MDPI

Article

Source Coding Options to Improve HEVC Video Streaming in Vehicular Networks

P. Pablo Garrido [1]*, Manuel P. Malumbres [1], Pablo Piñol [1] and O. López-Granado [1]

[1] Department of Physics and Computer Architecture, Miguel Hernández University, 03202 Elche, Alicante, Spain; pgarrido@umh.es (P.P.G.A.); 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

Academic Editor: name
Version August 20, 2018 submitted to Sensors

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) 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 quality of video delivery in this kind of networks. The experimental results show that packet losses significantly reduces video quality when low level of interference is found in an urban scenario. By using different INTRA refresh options combined with appropriate tile coding we will protect at some extent the video delivery 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, contextaware 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 the vehicles, the 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, specially 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 allow 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 [6]. In particular, we have developed a tuned version of the HEVC reference software HM (HEVC Test Model) [7]. 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 become 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. At one hand, INTRA refresh technique will help to stop the error propagation produced by a single packet lose in the prediction process of future (next) video frames. So, we have analyzed the behavior of several INTRA refresh approaches in order to determined the ones that higher video quality deliver to the user with an acceptable bitrate overhead. At the other hand, TILE-based coding allows to partitioning each video frames in blocks (Tiles), being completely independent among them in terms of coding/decoding processing. This may be useful to mitigate the impact of packet losses when comparing to no-tiling approaches. When no-tiling approaches (the default coding option) are used, a frame is encoded and typically fragmented in several network packets. In order to decode the frame all packets need to be received. If just one packet is lost, the frame is undecodable although the rest of packets have been correctly received. If tiling is used, a frame could 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 HEVC coding options that better video quality provides to the user, showing the importance they deserve when dealing with video delivery in VANET scenarios.

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 has been proposed in the literature to improve video streaming over not reliable medium like the wireless channel in general, and VANETs in particular [8].

Error Concealment (EC) methods try to minimize the impact of packets loss in the perceived quality of the received video sequence [9] [10] [11]. If any part of an encoded frame is missing, then the entire frame cannot be decoded. These methods try to deduce the missing parts and correct them by exploiting spatial similarities of the nearby areas, or temporal redundancies in the video sequence [12]. Spatial prediction can be only used for small areas [13], whereas the temporal prediction can replace a whole missing frame, replacing it with a previously received frame; this is known as 'Frame Copy Concealment' [14]. In order to make an estimation of the contents of a missing area, techniques based on Motion Vectors (MV) information can also be used.

Error Resilience (ER) are another group of techniques to achieve robustness of video sequences, which are applied at the sender side. One of these is the Feedback Channel, in which the receiver can request the retransmission of some missing part [15]. Another methods make use of substreams, such

as Multiple Description Coding (MDC) [16] or Layered Coding (LC). There has been a lot of research made to protect video sequences with Forward Error Correction (FEC), also known as Channel Coding [17]. This technique insert redundant data, named Error Correction Codes (ECCs), into the encoded video. These ECC 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) [18]. RaptorQ codes is an example of this kind of mechanism [19]. FEC can be used in all kind of networks, including VANETs [20].

Another group of ER techniques are the so-called 'intra refresh' methods. They encode the bit stream in such a way that it avoids fast quality degradation in the presence of data losses. An inter-coded frame, a frame that 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 become bit streams more robust. There are several works in the literature that exploit these techniques to improve video robustness like the ones found in [21] [22] [23].

To evaluate the performance of all 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 employ the Peak Signal-to-Noise Ratio (PSNR) metric. So, the error protection of our encoder will be based on those encoding options that best received video quality throws taking always into account the bitrate overhead that typically they provide.

There are several simulation frameworks proposed in the literature that follow our requirements, like EvalVid [24], 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 [25] EvalVid is integrated with ns-2. In [26] 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 [27] 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 [28].

Another example is HEVStream [29], 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 as 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 bitstream 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.). And, 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, 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 motivated us to develop the VDSF-VN [1] simulation framework that it is especially suited for vehicular networks. VDSF-VN will be the simulation framework we are going to use in this work, so we provide a brief introduction in the following section.

