TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

SERGIO MORESCHINI

CHANNEL RESOURCE ALLOCATION FOR MULTI-CAMERA VIDEO STREAMING IN VEHICULAR AD-HOC NETWORKS

Master of Science thesis

Examiner: Prof. Karen Egiazarian
Examiner: Dr. Evgeny Belyaev
Examiners and topic approved by the
Faculty Council of the Faculty of
Computing and Electrical Engineering
on 3rd February 2016

ABSTRACT

SERGIO MORESCHINI: Channel resource allocation for multi-camera video streaming in vehicular ad-hoc networks
Tampere University of Technology
Master of Science thesis, 77 pages
February 2016
Major: Signal Processing
Examiners: Prof. Karen Egiazarian, Dr. Evgeny Belyaev
Keywords: VANET, video coding, SKYPE, channel, network

This thesis studies the problem of channel resource allocation in a vehicular video communication system. The goal of the work is to investigate the allocation of resources in the case when each user estimates the channel without coordination from other users and then transmits the video data.

The analysis in this work has been carried out in two different stages. The first stage, reproduced the simulations conducted in early studies and then extended. In the second stage, an environment has been recreated based on the specifications of the standard IEEE 802.11p where the antennas are static and close to each other, so that packet losses can be caused mostly by congestion and collisions in random multiple access channels. To set up a network completely based on such standards specific antennas adapted to work at the specific frequency have been employed. At both stages, in order to transmit a video stream in the network and to collect the channel estimation information, an instant messaging software has been exploited, among those available we chose to use Skype.

This thesis shows results from simulations performed in a real-world environment. These prove that the behavior from the different users in the network has a direct effect on the quality perceived by the users themselves. From this study it was possible to define a new transmission system which involves greater knowledge from users regarding the network. Using this system it would be possible to achieve a more stable and fair level of visual quality for all users involved in the system.

PREFACE

This project was carried out at the Department of Signal Processing of Tampere University of Technology from August 2015 to February 2016, under the supervision of Prof. Karen Egiazarian and Dr. Evgeny Belyaev. I am infinitely grateful to them for giving me the opportunity to carry out this work, but especially, for always providing me with all the support needed. I express my gratitude to my supervisors in Roma Tre University: Dr. Federica Battisti and Dr. Marco Carli who allowed me to start this project and return to Tampere. I would like to make a special mention to Cristóvão Antunes Cruz for the enormous support provided. I also would like to thank the staff and colleagues at the Department of Signal Processing for making me feel like part of a big family.

A different kind of thanks is intended for those who have shared this *journey* with me in Italy and/or Finland. Of these, first of all, I feel the need to appoint Emanuele Palma, Andrea Ceccacci, Francesco Bottoni, Audrey Daudon and Luca Apicella, their advice and motivational push were priceless. Thanks to my "Italian" friends in Finland, they have been a constant source of inspiration, especially Andrea Milanti, Dr. Waqar Hussain, Lucio Azzari, Bruno Di Buó, Marco D'Ignazio, Davide Fantozzi and Francesco Di Capua.
Thanks to "The Panzed" and the other friends from Rome for making distance just a number. Thanks to my Erasmus Families from 2014 and 2015 for sharing with me unforgettable experiences, memories and an unbreakable bond.

The greatest of thanks is nevertheless dedicated to my parents, my brother and my grandparents, who were able to tolerate the distance and provided me with all the support that was humanly possible to them.

Tampere, 24.2.2016

Sergio Moreschini

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND SYMBOLS

BMA	Block Matching Algorithm
CCH	Control Channel
$CDMA$	Code Division Multiple Access
$CSMA$	Carrier Sense Multiple Access
$CSMA/CA$	Carrier Sense Multiple Access with Collision Avoidance
DCT	Discrete Cosine Transform
$DSRC$	Dedicated Short Range Communication
$FDMA$	Frequency Division Multiple Access
FEC	Forward Error Correction
FPS	Frame per Second
GOP	Group of Pictures
$IEEE$	Institute of Electrical and Electronic Engineers
$JPEG$	Joint Photographic Expert Group
MAC	Medium Access Control
MDS	Maximum Distance Separable
NLE	Non-Linear (Video) Editing
OBU	On Board Unit
$OFDM$	Orthogonal Frequency Division Multiplexing
PDU	Packet Data Unit
$PLCP$	Physical Layer Convergence Protocol
PMD	Physical Medium Access
QOS	Quality of Service
RS	Reed-Solomon
RSU	Road Side Unit
SCH	Service Channel
$TDMA$	Time Division Multiple Access
$V2I$	Vehicle-to-Infrastructure (communication)
$V2V$	Vehicle-to-Vehicle (communication)
$VANET$	Vehicular Ad-hoc Networks
$WAVE$	Wireless Access in Vehicular Environments

1. INTRODUCTION

In early 2000s, Vehicular Ad hoc Networks were catalogued as a part of MANETs, while today this is a widely studied topic which improves the safety, efficiency and entertainment of road travel services for both drivers and passengers. VANETs are thus defined as:

> "a class of wireless network, spontaneously formed between moving vehicles equipped with wireless interfaces that could have similar or different radio interface technologies, employing short-range to medium range communication systems. A VANET is a form of mobile ad hoc network, providing communications among nearby vehicles and between vehicles and nearby fixed equipment on the roadside [1]".

The standard that regulates the access to the wireless channel is IEEE 802.11p. The latter, together with the standards IEEE 1609.2, IEEE 1609.3 and IEEE 1609.4, constitutes what is currently the standard used in vehicular communications and is known as: Wireless Access in Vehicular Environment (WAVE). Some of the categories in which VANETs applications can be classified are:

- *Safety-related* when, as stated in [2], vehicles "exchange status information to increase safety awareness". Examples of these are traffic signal violation warning, curve speed warning, emergency control brake light, pre-crash sensing, cooperative forward collision warning, left turn assistant, lane-change warning and stop sign movement assistant.
- *Transportation efficiency*, developed to improve the flow of vehicles through services such as enhanced route guidance and navigation, green light optimal speed advisory and lane merging assistants.
- *Infotainment*, to provide the driver and passengers with non-safety-related

information and entertainment such as tolling, point of interest notifications, fuel consumption management, podcasting and wireless Internet Access [3].

In a vehicular communication environment there are usually two types of devices: On Boar Units (OBUs), which are special devices designed for transmission/reception of information, and the Road Side Units (RSUs) which are, as the word suggests, devices placed at the side of the carriageway developed to allow the transmission with the infrastructure network.

We assume that in this kind of network each user is provided with a series of sensors connected to its OBU, including a camera, and tries to send videos over the network. In addition, we assume that each user is capable of estimating the available bandwidth by themselves. Once the unit is aware of the information about the channel, it is therefore able to allocate the resources for the channel by setting the sending rate and choosing the appropriate protective mechanisms.

1.1 Objective and Scope of the Work

The main objective of this work is to provide an analysis of a channel resource allocation in a vehicular environment. The goal of the thesis is to investigate resource allocation in a 802.11p-based network in the case when each user independently tries to estimate the bandwidth. This project conducts an analysis based on video streaming in a vehicular environment.

1.2 Thesis Outline

This thesis is composed of 5 chapters. In Chapter 2, the theoretical concepts, vital to the understanding of the analysis performed in the thesis, will be introduced. Chapter 3 will introduce the software used in the practical implementation of the thesis work and the procedure by which the network based on the standard IEEE 802.11p has been created. In Chapter 4, the results will be proposed and also a solution to the problems detected in the analysis. Conclusions will be summarized in Chapter 5.

