por tanto, dada la naturaleza las redes vehiculares, el método de trabajo principal para la obtención de resultados es el uso de la técnica de simulación; esta técnica es la más utilizada en este campo puesto que permite la realización de diferentes escenarios de forma sencilla, así como la recogida de estadísticas y reproducibilidad posterior [13] [28].

2.1. Herramientas de simulación y gráficos

En el caso particular de una red vehicular, los datos recibidos por un vehículo pueden determinar su movilidad. Sería el caso, por ejemplo, cuando se recibe un mensaje de aviso indicando de un atasco o un accidente ocurrido en su ruta que mantiene una calle cortada al tráfico. De este modo, el conductor, o el propio vehículo, podrá anticiparse y buscar una ruta alternativa para evitar el atasco. Por tanto, la simulación de transmisión de datos en redes vehiculares requiere de dos componentes, un simulador de tráfico para controlar la movilidad de los vehículos, y un simulador de red para la comunicación entre los mismos. De las distintas alternativas existentes, en este trabajo se seleccionaron los simuladores OMNeT++ v4.6 como simulador de red y Simulation of Urban MObility (SUMO) v0.25.0 como simulador de tráfico, acoplados mediante el VEhicles In Network Simulation (Veins) v4.4. Todos ellos son de código abierto, lo cual es un factor clave que facilita la reproducibilidad de la investigación.

Por otro lado, para la simulación de redes vehiculares se necesita una red de carreteras basada en mapas reales por la que se moverán los vehículos. SUMO permite importar su red de carreteras desde diversos formatos, entre los que se incluye OpenStreetMap (OSM) [7], los cuales tienen suficientes datos y calidad para los experimentos a realizar, siendo además de uso abierto. Los mapas de OSM incluyen las edificaciones, que son reconocidas por SUMO, y que Veins identifica como obstáculos a la hora de calcular la propagación de la señal en las comunicaciones. Todos los mapas descargados desde OpenStreetMap están basados en el sistema de coordenadas World Geodetic System de 1984 (WGS84), el cual es el mismo elipsoide utilizado por el sistema de posicionamiento Global Positioning System (GPS) de Estados Unidos, y muy aproximado al utilizado por el Global Navigation Satellite System (GLONASS) desarrollado por la Unión Soviética. Actualmente, tanto GPS como GLONASS son sistemas globales de navegación por satélite o Global Navigation Satellite Systems (GNSS) muy utilizados, tanto en vehículos como en teléfonos inteligentes (*smartphones*), por lo que es factible superponer trazas de recorridos GPS reales (*tracks*) sobre mapas de OSM (archivos .gpx). De este modo, es posible comprobar la validez de la zona descargada de forma visual utilizando diversas

aplicaciones tales como JOSM (Java OpenStreetMap) [25] o JGPSTrackEdit [14].

Por último, para el análisis de los resultados se han generado gráficos mediante el programa estadístico R [23] (junto con el paquete rplotengine [6]), así como con Gnuplot [36], ambos multiplataforma y de código abierto.

2.2. Herramientas de codificación de vídeo

En este trabajo se ha utilizado el estándar High Efficiency Video Coding (HEVC) para la codificación de las secuencias de vídeo. Existen varios codificadores y decodificadores HEVC, tanto de uso comercial como de código abierto. Entre los de código abierto podemos citar el *software* de referencia para los codificadores HEVC, que es el HEVC Test Model (HM) [10] del Joint Collaborative Team on Video Coding (JCT-VC), así como otros que se han basado en éste. Se trata de un codificador que consigue la mejor eficiencia de codificación, aunque no tiene un buen rendimiento computacional. Otros codificadores se basan en HM, habiendo realizado ciertas optimizaciones en el código. Por ejemplo, ces265 [18], está desarrollado en C++ y soporta multi-hilo para procesamiento en paralelo, x265 [19] utiliza ensamblador y técnicas multi-hilo, y f265 [30] está implementado en C y ensamblador. Otro codificador HEVC es Kvazaar [31] [32], el cual está implementado en C, y es de código abierto, con una estructura modular que facilita su portabilidad y paralelización; consigue una eficiencia de compresión muy similar a HM pero con una mayor velocidad y menor consumo de recursos. Por último, también existe HomerHEVC (HEVC Open MPEG EncodeR) [11], que es un codificador en tiempo real multiplataforma, así como decodificadores HEVC de código abierto como el OpenHEVC [29], desarrollado para fines de investigación y demostración. En este trabajo de Tesis se ha utilizado el codificador y el decodificador del *software* de referencia HM.

En cuanto a las secuencias de vídeo, éstas pueden ser generadas por el propio usuario, o ser convertidas a formato YUV desde vídeos ya existentes en otros formatos con utilidades como *ffmpeg* o *mencoder* (incluida en MPlayer). También pueden ser descargadas desde diversos repositorios en Internet tales como Video Test Media [4], Video Trace Library [37], o las incluidas en las HEVC Common Test

Conditions (CTC) [5]. En concreto, la secuencia de vídeo utilizada fue 'BasketballDrill', perteneciente a estas últimas (Figura 2.1). Tiene una resolución de 832×480 píxeles y una duración de 10 segundos. Aunque la secuencia original consta de 500 *frames* a una tasa de 50 *frames* por segundo (fps), ésta fue submuestreada a la mitad, quedando con una longitud de 250 *frames* a una tasa de 25 fps, con objeto de reducir a la mitad el ancho de banda requerido para su transmisión (Tabla 2.1).

Figura 2.1: Secuencia de vídeo BasketBallDrill

Parámetro	**Valor**
Nombre	BasketballDrill
Resolución	832×480 pixeles
Profundidad de color	8 bits
Duración	10 s
Longitud	500 → 250 *frames*
Tasa	50 → 25 fps

Tabla 2.1: Parámetros de la secuencia de vídeo BasketBallDrill

Como resultado de la codificación de las secuencias de vídeo se obtiene una serie de imágenes o *frames*, los cuales pueden ser de tipo I, P, B. Puesto que un *frame* puede ser mayor que el tamaño máximo de paquete de datos que la red puede transmitir o Maximum Transmission Unit (MTU), especialmente los *frames* de tipo I, para su transmisión por la red es necesario encapsular cada *frame* en una serie de paquetes de red, es decir, realizar un proceso de paquetización, utilizando el protocolo Real-time Transmission Protocol (RTP) [33] [24]. En un trabajo de Tesis Doctoral anterior [22] se realizaron las modificaciones necesarias al codificador del

software HM para incluir un paquetizador RTP, generando como resultado, además de la secuencia codificada (*bitstream*), el archivo de traza de vídeo necesario para realizar la simulación con OMNeT++ y Veins. La traza de vídeo consiste en un fichero de texto con información sobre cada uno de los paquetes que componen cada *frame* de la secuencia de vídeo, incluyendo: número de paquete correlativo, tipo de *frame* al que pertenece (I, P, o B), instante de reproducción, tamaño del paquete (*bytes*), el número de fragmento dentro del *frame* y el número total de fragmentos del *frame* actual. Asimismo, también se modificó el decodificador para integrar el correspondiente depaquetizador y poder reconstruir la secuencia de vídeo codificada, así como otras modificaciones necesarias para que no se interrumpiese ante la ausencia de ciertos paquetes no recibidos. Todo ello se tomó como punto de partida para la presente Tesis Doctoral, modificando el formato de las trazas de vídeo y desarrollando otras herramientas para facilitar su manejo y automatizar el ciclo de trabajo con el fin de realizar experimentos a mayor escala, las cuales se detallan en la siguiente sección.

Capítulo 3

Resultados

En este capítulo se describen los resultados obtenidos, incluyendo las aplicaciones *software* desarrolladas (sección 3.1), así como la discusión de los resultados de los experimentos realizados (sección 3.2).

3.1. Herramientas desarrolladas

Durante el desarrollo del presente trabajo se detectó que las herramientas existentes para la realización de todo el ciclo de trabajo tenían un uso complejo, con una curva de aprendizaje bastante pronunciada. A pesar de la existencia de abundante documentación por parte de los desarrolladores de las herramientas citadas en las secciones anteriores, el tiempo necesario para la configuración de los archivos necesarios hasta su correcto funcionamiento era alto. Para la ejecución de las herramientas era necesario el uso de archivos por lotes o *scripts*, cuya funcionalidad es limitada y el lenguaje utilizado para escribir estos archivos depende del sistema operativo utilizado. Tienen además el inconveniente de que se ejecutan de forma secuencial, sin sacar provecho de multiproceso de forma sencilla.

Por todo ello, se consideró conveniente el desarrollo de alguna aplicación en un lenguaje de alto nivel con el fin de automatizar ciertas tareas y poder centrarse más en la tarea de investigación propiamente dicha. Se pretendía que las herramientas desarrolladas fueran fáciles de manejar, con un interfaz gráfico (GUI) intuitivo, y que

estuvieran disponibles para cualquier plataforma, todo ello con objeto de que pudieran ser utilizadas por cualquier otro investigador que realizase tareas similares. Por este motivo, se escogió el lenguaje de programación Java, ya que existe una máquina virtual Java o Java Virtual Machine (JVM) para diversas plataformas, al menos para aquellas plataformas para que las que también se han portado los simuladores utilizados: Windows, Linux, y MacOS X. Como resultado de estos desarrollos se obtuvo un conjunto de aplicaciones con el que se pretende cubrir el hueco existente para llevar a cabo este tipo de experimentos en cualquier plataforma. Se comenzó con el desarrollo de la aplicación *GatcomSUMO*, para continuar con un entorno de trabajo denominado *Video Delivery Simulation Framework over Vehicular Networks* (VDSF-VN), los cuales se describen en esta sección.

Para realizar una simulación con SUMO es necesario disponer de la red de carreteras (*Network*) por la que se moverán los vehículos, así como los archivos de demanda de tráfico (*Traffic demand*), que definen los vehículos y las rutas que seguirán los mismos. Estos y otros archivos utilizan el formato XML (Extensible Markup Language), que deben cumplir con una estructura y sintaxis concreta, así como ciertas restricciones. Además, es necesario ejecutar ciertos programas desde línea de órdenes (`netconvert`, `polyconvert`, `netgenerate`, `duarouter`, etc.), cuya sintaxis y posibles argumentos es necesario conocer. Todas estas tareas son muy propensas a errores y exigen experiencia previa, especialmente cuando la simulación se realiza acoplando SUMO con el simulador OMNeT++. En concreto, con éste último se requiere también un archivo de configuración para la definición de los diversos parámetros de los objetos involucrados en la simulación (omnetpp.ini), así como otros archivos en formato NED o XML para el enlace con SUMO.

Con la motivación de simplificar y automatizar la creación de los escenarios cuando se utilizan estos simuladores se desarrolló la aplicación gráfica *GatcomSUMO*. Existen varias herramientas que hacen SUMO más accesible a usuarios sin experiencia de programación y permiten superar algunos inconvenientes a la hora de generar los archivos de configuración como *TrafficModeler* [21], *SUMOPy* [26], o OSMWebWizard. Sin embargo, la aplicación *GatcomSUMO* desarrollada se adapta mejor al campo de las simulaciones VANET, especialmente cuando se utiliza SUMO conjuntamente con el simulador OMNeT++. Para empezar, los archivos de rutas

generados con SUMO, aunque sean válidos para SUMO podrían generar un error de ejecución durante la simulación con OMNeT++ en ciertos casos, mientras que si se utiliza *GatcomSUMO* se evitan dichos casos. Además de generar los archivos necesarios para ambos simuladores, la aplicación incorpora ciertas utilidades como un conversor entre los distintos sistemas de coordenadas utilizadas por ambos simuladores, y coordenadas geodésicas o métricas (UTM); de este modo se permite la colocación de elementos fijos tales como Road-Side Unit (RSU) en una posición exacta del escenario, independientemente de las calles o carreteras de la red. En cuanto a los nodos móviles, la aplicación también permite la generación de rutas (demanda de tráfico) de forma explícita o aleatoria, y para cualquier número de vehículos, evitando la creación de rutas inconexas. Asimismo, se integran ciertas utilidades para el cálculo de la atenuación de la señal en las comunicaciones mediante diferentes fórmulas como Free-Space Path Loss (FSPL) o Two-Ray, útiles para especificar el rango de comunicación de los nodos en el archivo de configuración.

En los experimentos realizados, los servidores de vídeo envían secuencias de vídeo en formato YUV previamente codificadas mediante el codificador HEVC. La secuencia codificada (*bitstream*) se compone de una serie de imágenes o *frames*, los cuales pueden ser de tipo I, P, B. Puesto que un *frame* puede ser mayor que el tamaño máximo de paquete de datos que la red puede transmitir o MTU, especialmente los *frames* de tipo I, es necesario encapsular cada *frame* en una serie de paquetes de red antes de su transmisión, es decir, realizar un proceso de paquetización RTP [33] [24]. Además, el simulador no transmite realmente el contenido de cada paquete sino que utiliza archivos de trazas de vídeo, es decir, archivos de texto con información de los distintos paquetes (instante de reproducción, tipo de *frame* al que corresponde, tamaño, etc.), transmitiendo paquetes del tamaño correspondiente en el instante adecuado. Una vez finalizada la simulación hay que realizar el proceso inverso, es decir, reconstruir la secuencia de vídeo codificada a partir de la información de las trazas de vídeo recibidas por los clientes mediante el depaquetizador, y decodificar el vídeo reconstruido para obtener de nuevo la secuencia de vídeo en el mismo formato que el original (YUV), el cual estará deteriorado debido a los paquetes perdidos durante la transmisión. Por último, hay que procesar y analizar los resultados obtenidos, tanto las estadísticas de red obtenidas durante la simulación, como la calidad

perceptual de los vídeos reconstruidos mediante el cálculo de alguna métrica como el Mean Square Error (MSE) o Peak Signal-to-Noise Ratio (PSNR) [2]. Como se puede observar, es necesario hacer uso de una serie de elementos para llevar a cabo un experimento de este tipo. En la bibliografía se han encontrado algunos entornos desarrollados con este propósito, tales como EvalVid [12] y algunos otros derivados de éste. Sin embargo, los *frameworks* analizados están orientados para su uso con otros simuladores (ns-2, ns-3, etc.), utilizan otro estándar de codificación de vídeo anterior (MPEG4, H.264/AVC), o no incluyen los modelos de movilidad necesarios para la simulación de redes inalámbricas. Es decir, no se encontró ningún *framework* que permitiese la simulación de la transmisión de secuencias de vídeo codificadas con HEVC para su uso con OMNeT++ y Veins, y la modificación de los existentes no es trivial.

Con objeto de cubrir el hueco existente a la hora de la realización de estos experimentos de forma sencilla, así como el análisis posterior de los resultados obtenidos, se desarrolló el entorno *Video Delivery Simulation Framework over Vehicular Networks* (VDSF-VN), el cual está compuesto por varios elementos. En primer lugar, el entorno incluye una aplicación denominada *GatcomVideo* que consiste en un interface gráfico o *front-end* desde donde se definen y ponen en ejecución las distintas tareas. Sin embargo, la aplicación delega gran parte de su trabajo en otras herramientas externas o *back-end*, como el codificador y decodificador de vídeo, paquetizador y despaquetizador, así como los simuladores mencionados anteriormente. El ciclo de trabajo de VDSF-VN se puede estructurar en tres pasos o fases: (1) pre-proceso, (2) simulación, y (3) post-proceso, que se describen brevemente a continuación junto con las herramientas externas concretas que se utilizan en cada caso.

