Evaluating HEVC Video Delivery in VANET Scenarios

P. Piñol*, A. Torres†, O. López*, M. Martínez* and M.P. Malumbres*
*Departamento de Física y Arquitectura de Computadores
Universidad Miguel Hernández de Elche
Email: {pablop, otoniel, mmrach, mels}@umh.es
†Department of Computer Engineering
Universidad Politécnica de Valencia
Email: atcortes@batousay.com

***Abstract*—Video delivery over VANETs is a difficult task. On the one hand, video streaming is a demanding application, where big deals of data need to be processed and transmitted. On the other hand, VANETs are error prone networks due to the high mobility of the nodes and the wireless channel. The sum of these two factors makes video transmission a hard to manage task. In this paper we evaluate the new emerging video coding standard HEVC and how it behaves under packet losses conditions in a VANET.**

***Keywords*—HEVC, VANET, packet loss, error concealment.**

I. Introduction

Video transmission over all kinds of networks is a field that has kept growing in the last years. Nowadays the improvements in network properties (increasing bandwidth in ADSL, 4G cellular networks, etc.) together with a larger efficiency of video codecs and an increase in processing ability and storing capacity of all kinds of devices (desktop computers, smartphones, tablet computers, ...) help making video streaming feasible. Lots of applications use video streaming, being video-sharing, advertising and video-conferencing three of the most widely spread.

Digital uncompressed videos contain large amounts of data. This is the reason why video codecs play an important role, as they are able to compress video sequences with large frame sizes and frame rates into relatively small bitstreams while keeping the quality of perception acceptable. The continuous improvements in the efficiency of video codecs are bound to an increase in their computational complexity. This complexity is partly alleviated by the growing computation ability of devices and by other techniques like parallel computing.

Communication networks is an area where data packet losses are encountered to a greater or a lesser extent. These losses are more pronounced in wireless networks and specifically even more considerable in VANETs (Vehicular Ad-Hoc Networks). VANETs are inhospitable environments for data transmission, even for basic data used for the management of the network itself. Wireless transmission and a high relative speed of nodes produce consequences like attenuation of signal caused by obstacles, Doppler Effect caused by relative speed, rapidly varying network topology caused by vehicles taking different routes, etc. So, video transmission over VANETs is a hard to manage task due to the combination of the high requirements of digital video and the hard conditions of the network. Nevertheless, video streaming can have multiple applications in VANETs, like context advertising, emergency video calls, tourist information, road conditions, control of traffic jams, ...

The mechanisms that reduce the number of lost packets at network level are useful and necessary. They are usually based on the introduction of certain levels of redundancy in the information which allows to retrieve some of the lost packets (or even all of them). But, in spite of these mechanisms, there are always some inevitable losses. In video streaming, data losses are not only produced by lost packets. Also, packets that arrive too late to be useful are discarded. So, techniques like traffic classification and priorization can also play an important role in data protection. In addition to the protection of packets at network level, data can be protected at application level. A simple example of this type of protection is buffering. Buffering allows enjoying watching a video without (excessive) interruptions in networks with an appreciable jitter.

In video streaming, as well as general techniques like buffering, we can use ad-hoc protection. When decoding a video stream which has some missing parts, not all the pieces of information have the same importance for the reconstruction of the video. The loss of certain packets will affect more than the loss of others in the final quality of the reconstructed video. For example, the loss of a frame used as a reference for other frames will bring a greater degradation of the reconstructed video than the loss of a frame which is not used as a reference by any other frame. This means that we can assign an "intelligent" protection to the packets in such a way that the more important a packet is, the larger the effort is made to protect it. Also, at encoding time, decisions can be made in order to make the coded sequence more or less resilient to errors and their propagation along the video sequence.

The aim of this work is to study how the new and emerging standard on video coding HEVC (High Efficiency Video Coding) [1] behaves when dealing with packet losses in VANET scenarios. It pretends to be useful for designing and implementing mechanisms of error resilience and concealment for this standard which will allow to palliate the damage that packet losses cause on the final quality of reconstructed video.