2. THEORETICAL BACKGROUND

2.1 System Overview

Let us consider a model similar to the one proposed in [4] and shown in Figure 2.1 where:

- M Road Side Units (RSU) are placed near the carriageway, those are connected both to the others and to the Internet.
- The distance between two RSUs is equal to the communication range.
- Since usually vehicles follow each other with irregular intervals, selection of few of them can be categorized as "scheduled".

Let us imagine a situation in which a vehicle wants to send video to a RSU. The block diagram in Figure 2.2 shows the different activities performed in the OBU in order to transmit the video.

1. Once the video is captured, it is necessary to perform a compression in order to reduce the amount of data it will send.
2. It is essential to protect the stream so that the decoder has the ability to recover data by using inter-packet forward error protection (FEC) via erasure codes.
3. Each user estimates the bandwidth and then, based on such estimation, selects the sending bit rate and protection scheme for a video bit stream.
4. Once the system is ready to transmit, it must follow the specifications provided by the transmission protocol. In the case of 802.11p, specifications are related to two different layers, thus different actions in different levels are carried out.

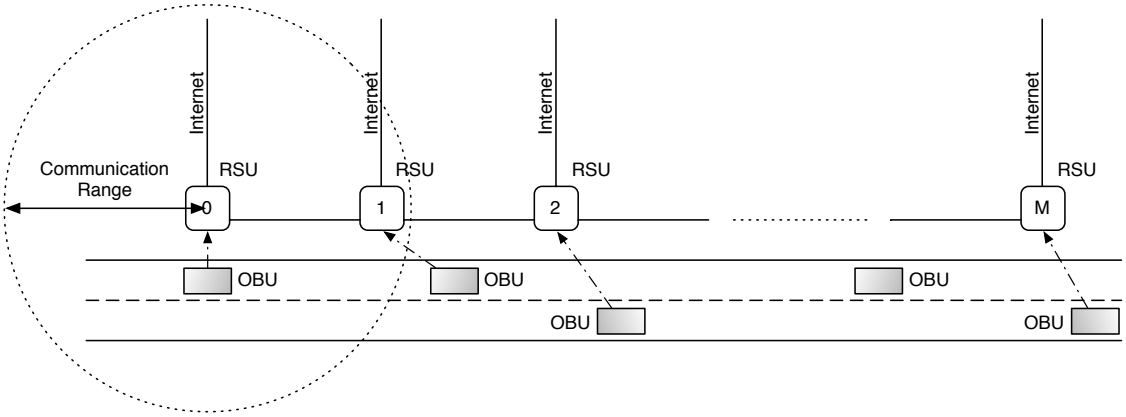


Figure 2.1 *Considered scenario*

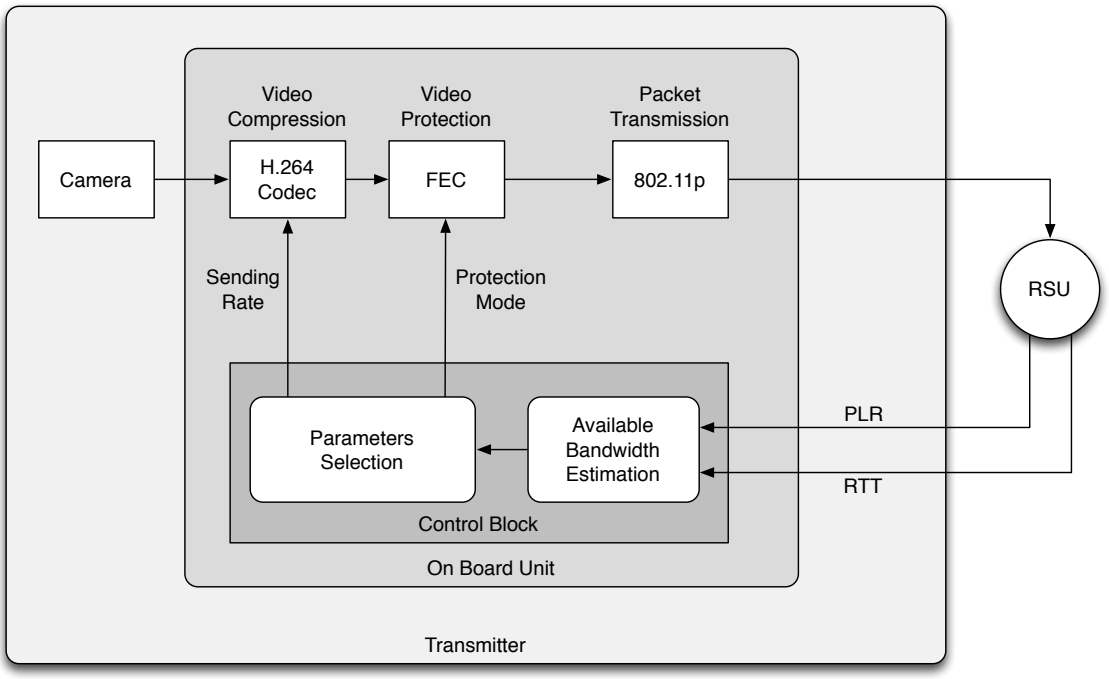


Figure 2.2 *On Board Unit*

(a) At the MAC layer, where the medium access scheme is defined, packets are prepared to be forwarded to the Physical layer;

(b) The Physical layer performs the transmission procedure.

In this chapter the theoretical background which describes the above points will be introduced. For this purpose the rest of the chapter is organized as follows. Section 2.2 defines the standard IEEE 802.11p and its Physical layer. Section 2.3 provides a brief comparison of the multiple access protocols in order to introduce the MAC layer of the standard 802.11p. In Section 2.4, the methods by which video compression is achieved are described, for this purpose the JPEG compression is described and the concepts of Motion Estimation and Motion Compensation are introduced. Section 2.5 illustrates a situation where packets are lost during the transmission. It then defines the probability of not restoring lost packets at the receiver. Section 2.6 is dedicated to the estimation of the available bandwidth.

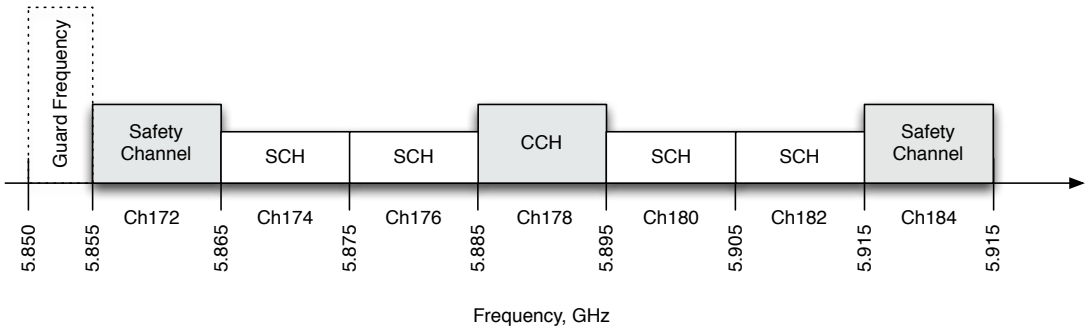


Figure 2.3 *IEEE 802.11p Channel Frequency band*

2.2 IEEE 802.11p

2.2.1 Dedicated Short Range Communication

In 1999, the U.S. Federal Communication Commission allocated 75 Mhz for Dedicated Short Range Communication (DSRC) spectrum at 5.9 GHz for vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications in North America. It is divided into a guard band and seven 10 Mhz channels: four Service Channels (SCH), two safety-related channels and one (5585-5895 Mhz) Control Channel (CCH) (Figure 2.3). The latter, which is channel 178, is responsible for link establishment and the control of transmission. Its importance is, however, related to its nature of being the only channel shared between all WAVE devices, providing a common point for the nodes. Service channels are used for normal communications, those have been divided in 10 MHz channels as they are designed for a parallel use of different applications, however, two adjacent channels can be combined to form a channel 20 Mhz. Channel 172 and 184 are both safety-related channels, furthermore the first is for serious security solutions, while the second can be seen as a spare channel, as it works as a protection against congestion in the other channels.