En la fase de pre-proceso se preparan las secuencias de vídeo en formato YUV para que puedan ser utilizadas en la simulación. Primero se codifican con el estándar HEVC mediante el *software* de referencia HM. A la hora de codificar una secuencia de vídeo se pueden especificar diversos parámetros como el modo de codificación, que marca la estructura de cada grupo de imágenes, o Group of Pictures (GoP): All-Intra, Low-Delay P, ... También se puede especificar si cada *frame* se divide en regiones denominadas *slices* o *tiles*. Los *slices* son regiones contiguas que se pueden codificar y decodificar de forma independiente, por lo que se obtiene una mayor resistencia

a los errores en la transmisión. Al utilizar varios *slices* por *frame*, la pérdida de un paquete de red ya no afecta a un *frame* completo sino sólo al *slice* al que pertenece. Sin embargo, el uso de *slices* introduce una sobrecarga en el *bitstream* debido a las cabeceras necesarias. Además, se pierde eficiencia de codificación al no poder aprovechar tanto la predicción espacial ni la predicción de vectores de movimiento. Por otro lado, con el estándar HEVC apareció un nuevo concepto denominado *tiles*, que consiste en dividir un *frame* en varias zonas rectangulares. Los *tiles* comparten cierta información de cabecera, por lo que la sobrecarga añadida al *bitstream* es menor, y gracias a su forma rectangular también son más eficientes que los *slices*. Aunque los *tiles* también se pueden codificar y decodificar de forma independiente en paralelo, no aportan la resistencia proporcionada por los *slices*, puesto que no son totalmente independientes. Con objeto de obtener lo mejor de ambas estructuras, en [22] se propuso el concepto de *tileslice*, que consiste en dividir un *frame* en zonas rectangulares (*tiles*), e insertar cada *tile* en un único *slice*. En lo siguiente se utilizará el término *tile*, aunque en realidad se utiliza este concepto de *tileslice*. Junto con el modo de codificación y número de *tiles* por *frame*, también se puede especificar el factor de cuantificación o Quantization Parameter (QP). Mediante este valor QP (0..51) se puede controlar la calidad o el *bitrate* necesario para la transmisión de la secuencia de vídeo codificada.

La codificación de una secuencia de vídeo es un proceso que tiene un alto coste computacional, y puede ser necesario realizarlo múltiples veces utilizando diferentes secuencias de vídeo, archivos de configuración, y valores de QP. El *framework* VDSF-VN se diseñó teniendo presente que hiciera uso de multitarea siempre que fuese posible y de forma transparente para el usuario, es decir, aprovechando los recursos de la máquina utilizada (*multi-core*); de esta manera, el tiempo requerido se reduce drásticamente. Además, se incluye también una utilidad que permite la búsqueda del valor de QP necesario para que la secuencia codificada tenga una calidad (valor de PSNR) o un *bitrate* determinado, la cual, en lugar de hacer una búsqueda lineal, realiza una búsqueda binaria con un coste computacional de $\mathcal{O}(log_2 N)$. Además de obtener las secuencias codificadas, en la fase de pre-proceso también se realiza la paquetización RTP y la generación de sus correspondientes archivos de traza. Como se ha comentado anteriormente, para la paquetización RTP fue necesario modificar

el codificador del *software* HM, lo cual se hizo en una Tesis Doctoral anterior [22].

En cuanto a la fase simulación, aunque podrían ejecutarse manualmente desde el entorno de OMNeT++ o desde línea de órdenes, mediante *GatcomVideo* se muestran los parámetros de las distintas simulaciones (a partir del archivo de configuración omnetpp.ini), y se pueden seleccionar aquellas de interés, ejecutándose en paralelo (hasta el número máximo de procesadores del computador). En cualquier caso, se requiere los simuladores SUMO y OMNeT++, así como el *framework* Veins. Además, fue necesario realizar una serie de modificaciones al proyecto de Veins que, por conveniencia, se hicieron de forma separada en un nuevo proyecto denominado "ppp_qos" que referencia al proyecto original de Veins, y que contiene sólo los ficheros modificados, así como otros nuevos. En resumen, las modificaciones se realizaron para disponer de las siguientes funcionalidades: (1) lectura de trazas de vídeo a transmitir por parte de los servidores de vídeo, (2) escritura de los archivos de trazas recibidas por parte de los clientes de vídeo (utilizados en la fase de post-proceso para poder reconstruir el vídeo recibido), (3) envío de paquetes como tráfico de fondo a distintas tasas, y (4) recogida de nuevas estadísticas. Entre las nuevas estadísticas definidas se incluyen las de la subcapa MAC ya existentes, pero desglosadas por categoría de servicio o Access Category (AC), así como otras nuevas tales como el porcentaje de ocupación del canal de transmisión o Channel Busy Ratio (CBR), el porcentaje de ocupación y tamaño medio de cada una de las cuatro colas AC, tanto para el canal de control (CCH) como para el canal de servicio (SCH). A nivel de la capa aplicación se han definido también estadísticas como carga transmitida o *load* (en bits por segundo y paquetes por segundo), carga recibida o *goodput*, *end-to-end delay*, *jitter*, y Packet Delivery Ratio (PDR), entre otras. Por último, también se han definido otras estadísticas relacionadas con la topología y movilidad de los vehículos, como el número de vecinos o la distancia entre pares concretos de nodos, las cuales permiten la validación de los escenarios implementados. La mayoría de estas estadísticas se recogen tanto a nivel global de la red como de forma individual por cada nodo, y de forma escalar (totales) o vectorial (valores a lo largo de toda la simulación). Con objeto de reducir espacio de almacenamiento, se implementó un nuevo tipo de estadísticas vectoriales que se registran únicamente promediadas cada segundo, permitiendo la realización de estudios por zonas o intervalos de tiempo

concretos de forma más óptima.

Tras la ejecución de cada una de las simulaciones se obtiene un archivo de resultados con estadísticas escalares (.sca) y otro con estadísticas vectoriales (.vec). A la hora de analizar los resultados obtenidos, aunque puede utilizarse el propio entorno del simulador OMNeT++, la aplicación *GatcomVideo* incluye una utilidad que permite definir cualquier número de gráficos incluyendo cualquiera de las estadísticas recogidas mediante un asistente o editando un archivo de texto, ya sean de una misma simulación, de varias simulaciones del mismo conjunto de simulaciones o diferente, o de un mismo o distinto experimento (configuración del archivo omnetpp.ini). Una vez definidos los gráficos, puede seleccionarse una zona concreta de estudio o el total de la simulación, si se quiere incluir intervalos de confianza (90 %, 95 %, 98 %, 99 %) en el caso de haber definido varias repeticiones en las simulaciones, así como los puntos por pulgada (dpi). Con todo ello, se pueden generar los gráficos correspondientes de forma desasistida utilizando tanto el paquete estadístico R (junto con el paquete 'rplotengine') como mediante Gnuplot. Como resultado se obtienen archivos gráficos en formato *raster* (.png) y vectorial (.eps), que pueden visualizarse desde la propia aplicación, así como los datos numéricos utilizados.

La fase de post-proceso consiste en analizar la calidad de las secuencias de vídeo recibidas, comparándolas con la secuencia original codificada. En los experimentos realizados, los servidores retransmiten una y otra vez una secuencia de vídeo. Por otro lado, los clientes de vídeo pueden recibir varias de estas secuencias, por lo que deben trocearlas. La aplicación identifica las distintas secuencias (en función de un número de paquete), y el usuario puede seleccionar aquellas que se hayan recibido en una zona (instante) de interés (zonas de buena recepción, zonas conflictivas, etc.). Puesto que lo que se transmite no es la secuencia de vídeo sino una simulación basada en la información del archivo de traza, el primer paso consiste en reconstruir el vídeo recibido a partir de la información de los paquetes recibidos para obtener los *frames* de la secuencia original. El vídeo reconstruido no estará completo, pues es posible que algunos de los paquetes que componen los *frames* se hayan perdido durante su transmisión. Por ello, al decodificar el vídeo reconstruido, los *frames* (o *tiles*) que no se hayan recibido completamente no podrán decodificarse. Como resultado de este proceso se obtiene un vídeo en el mismo formato que el original (YUV),

pero distorsionado, junto con estadísticas de paquetes perdidos y *frames* perdidos. Mediante la comparación entre la secuencia de vídeo original y la secuencia recibida distorsionada se puede determinar cómo afecta el estado de la red a su calidad percibida por un usuario final.

Para evaluar la calidad perceptual podrían realizarse encuestas subjetivas con personas reales que valoren la calidad del vídeo recibido según las recomendaciones ITU-T [8] [9] con una escala de 1 a 5, promediándose para obtener un valor Mean Opinion Score (MOS). Sin embargo, la realización de este tipo de tests tiene diversos inconvenientes, como por ejemplo, su coste. Por ello, se suelen utilizar unas métricas Quality-of-Experience (QoE) de calidad de vídeo como el Mean Squared Error (MSE) y Peak Signal-to-Noise Ratio (PSNR). Aunque existen otras métricas, en los experimentos realizados en este trabajo se han utilizado estas dos, ya que son las más utilizadas, sencillas y eficientes de calcular. El MSE (ecuación 3.1) se calcula para cada *frame* n promediando las diferencias cuadradas de intensidad de cada pixel (i, j) del mismo *frame* en la secuencia original (Y_S) y distorsionada o reconstruida (Y_D), donde $i \in 1..N_{col}$ y $j \in 1..N_{row}$, siendo N_{col} y N_{row} el ancho y alto de los *frames* (en píxeles). El PSNR puede calcularse a partir del MSE para el componente de la luminancia (Y), y el valor máximo (V_{peak}), como se muestra en la ecuación 3.2.

$$MSE = \sqrt{\frac{\sum_{i=1}^{N_{col}} \sum_{j=1}^{N_{row}} [Y_S(n,i,j) - Y_D(n,i,j)]^2}{N_{col} \cdot N_{row}}} \tag{3.1}$$

donde:

N_{col} = ancho de las imágenes a comparar
N_{row} = alto de las imágenes a comparar
$Y_S(n,i,j)$ = luminancia (Y) del pixel (i, j) del *frame* n del vídeo de referencia
$Y_D(n,i,j)$ = luminancia (Y) del pixel (i, j) del *frame* n del vídeo reconstruido

$$PSNR(n)_{dB} = 20 \cdot log_{10}\left(\frac{V_{peak}}{MSE}\right) \tag{3.2}$$

donde:

$V_{peak} = 2^k - 1$ (valor máximo)
k = número de bits por pixel

Las ecuaciones anteriores permiten medir la similitud entre dos imágenes, por lo que en el caso de secuencias de vídeo es necesario comparar uno a uno cada par de *frames* del vídeo original de referencia (Y_S) y el vídeo reconstruido (Y_D), y obtener el valor medio del componente Y (luminancia). Si no hay pérdida de paquetes durante la transmisión, ambas secuencias de vídeo deberán tener el mismo valor en estas métricas.

Por último, al igual que para las estadísticas de red, la aplicación permite la generación de gráficos definidos por el usuario, en este caso, eligiendo entre las siguientes estadísticas: paquetes perdidos en valor absoluto o porcentaje (`LP` o `LP%`), *frames* perdidos en valor absoluto o porcentaje (`LF` o `LF%`), *bitrate* (`BR`) del video que transmite el servidor, su valor PSNR (`PSNR_TX`), o el PSNR del vídeo que recibe el cliente (`PSNR_RX`), valor final o *frame-a-frame*.

3.2. Discusión de los resultados

Con el *framework* VDSF-VN propuesto se han realizado diversos experimentos para evaluar el rendimiento de la transmisión de secuencias de vídeo en escenarios de redes vehiculares. Las técnicas con las que se ha experimentado pueden clasificarse en dos grupos: (1) actuando sobre los parámetros del codificador de vídeo HEVC (*source-coding*), y (2) utilizando calidad de servicio (QoS). En cualquier caso, se evalúa la robustez de la secuencia de vídeo cuando se transmite en presencia de diferentes niveles de otro tráfico existente en la red (tráfico de fondo o *background*). Aunque se tienen en cuenta las estadísticas de red recogidas durante las simulaciones (*end-to-end delay*, *goodput*, *packet delivery ratio*, etc.), la métrica principal para medir la calidad de vídeo percibida es el valor PSNR del vídeo recibido y reconstruido. Para ciertos casos, dependiendo del modo de codificación, del nivel de tráfico de fondo, etc., la calidad se mantiene por encima de un umbral aceptable, mientras que para otros, la calidad no llega a dicho umbral. A continuación se resumen de forma global los resultados obtenidos mediante las técnicas mencionadas.

Con respecto al primer grupo de técnicas utilizadas, las opciones del codificador con las que se ha experimentado son, principalmente, el modo de codificación con

distinto patrón de refresco de *frames* de tipo I (*Intra refresh*), y el uso de *tiles* por *frame*. Con estos experimentos se pretendía encontrar el nivel óptimo de refresco Intra, así como el número de *tiles* por *frame* requerido para garantizar una buena calidad de vídeo recibida.

Se comenzó analizando la técnica de refresco Intra por sí sola, es decir, utilizando un único *tile* por *frame* (1×1) e incrementando el tráfico de fondo. La pérdida de un único paquete implica que el *frame* correspondiente no pueda decodificarse, aunque el resto de paquetes se hayan recibido correctamente, por lo que tiene un gran impacto en la calidad del vídeo recibido. Mediante la técnica de refresco Intra se evita la propagación temporal de errores por la pérdida de un paquete de un *frame* a los *frames* posteriores, aunque tiene el inconveniente de que el *bitrate* necesario se incrementa. Se utilizaron nueve modos de codificación de refresco Intra distintos, desde el modo AI hasta el modo LP, pasando por otros modos con una proporción distinta de *frames* de tipo I. Analizando las estadísticas de red recogidas durante la simulación, el tráfico de fondo afecta por igual al *Goodput* y Packet Delivery Ratio (PDR) de todos los modos de codificación, ordenándose las curvas según su mayor o menor *bitrate*, tal y como se esperaba. Sin embargo, para niveles altos de tráfico de fondo y para el caso de mayor *bitrate*, el modo AI, la red alcanza la saturación, perdiéndose gran cantidad de paquetes. Las estadísticas de *end-to-end delay* (E2E delay) y *jitter* muestran un comportamiento similar. Se puede observar que los modos de codificación con menor *bitrate* tienen un menor *delay* que los de mayor *bitrate* conforme se incrementa el tráfico de fondo. En el caso particular del modo AI se llega a tener un gran *delay*, ya que se alcanza el punto de saturación. Los valores altos del *jitter* se deben a la alternancia entre canal de control (CCH) y canal de servicio (SCH) que realiza la subcapa MAC del estándar IEEE Std. 802.11p-2010, puesto que todos los paquetes que se generan a nivel aplicación en los intervalos correspondientes a CCH deben esperar hasta el siguiente periodo de SCH. El uso de un mayor *bitrate* implica un mayor número de paquetes por cada *frame* codificado en el proceso de paquetización. Esto hace que el número de paquetes perdidos en valor absoluto sea mayor, aunque menor su proporción, es decir, menor Packet Lost Ratio (PLR), lo cual es coherente con la estadística PDR anterior. Del mismo modo, puesto que la pérdida de un único paquete impide la decodificación

del *frame* completo, la probabilidad de perder un *frame*, o Frame Lost Ratio (FLR), es mayor. Sin embargo, la pérdida de un mayor número de *frames* no significa una peor calidad del vídeo recibido, sino que depende mucho del modo de codificación. A la vista de los resultados, los modos más robustos al tráfico de fondo y que obtienen una mayor calidad (valor PSNR), son AI, IPIP, e I3P, es decir, aquellos con mayor refresco Intra, superando el valor umbral que se considera un nivel aceptable (28 dB) cuando existe un nivel de tráfico de fondo bajo, aunque no cuando éste es moderado o alto. El resto de modos no alcanzan dicho umbral en ningún caso. En concreto, los peores valores se consiguieron con los modos LP e IPx, puesto que si se pierde un paquete no puede decodificarse su correspondiente *frame* ni todos los *frames* que referencian a éste. En conclusión, el uso de refresco Intra es necesario para evitar la propagación de error temporal, obteniendo mejor calidad para todos los niveles de tráfico de fondo. Sin embargo, para el caso de presencia de tráfico de fondo moderado o alto no es suficiente, y se requieren otras técnicas complementarias.