Different studies can be found about HEVC efficiency, usually comparing it with the previous standard H.264/AVC

(Advanced Video Coding), like [2] and [3]. One of the first works that studies HEVC under packet losses conditions is [4] where the authors developed a framework for evaluating HEVC encoded content under a range of packet loss, bandwidth restriction and network delay scenarios. On the other hand, several solutions have been proposed for improving video streaming over VANETs. Most of these works evaluate video quality taking only into consideration the percentage of lost packets [5]. In [6] and [7] the authors propose network solutions for the delivery of video over VANETs (by measuring video quality in an analytical way). Our work evaluates HEVC video transmission over VANETs in a more realistic way. First, VANET simulators are used to implement scenarios with real world maps. Then, real HEVC encoded video is streamed through these scenarios. At last, reconstructed videos are obtained and video quality with respect to the zero-loss video sequence is measured.

The rest of the paper is structured as follows. In Section II a brief description of HEVC is presented, highlighting the new features which are different from its predecesors. In Section III it is described the environment where the tests have been done. Section IV explains the results obtained for the different configurations that we have tested. At last, in Section V, several conclusions are drawn.

II. HEVC

In 2004 and 2007 ITU-T VCEG (Video Coding Experts Group) and ISO/IEC MPEG (Moving Pictures Experts Group), respectively, began the research on new techniques to improve the previous video coding standard, H.264/AVC. By 2010 the two organisms joined their forces, formed the JCT-VC (Joint Collaborative Team on Video Coding) and issued a Call for Proposals for the development of a new standard. The aim of the project was doubling the coding efficiency of H.264/AVC, this is, achieving a 50% reduction in bit rate at the same level of quality. On January 25, 2013 the new standard, HEVC, was agreed. Some works discuss if the real improvement of the new standard over the previous one has reached the proposed goal or not. What seems clear is that HEVC outperforms H.264/AVC and more noticeably at very high resolutions.

HEVC is very similar to H.264/AVC. They follow the same hybrid compression scheme. First, a frame is divided into small blocks and some estimation and compensation is done. Estimation searchs for similar pixels around the block in the same frame (*intrapicture*) or for a similar block in other (previously coded and decoded) frames (*interpicture*) and produces a "block prediction". Compensation substracts that "prediction" from the present block. Then the residuum is transformed into the frequency space. The coefficients obtained by this operation are then quantized in order to reduce the number of bits needed to represent them. The quantized coefficients are, finally, entropically coded to further compress the bitstream. The coded frame is then decoded and kept in a buffer so it can be taken as a reference for other frames. A good overview of the new emerging standard is provided by some of the members of JCT-VC in [8].

Although HEVC and H.264/AVC share the same compression scheme, there are some differences between them which allow HEVC outperforming H.264/AVC. In H.264/AVC the small units in which a frame is divided have a fixed size of 16x16 luma data samples (and its correspondant chroma samples) and are called MacroBlocks (MB). In HEVC these units are called Coding Tree Units (CTU) and can have a size of 16x16, 32x32 or 64x64 luma samples. When coding large homogeneous regions in a frame, large blocks behave much better than small blocks. The number of *intrapicture* prediction modes has been increased from 10 (H.264/AVC) to 25 (HEVC). These new modes allow to get a better prediction of the current coding unit and therefore residual samples are smaller and provide higher compression rates. A deblocking filter is used both in H.264/AVC and HEVC. After deblocking, and before using the reconstructed frame as a reference for motion estimation, a new process is introduced in HEVC: Sample Adaptive Offset (SAO) allows a better reconstruction of samples by classifying groups of them and applying a different offset for each group. This method increases the quality of the reconstructed frames. The entropy encoder for H.264/AVC, CABAC (Context Adaptive Binary Arithmetic Coder), has been upgraded and made more efficient in HEVC. Two new structures have been defined in HEVC that did not exist in H.264/AVC: tiles and WPP (Wavefront Parallel Processing). Both of them allow the introduction of parallel processing in coding and decoding frames. Tiles are rectangular regions of a picture that can be indepently decoded. WPP is a technique that divides a frame into rows of CTUs. Every row of CTUs can be indepently encoded (or decoded) after some pieces of information are gathered which are needed for prediction and adaptation of the entropy encoder (or decoder). There are more differences between both codecs. For further information about HEVC the reader can consult [8].