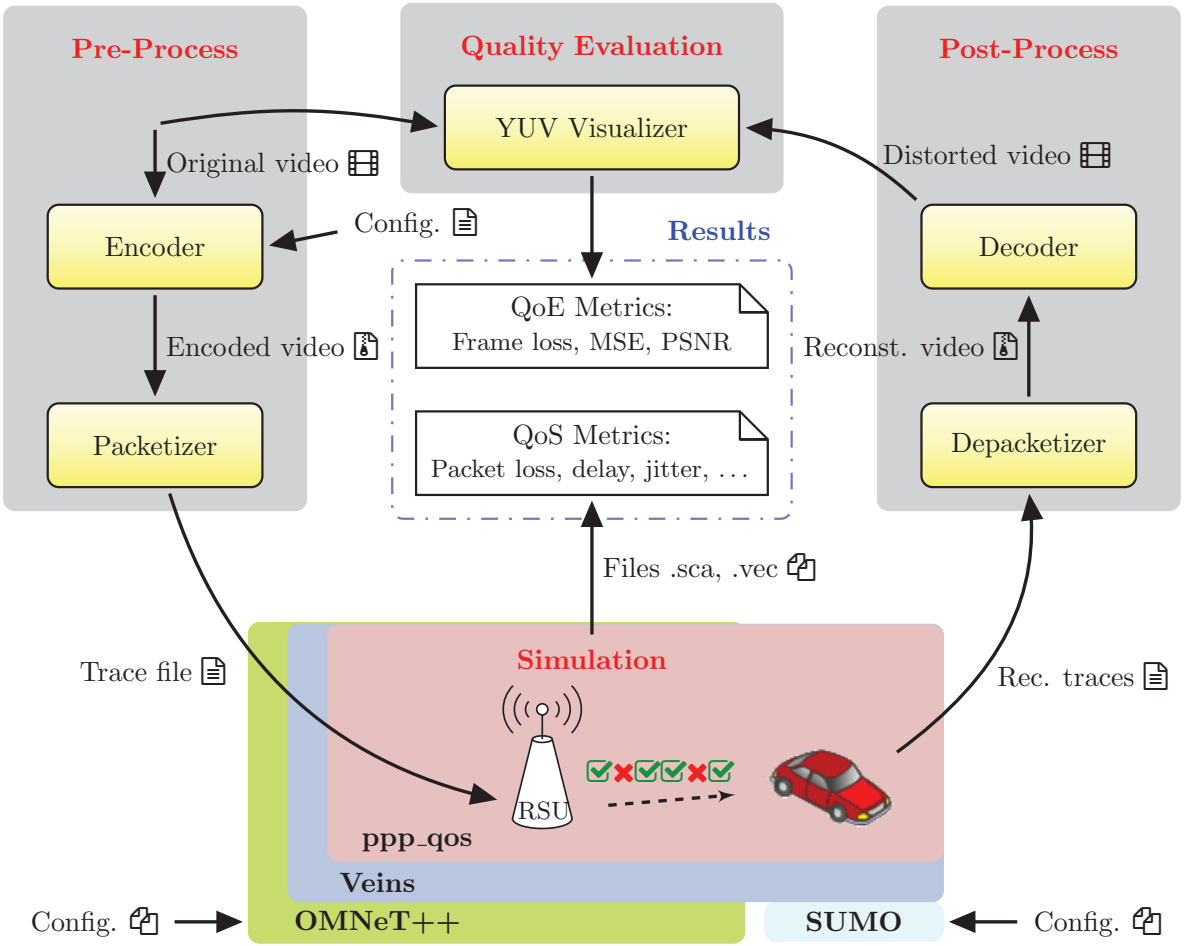

Figure 1. Workflow of VDSF-VN

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" [30] application was also developed to ease all the configuration duties by means of a Graphical User Interface (GUI) .

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, even 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: In charge of encoding the desired video sequences and then perform the Real-time Transmission Protocol (RTP) packetization of the encoded bitstream and generate the corresponding video trace file. This step is a high 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: 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 Veins project. 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: to get all the statistics defined at different network levels: Physical, MAC, and Application, with the corresponding plots. In order to compute application performance metrics like Frame Loss Rate (FLR) and PSNR, the bitstream needs to be conformed with the received packets and then decoded to obtain the reconstructed video that will be rendered to the user.

4. Performance evaluation

This section describes the methodology followed during the simulation and the experiments conducted with the proposed VDSF-VN framework for evaluating the video streaming in an urban VANET scenario. In the following, we will describe the simulation setup where the network scenario, network simulation parameters, the video source, and the video encoding options are explained.

4.1. Scenario description

The VDSF-VN includes a tool, GatcomSUMO [30], that allows to define and configure the network scenario that will be used in the simulation. Different vehicular network scenarios may be designed, regular (mesh, spider, etc.) or irregular (random). But also it is able to import real urban vehicular scenarios from OpenStreetMap [5], 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 on a square area sized 2000x2000 m. (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. In order 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.

On the other hand, several mobile vehicles are defined, which travels 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 to the first one (`node[1..10]`),

Table 1. Simulation parameters

Parameter	Value
Simulation area	2000 x 2000 m^2
Simulation time	340 s
# of RSUs	3
# of client vehicles	1
# 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
Bitrate	18 Mbps
Transmit power	20 mW
RX Sensitivity	-89 dBm
Communication range	510.87 m
MAC queues size	0 (infinite)