En un segundo conjunto de experimentos se estudió otra técnica que podría ayudar a mejorar la robustez de la transmisión de vídeo. Se trata de la codificación basada en *tiles*, la cual permite realizar una partición de cada *frame* en bloques que son independientes desde el punto de vista de la codificación y la decodificación. De este modo, la pérdida de un único paquete no implica la pérdida de un *frame* completo, sino sólo el *tile* correspondiente, es decir, el *frame* puede ser parcialmente decodificado, obteniendo una mayor calidad final del vídeo que cuando no se usa la técnica de particionamiento en *tiles*, aunque el porcentaje de paquetes perdidos sea el mismo en ambos casos. Se utilizaron siete valores distintos: 1, 2, 4, 6, 8, 10 y 16 *tiles* por *frame*, correspondientes a los patrones uniformes de 1×1, 1×2, 2×2, 2×3, 2×4, 2×5, y 4×4, respectivamente. Estos patrones se utilizaron conjuntamente con el modo de codificación AI, que, a la vista de los resultados del experimento anterior, era el que obtenía la mejor calidad para todos los niveles de tráfico de fondo. El uso de *tiles* también introduce una sobrecarga en el *bitstream* resultante (hasta un 6.8 % en el peor de los casos para el modo AI), debido a ciertas cabeceras necesarias y una menor eficiencia en la codificación. Como consecuencia, los resultados muestran que la red se satura antes conforme se utiliza un mayor número de *tiles* por *frame*, para el mismo nivel de tráfico de fondo, por lo que tampoco conviene utilizar un número

elevado. Los gráficos de las estadísticas de red son similares a las del experimento anterior, pero incluyendo el hecho de un mayor *bitrate*. Por ejemplo, en los gráficos de Goodput y PDR, las curvas se ordenan de mayor a menor *bitrate* hasta que la red se satura, momento en que el orden se invierte debido a que afecta más a las secuencias de vídeo que requieren mayor *bitrate*. Lo mismo sucede en el caso del E2E Delay, en el que las curvas se ordenan también según el *bitrate*, aunque con valores muy aproximados. Sin embargo, las diferencias se amplían claramente entre los distintos casos conforme se aumenta el tráfico de fondo y la red llega a saturación. A la vista de los resultados, lo más relevante es que conforme se incrementa el número de *tiles* por *frame* se reduce el valor del Tile Lost Ratio (TLR) y se obtiene una mejor calidad del vídeo reconstruido, manteniéndose el valor de PSNR por encima del umbral aceptable hasta niveles de tráfico de fondo moderado. Con objeto de no incrementar innecesariamente el *bitrate* necesario, se ha estimado que un valor de 4 o 6 *tiles* por *frame* puede ser adecuado.

Las dos técnicas anteriores se analizaron también de forma combinada, esto es, se estudiaron los distintos modos de codificación con refresco Intra y los diferentes valores de fragmentación en *tiles* del experimento anterior, llegando a la conclusión de que el uso de *tiles* aumenta la calidad para cualquier modo de codificación utilizado. En concreto, el uso de 4 *tiles* por *frame* permitió a los modos AI e IPIP (los de mayor refresco Intra) alcanzar el umbral de calidad aceptable para niveles moderados de tráfico de fondo, aunque no para niveles altos. El resto de modos de codificación no alcanzaron dicho umbral en ningún caso. De todo ello se concluye que las dos técnicas analizadas se complementan de forma satisfactoria, aconsejando el uso de los modos de mayor refresco Intra, como AI, y el uso de 4 o 6 *tiles* por *frame*.

Con la intención de mejorar aun más la robustez del vídeo transmitido se hizo uso de calidad de servicio (QoS). En concreto, en los siguientes experimentos se utilizó la diferenciación de tráfico a nivel MAC que definió el estándar IEEE Std. 802.11p-2010, similar a la definida en el estándar IEEE Std. 802.11e-2005, aunque adaptando los parámetros de acceso al canal EDCA (Enhanced Distributed Channel Access) de las cuatro categorías de acceso (AC) a las particularidades de las redes vehiculares. De este modo se puede asignar distinta prioridad a los paquetes a transmitir, los

cuales se clasificarán en una de las cuatro colas de la subcapa MAC en función de su prioridad, que de menor a mayor son: AC_BK (Background), AC_BE (Best-Effort), AC_VI (Video), y AC_VO (Voice). Los estándares anteriores actualmente se han integrado en el IEEE Std 802.11-2016, requiriendo la habilitación del atributo OCBActivated (`dot11OCBActivated`) para permitir la transmisión de tramas sin requerir la asociación y autenticación previa con un Basic Service Set (BSS) o Punto de Acceso (AP), por lo que a este tipo de comunicación se le denomina comunicación Outside the Context of a BSS (OCB), y al estándar 802.11p también se le conoce como 802.11-OCB. De este modo es posible la comunicación directa entre vehículos (V2V) o entre vehículos y la infraestructura desplegada a lo largo de las carreteras (V2I).

En los experimentos realizados, se asignaron diferentes niveles de prioridad a los paquetes a transmitir en función del tipo de *frame* de vídeo al que pertenecían (I, P, B), variando la proporción de *frames* afectados (0, 25, 50, 75 y 100 %), asignando la misma prioridad a todos los paquetes que constituyen un mismo *frame*. Se seleccionaron 3 modos de codificación con distinta proporción de refresco Intra: AI, I7P, y LPI4, y la misma secuencia de número de *tiles* por *frame* y patrón de los experimentos anteriores (1, 2, 4, 6, 8, 10 y 16). Del mismo modo, se repitieron las simulaciones con distintos niveles de tráfico de fondo, a cuyos paquetes se les asignó la mínima prioridad (AC_BK). Debido a la gran cantidad de combinaciones de simulaciones a realizar, y puesto que se comprobó que los otros valores eran casos intermedios, en la práctica se puede considerar que se realizaron los siguientes tres experimentos únicamente (correspondientes a los casos 0 % y 100 %): (1) asignando la misma prioridad a todos los paquetes (AC_BK), ya sean de vídeo o de tráfico de fondo, (2) asignando una mayor prioridad (AC_VI) a todos los paquetes pertenecientes a los *frames* de tipo I únicamente, y (3) asignando una mayor prioridad (AC_VI) a todos los paquetes de vídeo, independientemente del tipo de *frame* al que pertenecen. El primer caso consiste en no realizar diferenciación de tráfico, es decir, no utilizar QoS, y sería equivalente al servicio Best-Effort que ofrece el estándar IEEE 802.11 por defecto. El motivo del segundo grupo de experimentos se basaba en el hecho de que hay *frames* más importantes que otros, es decir, que la pérdida de ciertos *frames* afecta más a la calidad de vídeo final que la pérdida de otros tipos de *frames*, debido

a las interdependencias existentes entre *frames* [20]. Por último, el tercer experimento es equivalente al mecanismo EDCA del estándar IEEE 802.11e. A continuación se resumen los resultados obtenidos con los tres experimentos mencionados.

Confirmando los resultados de trabajos anteriores, se observa que se requiere el uso de refresco Intra. En concreto, se obtienen los mejores resultados con el modo AI, el de mayor refresco Intra, seguido del modo LPI4. En cuanto al modo I7P, el de menor refresco Intra de los tres modos utilizados en este último trabajo, no alcanza el umbral mínimo considerado como aceptable (28 dB) cuando el nivel de tráfico de fondo es moderado o alto en ninguno de los tres experimentos, por lo que podría ser descartado. Como se comentó anteriormente, el uso de un mayor número de *tiles* por *frame* mejora la calidad percibida, ya que, la pérdida de un paquete de red ya no afecta a un *frame* completo sino sólo al *tile* al que pertenece. Sin embargo, puesto que el uso de *tiles* introduce cierta sobrecarga en el *bitstream*, con objeto de no saturar la red en exceso, parece conveniente un valor de 6 u 8 *tiles* por *frame* como mucho.

Cuando no se utiliza QoS (experimento 1), tanto el tráfico de vídeo como el tráfico de fondo sufren de igual manera al aumentar éste último, ya que ambos tienen la misma prioridad para acceder al medio. En el caso de asignar una mayor prioridad únicamente a los *frames* de tipo I (experimento 2), surge el inconveniente de que, como el flujo de vídeo tendrá *frames* con distintas prioridades, los paquetes se clasificarán al llegar a la capa MAC en colas diferentes según su prioridad. Esto hará que los paquetes se transmitan en orden distinto a su orden natural de reproducción, por lo que el receptor necesitará utilizar algún tipo de memoria intermedia (*buffer*) con objeto de almacenar los paquetes y ordenarlos antes de su decodificación y reproducción. Además, los paquetes de menor prioridad podrían sufrir retrasos considerables, haciendo que este método no sea factible en aplicaciones en tiempo real. En cuanto a la calidad final del vídeo recibido, aunque mejora respecto al experimento 1, se obtienen mejores resultados con el experimento 3, tal y como se comenta a continuación. Al asignar una mayor prioridad a todos los *frames* de vídeo (experimento 3, que para el caso del modo AI coincide con el experimento 2), las pérdidas en el tráfico de vídeo se reducen considerablemente, sin perjudicar demasiado al resto de tráfico de menor prioridad existente en la red (tráfico de fondo). Esto se debe a

que el tráfico de mayor prioridad obtiene un mejor aprovechamiento del canal por requerir un menor tiempo para el acceso al canal (CW), así como un menor tiempo entre el envío de tramas (AIFSN). Por último, se obtienen los mejores resultados con el experimento 3, seguido del experimento 2, y por último, el experimento 1 (sin QoS). Es decir, la asignación de mayor prioridad a todos los *frames* de vídeo (independientemente de su tipo) proporciona la mejor calidad final del vídeo reconstruido en las condiciones más adversas.

Capítulo 4

Conclusiones y trabajo futuro

4.1. Conclusiones

El conjunto de herramientas desarrolladas ha demostrado ser una contribución importante a la metodología utilizada actualmente para la evaluación de la calidad de vídeo percibida por el usuario final mediante la técnica de simulación. Por un lado, la herramienta *GatcomSUMO* permite la preparación de los archivos de simulación (red de carreteras, movilidad de los vehículos, posicionamiento de RSUs, etc.) en un tiempo mínimo, evitando los errores en tiempo de ejecución más habituales que ocurren durante la simulación, sin necesidad de conocer la sintaxis y extensa documentación de los simuladores OMNeT++ y SUMO. Por otro lado, el *framework* VDSF-VN es específico para la simulación de la transmisión de vídeo en redes vehiculares. Permite realizar el proceso completo, desde la codificación y generación de trazas de vídeo, siguiendo con la ejecución de las diferentes secuencias de simulación mediante los simuladores anteriores, y por último, la reconstrucción de las secuencias de vídeo recibidas y el análisis de la calidad percibida (PSNR). Todo ello se ejecuta desde un interface gráfico que lanza los procesos necesarios de forma concurrente, sacando partido al uso de multiproceso para reducir en lo posible la duración de todo el ciclo de trabajo. Desde el mismo interface gráfico pueden generarse y visualizarse los gráficos con los resultados de las estadísticas recogidas, en diversos formatos y resolución. Todas estas herramientas son multiplataforma (Windows, Linux, MacOS X), y están disponibles como código abierto.

El objetivo de los experimentos realizados es poder generar y transmitir secuencias de vídeo de forma más robusta, que mantengan un nivel de calidad perceptual aceptable en condiciones de tráfico de fondo moderado o alto en una red vehicular. En primer lugar, se comprobó que es necesario utilizar modos de codificación con refresco Intra, tales como All Intra (AI) o LPI4. Estos modos son más robustos en presencia de niveles de tráfico de fondo bajos, aunque no lo suficiente con mayor nivel de carga en la red. Posteriormente se combinaron los modos de codificación anteriores con el uso de particionado en *tiles* de los distintos *frames*. A pesar de incrementar el *bitrate* del *bitstream* resultante, la calidad del vídeo percibido mejora notablemente, hasta cierto punto; valores intermedios (hasta un máximo de 8 *tiles* por *frames*) consiguen un buen incremento de calidad sin saturar la red innecesariamente. Por último, se combinó todo lo anterior con el uso de calidad de servicio, asignando una mayor prioridad (AC_VI) a los paquetes correspondientes a los *frames* de vídeo. Primero se asignó mayor prioridad únicamente a los *frames* de tipo I, aunque se obtuvieron mejores resultados asignando esa prioridad a todos los *frames* de vídeo, independientemente de su tipo, evitándose además el problema de que los paquetes lleguen desordenados al receptor. El hecho de asignar mayor prioridad a algunos o todos los *frames* de vídeo no perjudica demasiado al resto de tráfico de menor prioridad existente en la red (tráfico de fondo), salvo en los casos de tráfico de fondo muy elevado, obteniéndose un mejor aprovechamiento del canal inalámbrico. A la vista de los resultados, mediante la combinación de las diversas técnicas empleadas se incrementa la robustez en la transmisión de flujos de vídeo en una red vehicular, mejorándose la calidad percibida por el usuario final.

4.2. Trabajo futuro

En las herramientas de *software* desarrolladas son varias las posibles ampliaciones que se podrían realizar. Por ejemplo, el entorno de trabajo VDSF-VN actualmente está preparado para el uso del codificador y decodificador HEVC del *software* de referencia HM. Podría ser interesante permitir el uso de otros codificadores HEVC de mayor rendimiento, tal como el codificador HEVC en tiempo real Kvazaar [31], x265 [19], o f265 [30], entre otros, así como otros decodificadores como el open-

HEVC [29]. De este modo, se facilitaría el uso del entorno por otros usuarios que utilicen distintos codificadores actuales (HEVC, VP10, ...), pudiendo adaptarse fácilmente a otros estándares pasados (H.264/AVC) o futuros. Por otro lado, actualmente el entorno permite el cálculo de dos métricas de la calidad de vídeo percibida, el MSE y el PSNR. Sin embargo, algunos trabajos indican que el PSNR, bajo ciertas circunstancias, no coincide exactamente con la opinión subjetiva de las personas, por lo que se han desarrollado diversas métricas que se correlacionan mejor con la calidad percibida, tal como el índice Structural SIMilarity (SSIM) [34], y otras que han surgido como mejora de la anterior como el Multi-Scale SSIM o MS-SSIM [35], o el Range SSIM (R-SSIM) [15]. Existen otras métricas que también funcionan bien pero son más complejas que las anteriores, tal como el índice Visual Information Fidelity (VIF) [27]. El entorno VDSF-VN podría ampliarse para calcular también alguna de estas nuevas métricas de la calidad percibida.