For our experiments we have used version HM-9.0 of the HEVC reference software [9]. When HEVC reference software encodes a video sequence, it produces a bitstream with a format that is appropriate to be stored in a file. We have modified reference software (encoder and decoder) to produce a bitstream which is separated into RTP (Real-time Transport Protocol) packets in order to transmit them through VANETs and then reconstruct the decoded video sequence. HEVC reference software decoder crashes when it tries to decode a bitstream with missing parts. So we have also modified the decoder to make it robust against packet losses. This robustness implies that missing packets will not make the decoder fail and it will continue decoding the rest of the received RTP packets. With these modified versions of the encoder and the decoder we can build the reconstructed version of the video and measure its quality.

III. Framework

A. Simulators

For the tests we have used two main blocks. On the one hand, an open-source vehicular traffic simulator: SUMO (Simulation of Urban MObility) [10]. This simulator models the behaviour of vehicles in urban scenarios, taking into consideration the interaction of vehicles with other vehicles, junctions, traffic lights, multilane roads, etc. On the other hand, we have used OMNeT++ [11] for simulating communications. OMNeT++ can be considered a framework for building specific network simulators. So we have used two of the projects developed for OMNeT++ named MiXiM (Mixed Simulator)

[12] and Veins (Vehicles in Network Simulations) [13]. MiXiM is a simulator that models fixed and mobile wireless networks. Veins adds specific IEEE protocols approved for their use in VANETs (IEEE 802.11p and IEEE 1609 WAVE) to MiXiM. One of the useful features of Veins is the modeling of the Control Channel (CCH) and the Service Channels (SCH) defined by IEEE 1609.4 - Multi-Channel Operation [14]. Another important feature is modeling obstacles which will attenuate wireless signals in a realistic way. The two blocks: SUMO and OMNeT++/MiXiM/Veins are connected by TraCI (TRAffic Control Interface). TraCI creates a TCP connection between SUMO (*TraCI-Server*) and OMNeT++ (*TraCI-Client*). OMNeT++ can send commands to SUMO to change the behaviour of vehicles (for example, turning the speed of a vehicle into 0 in order to simulate an accident). OMNeT++ reads periodically from SUMO the position of every vehicle in the simulation. The visualization of the experiments is carried out by OMNeT++ GUI (Graphical User Interface).

B. Add-ons

For the simulations we have developed a module that allows us to define servers (video transmitters) and clients (video receivers). A server announces a video service through the Control Channel periodically at 1 second intervals. These messages include information with the number of the Service Channel in which the transmission is taking place. The nodes of the network listen periodically to the CCH (this is one of the features of IEEE 1609.4) in order to get beacons from other vehicles, emergency messages and announcements of services. After the reception of an announcement of a video service, a node marked as a client, switches to the specified SCH, an begins capturing video packets. Client nodes save packet information and also create files with basic statistics, like the percentage of lost packets. From the file of received packets we will obtain the reconstruction of the decoded video in order to evaluate its quality. We have also introduced another kind of node: the background traffic source. This node inserts CBR (constant bit rate) traffic into the network, by injecting packets of a specified size at an indicated rate (packets per second).

C. Scenario and configurations

Some parameters are common for all the experiments and some others are changed in order to check the impact of them in the experiments.