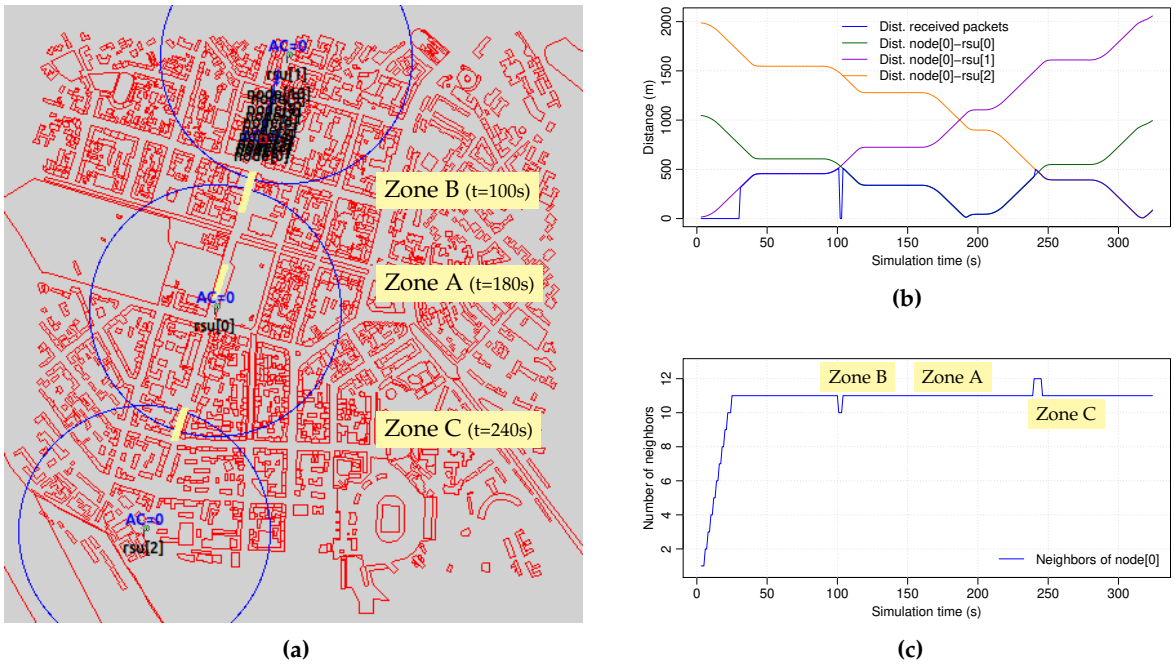

Figure 2. (**a**) Scenario. (**b**) Distance from client to RSUs. (**c**) Number of client neighbors

which send packets at different bitrates 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, in average, the total aggregate background traffic is {0,120,250,500,750} pps.

All the vehicles travel at a maximum speed of 14 m/s (50 km/h). The simulation time is 340 seconds, which is time enough to all the vehicles travel from the beginning to the end of the avenue, receiving the video sequences sent by all 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 at all times. The plot also shows the time when passing through three selected zones that 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 video sequence sent by the video servers is 'BasketballDrill', which belongs to the HEVC Common Test Conditions set [7]. As shown in Table 3, it has a resolution of 832x480 pixels and a duration of 10 seconds. 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 x 480 pixels
Duration	10 s
Length	250 frames
Rate	25 frames per second

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)

This video sequence was encoded with the HEVC reference software HM (HEVC Test Model) [7]. The resulting bitstream consists on 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 bitstreams 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 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 can not correctly predict the content of 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) that be 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 bitstreams with high video quality. In our experiments, the video quality, measured in terms of Peak Signal-to-Noise Ratio (PSNR), for all the generated bitstreams 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.

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

Mode	QP	Tiles	Bitrate (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

Apart from the compression mode and QP value, the video encoder accepts many other parameters. For example, HEVC allows to split a frame into a number of 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 lose encoding performance since spatial correlation is limited to the slice/tile domain, and their use require extra bitstream signaling information [31].

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

- Encoding modes (x9): {AI, IPIP, I3P, I7P, I15P, IPx, IPx25pctCTU, LPI4, LP} (see Table 4).
- QP values (x1): the QP value used for each encoding mode was fixed to achieve the desired video quality (PSNR $\approx$ 36 dB) when using 1 tile per frame (see Table 5).
- Number of tiles per frame (x7): seven values were considered to encode each bitstream: {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: {1x1, 1x2, 2x2, 2x3, 2x4, 2x5, 4x4}, respectively.

So, all of these combinations make a total of 63 different bitstreams (9x1x7). As can be appreciated in Table 5, the PSNR value of the original encoded video are approximately the same for all of them, whereas each case has a different bitrate value. As an example, the greatest bitstream is achieved with AI mode (3.417 Mbps), whereas the lowest corresponds with 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 bitstream [32]. 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 packet payload, and the total number of fragments of the current frame.

We have modified the HM software to include RTP bitstream packetization [33]. 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 bitstreams achieve the best trade-of between reconstructed video quality and resulting bitrate. 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 will analyze the effect of increasing the number of tiles per frame with the All-Intra (AI) coding mode. And finally we will see in action both INTRA refresh modes and the use of tiles to evaluate their overall behavior on video resilience, video quality and coding bitrate.

For each simulation we have collected and analyzed several simulation metrics in a fragment of the whole simulation, starting at 180 seconds from simulation start (Zone A). Zones B ad C are not shown, as the % 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).

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 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 the video packets sent by the server. As all the video servers (RSUs) transmits 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 received all the corresponding packets 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 will be the Frame Loss Ratio (FLR).
- Peak Signal-to-Noise Ratio (PSNR): the objective quality measure of the video quality.

5.1. INTRA refresh coding modes

In this section the effect of different INTRA refresh modes is analyzed. In this case, 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 rates. As can be seen, it affects to all the encoding modes in a similar way, proportionally to the different encoding bitrates. 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 bitrates.

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 of network saturation at the highest background traffic load, achieving average delays of up to 6 seconds (network saturation). The rest of coding modes show two different behaviors, (a) a soft monotonic delay increase for coding modes with the highest bitrates (i.e. IPIP and I3P) and (b) similar packet delays for the rest of modes that require less bandwidth (bitrate). Among the last coding modes we can observe that the ones with lowest bitrates seems to have less delay as background traffic increases. So, as being the coding modes with lowest bitrates 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.