A la vista de los resultados de los experimentos realizados, a pesar de las mejoras obtenidas podrían analizarse otras técnicas adicionales para combinarlas con las técnicas utilizadas con objeto de hacer más robusta la transmisión de vídeo. Por ejemplo, podrían utilizarse técnicas de corrección de errores como Forward Error-Correction (FEC). También podría diseñarse algún esquema adaptativo que tenga en cuenta el nivel de saturación de la red, o el tamaño de las distintas colas del MAC. En los experimentos realizados, la transmisión de flujos de vídeo se ha realizado en modo *broadcast*, y podría ser interesante la comparación con la transmisión en modo *unicast*, así como explorar la interacción entre la capa aplicación y la capa MAC del IEEE Std. 802.11p-2010 del modelo de referencia WAVE (Wireless Access in Vehicular Environments).

Bibliografía

[1] Advanced Video Coding (AVC) for generic audiovisual services. ITU-T Recommendation H.264, 2003.

[2] Studio encoding parameters of digital television for standard 4:3 and widescreen 16:9 aspect ratios. ITU-R Recommendation BT.601-7, March 2011.

[3] High Efficiency Video Coding (HEVC). ITU-T Recommendation H.265, 2013.

[4] Arizona State University (ASU). Video Trace Library. Available online: `http://trace.eas.asu.edu/index.html` (accessed on August 20, 2018).

[5] F. Bossen. Common Test Conditions and Software Reference Configurations. Technical Report JCTVC-L1100, Joint Collaborative Team on Video Coding, Geneva, January 2013.

[6] P.-P. Garrido Abenza. rplotengine: R as a Plotting Engine. Available online: `https://cran.r-project.org/package=rplotengine` (accessed on September 28, 2019).

[7] M. M. Haklay and P. Weber. OpenStreetMap: User-Generated Street Maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

[8] International Telecommunication Union. ITU-T Recommendation P.800.1: Mean Opinion Score (MOS) terminology. Technical report, July 2006.

[9] ITU-T Recommendation P.910. Subjective video quality assessment methods for multimedia applications, Apr. 2008.

[10] Joint Collaborative Team on Video Coding (JCT-VC). HEVC reference software HM (HEVC Test Model) and Common Test Conditions. Available online: `https://hevc.hhi.fraunhofer.de` (accessed on August 20, 2018).

[11] Juan Casal. HomerHEVC. Available online: `http://homerhevc.com` (accessed on April 10, 2019).

[12] J. Klaue, B. Rathke, and A. Wolisz. *EvalVid – A Framework for Video Transmission and Quality Evaluation*, pages 255–272. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[13] J. Lessmann, P. Janacik, L. Lachev, and D. Orfanus. Comparative Study of Wireless Network Simulators. In *Seventh International Conference on Networking (ICN 2008)*, pages 517–523, April 2008.

[14] H. Lutnik. JGPSTrackEdit. Available online: `https://sourceforge.net/projects/jgpstrackedit/` (accessed on August 20, 2018).

[15] W. S. Malpica and A. C. Bovik. Range image quality assessment by structural similarity. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1149–1152, April 2009.

[16] S. Mehta, N. Ullah, M. H. Kabir, M. N. Sultana, and K. S. Kwak. A Case Study of Networks Simulation Tools for Wireless Networks. In *2009 Third Asia International Conference on Modelling Simulation*, pages 661–666, May 2009.

[17] I. Minakov, R. Passerone, A. Rizzardi, and S. Sicari. A Comparative Study of Recent Wireless Sensor Network Simulators. *ACM Trans. Sen. Netw.*, 12(3):20:1–20:39, July 2016.

[18] Muhammad Usman Karim Khan and Muhammad Shafique and Jörg Henkel. ces265. Available online: `http://sourceforge.net/projects/ces265/` (accessed on April 10, 2019).

[19] MulticoreWare. HEVC x265. Available online: `http://x265.org/` (accessed on April 10, 2019).

[20] P. Orosz, T. Skopkó, and P. Varga. Towards estimating video QoE based on frame loss statistics of the video streams. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 1282–1285, May 2015.

[21] L. G. Papaleondiou and M. D. Dikaiakos. TrafficModeler: A Graphical Tool for Programming Microscopic Traffic Simulators through High-Level Abstractions. In *VTC Spring 2009 - IEEE 69th Vehicular Technology Conference*, pages 1–5, April 2009.

[22] P. J. P. Peral. *Robust Video Streaming over Vehicular Networks*. PhD thesis, Departamento de Ingeniería de Sistemas y Automática, Universidad Miguel Hernández de Elche, Sept. 2015.

[23] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2016.

[24] T. Schierl, A. Eleftheriadis, S. Wenger, and Y.-K. Wang. RTP Payload Format for Scalable Video Coding. RFC 6190, May 2011.

[25] I. Scholz, D. Stöcker, et al. JOSM. Available online: `https://josm.openstreetmap.de/` (accessed on August 20, 2018).

[26] J. Schweizer. *SUMOPy: An Advanced Simulation Suite for SUMO*. Lecture Notes in Computer Science (LNCS). Springer-Verlag Berlin Heidelberg, November 2014.

[27] H. R. Sheikh and A. C. Bovik. Image information and visual quality. *IEEE Transactions on Image Processing*, 15(2):430–444, Feb 2006.

[28] A. M. Uhrmacher, S. Brailsford, J. Liu, M. Rabe, and A. Tolk. Reproducible Research in Discrete Event Simulation: A Must or Rather a Maybe? In *Proceedings of the 2016 Winter Simulation Conference*, WSC '16, pages 1301–1315, Piscataway, NJ, USA, 2016. IEEE Press.

[29] VAADER Team (IETR/INSA Rennes) et al. openHEVC. Available online: `http://openhevc.insa-rennes.fr` (accessed on April 10, 2019).

[30] Vantrix. HEVC f265. Available online: `https://vantrix.com/f-265/` (accessed on April 10, 2019).

[31] M. Viitanen, A. Koivula, A. Lemmetti, J. Vanne, and T. D. Hämäläinen. Kvazaar HEVC encoder for efficient intra coding. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1662–1665, May 2015.

[32] M. Viitanen, A. Koivula, A. Lemmetti, A. Ylä-Outinen, J. Vanne, and T. D. Hämäläinen. Kvazaar: Open-Source HEVC/H.265 Encoder. In *Proceedings of the 24th ACM International Conference on Multimedia*, MM '16, pages 1179–1182, New York, NY, USA, 2016. ACM.

[33] Y. Wang, Y. Sanchez, T. Schierl, S. Wenger, and M. Hannuksela. RTP Payload Format for High Efficiency Video Coding. RFC 7798, 2016.

[34] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[35] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thirty-Seventh Asilomar Conference on Signals, Systems Computers*, volume 2, pages 1398–1402 Vol.2, Nov 2003.

[36] T. Williams and C. Kelley. Gnuplot. Available online: `http://www.gnuplot.info` (accessed on September 12, 2019).

[37] Xiph.org. Video Test Media. Available online: `https://media.xiph.org/video/derf/` (accessed on August 20, 2018).

[38] E. Yücesan and S. H. Jacobson. Building Correct Simulation Models is Difficult. In *Proceedings of the 24th Conference on Winter Simulation*, WSC '92, pages 783–790, New York, NY, USA, 1992. ACM.

Anexos

Anexo A

Acrónimos

AC	Access Category
AI	All Intra
AIFSN	Arbitration Inter-Frame Space Number
AVC	Advanced Video Coding
BSS	Basic Service Set
CBR	Channel Busy Ratio
CCH	Control Channel
CTC	Common Test Conditions
CW	Contention Window
DPI	Dots Per Inch
EDCA	Enhanced Distributed Channel Access
FEC	Forward Error Correction
FSPL	Free-Space Path Loss
GLONASS	Global Navigation Satellite System
GNSS	Global Navigation Satellite System
GOP	Group of Pictures
GPS	Global Positioning System
GUI	Graphical User Interface
HEVC	High Efficiency Video Coding
HM	HEVC Test Model
IEEE	Institute of Electrical and Electronic Engineer

IPTV	Internet Protocol Television
ITS	Intelligent Transportation Systems
ITU	International Telecommunication Union
JCT-VC	Joint Collaborative Team on Video Coding
JVM	Java Virtual Machine
LAN	Local Area Network
LP	Low-delay P
MAC	Medium Access Control
MOS	Mean Opinion Score
MPEG	Moving Picture Experts Group
MSE	Mean Squared Error
MTU	Maximum Transmission Unit
OCB	Outside the Context of a BSS
OSM	OpenStreetMap
OTT	Over-The-Top
PDR	Packet Delivery Ratio
PSNR	Peak Signal-to-Noise Ratio
QoE	Quality-of-Experience
QoS	Quality-of-Service
QP	Quantization Parameter
RSU	Road-Side Unit
RTP	Real-time Transmission Protocol
SCH	Service Channel
SSIM	Structural SIMilarity
SUMO	Simulation of Urban MObility
TLR	Tile Lost Ratio
UTM	Universal Transverse Mercator
V2V	Vehicle-to-Vehicle
V2I	Vehicle-to-Infrastructure
VANET	Vehicular Ad-hoc NETwork
VDSF-VN	Video Delivery Simulation Framework over Vehicular Networks
Veins	VEhicles In Network Simulation

VIF	Visual Information Fidelity
WAVE	Wireless Access in Vehicular Environments
WGS	World Geodetic System
WLAN	Wireless Local Area Network
XML	Extensible Markup Language

Anexo B

Publicaciones

En este anexo se muestra una copia completa del trabajo principal que da soporte a esta Tesis Doctoral, cuyos datos bibliográficos completos son los siguientes:

- **Garrido Abenza, P.P.**, Malumbres, M.P., Piñol, P., López-Granado, Otoniel, *Source Coding Options to Improve HEVC Video Streaming in Vehicular Networks*, Sensors Journal (ISSN: 1424-8220) - Special Issue "Advances on Vehicular Networks: From Sensing to Autonomous Driving", Vol. 18, Issue 9, 2018. DOI: https://doi.org/10.3390/s18093107
  Factor de impacto: 3.031, JCR: 15/61 (Q1) en la categoría "Instruments & Instrumentation"

sensors

MDPI

Article

Source Coding Options to Improve HEVC Video Streaming in Vehicular Networks

Pedro Pablo Garrido Abenza *, Manuel P. Malumbres, Pablo Piñol and Otoniel López-Granado

Department of Physics and Computer Architecture, Miguel Hernández University, 03202 Elche, Spain; mels@umh.es (M.P.M.); pablop@umh.es (P.P.); otoniel@umh.es (O.L.G.)

* Correspondence: pgarrido@umh.es; Tel.: +34-96-665-8387

Received: 21 August 2018; Accepted: 12 September 2018; Published: 14 September 2018

Abstract: Video delivery in Vehicular Ad-hoc NETworks has a great number of applications. However, multimedia streaming over this kind of networks is a very challenging issue because (a) it is one of the most resource-demanding applications; (b) it requires high bandwidth communication channels; (c) it shows moderate to high node mobility patterns and (d) it is common to find high communication interference levels that derive in moderate to high loss rates. In this work, we present a simulation framework based on OMNeT++ network simulator, Veins framework, and the SUMO mobility traffic simulator that aims to study, evaluate, and also design new techniques to improve video delivery over Vehicular Ad-hoc NETworks. Using the proposed simulation framework we will study different coding options, available at the HEVC video encoder, that will help to improve the perceived video quality in this kind of networks. The experimental results show that packet losses significantly reduce video quality when low interference levels are found in an urban scenario. By using different INTRA refresh options combined with appropriate tile coding, we will improve the resilience of HEVC video delivery services in VANET urban scenarios.

Keywords: vehicular networks; video delivery; QoS; QoE; HEVC; OMNeT++; Veins; SUMO

1. Introduction

Among the potential applications that may be supported by Vehicular Ad-hoc NETworks (VANETs), video delivery is one of the most resource-demanding. Several application scenarios may require video delivery services in either on-demand or real-time live video streaming, using unicast, multicast or broadcast communications. We may find scenarios where video delivery is required, like the ones related to accidents/rescue assistance, V2X real-time video, context-aware video broadcasts, security surveillance services, driving assistance, etc. However, multimedia streaming over VANETs is a very challenging issue mainly due to the high mobility of vehicles, bandwidth limitations, and the high loss rates typically found in wireless communications. In addition, video transmission requires a high bandwidth with a bounded packet delay, especially when real-time restrictions are mandatory. So, when video suffers from packet losses and/or highly variable packet delays, the user perceived video quality is seriously reduced.

In this work, we analyze the performance of video delivery in a typical VANET urban scenario by means of a simulation framework named Video Delivery Simulation Framework over Vehicular Networks (VDSF-VN) [1], which allows us to model in detail the different actors involved in a video streaming session. VDSF-VN works with the OMNeT++ simulator [2], together with the Veins (VEhicles In Network Simulation) framework [3] to conduct the network simulations, and with SUMO (Simulation of Urban MObility) [4], as the vehicular traffic simulator.

Inside VDSF-VN, the selected source video sequence is encoded with the High Efficiency Video Coding (HEVC) standard [5]. In particular, we have developed a tuned version of the HEVC reference software HM (HEVC Test Model) [6] that it is able to seamlessly work with OMNeT++/Veins/SUMO simulation tools. Also, it has been necessary to develop several software modules, such as a packetizer/depacketizer tool, an RTP packet trace module, etc. All of these software modules are governed from a graphical application, named GatcomVideo, that will significantly improve the usability and automation of the proposed VDSF-VN framework.

Although the simulation framework provides a lot of useful performance metrics (end-to-end delay, goodput, packet delivery ratio, etc.), our main performance metric is the video quality delivered to the user, since it will be used to determine the minimum quality levels that a video delivery application should provide. After analyzing several simulation runs we notice that the received video quality becomes unacceptable when the first interferences appear in the network. So, we proceed to study some coding options implemented in the HEVC video encoder, in order to improve the final video quality delivered to the user.

These options are based on INTRA refresh and tile-based video coding techniques. On the one hand, INTRA refresh technique will help to stop the error propagation produced by a single packet loss in the prediction process of future (next) video frames. So, we have analyzed the performance of several INTRA refresh approaches in order to determine the ones which deliver a higher video quality to the user with an acceptable bit rate overhead. On the other hand, TILE-based coding allows partitioning each video frame in blocks (Tiles), which are completely independent regarding coding/decoding. This may be useful to mitigate the impact of packet losses when compared with no-tiling approaches. When no-tiling approaches (the default coding option) are used, an encoded frame is typically fragmented into several network packets. To decode the frame, all those packets need to be received. If just one packet is lost, the frame is undecodable although the rest of packets had been correctly received. If tiling is used, a frame can be partially decoded (not completely lost) improving the final video quality at the same packet error rate.

A thorough experimental process was developed in order to identify the above mentioned HEVC coding options which offer the best video quality to the user. The simulation results obtained from the experimental tests will show the behavior of the HEVC coding options under different network impairment degrees. This knowledge will provide us with the ability to decide which level of INTRA refreshing is required and how many tiles per frame should be used to guarantee the minimum video quality required by the video delivery application.