1) Fixed parameters: The scenario in which vehicles move is an area of 2000m x 2000m located in the city of Kiev, the capital of Ukraine, obtained from the free geographic data base OpenStreetMap [15] and converted to SUMO format with the tools included in SUMO (*netconvert*, *polyconvert*). There is a large avenue of around 2000m that crosses the scenario from north to south. Three RSUs (Road Side Unit) are positioned in that avenue. The coverage radius of the wireless signal is 500m. The 3 RSUs are placed in the following way: one node is in the middle of the main avenue and the other two are at the beginning and the end of the avenue. These two nodes have a small zone in which their coverage overlaps with the middle node coverage. Figure 1 shows three big circles of the same color (*red*) which are the 3 RSUs (2 in the edges and 1 in the middle of the scenario). Every time a packet is sent by an RSU the circle changes its color. There is also a big

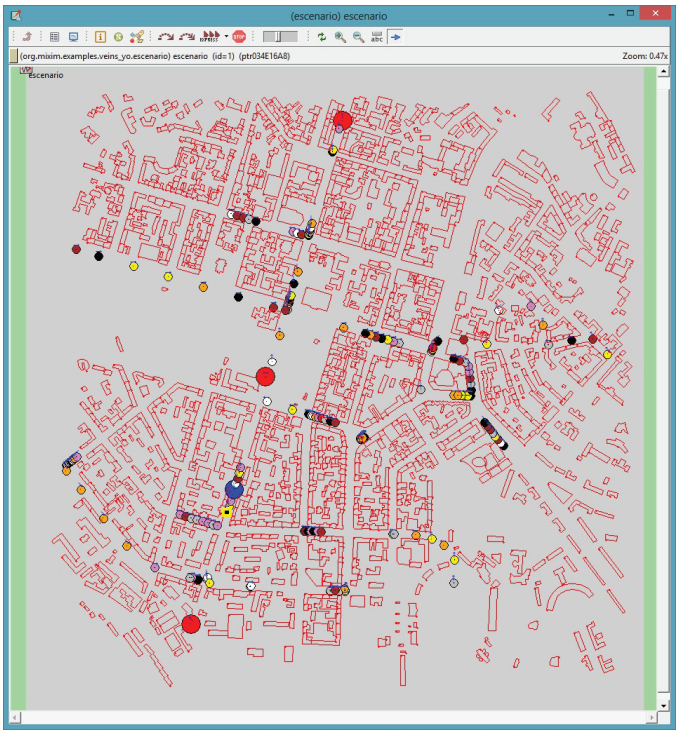

Fig. 1. VANET scenario in OMNeT++. *(red rectangles = buildings // big red circles = RSUs // big blue circles = video clients // small black&yellow squares = background traffic sources // small circles = other vehicles)*

circle of another color (*blue*) which helps us to identify cars receiving video. Every time a video packet is received its circle changes its color. These changes of color in RSUs and video clients helps us to see what is really happening. Background traffic nodes are represented by a small (*black&yellow*) square which also changes its color for every sent packet. At last, small circles represent other vehicles.

A car driven along this avenue is always reached by the signal of, at least, one of the RSUs. The maximum vehicle speed has been stablished to 14 m/s (50.4 km/h). Traffic lights are active. We have left the physical parameters of the network interface cards as they are configured in the Veins example [13]. Every data packet is sent to the network using broadcast.

The 3 RSUs broadcast the same video sequence in a synchronized way. They synchronize themselves by means of the infrastructure network. In this manner, when a car moves into the coverage area of an RSU and leaves the previous one, it can keep receiving the same video sequence. Typical applications of this could be contextual advertising or some kind of information about the city, like tourist information, local news, traffic jams, etc.

We have also launched the symmetric experiment, in which a vehicle transmits a coded video to the RSUs, which are connected to infrastructure networks, in order to send the video where it may be of any use. Such a transmission could be useful in the following scenario: a crashed car records a video including sequences of the condition of the passengers, an emergency video-call, or the state of the surroudings of the car (chemical spills, fire, other crashed vehicles, injured pedestrians, ...). Then it sends the video to another car that

passes near it. This car delivers the video to the RSUs, and the RSUs forward that video through the infrastructure network. This is sometimes refered in the literature as "data muling". This video could be used by emergency services in order to help the people in the crashed vehicle, to protect the area near the accident and also for example to reroute the traffic in the surroundings.