The development of 802.11p is closely related to the introduction of DSRC. In [5] IEEE 802.11p is defined as:

> "a collection of Physical layer (PHY) specifications and media access control (MAC) for implementing WLAN in the 2.4, 3.6, 5 and 60 GHz frequency bands."

Table 2.1 *Modulation schemes and data rate [5]*

Modulation type	**BPSK**	**QPSK**	**16-QAM**	**64-QAM**
Coding Rate	1/2, 3/4	1/2, 3/4	1/2, 3/4	2/3, 3/4
Coded bit rate [Mbps]	6	12	24	36
Data Rate [Mbps]	3, 4.5	6, 9	12, 18	24, 27
Data bits per OFDM Symbol	24 , 36	48, 72	96, 144	192, 216

2.2.2 Physical Layer of IEEE 802.11p

The Physical layer can be further divided in the Physical Layer Convergence Protocol (PLCP) and the Physical Medium Access (PMD), the modulation employed is OFDM. The first one is responsible for communicating with the MAC layer, this is why it performs a convergence process necessary to transform the various Packet Data Unit (PDU) in a OFDM frame. The second, on the contrary, is related to the physical transmission medium.

The Protocol Packed Data Unit in this layer is composed of three fields: Preamble, Signal and Data Fields.

1. The preamble is always at the beginning of the frame and is thus necessary for correcting the frequency and timing offset.
2. The signal field is used to specify the transmission rate and information related to the length of the packet.
3. In the data field, the OFDM symbols are transported.

As previously introduced, the modulation technique used at the Physical layer is the Orthogonal Frequency Division Multiplexing. Each of the 52 subcarriers is modulated by using traditional modulation techniques as QAM and PSK. As can be seen from Table 2.1, the combination of a chosen code rate and a modulation type affects the data rate of the system. The great advantage of the OFDM is the orthogonality between the different subcarriers, due to which it is impossible for the signals carried by those to interfere with each other, thus making unnecessary the existence of guard bands among them.

The transmission process shown Figure 2.4 can be described as follows:

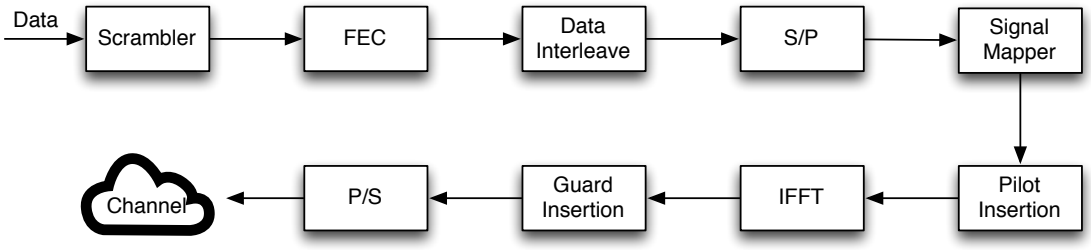


Figure 2.4 *PHY Transmitter Block Diagram*

- The data received from the data link layer is scrambled, not for encryption purposes, but to avoid long sequences of bit which may generate errors. For this purpose it employs several kinds of polynomials to produce sequences of 127 bits. It is however necessary to protect the data from other kinds of errors, this process is composed of two types of protection, the first in order to reduce the bit error rate and the second to reduce burst errors.

- The process of reduction of the bit error is performed by using FEC mechanisms. Such techniques allow the message to be encoded in a redundant way by using an error-correcting code. Along with this, 802.11p may, however, use convolutional codes to reduce such bit error rate caused by channel effects or interference.

- The output data from the previous block is thus interleaved by performing a permutation both in time and frequency domain in order to reduce burst errors cause by channel fading. The first is performed to prevent two successive bits to be encoded in two adjacent subcarriers; while the second allows two successive bits to be represented by the most and least significant bits of the chosen constellation.

- Data is converted form serial to parallel.

- The parallel stream is then grouped by using a mapper. The way in which it is grouped depends on the kind of modulation chosen, these, as shown in Table 2.1, can be: BPSK, QPSK, 16-QAM and 64-QAM.

- Before applying IFFT, it is necessary to choose the four carriers which are going to carry the pilot signal. Pilot symbols are used to detect the changes made in the signal and for channel estimation; pilot subcarriers are necessary

to increase the robustness in the receiver against frequency offsets and phase noise. Usually, of the 52 information carriers available, the four chosen for the pilot signal are: -21, -7, 21 and 7.

- Data can now be modulated into orthogonal carriers by performing an IFFT, more in particular the OFDM symbols are converted from frequency to time domain through this operation.
- The data is converted into a serial stream of bits to build the OFDM frame.
- In order to prevent interference caused by multi-path propagation, and to ensure the separation for the different sub-bands it is necessary to insert guard subcarriers on the OFDM spectrum side before sending the data in the channel.

As the two most important characteristics for a WAVE network are robustness and prompt response, it is therefore necessary to meet some latency requirements and ensure protection for effects such as: Rayleigh fading, frequency selective fading, delay spread and Doppler shift. For this reason at the receiver side the same operations are performed in reverse, but it also added an equalizer block to reduce non-linear effects.

If compared with the 802.11a protocol, 802.11p presents some differences. The operative frequency is 5.9 GHz instead of 5 GHz, moreover, as the duration of each OFDM is doubled from 8 to 4 microseconds, the bandwidth used is 10 MHz instead of 20 MHz and the guard interval between OFDM symbols is doubled. This results in an improved protection against Intersymbolic interference caused by the high mobility.

2.3 Random Multiple Access in 802.11p

2.3.1 Existing Multiple Access Protocols

The protocols are a set of rules and agreements among communication entities, enabling information transmission by using different kinds of variations of a physical quantity. Such rules, also known as standards, determine the syntax, semantics and synchronization in a communication. It is therefore essential to define proper schemes, in such a way that the information can be sent to achieve desirable performance characteristics. These access schemes are known in literature as *Multiple Access Protocols*.

A protocol can be cataloged as a conflict-free, only if it is able to ensure a successful transmission (i.e. no interference); the only possible way to perform it is through a channel allocation. A conflict-free protocol includes:

- Time Division Multiple Access *(TDMA)*: the channel is divided by assigning the entire bandwidth to a single user for a time range.
- Frequency Division Multiple Access *(FDMA)*: a fraction of the frequency is given to a user for the whole time.
- Code Division Multiple Access *(CDMA)*: part of the bandwidth is assigned for a fraction of time to each user as in a spread spectrum base system.