With regard the 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 Figures 4a and 4b, the PLR and FLR application metrics are shown. The PLR follows a behavior coherent with the PDR plot described before: higher the coding mode bitrate, lower the PLR

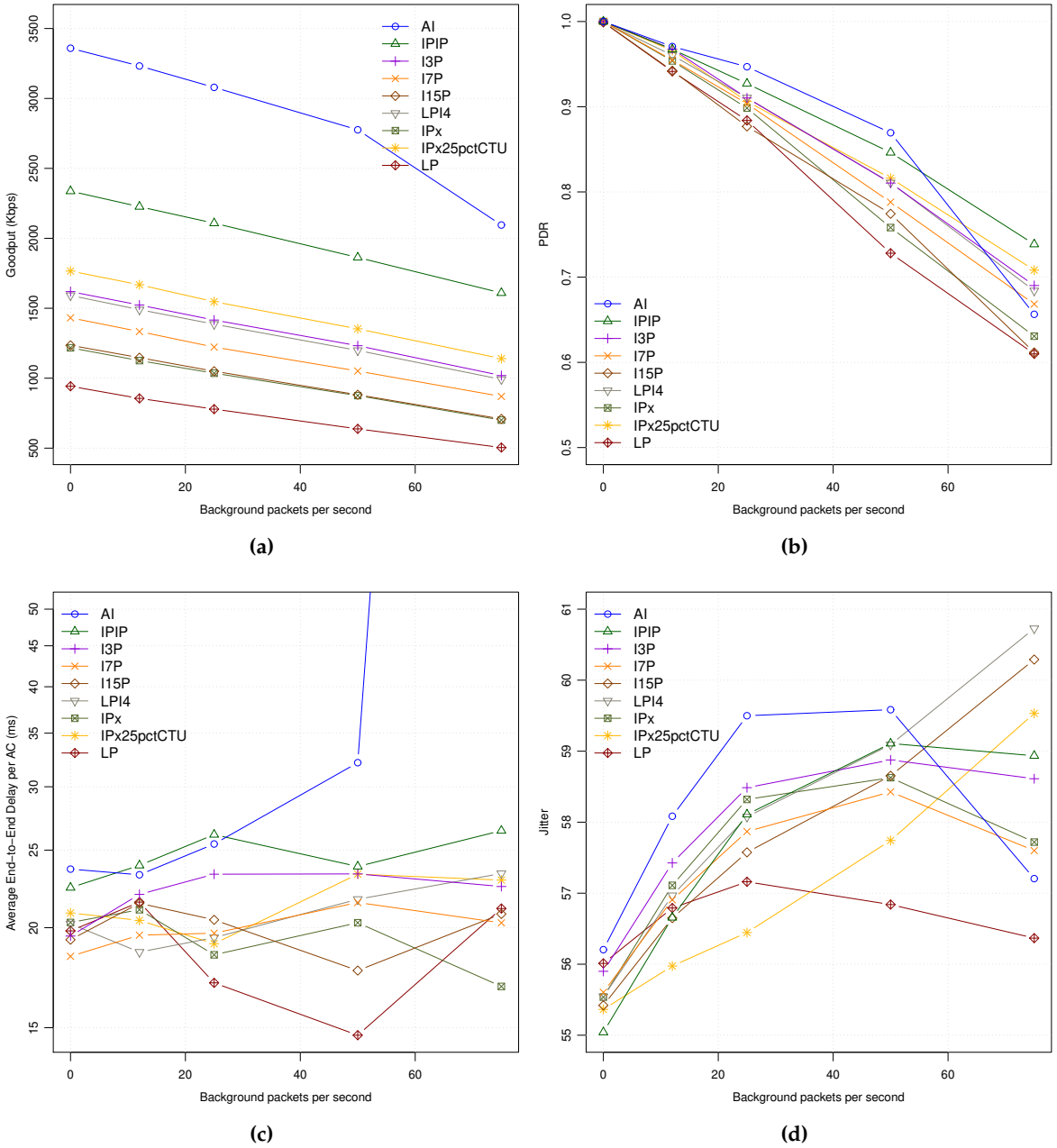

Figure 3. APP statistics for all the encoding modes: (**a**) GP. (**b**) PDR. (**c**) E2E-Delay. (**d**) Jitter

value. However, when analyzing the FLR, we can observe that the number of lost frames is inversely proportional to the coding mode bitrates under similar network load conditions. This is due to the bitstream packetization process that produces a higher number of packets per encoded frame for coding modes with higher bitrates. So, the probability of loosing 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, 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.