We have used the video sequence named "Race Horses" which is one of the sequences used in JCT-VC Common Conditions [16]. It has a frame size of 832x480 pixels, a frame rate of 30 frames per second and a total length of 300 frames. We have coded it with a Quantization Parameter (QP) value of 37. The coded video sequence is sent through the network in a cyclic way.

2) Variable parameters: Now we will enumerate the different parameters that we have varied to make comparisons between the experiments. Some experiments have been done with no background traffic and no beacons sent by the vehicles. These experiments try to find out the "practical" minimum of losses that would appear without any traffic in the network. These losses are due to collisions in the overlapping zones. The rest of the experiments have been done with some background traffic using different packet sizes at different packet rates.

As it has already been mentioned, the coded video sequence is sent in a cyclic way. It has been coded using Lowdelay-P (LP) configuration file from Common Conditions [16]. LP mode generates a bitstream where the first frame of the sequence is coded as an I frame and the rest of the frames are coded as P frames with the restriction that reference frames are always selected from previous frames (in display order). This mode is more efficient than All Intra (AI) mode (where all the frames are coded as I ones) because of temporal estimation and compensation. LP generates a smaller bitstream and so a smaller bitrate. This mode is faster than Random Access (RA) mode because it does not have to wait for future frames (in display order) to encode or decode the bitstream. RA is more efficient but slower. AI is faster but less efficient. We have chosen a compromise between bitrate and coding/decoding speed.

The first packets of a video sequence are essential for the decoder to do any decoding. If they are missing the decoding process simply cannot begin and no reconstructed video can be built. This is the reason why we assume that the first packets of a sequence are protected by some network mechanism and they cannot be missed.

We have coded the original sequence at 1, 2, 4, 8, 13 and 26 slices per frame. Every slice will travel in a different RTP packet. If we divide a frame in several slices the loss of one RTP packet will not imply the loss of the complete frame but only part of it (because slices can be independently decoded). As slices are independent, compression is not as efficient at 26 slices per frame as at 1 slice per frame. The frame rate of the sequence is 30 frames per second so we will send 30, 60, 120, 240, 390 an 780 packets per second for each number of slices per frame, to keep the video frame rate. The more slices per frame used the smaller the packets will be. This variability in the size and number of packets per second will lead to diverse results when competing with other applications for the SCH (here represented by the injection of background traffic).

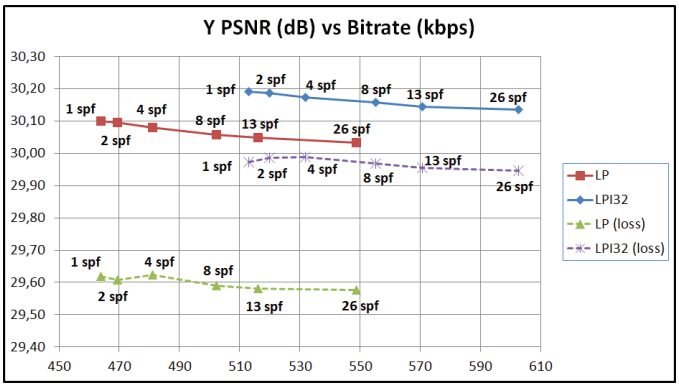

Fig. 2. Rate-Distortion curves for *Lowdelay-P (LP)* and *Lowdelay-P-I32-CDR (LPI32)* modes, without any losses and with simple losses (without background traffic)

The first experiments were done by using pure LP mode. This mode only introduces an I frame at the beginning of the sequence, so when packet losses appear, the error in one slice "infects" the frames that use this one as reference and there is a great drift. For alleviating this effect we have modified LP configuration file and created "Lowdelay-P-I32-CDR" (LPI32) mode. By "*I32-CDR*" we mean that the encoder inserts an I CDR frame every 32 frames. In this way an error does not propagate further than 32 frames away (approximately one second of video).