In a random access protocol a node always transmits at the full possible rate in the channel. When a collision between two or more nodes occurs, each node involved transmits repeatedly its frame until it reaches the destination without collision. Moreover, in order to increase its chances of finding the channel free, the node does not transmits the frame instantly, but it waits a random amount of time before retransmitting. This is the basic principle that characterizes the Aloha protocol. Another type of random access protocol is the Carrier Sense Multiple Access. As its name indicates, the protocol consists of two actions:

- **Carrier Sense:** Each station in the process of transmitting senses the medium and decides to transmit only if it is free (*listen before talking*).

***Table** 2.2 Parameters for different Application Categories in IEEE 802.11p*

AC	CW_{min}	CW_{max}	AIFSN
Background traffic (BK)	15	1023	9
Best Effort traffic (BE)	15	1023	6
Video traffic (VI)	7	15	3
Voice traffic (VO)	3	7	2

- **Multiple Access:** Despite the carrier sense it is still possible that two stations, detecting the channel as free, try to transmit simultaneously, and as the propagation time of the signals is not zero, it is possible that a station detects the channel as free even if another station has already started the transmission. This is why in CSMA the vulnerable period is equal to the transmission delay. When a collision occurs every user involved reschedules a retransmission of the collided packet randomly.

2.3.2 MAC Layer of IEEE 802.11p

The medium access scheme used at the MAC layer is: Enhanced Distributed Channel Access (EDCA), which is based on CSMA/CA technique. It is able to ensure a minimal distance between users, by sensing the channel using two techniques: the first, related to the energy detected by the antenna known as a *physical carrier sensing* and through a reservation technique known as a *virtual carrier sensing*. The latter, similar to the CSMA/CA techniques, takes more effort and is only used for the service channel. The goal of EDCA is not only to provide a collision avoidance but also to provide a Quality of Service (QoS). In order to ensure this, the protocol provides Access Classes (ACs), to which different Arbitration Inter-Frame Space Numbers (AIFSN) and different Contention Windows (CWs) values as shown in Table 2.2 correspond.

Backoff procedure

The backoff procedure is described in [6] as follows:

1. The node selects a random backoff time in an interval $[0, CW_{min} + 1]$;
2. Whenever a transmission fails the interval size will double until the value CW_{max} is reached;
3. This value will be decreased only when the channel will be sensed as free;
4. When the backoff value reaches the value 0, it will be allowed to send.

It is important to specify that 802.11p defines only the signalling techniques and interface functions at the MAC layer; other features are defined by other standards. As an example IEEE 1609.3, defines WSMP (Wave Short Message Protocol), which is an alternative to IP, introduced to set the channel and power in transmission. The protocol stack of a DSRC communication is therefore composed of different standards working in conjunction.

Transmission procedure

During transmission the operations performed include the following:

- When a packet is received by the MAC layer from the Logical Link Control, the Channel Router in the MAC layer checks the Ethertype field in it, where it specifies if the protocol is WSMP or IP.
- If the packet is detected as WSMP the channel number in the WSMP header is analyzed. Depending on whether it has been transmitted on a CCH or a SCH, and depending on the Access Category, it will be therefore sent in a proper queue (CCH or SCH) or discarded if the channel number results as not valid.
- On the contrary, if the packet is detected as IP, it will be directly sent in the proper queue of the SCH.

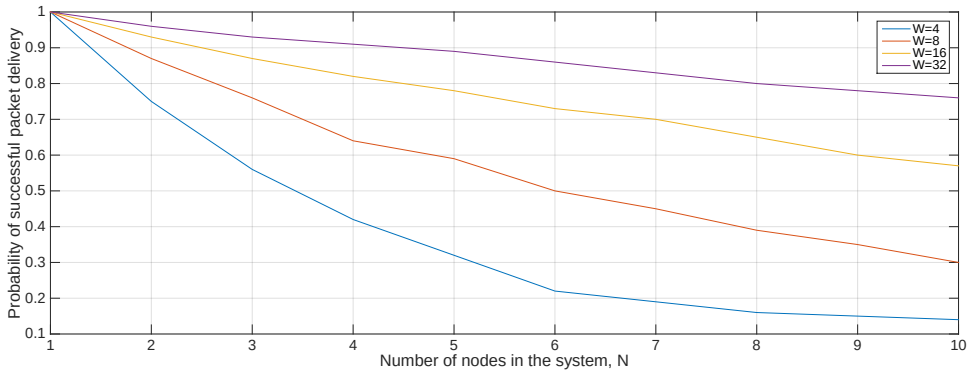


Figure 2.5 *Probability of successful broadcast on number of nodes [7].*

- After all the packets have been sent in the proper queue, the MAC layer must choose the packet to which the shortest backoff time has been assigned. It is thus necessary to check again the Ethertype field in order to know the rate and power to apply to the outgoing packet.
- Once known this, the MAC layer needs to communicate with the Physical layer:

  1. The Physical layer informs the MAC layer in regards to the free channel.
  2. The MAC layer answers back, for the purpose of setting the necessary parameters for transmission: power and rate.
  3. The MAC layer needs to know how much time is required for the transmission of the packet, it asks the Physical layer.
  4. The Physical layer answers back to the previous request.
  5. If the time exceeds, the remaining range in the channel (CCH or SCH) interval, that packet is temporarily stored and then transmitted at the beginning of the next time interval, if not is sent immediately.
  6. Following this is the effective exchange of the packet between MAC and PHY. It is performed through a series of exchanges of primitives between the two. An acknowledgement (ACK) of the correct transfer is always sent.
  7. After the package has been transferred in its entirety, the MAC layer terminates the connection.

Evaluation of the transmission performance

In [7], a model which evaluates the broadcasting performances on CCH in IEEE 802.11p was created. Figure 2.5, shows the results achieved in the previously mentioned work concerning the probability of success for the delivery of packets for different amounts of users. We can also state that the increasing number of users leads to a subsequent increase in the quantity of packets which are lost. Each curve represents a different value for the contention window W from where the user can randomly choose a backoff value to be assigned to the packet that is intended to send. Furthermore, this figure has been created by setting two fundamental parameters: data rate and packet length. The first has been fixed to 3 Mbps since it provides the highest level of robustness, the latter in order to provide a higher level of protection has been set to 500 bytes. It is therefore clear, from the figure, that by increasing the contention window the probability of correctly sent packets increases since the probability of collision between these is reduced. It must be stated, however, that a high value of W can lead to further losses due to expiry time.

2.4 Video Compression

2.4.1 The Need for Compression

Let us assume that, for example, we want to transmit a video in a communication system; the amount of bits per second transmitted for a high definition video with refresh rate of 30 Frames Per Second (FPS) is:

$$30\,\text{FPS} \times 1920 \times 1080\,\text{pixel/frame} \times 8 \times 3\,\text{bit/pixel} \cong 1.5\,\text{Gbps}. \tag{2.1}$$

This video stream must be transmitted on a vehicular network, since the lowest data rate supports approximately 3 Mbps, therefore a compression by a factor of 525:1 is required. In order to reduce the amount of data to transmit, and occasionally to limit signalling and control data, efficient video compression standards have been developed. Such compression algorithms are based on reduction of spacial (image coding) and temporal redundancy (motion compensation). In a vehicular environment, additional characteristics are also required, one of these for example is error robustness.

2.4.2 JPEG Compression Standard

The JPEG standard is a lossy compression algorithm applicable to a single image. It performs the compression by eliminating the colors at high frequencies, as the human eye is not able to see such colors. Figure 2.6 shows the main stages of the JPEG compression standard.

1. The first stage is called image preparation, used to pre-process the image before the actual encoding steps. Let us assume the image we want to compress is in the RGB 24-bit color format, the first step is applying a color conversion to the YUV color format. Usually this process is composed of two parts: in the first an RGB image is converted in the YUV 4:4:4 color format where each pixel is composed of one luminance component and two chrominance components. In the second, a conversion to the format YUV 4:2:0 is applied by calculating the average of the chrominance components of the image. After separating the luminance and chrominance components, the image is partitioned into blocks of size 8×8 pixels.

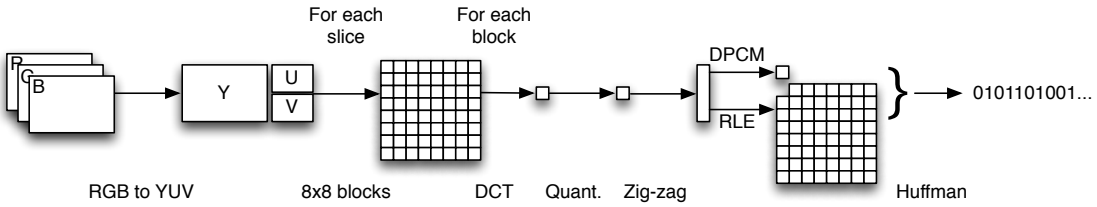


Figure 2.6 *JPEG compression*

2. Before applying the DCT transform, each block is level shifted by subtracting the amount 2^{n-1}, where 2^n represents the maximum number of grey levels in the image, then the DCT of each block is computed. It results in new blocks where low frequencies are located in the upper left and the high frequencies in the lower right.

3. The DCT coefficients are then Quantized. After this process most of the values at high frequencies are reduced to zero.

4. The coefficients are then reordered in a "zig-zag" order for the purpose of creating a 1-D sequence of quantized coefficients; at the end of the sequence there is a long series of zeros.

5. Since the DC coefficients show a high correlation between them, a differential encoding (DPCM) between the DC coefficient analyzed and the previous one is applied.

6. The resulted one-dimensional AC sequence is encoded by using a two step entropy encoding: first, a run-length coding is applied, then the output is converted to binary by a Huffman coding obtained with specific tables.

2.4.3 Video Coding Standard

In order to achieve a better compression/quality ratio than JPEG, the H.264/AVC video coding standard is applied in many applications, such as video streaming over the Internet. According to the standard, there are three types of frames in a video sequence:

- *I-frames (intra-coded images):* These are used for random access in a video stream, the intra-coded images, are fixed images which need to be encoded

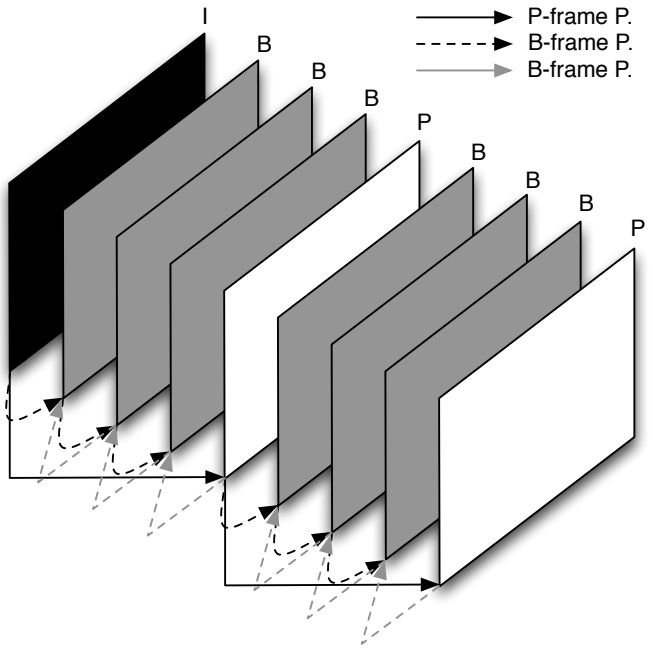


Figure 2.7 *Group of Picture.*

without any kind of connection to other images; a compression similar to JPEG is applied to this kind of images.

- *P-frames (Predictive-coded frames):* These require information from the previous frames (I or P) for encoding and decoding. The idea behind the P-frames encoding is a temporal redundancy. The amount of data used to create this type of frame consist of motion vectors and transform coefficients describing prediction correction. It involves the use of motion compensation.

- *B-frames (Bi-directionally predictive-coded frames):* need information from both the preceding frame and the subsequent in regards for both the encoding and decoding, on the other hand, this is the kind of frame that ensures the best compression rate. Together with the P-frame they are often known as inter-coded images.

The set of images that is grouped together has to be processed by the decoder in the forward mode and stored: it is known as *group of picture (GOP)*. The GOP structure is referred to by two numbers: the first is the distance between two anchor frames (I or P), the second is the distance between two full images (I-frames). Of

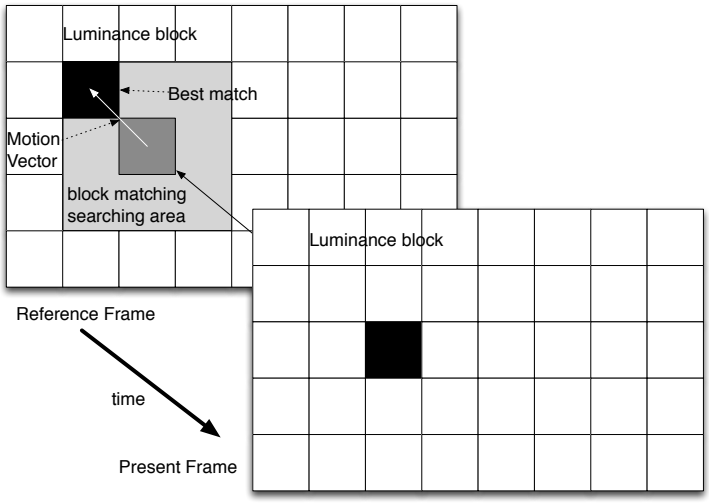


Figure 2.8 *Block-Based Motion Estimation and Compensation*

course the best resolution would be achievable through the use of only I-frames. Conversely, the compression is obtainable only with the introduction of B-frames.

2.4.4 Motion Estimation and Compensation

The goal of motion estimation is to obtain motion information from 3-D objects of the scene in order to describe those in the 2-D plane by the use of vectors. To reduce computation and storage complexity, motion parameters of objects are thus estimated by comparing one frame with the previous.

The motion-related problems can be summarized in three sections:

1. Motion detection is the process of detecting variations related to an object.
2. Motion segmentation is the process of separating objects in a video in order to identify the boundaries of such objects.
3. Motions estimation is the measurement of the motion parameters for each moving object in the scene.

The most popular method for motion estimation is the block matching algorithm (BMA): the frame is divided in blocks and by using correlation techniques, the best

match between the current frame block and the previous frame are searched. The difference between such frames is represented by a motion vector. Each block is therefore treated as an independent object for which it assumes a simple parametric motion.

An example is depicted in Figure 2.8, where, in order to reduce the computational cost of the operation it is assumed that the block can only be in a position adjacent to the previous. Like this, the motion vector chosen for the entire block, cannot exceed a maximum range for the horizontal and vertical components (in the example 1,1). It is therefore very important to choose a proper matching function since it has a direct impact on the computation complexity and the accuracy in the calculation of the motion vector; the most used are mean square error and mean absolute difference [8].

Motion Compensation is a technique that exploits knowledge of motion, obtained by the motion estimation, in order to compress data in a video. It is based on motion estimation since it exploits blocks from a past frame to construct a replica of the current frame. In the encoder, the prediction filter reads the previous frame and creates the motion compensation vectors. Since it is not always possible to recreate an image from the previous, along with the motion vector it also creates a compensation matrix of the prediction error. The decoder first applies the matrix containing the motion vector, then the prediction error compensation is applied to create the new P-frame.

In H.264/AVC this technique has been improved by allowing a sub-pixel-accuracy block matching. Furthermore, size and shape of the block can now be adjusted to facilitate the matching operations, however, if no matching block can be detected, an intra-coded macro block can be used. In this technique the individual frame is processed using the prediction of small blocks of pixels within each macro block, i.e. a match is attempted with the already encoded pixels which are located on the side of the block to be coded.

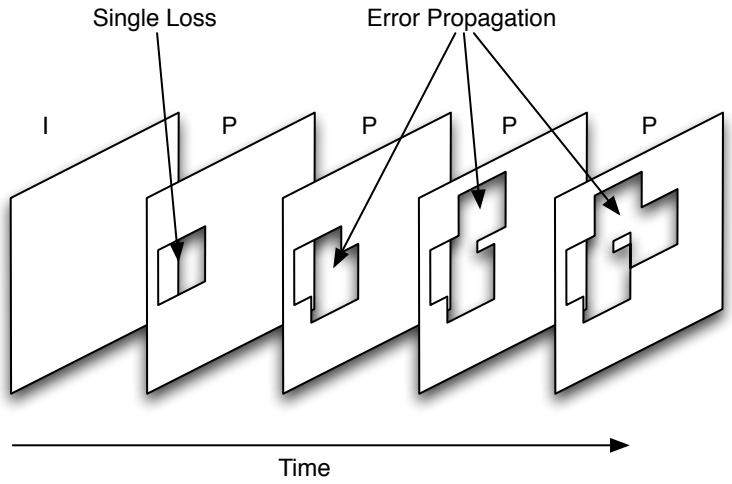


Figure 2.9 *Error propagation for motion-compensated video.*

2.5 Video Stream Protection at the Application Layer

2.5.1 Video Communication with Packet Loss

In an environment as the one shown in Section 2.1, it is very unlikely for a complete data stream to succeed in sending without any lost or corrupted packets. This can happen basically for three reasons:

1. Signal fading often caused by multi-path propagation;
2. Obstacles between the transmitter and receiver;
3. Collision of packets generated by an excessive number of simultaneous transmissions.

Let us imagine that in such an environment a certain vehicle is transmitting a video stream to another, in addition, this video has been compressed using the techniques described in the previous section. If the data sent suffered any loss, it would be impossible for the receiver to reconstruct part of the video. Moreover, as shown in Figure 2.9, such loss, in most cases leads to a broadening of the error until the next I-frame is reached. The examples provided in Figures 2.10, 2.11, 2.12 and 2.13 highlight the effect of error propagation. In particular, focusing on the mouth and on

Figure 2.10 *Example of artifacts caused by packet loss; Frame 1*

Figure 2.11 *Example of artifacts caused by packet loss; Frame 2*

Figure 2.12 *Example of artifacts caused by packet loss; Frame 3*

Figure 2.13 *Example of artifacts caused by packet loss; Frame 4*

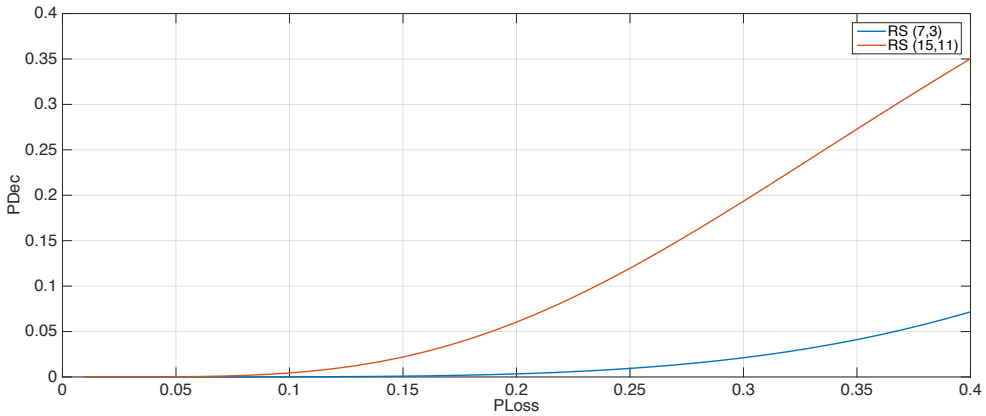


Figure 2.14 *Probability of packets not recovered over packet loss*

the chin, we may notice that in the first two frames, the error is almost nonexistent. In the third frame the blockiness begins to be visible and eventually becomes clearly evident in the fourth frame.

2.5.2 Forward Error Correction at the Application Layer

If in a communication channel a packet suffers from an erasure, it is still possible to recover the information. The only feasible solution is through the use of erasure codes at the application layer, which are a particular kind of Forward Error-Correction Codes. Some of the existing erasure codes are: Fountain codes [10], Raptor codes [11] and Reed-Solomon (RS) codes [12]. Let us consider RS codes as an example, these are linear and cyclic block codes for error detection and protection:

- linear: whenever two codes are added another codeword is produced.
- cyclic: the cycling shifting of a codeword produces a new one.

A Reed-Solomon code can be defined as an (n, k) code, where n is the block length (in symbols) and k the number of information symbols.

Systematic RS (n, k) codes in the finite Galois Field (2^8) can be used to protect inter-packets from loss in order to gain a higher visual quality. In this method, $r = n - k$ parity bytes from k source bytes with same index are created then, if at least k packets are received from n, the RS decoder is capable of recovering all the lost packets. The probability that a packet is lost and not restored by the decoder is [9]:

$$p_{dec}(p, n, k) = \sum_{i=n-k+1}^{n} \frac{i}{n} \cdot \binom{n}{k} \cdot p^i \cdot (1-p)^{n-1} \tag{2.2}$$

Figure 2.14 shows this function in relation to RS(7,3) and RS(15,11). In the latter it is possible to notice that, even if r is the same for the two RS codes, a smaller value of n ensures higher chances of recovering the packets.

2.6 Available Bandwidth Estimation

2.6.1 Bandwidth Definition

The network can be seen as a chain between the sender and the receiver, and each part of it has its own possible *maximum rate* at which it can send data. The term *bandwidth* is used to indicate the maximum rate at which the data can be forwarded. This is why, as a chain, the slowest element in the complete end-to-end system sets the *bottleneck bandwidth* i.e. the maximum rate at which data can be sent along the whole path.

Let us consider a network path P, characterized as a sequence of first-come-first-serve store and forward links which are fixed for the whole duration of the measurements. It connects and allows the sender S to forward packets to the receiver R, through H links with a constant rate of C_i bit per second, known as capacity rate or *Transmission Rate*. The bottleneck bandwidth is defined by the link with minimum transmission rate over the path:

$$C = \min_{i=0,\ldots,H} C_i. \tag{2.3}$$

Thus, the available bandwidth is:

$$A = \min_{i=0,\ldots,H} [C_i(1 - u_i)], \tag{2.4}$$

where u_i is the percentage of use of the link over a certain time interval. There are several techniques for the estimation of the available bandwidth, the most notable are packet-train dispersion, the one-way-delay-based model and the goodput-based model.

2.6.2 Packet-Train Dispersion

The Packet-Train dispersion technique is based on the concept of dispersion which is defined in [13] as the "time spacing from the last bit of the first packet to the last bit of the second packet" and proves to be easily calculated from any receiver. If a pair of packets, k and $k+1$, does not experience any cross traffic in the path, the dispersion δ^k will always be equal to the transmission delay τ_n at the narrow link:

$$\delta^k = \max_{i=0,\ldots,H} \tau_i = \frac{L}{\min_{i=0,\ldots,H}\{C_i\}} = \frac{L}{C} = \tau_n \tag{2.5}$$

where L is the size of the packets and C is the bottleneck bandwidth. On the contrary, in the event of cross traffic, the packets will suffer from additional delays.

Let us now consider the case where the sender transmits N back-to-back packets of the same size to the receiver, with $N > 2$, known as packet train, or simply train. The receiver, in this technique, measures the total dispersion of the train:

$$\Delta(N) = \sum_{k=1}^{N-1} \delta^k \tag{2.6}$$

The bandwidth estimation is performed as:

$$b(N) = \frac{(N-1)L}{\Delta(N)} = \frac{L}{\bar{\delta}(N)}, \tag{2.7}$$

where $\bar{\delta}(N)$ is the average dispersion among successive packet pairs inside the train:

$$\bar{\delta}(N) = \frac{\Delta N}{N-1} = \frac{\sum_{k=1}^{N-1} \delta^k}{N-1}. \tag{2.8}$$

This technique exploits the concept of packet pairs in order to calculate the average dispersion among these, which basically represents the average transmission delay experienced by the pairs in the path. It can be implemented in numerous systems, including video transmission, as it is only required with the use of packets having the same length and the knowledge of the packet arrival time. However, even if the dispersion $\Delta(N)$ increases with the increasing of N, the same is true for the cross traffic noise introduced by the dispersion itself, so "as the train length N increases, more cross traffic packets can interfere with the packet train, resulting in bandwidth measurements that are less than the path capacity C" [13].

2.6.3 Goodput-based Bandwidth Estimation

In [14] a depth analysis of the Skype congestion control algorithm was performed. From this study it appears that a Skype call is composed of two streams, one characterized by the video sent from the transmitter to the receiver and another, from the receiver to the transmitter, in which the information related to the round trip time and packet loss ratio are sent. This paper suggests that Skype is able to analyze the available bandwidth b by calculating the goodput as:

$$b = \frac{(\Delta s - \Delta l)}{\Delta T} \tag{2.9}$$

where Δs represents the number of bits sent in the period ΔT and Δl the amount of bits lost in ΔT.

2.7 Conclusions

In this chapter the theory related to this thesis work has been introduced. From the analysis of the topics previously studied some key points can be emphasized:

- The standard IEEE 802.11p has been defined. The basic concepts about the specifications related to the Physical layer have been described, following this is the description of how the transmitter creates the output flow.
- A brief comparison between the various multiple access protocols has been provided before moving to the description of the specification for the MAC layer from the standard 802.11p. The medium access scheme has been defined, the operations performed at this level for the purpose of transmission have also been described.
- In order to analyze the modalities with which the spatial and temporal redundancies are eliminated in a video, the various steps of the JPEG compression have been described and the concepts of Motion Estimation and Motion Compensation were introduced.
- It also described what happens when an error is propagated during a video transmission. As well, a RS decoder capable of performing error correction in order to protect inter-packets has been introduced.
- In the last section the parameters characterizing the channel estimation have been defined. The main definitions about bandwidth estimation have been provided and the main techniques employed have been presented.

3. RESEARCH METHODOLOGY AND MATERIALS

3.1 Introduction

In this work specific applications have been employed; Figure 3.1 and Figure 3.2 show in which way those are related to each other according to their use in the different simulated environments. Figure 3.1 is indeed related to the environment built for the analysis of channel resource allocation for different values of PLR, while Figure 3.2 is based on the 802.11p recreated environment.

The chosen software for streaming video over the network is Skype. This is because even if several softwares for instant messaging are available, it is the only one capable of ensuring transmission and protection, and also shows information about what is happening in the network. In order to transmit a video between users it was necessary to use one of the freeware softwares compatible with Skype, the one chosen one in this instance is ManyCam. Once the communication has started, the need to capture data commences, this task was carried out through a script launched with the NirCmd tool. In the first environment built it was necessary to change some network parameters, for this purpose NEWT was used. In order to build an 802.11p-based environment, routers have been employed, these have been modified to operate at frequencies provided by the standard. This was possible by modifying specific parameters of the installed operative system: OpenWrt. The PSNR has been analyzed through the use of conversion tables generated by exploiting the ability of the tool AviSynth.

For the purpose of better understanding the concept of available bandwidth estimation, at a different stage some experiments have been performed. This task was carried out by using the Iperf tool.

In the first section, the above mentioned applications will be introduced, it will

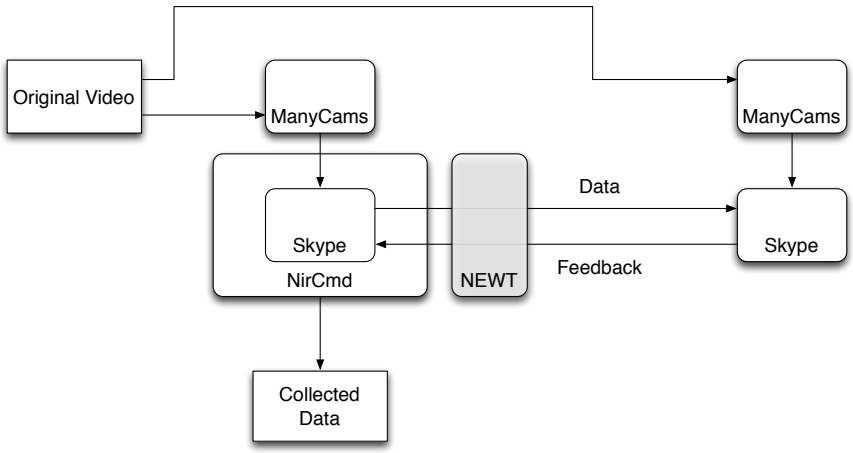


Figure 3.1 *Software employed for the analysis of the behavior of Skype depending on packet loss rate*

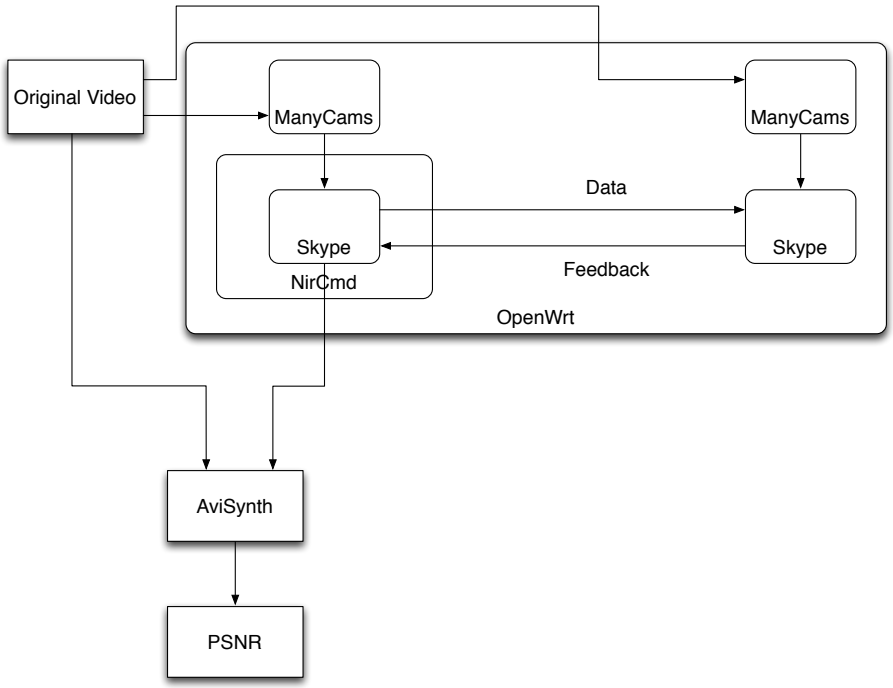


Figure 3.2 *Software employed for the analysis of the behavior of Skype in a VANET environment*

also explain how they were used in the created systems in more detail. The second section of this chapter will cover a description of the OpenWrt distribution, also the process by which the second network environment was built, based completely on the 802.11p standard, will be described. In the last section the gathered conclusion will be exposed.

3.2 Software Employed in the Work

3.2.1 Skype

Skype is a software which specializes in providing video and voice calls from computers, tablets and other mobile devices via the Internet to other devices or telephones/smartphones. It can offer many other services such as instant messaging through which it also holds the possibility to perform file sharing. It was created in August 2003, it was later bought by eBay which later has sold a majority stake to FREE inc.; it has been a part of the Microsoft Corporation since May 2011. The software was originally named "Skyper", abbreviation for "Skyper peer-to-peer", as originally featured a hybrid peer-to-peer and client-server system, however nowadays is powered by Microsoft-operated super nodes. In [15], Skype is defined as an "overlay peer-to-peer network". It consist of two kinds of nodes: ordinary hosts and super nodes. A user running Skype's application represents an ordinary host; a super node is an "ordinary host's end-point on the Skype network"; i.e. any device with a public IP address, high computational capability and bandwidth can be a super node. Super nodes are employed for the login process, they act as intermediates between the ordinary host and the Skype login server which is responsible for username and passwords. In the Skype network each client should be able to set up a table of reachable nodes, such table is know as host cache (HC) and contains the IP address and port number of each node. A Skype client, once logged, opens a TCP and a UDP port for listening. Usually a TCP channel is required for background communication of information about the network, while the UDP channel is used for data transmission between nodes.

It has been exploited in this thesis work not only because is the currently most widely used software for video calls, but because even if it is a closed-source software, it does provide certain information about its functioning capabilities. In particular, unlike other competitor software, this type of information is related to what is happening in the present network; the update time is about one second (non-optimal network conditions might not be able to detect such information, i.e. the update may take longer). It is possible to access such information by opening the window: *Call Technical Info* (Figure 3.3). In order to ensure a connection independent from any kind of external factor, after the call, in every simulation the internet connection has been removed. It has been possible to notice that the call was not interrupted by this process, furthermore, sniffing the connection, only a few TCP packets have

Figure 3.3 *Call Technical Info window*

been shown. By enabling only the UDP channel from the third set of simulations, an environment completely based on 802.11 has been created.

3.2.2 ManyCam

ManyCam is presented as a "live studio and webcam effects software" [16]. It is a webcam utility which can provide various effects, between these there is the possibility to insert static or dynamic backgrounds. It is compatible with the majority of instant messaging programs, making it also possible to use the webcam on multiple applications simultaneously.

A camera emulator has been employed in this work where the image source is a video file and it is looped. The video which has been used in the first set of experiments

is `Akiyo` from Joint Video Team; the reason why this video has been chosen is the minimum motion that characterized it, thus similar to any video call. Additional video used in the simulations were `Foreman` and a noise generated video. The first has been chosen as it represents a more dynamic solution compared with the previously cited `Akiyo`. However, we must state that for this purpose any application capable of operating as a camera emulator could be used.

3.2.3 NirCmd

NirCmd is a command line utility for task automation. It is a professional tool developed for all Windows platforms, designed for expert users who are familiar with the Linux command line. The idea behind this is to use it within batch file or as a link, to be run through a keyboard combination or scheduled tasks.

NirCmd has been employed in this work to save a screenshot in the folder *catture* every second by running the script:

```
nircmd.exe loop 60 1000 savescreenshot C:\Users\Sergio\Desktop
\catture\scr~$currdate.MM_dd_yyyy$-~$currtime.HH_mm_ss$.png
```

3.2.4 Iperf

In this work, Iperf has been used to calculate the bandwidth of a channel. It is a [17] "tool for active measurements of the maximum achievable bandwidth on IP networks". It is mostly related to TCP protocol, but it has been extended to UDP protocol. The network link is delimited by two (or more) hosts running Iperf. It tests the link by providing information about latency, jitter and datagram loss. The first version of Iperf was developed by NLANR/DAST, as it regards the latest version instead, Iperf3, is principally developed by ESnet / Lawrence Berkeley National Laboratory. Some of its features are:

- TCP and STCP:
  - Measures bandwidth;
  - Report MSS/MTU size;

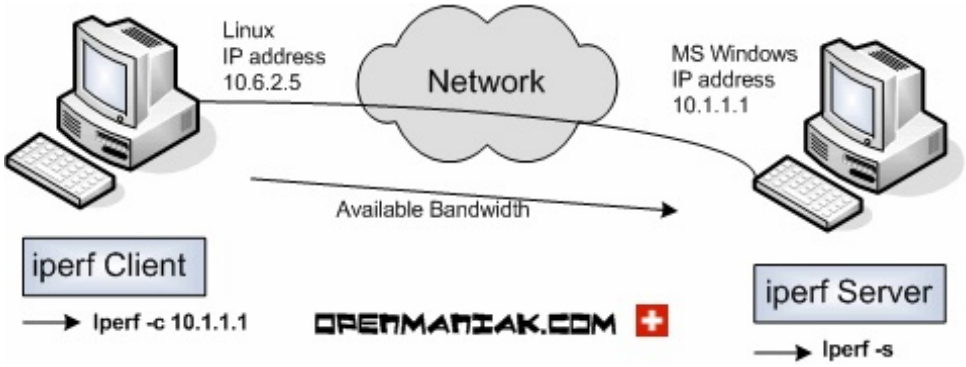


Figure 3.4 *Example of an Iperf Configuration from [18]*

  - Support for TCP window size;

- UDP:
  - Client can create UDP streams of specified bandwidth;
  - Measure Packet Loss;
  - Measure Delay Jitter;
  - Multicast Capable;
- