Universidad Miguel Hernández de Elche

Departamento de Ingeniería de Sistemas y Automática

Robust Video Streaming over Vehicular Networks

Ph.D. Thesis

A dissertation for the degree of Doctor of Philosophy by
Pablo José Piñol Peral

Advisors
Manuel José Pérez Malumbres
Otoniel Mario López Granado

Acknowledgements

"Lo importante es quererse..."
Isabel Cuevas Rubio

Dedicated to my family. I love you more every single day of my life.

I want to thank my thesis advisors for their guidance in this work and for their friendship.

I want to thank my colleagues (and ex-colleagues) from the *Universidad Miguel Hernández de Elche* for their contribution to this work, their good company all these years, and for their friendship.

I want to thank my colleagues from the *Universidad Politécnica de Valencia* and the *Universidad de Zaragoza* for their assistance in different stages of the development of this work and for their friendship.

I want to thank my family for all the support that they have provided me, and for the sacrifices that we all made to make this thesis come true.

Zanx.

Abstract

In the near future, *vehicular networks* will be as common as *smartphones* are nowadays. Vehicles will be equipped with a variety of sensors; they will possess processing and storage capabilities, and will be able to communicate with each other and with infrastructure. These three characteristics will make Intelligent Transportation Systems possible. As defined in Directive 2010/40/EU of the European Parliament, "Intelligent Transport Systems (ITS) are advanced applications which, without embodying intelligence as such, aim to provide innovative services relating to different modes of transport and traffic management and enable various users to be better informed and make safer, more coordinated and smarter use of transport networks. ITS integrate telecommunications, electronics and information technologies with transport engineering in order to plan, design, operate, maintain and manage transport systems. The application of information and communication technologies to the road transport sector and its interfaces with other modes of transport will make a significant contribution to improving environmental performance, efficiency, including energy efficiency, safety and security of road transport, including the transport of dangerous goods, public security and passenger and freight mobility, whilst at the same time ensuring the functioning of the internal market as well as increased levels of competitiveness and employment."

One of the technologies that may have multiple beneficial uses in ITS applications is *video streaming*. Video streaming can be linked to applications that range from digital entertainment (such as video on demand) to road safety (such as emergency video calls), or others related with business applications (such as contextual advertising or tourist information). Although video streaming over vehicular networks may be very useful, it comes with two big challenges. On the one hand, because of its wireless nature and because of the mobility of its nodes, vehicular networks are packet loss prone environments. On the other hand, digital video has huge requirements regarding resources, both because of the high volume of data with which it deals, and because of the necessity of computing power in order to compress video sequences. This need for compressing video data makes encoded video bit streams vulnerable to packet loss because the video encoding process makes use of redundancy to a certain level, and the loss of some pieces of data may have an adverse effect in large regions of the video sequence.

In order to provide video streaming with protection against data loss, we have addressed this issue with the combination of two different approaches. The first improves bit stream error resilience by acting on a source coding level, i.e., adapting and adjusting video encoding in such a way that the compressed bit stream can obtain a better quality level in the presence of data loss. As this approach does not obtain optimal results, a second mechanism has been used in combination with the first one, i.e., the use of Forward Error Correction (FEC) techniques to minimize the effect of packet loss. For the first of the two approaches, we have selected the High Efficiency Video Coding (HEVC) standard, and, for the second of the approaches, we have selected RaptorQ codes technology. HEVC is the most recent video coding standard developed by the Joint Collaborative Team on Video Coding (JCT-VC). To add robustness to video streams on a source coding level, we have combined three strategies: (a) we have implemented an error concealment mechanism at the decoder that alleviates the effects of missing frames or parts of a frame; (b) we have proposed and studied the division of each frame into a different number of tiles (which is a brand new feature of the HEVC video encoder) in order to stop the multiplication factor of packet loss when every frame is encoded as a whole; and (c) we have proposed several encoding modes that increase the intrinsic error resilience of encoded video bit streams. Regarding the FEC approach, fine tuning of RaptorQ technology has been performed to adjust it to better fulfill the specific video streaming requirements in vehicular networks. Parameters like the protection period, the repair symbol size, and the amount of protection have been evaluated, and the best combination of these parameters has been selected to achieve the maximum recovery features while keeping the protected bit stream values within network constraints. To check the different proposals, a number of tests have been performed with the well-known vehicular traffic simulator SUMO and a combination of network simulators (OMNeT++/MiXim/Veins) using realistic urban scenarios under different network conditions. Results show that selecting the appropriate configuration improves video streaming to keep the reconstructed video quality within suitable bounds.

Preface

Motivation

The research topic in which the present work is focused is video streaming over vehicular networks.

Due to the big deal of data that digital video generates, it is essential to compress video sequences in order to store and/or transmit them. The compression procedure increases data vulnerability because of the interrelationships created by the encoding process. These dependencies imply that the loss of a small amount of information may entail the effective loss of even the whole contents in a sequence. In order to stream video data over vehicular networks (which are error-prone environments because of mobility and wireless communications), the inclusion of protection mechanisms becomes indispensable in order to improve video error resilience to provide a suitable quality of experience to users. The work in this thesis proposes and analyzes various alternatives to enhance the robustness of video streaming over vehicular networks

Objectives

The main objective of this work is to study the video streaming over vehicular networks, and propose and evaluate mechanisms that improve its robustness and guarantee the quality of experience to users.

This global objective is composed of the following specific objectives:

- Propose video protection mechanisms based on source coding
- Propose video protection mechanisms based on channel coding
- Evaluate these methods in a realistic vehicular network scenario

Thesis organization

The thesis is organized as follows: Chapter 1 presents the fundamentals of the topics related with this work. In Section 1.1, Intelligent Transportation Systems and vehicular networks are introduced, together with the most relevant communication protocols for these networks. In Section 1.2, the hybrid video coding scheme is explained and the main features of the High Efficiency Video Coding standard are detailed. Section 1.3 presents RaptorQ codes, the Forward Error Correction technique used in this work for the protection of network packets. Section 1.4 cites some related work and the main differences with the present work. In Chapter 2, the proposals to protect the encoded video bit streams are specified. The alternatives selected for source coding protection are developed and tested in Section 2.1, and the corresponding FEC system is analyzed and fine tuned to adapt it to the specific peculiarities of video protection in Section 2.2. In Chapter 3, the evaluation of the performance of the protection proposals in a realistic vehicular scenario is accomplished. In Section 3.1, the framework for the tests is presented, together with the set of simulators used in the experiments. In Section 3.2, the results yielded are shown and the analysis of the different measurements obtained is carried out. Chapter 4 summarizes the contributions of this work and outlines future work proposed to extend the research performed in this thesis. Also, in that chapter, the published works originating from the development of this thesis are cited.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 Intelligent Transportation Systems

Our lives have undergone a radical change since cell phones are no longer just telephones, and have become *smartphones*, with good processing capabilities, large storage capacity, and above all, great connectivity. This connectivity allows us to have all kinds of information available at all times. And we are not only information consumers but active producers as well. In the near future, vehicles will be equipped with lots of sensors (nowadays they already are) that will be able to take internal and external measurements. Vehicles will also be provided with a certain computing capability and storage capacity, which will allow them to process information, and they will also carry communications equipment. These three elements (sensors + computing ability + connectivity) will make Intelligent Transport Systems (ITS) possible, where vehicles will have the ability to communicate with each other and with infrastructure in an intelligent way.

There are lots of applications envisioned for ITS. In [1], we can find a classification of ITS services in four categories: (a) active safety; (b) public service; (c) improved driving; and (d) business and entertainment. Active safety is obviously the core part of the majority of the proposals but, recently, other aspects like driving efficiency are gaining growing attention. In the business world, a lot of promising applications arise. Not only can transportation companies obtain big savings or optimize resources, but also new areas of business will probably appear for other types of companies. In relation to ITS *infotainment* applications, if we simply glance at the existing plethora of applications for smartphones and digital tablets, we can figure out that applications related with information and/or entertainment will probably expand in an exponential manner.

Active road safety applications are mainly oriented to avoid accidents involving vehicles, and principally focused on preserving passengers' and pedestrians' lives. These applications try to prevent accidents by warning drivers about risks. And even further, if an accident is unavoidable, ITS applications can take the appropriate measures to prepare the vehicle in order to minimize the resulting damages. Post-accident applications can help to protect other vehicles by informing them that they are approaching an accident site, and can also send alerts with critical information about the accident and the condition of the injured, which can help the rescue party or the medical services provide better assistance.

Public services like police, fire departments, or patient transportation can benefit from ITS applications. Electronic license plates or electronic driver licenses can ease traffic control and surveillance by police officers. Virtual sirens can alert surrounding vehicles that emergency transport is approaching, even if the flashing lights and audible siren are not perceptible because of the distance. Traffic signal preemption for emergency ambulances can provide them with the right of way and reduce the time needed to arrive at the accident site and to transport the injured to a hospital.

Applications within the *improved driving* category aim at objectives like route optimization (avoiding congested areas), assisted merging into flowing traffic (when entering highways), parking spot locator services, and speed management. Speed management can help the driver adjust the vehicle speed both for obeying speed limits and for adapting its velocity to the road traffic conditions in order to achieve smooth driving and avoid unnecessary stopping. The "green-light wave" speed optimization also falls into this category. These applications can use information obtained directly from surrounding vehicles and also data aggregation obtained from distant sources.

Business and entertainment services cover a broad spectrum of applications. Transportation companies can optimize management of their fleets. Garages can perform wireless diagnostics of vehicles and vehicles can make an appointment with a garage if a breakdown is detected. Automatic payment in places like parking lots, gas stations, and highways can ease the use of these services. Contextual advertising will take commercials to a new and more efficient level. As stated before, the number of applications related to information consumption and entertainment will probably experience unpredictable and exponential growth.

In Table 1.1, we can see a few illustrative examples for the four categories mentioned before. Will ITS have a "killer app", which will assist the decisive establishment of intelligent vehicles?

ITS applications for active safety and for the remaining goals will bring transportation to a new and promising scenario, but whereas a world of opportunities will appear, ITS will also have to face a certain number of challenges. These challenges will not only have to deal with intrinsic network problems, like a changing topology, but with other issues like privacy or legal liability of the vehicle drivers as well. In Table 1.2, some of the main ITS challenges are listed. These challenges have become open problems in the development and deployment of ITS in which experts are carrying out their research.

From its origins, plans for the development of ITS have been promoted by state governments and also supranational administrations. Industry has played a very important role in ITS research and academia has also showed growing interest in ITS-related lines of research. Amongst public organisms that have

Table 1.1. Examples of ITS applications.

Category	Application examples
Active safety	Curve speed warning. Low bridge warning. Work zone warning. Visibility enhancer. Pre-crash sensing. Post-crash warning. Intersection collision warning. SOS service. Road condition warning. Crossing pedestrians. Lane change warning. Blind spot warning. Rail collision warning. Emergency electronic brake lights. Forward/Rear collision warning.
Public service	Approaching emergency vehicle warning. Emergency vehicle signal preemption. Emergency vehicle at scene warning. Stolen service vehicles tracking. Electronic license plate. Electronic driver's license. Vehicle safety inspection.
Improved driving	Left turn assistant. Highway merge assistant. Enhanced route guidance and navigation. Map download/update. Parking spot locator service. Intelligent traffic flow control. Cooperative glare reduction. Cooperative adaptive cruise control.
Business and entertainment	Toll collection. Parking payment. Gas payment. Wireless vehicle diagnostics. Rental car processing. Hazardous material cargo tracking. Fleet management. Instant messaging. Internet service provisioning. Point-of-interest notification.

promoted the development of ITS services and the creation of standards, we must cite the U.S. Department of Transportation (DOT) [2]; some public institutions in Japan such as the Ministry of Internal Affairs and Communications (MIC) [3], the Ministry of Land Infrastructure and Transport (MLIT) [4], and the National Institute for Land and Infrastructure Management (NILIM) [5]; and in Europe, besides the individual efforts of several countries, we must cite the European Commission which, by means of the Framework Programs, has supported many ITS initiatives. Also, a number of consortia and partnerships (some of them private, some of them mixed public/private) have been born that promote both projects and standardization proposals. Some of them are the Vehicle Safety Communications (VSC) Consortium [6][7], ITS Info-communications Forum - Japan [8], Intelligent Transportation Systems and Services for Europe (ERTICO) - ITS Europe [9], Car 2 Car Communications Consortium (C2C-CC) [10], and the European Council for Automotive R&D (EUCAR) [11]. Compiling an exhaustive list of ITS projects (and their goals) carried out by the cited organizations (and others) is beyond the scope of this thesis, but some of them are listed in Table 1.3 for illustrative purposes.

The communication networks which define the connectivity axis of ITS services are called *vehicular networks*.

Table 1.2. ITS challenges.

Challenge	Description
Geographical addressing	For some applications, every network address has to be associated to the geographical position of the vehicle.
Risk analysis and management	The network formed by vehicles and infrastructure is susceptible to security attacks. These threats have to be addressed.
Data-centric trust and verification	As the information travelling in the network may be critical, integrity, confidentiality, and authentication of the data need to be guaranteed.
Anonymity, privacy, and liability	Drivers' anonymity and privacy are basic rights in some countries. Maintaining anonymity and privacy while providing means to require legal liability of drivers is not an easy to solve issue.
Delay constraints	Some applications cannot tolerate large delay values, because information is only useful if it is delivered in a timely manner.
Prioritization of data packet and congestion control	Emergency data needs to be delivered in a prioritized way. Emergency messages have to be "insured" against low connectivity but also against broadcast storms to guarantee their delivery.
Reliability and cross-layering between transport and network layers	As network connectivity is not a constant feature, both transport and network layers need to cooperate in order to increase reliability in information transportation.

1.1.1 Vehicular networks

Vehicular networks are composed of mobile nodes (vehicles) and static nodes (infrastructure). Communication between two mobile nodes is referred to as V2V (vehicle-to-vehicle) communication, while communication between a mobile node and a static node is referred to as V2I (vehicle-to-infrastructure) communication. Within vehicular networks, V2V and V2I are, obviously, wireless communications, but static nodes can also make use of wired networks, for communicating with each other or for acting as a gateway of the vehicular network towards the Internet. On the infrastructure side of vehicular networks, we can use both already deployed wireless networks, which implement well-known technologies, or new infrastructure that make use of the latest communication standards that have been specifically developed for their use in these networks.

Two of the basic characteristics of vehicular networks are, as stated before, (a) the wireless communication channel and (b) the node mobility. Usually

Table 1.3. ITS projects in the USA, Japan, and Europe.

Project	Region/Period/Goals
Vehicle Safety Communications (VSC)	USA (2002-2004). Development of traffic safety applications: (1) cooperative forward collision warning, (2) curve speed warning, (3) precrash sensing, (4) traffic signal violation warning, (5) lane-change warning, (6) emergency electronic brake light, (7) left turn assistant. [6]
IntelliDrive	USA (2004-2009). Verify and enhance WAVE/IEEE 1609 features. Enable secure wireless communication among vehicles and between vehicles and roadway infrastructure. [12]
V2V Communications for Safety	USA (2009-*present*). Facilitate and help the deployment of the V2V communication based safety systems that should enhance safety throughout the vehicle fleet within the USA. [13]
Electronic Toll Collection (ETC)	Japan (1993-*present*). Development of a common ETC system capable of both prepay and postpay systems, confirmable usage records, which are written into IC (Integrated Circuits) cards. System must be available for all vehicles, using V2I communication throughout Japan. [14]
Advanced Safety Vehicle (ASV)	Japan (1991-*present*). Develop methods and devices to improve the safety of the transportation system, such as emergency braking, parking aid, blind curve accidents, right turn assistance and pedestrian accidents, blind intersection, and image of cognitive assistance. [15]
ITS-Safety 2010	Japan (2006-2011). Focus on ITS safety and the use of the V2V communications system and the V2I communications system. Use millimeter wave radar system to sense the distance between vehicles and a vehicle and obstacles. [16]
Communications for eSafety (COMeSafety)	Europe (2006-2010). FP6. Coordination and consolidation of the research results obtained in a number of European projects and organizations and their implementation. Frequency allocation of the spectrum for ITS applications. [17]
Cooperative Vehicle-Infrastructure Systems (CVIS)	Europe (2006-2010). FP6 project that designed, developed, and tested technologies needed for vehicles to communicate with each other and with the nearby road infrastructure. [17]
GeoNet	Europe (2008-2012). FP7 project that develops geographic addressing and routing (geonetworking) solutions using reliable and scalable communication capabilities, which enable the exchange of information in a particular geographic area, usually located far away from the source of information. [18]

they are also characterized by (c) the high relative speed of mobile nodes. These three characteristics by themselves, and also their combination, may come attached with some drawbacks. First of all, wireless networks are, by nature, packet loss prone networks. This may be one of the most important drawbacks that we will have to face. Wireless signals can be blocked by physical obstacles (buildings, other vehicles, etc.). The emission of different signals can cause interference. Multipath fading can appear because mobile nodes are moving while they are transmitting or receiving data. Also, the motion of nodes can cause artifacts like the Doppler effect. Because of reasons like the time of the day, or the particular area of a city, we may have very dense networks (for instance, in a city at rush hour) and also sparse networks (in the same city but at nighttime). On the one hand, a sparse network may have low connectivity in which nodes may become isolated/unreachable. Applications where delay can be accepted may require the use of specific techniques borrowed from Delay Tolerant Networks (DTN) in order to fulfill data transmission, but delay sensitive applications may not work at all. On the other hand, a dense network will have to deal with congestion, which can lead to packet loss and to a delay increase in data delivery, too. In a dense network, other problems, such as a broadcast storm, may appear when an emergency message needs to be distributed if the appropriate strategies for rebroadcasting are not used. Wireless communications can suffer from security attacks against availability, rendering the network inoperative. Data confidentiality violation, data integrity tampering, and identity theft are implicit risks in wireless networks. The high relative speed of mobile nodes means, in practice, that the time window in which two mobile nodes can communicate with each other in a direct way is very narrow. There is an exception to this rule when two vehicles are traveling in the same direction on a road and they maintain very similar speeds. The mobility of nodes also affects in a significant way the topology of the network. Vehicular networks are continuously changing their topology. This a big drawback for routing tasks, and, implicitly, unicast communications. All these are intrinsic problems due to the specific characteristics of vehicular networks.

1.1.2 Network technologies

Regarding the technologies that can provide support to vehicular networks, we can make a classification by dividing them into two categories: long range communications and short range communications.

Long range communications are well suited to V2I use. The two technologies that better fulfill the constraints of vehicular networks are cellular networks (3G, 4G, future 5G) and WiMAX (Worldwide Interoperability for Microwave Access) [19]. Other types of technologies may be useful in some unusual situations, or in certain geographic areas. For instance, satellite Internet access may be very helpful in areas where cellular and WiMAX networks have not been (and probably never will be) deployed. Emerging technologies may open new alternatives for long range communications, like the use of cognitive radio to access TV white spaces. But currently, cellular networks and WiMAX are the best choices, and, of these two, cellular networks have a broader spread coverage area. Long range communications have some advantages. The main benefit is that these networks are already deployed and working, and use well-known technology, so they can be used right now. Integration of vehicles with existing cellular infrastructure via smartphones and digital tablets can be accomplished in a straightforward way. ITS applications that use centralized services can take particular advantage of long range communications. One of the drawbacks of long range communications is an excessive delay for communications between locally close nodes (in comparison with short range communications). Also, long range communications are highly dependent on network access providers.

Short range communications have a natural orientation to V2V communications, although they can be used by infrastructure, both to act as one more node in the network topology, and to act as a network gateway. Preexisting technologies like ZigBee, Bluetooth, and WiFi are not useful because of vehicle mobility. Instead, these technologies may be useful in particular examples of use, like when a vehicle arrives at a parking lot, a gas station, or a private garage annexed to a house. New short range technologies have been developed for their specific use in vehicular networks. In the USA and Europe, the new IEEE 802.11p standard (amendment "p" of the IEEE 802.11 standard) has been selected for PHY and MAC layers of the network. For the upper layers of the network, Europe and the USA have gone different ways. The USA has adopted the IEEE 1609 family of standards. The combination of IEEE 802.11p and the IEEE 1609 family of standards is known as WAVE (Wireless Access in Vehicular Environments). Europe, instead, has adopted ETSI ITS-G5 [20]. Japanese short range communications are based on other different technologies and use the standards developed by the Association of Radio Industries and Businesses (ARIB): ARIB STD-T75 [21] and ARIB STD-T88 [22]. Short range communications have the smallest latency possible and are independent from network access providers, as they use frequency bands specifically allocated by governments for ITS use. However, they need gateways to access centralized services, and there is still scarce market penetration in vehicles and a lack of infrastructure (in both the USA and Europe).

Short range technologies are used to create self-configuring networks in which nodes change their links to other nodes because of vehicle mobility. The main consequence is that the topology of the network is continuously changing. These networks are referred to as Vehicular Ad-Hoc Networks (VANETs). The term "ad-hoc" emphasizes the sporadic nature of connectivity between nodes. VANETs are considered a subclass of Mobile Ad-Hoc Networks (MANETs) in which the nodes have restrictions in their range of motion, which is limited to streets, roads, and highways (apart from a few rare exceptions, like all terrain vehicles on a field trip).

Short range and long range communications are especially suitable for different types of ITS applications. A right approach to provisioning ITS with communication abilities is to provide vehicles with both types of technologies (short and long range) in order to provide applications with the best communication choice. Several combinations of use may arise. One of them is using vertical handover. In vertical handover, the vehicle can select which of the technologies to use at every moment. Switching between technologies is transparent for users, and the switch decision is made considering diverse criteria like coverage, available bandwidth, or even service cost. Another alternative is to use both technologies simultaneously.

This thesis deals with video streaming over vehicular networks and specifically over VANETs using WAVE, so in the next section the standards that form this technology will be detailed.

1.1.3 IEEE 802.11p and IEEE 1609 family of standards

In 1997, the Intelligent Transportation Society of America (ITSA) petitioned the Federal Communications Commission (FCC) for 75 MHz of bandwidth in the 5.9 GHz band with the specific goal of supporting Dedicated Short-Range Communications (DSRC) for ITS. The FCC granted the request in October 1999. The DSRC-based ITS radio services received 75 MHz of spectrum in the 5.85-5.925 GHz range. In 2004, Task Group "p" (TGp) of the IEEE 802.11 Working Group assumed the role of developing an amendment to the IEEE 802.11 standard [23] to include vehicular networks. The document is known as IEEE 802.11p [24]. The IEEE 1609 Working Group undertook the task of developing specifications to cover additional layers in the protocol suite. Currently, the IEEE 1609 standards set consists of six documents: IEEE 1609.0 - IEEE Guide for Wireless Access in Vehicular Environments (WAVE) - Architecture [25], IEEE 1609.2 - Security Services for Applications and Management Messages [26], IEEE 1609.3 - Networking Services [27], IEEE 1609.4 - Multi-channel Operation [28], IEEE 1609.11 - Over-the-Air Electronic Payment Data Exchange Protocol for Intelligent Transportation Systems (ITS) [29], and IEEE 1609.12 - Identifier Allocations [30]. As stated before, the IEEE 802.11p standard, together with the IEEE 1609 family of standards, are called WAVE.

A WAVE system consists of Road Side Units (RSUs), which can be installed on light poles, traffic lights, road signs, or newly deployed infrastructure; and On Board Units (OBUs), which are mounted in vehicles.

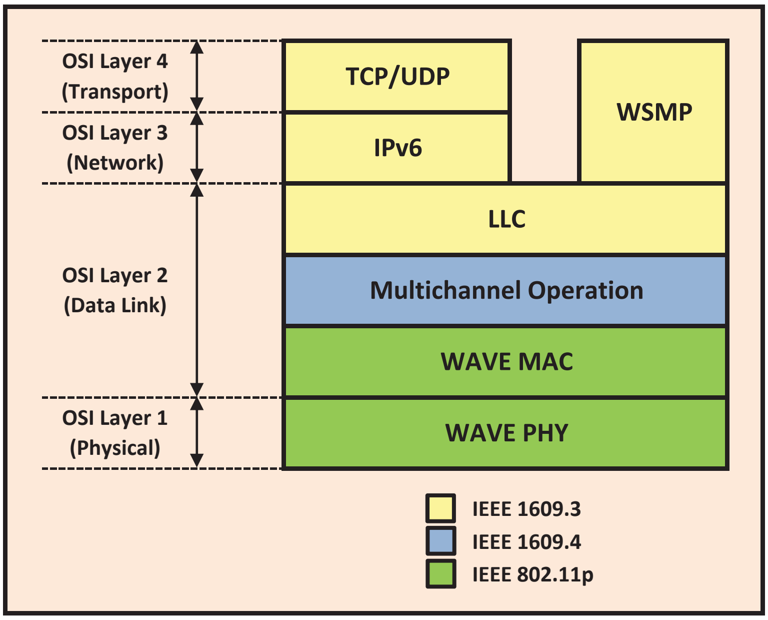


Figure 1.1. WAVE communication stack.

Vehicles are usually moving but they can also operate as static nodes while they are parked. By default, WAVE units operate independently, exchanging information over a fixed radio channel known as the Control Channel (CCH). However, they can also organize themselves in small networks called WAVE Basic Service Sets (WBSSs), which are similar in nature to the service sets defined in the IEEE 802.11 standard. WBSSs can consist of OBUs only, or a mix of OBUs and RSUs. All the members of a particular WBSS exchange information through one of several radio channels known as Service Channels (SCHs). Through the appropriate portals, a WBSS can connect to a wide-area network.

The WAVE architecture supports two protocol stacks, as shown in Figure 1.1. In the terminology of the OSI model, both stacks use the same physical and data-link layers, and they differ from each other in the network and transport layers. The WAVE standards do not specify session, presentation, or application layers. The two stacks supported by WAVE are Internet Protocol version six (IPv6) and a proprietary one known as WAVE Short-Message Protocol (WSMP). The reason for having two protocol stacks is to accommodate high-priority, time-sensitive communications, as well as more traditional and less demanding exchanges such as Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) transactions.

The WAVE architecture is based on the IEEE 802.11p standard, which

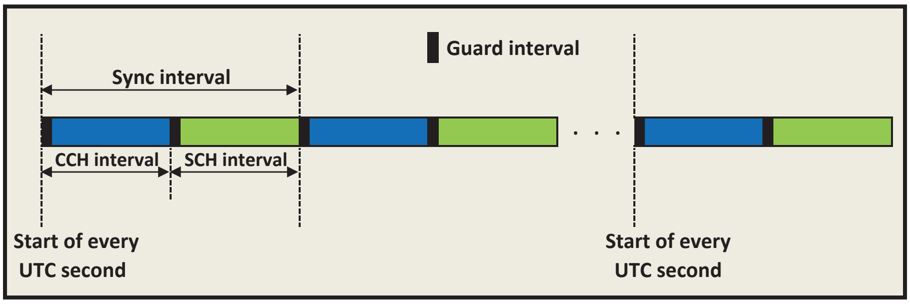


Figure 1.2. CCH, SCH, Guard and Sync intervals.

specifies layer 1 and part of layer 2 of the protocol stack. WAVE units are required to divide their time between the CCH and the SCHs. Therefore, the WAVE protocol stack includes a sublayer at the level of OSI layer 2, dedicated to controlling this multi-channel operation. This sublayer (including the associated management functions) is specified in IEEE 1609.4. The remaining part of OSI layer 2, the Logical Link Control (LLC), follows the IEEE 802.2 standard. At the level of OSI layers 3 and 4, IEEE 1609.3 specifies the aforementioned WSMP and explains how to incorporate traditional IPv6, UDP, and TCP in the systems.

The main specific characteristics of 802.11p are the following: ability to carry out communications in a highly mobile environment; 10 MHz channels (one-half the data rates of 802.11); one CCH and six SCHs; unique ad-hoc mode; random MAC address; high accuracy for the Received Signal Strength Indication (RSSI); and 16 QAM used in the high-speed mobile environment.

As illustrated in Figure 1.2, IEEE 1609.4 describes a concept of channel intervals in which time is divided into alternating Control Channel (CCH) and Service Channel (SCH) intervals; this is called Multi-channel Operation. The general concept calls for each interval to be 50 ms long. A pair of CCH and SCH intervals form a Sync interval. There are ten Sync intervals per second. This is motivated by a desire to map Sync intervals to the generally assumed 10 Hz vehicle safety messaging rate. The start of a CCH interval is aligned with the start of a Coordinated Universal Time (UTC) second or multiples of 100 ms thereafter. It is generally envisioned that a DSRC on board unit should, by default, be tuned to the CCH to send and receive safety messages continuously. If it is engaged in some non-safety application communications in a SCH, then it is expected to actively switch between CCH and SCH channels for the duration of the service session. With this alternating channel access, the DSRC radio is used for safety communications during CCH intervals and used for other applications during SCH intervals. Each DSRC radio, even if it is in continuous access on the CCH, is expected to track the start and end of CCH and SCH intervals at all times. The concept is that such a radio would send safety messages during the CCH interval for the benefit of other nearby radios that might be engaging in alternating channel access. The standard further defines a Guard interval at the start of each channel interval, be it SCH or CCH. This is meant to account for radio switching and timing inaccuracies between different devices. Accordingly, the Guard interval is defined as the sum of the Sync_Tolerance and Max_Channel_Switch_Time parameters. Sync_Tolerance describes the expected precision of a device's internal clock in aligning to the UTC time. Max_Channel_Switch_Time is the time overhead for a radio to be tuned to and made available in another channel. Currently, the assumed value for the Guard interval ranges from 4 to 6 ms. If a radio is actively switching channels, it suspends MAC activities at the start of a Guard interval. After the channel switching and at the end of the Guard interval, it starts the communications activities on the new channel or resumes such activities if they were suspended from the last Sync interval. This multi-channel operation forces the CCH and the SCH to share the available bandwidth, and also adds two more drawbacks. The first one is the penalization that the Guard intervals introduce. The second one is that every packet that is not sent during the interval to which it belongs has to wait until the next appropriate interval arrives. This may cause collisions at the beginning of the next appropriate interval if some vehicles have pending packets.

1.2 Video coding

Video streaming can be very useful in ITS applications in the four previously depicted categories: *road safety*, *improved driving*, *public services*, and *business and infotainment*. For instance, in *road safety*, it can help us to monitor the weather conditions (fog, rain, etc.) of a certain area to which we are approaching, or the traffic flow density on a long avenue or on a highway where an accident has happened. Regarding *improved driving*, we can enhance our visibility by virtually seeing "through" other vehicles, like a huge lorry, by means of receiving visual information captured and sent by the lorry itself to the rear vehicles. *Public services* (medical ambulances, fire brigades, etc.) can improve their intervention if they receive a video sequence in which they can see and precisely assess the condition of injured people or the crashed vehicles after an accident (emergency video call). The *business and infotainment* category may have lots of examples where video streaming can be useful. Contextual advertising, tourist information, video conference, and Video On

Demand (VOD) are a few examples of ITS applications belonging to this category. It is very important to remark that drivers should not, under any circumstance, take their eyes off the road while driving. Video streaming applications must be used by other people inside the vehicle (copilot, passengers), or by the driver, but only when the vehicle is completely stopped.

As we have just said, video streaming can be beneficial for ITS applications, but there is a drawback: digital video generates very big deals of data. This huge amount of captured data needs to be stored and, above all else, it has to be transmitted. To illustrate this statement, we will give an example: a digital raw video sequence of 832x480 pixels per frame, in YUV 4:2:0 color format (12 bits per pixel), at a frame rate of 30 frames per second, generates a bit flow of 143.77 Mbps. Currently, this bit rate is not manageable by a vehicular network, taking into consideration current network characteristics (neither is it by many other communication networks). This is the main reason why video compression becomes mandatory. Continuing with the previous example, this same video sequence can be compressed while keeping a very high visual quality level, generating a bit stream of 1.67 Mbps. This means 1/86 of the original value. The two main drawbacks of video coding are: (a) the computational cost introduced by the encoding and decoding processes (usually the encoding process is several orders of magnitude more complex than the decoding process), and (b) the fact that the encoding process increases the vulnerability of video against data loss (and vehicular networks are loss prone scenarios by nature). In this thesis, we have used a hybrid video coding scheme that will be explained in the next section.

1.2.1 Hybrid video coding

Hybrid video coding is a scheme used by many video *codecs* (coder-decoder) that includes both *predictive coding* and *transform coding* in the compression and decompression processes. A hybrid video coding scheme consists of several stages. First, it makes use of the temporal or spatial redundancy present in a video sequence in order to *predict* a region of a frame, and this prediction is subtracted from the region that is currently being encoded. Then the residuum of this subtraction is *transformed* into the frequency domain and the resulting coefficients are quantized (lossy compression). Finally, the quantized coefficients are ordered and entropy coded. We will now explain this process in a more detailed way.

In the encoding process, each frame is divided into small square regions. Let's call them blocks. These blocks can be encoded using one of three modes: (a) without any prediction, (b) using spatial prediction, or (c) using temporal prediction. *Spatial prediction* exploits redundancy within a frame. In order to encode a block, it uses previously encoded regions of the same frame to search for pixel information that is similar to that block in order to create a candidate. Then, this candidate is subtracted from the current block and thus we obtain the residuum of the prediction. This method is also called *intra*-frame prediction. *Temporal prediction* uses previously encoded frames to estimate a block candidate by means of the search of a similar block in other frames (called reference frames), which have been previously encoded, decoded, and stored in a buffer. It exploits temporal redundancy, taking advantage of the fact that nearby frames usually contain blocks that are very similar to the current block and so the residuum of the compensation is close to zero. This method is also called *inter*-frame prediction. For the three encoding modes, a domain transform is applied to the resulting values, so they are converted into the frequency domain. Then, these coefficients are quantized. This quantization introduces a loss of quality (to a certain extent) in the video stream, because, after the quantization process, the original coefficients cannot be exactly recovered. In the decoding process, the quantized coefficients will be scaled and these scaled coefficients will not be exactly the same as the original ones. This is the reason for the loss of information (lossy compression). Depending on the quantization step, the set of quantized coefficients will be more detailed (lower compression, higher quality), or less detailed (higher compression, lower quality). Finally, the quantized coefficients, together with the side information of the process (selected coding mode, motion vectors, etc.), are compressed by an entropy encoder.

1.2.2 High Efficiency Video Coding

In January 2013, High Efficiency Video Coding (HEVC) [31] was agreed as a video coding standard. By 2010, the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Pictures Experts Group (MPEG) joined their research efforts and constituted the Joint Collaborative Team on Video Coding (JCT-VC) in order to develop a new video coding standard that would improve the previous one, H.264/AVC (Advanced Video Coding) [32], and would be able to keep pace with the growing resolutions and frame rates of new video contents. The new standard follows the same block-based hybrid compression scheme as its predecessor, but includes a good number of refinements and new features that make it nearly double the coding efficiency as H.264/AVC. These refinements make HEVC one of the current most proficient video *codecs*.

As in previous video coding standards by ITU-T and ISO/IEC, only the bit stream and the decoding process have been standardized. The syntax and semantics of the bit stream are part of the standard, and also the processes that

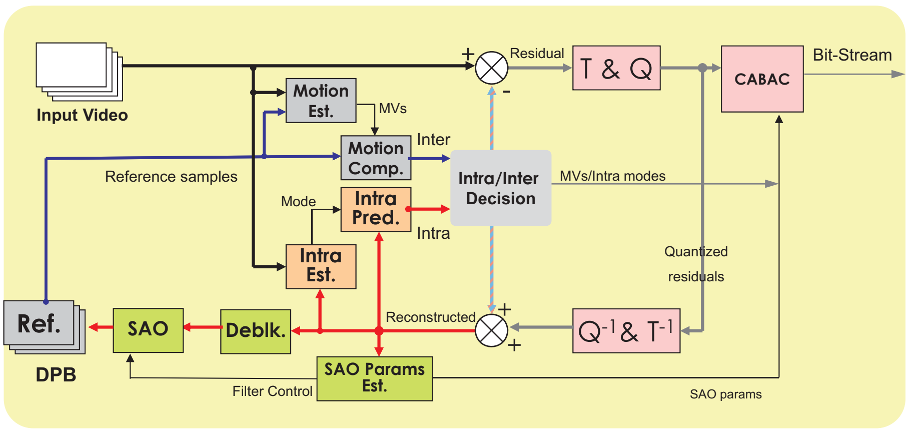


Figure 1.3. HEVC encoder. *(**T**=Transform; **Q**=Quantization; **CABAC**=Context-Adaptive Binary Arithmetic Coding; **DPB**=Decoded Picture Buffer; **SAO**=Sample Adaptive Offset; **MVs**=Motion Vectors; **Ref.**=Reference Frames).*

show how to correctly decode a bit stream. The process to encode a video sequence is not part of the standard. It is only remarked that an HEVC encoder must generate a bit stream 100% compliant with the constraints requested.