The rest of the paper is structured as follows. Section 2 shows some related works, specifically some techniques and simulation tools used in the literature for improving the video quality perceived by the final user in wireless networks. In Section 3, we briefly describe the video delivery simulation framework we have used in this work. Section 4 describes the simulation methodology, the VANET scenario and the video sequence used. The simulation results of the experimental tests are presented and discussed in Section 5. Finally, conclusions are drawn in Section 6, along with references to future work.

2. Related Works

Many different techniques have been proposed in the literature to improve video streaming over not reliable medium like the wireless channel, in general, and VANETs, in particular [7].

Error Concealment (EC) methods try to minimize the impact of packets loss in the perceived quality of the received video sequence [8–10]. If any part of an encoded frame is missing, then the entire frame cannot be decoded. These EC methods try to predict the missing parts and correct them by exploiting spatial similarities of the nearby areas, or temporal redundancies in the video sequence [11]. Spatial prediction can be only used for small areas [12], whereas the temporal prediction can replace a whole missing frame, replacing it with a previously received frame. This is known as 'Frame Copy

Concealment' [13]. To make an estimation of the contents of a missing area, different techniques based on Motion Vectors (MV) information can also be used.

Error Resilience (ER) is a group of techniques which aim to provide robustness to video sequences, which are applied at the sender side. One of these is the Feedback Channel (FC), in which the receiver can request the retransmission of some missing part [14]. Other methods make use of substreams, such as Multiple Description Coding (MDC) [15] or Layered Coding (LC).There has been a lot of research done to protect video sequences with Forward Error Correction (FEC), also known as Channel Coding [16]. This technique inserts redundant data, called Error Correction Codes (ECCs), into the encoded video. These ECCs can be used for detecting missing data on the receiver side, and even, to recover them without retransmission, which is not feasible in some scenarios like real-time applications (e.g., live video streaming) [17]. RaptorQ codes is an example of this kind of mechanism [18]. FEC can be used in all kind of networks, including VANETs [19].

Another group of ER techniques are the so-called 'intra refresh' methods. They encode the bit stream in such a way that it prevents fast quality degradation in the presence of data losses. An inter-coded frame, a frame which it is predicted using other frames as references, although correctly received, can be affected by a reference frame with errors, and it can propagate the error to other inter-frames which use it as reference. A proper insertion of intra-coded frames together with an adequate selection of reference frames may increase the error resilience of an encoded bit stream. So, defining appropriate INTRA refresh encoding modes will make bit streams more robust. There are several works in the literature that exploit these techniques to improve video robustness like the ones found in [20–22].

To evaluate the performance of source coding techniques we need a simulation framework to analyze their effects in the final video quality experienced by a VANET user. To measure video quality we will use the Peak Signal-to-Noise Ratio (PSNR) metric. So, the error protection of our encoder will be based on those encoding options which provide a better received video quality, taking into account the bit rate overhead which they typically provide.

There are several simulation frameworks proposed in the literature that follow our requirements, like EvalVid [23], which allows the video quality evaluation of MPEG4 video transmitted over a real or simulated communication network. Besides measuring several metrics of the underlying network, like loss rate, delay, and jitter, they also support a video quality evaluation of the received video based on the frame-by-frame PSNR calculation. Although no specific network simulator is proposed, authors argue that theoretically any simulator could be used. In this sense, several works have extended EvalVid to include a network simulator. For example, in [24] EvalVid is integrated with ns-2. In [25] authors developed a tool named QoE Monitor, which is based on the EvalVid architecture and consists of a new module for the ns-3 simulator. Beside this, this tool can be extended with current video coding standards. In [26] a simulation framework named Mobile Multi-Media Wireless Sensor Networks (M3WSN) is presented, which is based on EvalVid too. This framework uses the OMNeT++ simulator and Castalia framework [27].

Another example is HEVStream [28], which is a realistic testbed environment that aims to evaluate HEVC video streaming under different conditions. However, it is hardware-based and we need to use the simulation technique because our target is to evaluate video streaming in vehicular networks.

Despite of the different existing video frameworks (most of them based on the initial EvalVid approach), none of them allows the transmission and quality evaluation of video sequences with the combination of the OMNeT++ simulator, the Veins framework and the SUMO traffic mobility model for vehicular networks. Some analyzed frameworks lack of an updated video codec module, being not trivial to change it with another one, because the packetizer needs to be properly adapted to the intrinsic features of the target bit stream syntax. Also, node mobility models are too simple in most approaches to fit with the particularities of vehicular networks (roads, streets, lanes, stops, traffic lights, etc.). In addition, finally, the different modules of the framework need to be completely integrated in order to launch global simulations specifying the detailed configuration of every module,

Sensors **2018**, *18*, 3107 4 of 18

and performing, on the fly, all the processes, from the video encoding at source node to the decoding process at the receiver end, passing through the network simulation of the video delivery in a realistic vehicular network scenario. These aspects have motivated us to develop the VDSF-VN [1] simulation framework which it is especially suited for vehicular networks. VDSF-VN is the simulation framework that we use in this work, so we provide a brief introduction of it in the following section.

3. Simulation Framework: VDSF-VN

In Figure 1, we show the different elements that form the VDSF-VN simulation framework architecture. Some of those elements have to do with the video coding and decoding processes (encoder, packetizer, etc.), whereas some others are related to the simulation process (OMNeT++, Veins and SUMO). Apart from these, a graphical application named "GatcomVideo" was developed which acts as a front-end that manages the whole process. Finally, in order to build the simulation network and configuration files, the "GatcomSUMO" [29] application was also developed to ease all the configuration duties by means of a Graphical User Interface (GUI).

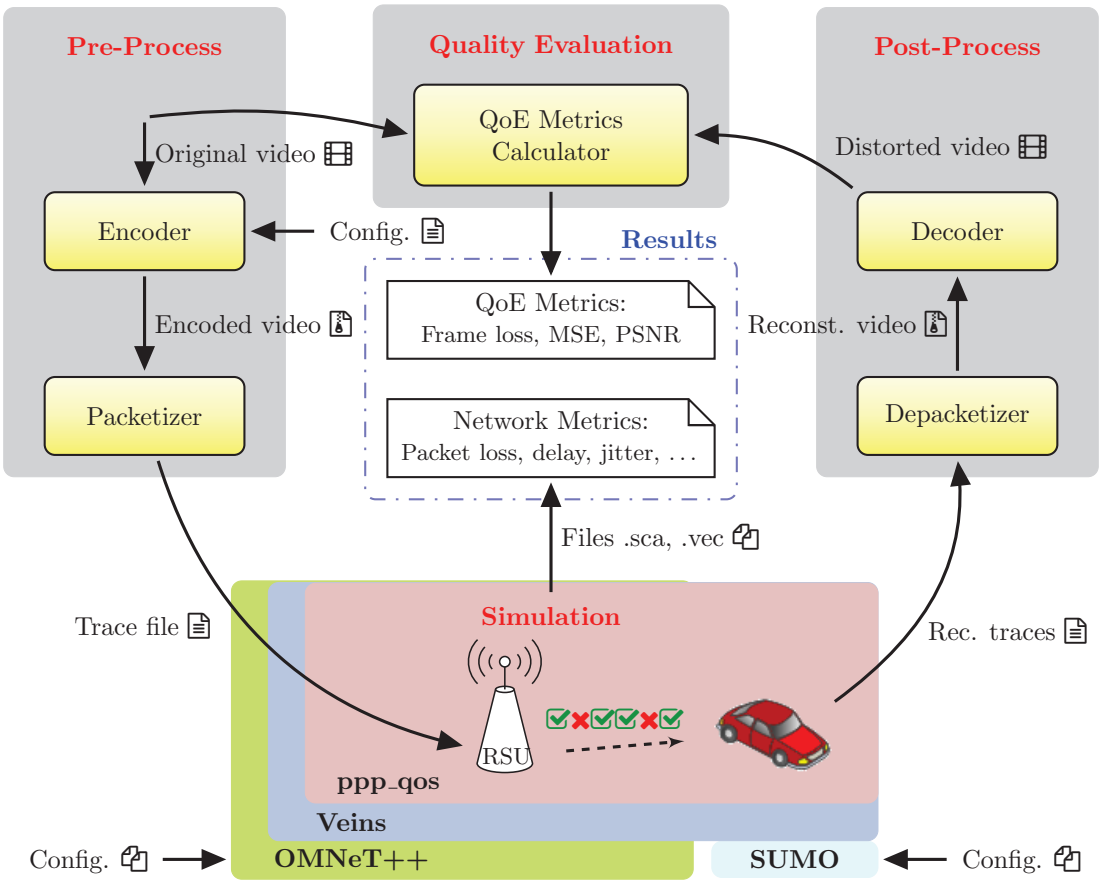

Figure 1. Workflow of VDSF-VN.

The VDSF-VN framework coordinates and manages all the necessary tasks involved with this kind of experiments, such as, source video encoding, trace files generation, video decoding tasks, running the simulation sequences and obtaining the desired plots from the simulation statistic results. VDSF-VN was also designed to be multithread-aware, so several tasks or simulation runs may be launched in parallel depending on the available hardware resources.

As depicted in Figure 1, all of these tasks can be structured in three main steps: (1) pre-process, (2) simulation, and (3) post-process.

1. Pre-process: This step is in charge of encoding the desired video sequences and then perform the Real-time Transmission Protocol (RTP) packetization of the encoded bit stream and generate the corresponding video trace file. This step is a time-consuming task, because the video encoding requires a great amount of computation and the video sequence may be encoded with different configuration parameter sets.
2. Simulation: This step includes the actions to prepare the network scenario and run the simulations with OMNeT++, Veins and SUMO. The trace file from the pre-process step is loaded by the video servers to simulate the video broadcasting. During the simulation, the client nodes write the correctly received video packets into an output trace file, which will be used in the post-process step. This is done in an OMNeT++ project named "ppp_qos", which references the original Veins project, but it is implemented in a separate project for convenience. This project includes the "TraCIDemo11p" application module provided in Veins, which has been extended with: (a) the support of HEVC video trace files, (b) statistics at application level (load, goodput, end-to-end delay), and (c) statistics related to the mobility of the vehicles (the distance between selected pairs of nodes, the number of neighbors during the simulation, etc.). The "ppp_qos" project also extends the MAC layer with many statistics like the Channel Busy Ratio (CBR), that may be local statistics (i.e., per node), global network statistics or statistics grouped by Access Category (AC).
3. Post-process: This step is taken to get all the statistics defined at different network levels: Physical, MAC, and Application, with the corresponding plots. To compute the application performance metrics like Frame Loss Rate (FLR) and PSNR, the bit stream needs to be conformed with the received packets and then decoded to obtain the reconstructed video which will be rendered to the user. Many works show that PSNR, under some circumstances, does not properly assess the quality perceived by users. In the literature, we can find quality assessment metrics that better correlates with human perceived quality like MS-SSIM [30]. These video quality metrics will be incorporated in future versions of VDSF-VN as an optional feature. However, we will use PSNR metric in this work, as most simulation frameworks also use it.

4. Performance Evaluation

This section describes the methodology used in the simulation and the experiments conducted with the proposed VDSF-VN framework for evaluating the video streaming in an urban VANET scenario. Now, we will describe the simulation setup which includes the network scenario, the network simulation parameters, the video source, and the video encoding options.

4.1. Scenario Description

The VDSF-VN framework includes a tool, GatcomSUMO [29], that allows defining and configuring the network scenario that will be used in the simulation. Different vehicular network scenarios may be designed: regular (mesh, spider, etc.) or irregular (random). It is also possible to import real urban vehicular scenarios from OpenStreetMap [31], so the desired area of a city may be used as our simulation scenario. In this work, we have used a portion of the city of Kiev (Ukraine), which consists of a square area sized 2000×2000 m^2 (Figure 2a), downloaded from OpenStreetMap and imported into SUMO. The main simulation parameters are summarized in Table 1.

Three fixed Road Side Units (RSUs) are placed along a central avenue (`rsu[0..2]`), which act as video servers delivering the same video sequence in a synchronized and cyclic way. The parameters of the network cards are set with the values shown in Table 2. The radio transmission range, which is depicted with a blue circle in Figure 2a, is the default value used in Veins. To get more accurate results, the simulation takes into account the obstacles such as buildings. Specifically, the radio propagation model used is `SimpleObstacleShadowing` instead of the `SimplePathLoss` model.

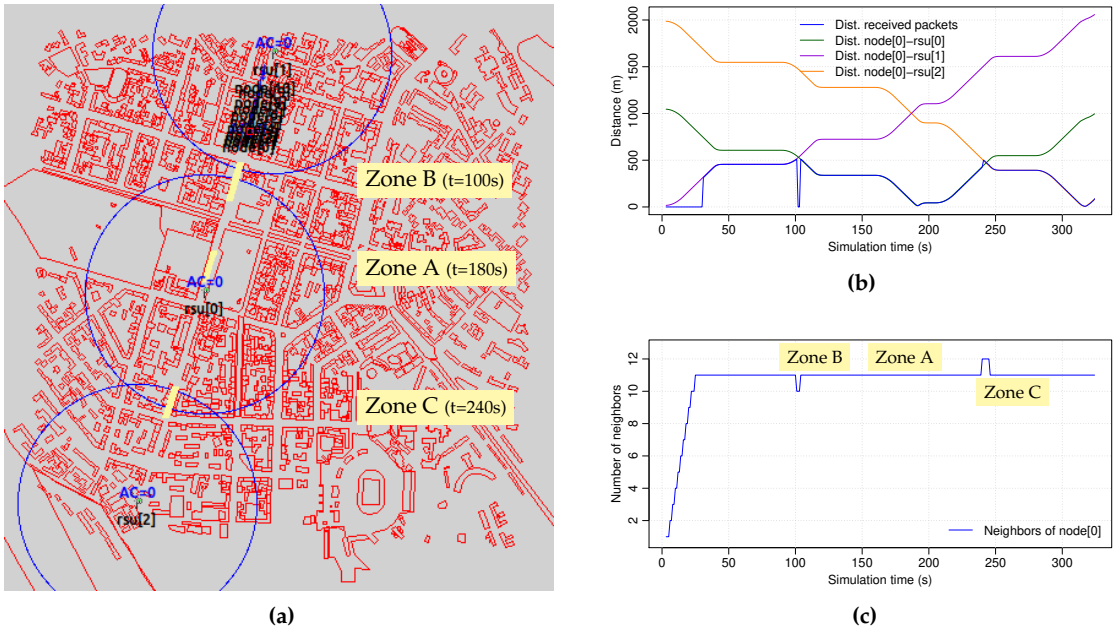

Figure 2. (**a**) Scenario. (**b**) Distance from client to RSUs. (**c**) Number of client neighbors.

Table 1. Simulation parameters.

Parameter	Value
Simulation area	2000×2000 m^2
Simulation time	340 s
Number of RSUs	3
Number of client vehicles	1
Number of background vehicles	10
Background traffic load	{0,12,25,50,75} pps
Max. speed of vehicles	14 m/s (50 km/h)

Table 2. PHY/MAC parameters.