IV. Results

A. Rate-Distortion in absence of losses

In Figure 2 we can see the rate-distortion curves for the video sequence coded in LP and LPI32 modes without any losses (solid curves). Bitrates for LP mode ranges from 463.83 kbps (when we code it with 1 slice per frame) to 548.81 kbps (when we code it with 26 slices per frame) with only 0.05 dB of difference in PSNR (Peak Signal-to-Noise Rate). This means a 18.3% of increase in the bitrate. Such an increase is due mainly to the overhead introduced by dividing a frame into so many slices, and is also due to the fact that coding small parts independently is less efficient than coding the frame as a whole. We have also found that adding an I CDR frame every 32 frames (in LPI32) increases the bitstream around a 10% over the pure version of LP and the PSNR value increases around 0.1 dB.

B. Rate-Distortion with simple losses

The first experiment consists in measuring packet losses in the absence of background traffic. The client vehicle travels from the beginning of the avenue until the end, and receives video packets from each of the 3 RSUs when it is in their zone of coverage. Packet losses occur only in the areas where the signals of two RSUs are overlapped. Percentage of losses is around 0.57% for every combination of slices per frame. In Figure 2 the dashed curves show the PSNR values of the video sequence for this "simple" losses. It can be clearly seen that LP reduces its quality more than LPI32. LPI32 reduces its quality around 0.20 dB and LP around 0.47 dB. This is due to the fact that when isolated losses occur, LPI32 recovers very quickly from errors and LP does not.

C. Percentage of packet losses with background traffic

In Table I we can see the percentage of lost packets at different background traffic rates (packets per second, packet sizes) for each of the numbers of slices per frame. First we can see that data losses are more influenced by the packet rate than by the packet size. At 30 pps of 512 bytes (122.88 kbps) and at 30 pps of 1024 bytes (245.76 kbps) the percentages of lost packets are very similar. A similar behaviour can be seen for 90 pps: doubling the size of the packet (and therefore the data rate) does not result in a significant packet loss increase. At 780 pps, nearly 1 of every 5 packets gets lost. This rate of losses is hard to deal with, and especially if we are dealing with video streaming. This is what would happen if two different video streams at 26 slices per frame would compete in the same SCH. Secondly, we can see that for 30 pps and 90 pps traffic, when the number of slices per frame increases, the percentage of lost packets decreases. This is because, at high video packet rates, for example at 26 slices per frame, a high number of video packets are not competing with background traffic. This may seem a kind of solution (to increase the number of slices per frame) but this may make the problem worsen due to channel saturation, like stated before. So, it seems that keeping the number of slices per frame at an intermediate level may be a good compromise.

TABLE I. PERCENTAGE OF PACKET LOSSES AT DIFFERENT BACKGROUND TRAFFIC. (PPS=PACKETS PER SECOND / B=BYTES / SPF=SLICES PER FRAME)

% of losses	1spf	2spf	4spf	8spf	13spf	26spf
30pps/512B	6,9	4,9	2,7	1,6	1,3	0,9
30pps/1024B	7,4	4,6	2,8	1,6	1,3	1,0
90pps/512B	12,2	13,2	9,5	5,4	3,6	2,2
90pps/1024B	12,9	13,4	9,9	5,5	3,8	2,4
780pps/512B	20,1	17,9	18,1	18,6	19,3	20,5
780pps/1024B	19,5	20,7	21,3	21,6	21,9	34,5

D. PSNR with background traffic

In Figure 3 the rate-distortion curves for the tests of the previous section have been drawn. As it was clear for the percentage of lost packets, video quality values also indicate that packet rate is more important than packet size when trying to guarantee a certain video quality. Video sequences which lie under 25 or 26 dB are not suitable for watching. So, by combining data from Table I and Figure 3 we can state that percentages of packet losses over 1.3% lead to decoded sequences in bad conditions for their use. HEVC is very sensitive to packet losses, even when introducing an I frame every second.

E. LP vs LPI32

At the first experiments we could state that pure LP mode would not resist even the minimum packet loss. This becomes noticeable in the first tests with "simple" losses. That was the reason for using LPI32 mode for our experiments. Now, we can see by means of Figure 4 that, when background traffic is delivered, this situation gets even worse. With a background traffic of 30 pps and 1024 bytes/packet, LPI32 suffers a drop of 2.76 dB at 26 slices per frame, and a drop of 8.82 dB at 1 slice per frame. In the same conditions LP suffers a drop of 5.97 dB at 26 slices per frame and 14.95 dB at 1 slice per frame.

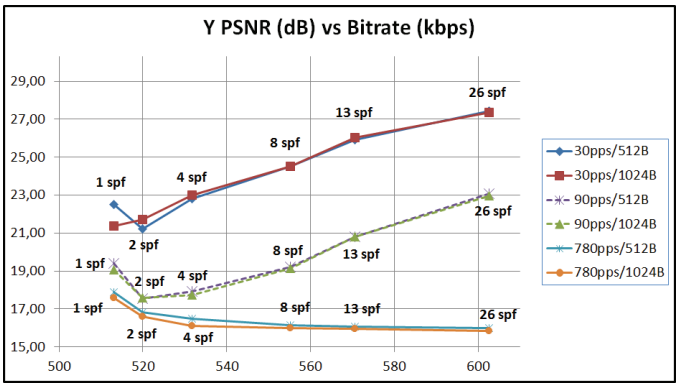

Fig. 3. Rate-Distortion curves for *Lowdelay-P-I32-CDR (LPI32)* mode, with background traffic.

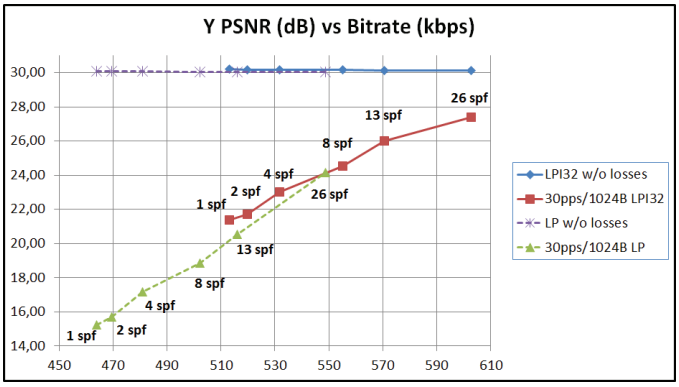

Fig. 4. Rate-Distortion curves for *Lowdelay-P (LP)* and *Lowdelay-P-I32-CDR (LPI32)* modes, with background traffic.

This indicates that intra refresh improves error resilience of coded videos in error prone networks (although it may not be enough).

F. Simple Error Concealment Method

In order to improve video quality results we have modified the reference software, implementing a simple error concealment method. If one slice is missing for decoding then the decoder copies the corresponding slice of the previously decoded frame. So, instead of a gray area for the missing slice, we will get an approximation to the lost data. Although PSNR values do not increase much with this method, it can be seen from figures 5, 6 and 7 that subjective evaluation shows an improvement. This is clearly noticed when playing the sequence. This method is not of much help when severe losses occur. However, for small percentages of losses, it smooths the errors and conceals some annoying artifacts.

G. The symmetric experiment

As we have mentioned before, we have also performed the symmetric experiment. Here, the vehicle moving along the avenue sends a video sequence and the RSUs act as video clients. The first and simple experiment, with no background traffic, produces a 0% of packets lost. In this case, when the vehicle is in one of the overlapped areas, the RSUs are merely listening, so there are no packet losses. Also, since the RSUs are located so that they cover the whole avenue, they get all

Fig. 5. Subjective evaluation. Encoded/decoded frame number 276 without any losses

Fig. 6. Subjective evaluation. Reconstruction of frame 276 under packet losses without activating the simple error concealment method

Fig. 7. Subjective evaluation. Reconstruction of frame 276 under packet losses with the simple error concealment method active

the sent packets. When the vehicle is in an overlapped area, then 2 RSUs receive the same packets. This does not cause any problem because the application checks this, simply discarding duplicated packets. When adding background traffic the results are very similar of those shown in Figure 3. This experiment has revealed that a single vehicle sending video to the RSUs does not create any new problem; instead, it is a softer one.