An encoding algorithm producing an HEVC compliant bit stream would typically proceed as shown in Figure 1.3. Each frame is split into block-shaped regions, with the exact block partitioning being conveyed to the decoder. The first frame of a video sequence is coded using only intra-frame prediction (that uses some prediction of data spatially from region-to-region within the same frame, but has no dependence on other frames). For all remaining frames of a sequence or between random access points, inter-frame temporally predictive coding modes are typically used for most blocks. The encoding process for inter-frame prediction consists of choosing motion data comprising the selected reference frame and Motion Vector (MV) to be applied for predicting the samples of each block. This process is called Motion Estimation (ME). The encoder and decoder generate identical inter-frame prediction signals by applying Motion Compensation (MC) using the MV and mode decision data, which are transmitted as side information. The residual signal of the intra- or inter-frame prediction, which is the difference between the original block and its prediction, is transformed by a linear spatial transform. The transform coefficients are then quantized, entropy coded using CABAC (Context-Adaptive Binary Arithmetic Coding), and transmitted together with the prediction information. The encoder duplicates the decoder processing loop such that both will generate identical predictions for subsequent data. Therefore, the quantized transform coefficients are

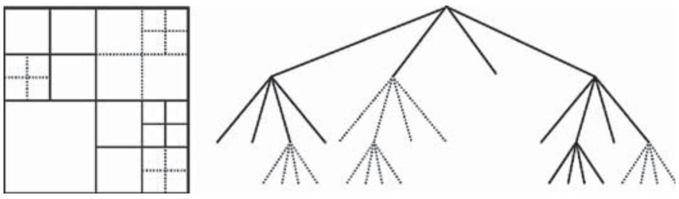


Figure 1.4. Example of a subdivision of a CTU (left) and the corresponding quadtree (right).

constructed by inverse scaling and are then inverse transformed to duplicate the decoded approximation of the residual signal. The residual is then added to the prediction, and the result of that addition may then be fed into one or two loop filters to smooth out artifacts induced by block-wise processing and quantization. The final frame representation (that is a duplicate of the output of the decoder) is stored in a Decoded Picture Buffer (DPB) to be used for the prediction of subsequent frames. In general, the order of the encoding or decoding processing of frames often differs from the order in which they arrive from the source, necessitating a distinction between the decoding order (i.e., bit stream order) and the output order (i.e., display order) for a decoder. Some of the features of HEVC are explained as follows.

(a) *Coding Tree Units*. The encoding process begins by dividing a frame into blocks. In H.264/AVC (the previous video coding standard by ITU-T and ISO/IEC), those blocks have a size of 16x16 pixels and are called Macro-Blocks (MB). In HEVC, the blocks into which every frame is divided are called CTUs (Coding Tree Units) because they can be recursively decomposed into smaller blocks, forming a structure that is called *quadtree* (see Figure 1.4). CTUs can have a maximum size of 64x64 pixels. This increase in the size of coding blocks (regarding H.264/AVC) improves coding efficiency mainly in homogeneous regions of frames, and more noticeably in video sequences with high resolutions. In Figure 1.4, the subdivision of a CTU and the corresponding quadtree are shown.

(b) *Intra-frame prediction*. The decoded boundary samples of adjacent blocks are used as reference data for spatial prediction. Intra-frame prediction supports 33 directional modes (compared to eight such modes in H.264/AVC), plus planar (surface fitting) and DC (flat) prediction modes. The selected intra-frame prediction modes are encoded by deriving most probable modes (e.g., prediction directions) based on those of previously decoded neighboring blocks.

(c) *Inter-frame prediction*. Quarter-sample precision is used for the MVs, and 7-tap or 8-tap filters are used for interpolation of fractional-sample

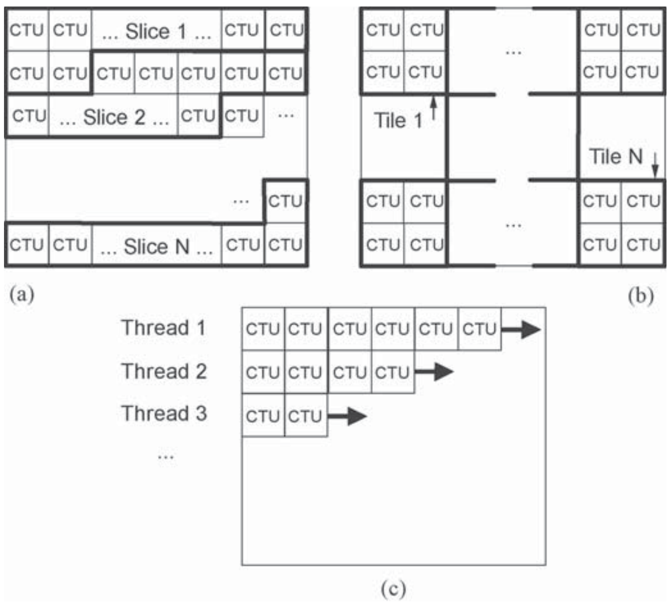


Figure 1.5. Representation of: (a) Slices; (b) Tiles; and (c) Wavefront Parallel Processing.

positions (compared to six-tap filtering of half-sample positions followed by linear interpolation for quarter-sample positions in H.264/AVC). As in H.264/AVC, multiple reference frames are used. For the predictions, either one or two motion vectors can be transmitted, resulting in either unidirectional predictive or bidirectional predictive coding, respectively.

(d) *Motion vector signaling.* Advanced Motion Vector Prediction (AMVP) is used, including derivation of several most probable candidates based on data from adjacent blocks and the reference picture. A merge mode for MV coding can also be used, allowing the inheritance of MVs from temporally or spatially neighboring blocks. Moreover, compared to H.264/AVC, improved skipped and direct motion inference are also specified.

(e) *Transform coding.* The prediction residual is coded using block transforms. Integer basis functions similar to those of a Discrete Cosine Transform (DCT) and a Discrete Sine Transform (DST) are defined.

(f) *Entropy coding.* Context-Adaptive Binary Arithmetic Coding (CABAC) is used for entropy coding. This is similar to the CABAC scheme in H.264/AVC, but has undergone several improvements to improve its throughput speed (especially for parallel-processing architectures) and its compression performance, and to reduce its context memory requirements.

(g) *Sample Adaptive Offset (SAO).* Nonlinear amplitude mapping is introduced within the inter-frame prediction loop after the deblocking filter. Its goal is to better reconstruct the original signal amplitudes by using a look-up table that is described by a few additional parameters that can be determined by histogram analysis on the encoder side.

(h) *Slices.* A slice is a data structure that can be decoded independently from other slices of the same picture in terms of entropy coding, signal prediction, and residual signal reconstruction. A slice can either be an entire frame or a region of a frame (which is composed by consecutive CTUs in raster order). A frame divided in several slices is shown in Figure 1.5.a. One of the main purposes of slices is resynchronization in the event of data loss. In the case of packetized transmission, the maximum number of payload bits within a slice is typically restricted, and the number of CTUs in the slice is often varied to minimize the packetization overhead while keeping the size of each packet within this bound.

(i) *Tiles.* The option to partition a picture into rectangular regions called *tiles* has been specified. The main purpose of tiles is to increase the capability for parallel processing rather than provide error resilience. Tiles are independently decodable regions of a frame that are encoded with some shared header information. Tiles can additionally be used for the purpose of spatial random access to local regions of video pictures. A typical tile configuration of a picture consists of segmenting the picture into rectangular regions with approximately equal numbers of CTUs in each tile (see Figure 1.5.b). Tiles provide parallelism on a coarse level of granularity (picture/subpicture), and no sophisticated synchronization of threads is necessary for their use.

(j) *Wavefront Parallel Processing (WPP).* When Wavefront Parallel Processing is enabled, a slice is divided into rows of CTUs. The first row is processed in an ordinary way; the second row can begin to be processed only after two CTUs have been processed in the first row; the third row can begin to be processed only after two CTUs have been processed in the second row, and so on. The context models of the entropy coder in each row are inferred from those in the preceding row with a two-CTU processing lag. WPP provides a form of processing parallelism at a rather fine level of granularity, i.e., within a slice. Figure 1.5.c shows the parallelization scheme provided by WPP. WPP may often provide better compression performance than tiles.

Slices and tiles have been specifically used in our research. In the next chapters, information about these two features will be examined.

1.2.3 Error concealment and error resilience

Error concealment (EC) techniques try to hide (minimize) the impact of data loss in the quality of a delivered video. These techniques are applied on the receiver side while the received bit stream is being decoded. They use the correctly received data to infer what the missing or corrupted data should originally be, or, at least, approximate it. If one part of an encoded frame gets lost (for example, a slice), then the region that this slice covers cannot be decoded and rendered. If this area is really small then, by exploiting similarities of nearby pixels, spatial EC can be accomplished by interpolating the values of the correctly decoded pixels that are located around the missing area [33]. If the small missing area corresponds to a uniform region, then the difference between the missing original area and the error-concealed one may be slight and so the quality loss may be correspondingly small. This does not usually apply to large areas unless they are also uniform (which is not usually probable). Another well-known EC technique exploits temporal redundancy in video sequences by applying a motion compensated concealment with the zero motion vector (defined as “zero-MV concealment” in [34]). In this technique, the missing area of a frame is replaced by the co-located area of the previous frame. This technique has a particular case in which a whole frame is missing or corrupted. In this case, the last correctly decoded frame is duplicated in order to conceal the loss of the missing one. This technique is known as “frame copy concealment” in [35]. Another method used to achieve EC is to use the motion vectors around the missing area (in the case of an inter-frame predicted zone) in order to make an estimation of the motion vectors for each missing block. After that, motion compensation is calculated with the use of the estimated MVs [36]. In some video *codecs* (such as H.264/AVC), encoded data can be partitioned into different categories regarding the importance of their elements in the final quality of the reconstructed video when they are missing. In this case, the division of encoded data does not only have to do with spatial partitions (frames, slices), but with the “semantic” contribution to the decoding process as well. When data partition is used, more sophisticated EC can be achieved, taking into consideration which of the partitions is missing or corrupted [37].

Each one of the aforementioned techniques can be more or less useful depending on the magnitude of data loss, and also depending on the particular features of the video fragment that needs to be recovered (spatial homogeneity, scenes with much or little motion, etc.). In most cases, EC is not completely satisfactory (i.e., the error cannot be completely concealed), but also in most cases, the error-concealed video quality increases over the “turning a blind eye” approach.

Error resilience (ER) techniques try to protect video sequences against transmission loss and errors by acting on the source of the streaming. Some of these techniques protect data by introducing mechanisms that try to recover lost packets in a direct way; some others change the video encoding process in order to ease the work of error concealment tools on the receiver side; and some others tune the encoding decisions to improve the final reconstructed video quality in the presence of information loss. Providing ER properties to a video sequence has a drawback: the data rate increases. In some cases, this is a result of adding redundancy to the video stream (in order to recover missing information); in other cases, it is due to a change in the coding settings, which increases error resistance, but produces a decrease in compression efficiency.

In HEVC, some of the ER techniques included in H.264/AVC [38] have been withdrawn, such as (a) Flexible Macroblock Ordering (FMO) (together with Slice Groups), which can help error concealment techniques (for example, with a "checkerboard" distribution of MacroBlocks) or provide special protection to Regions Of Interest (ROI); (b) Data Partitions, which allow graceful degradation of frames even if part of the encoded bit stream is lost or corrupted; and (c) SP and SI frames, which allow drift-free bit stream switching in scenarios where the propagation delay is low enough to allow for ACK-based retransmissions.

One of the HEVC features (and also of other *codecs*) that greatly increases coding efficiency but which also increases vulnerability of the bit stream against data loss is inter-frame coding. An inter-coded frame contains CTUs that are based on motion estimation and compensation, using other frames as reference. So, for an inter-coded frame to be correctly decoded, firstly, it needs to be correctly received and, secondly, all the reference frames that it uses must have been correctly decoded. Note that if any of the frames used as reference is an inter-coded frame, then all the reference frames used by it need also be correctly decoded, and so on. Therefore, inter-coded frames introduce a high grade of dependency between frames, which can lead to an "avalanche" effect in error propagation. To stop this drift, usually intra-frame coding is somehow used.

Some of the ER techniques utilize a *feedback channel* that the receiver uses to warn the bit stream source about the missing or erroneous packets. In this situation, the source can react both (a) resending the missing/erroneous packets or (b) encoding the next undelivered frame as an intra-coded frame in order to stop inter-coded frame drift, and setting the reference picture list to avoid using reference pictures, which are located before this intra-coded frame. Both the (a) and (b) classes of techniques introduce some delay, which is due to the time needed for the detection of the missing/erroneous packet, plus the time that the warning message takes to arrive at the bit stream source.

In addition to this delay, the first class of techniques adds another delay, because the receiver has to wait until the resent packets arrive. In the second class of techniques, the delay is increased because of the time that the source needs to re-encode the next undelivered frame. Feedback-based techniques are not feasible in situations in which there is no possibility of having such a feedback channel (such as video broadcasting), or when time constraints do not allow the extra delay. Feedback-based ER methods have not been taken into consideration for this work because of the specific features of vehicular networks.

Other ER techniques consist of dividing bit stream into several substreams. Within this category, two different classes of techniques can be found: (a) Multiple Description Coding (MDC) and (b) Layered Coding (LC). *MDC* divides the bit stream into two or more substreams called "descriptions." Each description can be used independently to create a reconstructed version of the video sequence, but with limited quality. If more descriptions are received, quality will increase. The different descriptions are usually sent through different paths in order to ensure that, at least, one of the descriptions will be able to arrive at its destination and the video sequence will be decodable. *LC* divides the bit stream into several layers: one base layer and one or more enhancement layers. The base layer is essential for the decoding; enhancement layers progressively improve quality. There is a hierarchical dependency between enhancement layers, because they have a fixed order, and to use one enhancement layer, the previous enhancement layers need to be correctly received. LC is also a useful tool for spatial or fidelity scalable video coding. Unequal Error Protection (UEP) strategies can be used together with both MDC and LC to protect one or more substreams. A logical rule in LC is the following: the more important a layer is, the more protection it should receive. In MDC, one or more substreams may be protected while leaving some others unprotected. This behavior reduces the introduced overhead.

Another group of ER techniques are the so-called "intra refresh" methods. They encode the bit stream in such a way that it avoids quality from collapsing in the presence of data loss. An inter-coded frame, although correctly received, can be "infected" by an erroneous reference frame. And this "infected" frame can "infect" other inter-frames if they use it as reference. As stated before, an intra-coded frame, if properly received, may stop the error propagation throughout inter-coded frames. Although this is not always true, as it depends on the reference frame selection scheme. For example, one inter-coded frame may have a reference frame that is located before the "troubleshooter" intra-coded frame. So, in this case, the intra-coded frame does not at all stop the error expansion through inter-coded frames. A proper insertion of intra-coded frames together with an adequate selection of reference frames can add error resilience features to an encoded bit stream. Later on, in the following chapters, we will outline some of the “encoding modes” proposed in this work to make bit streams more robust.

Not only may a complete intra-coded frame be inserted into the bit stream to stop error propagation, but intra refresh strategies can also be carried out on a small scale (such as on the CTU level) or on a medium scale (such as on the slice level). These different scales of intra refreshing can have different results on the final reconstructed video sequence.

In this work, we have explored intra-frame refresh on frame, tile, and CTU levels, together with the use of Application Layer Forward Error Correction (AL-FEC).

1.3 Forward Error Correction

Forward Error Correction (FEC), also known as *channel coding*, is a technique developed in 1950 by Richard Hamming. It is used in telecommunication channels where data loss occurs. The main basis of this technique is to add some redundancy data, called Error Correction Codes (ECCs), to the original data. ECCs are mixed with the original data in such a way that missing information can be detected on the receiver side. If transmission errors remain under a certain limit they can be detected, and, in some cases, the redundancy data added can be used to recover the missing original data without any retransmission. This technique is especially useful in networks or applications where retransmissions of the lost data are not feasible. Examples of this type of scenarios are one-way communication channels (digital TV), long-delay networks (satellite communications), real-time applications where the delay of a retransmission is not acceptable (live video streaming), and so on.

In this thesis, in order to add channel protection to the data transmissions, we have used RaptorQ codes. They are a type of ECCs that exhibit good recovery properties and computationally efficient performance.

1.3.1 RaptorQ codes

Raptor codes, invented by Shokrollahi [39] [40], are a type of Fountain codes and are based on Luby Transform (LT) codes [41]. Raptor codes are an FEC technology that implements Application Layer protection against network packet losses. RaptorQ codes are a new family of codes that provide superior flexibility, support for larger source block sizes, and better coding efficiency than Raptor codes. They have several features that make them an interesting technology. One of them is that they can encode (protect) and decode (restore) data in linear time. They can also add variable levels of protection to better suit the protection to the network characteristics (e.g., Packet Loss Ratio, maximum bandwidth, etc.). One of their outstanding features is that, once the repair packets are added to the data flow, the RaptorQ decoder is able to recover the complete original data by receiving approximately the same amount of data that was originally produced (regardless whether received packets are original packets or repair packets). RaptorQ codes are very efficient and they usually have small memory and processing requirements, so they can be used in a wide variety of devices (from smartphones to big servers). Raptor and RaptorQ codes have been standardized by IETF [42] [43] and are used in 3GPP Multimedia Broadcast Multicast Services (MBMS) for file delivery and streaming.

This is how RaptorQ codes operate. The RaptorQ encoder receives a data stream (source packets) during a specified protection period. These packets are put together into memory to form a source block. A 4-byte FEC-trailer is added to each received packet. This trailer identifies the packet and the protection period to which it belongs. These FEC-protected packets are sent through the network. When the protection period finishes, the source block in memory is FEC-encoded into repair symbols that are placed into repair packets and sent through the network. On the receiver side, the RaptorQ decoder receives protected source packets and repair packets. Some of these packets may get lost or corrupted and the RaptorQ decoder tries to recover lost packets out of the group of source and repair packets correctly received.

Latency (and, in some cases, memory consumption) is the drawback of the RaptorQ protection scheme. As the protection window increases, the delay grows. Live events or real-time applications, like video conference, will have to use short periods for the protection window to keep latency within reasonable limits. Other types of streaming applications, like IPTV, may tolerate wider protection windows (while keeping latency within a reasonably interaction response time). And some other applications, like Video On Demand, can be much more flexible in enlarging the protection period that will provide greater bandwidth efficiency.

1.4 Related work

Several works can be found in the literature that address HEVC evaluation under data loss. In [44], the authors evaluate the efficiency of HEVC and compare it with H.264/AVC for streaming over best-effort networks (the Internet). They use three different video sequences encoded at five different bit rates and use the 1%, 3%, and 5% loss patterns specified in [45]. The frame copy EC method is used to conceal errors. They configure HEVC and H.264/AVC to obtain videos with the same quality (with no data loss) and then evaluate the perfomance of the *codecs* under loss conditions. They conclude that HEVC is more efficient than H.264/AVC (around 30% - 45%), but H.264/AVC is more resilient to data loss. In [46], the author compares HEVC with H.264/AVC in wireless environments. He uses one video sequence to evaluate packet losses that range from 0% to 40% (in steps of 10%) using channel conditions borrowed from [47]. He uses DFRM (Decoded Frame Rate Metric), SSIM (Structural SIMilarity), and PSNR (Peak Signal-to-Noise Ratio) metrics to evaluate the transmitted video quality under wireless conditions. His main conclusion is that HEVC is more error resilient than H.264/AVC under these loss conditions. In [48], the authors have developed a complete framework for testing HEVC under different packet loss rates, bandwidth restrictions, and network delays. This framework first generates a trace file with the bit stream information and then uses this trace file for the streaming simulations. On the receiver side, a log file is created that gathers the transmission results (including the missing packets). In the final stage, the framework decodes the complete bit stream (without any loss) and then overrides the areas corresponding to the missing packets (which are tagged in the log file). The authors use this framework to evaluate concurrent multipath transmission in multihomed mobile networks.

Video streaming in vehicular networks has also been studied by several authors. In [49], the authors enumerate some of the open problems (and proposed solutions) in video streaming in VANETs, including link connectivity, error resilience, clustering, and multihop routing. In [50], the authors address the problem of video streaming over urban VANETs. They use the H.264/AVC *codec* and some of its error resilience features, such as Flexible Macroblock Ordering and Redundant Frames, to protect the encoded bit stream. They use GloMoSim (Global Mobile System Simulator) to conduct IEEE 802.11p standard-based vehicular network simulations. In [51], the authors use the HEVC *codec* to encode video sequences in order to evaluate several flooding schemes for soft real-time video transmission in VANETs. The target application can be the timely delivery of video recorded by vehicles involved in an accident. This type of information may be useful for other vehicles located near the involved vehicles and also for public services, which may be located far from the accident site. They evaluate both the packet arrival ratio and PSNR value of the reconstructed video sequences.

Forward Error Correction is a well-known technique used to protect data delivery in all types of networks. It is, therefore, also applied in vehicular networks. In [52], the authors use FEC techniques for the protection of real-time multimedia streaming over vehicular networks. They propose a technique that is based on the optimization of the FEC and interleaving parameters under real-time constraints. Interleaving can help reduce the length of error bursts in order to improve correction abilities. In [53], the authors conduct thorough research on the protection of content delivery in vehicular environments searching for the best packet size in order to maximize throughput. They use different FEC techniques to protect data and provide their results in terms of packet arrival ratio and file transfer time.

One of the differences between the work done in this thesis compared to other works is that we have used the real HEVC reference software in the decoding of damaged/incomplete bit streams. The original reference software crashes (it results in an illegal instruction exception and stops working) if it tries to decode a bit stream with missing parts. This occurs because the decoder expects data with a specific format and when it encounters a gap, it cannot control the execution, and the program stops (without decoding the remainder of the bit stream). In order to use the real reference software decoder with "holey" bit streams, we modified the reference software to be aware of bit stream losses, allowing the decoder to continue its work until the end of the video sequence. Other works "emulate" the behavior of the decoder in the presence of missing parts by decoding the complete bit stream, and after that they remove the parts that correspond to the missing packets. This is not the real behavior of the HEVC decoder because, by proceeding in this way, the decoder can use all the information present in the bit stream for the decoding process and, as we realized in the patching process, some pieces of data depend on others (which may be missing) in the decoding process.

Some works use synthetic video data for their experiments, whereas we use well-known video sequences (belonging to the set in "Common Conditions for the evaluation of HEVC" [54]). Also, in some works, the evaluation of the quality of the decoded video sequence is represented in terms of Packet Loss Ratio (PLR) or other network performance metrics. PLR has influence on the final video quality but it is not a straightforward measure of video quality. The "semantic" meaning of the missing data is more important than the amount of data loss. Thus we use a video quality measure (PSNR) to measure the final reconstructed video quality.

We have introduced RaptorQ codes, a FEC protection strategy, by using the real software RaptorQ encoder and decoder. This means that when we protect bit stream network packets, a file is first generated with the "source" packets and the "repair" packets, and then a simulation of their transmission through the network is performed. After that, the RaptorQ decoder is used to try to recover missing packets in a real way. Again, no simulation or formulas have been used for the use of this type of FEC.

Regarding vehicular networks, we have used real maps for the experiments, obtained from geographic databases. The motion of the vehicles is accurately simulated by vehicular simulators that take into consideration lanes, traffic lights, crossroads, etc., and the interaction of vehicles with them and with other vehicles (decelerating and stopping, accelerating, etc.).

To obtain the results of video streaming over vehicular networks, we have conducted individual simulations for every tested combination, i.e., we have not used loss patterns. One video sequence (with different encoding settings and different protection parameters) may lead to different results in the loss patterns (due to its bandwidth usage, packet rate, etc.). This is the reason for performing each test with its own characteristics.

At last, we have used a mechanism that, to our knowledge, no other work has introduced: the combination of slices together with tiles to take advantage of the combined benefits of each one of them (error resilience features combined with coding efficiency).

We use a combination of source coding methods with channel coding methods to provide robust video streaming over vehicular networks.

Chapter 2

Video-protection proposals

Contents

2.1 High Efficiency Video Coding protection

The main objective of this thesis is to design proposals for robust video streaming over vehicular networks and evaluate them in a realistic environment. These proposals are based on two different (and complementary) approaches. The first approach focuses on video encoding itself and tries to produce a robust bit stream by means of error resilience and error concealment techniques. The second approach tries to provide data packets with protection against loss by using an Application Layer Forward Error Correction (AL-FEC) method: RaptorQ codes, which add redundant data to the bit stream in order to recover lost packets. This technique will be evaluated in detail in Section 2.2.

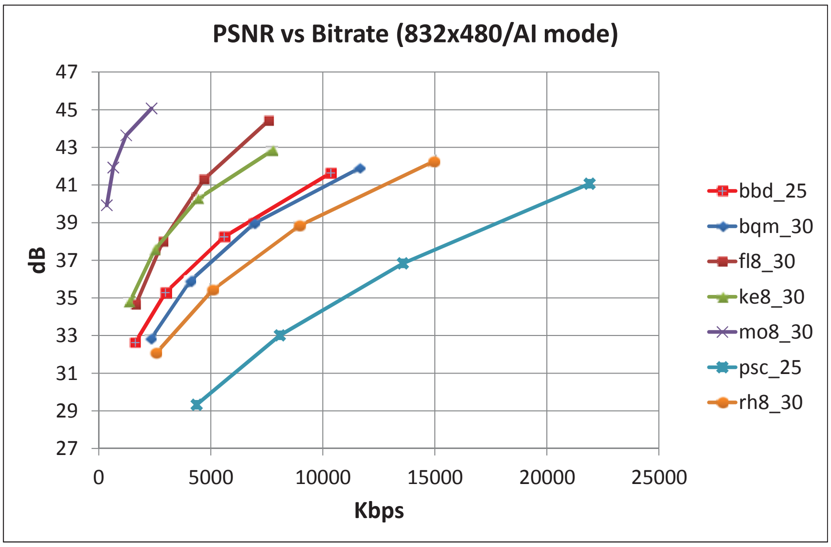
For the encoding of video sequences, the HEVC standard has been selected. In the Call For Proposals for the development of HEVC, one of the targets was doubling its predecessor's (H.264/AVC) efficiency, i.e., to be able to generate a bit stream 50% smaller than H.264/AVC on the same quality level. As stated in the introductory chapter, other works confirm that the real efficiency gains range from 30% to 45%. This "profit margin" is generally used to stream video sequences using fewer resources (by producing lower bit rates), or to stream sequences with higher resolutions and frame rates using the same resources. In this work, the room of improvement of HEVC over H.264/AVC is used to strengthen the encoded video bit stream (producing a much more robust video stream with the same bit rate as H.264/AVC).

First of all, the HEVC performance will be analyzed by encoding several video sequences on different compression levels and for two encoding modes. Fourteen video sequences, which belong to the HEVC "common test conditions" [54], have been selected. They are enumerated in Table 2.1. Seven of them have a resolution of 832x480 pixels and, the other seven, have a resolution of 416x240 pixels. Each sequence has a frame rate of 25 or 30 frames per second (FPS). A visual representation of the selected sequences is depicted in Appendix II. The two encoding modes used for the evaluation of HEVC are *All Intra* (AI) mode and *Low-delay P* (LP) mode, which are included in the HEVC reference software [55]. In AI mode, every frame of a video sequence is encoded as an I (*intra*) frame, so temporal prediction is not used at all. This mode has inherent error resilience properties because errors do not propagate to other frames. The main drawback of this mode is that it produces a bit stream with a high bit rate. One of the advantages of this encoding mode is that it encodes video faster than other encoding modes do. In LP mode, the first frame of the sequence is encoded as an I frame, and then P (*predictive*) frames are generated for the rest of the sequence. P frames are

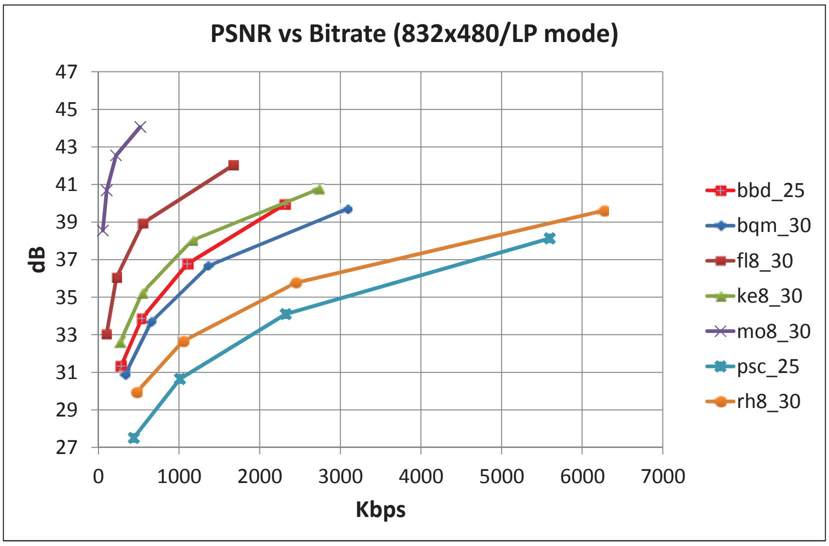
Table 2.1. Video sequences used in the tests.

Video sequence	Acronym	Resolution	FPS
BasketballDrill	bbd_25	832x480	25
BQMall	bqm_30	832x480	30
Flowervase	fl8_30	832x480	30
Keiba	ke8_30	832x480	30
Mobisode2	mo8_30	832x480	30
PartyScene	psc_25	832x480	25
RaceHorses	rh8_30	832x480	30
BasketballPass	bbp_25	416x240	25
BlowingBubbles	blo_25	416x240	25
BQSquare	bqs_30	416x240	30
Flowervase	fl4_30	416x240	30
Keiba	ke4_30	416x240	30
Mobisode2	mo4_30	416x240	30
RaceHorses	rh4_30	416x240	30

mainly formed by inter-coded CTUs, i.e., CTUs that are encoded by using motion estimation and compensation using other frames as reference. A P frame can also contain intra-coded CTUs (e.g., if the encoder estimates that those CTUs are more efficiently encoded in an intra way). In LP mode, the encoder has four available reference frames for every P frame from which to select the most appropriate one. In other encoding modes, reference frames can be selected from "future" frames (in display order), but, in LP mode, reference frames are always located before the current frame. The term "low-delay" is used to indicate that encoded frames can be decoded in display order and no frame has to wait for "future" frames to be decoded first. LP mode is much more efficient than AI mode and it generates a much smaller bit stream than AI mode on the same quality level, but a simple error in a frame is increased and propagated until the end of the sequence, even if the rest of the frames are correctly received. The compression rate of both modes can be selected by means of the Quantization Parameter (QP). This parameter is used by the quantization process, which entails the loss of precision in the transformed coefficients (lossy compression). This precision loss introduces distortion to a certain degree. If a high value is selected for the QP, the generated bit stream has a low bit rate and, correspondingly, low visual quality. If a low value is selected for the QP, the generated bit stream has a high bit rate and high visual quality. For the analysis of HEVC, 4 values for



(a) *AI mode*

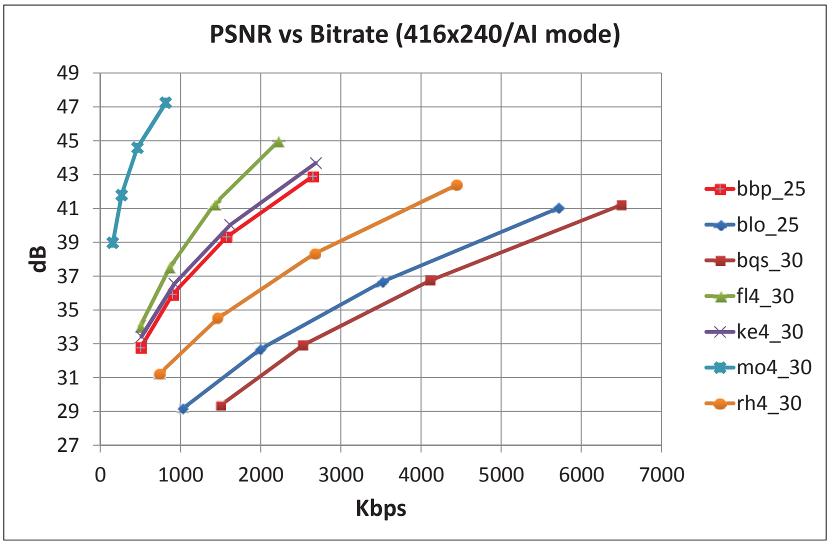


(b) *LP mode*

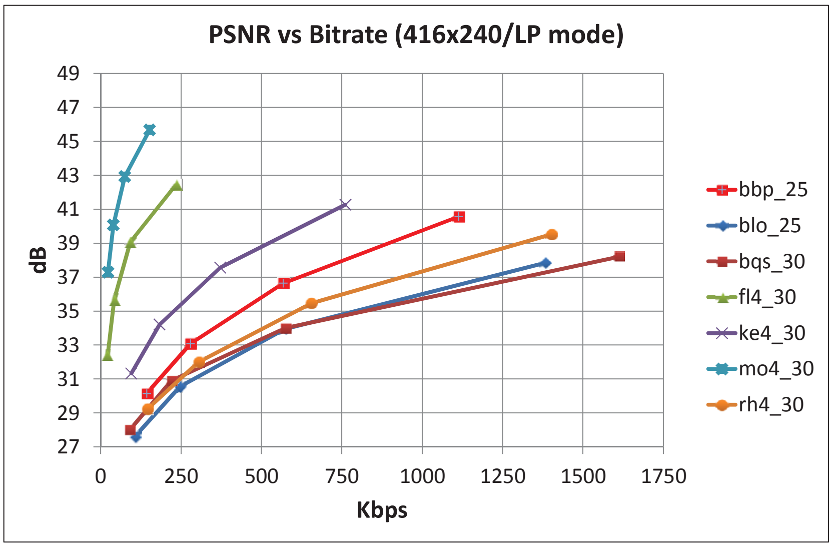
Figure 2.1. Rate/Distortion representation for 832x480 sequences encoded with the AI and LP modes at 1 slice per frame layout.

the QP have been used (22, 27, 32, and 37) as it is specified in the common test conditions.

In the example provided in the introductory chapter, it was explained that a digital raw video sequence of 832x480 pixels per frame, in YUV 4:2:0 color



(a) *AI mode*



(b) *LP mode*

Figure 2.2. Rate/Distortion representation for 416x240 sequences encoded with the AI and LP modes at 1 slice per frame layout.

format (12 bits per pixel) at a frame rate of 30 fps, generates a bit flow of 143.77 Mbps. A raw video sequence with a resolution of 416x480 pixels and a frame rate of 30 fps generates a bit flow of 35.94 Mbps. If these values are compared with those in Figure 2.1 (832x480 pixels per frame) and Figure 2.2

(416x240 pixels per frame) for the encoded sequences, they show that HEVC provides very good compression rates. For the 832x480 resolution, in AI mode (Figure 2.1(a)), the best rate/distortion proportion is obtained for sequence *Mobisode2* (`mo8_30`). It produces (at a value of 22 for the QP) a bit rate of 2.36 Mbps with a PSNR value of 45.07 dB. This is equivalent to 1.64% of the raw bit rate. The least efficient sequence is *PartyScene* (`psc_25`) with a bit rate of 21.90 Mbps and a PSNR value of 41.07 dB. This bit rate is equivalent to 18.28% of its raw bit rate (25 fps). For LP mode (Figure 2.1(b)), `mo8_30` produces a bit rate of 0.52 Mbps (0.36% of the original data) with a PSNR value of 44.07 dB, and `psc_25` produces a bit rate of 6.26 Mbps (5.22% of the original data) with a PSNR value of 39.60 dB. These figures demonstrate how LP mode clearly outperforms AI mode regarding coding efficiency. Also they reveal that source coding is highly dependent on the input data (raw video sequence) reflected in the wide range covered by rate/distortion curves. At the opposite end of the curves, when a value of 37 for the QP is used, lower bit rates can be obtained, and, as a result, lower quality values are produced. A value of only 0.05 Mbps is needed to obtain a PSNR value of 38.55 dB for `mo8_30` sequence in LP mode (this represents a mere 0.04% of the original bit rate). In Figure 2.2, the rate/distortion curves for the 416x240 sequences are shown. Again, there is a huge variability in the results of HEVC encoding regarding different sequences. Sequence *BQSquare* (`bqs_30`) provides the least efficient results, ranging from 18.08% of the original data (AI mode / QP value of 22) to 0.26% of the original data (LP mode / QP value of 37). Sequence *Mobisode2* (`mo4_30`) obtains the best rate/distortion curve, ranging from 2.27% of the original data (AI mode / QP value of 22) to 0.06% of the original data (LP mode / QP value of 37).