Parameter	Value
Carrier frequency	5.890 GHz
Propagation model	SimpleObstacleShadowing
Bit rate	18 Mbps
Transmit power	20 mW
RX Sensitivity	-89 dBm
Communication range	510.87 m
MAC queues size	0 (infinite)

On the other hand, several mobile vehicles are defined, which travel together along the simulation time. The first one is the video client (`node[0]`), which receives the video sequence sent by the video servers. Then, ten vehicles acting as background traffic nodes follow the first one (`node[1..10]`), and send packets at different bit rates as background traffic load. To represent different levels of network loads, each background traffic node injects packets with a size of 512 bytes in a sequence of the following rates: {0,12,25,50,75} packets per second (pps). Therefore, on average, the total aggregate background traffic is {0,120,250,500,750} pps, respectively.

All the vehicles follow the same fixed route along the avenue, which is defined as a list of edges. This can be done manually, with the help of some utility provided by SUMO like `duarouter`, or in a graphical way with the GatcomSUMO application. The maximum speed of the vehicles was

fixed to 14 m/s (50 km/h), although their velocity is not constant, as the traffic simulator takes into account intersections, traffic-lights, the presence of other vehicles, etc. SUMO is a microscopic vehicular simulator that slows down or accelerates each vehicle according to the traffic conditions. The mobility model used is the default one in SUMO, that is, a modified version of the Krauss model [32], which is a car following model. Apart from the mobility model (`carFollowModel`) and the maximum speed (`maxSpeed`), additional parameters like acceleration (`accel`), deceleration (`deccel`), or minimum distance to the vehicle ahead (`minGap`) are also used.

The simulation time is 340 s, which is time enough for all the vehicles to travel from the beginning to the end of the avenue, receiving the video sequences sent by the three RSUs. Figure 2b shows the distance between the client and the RSUs. Figure 2c shows the number of neighbors of the client node along the entire simulation, showing that all of the background vehicles are within the communication range of the client car all the time. The plot also shows the instant when the node passes through the three selected zones which represent different network situations: an area where the client node has full coverage of one of the RSUs (Zone A), the shadow area between `rsu[1]` and `rsu[0]` (Zone B), as well as the overlapping area between `rsu[0]` and `rsu[2]` (Zone C).

4.2. Video Sequence

The selected video sequence is 'BasketballDrill', which belongs to the HEVC Common Test Conditions set [6]. As shown in Table 3, it has a resolution of 832×480 pixels and a duration of 10 s. The original sequence is 500 frames long at a rate of 50 frames per second (fps), but it was sub-sampled at 25 fps with a length of 250 frames in order to reduce the required network bandwidth.

Table 3. Video sequence.

Parameter	Value
Name	BasketballDrill
Resolution	832×480 pixels
Duration	10 s
Length	250 frames
Rate	25 frames per second

This video sequence was encoded with the HEVC reference software HM (HEVC Test Model) [6]. The resulting bit stream consists of a sequence of frames, which can be of three types: I frames (Intra coded), P frames (Predictive coded) and B frames (Bi-directionally predictive coded). A great number of possible encoded bit streams can be generated from a specific raw video sequence depending on the desired HEVC configuration parameters (encoding mode, quantization level, INTRA refreshing, etc.). For example, we can select the main encoding modes provided by the HEVC reference software, namely, All Intra (AI), Low-delay P (LP), Low-delay B (LB), and Random Access (RA), or we can configure other specific encoding modes like the ones that we are going to analyze in this work, shown at Table 4. AI mode is robust to packet loss, as all the frames of the video sequence are encoded as I frames, that is, without using any other frame as reference. On the contrary, LP mode is less robust than AI because it is sensible to packet losses due to the dependencies between frames. In LP mode, the first frame is encoded as an I frame, and the rest of the frames are encoded as P frames, which use previously encoded frames as reference. So, if losses occurs in previous reference frames we cannot correctly predict the content of the current one since we have no information from the past frames. The error produced in the past affects to the encoding performance of future frames (error propagation). However, LP mode is very efficient regarding compression performance because of the use of motion estimation and compensation. This is the reason we propose a set of encoding modes with different levels of periodic updates (INTRA refresh) which is able to efficiently mitigate the error propagation effects without significant performance degradation.

Another coding parameter to define is the Quantization Parameter (QP), which is used to adjust the compression level. A low QP value implies a soft quantization that will result in larger bit streams with higher video quality. In our experiments, the video quality, measured in terms of PSNR, for all the generated bit streams was fixed to the same value by adjusting the compression level (i.e., the QP value) in each case, targeting to a desired PSNR value of approximately 36 dB.

Apart from the compression mode and QP value, the video encoder accepts many other parameters. For example, HEVC allows splitting a frame into several independent regions (tiles or slices). This will be useful for our purposes since it will provide an extra robustness to video delivery when packet losses start to damage the video packet streaming. However, these partitioning modes lower the encoding performance since spatial correlation is limited to the slice/tile domain, and their use requires extra bit stream signaling information [33].

As a result, many combinations of parameters need to be considered. In this work, the selected encoding modes, QP values and number of tiles per frame are the following:

- Encoding modes (×9): {AI, IPIP, I3P, I7P, I15P, IPx, IPx25pctCTU, LPI4, LP} (see Table 4).
- QP values (×1): the QP value used for each encoding mode was fixed to achieve the desired video quality (PSNR ≈ 36 dB) when using 1 tile per frame (see Table 5).
- Number of tiles per frame (×7): seven values were considered to encode each bit stream: {1, 2, 4, 6, 8, 10, 16}. The size of these partitions can be specified with the number of rows and columns (uniform), or with the number of Coding Tree Units (CTUs) per each row and column (non-uniform). In this work, the following uniform tile patterns were considered: {1×1, 1×2, 2×2, 2×3, 2×4, 2×5, 4×4}, respectively.

Table 4. Encoding modes.

Mode	Frame layout	Description
AI	IIIIIIII...	Every frame is an I frame (All intra)
IPIP	IP IP IP IP I...	Alternating I and P frames
I3P	IPPP IPPP I...	An I frame followed by three P frames
I7P	IPPPPPPP IPPPPPPP I...	An I frame followed by seven P frames
I15P	IPPPPPPPPPPPPPPP I...	An I frame followed by fifteen P frames
IPx	IPPPPPPPPPPPPPPP...	Similar to LP but each P frame reference the previous frame only
IPx25pctCTU	IPPPPPPPPPPPPPPP...	Similar to IPx but 25% of CTUs are forced to be Intra refreshed
LPI4	IPPP IPPP IPPP...	Similar to LP but with an I frame every four frames
LP	IPPPPPPPPPPPPPPP...	An I frame followed by only P frames (Low-delay P)

Table 5. QP values for achieving PSNR ≈ 36 dB (1 tiles per frame).

Mode	QP	Tiles	Bit rate (Mbps)	PSNR (dB)
AI	31	1	3.417	35.863
IPIP	30	1	2.378	36.042
I3P	30	1	1.647	35.761
I7P	29	1	1.457	36.071
I15P	29	1	1.257	35.830
IPx	28	1	1.239	35.782
IPx25pctCTU	28	1	1.797	35.812
LPI4	29	1	1.620	36.045
LP	28	1	0.959	36.160

So, all of these combinations make a total of 63 different bit streams (9×1×7). As it can be appreciated in Table 5, the PSNR value of the original encoded video is approximately the same for all of them, whereas each case has a different bit rate value. As an example, the largest bit stream is achieved with AI mode (3.417 Mbps), whereas the smallest one corresponds to LP (0.959 Mbps); the other combinations are intermediate cases.

Once video encoding is done, we proceed to build a trace file from the encoded bit stream [34]. A trace file is a text file including an ordered list of packets to be transmitted, and is loaded by the video servers before sending it to the clients in the simulation step. Since an encoded frame may be larger than the network Maximum Transmission Unit (MTU), it is necessary to encapsulate it into several packets (packetization). In the trace file, each packet is defined with the following fields: a correlative packet number, the frame type it belongs to (I, P, or B), the playback time (ms), the packet size (bytes), the frame offset of the packet payload, and the total number of fragments of the current frame.

We have modified the HM software to include RTP bit stream packetization [35]. In addition, it was mandatory to modify the HM decoder in order to get the reconstructed video sequence because the original HM decoder crashes when any piece of information is lost, so we have strengthened the decoder to be robust against packet losses.

At last, once the video sequence is reconstructed we compute, among other application metrics, the PSNR value relating to the original video sequence, which is the most commonly used metric for measuring the video quality.

5. Simulations Results

The purpose of this paper is to find out how different encoding modes affect the video delivery in vehicular networks, that is, to discover which of the generated bit streams achieves the best trade-off between reconstructed video quality and resulting bit rate. So, we start with an evaluation of the different INTRA refresh modes without using tile partitioning (i.e., only one tile per frame). Then we analyze the effect of increasing the number of tiles per frame with the All-Intra (AI) coding mode. In addition, finally we will see in action both INTRA refresh modes and the use of tiles to evaluate their overall behavior on error resilience, video quality and coding bit rate.

Although there are much more metrics available, we have selected the following application layer metrics:

- Goodput (GP): the total number of received video packets at the client application layer divided by the application runtime (i.e., from the sending time of the first packet until the time when the last packet is received).
- End-to-End (E2E) Delay: the average time each packet requires in order to be delivered to the destination application (running at a particular node).
- Jitter: the average E2E delay variation between two consecutive packets received at the destination node.
- Packet Delivery Ratio (PDR): the relation between the total number of video packets received by the client with respect to the video packets sent by the server. As all the video servers (RSUs) transmit the same video sequence in a synchronized and cyclic way, in this work, only the number of packets sent by the associated RSUs is considered.
- Packet Loss Ratio (PLR): the percentage of lost packets out of the total number of packets of the video sequence.
- Tile Loss Ratio (TLR): the percentage of tiles that have not been completely received out of the total number of tiles of the delivered video sequence. In case of using a 1 tile per frame layout (i.e., no-tiling), this value corresponds to the Frame Loss Ratio (FLR).
- Peak Signal-to-Noise Ratio (PSNR): the measure of the objective video quality. The PSNR is computed frame-by-frame for the luminance (Y) component as shown in Equation 1, averaged for all the frames. The Mean Squared Error (MSE) is computed for each frame n by averaging the squared intensity differences between each pixel (i, j) of the same original (Y_S) and distorted (Y_D) frame, as shown in Equation 2, where $i \in 1..N_{col}$ and $j \in 1..N_{row}$, being N_{col} and N_{row} the width and height of the frames (in pixels).

$$PSNR(n)_{dB} = 20 \cdot log_{10} \left(\frac{V_{peak}}{\sqrt{MSE(n)}} \right) \quad (1)$$

$$MSE(n) = \frac{1}{N_{col} \cdot N_{row}} \sum_{i=1}^{N_{col}} \sum_{j=1}^{N_{row}} [Y_S(n,i,j) - Y_D(n,i,j)]^2 \quad (2)$$

The metrics are collected in both scalar (minimum, maximum, mean, etc.) and vector format. Scalar metrics allow to study the overall performance of the network, whereas the vector data allows studying the evolution of such metrics over the entire simulation. To drastically reduce the huge amount of data needed to store the vector data, some of these metrics are also collected in vector format but averaged every second only; in this way, it is possible to analyze the behavior of the network in any zone defined by two given simulation times.

In this work we have analyzed the selected simulation metrics in 3 different fragments of the whole simulation of 10 s in length each:

- Zone A (t = 180..190s): an area where the client node has full coverage of one of the RSUs (`rsu[0]`).
- Zone B (t = 100..110s): a shadow area between two RSUs (`rsu[1]` and `rsu[0]`), where none of them are "visible" for the client node.
- Zone C (t = 240..250s): an area where the coverage of two RSUs are overlapped (`rsu[0]` and `rsu[2]`).

Zones B ad C are not shown, as the percentage of lost packets is too high. The location of the RSUs should be adjusted in Zone B, and a handover mechanism should be used in case of overlapping of the transmission ranges (Zone C). Therefore, only results for Zone A are shown below.

5.1. INTRA Refresh Coding Modes

In this section the effect of different INTRA refresh modes is analyzed. Since we want to determine the potential of INTRA Refresh alone, only one tile per frame is used, that is, the experiments do not use tile partitioning.

Figure 3a shows the achieved Goodput for each encoding mode under different background traffic loads. As can be seen, it affects to all the encoding modes in a similar way, proportionally to the different encoding bit rates. Only in the case of the maximum background traffic load (75 pps), the AI mode shows an abrupt descent as the network entered in saturation state, being the rest of coding modes with moderate to high network loads depending on their corresponding bit rates.

As can be seen in Figure 3b, the network Packet Delivery Ratio (PDR) is 1.0 for all the encoding modes when no background traffic is used, meaning that all the packets arrive to their destination. As the background traffic increases, the network PDR decreases for all the encoding modes, being the 50 pps background traffic load a very high network load where more than 15% of packets are lost in all coding modes. Similarly to the previous picture, all the curves are sorted by their consumed bandwidth (see Table 5).

The End-to-End (E2E) Delay shows similar behavior for all the encoding modes while increasing the background traffic (Figure 3c). The AI coding mode suffers from network saturation at the highest background traffic load, achieving average delays of up to 6 s (network saturation). The rest of coding modes show two different behaviors: (a) a soft monotonic delay increase for coding modes with the highest bit rates (i.e., IPIP and I3P), and (b) similar packet delays for the rest of modes that require less bandwidth (bit rate). Among the last coding modes we can observe that the ones with lowest bit rates seem to have less delay as background traffic increases. So, as being the coding modes with lowest bit rates they have a lower probability of sending video packets during the control channel period, reducing the average latency and also the jitter, as explained next.

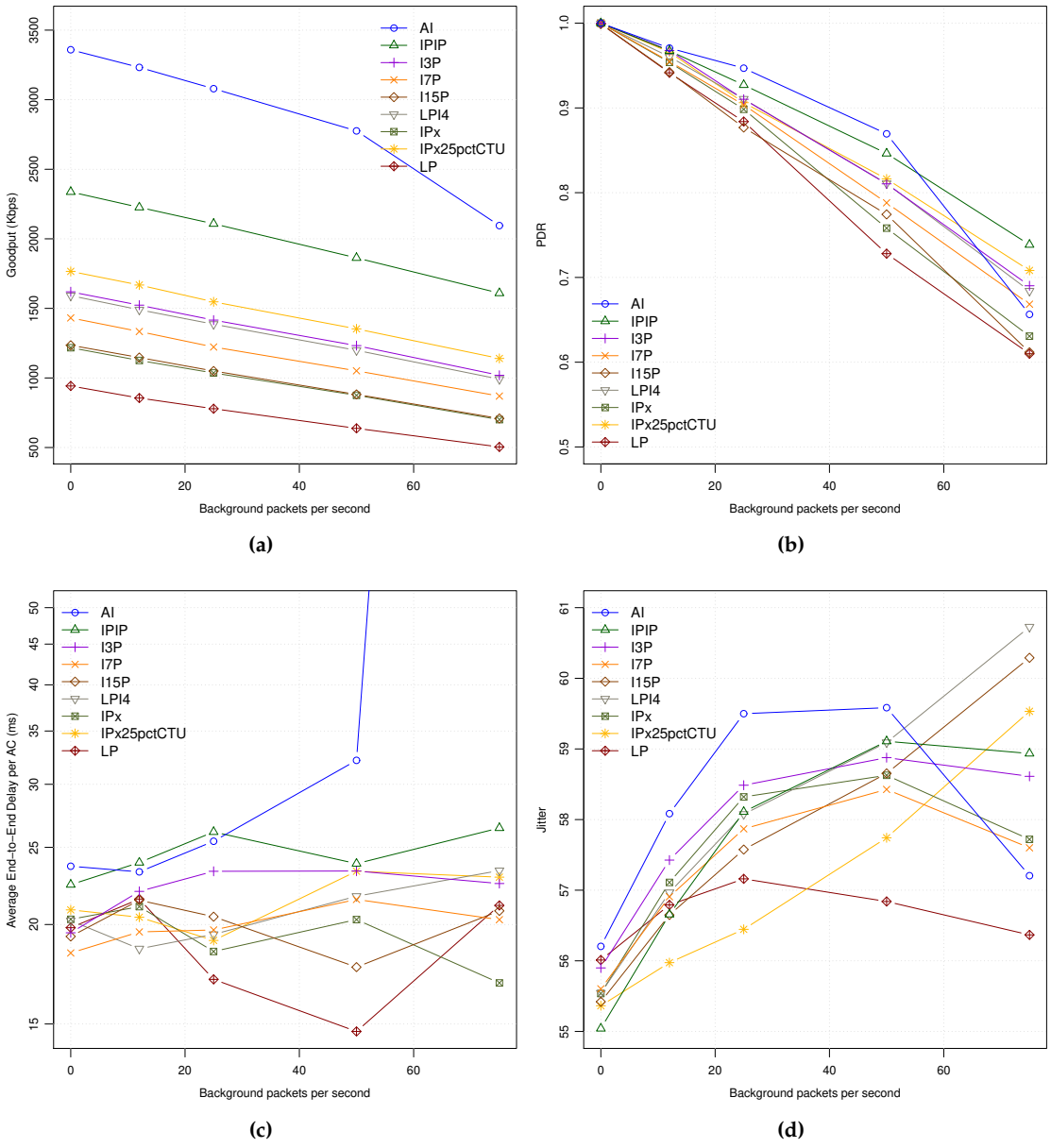

Figure 3. APP statistics for all the encoding modes: (**a**) GP. (**b**) PDR. (**c**) E2E-Delay. (**d**) Jitter.

Regarding Jitter, Figure 3d shows a high average jitter values from 55 milliseconds to more than 60 at high background traffic loads. The main reason for the high jitter observed may be due to the behavior of 802.11p MAC operation, where the time is multiplexed between service and control channels (near the 50% of channel time for each). So, all packets generated during the control channel time period have to wait until the service channel period starts, producing timing distortions in the continuous packet video delivery traffic pattern that seriously affect to Jitter.

In Figure 4a,b, the PLR and FLR application metrics are shown. The PLR follows a behavior coherent with the PDR plot described before: the higher the coding mode bit rate, the lower the PLR value. However, when analyzing the FLR, we can observe that the number of lost frames is inversely proportional to the coding mode bit rates under similar network load conditions. This is due to the bit stream packetization process that produces a higher number of packets per encoded frame for coding

modes with higher bit rates. So, the probability of losing one frame is higher, producing more frame losses. In Figure 4c, the PSNR values for all the encoding modes of a video sequence received in Zone A are plotted. As can be seen, the highest video quality is achieved by the AI coding mode followed by IPIP and I3P coding modes. However, as background traffic increases, the PSNR values fall down under unacceptable quality levels (below 28 dBs) for most video contents. On the other hand, LP and IPx have the worst PSNR values; when the first packet losses appear, the video is degraded, being unable to stop error propagation due to their poor INTRA refresh capabilities.

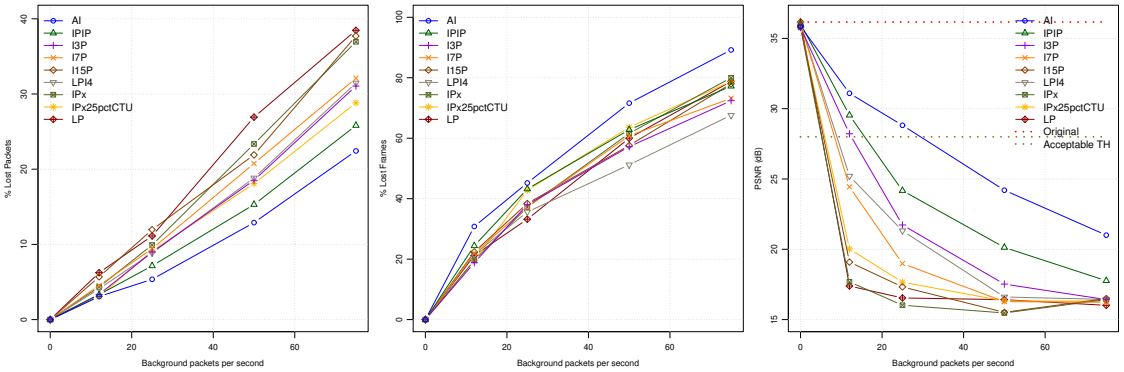

Figure 4. QoE statistics for all the encoding modes: (**a**) PLR. (**b**) FLR. (**c**) PSNR.

These results demonstrate the benefits of using INTRA refresh encoding modes for improving the reconstructed video quality. However, the price we have to pay for is the increased bit rates, which can cause a greater loss of packets when the network gets saturated.

5.2. Tile Partitioning for the All-Intra Coding Mode

After analyzing the INTRA refresh coding modes with their pros and cons, we proceed to study another error resilience technique that may efficiently work together with INTRA refresh. We propose the use of frame partitioning that allow fragmenting each frame into independently encoded/decoded tiles. The idea is to significantly reduce the number of frame losses. In this section we will study the effect of using 1, 2 , 4, 6, 8, 10 and 16 tiles per frame with the AI coding mode.

As it was explained in the introduction section, using tiles introduces an overhead in the bit stream as required signaling in form of headers, so higher bit streams are produced. From Table 6, the use of tiles for the AI coding mode increase the bit rate from 0.6% to 6.8% when using 2 and 16 tiles, respectively. With respect to the reconstructed video quality, we can see that using tiles has no effect.

Table 6. Bit rate and PSNR for AI for different tiles per frame layouts.

Mode	QP	Tiles	Bit rate (Mbps)	PSNR (dB)
AI	31	1	3.417	35.863
AI	31	2	3.438	35.866
AI	31	4	3.483	35.863
AI	31	6	3.516	35.868
AI	31	8	3.543	35.864
AI	31	10	3.578	35.862
AI	31	16	3.648	35.862

In Figure 5, we show the simulation results with the same network metrics used in previous experiments. Figure 5a shows the Goodput for the different tile layouts. As expected, at low background traffic levels the curves are sorted by their corresponding bit rate. However, when

background traffic increases with values greater or equal than 50 pps, and we use a high number of tiles per frame, the trend is inverted. Again, this is because the network begins to be saturated, and this situation affects more to the video sequences that require more bandwidth. The same behavior can be observed in the PDR metric shown in Figure 5b.

As shown in Figure 5c, the average E2E delay keeps similar values for all the tile number configurations at low to moderate background traffic levels, showing always slightly higher delay values when (a) increasing the number of tiles, and (b) the background traffic level increases. Notice that at 50 pps background traffic loads, only the configurations of up to 4 tiles maintain the same behavior than the one explained before at lower background traffic levels, being a symptom of entering network saturation for the AI encoding mode.

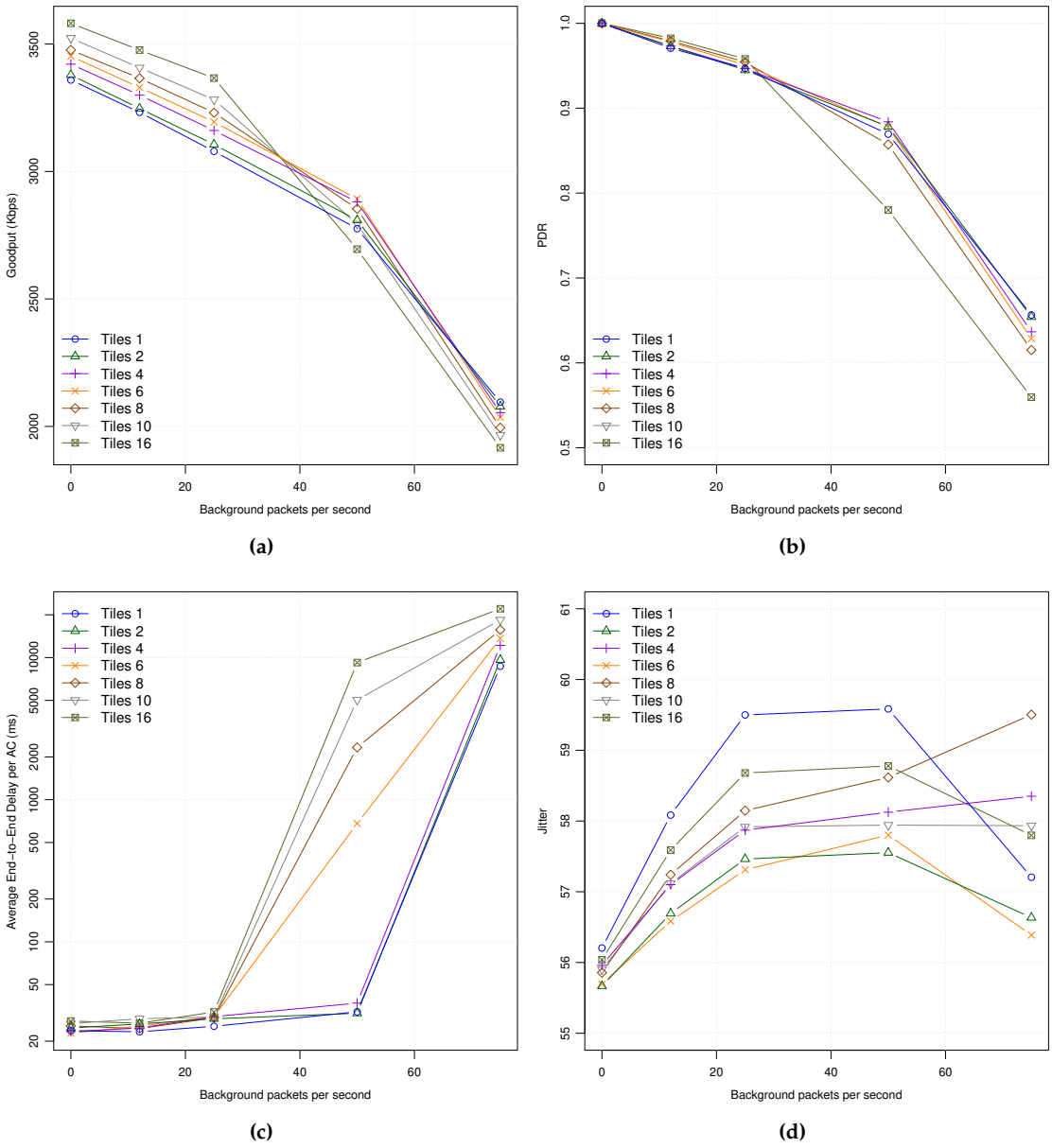

Figure 5. APP statistics for AI coding mode and for all the tiles per frame layouts: (**a**) GP. (**b**) PDR. (**c**) E2E-Delay. (**d**) Jitter.

Sensors **2018**, *18*, 3107 14 of 18

Figure 5d shows the average jitter varying the number of tiles per frame. As it can be seen, the experienced jitter is always lower on average than the one obtained with just one tile, observing better response when using a low number of tiles per frame (from 2 to 6).

From Figure 6a, we may observe a lineal increase of PLR up to the network saturation point, where a higher number of tiles always produce a lower PLR. However, in Figure 6b the TLR metric shows an interesting behavior, achieving lower TLR values when the number of tiles per frame increases. As a consequence, the video quality of the received video will be better as shown in Figure 6c. So, we can conclude that when the number of tiles per frame increases, the error resilience of the video stream also increases, obtaining acceptable video quality levels from low to moderate background traffic levels. Taking into account that the more tiles per frame the more overhead in the bit stream, several tiles of 4 or 6 would be a good trade-off.

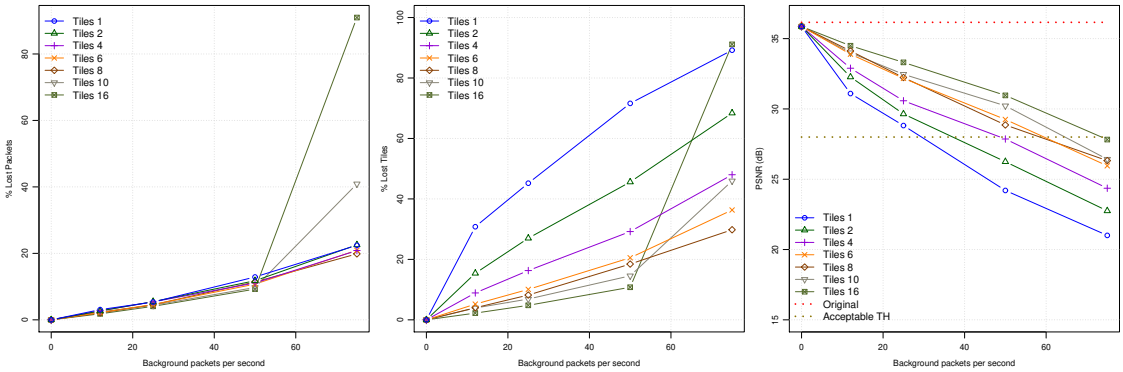

Figure 6. QoE statistics for AI for all the tiles per frame layouts: (**a**) PLR. (**b**) TLR. (**c**) PSNR.

5.3. Global Evaluation

Now, we are going to analyze the combined effect of both INTRA refresh coding modes and tiles. So, we will show the PLR, TLR and PSNR metrics for all the encoding modes as a function of the number of tiles for every coding mode for a particular background traffic level. From the previous experiments, only 12, 25, and 50 pps background traffic levels will be explored in Figure 7, since at higher traffic loads the video quality levels are unacceptable.

As it can be seen in Figure 7, there is a general behavior from low to moderate-high background traffic loads where, as the number of tiles increases (a) the PLR decreases, especially fast with the lowest bit rate coding modes (LP coding mode reduces PLR to the half by using 6 tiles); (b) the TLR also decreases at the same pace for all the coding modes, significantly reducing TLR six times for 12 pps background traffic load; and (c) the resulting PSNR also shows improvements as the number of tiles increases for all coding modes. At the other hand, when the background traffic load increases, both PLR and TLR also increases, following the same pattern as depicted before. As a consequence, the PSNR video quality decreases in general (e.g., IPIP coding mode using 6 tiles reduces PSNR in 2.5 dBs when going from 12 pps to 25 pps network load). If we define a threshold for minimum accepted video quality in 28 dBs, AI, IPIP, I3P and LPI4 will be the only coding modes that keep over that quality threshold when working with a frame partitioning of 4 tiles and low network loads (12 pps). The I7P coding mode could belong to the selected group when using 6 tiles per frame. The rest of coding modes will not satisfy the minimum quality requirement. At moderate network loads (25 pps), only AI and IPIP will be above the defined video quality threshold for 4 tiles per frame, discarding all the rest of modes. In addition, finally, at high network loads (close to saturation) only the AI coding mode will be the one that remains over the quality threshold with 6 tiles. If the number of tiles were not an issue, the IPIP coding mode could be also selected using at least 8 tiles per frame.

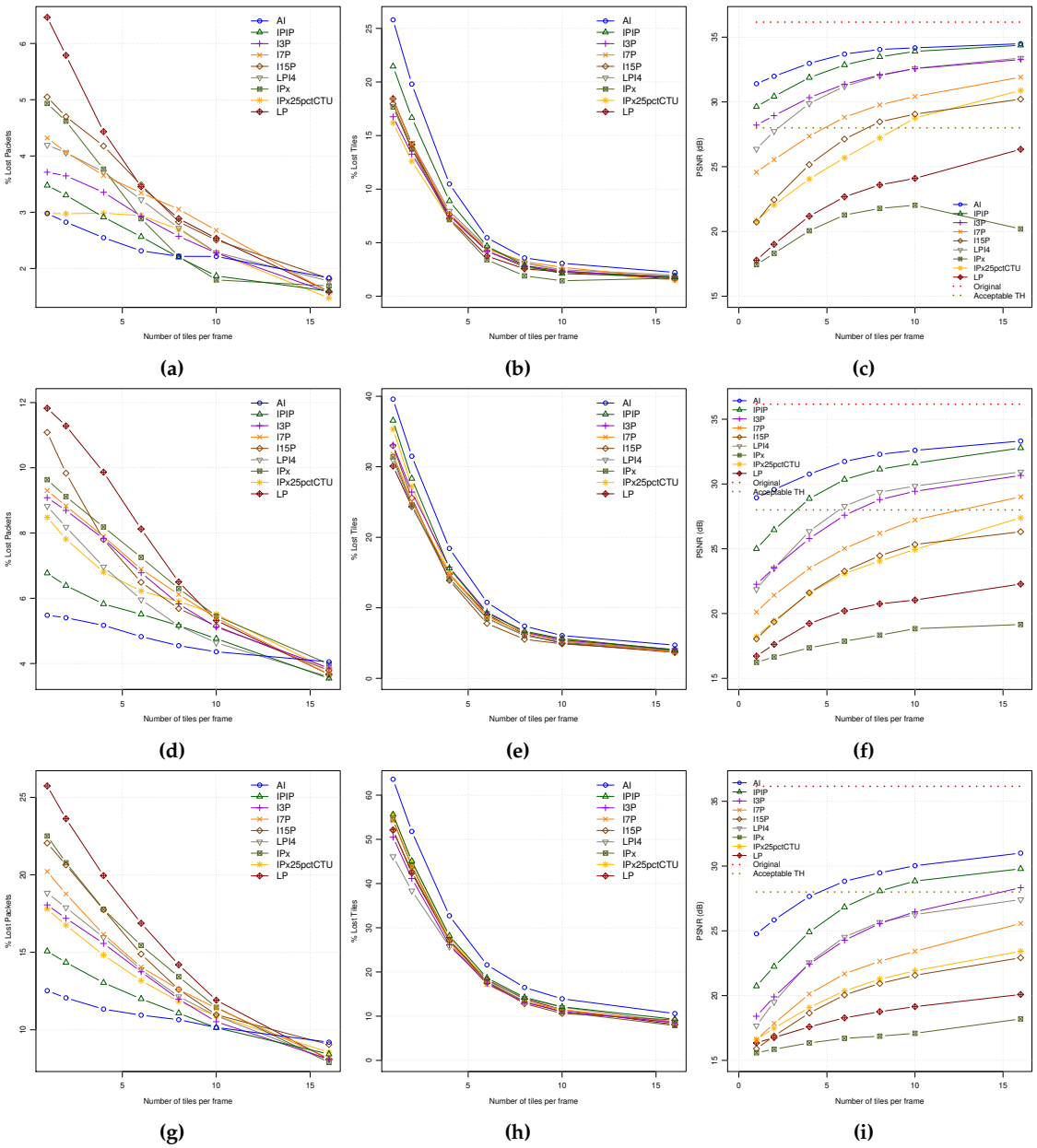

Figure 7. QoE statistics for all the encoding modes for all the tiles per frame layouts at different background traffic levels: (**a,b,c**) PLR, TLR, PSNR (12 pps). (**d,e,f**) PLR, TLR, PSNR (25 pps). (**g,h,i**) PLR, TLR, PSNR (50 pps).

From this study, we may conclude that the coding modes with higher "intra refresh" degree are the ones that better stand up in front of packet losses, but they are the ones that higher network bandwidth will require. On the other hand, when using tiles, the behavior is very similar for all the coding modes, resulting in PSNR improvements. However, the consequence of using a higher number of tiles, as for high intra refreshing coding modes, is also a bit stream size increase. So, a trade-off should be defined between error resilience and bandwidth (network resources), what should be finally determined by the application requirements. We may propose a selection criteria of using a frame partitioning in no more than 6 tiles (no significant improvements are found beyond), and the coding mode that, remaining over the application quality threshold, requires the lowest bit rate.

6. Conclusions

In this work, we have introduced VDSF-VN, a simulation framework based on the OMNeT++ network simulator, the Veins framework and the SUMO mobility traffic simulator, which was designed to study and evaluate the robustness of a video delivery stream in vehicular ad-hoc network scenarios. Using the proposed simulation framework we analyzed different coding options, available at HEVC video encoder, to improve the robustness of the resulting video stream when it is faced with error-prone network scenarios like urban vehicular networks. We have tested different INTRA refresh coding modes and frame partitioning schemes by means of tiling, so we can determine the configurations that better performance and robustness provide at different network loads.

The experimental results show that error resilience improves as the INTRA refresh level increases. So, the best coding mode would be AI followed by IPIP, I3P, and LPI4. Notice that these encoding modes require high bit rates with respect the others, so a trade-off between error resilience and final bit rate should be defined, which strongly depends on the application requirements and the available network resources. With respect to frame partitioning, we have observed that the use of tiles has a positive impact when combined with INTRA refresh coding modes. So, acceptable video quality levels are achieved from low to moderate background traffic loads, reinforcing the properties of the INTRA refresh coding modes. Again, a trade-off should be applied since the use of tiles increases the bit stream, so the recommendation is not to use more than 6 tiles per frame since when using more tiles the improvements are, in general, negligible.

The main conclusion is that by using different INTRA refresh options combined with appropriate tile coding we will improve the protection of video streaming in VANET urban scenarios, allowing to keep acceptable video quality at high packet loss rates. We can use these error resilience techniques at encoding/decoding sides to protect the video streams. However, we need to explore additional techniques, like channel coding (FEC), the use of QoS at MAC level, and efficient error concealment, which, combined with the ones proposed here, minimize the introduced bit stream overhead and maximize the final error resilience. The final goal is focused to provide robust and efficient video streams which are able to fight against moderate to high packet network error conditions, keeping the video quality over the threshold demanded by the application. As future work, we are planning to use the available MAC QoS features, and later we will extend our error resilience architecture with packet-based FEC proposals, and advanced error concealment strategies.

Author Contributions: Funding acquisition, O.L.G.; Investigation, P.P.G.A.; Software, P.P.G.A. and P.P.; Supervision, M.P.M. and P.P.; Validation, M.P.M. and P.P.; Writing—original draft, P.P.G.A., M.P.M., P.P. and O.L.G.; Writing—review & editing, P.P.G.A., M.P.M., P.P. and O.L.G.

Acknowledgments: This work has been supported by the Spanish Ministry of Economy and Competitiveness under Grant TIN2015-66972-C5-4-R, co-financed by FEDER funds (MINECO/FEDER/UE).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Garrido Abenza, P.P.; Piñol Peral, P.; P. Malumbres, M.; López-Granado, O. Simulation Framework for Evaluating Video Delivery Services over Vehicular Networks. In Proceedings of the IEEE 88th Vehicular Technology Conference (VTC-Fall), Chicago, IL, USA, 27–30 August 2018.
2. Varga, A.; Hornig, R. An overview of the OMNeT++ Simulation Environment. In Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops, Marseille, France, 3–7 March 2008.
3. Sommer, C.; German, R.; Dressler, F. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Trans. Mob. Comput.* **2011**, *10*, 3–15. [CrossRef]
4. Krajzewicz, D.; Erdmann, J.; Behrisch, M.; Bieker, L. Recent Development and Applications of SUMO - Simulation of Urban MObility. *Int. J. Adv. Syst. Meas.* **2012**, *5*, 128–138.
5. High Efficiency Video Coding (HEVC). ITU-T Recommendation H.265, 2013. Available online: http://www.itu.int/rec/T-REC-H.265 (accessed on 13 September 2018).

6. Joint Collaborative Team on Video Coding (JCT-VC). HEVC reference software HM (HEVC Test Model) and Common Test Conditions. Available online: https://hevc.hhi.fraunhofer.de (accessed on 13 September 2018).
7. Vineeth, N.; Guruprasad, H.S. A Survey on the Techniques Enhancing Video Streaming in VANETs. *Int. J. Comput. Networking Wireless Mobile Commun.* **2013**, *3*, 37–46.
8. Wang, Y.; Wenger, S.; Wen, J.; Katsaggelos, A.K. Error resilient video coding techniques. *IEEE Signal Process Mag.* **2000**, *17*, 61–82. [CrossRef]
9. DeBrunner, V.; DeBrunner, L.; Wang, L.; Radhakrishnan, S. Error control and concealment for image transmission. *IEEE Commun. Surv. Tutorials* **2000**, *3*, 2–9. [CrossRef]
10. Cui, Z.; Gan, Z.; Zhan, X.; Zhu, X. Error concealment techniques for video transmission over error-prone channels: a survey. *J. Comput Inf. Syst.* **2012**, *8*, 8807–8818.
11. Kung, W.Y.; Kim, C.S.; Kuo, C.C.J. Spatial and Temporal Error Concealment Techniques for Video Transmission Over Noisy Channels. *IEEE Trans. Circuits Syst. Video Technol.* **2006**, *16*, 789–803. [CrossRef]
12. Stockhammer, T.; Hannuksela, M.M.; Wiegand, T. H.264/AVC in wireless environments. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 657–673. [CrossRef]
13. Bandyopadhyay, S.K.; Wu, Z.; Pandit, P.; Boyce, J.M. An Error Concealment Scheme for Entire Frame Losses for H.264/AVC. In Proceedings of the 2006 IEEE Sarnoff Symposium, Princeton, NJ, USA, 27–28 March 2006. [CrossRef]
14. Girod, B.; Farber, N. Feedback-based error control for mobile video transmission. *Proc. IEEE* **1999**, *87*, 1707–1723. [CrossRef]
15. Kim, C.S.; Lee, S.U. Multiple description coding of motion fields for robust video transmission. *IEEE Trans. Circuits Syst. Video Technol.* **2001**, *11*, 999–1010. [CrossRef]
16. Hagenauer, J.; Stockhammer, T. Channel coding and transmission aspects for wireless multimedia. *Proc. IEEE* **1999**, *87*, 1764–1777. [CrossRef]
17. Wu, J.; Tan, R.; Wang, M. Streaming High-Definition Real-Time Video to Mobile Devices With Partially Reliable Transfer. *IEEE Trans. Mob. Comput.* **2018**. [CrossRef]
18. Minder, L.; Shokrollahi, A.; Watson, M.; Luby, M.; Stockhammer, T. RaptorQ Forward Error Correction Scheme for Object Delivery. Available online: https://www.rfc-editor.org/rfc/pdfrfc/rfc6330.txt.pdf (accessed on 13 September 2018).
19. Pasin, M.; Petracca, M.; Bucciol, P.; Servetti, A.; Martin, J.C.D. A survey of error-concealment schemes for real-time audio and video transmissions over the Internet. In Proceedings of the 4th Biennial Workshop on DSP for In-Vehicle Systems and Safety, Dallas, TX, USA, 1–2 July 2009.
20. Zhang, R.; Regunathan, S.L.; Rose, K. Video coding with optimal inter/intra-mode switching for packet loss resilience. *IEEE J. Sel. Areas Commun.* **2000**, *18*, 966–976. [CrossRef]
21. Calafate, C.T.; Malumbres, M.P.; Manzoni, P. Performance of H.264 compressed video streams over 802.11b based MANETs. In Proceedings of the 24th International Conference on Distributed Computing Systems Workshops, Tokyo, Japan, 23–24 March 2004; pp. 776–781. [CrossRef]
22. Chen, H.; Zhao, C.; Sun, M.T.; Drake, A. Adaptive Intra-refresh for Low-delay Error-resilient Video Coding. *J. Visual Commun. Image Represent.* **2015**, *31*, 294–304. [CrossRef]
23. Klaue, J.; Rathke, B.; Wolisz, A. *EvalVid – A Framework for Video Transmission and Quality Evaluation*; Springer: Berlin, Heidelberg, 2003. [CrossRef]
24. Ke, C.; Shieh, C.; Hwang, W.; Ziviani, A. An Evaluation Framework for More Realistic Simulations of MPEG Video Transmission. *J. Inf. Sci. Eng.* **2008**, *24*, 425–440.
25. Saladino, D.; Paganelli, A.; Casoni, M. A tool for multimedia quality assessment in NS3: QoE Monitor. *Simul. Modell. Pract. Theory* **2013**, *32*, 30–41. [CrossRef]
26. Rosário, D.; Zhao, Z.; Silva, C.; Cerqueira, E.; Braun, T. An OMNeT++ Framework to Evaluate Video Transmission in Mobile Wireless Multimedia Sensor Networks. In Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques, Cannes, France, 6–8 March 2013; pp. 277–284.
27. Pediaditakis, D.; Tselishchev, Y.; Boulis, A. Performance and scalability evaluation of the Castalia wireless sensor network simulator. In Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, Malaga, Spain, 16–18 March 2010.
28. Nightingale, J.; Wang, Q.; Grecos, C. HEVStream: a framework for streaming and evaluation of high efficiency video coding (HEVC) content in loss-prone networks. *IEEE Trans. Consum. Electron.* **2012**, *58*, 404–412. [CrossRef]

29. Garrido Abenza, P.P.; P. Malumbres, M.; Piñol Peral, P. GatcomSUMO: A Graphical Tool for VANET Simulations Using SUMO and OMNeT++. In Proceedings of the 2017 SUMO User Conference, Berlin, Germany, 8–10 May 2017; pp. 113–133.
30. Wang, Z.; Simoncelli, E.P.; Bovik, A.C. Multiscale structural similarity for image quality assessment. In Proceedings of the Thirty-Seventh Asilomar Conference on Signals, Systems Computers, Pacific Grove, CA, USA, 9–12 November 2003; pp. 1398–1402. [CrossRef]
31. Haklay, M.M.; Weber, P. OpenStreetMap: User-Generated Street Maps. *IEEE Pervasive Comput.* **2008**, *7*, 12–18. [CrossRef]
32. Krauss, S. Microscopic Modeling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics. Ph.D. Thesis, University of Cologne, Cologne, Germany, April 1998.
33. Misra, K.; Segall, A.; Horowitz, M.; Xu, S.; Fuldseth, A.; Zhou, M. An Overview of Tiles in HEVC. *IEEE J. Sel. Top. Sign. Proces.* **2013**, *7*, 969–977. [CrossRef]
34. Seeling, P.; Reisslein, M. Video Transport Evaluation With H.264 Video Traces. *IEEE Commun. Surv. Tutorials* **2012**, *14*, 1142–1165. [CrossRef]
35. Wang, Y.; Sanchez, Y.; Schierl, T.; Wenger, S.; Hannuksela, M. RTP Payload Format for High Efficiency Video Coding. Available online: https://www.rfc-editor.org/rfc/pdfrfc/rfc7798.txt.pdf (accessed on 13 September 2018).