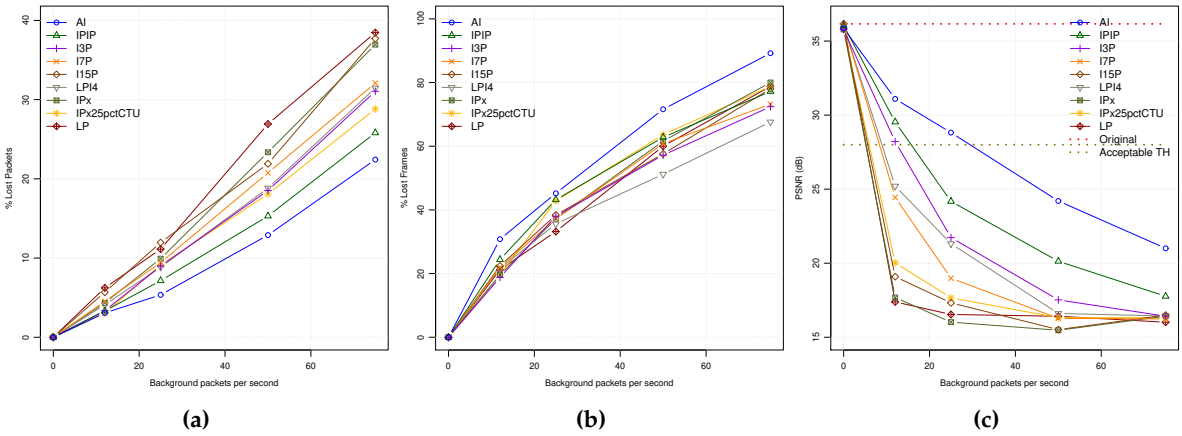

Figure 4. QoE statistics for all the encoding modes: (**a**) PLR. (**b**) FLR. (**c**) PSNR

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

Mode	QP	Tiles	Bitrate (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

These results demonstrate the benefits of using INTRA refresh encoding modes for improving the reconstructed video quality. However, the price we have to pay is the increased bitrates, which can cause a greater loss of packets in case of 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 techniques that allow fragmenting each frame in independently encoded/decoded tiles. The idea is 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 explained in the introduction section, using tiles introduces an overhead in the bitstream as required signaling in form of headers, so higher bitstreams are produced. From Table 6, the use of tiles for the AI coding mode increase the bitrate from 0.6% to 6.8% when using 2 and 16 tiles, respectively. With respect the reconstructed video quality, we can see that using tiles has no effect.

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 bitrate. 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 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

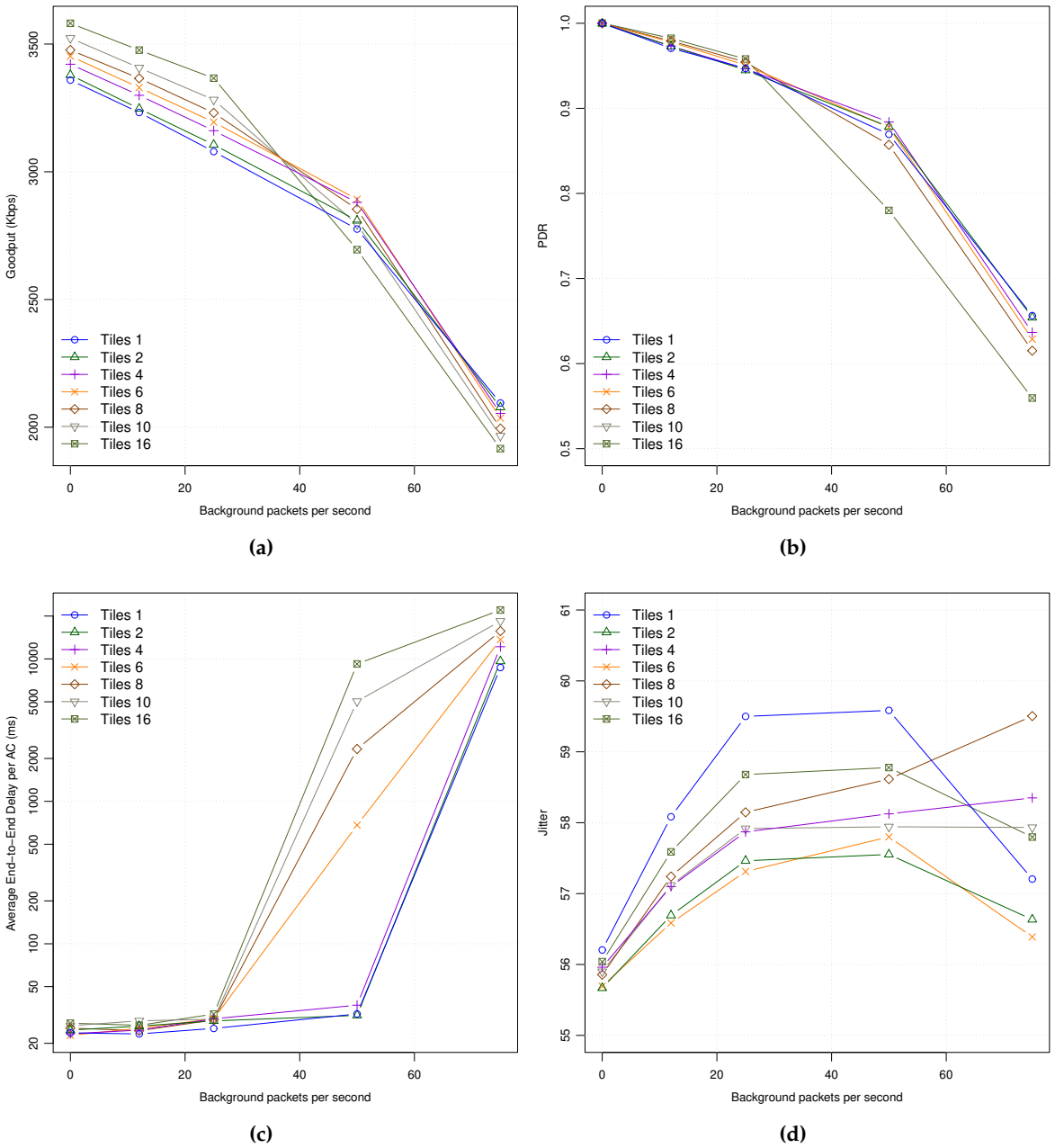

Figure 5. APP statistics for AI for all the tiles per frame layouts: (**a**) GP. (**b**) PDR. (**c**) E2E-Delay. (**d**) Jitter

behavior than the one explained before at lower background traffic levels, being a symptom of entering network saturation for the AI encoding mode.