2.1.1 Slices and tiles

Slices are regions of an encoded frame that can be independently decoded, regarding other slices of the same frame. This implies that no spatial prediction can cross slice limits, and other elements in the bit stream that can be predicted, like motion vectors, cannot make use of information from outside slice boundaries either. Slices are not a new concept in HEVC and they were used in previous standards. The main purpose of slices is providing a certain level of error resilience to a frame. This makes them especially useful in packet loss prone scenarios. If a video sequence is encoded using only one slice per frame and the encoded slice is bigger than the network Maximum Transmission Unit (MTU), then the slice is divided into several fragments that travel inside several network packets. If one of these packets gets lost, then the rest of the fragments of the slice become completely useless because slices cannot be decoded if they are not complete. In this way, the real loss of a single packet implies the effective loss of all the packets that form the whole frame. But, if each frame is divided into several slices and each encoded slice is smaller than the network MTU, then each network packet contains a complete slice. The loss of a single packet does not imply the loss of the whole frame, but only the loss of part of the frame (the part covered by the slice), because the rest of the slices of the frame can be decoded (this is the aftermath of their independence). Slices consist of correlative CTUs in raster scan order (see Figure 2.3).The main disadvantage of dividing a frame into slices is that the coding efficiency is reduced because of the loss of some dependencies and predictions and because of the overhead introduced by slice headers. Slices are independent syntax units inside the encoded bit stream.

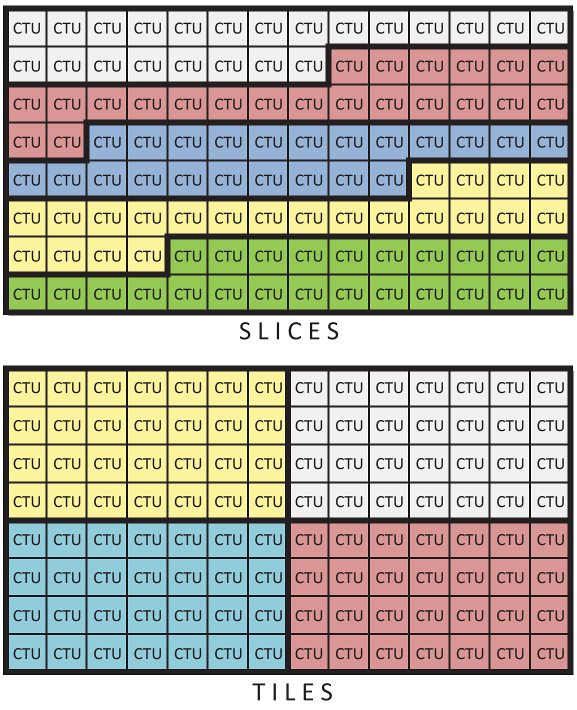


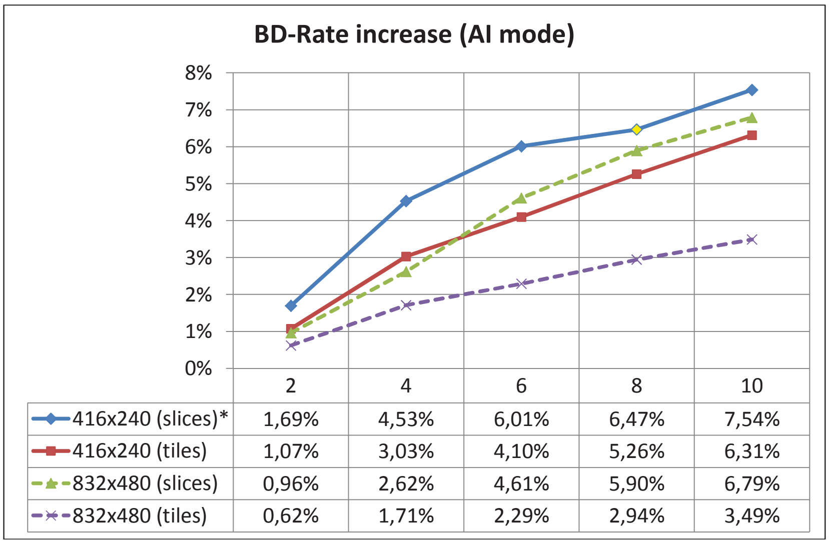
Figure 2.3. Representation of a frame divided into five slices (above) and a frame divided into four tiles (below). Slices and tiles consist of Coding Tree Units.

One of the new features presented in HEVC is the ability to divide a frame into rectangular regions, called *tiles* (see Figure 2.3), with the aim of incorporating parallel capabilities to video encoders and decoders. Tiles are independently decodable regions of a frame, which are encoded with some shared header information, but they are not independent syntax units of the bit stream. As they do not include heavy headers, and because of their rectangular shape, tiles are more flexible and efficient than slices, but they are not useful for error resilience purposes because they are not completely independent (as they do not form independent syntax units).

In the search for robust video streaming techniques, while attempting to keep the overhead introduced by slices low, we have devised a new element that could be called *tileslice*. First, one frame is divided into several tiles, and then each tile is placed into one slice. Every *tileslice* of a frame is a different syntax unit (because it is a slice), which has a rectangular shape (and this improves coding efficiency over typical shape slices). This is a possibility that HEVC includes in the definition of the standard, so it is not really a new invention, but, although it is a very simple idea, there are no previous works, to the author's knowledge, that use this particular combination of slices and tiles. Maybe this is due to the fact that, although slices are well known elements used in previous video coding standards, tiles are relatively new objects in the video coding research timeline.

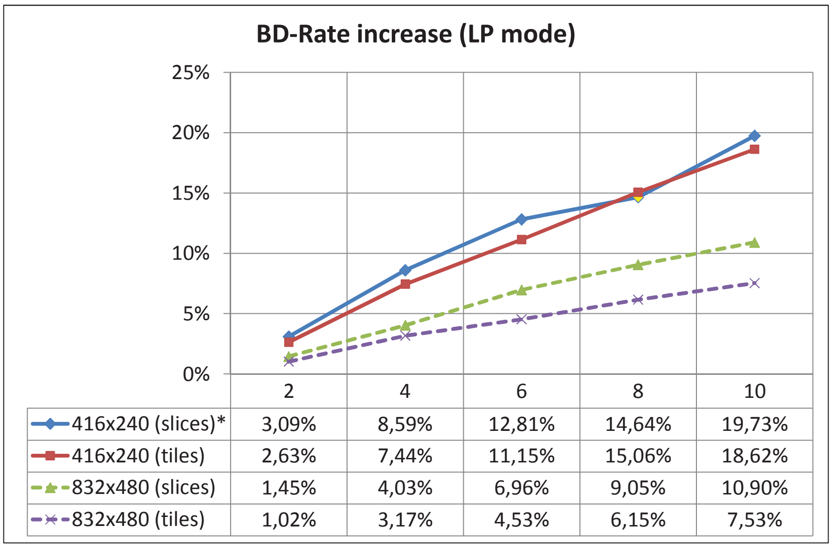
As *tileslices* contain slice headers, overhead introduced by them is not avoidable. The question that arises is the following: is it really worth using *tileslices* instead of typical slices? To answer this question, several overhead measurements have been made. From this point forward we will use the term *tile* to refer to what we have defined as *tileslice*, i.e., one rectangular slice that contains one tile. The size of a single slice (or tile) will depend on several factors: the resolution of the video frame, the type of the frame (I, P, ...), the quantization parameter (QP), etc. In our tests we have compared the performance of tiles over slices for six different layouts: 1, 2, 4, 6, 8, and 10 slices (or tiles) per frame. When we divide a frame into a certain number of slices the partition of the frame is almost unique because slices are formed by consecutive CTUs. But partitioning a frame into tiles is more flexible, because we can define the height of each one of the tile rows and the width of each one of the tile columns independently. From our experiments we have stated that tiles with a shape that approximates a square are more efficient than tiles with a shape that approximates a flat rectangle. For a visual comparison of the slice and tile partitions used in our evaluations, see the sketches in Appendix III. These pictures can provide a better understanding of the reasons for the differences in coding efficiency due to redundancy reasons.

In Figure 2.4, Bjørntegaard Delta Rate (BD-Rate) [56] values are averaged both for the whole set of 832x480 sequences and for the whole set of 416x240 sequences, obtained when encoding them with 2, 4, 6, 8, and 10 slices (or tiles) per frame over encoding them with 1 slice per frame (using the 4 values of the QP mentioned before: 22, 27, 32, and 37). BD-Rate is a measurement



	2	4	6	8	10
◆ 416x240 (slices)*	1,69%	4,53%	6,01%	6,47%	7,54%
■ 416x240 (tiles)	1,07%	3,03%	4,10%	5,26%	6,31%
▲ 832x480 (slices)	0,96%	2,62%	4,61%	5,90%	6,79%
✕ 832x480 (tiles)	0,62%	1,71%	2,29%	2,94%	3,49%

(a) *AI mode*



	2	4	6	8	10
◆ 416x240 (slices)*	3,09%	8,59%	12,81%	14,64%	19,73%
■ 416x240 (tiles)	2,63%	7,44%	11,15%	15,06%	18,62%
▲ 832x480 (slices)	1,45%	4,03%	6,96%	9,05%	10,90%
✕ 832x480 (tiles)	1,02%	3,17%	4,53%	6,15%	7,53%

(b) *LP mode*

Figure 2.4. Increase percentage of BD-rate for different slices per frame and tiles per frame layouts. *(* : in "416x240 (slices)" curves a 7 slices per frame layout is used instead of the indicated 8 slices per frame layout.)*

that indicates the percentage of increase (or decrease) of bit rate at the same visual quality. Because of the number of CTUs contained in one 416x240 frame, it is not possible to divide it into 8 similar slices, so, for the tests with

416x240 resolution sequences, partitions into 7 slices per frame are compared with partitions into 8 tiles per frame. These comparisons are not completely fair, because 7 slices per frame include one less slice header than 8 tiles per frame in every frame, so overhead due to slice headers favors the 7 slices per frame layout. The asterisk (*) that appears in this figure is to note this fact. Except in this specific case, all the values in the curves that represent the overhead of tile partition of frames are lower than those representing the overhead of slice partition of frames. This is the reason why the tile partition of frames will be used in the protection of the video bit streams. The maximum overhead savings are obtained for both LP and AI modes at 10 tiles per frame for the 832x480 sequences, where an average value around 3.3% of lower overhead (over 10 slices per frame) is obtained.

2.1.2 Encoding modes

In this first approach, which tries to make video bit streams intrinsically robust, encoding modes play a decisive role. In the previous tests, two encoding modes have been used: All Intra mode and Low-delay P mode. AI mode offers better error resilience properties than LP mode because an error in one tile does not propagate to other frames. On the other hand, LP mode is very sensitive to lost tiles, because P frames are “infected” by erroneous reference frames, and these infected P frames, when used as reference frames, “spread disease” until the end of the sequence. There is also an unpredictable reaction in LP mode in the presence of packet loss. If, for example, an LP encoded sequence loses the last frame, then the error affects only that frame. But if the first frame (I) gets lost, then no frame can be correctly reconstructed at all. So, the position of the missing parts is also important when dealing with errors in LP mode. These two encoding modes have been determined as our upper and lower bounds regarding error resiliency. One of the previously mentioned error resilience techniques are intra refresh methods. In this work, 7 new encoding modes have been proposed and evaluated, which introduce intra refresh to some extent. Table 2.2 enumerates the 9 encoding methods evaluated, which will be explained now.

- **AI** - Encodes every frame as an I frame. This mode is considered our upper bound mode.

- **LP** - Encodes the first frame as an I frame and the rest of the frames as P frames. Four previous reference frames are available for temporal estimation and compensation for each P frame. This mode is considered our lower bound mode.

- **IPx** - Similar to LP mode but every P frame uses only the previous frame as reference frame, instead of having 4 reference frames.
- **IPx25pctCTU** - Similar to IPx mode, where 25% of the CTUs are forced to be intra refreshed.
- **IPx25pctTIL** - Similar to IPx mode, where 25% of the tiles are forced to be intra refreshed.
- **IPxpattern** - Similar to IPx mode, where 25% of the tiles are forced to be refreshed following a specific pattern, which covers all the tiles of a whole frame every 4 frames.
- **IPPP** - Similar to IPx mode, with an I frame inserted every 4 frames.
- **LPI4** - Similar to LP mode, with an I frame inserted every 4 frames.
- **IPIP** - I and P frames are inserted alternatively. Every P frame always uses the previous I frame as reference.

Table 2.2. Frame layout and description for the evaluated encoding modes.

Acronym	Frame layout	Description
AI	IIIIIIIIIIII...	All Intra mode
IPIP	IPIPIPIPIPIP...	Alternating I and P
IPPP	IPPPIPPPIPPP...	One I, three Ps
IPx	IPPPPPPPPPPP...	Previous reference P
IPx25pctCTU	IPPPPPPPPPPP...	Random CTU refresh
IPx25pctTIL	IPPPPPPPPPPP...	Random TILES refresh
IPxpattern	IPPPPPPPPPPP...	Refresh pattern
LP	IPPPPPPPPPPP...	Low-delay P mode
LPI4	IPPPIPPPIPPP...	LP, one I every 4 frames

The 9 encoding modes can be classified into four groups, depending on the whole-frame refreshing rate: (a) LP and IPx modes only have an I frame at the beginning of the sequence, so no refreshing is performed; (b) IPPP, IPx25pctCTU, IPx25pctTIL, IPxpattern, and LPI4 modes have an average frame-refresh rate of one complete intra refreshed frame every four frames; (c) IPIP mode has an intra frame-refresh rate of one frame out of two; (d) AI mode uses a full rate of intra-refreshed frames. For the following tests, we have selected the BasketballDrill sequence (832x480 pixels, 25 fps, bbd_25), which exhibits an average coding performance amongst the 832x480 sequences set. In Figure 2.5, the resulting rate/distortion curves for the 9 encoding modes (for a very wide range of QP values) are plotted. LP mode results the most efficient mode of all, and AI the least efficient. To evaluate the 9 encoding modes in fair conditions, we have selected an individual QP value for every mode so that they generate a bit stream with a similar bit rate. The QP values used for each encoding mode and the bit rate and PSNR values obtained are listed in Table 2.3. Figure 2.6 is a zoomed version of Figure 2.5, which shows the curves for the selected QP values. The curves corresponding to encoding modes belonging to group (b) exhibit very similar coding efficiency. On the contrary, in group (a), LP and IPx curves are not close to each other, and LP clearly outperforms IPx regarding coding efficiency. The PSNR values range from 31.07dB (AI) to 36.77 dB (LP) in a no-loss scenario. The list of the 9 encoding modes, ordered by their coding efficiency, is the following: LP, IPx, LPI4, IPPP, IPxpattern, IPx25pctTIL, IPx25pctCTU, IPIP, and AI.

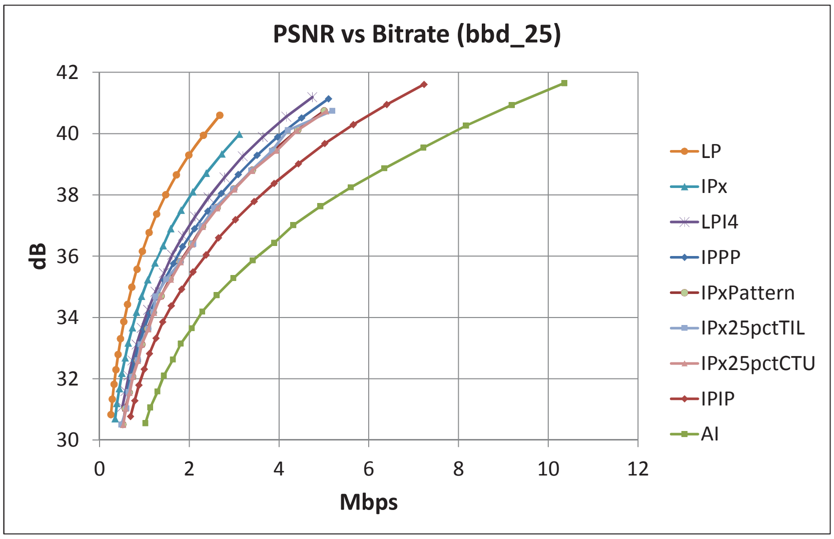


Figure 2.5. Rate/Distortion representation for bbd_25 sequence encoded with all the encoding modes.

2.1.3 Random packet loss

As stated in the introductory chapter, the HEVC reference software decoder crashes when some part of the bit stream is missing. So, in order to evaluate the robustness of protected video bit streams in vehicular networks (which are packet loss prone networks), a modification of the reference software (in order

Table 2.3. Selected values for the QP parameter and bit rate and the PSNR values obtained for each encoding mode.

Encoding mode	QP	Bit Rate (Mbps)	PSNR (dB)
AI	40	1.13	31.07
IPIP	36	1.11	32.82
IPPP	33	1.11	34.13
IPx	29	1.07	35.23
IPx25pctCTU	32	1.08	33.62
IPx25pctTIL	32	1.09	33.64
IPxpattern	32	1.07	33.63
LP	27	1.10	36.77
LPI4	32	1.08	34.24

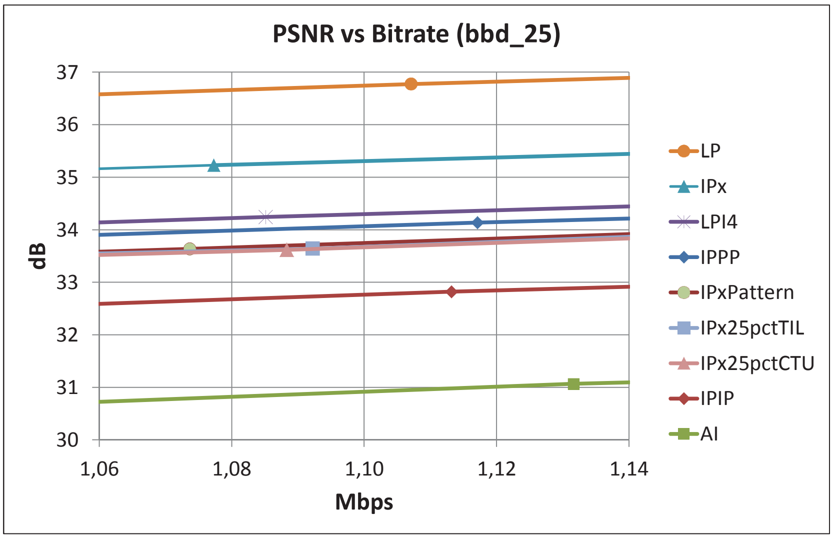


Figure 2.6. Zoom in the range of 1.06 to 1.14 Mbps of Rate/Distortion representation for bbd_25 sequence encoded with all the encoding modes.

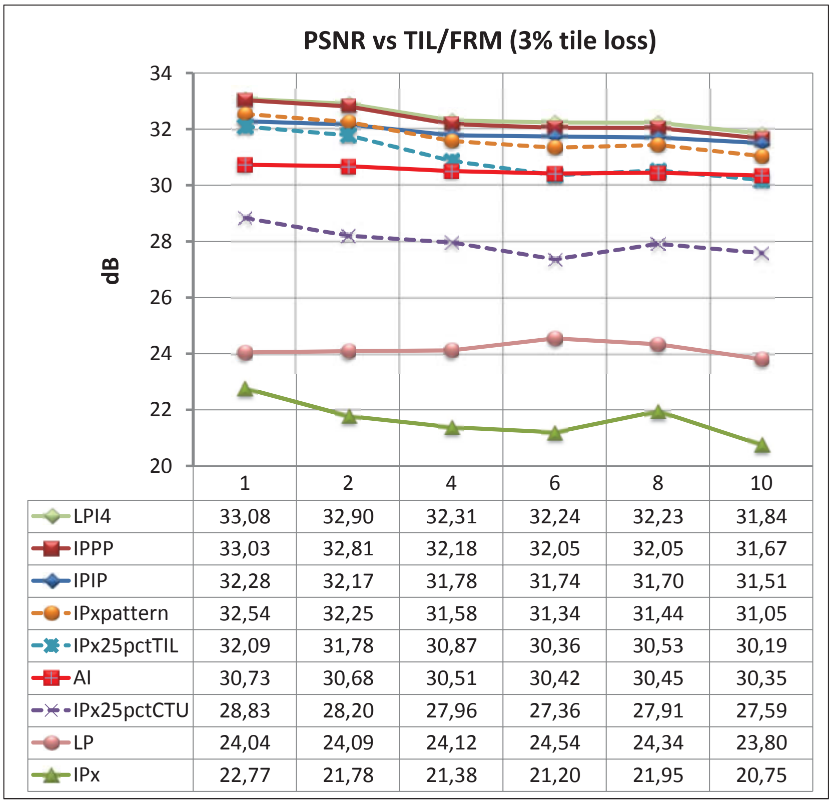
to make it resistant to data loss) became mandatory. Other works emulate the behavior of the decoder under loss conditions by decoding the whole bit stream and then removing some parts of the decoded frames. This is not the real behavior of the decoder, because, when some information is missing, the decoding process does not provide the same results as the "emulated-loss" versions. This one was the first of the modifications that we made to the reference software. The second modification was adding the ability to select a basic error concealment method, mentioned in the introductory chapter: the "zero-MV concealment" method. In this technique, the missing area of a frame is replaced by the co-located area of the previous frame. The third modification was made in the reference software encoder to implement some of the previously presented encoding methods, i.e., IPx25pctCTU, IPx25pctTIL, and IPxpattern.

The performance of the proposed encoding modes and the overhead introduced by tiles and slices have already been evaluated in a no-loss scenario. Now, the performance of the combination of the 9 encoding modes with the different tile partition layouts (1, 2, 4, 6, 8, and 10 tiles/frm), under different percentages of tile loss, will be evaluated. Six different tile loss rates have been selected for the tests: 1%, 3%, 5%, 7%, 10%, and 20%. For every one of these loss rates, 5 different seeds for the random number generator have been used, and the results have been averaged.

Table 2.4. Average difference of the PSNR value obtained with and without the use of the Error Concealment (EC) for different tile loss ratios for every one of the encoding modes averaged over all the tiles per frame layouts. Positive values mean that the EC version is better than the non-EC version. Negative values mean that the non-EC version is better than the EC version.

PSNR gain (dB)	1%	3%	5%	7%	10%	20%
AI	0.14	0.37	0.56	0.74	0.93	1.34
IPIP	0.22	0.59	0.86	1.08	1.35	1.71
IPPP	0.40	0.93	1.24	1.43	1.67	1.74
LPI4	0.48	1.12	1.53	1.71	1.94	1.95
IPxpattern	0.39	0.90	1.23	1.45	1.68	1.71
IPx25pctTIL	0.55	1.12	1.40	1.59	1.69	1.53
Average	**0.36**	**0.84**	**1.14**	**1.33**	**1.54**	**1.66**
IPx25pctCTU(*)	*2.54*	*3.02*	*2.91*	*2.31*	*1.83*	*0.65*
LP(*)	*4.06*	*2.80*	*2.01*	*1.12*	*0.52*	*-0.75*
IPx(*)	*4.96*	*1.74*	*0.36*	*-0.57*	*-1.26*	*-2.27*
Average(*)	***3.85***	***2.52***	***1.76***	***0.95***	***0.36***	***-0.79***

In Table 2.4, the differences in the PSNR value when using the mentioned EC method or not are presented. For every encoding mode and tile loss percentage, we have averaged the PSNR values for all the tile partition layouts (1, 2, 4, 6, 8, and 10 tiles/frm) and for the 5 random number generator seeds. The encoding modes are grouped in two categories: (a) the encoding modes where the difference in the PSNR value increases when the tile loss rate increases, and (b) the encoding modes where the difference in the PSNR value decreases when the tile loss rate increases. In this second category, some values are negative, which means that the non-EC version of the decoded video has better PSNR values than the EC version. Later on, we will see that these three encoding methods provide very low PSNR values in the presence of data loss (over 1%), so the difference in the PSNR values shown in this table do not represent a real improvement or worsening of the video quality. Except for the four negative values (which appear in LP and IPx encoding modes at high tile loss percentages), the rest of the differences in the PSNR value are positive; this means that the EC decoder version provides better PSNR values than the version without EC. In the six encoding methods of the top of the table, the average gains for all of them range from 0.36 dB to 1.66 dB. As a product of these observations, from this point forward we will always use the EC decoder version for the reconstruction of video sequences.

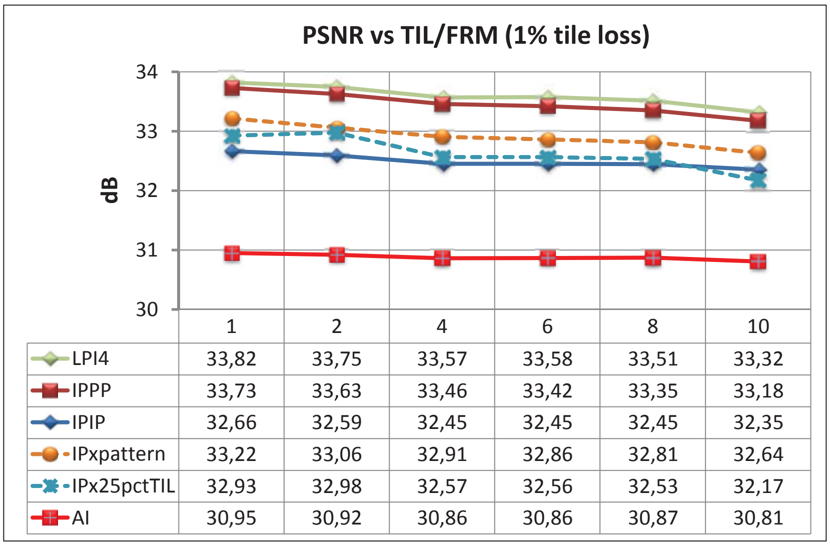


	1	2	4	6	8	10
LPI4	33,08	32,90	32,31	32,24	32,23	31,84
IPPP	33,03	32,81	32,18	32,05	32,05	31,67
IPIP	32,28	32,17	31,78	31,74	31,70	31,51
IPxpattern	32,54	32,25	31,58	31,34	31,44	31,05
IPx25pctTIL	32,09	31,78	30,87	30,36	30,53	30,19
AI	30,73	30,68	30,51	30,42	30,45	30,35
IPx25pctCTU	28,83	28,20	27,96	27,36	27,91	27,59
LP	24,04	24,09	24,12	24,54	24,34	23,80
IPx	22,77	21,78	21,38	21,20	21,95	20,75

Figure 2.7. PSNR for all the til/frm layouts and all the encoding modes with 3% of tile loss ratio.

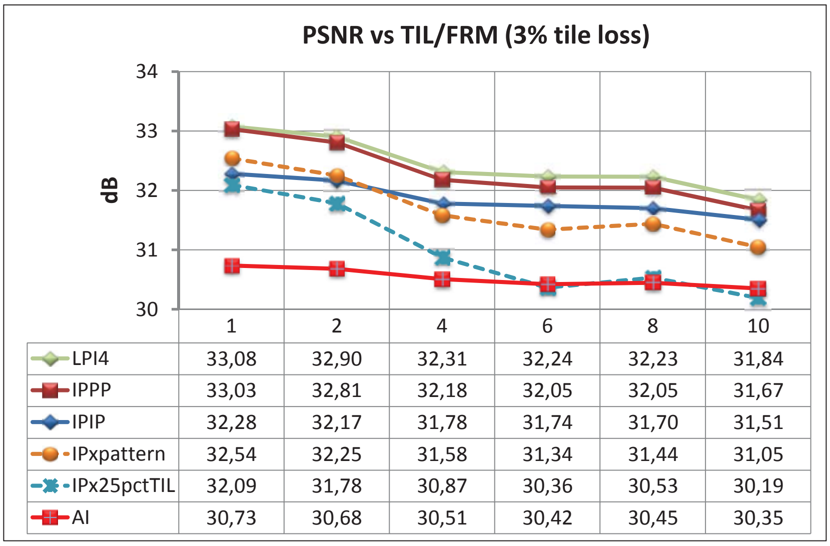
In Figure 2.7, the PSNR values obtained for each one of the encoding modes for a 3% tile loss are shown. Three of the encoding modes show a non-robust response to tile loss (PSNR values under 29 dB are considered low quality values). For two of them, LP and IPx, this is the expected result, as they have no intra refreshing strategy. But the bad performance for IPx25pctCTU was not expected, because in this mode 25% of the CTUs (one out of four) are forced to be intra-coded. Previously, this encoding mode showed the same coding efficiency as IPx25pctTIL and IPxpattern (which share the same percentage of intra refreshed areas), but it seems clear that a random CTU intra refresh provides worse protection against tile loss than a random tile intra refresh or a pattern tile intra refresh. Which is the reason why intra refresh on the CTU level is not 100% effective? The reason is that intra-frame coding uses pixel information (surrounding the CTU) to compute a prediction, which is used both in the encoding process and also in the decoding process. If the pixels used to compute that prediction belong to inter-coded CTUs whose reference frames are corrupted, then the pixels used for the intra-frame prediction are not correct, and, even if the intra CTU is correctly received, it will be incorrectly decoded. If intra refresh is carried out on tile level, as all the CTUs that belong to a tile do not depend on CTUs from outside that tile, then every correctly received intra CTU will be correctly decoded. Intra refreshing on the CTU level does not provide the bit stream with enough robustness.

In figures 2.8, 2.9, and 2.10, LP, IPx, and IPx25pctCTU modes have been removed in order to better compare the rest of the encoding modes. These figures show the PSNR values of the encoding modes when using 1, 2, 4, 6, 8, and 10 tiles per frame for 1%, 3%, 5%, 7%, 10%, and 20% tile loss. One of the consistent results throughout all the figures is that as the number of tiles per frame increases, the quality decreases. The reason is that the loss of small parts in many frames causes a worse effect than the loss of one big part in only one frame, even if the total percentage of the lost data is the same. Of all the modes depicted in these figures, IPx25pctTIL has the worst performance when the number of tiles increases for 3% tile loss and above. For 1% and 3% tile loss, the most robust encoding modes are LPI4 and IPPP. At 5% tile loss, these two modes and IPIP show very similar performance. And at 7% tile loss and above, IPIP is the most resilient mode. At 20% tile loss, AI mode is over IPIP for 4 tiles per frame layout and above, but low PSNR values at these points discard it as a good solution. At first sight, it seems that the 1 tile per frame layout is more robust than the other layouts. But, in the “real world”, tiles are sent inside network packets, and, as stated before, the proportion between tile size and network MTU, and correspondingly the number of network packets needed for each tile, can have an important role in the final quality of the video.



	1	2	4	6	8	10
LPI4	33,82	33,75	33,57	33,58	33,51	33,32
IPPP	33,73	33,63	33,46	33,42	33,35	33,18
IPIP	32,66	32,59	32,45	32,45	32,45	32,35
IPxpattern	33,22	33,06	32,91	32,86	32,81	32,64
IPx25pctTIL	32,93	32,98	32,57	32,56	32,53	32,17
AI	30,95	30,92	30,86	30,86	30,87	30,81

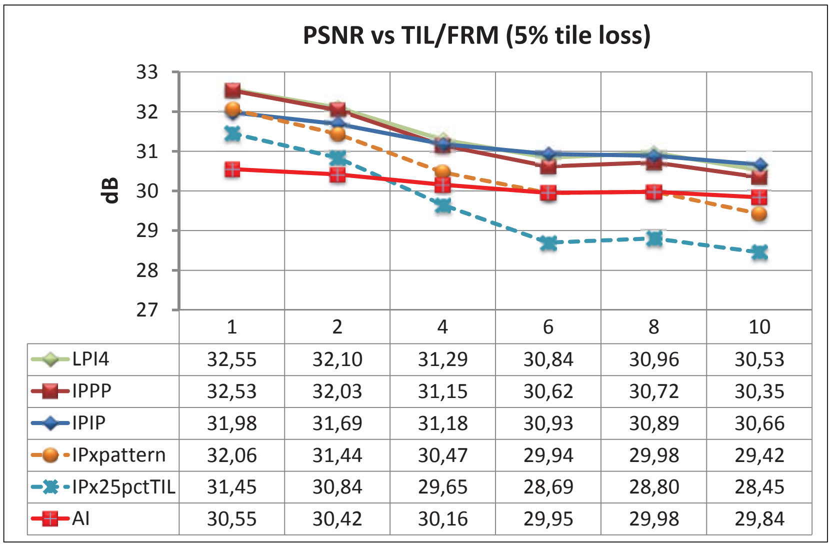
(a) *1% tile loss*



	1	2	4	6	8	10
LPI4	33,08	32,90	32,31	32,24	32,23	31,84
IPPP	33,03	32,81	32,18	32,05	32,05	31,67
IPIP	32,28	32,17	31,78	31,74	31,70	31,51
IPxpattern	32,54	32,25	31,58	31,34	31,44	31,05
IPx25pctTIL	32,09	31,78	30,87	30,36	30,53	30,19
AI	30,73	30,68	30,51	30,42	30,45	30,35

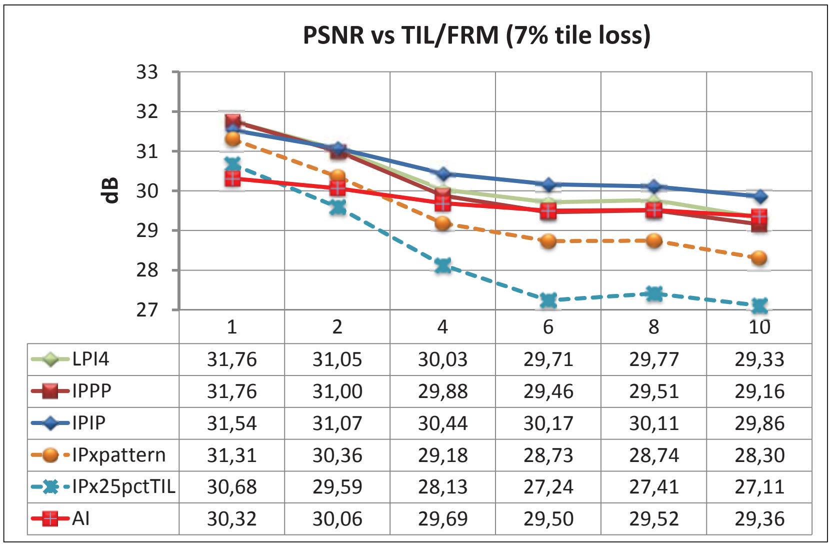
(b) *3% tile loss*

Figure 2.8. PSNR for all the til/frm layouts and all the encoding modes (except LP and IPx) with 1% and 3% of tile loss ratio.