V. Conclusions

From the experiments and measurements we have observed that HEVC is very sensitive to packet losses. Varying the number of slices per frame can lead to an improvement or a worsening of the situation, so an adaptive mechanism that measures the SCH saturation and changes the number of slices per frame can help with the percentage of packet losses. Also, we have observed that Intra refreshing improves the quality of the reconstructed video, but it is not enough to guarantee a good video experience. A simple mechanism of error concealment has been implemented in the reference software, and subjectively proved to give some relief to a damaged video. A combination of both protection of packets at network level, and error resilience techniques at application level is needed to guarantee the minimum video quality required.

Acknowledgment

This work was supported by *Spanish Ministerio de Ciencia e Innovación* under project TIN2011-27543-C03-03 and *Consellería de Educación, Formación y Empleo de la Generalitat Valenciana* under project ACOMP/2013/003.

References

[1] B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, Y.-K. Wang, and T. Wiegand, "High Efficiency Video Coding (HEVC) text specification draft 10," Joint Collaborative Team on Video Coding (JCT-VC), Geneva (Switzerland), Tech. Rep. JCTVC-L1003, January 2013.

[2] J.-R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the Coding Efficiency of Video Coding Standards - Including High Efficiency Video Coding (HEVC)." *IEEE Trans. Circuits Syst. Video Techn.*, vol. 22, no. 12, pp. 1669–1684, 2012.

[3] R. Garcia and H. Kalva, "Subjective evaluation of HEVC in mobile devices," in *Proceedings of SPIE*, ser. Multimedia Content and Mobile Devices, vol. 8667, 2013, pp. 86 670L–86 670L–9.

[4] J. Nightingale, Q. Wang, and C. Grecos, "HEVStream: a framework for streaming and evaluation of high efficiency video coding (HEVC) content in loss-prone networks," *IEEE Trans. Consumer Electronics*, vol. 58, no. 2, pp. 404–412, 2012.

[5] J.-S. Park, U. Lee, and M. Gerla, "Vehicular communications: emergency video streams and network coding," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 57–68, 2010.

[6] M. Asefi, J. W. Mark, and X. Shen, "A Cross-Layer Path Selection Scheme for Video Streaming over Vehicular Ad-Hoc Networks," in *Vehicular Technology Conference Fall (VTC 2010-Fall), 2010 IEEE 72nd*, 2010, pp. 1–5.

[7] M. Asefi, S. Cespedes, X. Shen, and J. W. Mark, "A Seamless Quality-Driven Multi-Hop Data Delivery Scheme for Video Streaming in Urban VANET Scenarios," in *Communications (ICC), 2011 IEEE International Conference on*, 2011, pp. 1–5.

[8] G. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1649–1668, 2012.

[9] HM Reference Software vers. 9.0, https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-9.0.

[10] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, December 2012.

[11] OMNeT++, http://www.omnetpp.org.

[12] MiXiM, http://mixim.sourceforge.net.

[13] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," *IEEE Transactions on Mobile Computing*, vol. 10, no. 1, pp. 3–15, January 2011.

[14] "IEEE Standard for Wireless Access in Vehicular Environments (WAVE)–Multi-channel Operation," *IEEE Std 1609.4-2010 (Revision of IEEE Std 1609.4-2006)*, pp. 1–89, 2011.

[15] OpenStreetMap, http://wwww.openstreetmap.org.

[16] F. Bossen, "Common test conditions and software reference configurations," Joint Collaborative Team on Video Coding (JCT-VC), Geneva (Switzerland), Tech. Rep. JCTVC-L1100, January 2013.