Figure 5d shows the average jitter varying the number of tiles per frame. As it can be seen, the experienced jitter is always lower in 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

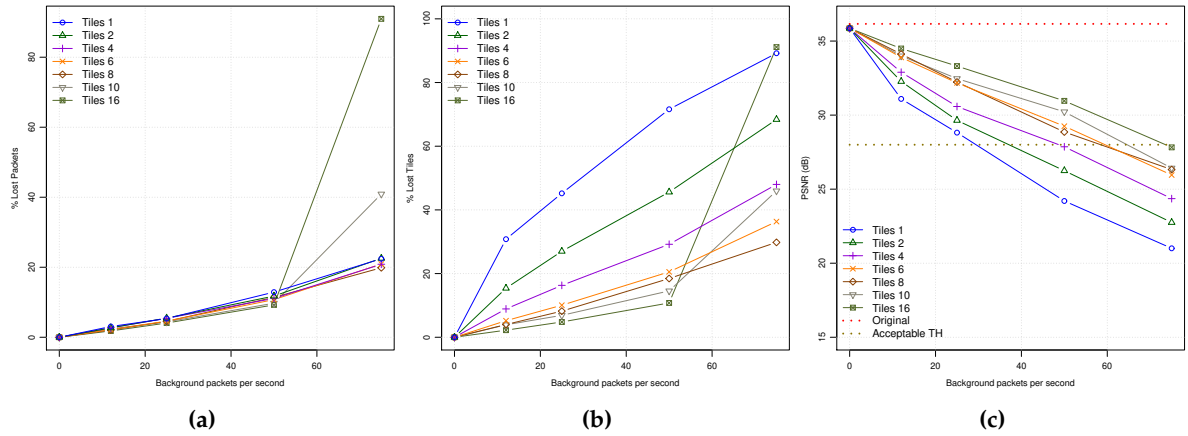

Figure 6. QoE statistics for AI for all the tiles per frame layouts: (**a**) PLR. (**b**) TLR. (**c**) PSNR

increases, obtaining acceptable video quality levels from low to moderate background traffic levels. Taking into account that as more tiles per frame more overhead in the bitstream, a number of tiles of 4 or 6 would be a good trade-off.

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 bitrate 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 increase 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. And finally, at high network loads (close to saturation) only the AI coding mode will be the one that stay over the quality threshold with 6 tiles. If the number of tiles were not an issue, the IPIP coding mode will be also selected using at least 8 tiles per frame.

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 bitstream size increase. So, a trade-off should be defined between video 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

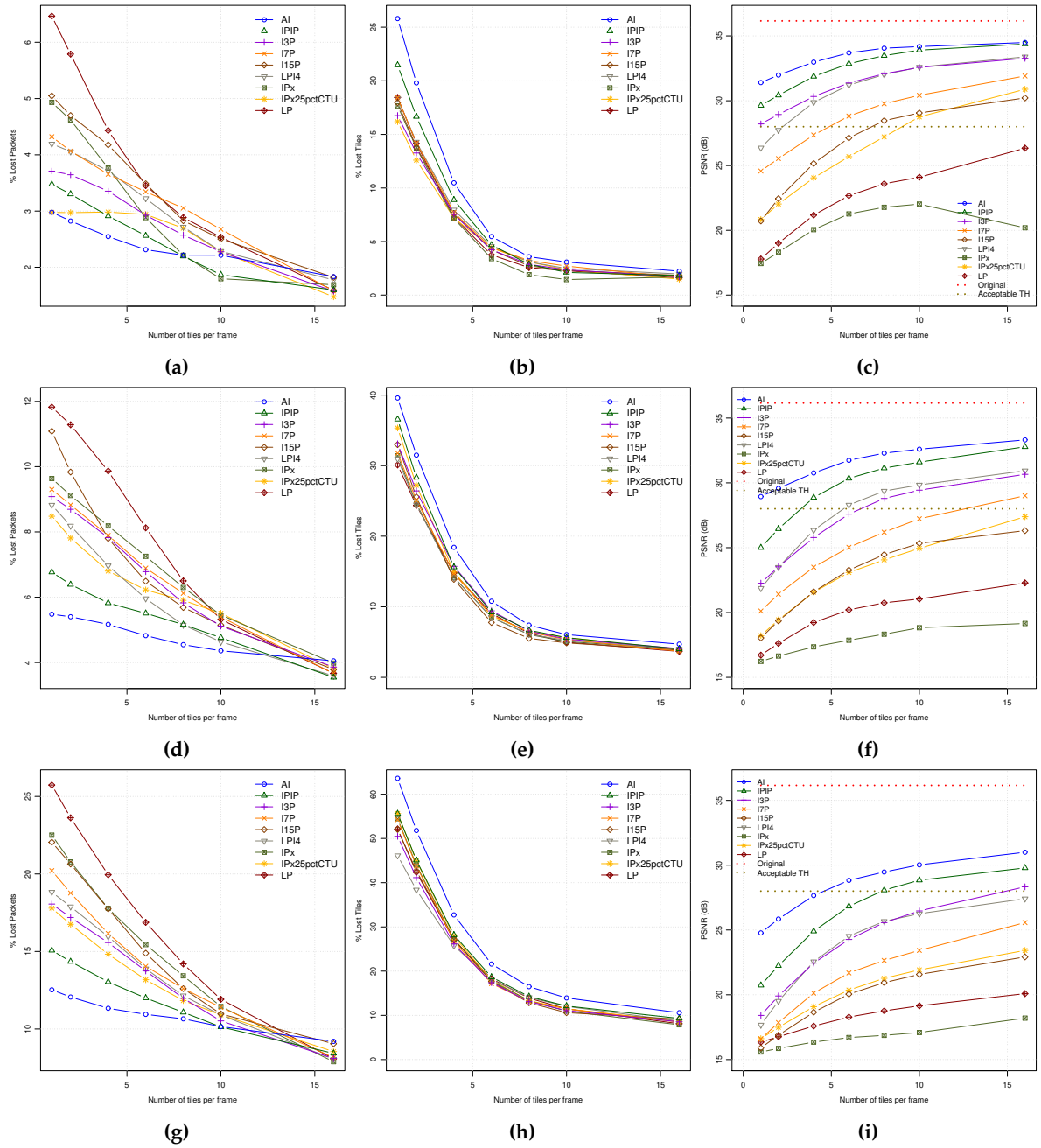

Figure 7. QoE statistics for all the encoding modes for all the tiles per frame layouts at different background traffic levels: (**a,b,c**) PLR, TLR, PSNR (12 pps). (**d,e,f**) PLR, TLR, PSNR (25 pps). (**g,h,i**) PLR, TLR, PSNR (50 pps)

more than 6 tiles (no significant improvements are found beyond), and the coding mode that, staying over the application quality threshold, requires the lowest bitrate.

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 bitrates with respect the others, so a trade-off between video resilience and final bitrate should to be defined, which strongly depends on the application requirements and the available network resources. With respect frame partitioning, we have observed that the use of tiles have 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 bitstream, being recommended not 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 higher 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), use of QoS at MAC level, and efficient error concealment, that combined with the ones proposed here minimize the introduced bitstream overhead and maximize the final video resilience. The final goal is focused to provide robust and efficient video streams that be 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 network QoS, and latter we will extend our error resilience architecture with packet-based FEC proposals, and advanced error concealment strategies.

Author Contributions: Validation and Supervision, M.P.M. and P.P.; Writing—Original Draft Preparation, P.P.G.A, M.P.M., P.P.; Writing—Review & Editing, P.P.G.A, M.P.M., P.P. and O.L.-G.; Funding Acquisition, 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.

1. Garrido Abenza, P.P.; Piñol Peral, P.; P. Malumbres, M.; López-Granado, O. Simulation Framework for Evaluating Video Delivery Services over Vehicular Networks. 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall), 2018, pp. 1–5.
2. Varga, A.; Hornig, R. An Overview of the OMNeT++ Simulation Environment. Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops, 2008, Simutools '08, pp. 60:1–60:10.
3. Sommer, C.; German, R.; Dressler, F. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing* **2011**, *10*, 3–15. doi:10.1109/TMC.2010.133.
4. Krajzewicz, D.; Erdmann, J.; Behrisch, M.; Bieker, L. Recent Development and Applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements* **2012**, *5*, 128–138.
5. Haklay, M.M.; Weber, P. OpenStreetMap: User-Generated Street Maps. *IEEE Pervasive Computing* **2008**, *7*, 12–18. doi:10.1109/MPRV.2008.80.
6. High Efficiency Video Coding (HEVC). ITU-T Recommendation H.265, 2013.
7. Joint Collaborative Team on Video Coding (JCT-VC). HEVC reference software HM (HEVC Test Model) and Common Test Conditions. https://hevc.hhi.fraunhofer.de. [Accessed May 11, 2018].
8. Vineeth, N.; Guruprasad, H.S. A Survey on the Techniques Enhancing Video Streaming in VANETs. *International Journal of Computer Networking Wireless and Mobile Communications* **2013**, *3*, 37–46.
9. Wang, Y.; Wenger, S.; Wen, J.; Katsaggelos, A.K. Error resilient video coding techniques. *IEEE Signal Processing Magazine* **2000**, *17*, 61–82. doi:10.1109/79.855913.

10. DeBrunner, V.; DeBrunner, L.; Wang, L.; Radhakrishnan, S. Error control and concealment for image transmission. *IEEE Communications Surveys Tutorials* **2000**, *3*, 2–9. doi:10.1109/COMST.2000.5340717.
11. Cui, Z.; Gan, Z.; Zhan, X.; Zhu, X. Error concealment techniques for video transmission over error-prone channels: a survey. *Journal of Computational Information Systems* **2012**, *8*, 8807–8818.
12. Kung, W.Y.; Kim, C.S.; Kuo, C.C.J. Spatial and Temporal Error Concealment Techniques for Video Transmission Over Noisy Channels. *IEEE Transactions on Circuits and Systems for Video Technology* **2006**, *16*, 789–803. doi:10.1109/TCSVT.2006.877391.
13. Stockhammer, T.; Hannuksela, M.M.; Wiegand, T. H.264/AVC in wireless environments. *IEEE Transactions on Circuits and Systems for Video Technology* **2003**, *13*, 657–673. doi:10.1109/TCSVT.2003.815167.
14. Bandyopadhyay, S.K.; Wu, Z.; Pandit, P.; Boyce, J.M. An Error Concealment Scheme for Entire Frame Losses for H.264/AVC. 2006 IEEE Sarnoff Symposium, 2006, pp. 1–4. doi:10.1109/SARNOF.2006.4534755.
15. Girod, B.; Farber, N. Feedback-based error control for mobile video transmission. *Proceedings of the IEEE* **1999**, *87*, 1707–1723. doi:10.1109/5.790632.
16. Kim, C.S.; Lee, S.U. Multiple description coding of motion fields for robust video transmission. *IEEE Transactions on Circuits and Systems for Video Technology* **2001**, *11*, 999–1010. doi:10.1109/76.946517.
17. Hagenauer, J.; Stockhammer, T. Channel coding and transmission aspects for wireless multimedia. *Proceedings of the IEEE* **1999**, *87*, 1764–1777. doi:10.1109/5.790636.
18. Wu, J.; Tan, R.; Wang, M. Streaming High-Definition Real-Time Video to Mobile Devices With Partially Reliable Transfer. *IEEE Transactions on Mobile Computing* **2018**, pp. 1–1. doi:10.1109/TMC.2018.2836914.
19. Minder, L.; Shokrollahi, A.; Watson, M.; Luby, M.; Stockhammer, T. RaptorQ Forward Error Correction Scheme for Object Delivery. RFC 6330, 2011. doi:10.17487/RFC6330.
20. 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. 4th Biennial Workshop on DSP for In-Vehicle Systems and Safety, Dallas, Texas, USA, 2009.
21. Zhang, R.; Regunathan, S.L.; Rose, K. Video coding with optimal inter/intra-mode switching for packet loss resilience. *IEEE Journal on Selected Areas in Communications* **2000**, *18*, 966–976. doi:10.1109/49.848250.
22. Calafate, C.T.; Malumbres, M.P.; Manzoni, P. Performance of H.264 compressed video streams over 802.11b based MANETs. Proceedings of the 24th International Conference on Distributed Computing Systems Workshops, 2004, 2004, pp. 776–781. doi:10.1109/ICDCSW.2004.1284121.
23. Chen, H.; Zhao, C.; Sun, M.T.; Drake, A. Adaptive Intra-refresh for Low-delay Error-resilient Video Coding. *Journal of Visual Communication and Image Representation* **2015**, *31*, 294–304. doi:10.1016/j.jvcir.2015.06.018.
24. Klaue, J.; Rathke, B.; Wolisz, A., EvalVid – A Framework for Video Transmission and Quality Evaluation; Springer Berlin Heidelberg: Berlin, Heidelberg, 2003; pp. 255–272. doi:10.1007/978-3-540-45232-4_16.
25. heng Ke, C.; kuen Shieh, C.; shyang Hwang, W.; Ziviani, A. An Evaluation Framework for More Realistic Simulations of MPEG Video Transmission. *Journal of Information Science and Engineering* **2008**, *24*, 425–440. cited By 174.
26. Saladino, D.; Paganelli, A.; Casoni, M. A tool for multimedia quality assessment in NS3: QoE Monitor. *Simulation Modelling Practice and Theory* **2013**, *32*, 30 – 41. doi:https://doi.org/10.1016/j.simpat.2012.11.011.
27. Rosário, D.; Zhao, Z.; Silva, C.; Cerqueira, E.; Braun, T. An OMNeT++ Framework to Evaluate Video Transmission in Mobile Wireless Multimedia Sensor Networks. Proceed. of the 6th International ICST Conference on Simulation Tools and Techniques, 2013, pp. 277–284.
28. Pediaditakis, D.; Tselishchev, Y.; Boulis, A. Performance and scalability evaluation of the Castalia wireless sensor network simulator. Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, 2010, p. 53.
29. 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 Transactions on Consumer Electronics* **2012**, *58*, 404–412. doi:10.1109/TCE.2012.6227440.
30. Garrido Abenza, P.P.; P. Malumbres, M.; Piñol Peral, P. GatcomSUMO: A Graphical Tool for VANET Simulations Using SUMO and OMNeT++. Proceedings of the SUMO User Conference 2017 (SUMO2017); , 2017; Vol. 31, pp. 113–133.
31. Misra, K.; Segall, A.; Horowitz, M.; Xu, S.; Fuldseth, A.; Zhou, M. An Overview of Tiles in HEVC. *IEEE Journal of Selected Topics in Signal Processing* **2013**, *7*, 969–977. doi:10.1109/JSTSP.2013.2271451.

32. Seeling, P.; Reisslein, M. Video Transport Evaluation With H.264 Video Traces. *IEEE Communications Surveys Tutorials* **2012**, *14*, 1142–1165. doi:10.1109/SURV.2011.082911.00067.
33. Wang, Y.; Sanchez, Y.; Schierl, T.; Wenger, S.; Hannuksela, M. RTP Payload Format for High Efficiency Video Coding. RFC 7798, 2016. doi:10.17487/RFC7798.