	1	2	4	6	8	10
LPI4	32,55	32,10	31,29	30,84	30,96	30,53
IPPP	32,53	32,03	31,15	30,62	30,72	30,35
IPIP	31,98	31,69	31,18	30,93	30,89	30,66
IPxpattern	32,06	31,44	30,47	29,94	29,98	29,42
IPx25pctTIL	31,45	30,84	29,65	28,69	28,80	28,45
AI	30,55	30,42	30,16	29,95	29,98	29,84

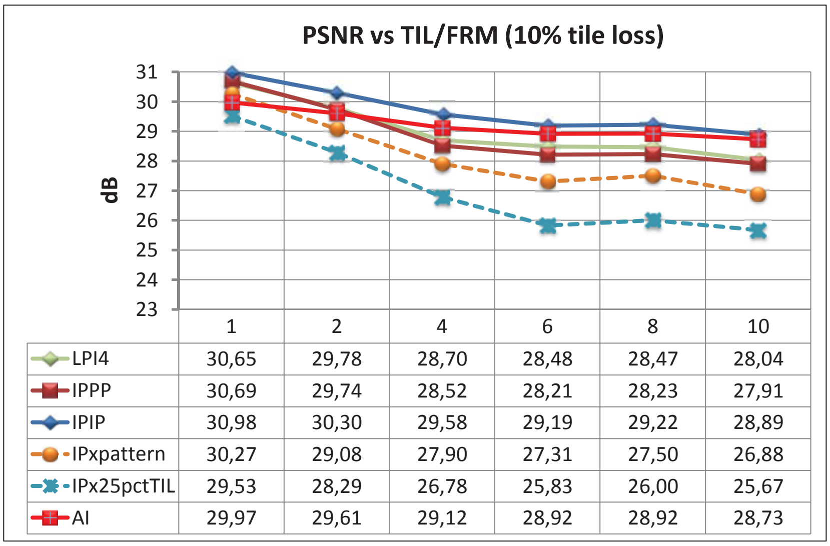
(a) *5% tile loss*



	1	2	4	6	8	10
LPI4	31,76	31,05	30,03	29,71	29,77	29,33
IPPP	31,76	31,00	29,88	29,46	29,51	29,16
IPIP	31,54	31,07	30,44	30,17	30,11	29,86
IPxpattern	31,31	30,36	29,18	28,73	28,74	28,30
IPx25pctTIL	30,68	29,59	28,13	27,24	27,41	27,11
AI	30,32	30,06	29,69	29,50	29,52	29,36

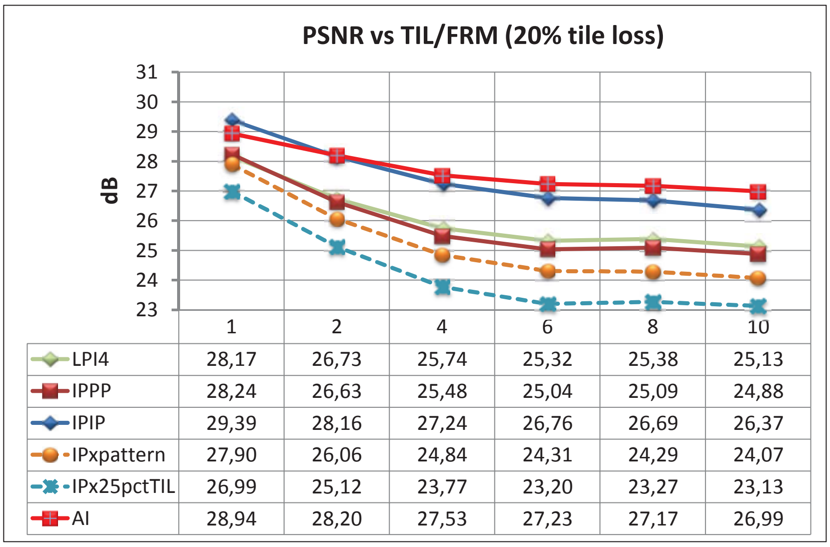
(b) *7% tile loss*

Figure 2.9. PSNR for all the til/frm layouts and all the encoding modes (except LP and IPx) with 5% and 7% of tile loss ratio.



	1	2	4	6	8	10
LPI4	30,65	29,78	28,70	28,48	28,47	28,04
IPPP	30,69	29,74	28,52	28,21	28,23	27,91
IPIP	30,98	30,30	29,58	29,19	29,22	28,89
IPxpattern	30,27	29,08	27,90	27,31	27,50	26,88
IPx25pctTIL	29,53	28,29	26,78	25,83	26,00	25,67
AI	29,97	29,61	29,12	28,92	28,92	28,73

(a) *10% tile loss*



	1	2	4	6	8	10
LPI4	28,17	26,73	25,74	25,32	25,38	25,13
IPPP	28,24	26,63	25,48	25,04	25,09	24,88
IPIP	29,39	28,16	27,24	26,76	26,69	26,37
IPxpattern	27,90	26,06	24,84	24,31	24,29	24,07
IPx25pctTIL	26,99	25,12	23,77	23,20	23,27	23,13
AI	28,94	28,20	27,53	27,23	27,17	26,99

(b) *20% tile loss*

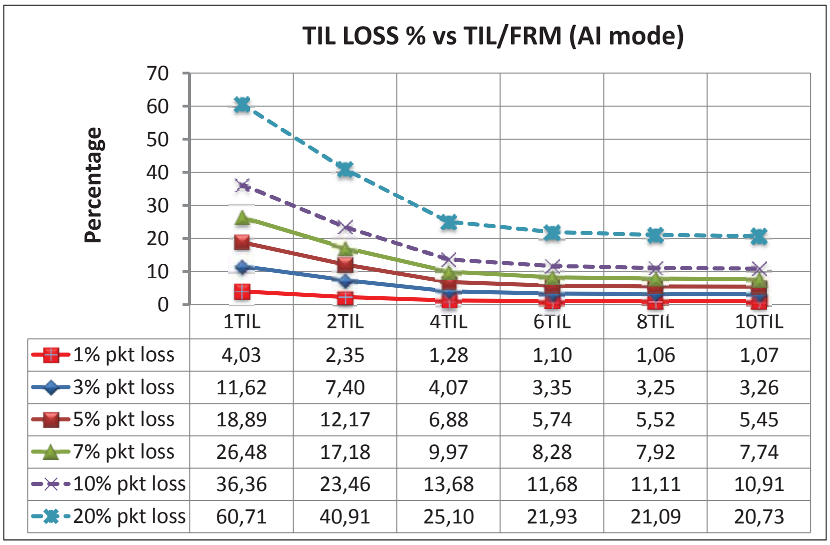
Figure 2.10. PSNR for all the til/frm layouts and all the encoding modes (except LP and IPx) with 10% and 20% of tile loss ratio.

Table 2.5. Average number of Packets Per Tile for every encoding mode.

Packets Per Tile	1 til	2 til	4 til	6 til	8 til	10 til
AI	4.38	2.52	1.37	1.17	1.12	1.09
IPIP	4.37	2.35	1.57	1.27	1.14	1.10
IPPP	4.40	2.46	1.49	1.27	1.17	1.10
IPx	4.18	2.38	1.41	1.14	1.05	1.02
IPx25pctCTU	4.27	2.41	1.41	1.13	1.04	1.02
IPx25pctTIL	4.26	2.42	1.43	1.20	1.08	1.07
IPxpattern	4.19	2.39	1.42	1.19	1.09	1.06
LP	4.30	2.43	1.45	1.20	1.10	1.06
LPI4	4.34	2.35	1.59	1.35	1.23	1.14

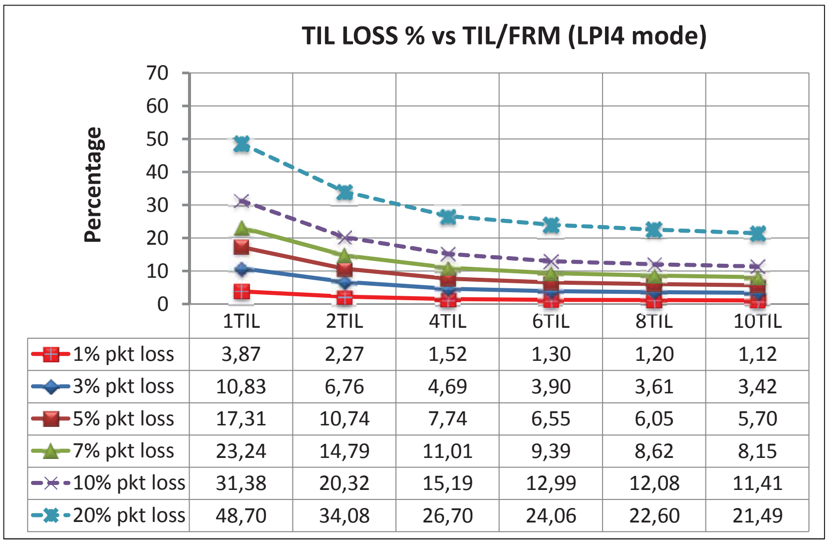
In Table 2.5, the average number of packets per tile is shown. For the 1 tile per frame layout, the proportion is over 4 packets per tile. This means that the loss of one network packet will probably entail the effective loss of four packets. For the 10 tiles per frame layout, the proportion is near 1 packet per tile. This means that the loss of one network packet will probably entail the effective loss of only that packet. To analyze the relation between packet loss percentage and tile loss percentage, Figure 2.11 shows this proportion for every one of the layouts tested, for both AI and LPI4 modes. It is clearly visible that the 1 tile per frame layout suffers from vulnerability in both modes, and the induced tile loss percentage at each packet loss percentage curve is closely related to the values in Table 2.5, so when the number of packets per frame tends to one the percentage of tile loss obtains its minimum value. Dividing a frame into 6, 8, or 10 tiles per frame produces similar tile loss rates, because the proportions of packets per tile for these three modes are close.

In Figure 2.12, the PSNR values for each tile per frame layout for all the encoding modes at 3% packet loss, is shown. Again, LP, IPx, and IPx25pctCTU obtain very low PSNR values, showing little resilience against errors. The other six modes have increasing PSNR values when the number of tiles per frame increases. In figures 2.13, 2.14, and 2.15, as in previous figures, the three non-robust encoding modes have been discarded for the sake of clarity. In these figures, the PSNR values are plotted for different percentages of packet loss. When a high number of tiles per frame is selected, the recovered video sequence obtains a better PSNR value. This is the opposite result to the one observed when PSNR was plotted *versus* TIL/FRM for different tile loss percentages (figures 2.8, 2.9, and 2.10). For moderate loss (1%), IPPP and LPI4 are still the best encoding methods. For a medium packet loss percentage (3%), IPPP



	1TIL	2TIL	4TIL	6TIL	8TIL	10TIL
1% pkt loss	4,03	2,35	1,28	1,10	1,06	1,07
3% pkt loss	11,62	7,40	4,07	3,35	3,25	3,26
5% pkt loss	18,89	12,17	6,88	5,74	5,52	5,45
7% pkt loss	26,48	17,18	9,97	8,28	7,92	7,74
10% pkt loss	36,36	23,46	13,68	11,68	11,11	10,91
20% pkt loss	60,71	40,91	25,10	21,93	21,09	20,73

(a) *AI mode*

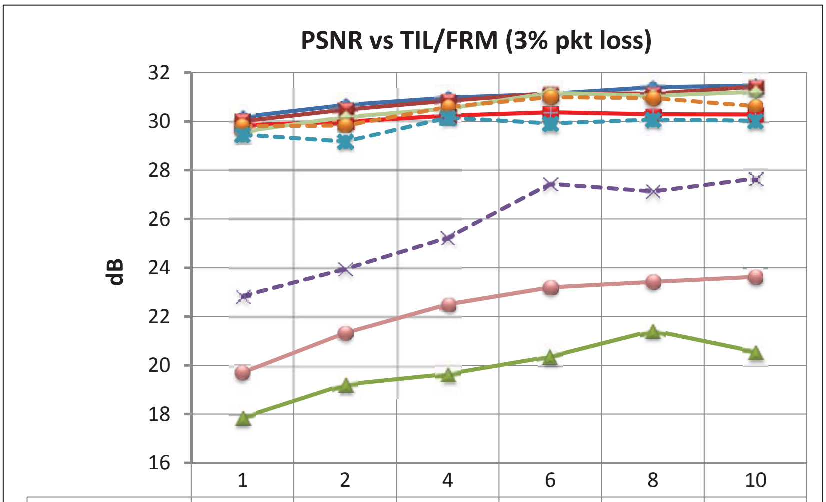


	1TIL	2TIL	4TIL	6TIL	8TIL	10TIL
1% pkt loss	3,87	2,27	1,52	1,30	1,20	1,12
3% pkt loss	10,83	6,76	4,69	3,90	3,61	3,42
5% pkt loss	17,31	10,74	7,74	6,55	6,05	5,70
7% pkt loss	23,24	14,79	11,01	9,39	8,62	8,15
10% pkt loss	31,38	20,32	15,19	12,99	12,08	11,41
20% pkt loss	48,70	34,08	26,70	24,06	22,60	21,49

(b) *LPI4 mode*

Figure 2.11. Tile loss percentage for every packet loss percentage for all the til/frm layouts and for the AI and LPI4 encoding modes. Non-FEC bit streams.

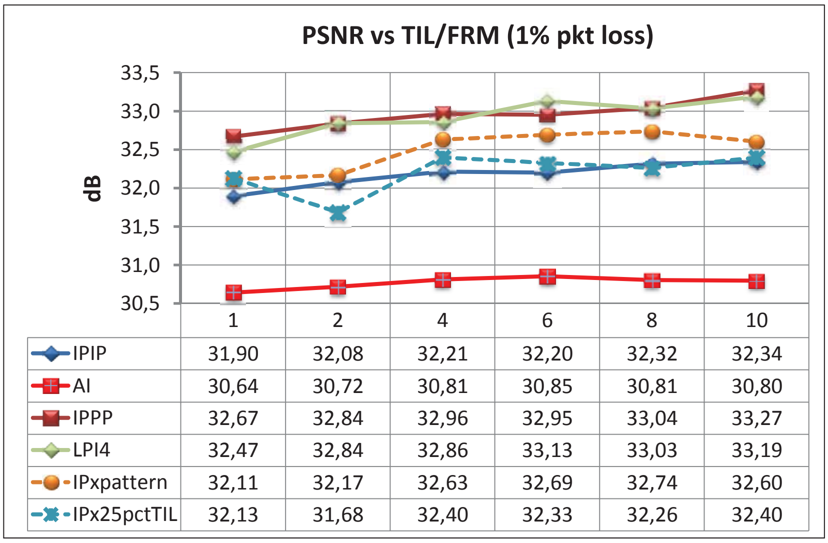
and IPIP are the best methods. For the rest (5%, 7%, 10%, and 20%), the best methods are AI and IPIP, although for values of 7% and higher, the PSNR values are very low.



	1	2	4	6	8	10
IPIP	30,19	30,67	30,98	31,13	31,40	31,47
AI	29,85	29,99	30,24	30,37	30,29	30,28
IPPP	30,01	30,48	30,85	31,15	31,13	31,43
LPI4	29,58	30,18	30,53	31,17	31,06	31,21
IPxpattern	29,82	29,83	30,59	30,99	30,96	30,63
IPx25pctTIL	29,46	29,17	30,15	29,92	30,07	30,03
IPx25pctCTU	22,84	23,95	25,24	27,44	27,13	27,65
LP	19,71	21,33	22,50	23,19	23,42	23,63
IPx	17,86	19,22	19,65	20,35	21,43	20,55

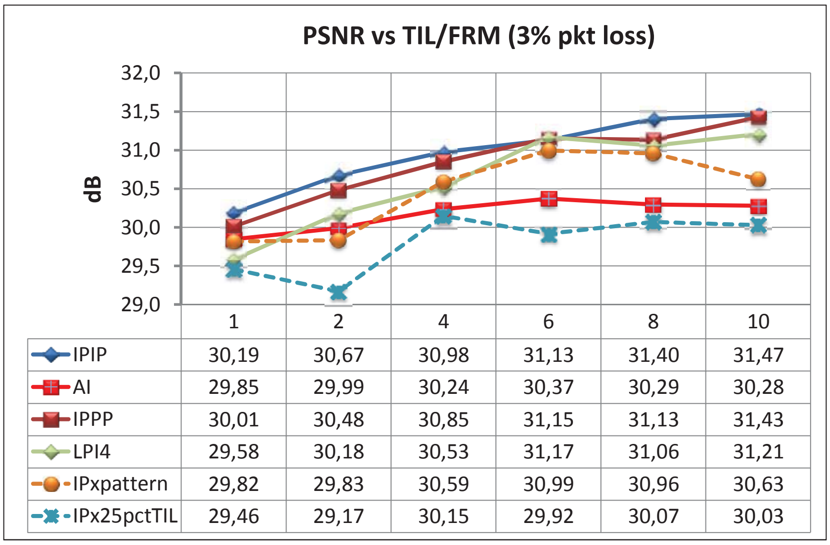
Figure 2.12. PSNR for all the til/frm layouts and all the encoding modes with 3% of packet loss ratio. Non-FEC bit streams.

From the evaluations of this section, some conclusions can be drawn. Of the 9 encoding modes evaluated, there are three that do not have good error resilience properties. Two of them (LP, IPx) have provided the expected performance results as they do not use any intra refreshing at all. But the third one (IPx25pctCTU), which refreshes one of every four CTUs, does not have the expected intra-refresh effect in the bit stream. Of the rest of the modes, two of them (LPI4 and IPPP) stand out with respect to the others when the percentage of loss (both tile loss and packet loss) remains low. And another method (IPIP) seems to have good performance when the percentage of loss increases. Although some of the methods exhibit good error resilience properties, this is not enough to guarantee robust video streaming. In the next section, we will evaluate RaptorQ codes, and will search for the best setup that can provide the best protection for the specific data packets of video bit streams.



	1	2	4	6	8	10
IPIP	31,90	32,08	32,21	32,20	32,32	32,34
AI	30,64	30,72	30,81	30,85	30,81	30,80
IPPP	32,67	32,84	32,96	32,95	33,04	33,27
LPI4	32,47	32,84	32,86	33,13	33,03	33,19
IPxpattern	32,11	32,17	32,63	32,69	32,74	32,60
IPx25pctTIL	32,13	31,68	32,40	32,33	32,26	32,40

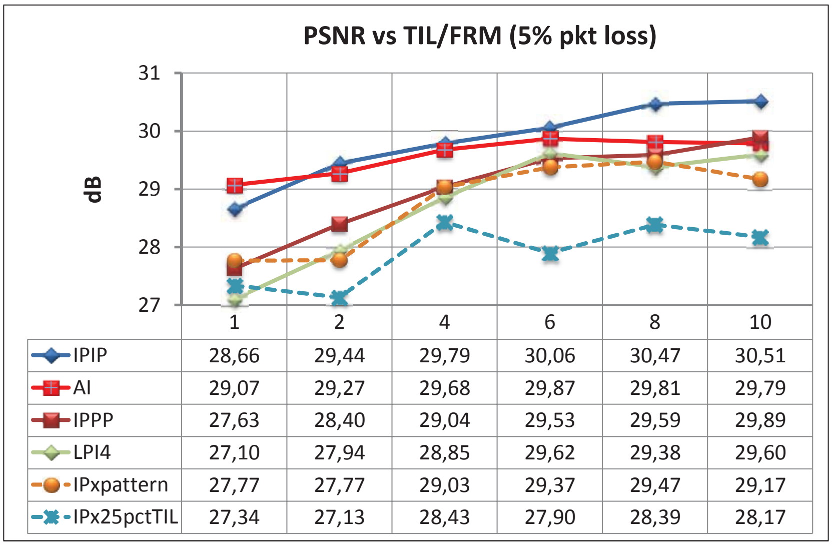
(a) *1% packet loss*



	1	2	4	6	8	10
IPIP	30,19	30,67	30,98	31,13	31,40	31,47
AI	29,85	29,99	30,24	30,37	30,29	30,28
IPPP	30,01	30,48	30,85	31,15	31,13	31,43
LPI4	29,58	30,18	30,53	31,17	31,06	31,21
IPxpattern	29,82	29,83	30,59	30,99	30,96	30,63
IPx25pctTIL	29,46	29,17	30,15	29,92	30,07	30,03

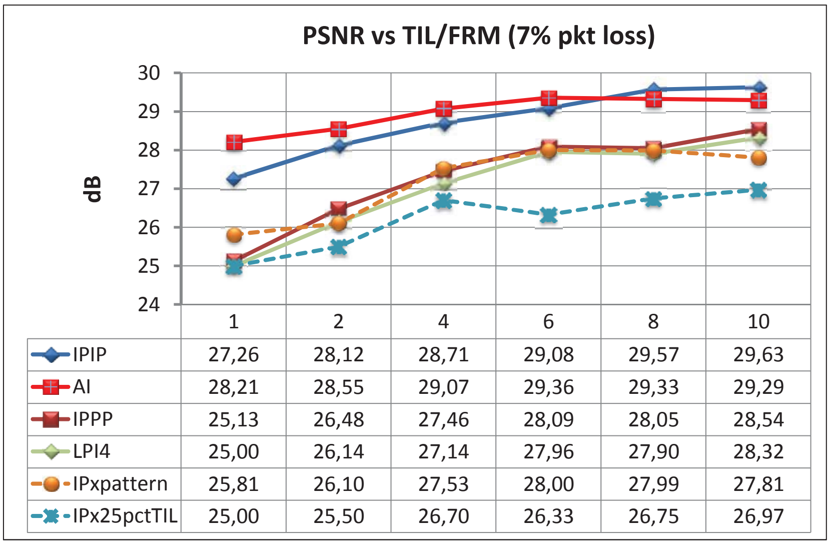
(b) *3% packet loss*

Figure 2.13. PSNR for all the til/frm layouts and all the encoding modes (except LP and IPx) with 1% and 3% of packet loss ratio. Non-FEC bit streams.



	1	2	4	6	8	10
IPIP	28,66	29,44	29,79	30,06	30,47	30,51
AI	29,07	29,27	29,68	29,87	29,81	29,79
IPPP	27,63	28,40	29,04	29,53	29,59	29,89
LPI4	27,10	27,94	28,85	29,62	29,38	29,60
IPxpattern	27,77	27,77	29,03	29,37	29,47	29,17
IPx25pctTIL	27,34	27,13	28,43	27,90	28,39	28,17

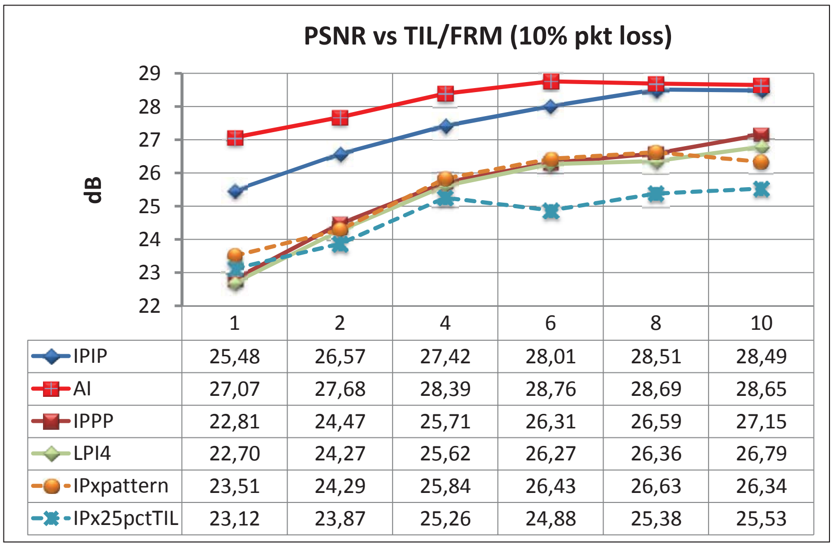
(a) *5% packet loss*



	1	2	4	6	8	10
IPIP	27,26	28,12	28,71	29,08	29,57	29,63
AI	28,21	28,55	29,07	29,36	29,33	29,29
IPPP	25,13	26,48	27,46	28,09	28,05	28,54
LPI4	25,00	26,14	27,14	27,96	27,90	28,32
IPxpattern	25,81	26,10	27,53	28,00	27,99	27,81
IPx25pctTIL	25,00	25,50	26,70	26,33	26,75	26,97

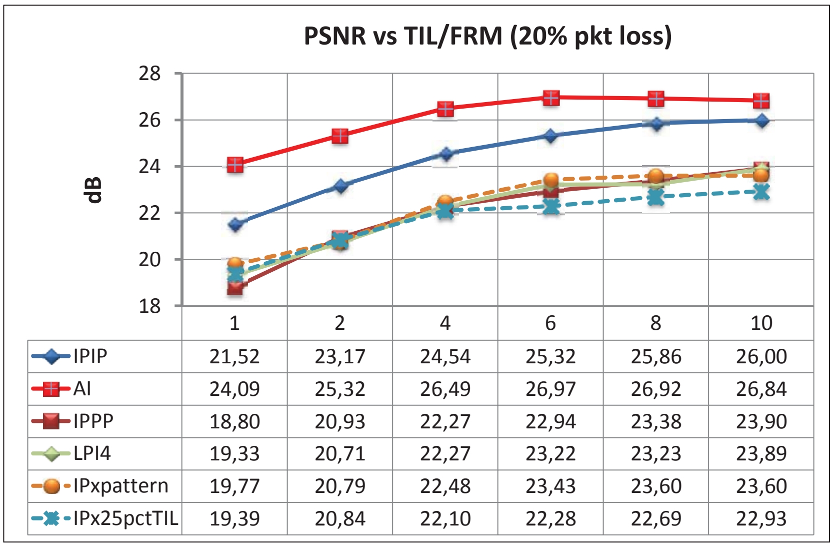
(b) *7% packet loss*

Figure 2.14. PSNR for all the til/frm layouts and all the encoding modes (except LP and IPx) with 5% and 7% of packet loss ratio. Non-FEC bit streams.



	1	2	4	6	8	10
IPIP	25,48	26,57	27,42	28,01	28,51	28,49
AI	27,07	27,68	28,39	28,76	28,69	28,65
IPPP	22,81	24,47	25,71	26,31	26,59	27,15
LPI4	22,70	24,27	25,62	26,27	26,36	26,79
IPxpattern	23,51	24,29	25,84	26,43	26,63	26,34
IPx25pctTIL	23,12	23,87	25,26	24,88	25,38	25,53

(a) *10% packet loss*



	1	2	4	6	8	10
IPIP	21,52	23,17	24,54	25,32	25,86	26,00
AI	24,09	25,32	26,49	26,97	26,92	26,84
IPPP	18,80	20,93	22,27	22,94	23,38	23,90
LPI4	19,33	20,71	22,27	23,22	23,23	23,89
IPxpattern	19,77	20,79	22,48	23,43	23,60	23,60
IPx25pctTIL	19,39	20,84	22,10	22,28	22,69	22,93

(b) *20% packet loss*

Figure 2.15. PSNR for all the til/frm layouts and all the encoding modes (except LP and IPx) with 10% and 20% of packet loss ratio. Non-FEC bit streams.

2.2 RaptorQ protection

RaptorQ codes can provide AL-FEC protection to video streaming. The primary benefit of using AL-FEC in streaming is to improve video quality by trying to recover lost packets. The cost of using FEC is additional network bandwidth to carry FEC repair data, and increased latency to support packet recovery from a block of packets. Increasing the FEC bandwidth (given a fixed FEC latency) leads to an increase in data protection. Similarly, increasing the FEC latency (given fixed FEC bandwidth) leads to an increase in data protection.

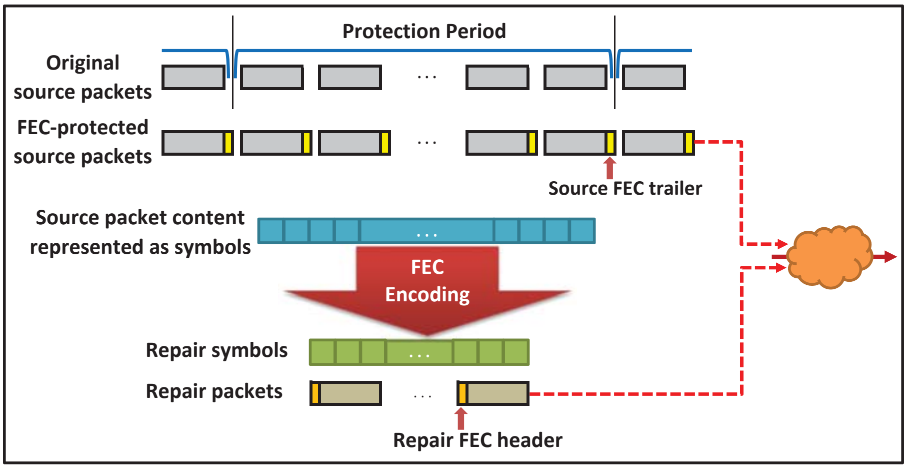


Figure 2.16. RaptorQ codes encoding process.

Figure 2.16 shows the way in which the RaptorQ encoder protects data. First of all, the RaptorQ encoder receives a data stream (*original source packets*) during a specified *protection period*. A 4-byte *FEC-trailer* is added to each received packet thus forming an *FEC-protected source packet stream*. This trailer identifies the packet and the protection period to which it belongs. When an FEC-protected packet is generated, it is then immediately sent through the network. The packet payloads for all source packets within the protection period are gathered and represented as a set of continuous *RaptorQ source symbols*. When the protection period finishes, these source symbols are used to compute *repair symbols* for that protection period by the *FEC encoding* process. The repair symbols are the same size as the source symbols and are carried in *repair packet* payloads. Each repair packet includes a 6-byte *FEC header* that prepends one or more FEC repair symbols. The repair FEC header identifies the associated protection period, the first symbol in the repair packet, and the number of source symbols in the protection period. Both the original source packets (with the FEC-trailer) and the associated repair packets are transmitted within the same protection period, and are used by a receiver application in the recovery process. Within each protection period, a receiver must receive "enough" source and repair symbols (from the FEC-protected source and repair packets) to be able to recover any dropped packets.

To specify the amount of bandwidth used to protect a stream, the "protection amount" parameter refers to the percentage of repair data symbols to be used relative to the quantity of source stream data. Because RaptorQ only generates whole symbols, the actual repair packet stream rate may not be precisely the specified protection amount for each protection period. Application-layer FEC codecs operate on discrete pieces of data called "symbols", which are derived from the original source data. The FEC symbol size is an important design parameter for streaming. The recommended method for choosing an appropriate symbol size is to first understand performance-related aspects that can be affected by symbol size. The streaming transmitter and streaming receiver must be configured with an identical symbol size, so evaluating the trade-offs for the selection of the symbol size is needed. Another consideration is the mapping of symbols to packets. In addition to the protection amount and the symbol size, there is another important design parameter to select: the protection period (or temporal window). As stated before, increasing the protection period leads to an increase in data protection, but also an increase in latency. Depending on the video streaming application, the amount of acceptable latency may vary. In the next section, the selection of the most appropriate RaptorQ design parameters for video streaming will be made.

2.2.1 RaptorQ setups

For the protection of the encoded bit streams, we have used the Qualcomm (R) RaptorQ (TM) Evaluation Kit [57]. For each one of the encoded bit streams (`bbd_25` sequence, using 9 encoding modes, with 6 til/frm layouts), we have generated several protected versions by combining different values for the parameters cited before: *protection amount*, *protection period*, and *symbol size*. The properties of the protected versions will determine the most suitable configurations for the protection of the video bit streams. Three different levels of protection amount have been tested: 10%, 20%, and 30%; seven protection periods have been used: 133, 166, 200, 250, 333, 500, and 1000 milliseconds; and three symbol sizes have been evaluated: 192, 450, and 1350 bytes. For each one of the symbol sizes, a different packaging scheme has been used to optimize the repair packet size. For a symbol size of 192 bytes, we have selected a scheme of 7 symbols per packet; for a symbol size of 450 bytes, we have selected a scheme of 3 symbols per packet; and for a symbol size of 1350, we have selected a scheme of 1 symbol per packet.

Table 2.6. Overhead for a protection amount of 10% for 1 til/frm and 10 til/frm layouts and different protection periods and protection symbol sizes.

10% prot.	1 til/frm			10 til/frm		
overhead	192 B	450 B	1350 B	192 B	450 B	1350 B
133 ms	11.94%	14.61%	23.49%	13.21%	16.78%	31.17%
166 ms	11.76%	14.37%	23.27%	13.17%	16.50%	30.82%
200 ms	11.67%	14.31%	22.89%	12.99%	16.40%	30.42%
250 ms	11.58%	14.07%	22.06%	12.91%	16.16%	29.43%
333 ms	11.46%	13.85%	21.55%	12.97%	15.97%	29.03%
500 ms	11.55%	13.80%	21.49%	12.78%	15.82%	28.36%
1000 ms	11.50%	13.58%	20.79%	12.50%	15.43%	27.52%

Table 2.7. Overhead for a protection amount of 20% for 1 til/frm and 10 til/frm layouts and different protection periods and protection symbol sizes.

20% prot.	1 til/frm			10 til/frm		
overhead	192 B	450 B	1350 B	192 B	450 B	1350 B
133 ms	22.59%	27.04%	42.23%	24.98%	31.32%	57.74%
166 ms	22.56%	27.27%	42.76%	24.93%	31.15%	57.39%
200 ms	22.58%	27.21%	43.11%	24.89%	30.99%	57.08%
250 ms	22.42%	26.77%	41.64%	24.81%	30.74%	56.09%
333 ms	22.35%	26.49%	40.96%	24.57%	30.45%	55.44%
500 ms	22.42%	26.53%	40.80%	24.53%	30.31%	55.00%
1000 ms	22.33%	26.39%	40.38%	24.18%	29.61%	53.58%

In tables 2.6, 2.7, and 2.8, the overhead values of the protected bit stream versions over the non-FEC version are shown for a protection amount of 10%, 20%, and 30%, respectively. In these tables, the overhead percentage is displayed for 1 til/frm and 10 til/frm layouts for the 3 symbol sizes used and for the 7 protection periods tested. Some conclusions can be drawn from the observation of these three tables. First of all, the smaller symbol size (192 bytes) obtains the lowest overhead values. This is due to the nature of the source data. As the source packets payloads have variable sizes and they range from small to big packets, a small symbol size optimizes the size of the repair packets. Instead, if we were protecting some other type of data with a constant

Table 2.8. Overhead for a protection amount of 30% for 1 til/frm and 10 til/frm layouts and different protection periods and protection symbol sizes.

30% prot.	1 til/frm			10 til/frm		
overhead	192 B	450 B	1350 B	192 B	450 B	1350 B
133 ms	33.52%	40.34%	62.90%	37.10%	46.14%	85.38%
166 ms	33.37%	39.99%	62.42%	36.80%	45.94%	84.47%
200 ms	33.32%	39.72%	62.42%	36.75%	45.73%	84.24%
250 ms	33.37%	39.69%	61.96%	36.52%	45.42%	82.98%
333 ms	33.21%	39.60%	61.28%	36.17%	44.94%	81.87%
500 ms	33.17%	39.51%	61.04%	36.05%	44.64%	81.34%
1000 ms	32.96%	38.99%	59.86%	35.71%	43.89%	79.65%

size payload, then a symbol size matching the payload size would be the most efficient. Another observed feature is that the 1 til/frm layout generates less overhead than the 10 til/frm layout. The overhead values for the rest of the layouts have not been included in these tables for the sake of concision and clarity, but they show monotonically increasing behavior: the higher the number of tiles per frame, the higher the overhead introduced by FEC protection. For a fixed symbol size and a fixed til/frm layout, the overhead values for the different protection periods show monotonically decreasing behavior: the wider the temporal window, the lower the overhead percentage. As the size of the protection period has a direct effect on latency, the selection of the proper temporal window seems clear: the highest acceptable latency will determine the most efficient protection period. In this work, as a "design decision", we have selected a protection period of 333 ms. Depending on the particular ITS application which uses video streaming, the selection of the protection window can vary if the delay requirements are stricter or more tolerant. None of the values for each one of the tables matches exactly with the protection amount value of that table. In Table 2.6, the lowest overhead value is 11.50%; in Table 2.7, the lowest overhead value is 22.33%; and in Table 2.8, the lowest overhead value is 32.96%. As the total payload size of the source packets belonging to a protection period does not always match a multiple of the symbol size, there is some extra overhead due to padding. Also, the FEC-trailer of each protected source packet and the FEC-header of each repair packet add extra overhead to the protected bit stream. A summarized version of the overhead differences produced by using symbol sizes of 450 and 1350 bytes over using a symbol size of 192 bytes are shown in Table 2.9. The overhead differences are noticeable, especially in the 1350 byte case.

Table 2.9. Overhead difference for using a symbol size of 450 bytes and 1350 bytes compared with using a symbol size of 192 bytes.

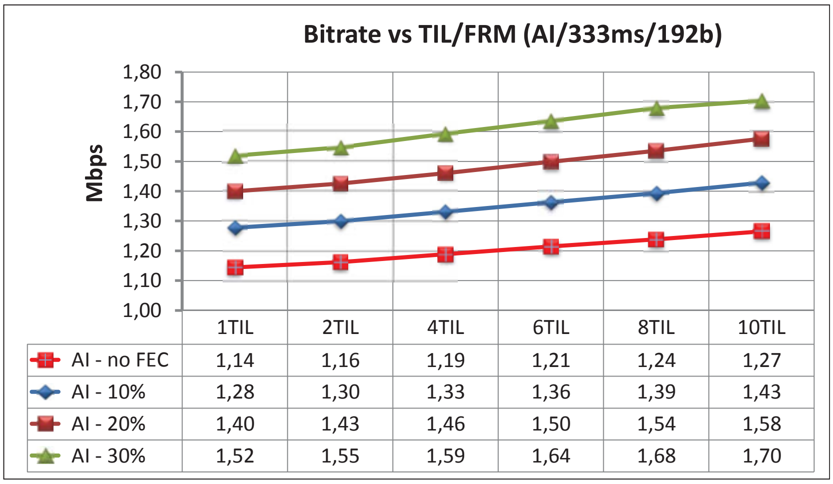
Ovhd.	10% prot.		20% prot.		30% prot.	
diff.	1 til	10 til	1 til	10 til	1 til	10 til
450 B	2.56%	3.31%	4.45%	6.10%	6.51%	8.96%
1350 B	10.97%	17.13%	19.64%	31.91%	28.84%	47.12%

In summary, the parameters selected for the protection of the video bit streams are a symbol size of 192 bytes because of efficiency (lowest overhead) and a protection period of 333 ms (maximum latency allowed, as a design decision). We will maintain the three values for the protection amount parameter in order to test their performance against packet loss. In Figure 2.17, the resulting bit rates of the protected (10%, 20%, and 30%) and non-protected versions of the bit streams encoded with the AI and LPI4 encoding modes for each one of the til/frm layouts are shown. In the non-FEC version, the bit rate increases as the number of tiles per frame increases. In the AI encoding mode the 10 til/frm layout has an overhead of 11.40% over the 1 til/frm layout. In the LPI4 encoding mode the 10 til/frm layout has an overhead of 9.09% over the 1 til/frm layout. The protected versions show similar behavior. The bit rate increases as the number of tiles per frame increases. The highest overhead difference is obtained comparing the bit rate of the non-protected 1 til/frm version and the bit rate of the protected bit stream with a protection amount of 30% and 10 tiles per frame. In both the AI and LPI4 encoding modes, this maximum overhead is around 49%. An excessive increase in bandwidth requirements for the protected bit stream may have the opposite effect: it may increase the percentage of packet loss and thus obtain a less robust bit stream.

2.2.2 Packet recovery

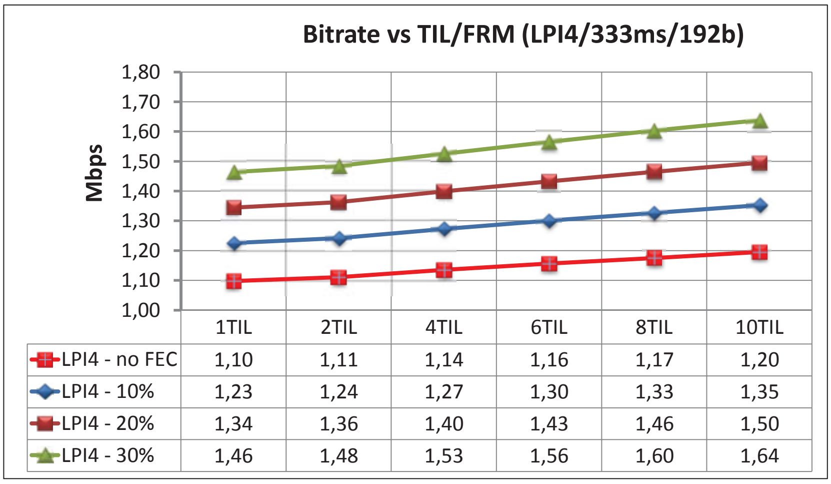
In order to evaluate the RaptorQ codes performance in the recovery of lost packets, we have followed the same procedure as in Section 2.1.3. Six different packet loss rates have been selected for the tests: 1%, 3%, 5%, 7%, 10%, and 20%, and for every one of these loss rates, 5 different seeds for the random number generator have been used, whose results have been averaged.

Although in the previous section we set a protection period of 333 ms because it obtains the most efficient performance for the highest latency allowed, we want to check if this selection is also efficient regarding data



	1TIL	2TIL	4TIL	6TIL	8TIL	10TIL
AI - no FEC	1,14	1,16	1,19	1,21	1,24	1,27
AI - 10%	1,28	1,30	1,33	1,36	1,39	1,43
AI - 20%	1,40	1,43	1,46	1,50	1,54	1,58
AI - 30%	1,52	1,55	1,59	1,64	1,68	1,70

(a) *AI mode*

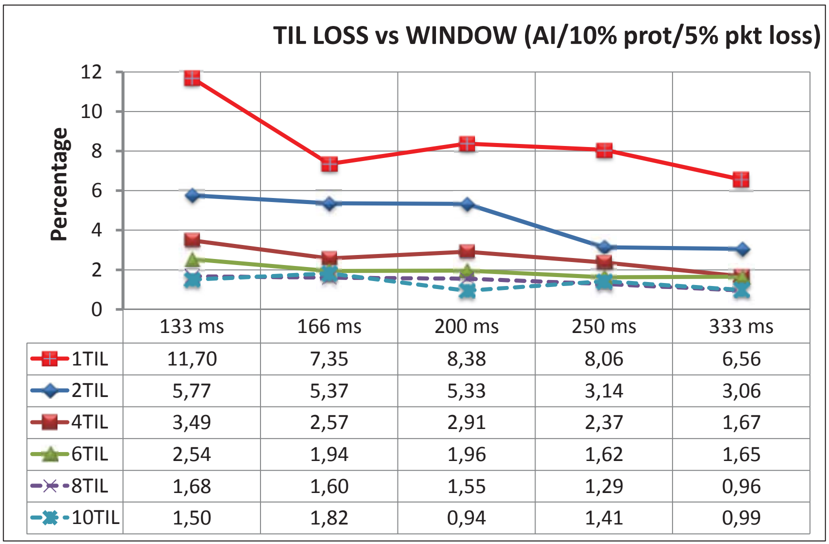


	1TIL	2TIL	4TIL	6TIL	8TIL	10TIL
LPI4 - no FEC	1,10	1,11	1,14	1,16	1,17	1,20
LPI4 - 10%	1,23	1,24	1,27	1,30	1,33	1,35
LPI4 - 20%	1,34	1,36	1,40	1,43	1,46	1,50
LPI4 - 30%	1,46	1,48	1,53	1,56	1,60	1,64

(b) *LPI4 mode*

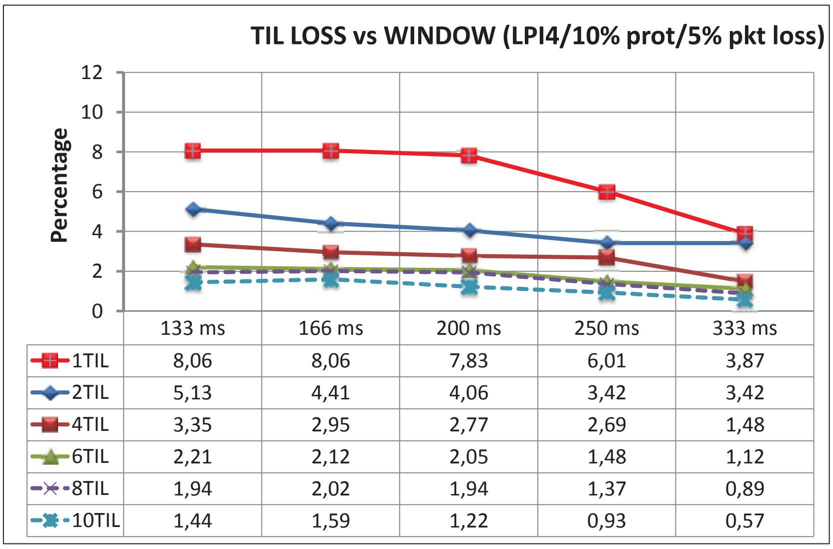
Figure 2.17. Resulting bit rate (Mbps) for different til/frm layouts for the AI and LPI4 encoding modes. Non-FEC bit streams and FEC-protected bit streams with a protection amount of 10%, 20%, and 30% using a protection period of 333 ms and a repair symbol size of 192 bytes.

protection. Figure 2.18 shows the tile loss percentage (after FEC recovery) for different temporal windows with a protection amount of 10% and a packet loss of 5%, for both the AI and LPI4 modes and for the six til/frm layouts defined. The 333 ms temporal window clearly outperforms the rest of the protection periods. Only a pair of values of other temporal windows have a lower percentage of tile loss, but they are negligible in practice (0.05% for 200 ms at



	133 ms	166 ms	200 ms	250 ms	333 ms
1TIL	11,70	7,35	8,38	8,06	6,56
2TIL	5,77	5,37	5,33	3,14	3,06
4TIL	3,49	2,57	2,91	2,37	1,67
6TIL	2,54	1,94	1,96	1,62	1,65
8TIL	1,68	1,60	1,55	1,29	0,96
10TIL	1,50	1,82	0,94	1,41	0,99

(a) *AI mode*

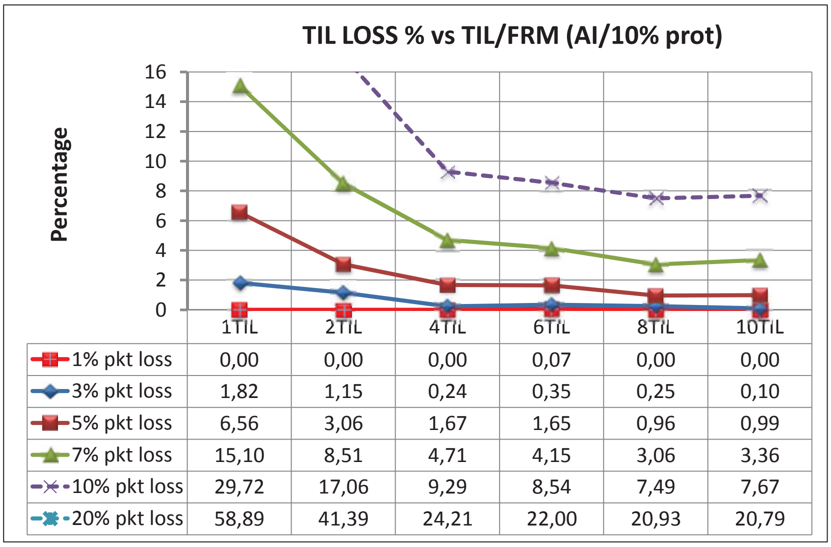


	133 ms	166 ms	200 ms	250 ms	333 ms
1TIL	8,06	8,06	7,83	6,01	3,87
2TIL	5,13	4,41	4,06	3,42	3,42
4TIL	3,35	2,95	2,77	2,69	1,48
6TIL	2,21	2,12	2,05	1,48	1,12
8TIL	1,94	2,02	1,94	1,37	0,89
10TIL	1,44	1,59	1,22	0,93	0,57

(b) *LPI4 mode*

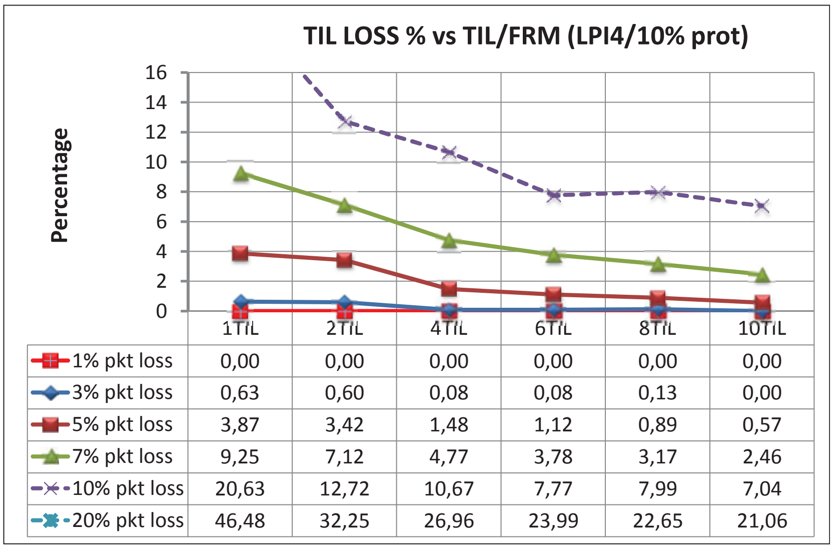
Figure 2.18. Tile loss percentage for different protection periods (windows) for all the til/frm layouts and for the AI and LPI4 encoding modes with 5% of packet loss ratio. FEC-protected bit streams with a protection amount of 10%.

10 til/frm, and 0.03% for 250 ms at 6 til/frm, both in the AI mode). Consequently, the selection of the 333 ms protection period produces both the



	1TIL	2TIL	4TIL	6TIL	8TIL	10TIL
1% pkt loss	0,00	0,00	0,00	0,07	0,00	0,00
3% pkt loss	1,82	1,15	0,24	0,35	0,25	0,10
5% pkt loss	6,56	3,06	1,67	1,65	0,96	0,99
7% pkt loss	15,10	8,51	4,71	4,15	3,06	3,36
10% pkt loss	29,72	17,06	9,29	8,54	7,49	7,67
20% pkt loss	58,89	41,39	24,21	22,00	20,93	20,79

(a) *AI mode*



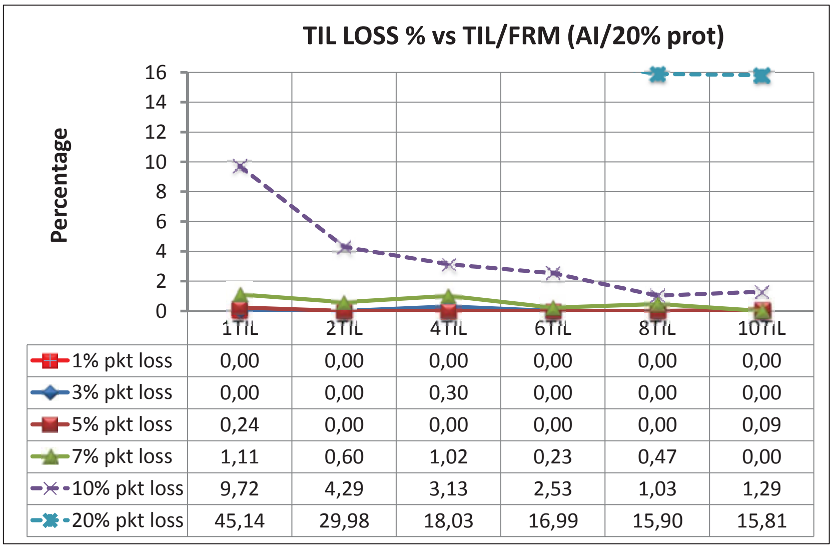
	1TIL	2TIL	4TIL	6TIL	8TIL	10TIL
1% pkt loss	0,00	0,00	0,00	0,00	0,00	0,00
3% pkt loss	0,63	0,60	0,08	0,08	0,13	0,00
5% pkt loss	3,87	3,42	1,48	1,12	0,89	0,57
7% pkt loss	9,25	7,12	4,77	3,78	3,17	2,46
10% pkt loss	20,63	12,72	10,67	7,77	7,99	7,04
20% pkt loss	46,48	32,25	26,96	23,99	22,65	21,06

(b) *LPI4 mode*

Figure 2.19. Tile loss percentage for every packet loss percentage for all the til/frm layouts and for the AI and LPI4 encoding modes. FEC-protected bit streams with a protection amount of 10% using a protection period of 333 ms and a repair symbol size of 192 bytes.

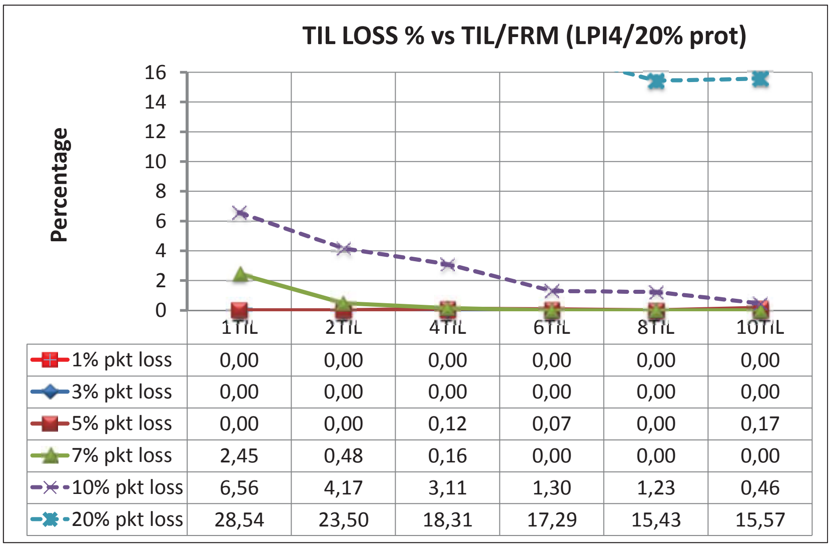
lowest overhead and the best percentage of recovery.

Figure 2.19 shows the tile loss percentage (after FEC recovery) for different



	1TIL	2TIL	4TIL	6TIL	8TIL	10TIL
1% pkt loss	0,00	0,00	0,00	0,00	0,00	0,00
3% pkt loss	0,00	0,00	0,30	0,00	0,00	0,00
5% pkt loss	0,24	0,00	0,00	0,00	0,00	0,09
7% pkt loss	1,11	0,60	1,02	0,23	0,47	0,00
10% pkt loss	9,72	4,29	3,13	2,53	1,03	1,29
20% pkt loss	45,14	29,98	18,03	16,99	15,90	15,81

(a) *AI mode*



	1TIL	2TIL	4TIL	6TIL	8TIL	10TIL
1% pkt loss	0,00	0,00	0,00	0,00	0,00	0,00
3% pkt loss	0,00	0,00	0,00	0,00	0,00	0,00
5% pkt loss	0,00	0,00	0,12	0,07	0,00	0,17
7% pkt loss	2,45	0,48	0,16	0,00	0,00	0,00
10% pkt loss	6,56	4,17	3,11	1,30	1,23	0,46
20% pkt loss	28,54	23,50	18,31	17,29	15,43	15,57

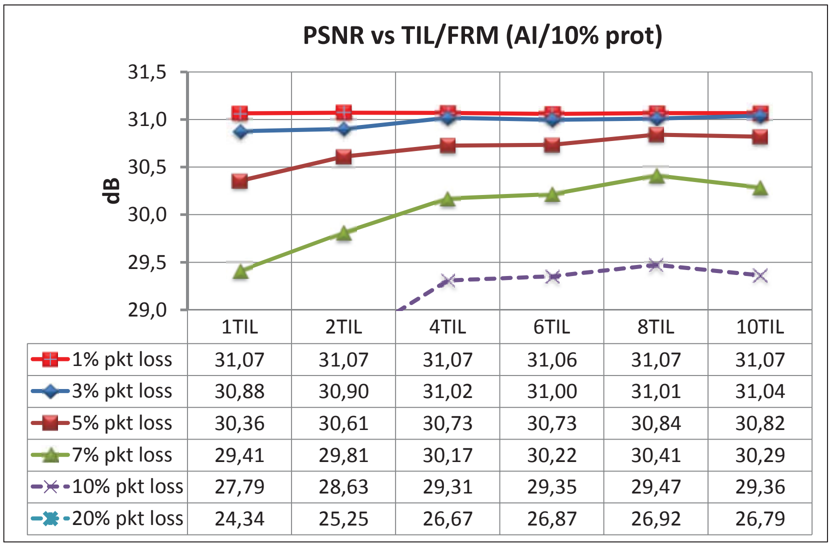
(b) *LPI4 mode*

Figure 2.20. Tile loss percentage for every packet loss percentage for all the til/frm layouts and for the AI and LPI4 encoding modes. FEC-protected bit streams with a protection amount of 20% using a protection period of 333 ms and a repair symbol size of 192 bytes.

til/frm layouts with a protection amount of 10% for both the AI and LPI4 modes and for the six specified packet loss rates (1%, 3%, 5%, 7%, 10%, and 20%).

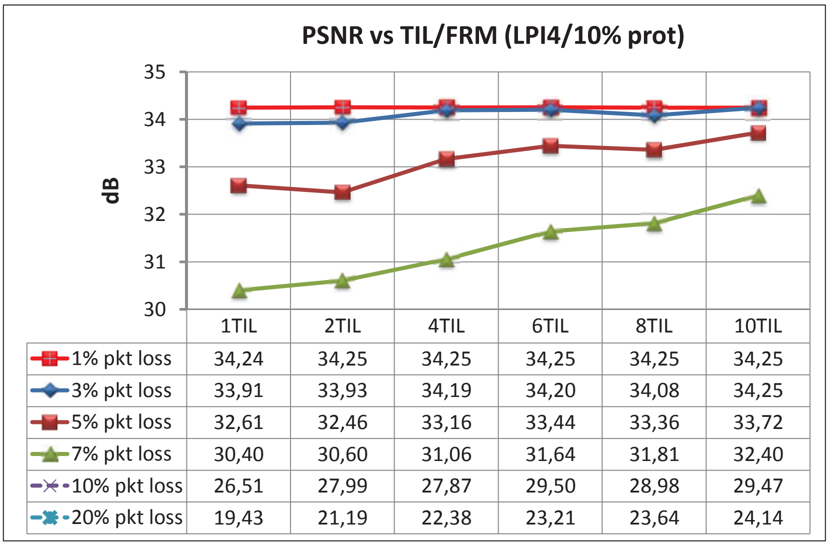
Figure 2.20 shows the same data for an FEC protection amount of 20%. In both figures, and for both the AI and LPI4 graphics, the y-axis has been fixed ranging from 0% to 16% to make comparisons between all of them clear. Note that the 20% packet loss curve is outside the bounds for this scale (or nearly out), so it is not completely sketched inside the graphic areas, but its values can be numerically inspected in the data tables. In the case of a protection amount of 10%, layouts of 4 til/frm and higher noticeably improve the recovery percentage over 1 til/frm and 2 til/frm layouts (in which the number of packets per tile penalizes the recovery process). A protection amount of 10% can almost completely recover a protected bit stream for a packet loss percentage of 1%, and keeps the tile loss percentage really low for a packet loss percentage of 3%. Even at a packet loss of 5%, a protection amount of 10% does a good job, especially for 8 and 10 til/frm layouts (where the final tile loss percentage is lower than 1% in both encoding modes). Data in Table 2.5 indicates that, for the 1 til/frm layout, the AI encoding mode obtains an average value of 4.38 packets per tile, and the LPI4 encoding mode obtains a very similar average value of 4.34 packets per tile. But the values of tile loss percentage for both modes for the 1 til/frm layout (in Figure 2.19) are not as similar. Although the average packets per tile are similar, the I and P frame distribution in each encoding mode is very different. In the AI mode all the frames are I frames. This implies that all the encoded frames have a similar size and the packets-per-tile ratio is constant for every frame. On the contrary, the LPI4 encoding mode inserts an I frame followed by three P frames. P frames are more efficient (and therefore they are smaller) than I frames, so the packets-per-tile ratio of a P frame is lower than that of an I frame. When the packets-per-tile ratio tends to 1 (high number of til/frm) these differences are narrow, but when the packets-per-tile ratio increases (low number of til/frm), the differences increase. This unbalance produces different results in the final tile loss percentage of both modes, and, in this situation, the LPI4 mode yields better recovery results than the AI mode. In the case of a protection amount of 20% (Figure 2.20), values of 5% of packet loss and lower are practically wiped out. For the LPI4 encoding mode, this is also true for 7% of packet loss and 4, 6, 8, and 10 til/frm layouts. The results in both figures show that the protection amount parameter does not have to be confused with the “recovery percentage.”

Figures 2.21 and 2.22 show the PSNR values corresponding to data in figures 2.19 and 2.20. The LPI4 PSNR values are better than those for the AI mode. The main reason is that the no-loss PSNR value for the LPI4 mode is better than for the AI mode (so, when the tile loss percentage tends to 0, the LPI4 mode quality is much higher than the AI mode quality). Only in situations where the tile loss percentage is high (e.g., for a protection amount of 10% and a packet loss of 10% or 20%), does the AI mode obtain better



	1TIL	2TIL	4TIL	6TIL	8TIL	10TIL
1% pkt loss	31,07	31,07	31,07	31,06	31,07	31,07
3% pkt loss	30,88	30,90	31,02	31,00	31,01	31,04
5% pkt loss	30,36	30,61	30,73	30,73	30,84	30,82
7% pkt loss	29,41	29,81	30,17	30,22	30,41	30,29
10% pkt loss	27,79	28,63	29,31	29,35	29,47	29,36
20% pkt loss	24,34	25,25	26,67	26,87	26,92	26,79

(a) *AI mode*



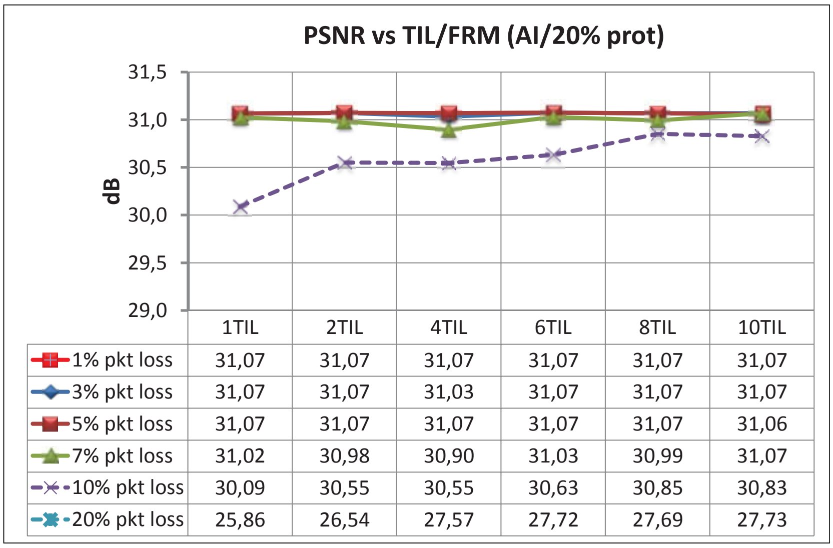
	1TIL	2TIL	4TIL	6TIL	8TIL	10TIL
1% pkt loss	34,24	34,25	34,25	34,25	34,25	34,25
3% pkt loss	33,91	33,93	34,19	34,20	34,08	34,25
5% pkt loss	32,61	32,46	33,16	33,44	33,36	33,72
7% pkt loss	30,40	30,60	31,06	31,64	31,81	32,40
10% pkt loss	26,51	27,99	27,87	29,50	28,98	29,47
20% pkt loss	19,43	21,19	22,38	23,21	23,64	24,14

(b) *LPI4 mode*

Figure 2.21. PSNR for every packet loss percentage for all the til/frm layouts and for the AI and LPI4 encoding modes. FEC-protected bit streams with a protection amount of 10% using a protection period of 333 ms and a repair symbol size of 192 bytes.

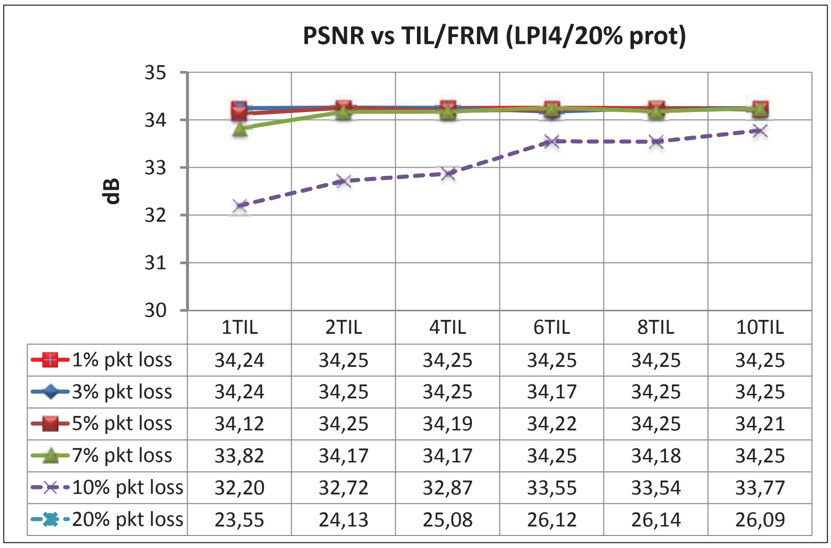
results than the LPI4 mode.

Note that these tests compare the tile loss percentage obtained by every



	1TIL	2TIL	4TIL	6TIL	8TIL	10TIL
1% pkt loss	31,07	31,07	31,07	31,07	31,07	31,07
3% pkt loss	31,07	31,07	31,03	31,07	31,07	31,07
5% pkt loss	31,07	31,07	31,07	31,07	31,07	31,06
7% pkt loss	31,02	30,98	30,90	31,03	30,99	31,07
10% pkt loss	30,09	30,55	30,55	30,63	30,85	30,83
20% pkt loss	25,86	26,54	27,57	27,72	27,69	27,73

(a) *AI mode*



	1TIL	2TIL	4TIL	6TIL	8TIL	10TIL
1% pkt loss	34,24	34,25	34,25	34,25	34,25	34,25
3% pkt loss	34,24	34,25	34,25	34,17	34,25	34,25
5% pkt loss	34,12	34,25	34,19	34,22	34,25	34,21
7% pkt loss	33,82	34,17	34,17	34,25	34,18	34,25
10% pkt loss	32,20	32,72	32,87	33,55	33,54	33,77
20% pkt loss	23,55	24,13	25,08	26,12	26,14	26,09

(b) *LPI4 mode*

Figure 2.22. PSNR for every packet loss percentage for all the til/frm layouts and for the AI and LPI4 encoding modes. FEC-protected bit streams with a protection amount of 10% using a protection period of 333 ms and a repair symbol size of 192 bytes.

layout for the same packet loss percentage. In the simulations in a realistic vehicular scenario (in the next chapter), different layouts produce different packet loss percentages to deal with, so every configuration does not “work” under the same conditions. Each different configuration (encoding mode, protection amount, til/frm, ...) may behave differently and will have to confront a different situation. In the next chapter, we will evaluate all the protection proposals (HEVC protection and RaptorQ protection) in a realistic vehicular network environment to check their performance in these types of scenarios.

Chapter 3

Evaluation

Contents

3.1 Test framework

For the evaluation of the previous proposals for providing robustness to video streaming over vehicular networks, we have used three main blocks (see Figure 3.1). The first of these blocks is formed by an open-source vehicular traffic simulator: SUMO (Simulation of Urban MObility) [58]. This simulator models the behavior of vehicles on routes, interacting with other vehicles, junctions, multi-lane roads, traffic lights, etc. The second of the blocks is in charge of the simulation of vehicular network communications, particularly those based on the IEEE 802.11p protocol and the IEEE 1609 family of standards (WAVE). For this task, we have selected OMNeT++ (Objective Modular Network Testbed in C++) [59]. OMNeT++ is not a network simulator itself but a framework that allows for the development of specific network simulators. An excerpt from its web page defines it like this: *"OMNeT++ is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators."* There are lots of simulators developed with OMNeT++. In this particular case, we have used the latest Veins (VEhicles In Network Simulation) implementation [60]. Veins is based on the MiXiM (MIXed sIMulator) project [61], which implements wireless and mobile networks. OMNeT++, MiXiM, and Veins are also open-source. The third of the blocks deals with video processing. It prepares the encoded bit streams for the tests and also evaluates the quality of the final recovered and reconstructed versions of the streamed sequences. The encoded bit streams are used to generate *video trace files* which are injected into the simulations. For this purpose, we have developed a new module in OMNeT++. This module provides a video sender with video packets so it can deliver them through a vehicular network scenario. Vehicles tagged as video receivers get video packets and write both a file with the correctly received packets and a file with several statistics such as packet loss ratio. These two files are subsequently used for evaluations. The vehicular simulator and the network simulator blocks are connected by TraCI (TRAffic Control Interface) [62]. TraCI creates a TCP connection to allow the communication between the two simulators. SUMO acts as a server (*TraCI-server*) and OMNeT++ acts as a client (*TraCI-client*). The TraCI-client can send commands to the TraCI-server to change the behavior of the vehicles. It also periodically requests data from the vehicular simulator to learn of the status of the simulation and the position of every single vehicle. This communication, in both directions, could be useful to model the vehicle behavior when receiving certain types of messages. For instance, vehicles receiving warnings about a congested road could change their route, and also accidents can be modeled by giving a vehicle the order to stop at a certain moment, which could trigger the recording and sending of video sequences so that emergency services can preview the accident environment.

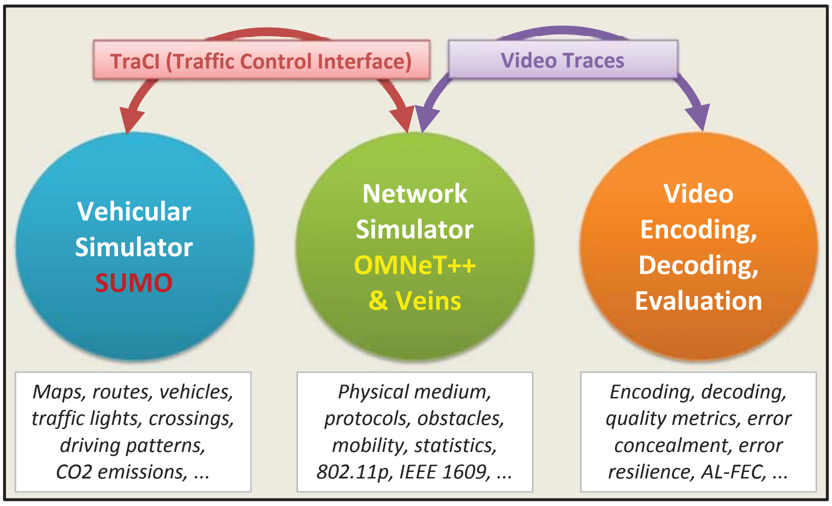
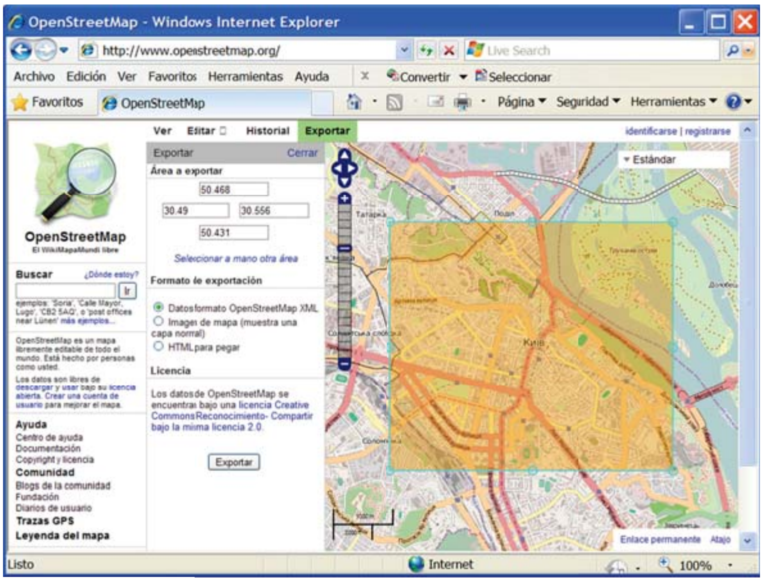

TraCI (Traffic Control Interface)
Video Traces
Vehicular Simulator SUMO
Network Simulator OMNeT++ & Veins
Video Encoding, Decoding, Evaluation
Maps, routes, vehicles, traffic lights, crossings, driving patterns, CO2 emissions, ...
Physical medium, protocols, obstacles, mobility, statistics, 802.11p, IEEE 1609, ...
Encoding, decoding, quality metrics, error concealment, error resilience, AL-FEC, ...

Figure 3.1. Test framework.

3.1.1 OpenStreetMap

As we just mentioned, the vehicular traffic simulator used herein is SUMO. In SUMO, road networks are XML files. These files can be generated by hand (by defining nodes and edges that connect them) or by importing data with different formats. We have chosen to use OpenStreetMap (OSM) [63] data because free maps from all around the world can be obtained. OpenStreetMap is a collaborative project whose aim is to provide a free map of the entire world (a public domain geographic database). As the OSM project grows, more and more data become available. It is mainly maintained by volunteers (in such a way as Wikipedia), therefore not all regions are described with a similar level of detail, but the main cities of the world are usually well documented. Recently, and primarily motivated by humanitarian actions taken after weather catastrophes, organizations such as the International Charter on Space and Major Disasters or the European Commission, provide the OSM database with geospatial information. OpenStreetMap does not only gather road information (including the number of lanes and their direction), but it also collects other kinds of data like buildings, rivers, parks, bus stops, schools, etc. Some pieces of this information can be used to define obstacles for wireless signals. From the OSM web page, XML data in OSM format can be downloaded and also bitmap images of the desired zone can be obtained. In



ejemplos: 'Soria', 'Calle Mayor, Lugo', 'CB2 5AQ', o 'post offices near Lünen' más ejemplos...

OpenStreetMap es un mapa libremente editable de todo el mundo. Está hecho por personas como usted.

Los datos son libres de descargar y usar bajo su licencia abierta. Crear una cuenta de usuario para mejorar el mapa.

Área a exportar: 50.468 / 30.49 / 30.556 / 50.431

Formato de exportación:
- [✓ Datos formato OpenStreetMap XML]
- Imagen de mapa (muestra una capa normal)
- HTML para pegar

Licencia: Los datos de OpenStreetMap se encuentran bajo una licencia Creative Commons Reconocimiento- Compartir bajo la misma licencia 2.0.

Figure 3.2. OpenStreetMap web page. Selecting a region and exporting XML data and bitmap images.

Figure 3.2, the OSM web page is shown, and especially the interface for exporting map data and map images. To select a certain region and export it to an OSM file, you can manually draw a rectangle over the displayed map or else introduce the coordinates of the region. For the scenario used in this work, we have selected a region in the city of Kiev (Ukraine). In Figure 3.3, the picked area (as a bitmap image) is shown. Figure 3.4 shows an orthophoto of the selected region. At the bottom of the picture, the Olympic National Sports Complex can be clearly seen.

3.1.2 SUMO

SUMO includes several tools to convert map data. For converting OSM data into SUMO road networks, we have used the *netconvert* tool. This tool has different options for importing data from many formats (OSM, openDrive, VISUM, VISSIM, RoboCup, MATsim) and several choices for adding traffic lights. For extracting buildings from OSM data, we have used the *polyconvert* tool. Once OSM data is converted into the SUMO format, it can be opened with SUMO-GUI (SUMO-Graphical User Interface), as shown in Figure 3.5. To define vehicles and the routes that they must trace, we have used the DUAROUTER tool, which is also a component part of the SUMO simulator. Before using DUAROUTER, *trips* and *flows* have to be defined. A *trip*

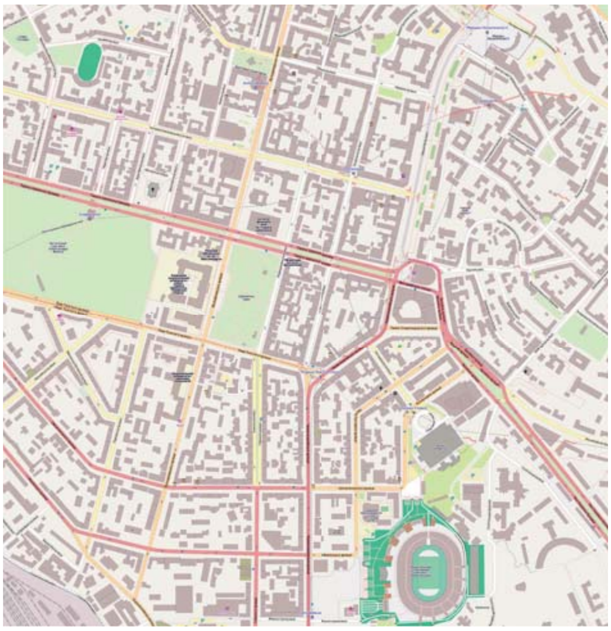


Figure 3.3. Downloaded bitmap image for the selected area obtained from OpenStreetMap.

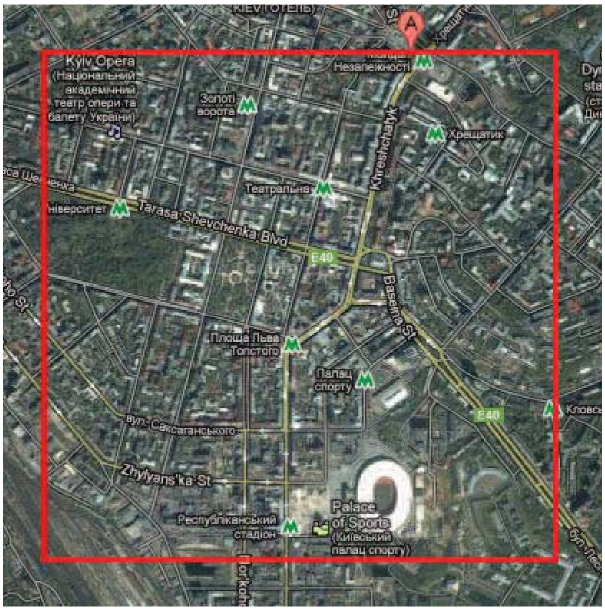
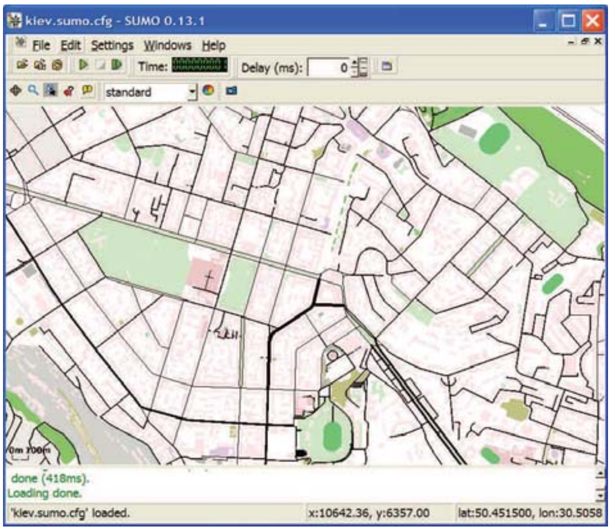


Figure 3.4. Partial view of the city of Kiev. The square indicates the area selected for the simulations.



done (418ms).
Loading done.
'kiev.sumo.cfg' loaded.
x:10642.36, y:6357.00
lat:50.451500, lon:30.5058

Figure 3.5. The SUMO simulator rendering the converted data obtained from OpenStreetMap.

consists of an origin, a destination, and the moment in which one vehicle parts from the initial edge. After a *trip* file is created (including several *trips*), the DUAROUTER tool calculates the route for each *trip* and generates a *route* file. Also, a file with several *flows* can be created for DUAROUTER. A *flow* is similar to a *trip* but the difference is that it is used by several vehicles departing periodically. From the *flow* file, DUAROUTER generates the different routes for the vehicles. By opening all these generated files (road network, buildings, vehicles/routes) with SUMO-GUI and running the simulation, the behavior of the vehicles through this scenario can be seen and checked: braking at junctions, stopping when traffic lights are red, accelerating when traffic lights change to green, etc.

3.1.3 OMNet++

As explained before, OMNeT++ is not a simulator itself but a framework for the development of simulators. It provides the structure and libraries for developing network simulators and, afterwards, creating network topologies for performing tests. It is public domain software, and it has a lot of contributors who have developed many projects (INET, Castalia, ReaSE, OverSim, INETMANET, MiXiM, Veins, etc.). OMNeT++ has some characteristics that make it interesting for our work: (a) it is an open-source tool, so it is free and customizable; (b) it has an easy-to-use GUI that is based on Eclipse (a well-known programming environment); (c) it works under

Linux/Unix, Windows, and Mac OS X, so it can be used in almost every computer; (d) the two projects that offer vehicular network support (MiXiM and Veins) are open and in continuous development, so it is probable that they will keep on adopting the new standards progressively. On the other hand, the main drawback is that the learning curve is not negligible. But this fact seems to be shared by most network simulators (OPNET, ns-2, ns-3, etc.). The two cited projects (MiXiM and Veins) have been put together to provide a vehicular network simulator.

MiXiM is a modeling framework for OMNeT++, created to simulate mobile and fixed wireless networks. It offers detailed models of radio propagation, interference estimation, and wireless MAC protocols. This project allows modeling, at a certain degree of detail, the characteristics of wireless devices. It defines several analogue models that add the effects of the channel to the transmitted signal by adding attenuation mapping (which defines the attenuation factors) to the signal. It also implements different physical and MAC layers models (CSMA, 802.11, etc.) and a module whose aim is to decide if the incoming signal is received correctly or perceived as noise. Diverse mobility patterns and a module to simulate battery consumption are also parts of the MiXiM project. MiXiM is the right framework over which the Veins project is built.

Veins is a tool developed by Christoph Sommer that has evolved over time. At the beginning, it was developed under the INET Framework package for OMNeT++ [64]. Its most recent versions are based on the MiXiM project, in which wireless communications are more detailed. Veins incorporates the IEEE protocols approved for its use in vehicular networks (namely IEEE 802.11p and IEEE 1609 WAVE) and an obstacle model [65] that simulates the attenuation caused upon the wireless signal by buildings.

Inside this vehicular network simulator, we have developed an application for running the experiments. It basically allows the injection of a video stream and the insertion of background traffic to produce congestion in the network (to simulate adverse conditions). It also gathers some statistics for subsequent analysis. The information that OMNeT++/MiXiM/Veins needs for simulating network transmissions is the packet size. It does not need to know the exact contents of the packet to run the simulations, so we have generated trace files from the encoded bit streams (with and without FEC protection) with the information needed for the simulation, that is, the size of every packet. We have also included the packet sequence number in order to be able to compare the received and decompressed videos with the original sequences. Our module lets the video transmitter read a trace file and send video packets. Vehicles receiving video write a similar trace file with the packets that they

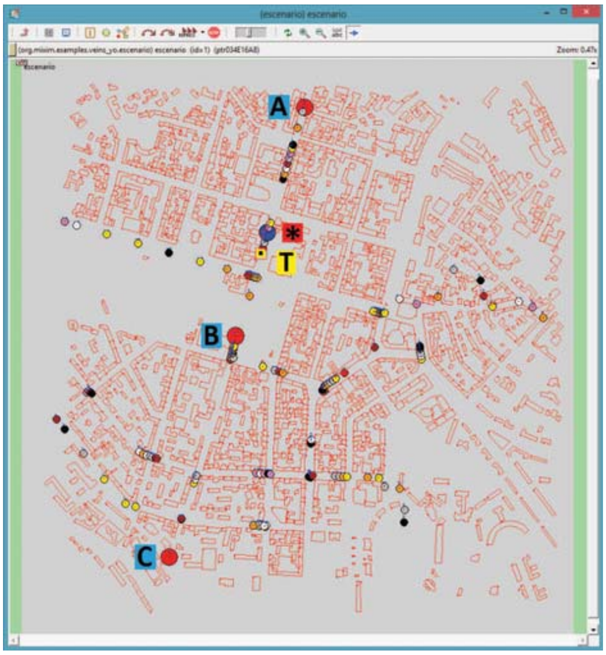


Figure 3.6. Vehicular network scenario in OMNeT++/MiXiM/Veins. *(red circles [A,B,C] = RSUs; blue circle [*] = video client; yellow square [T] = background traffic source; small circles = other vehicles; red polygons = buildings).*

receive correctly. Another file is written by every video receiver with some statistics.

The map zone selected for the tests (city of Kiev), loaded in OMNeT++/MiXiM/Veins, is displayed in Figure 3.6. Red drawings represent the buildings imported from OpenStreetMap. Veins uses a file with polygons to sketch the obstacles and to calculate the visibility between two wireless network interface cards. Veins, by default, represents vehicles with a small rectangle and a blue arrow (which is the vehicle's approximate direction of motion). In order to make data transmission in the simulation "visible" to our eyes, we have changed that representation. Circles of two different sizes surrounding the vehicles that transmit or receive data are drawn. A small circle around a car indicates that this car has received some piece of data (but not video data). These small circles cycle through seven different colors for making reception of data clearly noticeable. Big circles surround nodes that are transmitting or receiving video data. The circle around a video transmitter alternates between two colors (red/pink) each time it sends a video packet. The circle around a video receiver alternates between two colors (blue/green) each time it receives a video packet. This graphical representation helps us to visually check what is really happening in the simulation and to detect possible errors in the design of the experiments. Also, we can see how obstacles hinder the transmissions and contribute to packet losses.

3.1.4 Scenario setup

In the simulations, the layout shown in Figure 3.6 has been used. It is an area of 2000 m x 2000 m in the city of Kiev. There is a long avenue that crosses this area from north to south. Along the avenue, three Road Side Units (RSUs) have been positioned, tagged A, B, and C in the figure. They are around 1 km apart from each other. The coverage radius of all the wireless devices is 500 m, with a maximum data rate of 6 Mbps. There is a small area between RSUs A and B that is not covered by either of them, so a shaded area appears. There is also a small area halfway between RSUs B and C that is covered by both of them (where their signals overlap). Therefore, we have three different types of areas regarding transmission: (a) areas where a vehicle receives data from only one RSU; (b) one area where the signal is momentarily lost (between the A and B coverage areas); and (c) one area where the vehicle receives the signal from two RSUs (B and C). The three RSUs transmit the same video sequence simultaneously in a synchronized and cyclic manner. A total of 450 vehicles are inserted into the scenario, driving in different routes, which come and go from the simulation area. At every moment, there are simultaneously around 80 vehicles in the cited area. The maximum allowed speed is 50.4 km/h. Vehicles send ten *beacons* per second through the control channel (following the IEEE 1609.4 multi-channel operations). RSUs send periodically, through the control channel, advertisements of the video service that they offer, indicating the service channel used for the video stream. The video client vehicle (labeled in the figure as *) receives that invitation and commutes to the specified service channel in order to receive the video stream. This vehicle travels along the avenue, receiving the video stream from the RSUs through the specified service channel. Another vehicle drives nearby the video client which can act as a background traffic source (labeled as T in Figure 3.6) by sending packets through the wireless network at the specified Packets Per Second (PPS) rate. It is used to reproduce network congestion conditions. The video client will experience isolated packet losses (mainly due to background traffic) and bursty packet losses (around the limits of RSUs coverage).

3.2 Experiments and results

In this section, we will present the tests performed in the simulations and the analysis of the results obtained.

3.2.1 Experiments

In the experiments, the combination of the bit streams indicated in the previous sections has been used. For the tests, we have selected the BasketballDrill sequence (832x480 pixels, 25 fps, `bbd_25`), encoded with 9 different encoding modes (AI, IPIP, IPPP, IPx, IPx25pctCTU, IPx25pctTIL, IPxpattern, LP, and LPI4) and with 6 different layouts (1, 2, 4, 6, 8, and 10 til/frm). For each one of these bit streams, 4 different versions have been generated. One of these versions is encoded without any FEC protection, but with the appropriate division of tiles into network packets. The other three versions of every bit stream are FEC protected with RaptorQ codes for 3 different values of the protection amount parameter (10%, 20%, and 30%). Each one of these combinations (with or without protection) has been streamed by the three RSUs and received by the video client in the described vehicular scenario under 3 different "network conditions." The first group of tests has been conducted without injecting any background traffic. This group of tests is referred to as "ideal conditions" and permits the characterization of each one of the three zones mentioned before (the areas with coverage of only one RSU, the area with no coverage, and the area where the signals from two of the RSUs overlap). After that, experiments have been performed under 2 further network conditions: one with moderate background traffic (62 packets per second of 512 bytes), and another with more dense background traffic (125 packets per second of 512 bytes), both for protected bit streams and non-protected bit streams. The tests where the non-protected versions of the bit streams have been used serve to evaluate the performance of the encoding modes and the frame layouts, regarding error resilience. The tests where the FEC protection has been used serve to evaluate the specific level of protection added by RaptorQ codes. In the following section, the results obtained will be presented and analyzed.

3.2.2 Analysis of results

Ideal conditions

The first set of experiments in the vehicular scenario tries to evaluate the situation under ideal conditions, i.e., when no background traffic is injected into the service channel that is used by the RSUs to stream the video. For the analysis, we have selected the three different areas of the avenue cited before: **Zone I** is a zone with full coverage of one of the RSUs; **Zone II** has a small area where neither of the RSUs signals reach; and **Zone III** is located around the area where the signals from RSUs B and C overlap.

We have to differentiate between two measurements: the PLR (Packet Loss Ratio) and the TLR (Tile Loss Ratio). PLR is the percentage of network packets that get lost (from the total number of packets that form the sequence, including repair packets, if any). The TLR is the percentage of tiles (from the total number of tiles that form the sequence) that cannot be used in the decoding process because one or more of the network packets that carry parts of that tile get lost. These two concepts are not equivalent or directly proportional as explained before. We can have a low value for the PLR that leads to a high value of the TLR and vice versa. The average number of packets per tile was presented in Table 2.5.

In the experiments under "ideal conditions" (with no background traffic) in Zone I, every packet sent by the RSU is correctly received by the client vehicle for all the combinations of coding modes and tiles per frame and for both FEC-protected and non-protected streams. So, we have a PLR value of 0%. Consequently, no tile is missing in any of the sequences and the quality of the reconstructed video sequences is the quality directly provided by the encoding/decoding process (shown in Table 2.3). The coding efficiency of each one of the encoding modes prevails, so, in the absence of packet loss obtained in Zone I, the LP and IPx modes obtain the best PSNR values.

In tables 3.1 and 3.2, the average PLR, TLR, and PSNR values for bit streams without FEC protection, transmitted in zones II and III, under ideal conditions, are shown. In Table 3.1, the values are averaged for each encoding mode, over all the tested layouts (til/frm), and, in Table 3.2, the values are averaged for each layout, over all the encoding modes. In Zone II, we encounter a burst of packet loss corresponding to the area where neither of the RSUs signals arrive. The average PLR in this zone has a narrow range (from 1.98% to 2.01%). The TLR also has a narrow range (from 1.95% to 2.39%). As opposed to the results shown in Figure 2.11 (where the TLR was higher than the PLR, especially in 1 and 2 til/frm layouts), here, the PLR is always very similar to the TLR, even when a low number of tiles per frame is used. The reason is that, when isolated losses occur (like those generated by random packet loss), the real loss of one isolated packet entails the loss of the whole tile (which may mean the effective loss of several packets), but, when bursty losses occur, the loss of consecutive packets that belong to the same tile does not entail an increase in the TLR. For this reason, in Zone II, the PLR and

Table 3.1. PLR, TLR, and PSNR for non-protected bit streams under "ideal conditions" (0 pps) for zones II and III for every encoding mode averaged over all the til/frm layouts.

	Zone II			Zone III		
	PLR	TLR	PSNR	PLR	TLR	PSNR
AI	2.00%	2.03%	30.83dB	22.95%	24.12%	28.02dB
IPIP	1.99%	2.06%	32.51dB	22.50%	23.24%	28.98dB
IPPP	2.00%	2.04%	33.71dB	22.63%	23.48%	29.76dB
IPx	2.00%	2.15%	31.08dB	22.27%	23.74%	25.51dB
IPx25pctCTU	2.01%	2.09%	32.31dB	22.65%	24.17%	28.45dB
IPx25pctTIL	2.01%	2.00%	33.21dB	22.41%	23.61%	29.24dB
IPxpattern	2.01%	2.02%	33.20dB	22.50%	23.49%	29.35dB
LP	2.00%	1.95%	32.25dB	22.33%	23.60%	24.35dB
LPI4	1.99%	2.07%	33.74dB	22.27%	22.58%	29.64dB

Table 3.2. PLR, TLR, and PSNR for non-protected bit streams under "ideal conditions" (0 pps) for zones II and III for every til/frm layout averaged over all encoding modes.

	Zone II			Zone III		
	PLR	TLR	PSNR	PLR	TLR	PSNR
1 til/frm	1.98%	2.13%	32.42dB	22.91%	25.99%	28.19dB
2 til/frm	1.99%	2.03%	32.58dB	22.89%	24.84%	28.18dB
4 til/frm	2.01%	1.99%	32.56dB	22.64%	23.64%	28.07dB
6 til/frm	2.01%	2.10%	32.56dB	22.34%	22.56%	28.18dB
8 til/frm	2.00%	2.01%	32.55dB	22.08%	22.19%	28.15dB
10 til/frm	2.01%	2.03%	32.55dB	22.15%	22.14%	28.10dB

TLR have analogous values. In this zone, the best two modes regarding the PSNR value are LPI4 and IPPP, with 33.74 dB and 33.71 dB values, respectively. The IPx25pctTIL and IPxpattern modes also show good performance. As the TLR is not pronounced, all the encoding modes have PSNR values over 30 dB, even the LP, IPx, and IPx25pctCTU modes, which showed poor performance in the presence of packet loss. Under these conditions, the AI mode obtains the lowest PSNR value. Comparing the different frame layouts (in Table 3.2), no frame layout seems to perform better than the others, regarding the PSNR value.

In Zone III, where the signals of two of the RSUs overlap, the PLR is ten times higher than in Zone II. As the two RSUs are around 1 km away from each other, they cannot "see" each other (because the coverage radius of the wireless devices is 500 m). These collisions produce a high loss rate in the overlapped area, which is wider than the shaded area in Zone II. Some bursts of data loss appear in this zone. As in Zone II, the TLR values are only slightly higher than the PLR values. This is due again to the bursty nature of losses, where lost packets probably belong to the same tile(s). In this zone, as opposed to Zone II, the TLR values are high and this fact produces very low values for PSNR. The difference between the encoding modes that use some kind of intra refresh (such as IPPP) and the encoding modes that do not use intra refresh at all (such as LP) is emphasized. Also, in this zone, IPPP and LPI4 remain the best two encoding modes, and IPx25pctTIL and IPxpattern remain close. Here, the performance of the LP and IPx modes sinks. The same behavior as that for Zone II is observed regarding the frame layouts: all of them show very similar performance. Therefore, we can conclude that when bursty losses occur, the til/frm layout does not have any influence on the PLR value, or on the PSNR value.

Before analyzing the results for the FEC protected video sequences, it is useful to remember that the FEC protected bit streams are composed of two types of packets: (a) *protected source packets* (or, simply, protected packets), which include video data (one whole tile or one tile fragment) and a 4-byte FEC trailer (which identifies the protection period to which the protected packet belongs), and (b) *repair packets*, which are generated in the FEC-encoding process, including the information to be used in the recovery process in order to recover the data from missing packets. In order to evaluate FEC recovery efficiency, the ARPLR (After-Recovery Packet Loss Ratio) value measures the percentage of the source video packets missing over the total source video packets of the sequence after the FEC decoding process has been carried out.

In Table 3.3, the PLR, ARPLR, and TLR values for three of the encoding modes (AI, IPIP, and IPPP), protected with three different values for the protection amount parameter (10%, 20%, and 30%), averaged over all the different frame layouts, are shown. The rest of the encoding modes show similar behavior so, for the sake of concision, only three of them are included in the table. In Zone II, the PLR values are very similar to those in Table 3.1 (for the non-protected bit streams). The ARPLR values for this zone do not show good efficiency in the recovery of missing packets. In this zone, RaptorQ codes are not able to recover lost packets because of the bursty nature of the loss, so the ARPLR values are not much better than the PLR values, nor are they better than the PLR values of non-protected bit streams. RaptorQ codes

Table 3.3. PLR, ARPLR, and TLR for FEC-protected bit streams under "ideal conditions" (0 pps) for zones II and III for AI, IPIP, and IPPP encoding modes for three levels of the protection amount parameter (10%, 20%, and 30%) averaged over all the til/frm layouts.

	Zone II			Zone III		
	PLR	ARPLR	TLR	PLR	ARPLR	TLR
AI / 10%	2.00%	1.94%	2.03%	23.49%	29.05%	30.25%
AI / 20%	1.99%	1.78%	1.81%	23.94%	30.95%	32.37%
AI / 30%	2.01%	1.63%	1.67%	24.32%	28.62%	29.55%
IPIP / 10%	1.98%	1.89%	1.95%	23.08%	27.71%	28.22%
IPIP / 20%	2.00%	1.57%	1.62%	23.50%	28.05%	28.59%
IPIP / 30%	2.00%	1.42%	1.49%	24.05%	30.89%	31.61%
IPPP / 10%	2.00%	1.93%	1.98%	23.09%	26.49%	26.95%
IPPP / 20%	2.00%	1.78%	1.86%	23.55%	26.90%	27.45%
IPPP / 30%	2.00%	1.78%	1.89%	23.99%	28.43%	29.19%

have good recovery properties when dealing with isolated losses, but a minimum number of packets needs to be received for them to work properly, which is not the case when bursty losses are encountered. The recovery of missing packets from a burst of losses would need a very long protection window for any of the FEC techniques available (which would introduce a long delay) or else the incorporation of other complementary techniques like *interleaving*, which converts bursty losses into isolated losses, but which also introduces a non-negligible delay in the transmission process. So shaded areas (where no packet can be received) are not suitable for live video delivery and a readjustment of the RSUs location in order to eliminate these "black zones" may be the most appropriate action. In this zone, as happened with the non-protected bit streams, the ARPLR values do not generate high TLR values because of the bursty nature of the losses. In Zone III, the PLR is around 23%-24%. The bursty nature of losses in this zone makes FEC recovery unfeasible, and furthermore, there is a striking fact in Zone III measurements: the ARPLR values are higher than the PLR values. How can this happen? The PLR measurement takes into consideration the total number of packets in the FEC-encoded bit stream, which includes protected packets and repair packets. If losses are bursty and FEC recovery is not effective, only protected packets (which include video data) will be translated into source video packets (by just taking away the trailer). As correctly received repair packets may recover few (or none) source video packets, the proportion of missing source video packets over the total number of source video packets may produce a higher value for the ARPLR than for the PLR. As a consequence of the high TLR values in this zone, the reconstructed video sequences can be considered useless. A possible solution to the problem of overlapping resides in using different service channels for each one of the overlapping RSUs and incorporating a horizontal handover mechanism to guarantee a nearly uniform coverage area.

Background traffic conditions

In Table 3.4, the PLR and TLR values for bit streams without FEC protection for the six different layouts used averaged for all the encoding modes for the three zones under consideration and a moderate background traffic (62 pps/512 bytes) are shown. It can be observed that, as the number of tiles per frame increases, the PLR value decreases, and again, the poor performance of layouts with a low number of tiles per frame appears. In Zone I, the PLR obtained is exclusively due to background traffic (because, as stated before, for this zone and in the absence of background traffic, a PLR value of 0% is obtained). In zones II and III, the PLR value results from the combination of isolated packet losses and bursty packet losses. The TLR values for zones II and III indicate that all the bit streams received in these two zones are useless.

Table 3.4. PLR and TLR for non-protected bit streams under 62 pps background traffic conditions for zones I, II, and III for every til/frm layout averaged over all the encoding modes.

	Zone I		Zone II		Zone III	
	PLR	TLR	PLR	TLR	PLR	TLR
1 til/frm	7.82%	28.10%	16.87%	40.21%	27.32%	43.55%
2 til/frm	6.80%	14.56%	15.42%	26.06%	26.60%	34.14%
4 til/frm	5.49%	7.44%	13.77%	17.16%	25.65%	28.04%
6 til/frm	4.43%	5.13%	12.15%	13.49%	24.72%	25.33%
8 til/frm	3.73%	4.00%	10.99%	11.68%	24.01%	24.22%
10 til/frm	3.26%	3.42%	9.94%	10.34%	23.56%	23.61%

Table 3.5 shows the same measurements with more dense background traffic (125 pps/512 bytes). Again, zones II and III obtain very high TLR values, which produce very low quality video sequences. The quality of video sequences for Zone I is depicted in Figure 3.7 (62 pps) and Figure 3.8 (125 pps). The LP, IPx, and IPx25pctCTU encoding modes show poor performance for the two network conditions, in line with the previous tests. As was verified, 25% of intra refreshed CTUs does not contribute enough to make video robust. The IPIP encoding mode turns out to be the most effective under both

Table 3.5. PLR and TLR for non-protected bit streams under 125 pps background traffic conditions for zones I, II, and III for every til/frm layout averaged over all encoding modes.

	Zone I		Zone II		Zone III	
	PLR	TLR	PLR	TLR	PLR	TLR
1 til/frm	14.32%	44.63%	24.89%	54.02%	31.26%	54.17%
2 til/frm	14.01%	28.43%	24.71%	39.85%	30.78%	42.24%
4 til/frm	12.68%	16.88%	24.03%	29.37%	29.69%	33.40%
6 til/frm	10.69%	12.35%	21.75%	23.92%	27.65%	28.84%
8 til/frm	8.82%	9.52%	19.52%	20.53%	26.68%	27.16%
10 til/frm	7.43%	7.79%	17.80%	18.35%	25.82%	25.97%

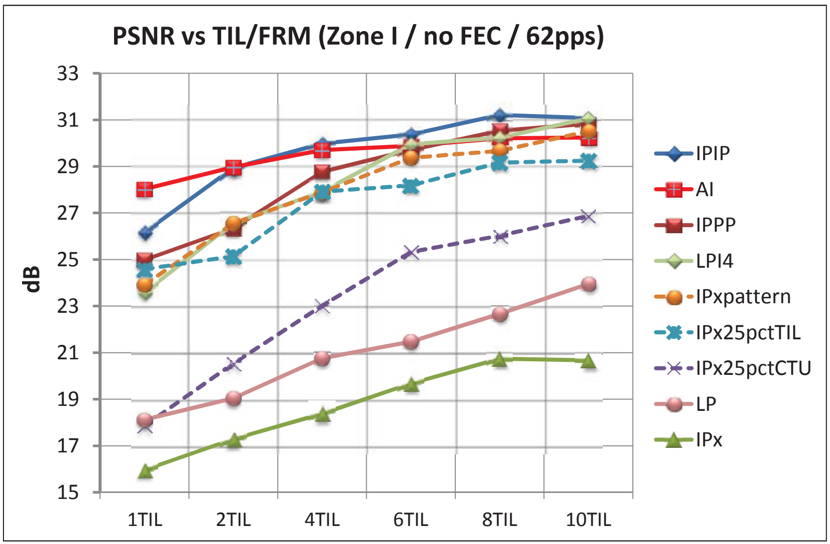


Figure 3.7. PSNR for every til/frm layout and every encoding mode. (Zone I; without FEC protection; 62 pps background traffic conditions).

network conditions (for a high number of tiles per frame). Under 125 pps traffic conditions, only the IPIP and AI encoding modes at a very high number of tiles per frame produce video sequences that lay at the lower quality bound, so, for these dense traffic conditions, FEC protection becomes neccesary.

Regarding the FEC protected bit streams, in Table 3.6, the performance of RaptorQ codes in zones I and II for background traffic of 62 pps for different values of the protection amount parameter (10%, 20%, and 30%) and different tiles per frame layouts averaged over all the encoding modes is shown. Under

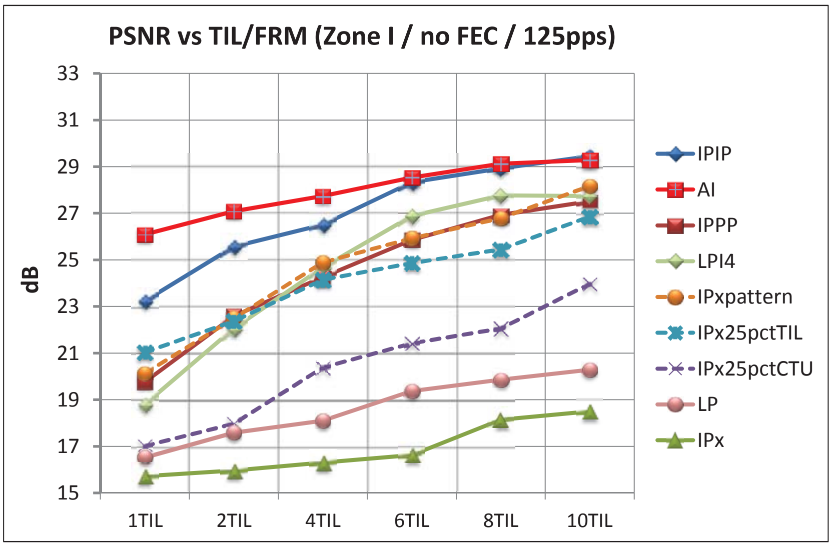


Figure 3.8. PSNR for every til/frm layout and every encoding mode. (Zone I; without FEC protection; 125 pps background traffic conditions).

these traffic conditions, in Zone I, a protection amount of 30% gets almost 100% of packet recovery for all the layouts of tiles per frame (only at 1 tile per frame, a very low ARPLR value of 0.02% remains). In the previous chapter, Figure 2.17 shows the bit rate values for two encoding modes (AI and LPI4) for different numbers of tiles per frame and for the 4 levels of protection used (no FEC protection, 10%, 20%, and 30%). Encoding a video sequence at 10 tiles per frame produces a bit stream around 9%-12% bigger than encoding it at 1 tile per frame. This overhead can be avoided in Zone I under these precise conditions if we protect the bit stream with a protection amount of 30%. Also, a protection amount of 20% nearly completely recovers all the missing source video packets, so the layouts with medium and low numbers of tiles per frame can be used, avoiding extra overhead and still obtaining the best video quality. If a protection amount of 20% or 30% is used, then the most efficient encoding mode (LP) provides the best quality, and, therefore, it is the best candidate to be used under these network conditions. In this zone (and also in Zone II), as the number of tiles per frame increases, the PLR value decreases. For a certain til/frm layout, the three different levels of FEC protection produce very similar PLR values (even more, when the percentage of protection amount increases the PLR value slightly decreases). The same behavior is also observed in Zone II. Therefore, increasing the bit rate because of the increase in the percentage of protection amount seems to have no adverse effect on the PLR values. Consequently, a bit stream with a higher value for the protection amount parameter directly obtains better ARPLR values. Even though, if all the

Table 3.6. PLR, ARPLR, and TLR for FEC-protected bit streams under 62 pps background traffic conditions for zones I and II for every til/frm layout for three levels of the protection amount parameter (10%, 20%, and 30%) averaged over all the encoding modes.

	Zone I			Zone II		
	PLR	ARPLR	TLR	PLR	ARPLR	TLR
1 til / 10%	6.96%	3.56%	11.80%	15.73%	13.55%	27.00%
1 til / 20%	6.89%	0.15%	0.40%	15.38%	11.22%	19.38%
1 til / 30%	6.31%	0.02%	0.05%	14.75%	9.62%	16.91%
2 til / 10%	6.44%	2.36%	4.97%	15.05%	13.21%	20.11%
2 til / 20%	6.34%	0.16%	0.26%	14.08%	9.91%	14.40%
2 til / 30%	6.05%	0.00%	0.00%	13.74%	9.03%	13.31%
4 til / 10%	5.47%	1.37%	1.85%	13.10%	10.26%	12.31%
4 til / 20%	5.17%	0.02%	0.03%	12.64%	8.56%	10.04%
4 til / 30%	5.10%	0.00%	0.00%	12.40%	8.13%	9.69%
6 til / 10%	4.57%	0.64%	0.72%	11.64%	9.18%	10.01%
6 til / 20%	4.49%	0.00%	0.00%	11.55%	8.11%	8.81%
6 til / 30%	4.30%	0.00%	0.00%	11.24%	7.42%	8.11%
8 til / 10%	3.77%	0.15%	0.16%	10.63%	8.37%	8.80%
8 til / 20%	3.76%	0.04%	0.04%	10.44%	7.47%	7.86%
8 til / 30%	3.70%	0.00%	0.00%	10.16%	6.85%	7.17%
10 til / 10%	3.20%	0.08%	0.08%	9.66%	7.38%	7.64%
10 til / 20%	3.17%	0.06%	0.06%	9.56%	7.00%	7.26%
10 til / 30%	3.11%	0.00%	0.00%	9.48%	6.52%	6.76%

applications that use the same service channel increase their bandwidth requirements up to levels that are too high (because of data protection), then the scenario becomes over-saturated. In order to avoid that type of situation, providing the maximum protection while maintaining the overhead under reasonable limits is the appropriate choice. By comparing the PLR and ARPLR values for Zone I and Zone II, the difference in the nature of losses can be noted. In Zone II, the efficiency of RaptorQ codes diminishes because packet losses include both isolated and bursty packet losses. The same measurements for Zone III are shown in Table 3.7. Here, the ARPLR values are higher than the PLR values. This indicates that the RaptorQ recovery process has no visible effect on the bit stream (as it happened in the "ideal conditions" case for this zone). The high TLR values observed in this zone produce very poor PSNR values, even with a protection amount of 30%

Table 3.7. PLR, ARPLR, and TLR for FEC-protected bit streams under 62 pps background traffic conditions for Zone III for every til/frm layout for three levels of the protection amount parameter (10%, 20%, and 30%) averaged over all the encoding modes.

	Zone III		
	PLR	ARPLR	TLR
1 til / 10%	27.56%	30.50%	39.99%
1 til / 20%	27.61%	30.94%	36.93%
1 til / 30%	27.67%	31.26%	36.05%
2 til / 10%	26.97%	32.43%	37.16%
2 til / 20%	26.83%	29.66%	31.92%
2 til / 30%	27.37%	29.66%	31.97%
4 til / 10%	26.12%	29.15%	30.24%
4 til / 20%	26.37%	27.57%	28.30%
4 til / 30%	26.83%	26.97%	27.73%
6 til / 10%	25.30%	26.38%	26.61%
6 til / 20%	25.80%	26.73%	26.86%
6 til / 30%	26.38%	26.79%	26.98%
8 til / 10%	24.77%	25.98%	26.08%
8 til / 20%	25.39%	26.23%	26.32%
8 til / 30%	26.17%	26.51%	26.59%
10 til / 10%	24.71%	25.82%	25.79%
10 til / 20%	25.47%	26.20%	26.18%
10 til / 30%	26.04%	26.53%	26.51%

(which cannot "heal" bursty packet losses).

In Figure 3.9, the PSNR values for every encoding mode and every til/frm layout in Zone I with a protection amount of 10% under 62 pps traffic background conditions are shown. The curves in that figure recommend the use of the IPPP and LPI4 encoding modes for the 4 and 6 til/frame layouts, and the use of the LP encoding mode for the 8 and 10 til/frm layouts. For the 1 and 2 til/frm layouts, the IPxpattern encoding mode shows the best PSNR values. By comparing Figure 3.9 and Figure 3.7, the benefit of using RaptorQ codes against isolated losses, even for a moderate protection amount of 10%, is proven. Figures 3.10, 3.11, and 3.12 show the PSNR curves for Zone II and 62 pps for a protection amount of 10%, 20%, and 30%, respectively. The LPI4 and IPPP modes, followed by IPxpattern, show the best performance in the three figures. In this zone and for these traffic conditions, an increase of the protection amount parameter from 20% to 30% does not improve PSNR values significantly for layouts with a high number of tiles per frame. It only improves PSNR in layouts with a low number of tiles per frame.

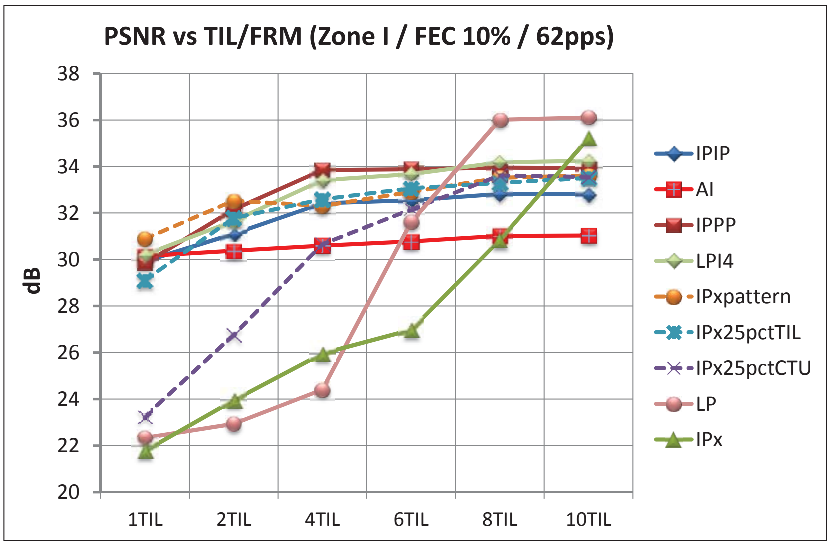


Figure 3.9. PSNR for every til/frm layout and every encoding mode. (Zone I; FEC protection amount of 10%; 62 pps background traffic conditions).

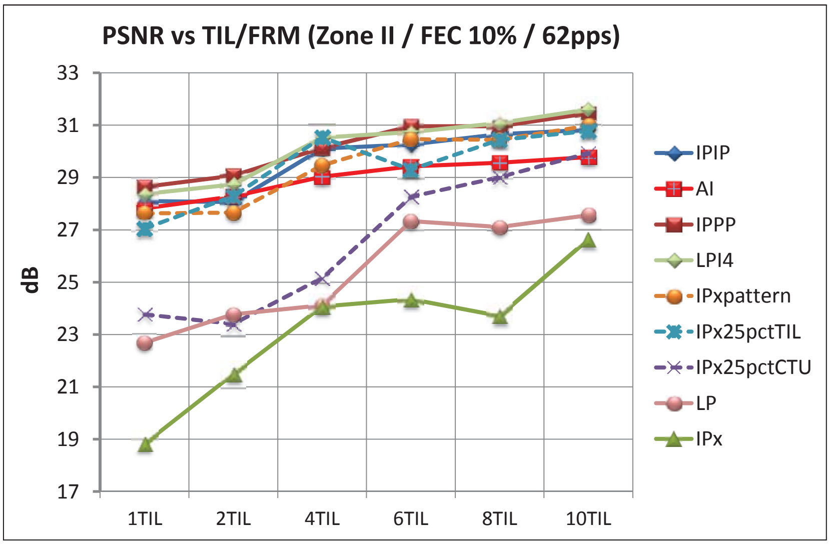


Figure 3.10. PSNR for every til/frm layout and every encoding mode. (Zone II; FEC protection amount of 10%; 62 pps background traffic conditions).

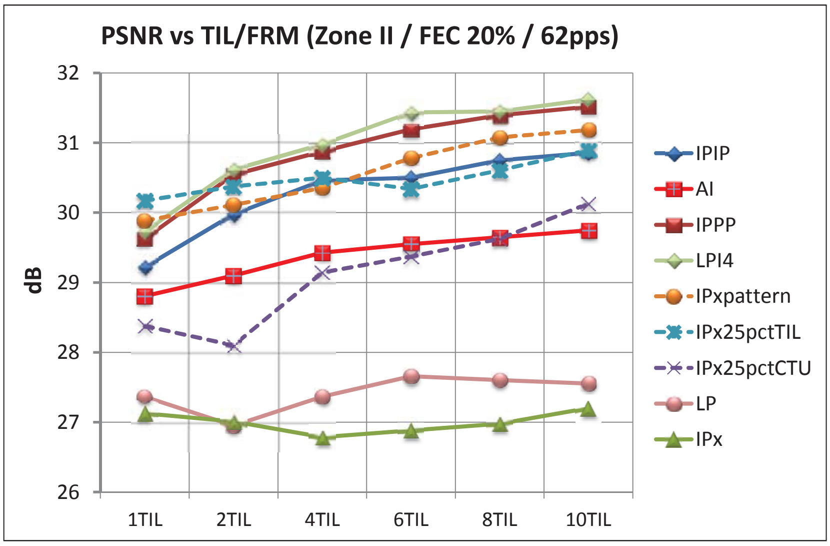


Figure 3.11. PSNR for every til/frm layout and every encoding mode. (Zone II; FEC protection amount of 20%; 62 pps background traffic conditions).

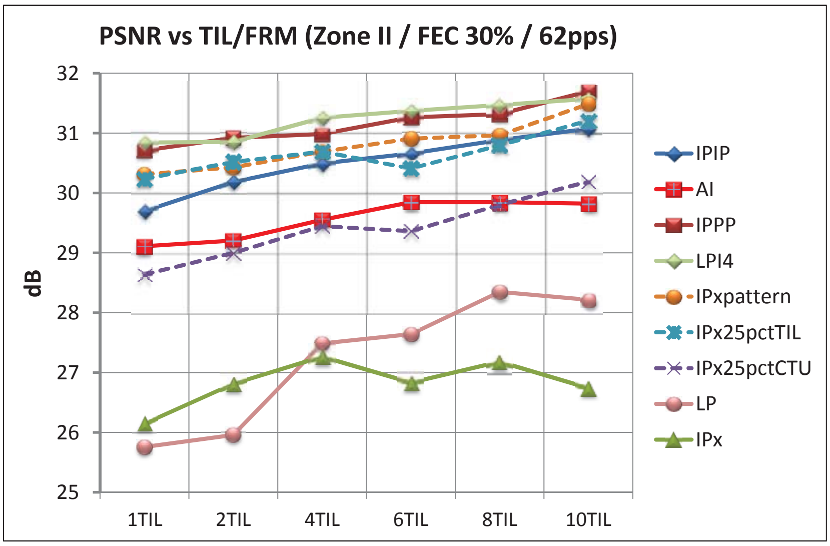


Figure 3.12. PSNR for every til/frm layout and every encoding mode. (Zone II; FEC protection amount of 30%; 62 pps background traffic conditions).

In the experiments where a more dense background traffic is used (125 packets per second of 512 bytes) the PLR values obviously increase. In Table 3.8, the PLR, ARPLR, and TLR values in zones I and II for background traffic of 125 pps are shown for different protection amount values and different tiles per frame layouts averaged over all the encoding modes. The same measurements for Zone III are shown in Table 3.9. If these two tables are compared with Table 3.5 (125 pps; bit streams without FEC protection), it is observed that the PLR values for zone I and II are very similar. This means that the added FEC protection (as in the 62pps case) does not have an adverse effect in the PLR values. In Zone III, for layouts with a low number of tiles per frame, the PLR values for the protected bit streams are higher than for the non-protected bit streams, but the resulting TLR values for both types of bit streams are higher than 37%, therefore, the reconstructed video sequences are useless at these high TLR values, and these differences do not mean in effect a improvement or a worsening of the video quality. In Zone I, a protection amount of 30% can restore almost all the missing packets for every layout. In this zone, a protection amount of 20% also obtains good ARPLR values, mainly for layouts with a high number of tiles per frame.

Regarding the encoding modes performance, in Figure 3.13, the PSNR values for Zone I with a protection amount of 10% and a background traffic of 125 pps are shown. The best quality is obtained for a layout of 10 til/frm and the LPI4 and IPxpattern encoding modes. For the 62 pps traffic conditions and a protection amount of 20% (Table 3.6), the recommendation was to use a low number of tiles per frame layout (to reduce overhead) and the LP encoding mode (which provided the best PSNR value). If the LP encoding mode and a layout with low number of tiles per frame (1, 2, or 4 til/frm) are selected and the network conditions change into 125 pps, the TLR values move away from 0% (3.54% - 16.34%), and PSNR values are low (see Figure 3.14). Therefore, as network conditions in a vehicular environment are continuously changing, an encoding mode, such as LP, which is so sensitive to packet loss (even for low TLR), is always a risky choice (unless 0% TLR could be guaranteed by high protection amounts).

Figures 3.15, 3.16, and 3.17 show the PSNR curves for a protection amount of 10%, 20%, and 30% in Zone II. For a protection amount of 10% and 10 til/frm, the TLR obtains an average value of 14.92% (Table 3.8). At these high TLR values, the IPIP encoding mode shows the best performance. In this zone, when using a protection amount of 20% and 30% the best PSNR values are obtained by the IPPP and LPI4 encoding modes. By comparing the curves in Figure 3.16 and Figure 3.17 it can be observed that a protection amount of 30% does not provide huge improvements over a protection amount of 20%, because, once the isolated losses are overcome, the unrecoverable bursty losses establish the maximum quality boundaries.

Table 3.8. PLR, ARPLR, and TLR for FEC-protected bit streams under 125 pps background traffic conditions for zones I and II for every til/frm layout for three levels of the protection amount parameter (10%, 20%, and 30%) averaged over all the encoding modes.

	Zone I			Zone II		
	PLR	ARPLR	TLR	PLR	ARPLR	TLR
1 til / 10%	14.19%	12.68%	38.56%	25.25%	29.32%	54.41%
1 til / 20%	14.55%	6.06%	16.34%	25.73%	25.64%	41.68%
1 til / 30%	14.22%	0.56%	1.62%	25.45%	20.76%	28.15%
2 til / 10%	13.99%	12.78%	25.38%	24.18%	30.88%	43.43%
2 til / 20%	14.36%	6.31%	12.13%	24.51%	24.38%	31.57%
2 til / 30%	13.78%	0.75%	1.33%	24.74%	21.53%	26.31%
4 til / 10%	12.81%	10.89%	14.80%	23.31%	26.45%	30.91%
4 til / 20%	12.07%	2.68%	3.54%	22.83%	20.93%	23.16%
4 til / 30%	11.90%	0.25%	0.30%	22.57%	17.31%	18.90%
6 til / 10%	10.58%	7.26%	8.40%	20.94%	22.08%	23.83%
6 til / 20%	10.21%	0.80%	0.89%	20.62%	15.40%	16.32%
6 til / 30%	10.50%	0.00%	0.00%	20.37%	13.79%	14.46%
8 til / 10%	9.03%	5.12%	5.56%	18.90%	18.23%	18.98%
8 til / 20%	8.50%	0.20%	0.23%	18.66%	13.19%	13.62%
8 til / 30%	8.79%	0.00%	0.00%	18.81%	12.37%	12.75%
10 til / 10%	7.68%	3.27%	3.47%	17.15%	14.56%	14.92%
10 til / 20%	7.71%	0.18%	0.19%	16.88%	11.44%	11.73%
10 til / 30%	7.58%	0.14%	0.14%	16.80%	11.09%	11.36%

Summary

From all the results obtained in the simulations in the vehicular scenario, several findings can be highlighted.

Vehicular scenario. Under "ideal conditions" for areas with good coverage of only one RSU, all the network packets arrive to the video clients, so the video quality is directly determined by the efficiency of the encoding mode used. In shaded zones where the signal does not reach, and in zones with overlapping signals, bursty losses appear. Under background traffic conditions, even for moderate traffic, isolated packet losses appear for areas with good signal coverage, mainly due to collisions and the WAVE multichannel operations functioning.

Table 3.9. PLR, ARPLR, and TLR for FEC-protected bit streams under 125 pps background traffic conditions for Zone III for every til/frm layout for three levels of the protection amount parameter (10%, 20%, and 30%) averaged over all the encoding modes.

	Zone III		
	PLR	ARPLR	TLR
1 til / 10%	34.68%	39.05%	57.14%
1 til / 20%	33.96%	35.15%	45.06%
1 til / 30%	33.95%	34.20%	40.97%
2 til / 10%	33.12%	37.71%	46.08%
2 til / 20%	33.25%	35.70%	40.89%
2 til / 30%	33.52%	33.85%	37.51%
4 til / 10%	32.20%	35.76%	38.52%
4 til / 20%	31.80%	30.66%	31.76%
4 til / 30%	32.03%	29.34%	30.07%
6 til / 10%	30.09%	32.45%	33.41%
6 til / 20%	30.31%	28.22%	28.45%
6 til / 30%	30.95%	28.59%	28.82%
8 til / 10%	28.26%	28.83%	29.10%
8 til / 20%	28.79%	26.76%	26.85%
8 til / 30%	29.15%	26.63%	26.74%
10 til / 10%	27.70%	27.61%	27.63%
10 til / 20%	28.09%	26.24%	26.20%
10 til / 30%	28.37%	26.83%	26.84%

Frame layout. For isolated losses, the selected layout has a direct influence on the PLR and TLR values. As the number of tiles per frame increases (a) the PLR value decreases and (b) for the same PLR value, a lower TLR value is obtained. When losses are bursty in nature the layout has little influence because (a) the PLR is very similar for all the til/frm layouts and (b) the TLR value is in line with the PLR value because missing packets in a burst probably belong to the same tile(s).

Encoding modes. The encoding modes that are inherently sensitive to errors (LP, IPx) provide the best performance in the total absence of losses, but rapidly worsen in the presence of losses, even for low TLR values. The encoding modes that have an implicit error resilience (AI, IPIP) are at a disadvantage because they start off with low quality values, therefore they only obtain better PSNR values for pronounced losses, and, in those situations, the

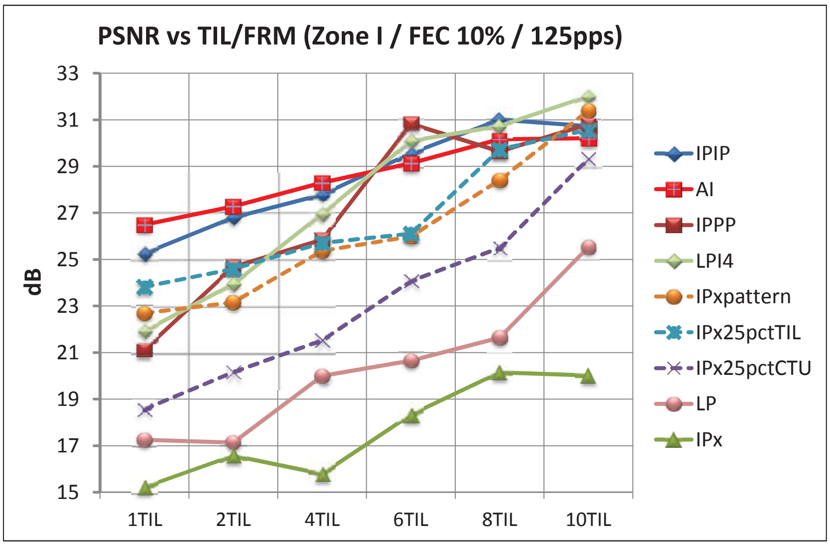


Figure 3.13. PSNR for every til/frm layout and every encoding mode. (Zone I; FEC protection amount of 10%; 125 pps background traffic conditions).

final quality is very near (or even under) the acceptable minimum quality. The "combined" encoding modes take advantage of their characteristics to provide the best quality in the majority of cases when packet losses appear. From all the "mixed" encoding modes proposed, the IPPP and LPI4 encoding modes show the best performance in most of the situations. The IPxpattern encoding mode (and, in some ocassions, the IPx25pctTIL encoding mode) offers decent results. The IPx25pctCTU encoding mode, in spite of using intra refresh in the same proportion as the IPxpattern and IPx25pctTIL encoding modes, does not provide the expected performance; therefore, it is rejected as a protection mechanism.

RaptorQ codes. RaptorQ codes have good performance when dealing with isolated losses, but they do not solve the problem of bursty losses. In those situations the use of other mechanisms, such as *interleaving*, may be used to convert bursty losses into isolated losses. Even though, RaptorQ codes (or other network packet protection tools) are neccesary in video streaming over vehicular networks because the protection provided by source coding is not enough to guarantee video delivery. The combination of the two protection approaches (RaptorQ codes and source coding) offers good performance: when RaptorQ codes are not able to recover all the missing packets, the source coding protection mitigates the propagation and multiplication effect of errors.

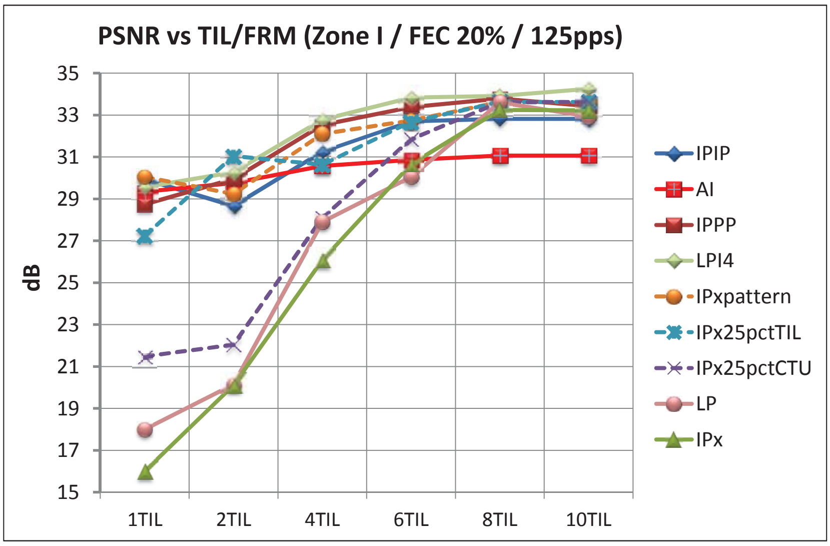


Figure 3.14. PSNR for every til/frm layout and every encoding mode. (Zone I; FEC protection amount of 20%; 125 pps background traffic conditions).

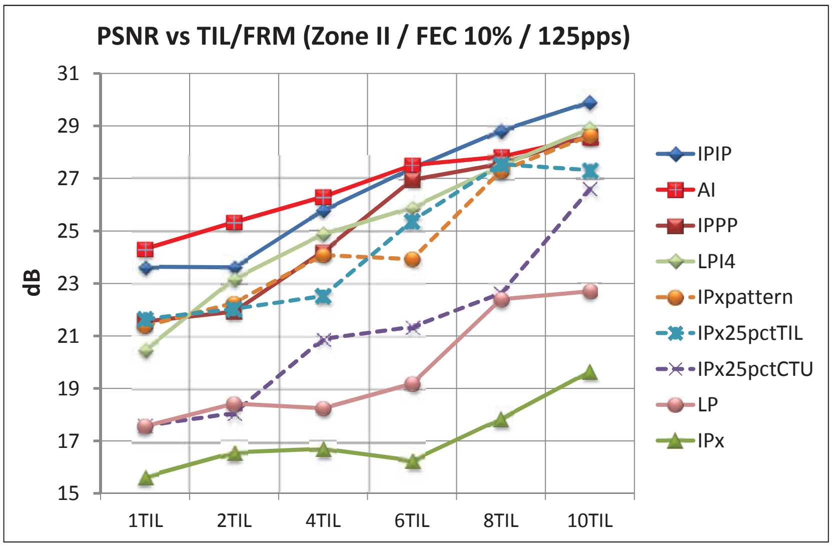


Figure 3.15. PSNR for every til/frm layout and every encoding mode. (Zone II; FEC protection amount of 10%; 125 pps background traffic conditions).

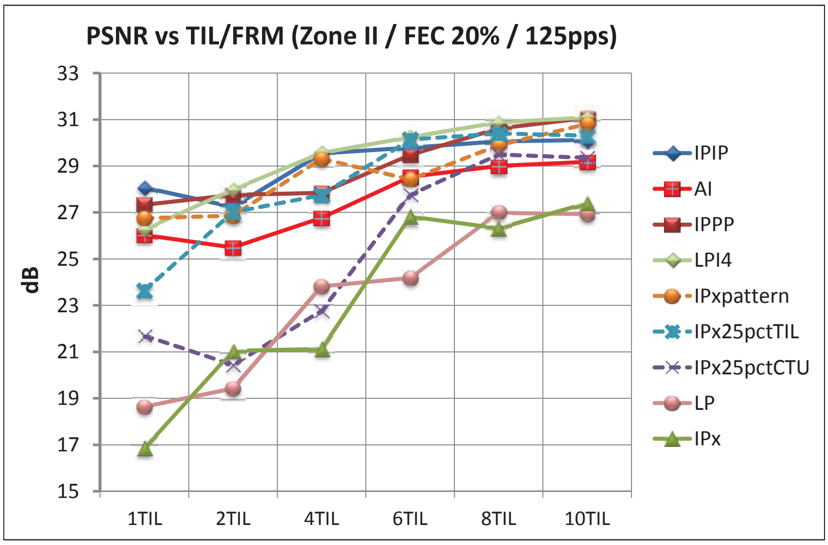


Figure 3.16. PSNR for every til/frm layout and every encoding mode. (Zone II; FEC protection amount of 20%; 125 pps background traffic conditions).

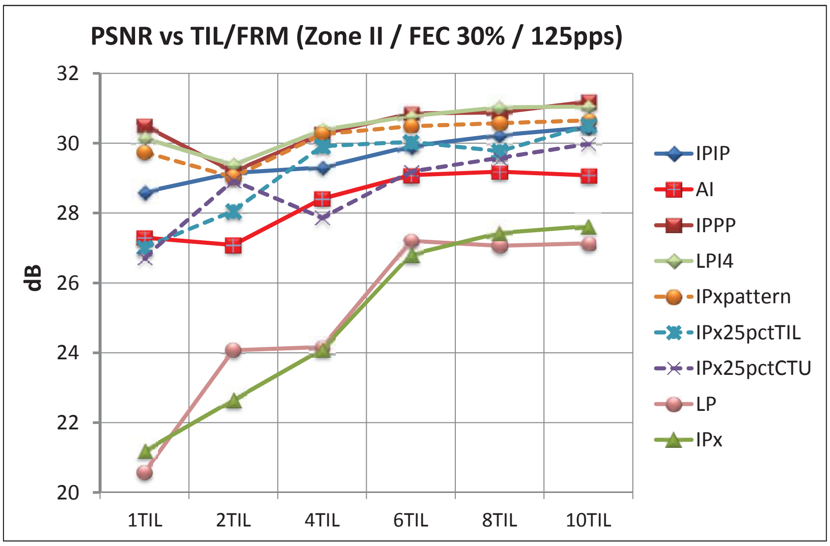


Figure 3.17. PSNR for every til/frm layout and every encoding mode. (Zone II; FEC protection amount of 30%; 125 pps background traffic conditions).

Chapter 4

Conclusions and future work

Contents

4.1 Conclusions

In this section, the main contributions and developments of this thesis are summarized.

To accomplish the main objective of this thesis, several proposals have been made and they have been evaluated both to measure their efficiency and their suitability for improving video robustness.

A new element from the HEVC standard has been introduced and evaluated, which, to the author's knowledge, has not been used before in the literature. This element can be called *tileslice* because it consists in the combination of a *slice* and a *tile*. It offers the advantages of *slices* by adding error resilience features to an encoded bit stream on a sub-frame level and also the advantages of *tiles* by increasing coding efficiency with respect to classical *slices*. An evaluation of the efficiency of 6 different layouts using *slices* or *tileslices* has been carried out. For these tests, a total of 14 video sequences belonging to the "common conditions" have been used, encoded with 2 encoding modes (AI, LP), with 4 QP values (22, 27, 32, 37). Results show that *tileslices* are always more efficient than classical *slices*. Regarding video streaming robustness, the layouts with higher numbers of *tileslices* per frame offer the best results. The PLR for these layouts is lower and, moreover, they avoid the multiplication effect that leads to high TLR values. There is an exception when the packet loss is bursty. For bursty packet losses, the multiplication effect does not appear, so, in this case, all the layouts offer very similar results.

Seven new encoding modes have been proposed in order to improve error resilience in the encoded bit streams. These modes are IPIP, IPPP, IPx, IPx25pctCTU, IPx25pctTIL, IPxpattern, and LPI4. They introduce intra refresh to a certain level. They have been evaluated, together with the well-known AI and LP modes. The LPI4 and IPPP modes have turned out to be the most effective when moderate losses occur. For an environment with 0% PLR, the LP mode would be the best choice because its coding efficiency is higher than the rest, but a vehicular network completeley free of packet loss is rather unlikely. For severe packet losses, the IPIP and AI modes obtain the best PSNR values, but most of the times these high TLR values entail a very low PSNR value that leads to useless sequences.

An Error Concealment method has been integrated in the HEVC reference software decoder and evaluations have been made comparing PSNR values for the EC decoder version against the non-EC decoder version. Average gains that range from 0.36 dB up to 1.66 dB show that the recommended action is always using the EC version decoder. EC is not a technique that can completely erase errors by itself, but it relieves the damaged video quality caused by errors that have not been protected by other means.

RaptorQ codes, an Application Layer Forward Error Correction technique, have been evaluated in order to obtain the most appropriate parameters that make them suitable for video streaming data protection. The optimal values for reducing overhead and maximizing recovery properties suggest a symbol size value of 192 bytes and a protection period of 333 ms. It has been proven that the longer the protection period, the higher the recovery percentage, so this parameter is set as a design decision considering this temporal window the maximum acceptable latency. RaptorQ codes have performed well in recovering isolated packet losses, but their performance tends to zero when they have to deal with bursty packet losses. Three different values for the protection amount parameter have been used (10%, 20%, and 30%). Obviously, RaptorQ codes with a higher protection amount will be able to recover lost packets in transmissions with higher PLR values, but the main drawback is that network overhead increases. In vehicular networks, bandwidth is a limitation factor, so an excessive protection amount can turn the solution into the problem (increasing channel saturation may lead to an increase in PLR).

Several other developments have been carried out in order to perform all the evaluations. The HEVC reference software decoder has been improved in order to be able to work with packet losses (the original reference software decoder crashes when some pieces of data are missing). A new module in OMNeT++ has been developed that allows injecting video data in vehicular network simulations.

4.2 Future work

Several proposals to extend the research carried out in this thesis are presented below:

- Utilization of the IPxpattern encoding mode to implement Unequal Error Protection based on Regions Of Interest by providing the most important frame regions with higher intra refreshing rates.
- Implementation and evaluation of complementary protection techniques, such as horizontal handover to avoid the overlapping problem, and interleaving to avoid the bursty nature of losses.
- Design and evaluation of protection mechanisms at a higher level, by implementing adaptive techniques that dynamically change the parameters

of protection (protection amount, protection period, encoding mode, etc.) using a feedback station and introducing adaptive mechanisms based on network conditions.

4.3 Publications

The published works originating from the development of this thesis are listed below.

- Pablo Piñol, Miguel Martínez-Rach, Otoniel López, Manuel Pérez Malumbres, "Protection of HEVC Video Delivery in Vehicular Networks with RaptorQ Codes", The Scientific World Journal, 2014.

- Piñol, P.J., Torres, A., Lopez Granado, O.M., Martinez Rach, M.O., Malumbres, M.P., "Evaluating HEVC Video Delivery in VANET Scenarios", IFIP Wireless Days, Valencia, 2013.

- Piñol, P.J., Torres, A., Lopez Granado, O.M., Martinez Rach, M.O., Malumbres, M.P., "An Evaluation of HEVC using Common Conditions", XXIV Jornadas de Paralelismo, JP2013, Madrid, 2013.

- Pablo Piñol, Otoniel López, Miguel Martínez, José Oliver, Manuel P. Malumbres, "Modeling video Streaming over VANETs", 7th ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks, PM2HW2N'12, pages 7-14, New York, 2012.

- Piñol, P.J., Martinez Rach, M.O., Lopez Granado, O.M., Oliver, J., Malumbres, M.P., "Video Transmission Simulations in Vehicular Adhoc Networks", XXIII Jornadas de Paralelismo, JP2012, Elche, 2012.

- Pablo Piñol, Otoniel Lopez, Miguel Martinez-Rach, Manuel P. Malumbres, José Oliver, Carlos T. Calafate, "Testing Performance of Current Video Codecs in Teleoperated Mobile Robot Applications: A Practical Experience", Mobile Robots Navigation, 1st Edition, p-501-514, Intech, ISBN: 978-953-307-076-6, 2010.

- Piñol, P.J., Martinez Rach, M.O., Lopez Granado, O.M., Malumbres, M.P., Oliver, J., "A Practical Study of Video Streaming Over IEEE 802.11 Wireless Networks using DirectShow Video Encoders", XVIII Jornadas de Paralelismo, JP2007, Zaragoza, 2007.

- P. Piñol, M. Martinez Rach, O. Lopez, M.P. Malumbres, J. Oliver, “Analyzing the Impact of Commercial Video Encoders in Remotely Teleoperated Mobile Robots through IEEE 802.11 Wireless Network Technologies”, 5th IEEE International Conference on Industrial Informatics, INDIN2007, pages 425-430, Viena, 2007.

Appendix I

Acronyms

AI	All Intra
AL-FEC	Application Layer Forward Error Correction
AMVP	Advanced Motion Vector Prediction
ARIB	Association of Radio Industries and Businesses
ARPLR	After-Recovery Packet Loss Ratio
ASV	Advanced Safety Vehicle
AVC	Advanced Video Coding
BD-Rate	Bjørntegaard Delta Rate
C2C-CC	Car 2 Car Communications Consortium
CABAC	Context-Adaptive Binary Arithmetic Coding
CCH	Control Channel
COMeSafety	Communications for eSafety
CTU	Coding Tree Unit
CVIS	Cooperative Vehicle-Infrastructure Systems
DCT	Discrete Cosine Transform
DFRM	Decoded Frame Rate Metric
DOT	Department of Transportation
DPB	Decoded Picture Buffer
DSRC	Dedicated Short-Range Communications
DST	Discrete Sine Transform
DTN	Delay Tolerant Networks
EC	Error Concealment
ECC	Error Correcting Code
ER	Error Resilience
ERTICO	Intelligent Transportation Systems and Services for Europe
ETC	Electronic Toll Collection

EUCAR	European Council for Automotive R&D
FCC	Federal Communications Commission
FEC	Forward Error Correction
FMO	Flexible Macroblock Ordering
FPS	Frames Per Second
GloMoSim	Global Mobile System Simulator
HEVC	High Efficiency Video Coding
OBU	On Board Unit
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
ITS	Intelligent Transportation System
ITSA	Intelligent Transportation Society of America
ITU-T	International Telecommunication Union - Telecommunication Standardization Sector
JCT-VC	Joint Collaborative Team on Video Coding
LC	Layered Coding
LLC	Logical Link Control
LP	Low-delay P
MANET	Mobile Ad-Hoc Network
MB	Macro-Block
MC	Motion Compensation
MDC	Multiple Description Coding
ME	Motion Estimation
MIC	Ministry of Internal Affairs and Communications
MiXiM	MIXed sIMulator
MLIT	Ministry of Land Infrastructure and Transport

MPEG	Moving Pictures Experts Group
MTU	Maximum Transmission Unit
MV	Motion Vector
NILIM	National Institute for Land and Infrastructure Management
OBU	On Board Unit
OSM	OpenStreetMap
OMNeT++	Objective Modular Network Testbed in C++
PLR	Packet Loss Ratio
PPS	Packets Per Second
PSNR	Peak Signal-to-Noise Ratio
QP	Quantization Parameter
ROI	Region Of Interest
RSSI	Received Signal Strength Indication
RSU	Road Side Unit
SAO	Context Adaptive Offset
SCH	Service Channel
SUMO	Simulation of Urban MObility
SUMO-GUI	SUMO-Graphical User Interface
SSIM	Structural SIMilarity
TCP	Transmission Control Protocol
TGp	Task Group “p”
TLR	Tile Loss Ratio
TraCI	TRAffic Control Interface
UDP	User Datagram Protocol
UEP	Unequal Error Protection
UTC	Coordinated Universal Time

V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
VANET	Vehicular Ad-Hoc Network
Veins	Vehicles In Network Simulation
VCEG	Video Coding Experts Group
VOD	Video On Demand
VSC	Vehicle Safety Communications
WAVE	Wireless Access in Vehicular Environments
WBSS	WAVE Basic Service Sets
WiMAX	Worldwide Interoperability for Microwave Access
WPP	Wavefront Parallel Processing
WSMP	WAVE Short-Message Protocol

Appendix II

Video sequences

Figure II.1. BasketballDrill (`bbd_25`), 832x480, 25 fps.

Figure II.2. BQMall (`bqm_30`), 832x480, 30 fps.

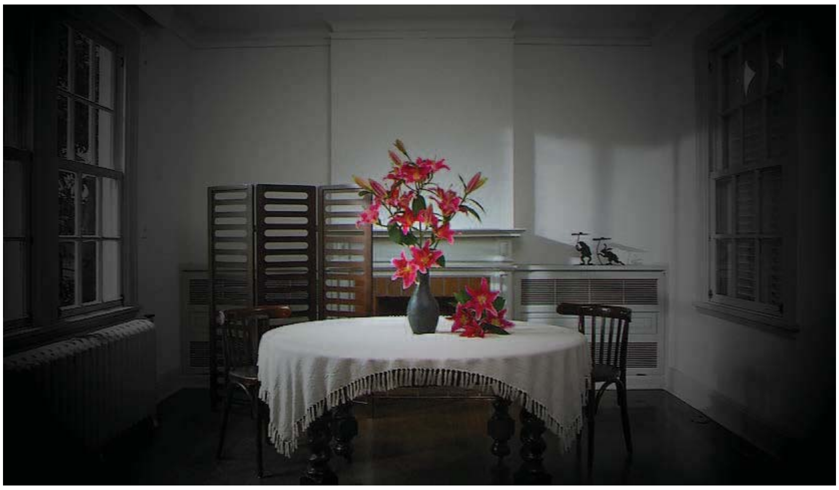


Figure II.3. Flowervase (`fl8_30`), 832x480, 30 fps.

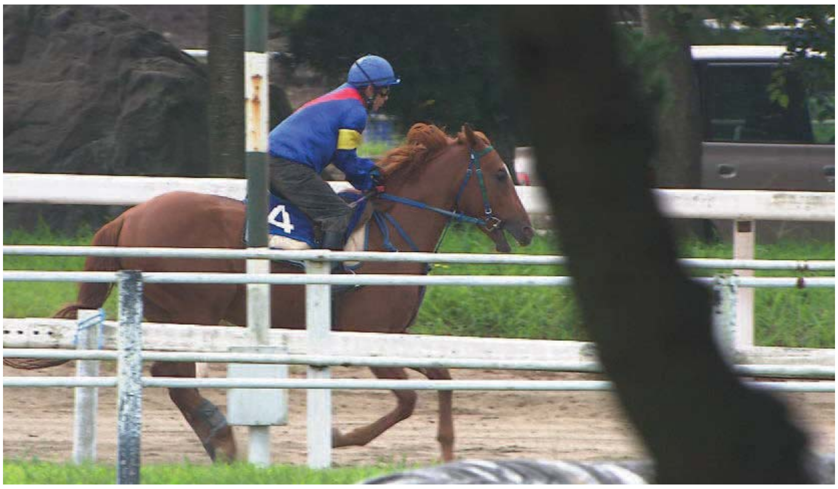


Figure II.4. Keiba (`ke8_30`), 832x480, 30 fps.

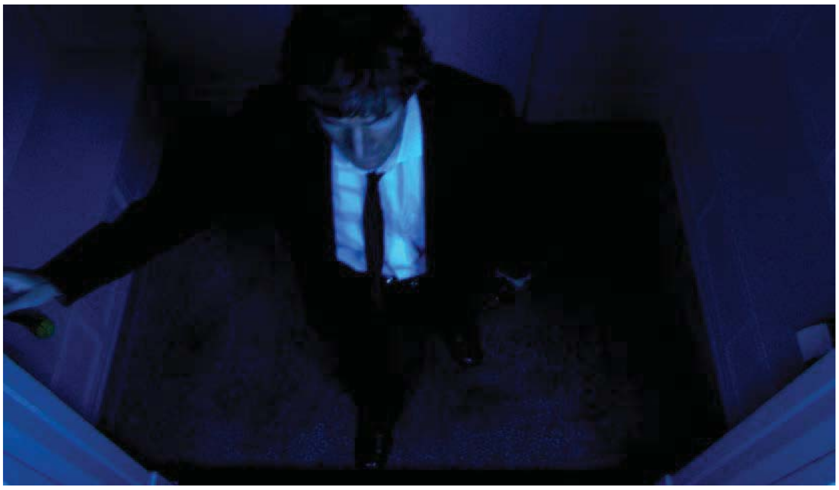


Figure II.5. Mobisode2 (`mo8_30`), 832x480, 30 fps.

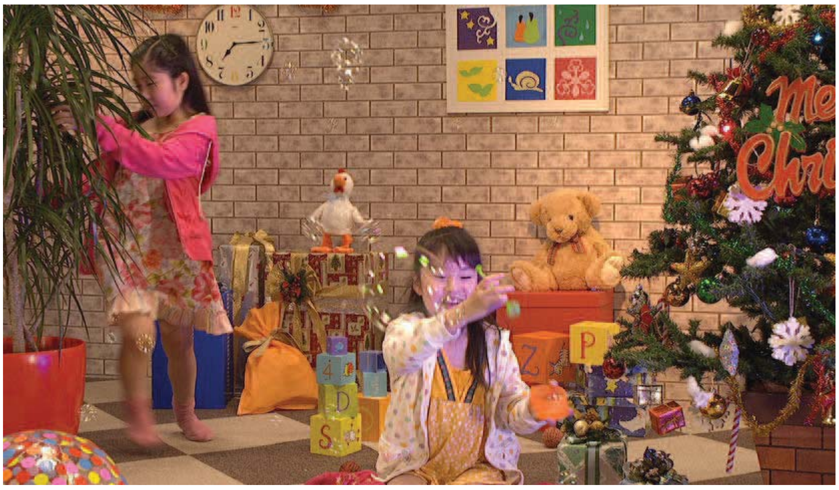


Figure II.6. PartyScene (`psc_25`), 832x480, 25 fps.

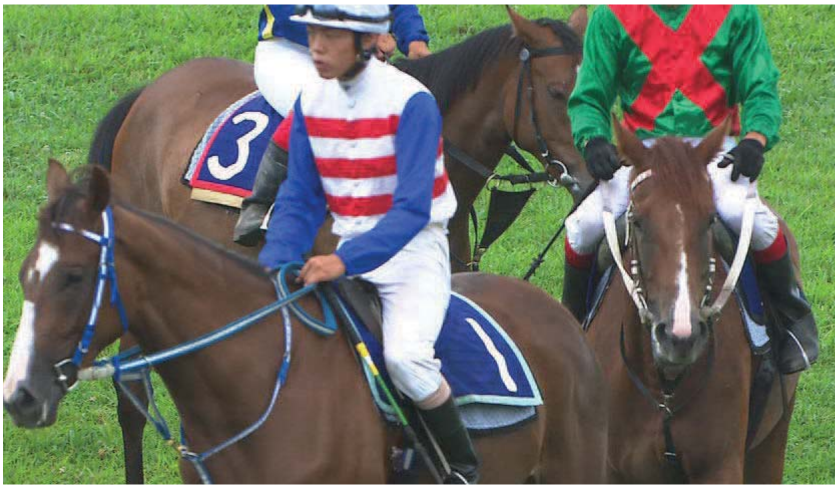


Figure II.7. RaceHorses (`rh8_30`), 832x480, 30 fps.

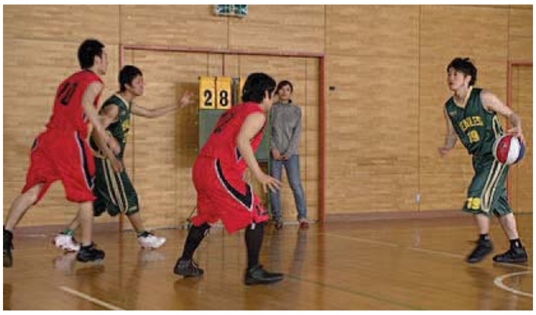


Figure II.8. BasketballPass (`bbp_25`), 416x240, 25 fps.

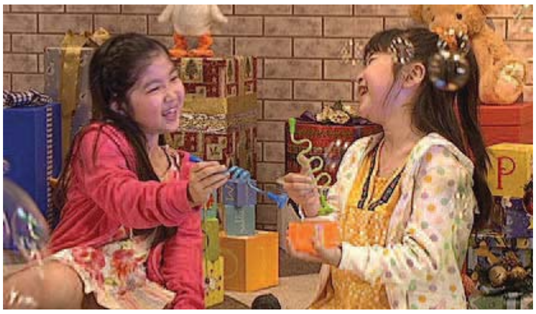


Figure II.9. BlowingBubbles (`blo_25`), 416x240, 25 fps.

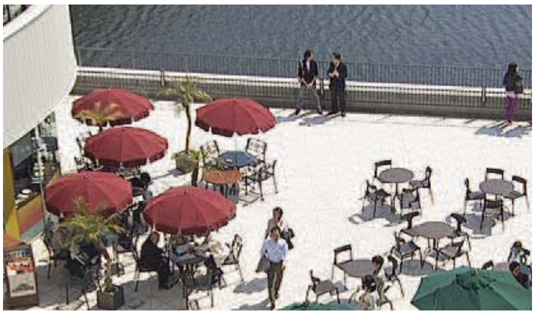


Figure II.10. BQSquare (`bqs_30`), 416x240, 30 fps.

Figure II.11. Flowervase (`fl4_30`), 416x240, 30 fps.

Figure II.12. Keiba (`ke4_30`), 416x240, 30 fps.

Figure II.13. Mobisode2 (`mo4_30`), 416x240, 30 fps.

Figure II.14. RaceHorses (`rh4_30`), 416x240, 30 fps.

Appendix III

Slice and tile partitions

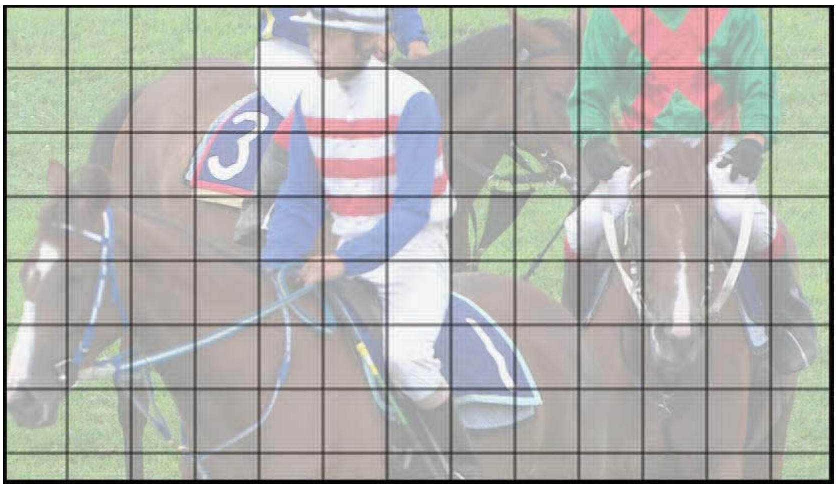


Figure III.1. 832x480, 1 slice/frame, 1 tile/frame.

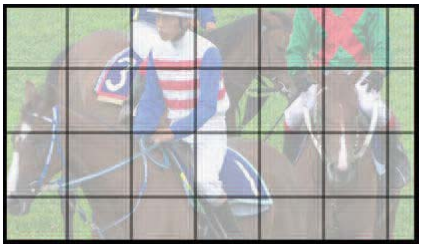
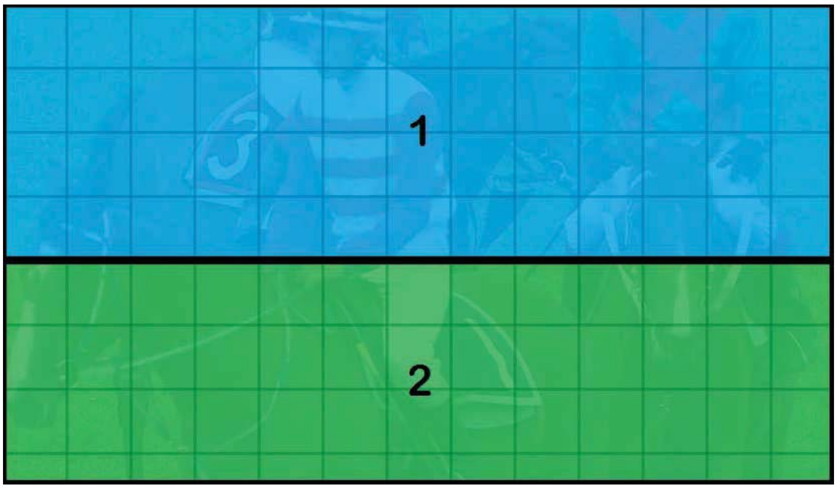
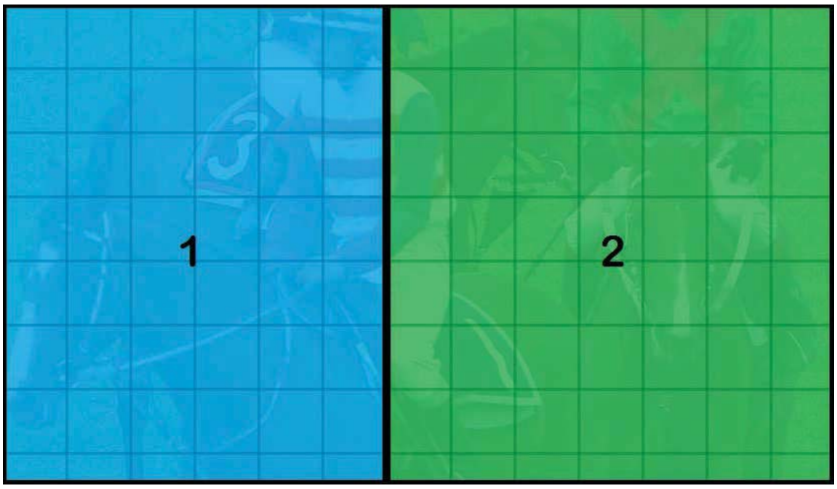


Figure III.2. 416x240, 1 slice/frame, 1 tile/frame.

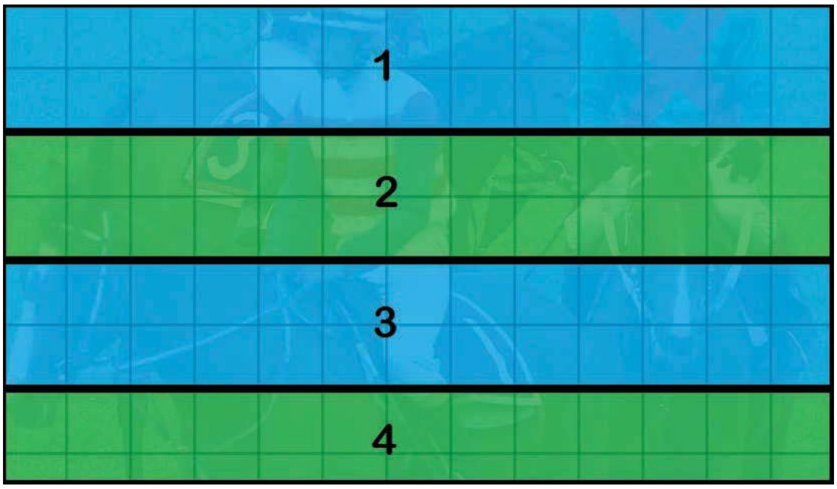


(a) 2 slices/frame

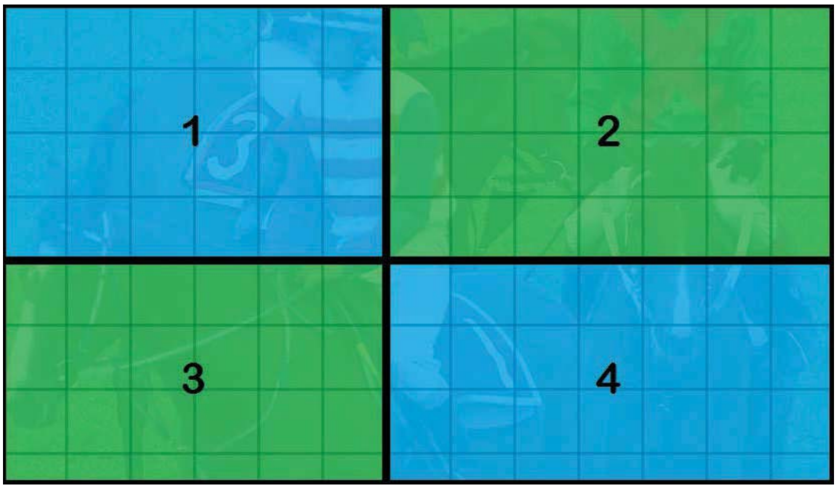


(b) 2 tiles/frame

Figure III.3. 832x480 partitions (I).

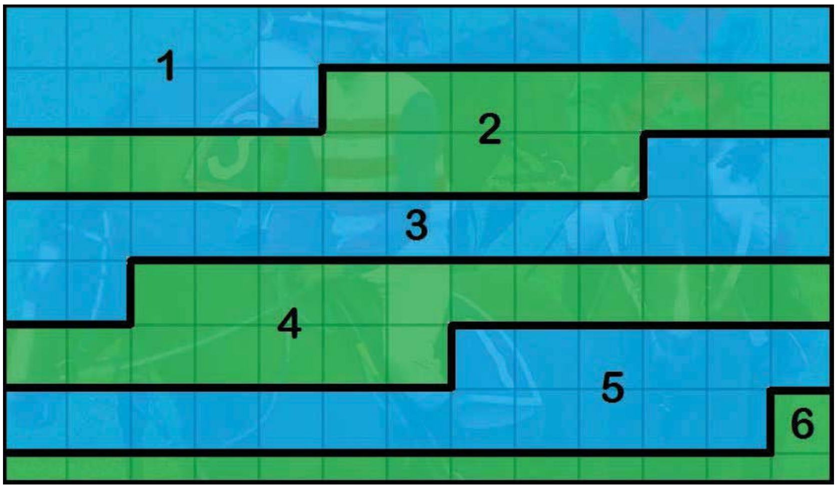


(a) 4 slices/frame

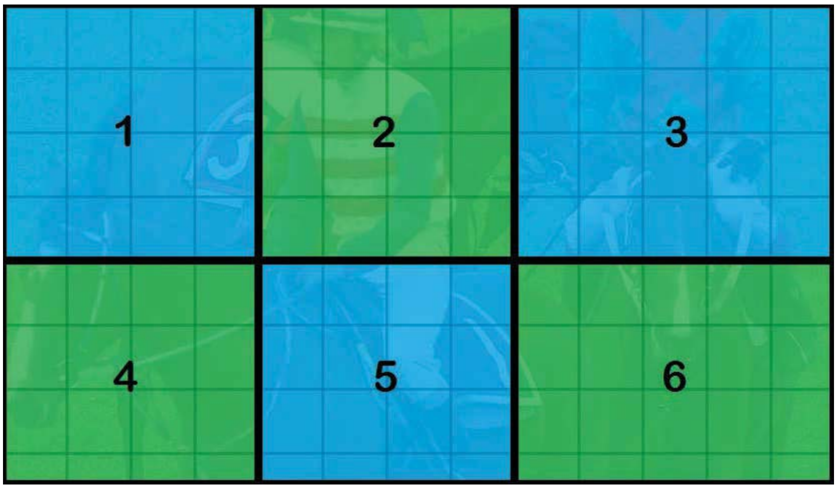


(b) 4 tiles/frame

Figure III.4. 832x480 partitions (II).

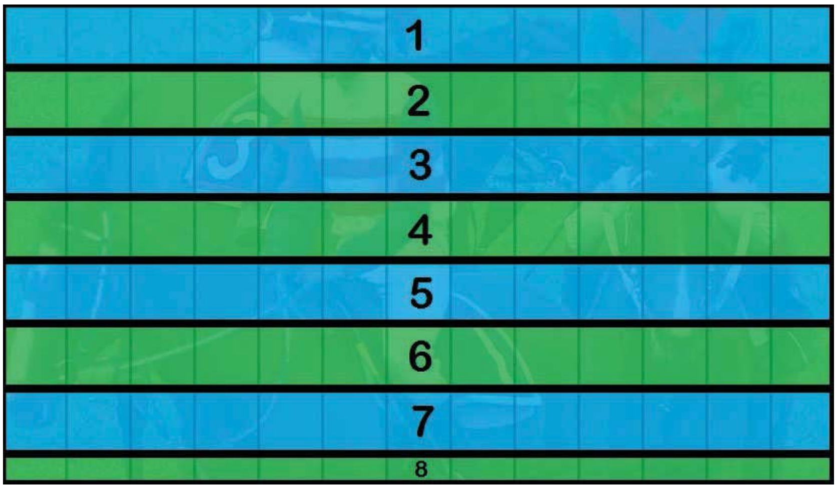


(a) 6 slices/frame

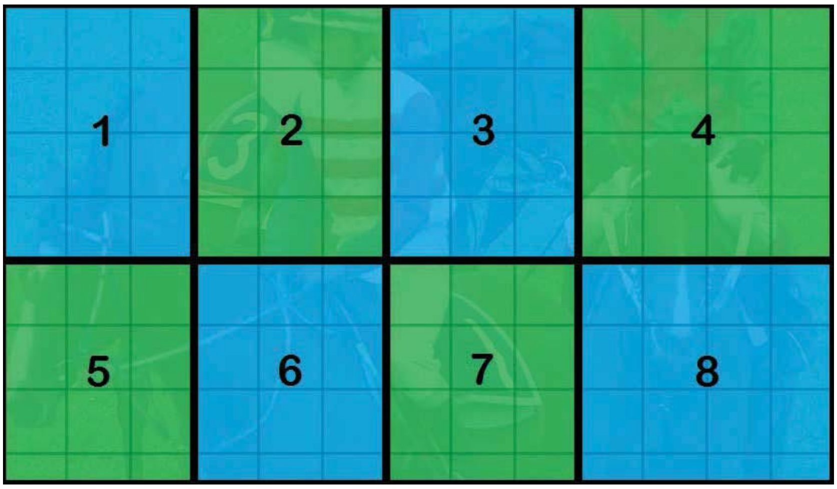


(b) 6 tiles/frame

Figure III.5. 832x480 partitions (III).

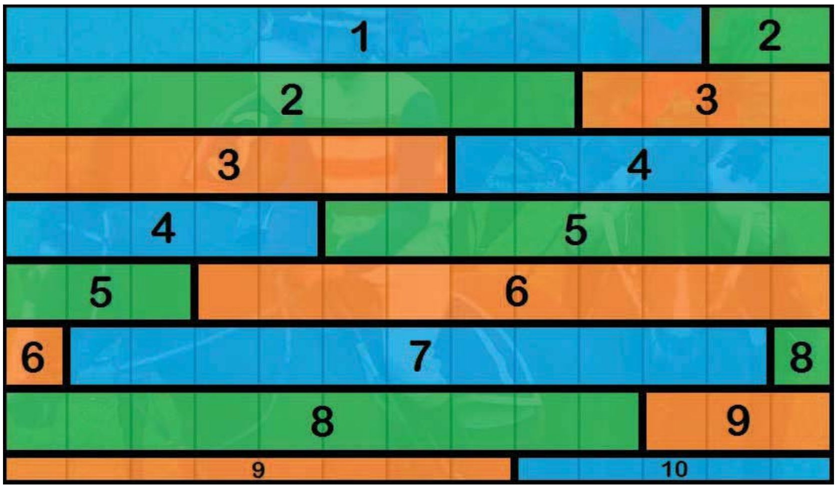


(a) 8 slices/frame

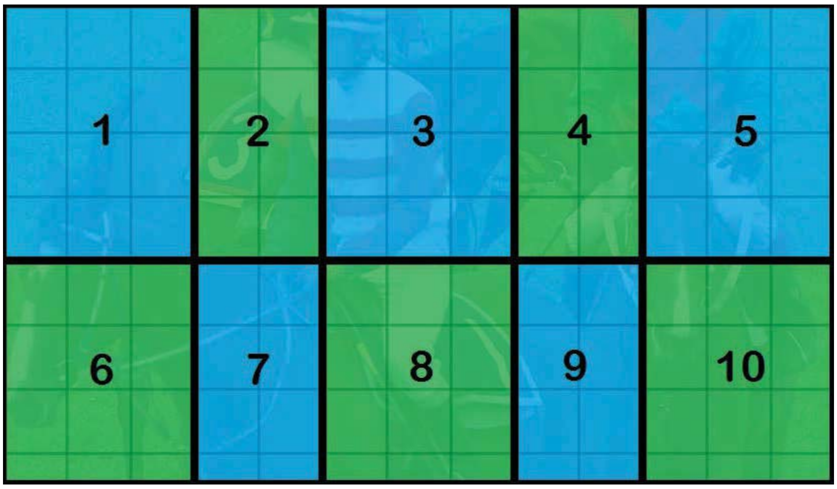


(b) 8 tiles/frame

Figure III.6. 832x480 partitions (IV).

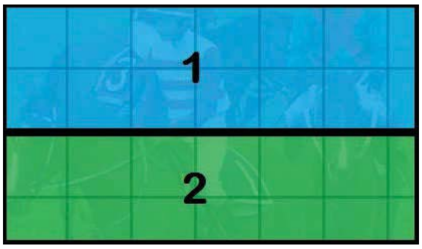


(a) 10 slices/frame

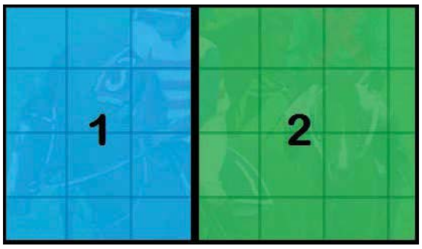


(b) 10 tiles/frame

Figure III.7. 832x480 partitions (V).

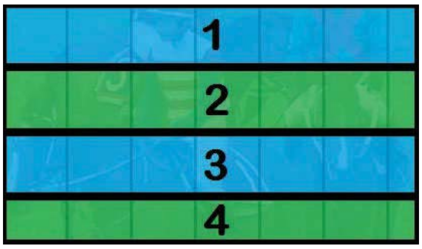


(a) 2 slices/frame

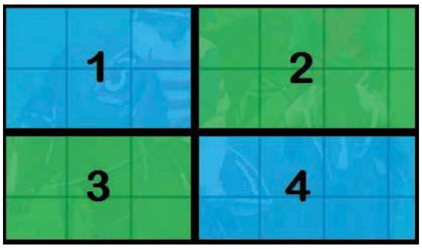


(b) 2 tiles/frame

Figure III.8. 416x240 partitions (I).

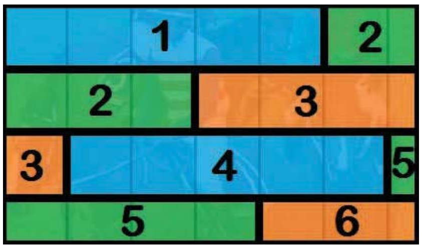


(a) 4 slices/frame

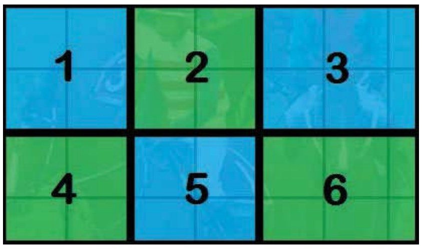


(b) 4 tiles/frame

Figure III.9. 416x240 partitions (II).

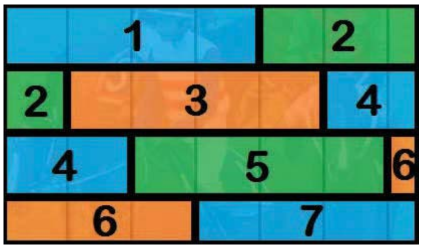


(a) 6 slices/frame

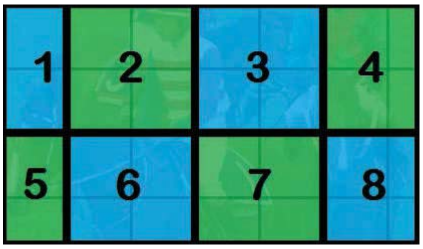


(b) 6 tiles/frame

Figure III.10. 416x240 partitions (III).

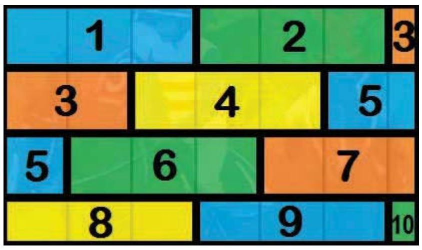


(a) 7 slices/frame

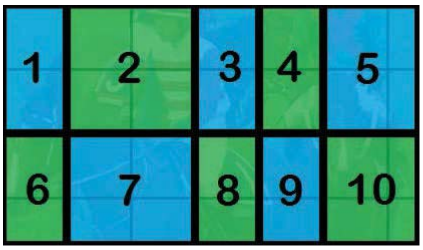


(b) 8 tiles/frame

Figure III.11. 416x240 partitions (IV).



(a) 10 slices/frame



(b) 10 tiles/frame

Figure III.12. 416x240 partitions (V).

Bibliography

[1] E. Schoch, F. Kargl, M. Weber, and T. Leinmuller. Communication Patterns in VANETs. *Communications Magazine, IEEE*, 46(11):119–125, November 2008.

[2] http://www.transportation.gov/. U.S. Department of Transportation (DOT).

[3] http://www.soumu.go.jp/english/. Ministry of Internal Affairs and Communications (MIC).

[4] http://www.mlit.go.jp/en/. Ministry of Land Infrastructure and Transport (MLIT).

[5] http://www.nilim.go.jp/english/eindex.htm. National Institute for Land and Infrastructure Management (NILIM).

[6] VSC. Vehicle Safety Communications: Final Report. *DOT HS 810 591*, April 2006.

[7] VSC-A. Vehicle Safety Communications - Applications: Final Report. *DOT HS 811 492A*, September 2011.

[8] http://www.itsforum.gr.jp/E_index.html. ITS Info-communications Forum.

[9] http://ertico.com/. Intelligent Transportation Systems and Services for Europe (ERTICO) - ITS Europe.

[10] https://www.car-2-car.org/. Car 2 Car Communications Consortium.

[11] http://www.eucar.be/. European Council for Automotive R&D (EUCAR).

[12] Ram Kandarpa, Mujib Chenzaie, Justin Anderson, Jim Marousek, Tim Weil, Frank Perry, Ian Schworer, Joe Beal, and Chris Anderson. Vehicle Infrastructure Integration Proof-of-Concept Technical Description: Infrastructure. *US DOT*, February 2009.

[13] http://www.its.dot.gov/safety/v2v_comm_plan.htm. Vehicle-to-Vehicle (V2V) Communications for Safety, Research Plan.

[14] http://www.go-etc.jp/english/index.html. ETC portal site GO!ETC.

[15] Igor Paromtchik and Christian Laugier. The Advanced Safety Vehicle Programme. *INRIA*, May 1998.

[16] S. Oyama. ITS Radio Communication in Japan. In *1st IEEE International Symposium on Wireless Vehicular Communications*, WiVEC '07, October 2007.

[17] R. Bossom, R. Brignolo, and Ernst T. D31 European ITS Communication Architecture - Overall Framework Proof of Concept Implementation. Technical Report FP6-027377, Information Society Technologies, Munich (Germany), October 2008.

[18] http://www.geonetproject.eu/. GeoNet project.

[19] IEEE Standard for WirelessMAN-Advanced Air Interface for Broadband Wireless Access Systems. *IEEE Std 802.16.1-2012*, pages 1–1090, Sept 2012.

[20] ETSI. Intelligent Transport Systems (ITS); Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band. Technical Report ETSI EN 302 663 V1.2.0, European Telecommunications Standards Institute (ETSI), Sophia Antipolis (France), November 2012.

[21] ARIB. Dedicated Short-Range Communication System. Technical Report ARIB STD-T75, Association of Radio Industries and Businesses (ARIB), Tokyo (Japan), September 2001.

[22] ARIB. DSRC Application Sub-Layer. Technical Report ARIB STD-T88, Association of Radio Industries and Businesses (ARIB), Tokyo (Japan), May 2004.

[23] IEEE Standard for Information technology - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2012*, pages 1–2793, March 2012.

[24] IEEE Standard for Information technology - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Amendment 6: Wireless Access in Vehicular Environments. *IEEE Std 802.11p-2010*, pages 1–51, July 2010.

[25] IEEE Guide for Wireless Access in Vehicular Environments (WAVE) - Architecture. *IEEE Std 1609.0-2013*, 2013.

[26] IEEE Standard for Wireless Access in Vehicular Environments (WAVE) - Security Services for Applications and Management Messages. *IEEE Std 1609.2-2013*, 2013.

[27] IEEE Standard for Wireless Access in Vehicular Environments (WAVE) - Networking Services. *IEEE Std 1609.3-2010*, 2010.

[28] IEEE Standard for Wireless Access in Vehicular Environments (WAVE) - Multi-channel Operation. *IEEE Std 1609.4-2010*, 2010.

[29] IEEE Standard for Wireless Access in Vehicular Environments (WAVE) - Over-the-Air Electronic Payment Data Exchange Protocol for Intelligent Transportation Systems (ITS). *IEEE Std 1609.11-2010*, 2010.

[30] IEEE Standard for Wireless Access in Vehicular Environments (WAVE) - Identifier Allocations. *IEEE Std 1609.12-2012*, 2012.

[31] Benjamin Bross, Woo-Jin Han, Jens-Rainer Ohm, Gary J. Sullivan, Ye-Kui Wang, and Thomas Wiegand. High Efficiency Video Coding (HEVC) text specification draft 10. Technical Report JCTVC-L1003, Joint Collaborative Team on Video Coding (JCT-VC), Geneva (Switzerland), January 2013.

[32] ITU-T and ISO/IEC JTC 1. Advanced Video Coding for Generic Audiovisual Services. *ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC) version 16*, 2012.

[33] T. Stockhammer, M.M. Hannuksela, and T. Wiegand. H.264/AVC in Wireless Environments. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):657–673, July 2003.

[34] Bernd Girod and Niko Farber. Error-Resilient Standard-Compliant Video Coding. In *Signal Recovery Techniques for Image and Video Compression and Transmission*, pages 175–197. Springer US, 1998.

[35] S.K. Bandyopadhyay, Zhenyu Wu, P. Pandit, and J.M. Boyce. An Error Concealment Scheme for Entire Frame Losses for H.264/AVC. In *Sarnoff Symposium, 2006 IEEE*, pages 1–4, March 2006.

[36] Ziguan Cui, Zongliang Gan, Xuefang Zhan, and Xiuchang Zhu. Error Concealment Techniques for Video Transmission over Error-prone Channels: a Survey. *J Comput Inf Syst*, 8(21):8807–8818, 2012.

[37] Xingjun Zhang, Xiaohong Peng, S. Fowler, and Dajun Wu. Robust H.264/AVC Video Transmission using Data Partitioning and Unequal Loss Protection. In *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, pages 2471–2477, June 2010.

[38] Stephan Wenger. H.264/AVC over IP. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):645–656, July 2003.

[39] Mohammad Amin Shokrollahi. Raptor Codes. *IEEE Transactions on Information Theory*, 52(6):2551–2567, 2006.

[40] Mohammad Amin Shokrollahi and Michael Luby. Raptor Codes. *Foundations and Trends in Communications and Information Theory*, 6(3-4):213–322, 2009.

[41] Michael Luby. LT Codes. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 271–280, 2002.

[42] M. Luby, A. Shokrollahi, M. Watson, and T. Stockhammer. Raptor Forward Error Correction Scheme for Object Delivery. In *IETF RMT Working Group, Work in Progress*. RFC 5053, 2007.

[43] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, and L. Minder. RaptorQ Forward Error Correction Scheme for Object Delivery. In *IETF RMT Working Group, Work in Progress*. RFC 6330, 2011.

[44] B. Oztas, M.T. Pourazad, P. Nasiopoulos, and V.C.M. Leung. A Study on the HEVC Performance over Lossy Networks. In *Electronics, Circuits and Systems (ICECS), 2012 19th IEEE International Conference on*, pages 785–788, Dec 2012.

[45] Stephan Wenger. NAL Unit Loss Software. Technical Report JCTVC-H0072, Joint Collaborative Team on Video Coding (JCT-VC), San Jose, February 2012.

[46] Kostas E. Psannis. HEVC in wireless environments. *Journal of Real-Time Image Processing*, pages 1–8, 2015.

[47] Hao Liu, Wenjun Zhang, and Xiaokang Yang. Error-resilience Packet Scheduling for Low Bit-rate Video Streaming over Wireless Channels. In *Proceedings of the 2006 IEEE International Symposium on Circuits and Systems. ISCAS 2006.*, pages 1 – 4, May 2006.

[48] James Nightingale, Qi Wang, and Christos Grecos. HEVStream: A Framework for Streaming and Evaluation of High Efficiency Video Coding (HEVC) Content in Loss-prone Networks. *IEEE Transactions on Consumer Electronics*, 58(2):404–412, May 2012.

[49] N. Vineeth and H.S. Guruprasad. A Survey on the Techniques Enhancing Video Streaming in VANETs. *International Journal of Computer Networking, Wireless and Mobile Communications*, 3(4):37–46, October 2013.

[50] N. Qadri, M. Altaf, M. Fleury, and M. Ghanbari. Robust Video Communication over an Urban VANET. *Mobile Information Systems*, 6(3):259–280, 2010.

[51] Alvaro Torres, Carlos T. Calafate, Juan-Carlos Cano, Pietro Manzoni, and Yusheng Ji. Evaluation of flooding schemes for real-time video transmission in VANETs. *Ad Hoc Networks*, 24(0):3–20, 2015.

[52] M. Pasin, M. Petracca, P. Bucciol, A Servetti, and J.C. De Martin. Error Resilient Real-Time Multimedia Streaming over Vehicular Networks. In *DSP Workshop for In-Vehicle Systems and Safety*, Dallas, Texas, USA, July 2009.

[53] Carlos T. Calafate, Giancarlo Fortino, Sascha Fritsch, Janio Monteiro, Juan-Carlos Cano, and Pietro Manzoni. An Efficient and Robust Content Delivery Solution for IEEE 802.11p Vehicular Environments. *Journal of Network and Computer Applications*, 35(2):753 – 762, 2012.

[54] Frank Bossen. Common Test Conditions and Software Reference Configurations. Technical Report JCTVC-L1100, Joint Collaborative Team on Video Coding (JCT-VC), Geneva (Switzerland), January 2013.

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

[56] Gisle Bjøntegaard. Improvements of the BD-PSNR model. Technical Report VCEG-M33, Video Coding Experts Group (VCEG), Berlin (Germany), July 2008.

[57] Qualcomm (R) RaptorQ (TM) Evaluation Kit, http://www.qualcomm.com/solutions/multimedia/media-delivery/raptor-evaluation-kit.

[58] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4):128–138, December 2012.

[59] OMNeT++ Discrete Event Simulator, http://www.omnetpp.org.

[60] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing*, 10(1):3–15, January 2011.

[61] MiXiM - Mixed Simulator, http://mixim.sourceforge.net.

[62] TraCI - Traffic Control Interface, http://sourceforge.net/apps/mediawiki/sumo/index.php?title=TraCI.

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

[64] INET Framework package, http://inet.omnetpp.org.

[65] C. Sommer, D. Eckhoff, R. German, and F. Dressler. A computationally inexpensive empirical model of ieee 802.11p radio shadowing in urban environments. In *Wireless On-Demand Network Systems and Services (WONS), 2011 Eighth International Conference on*, pages 84–90, Jan 2011.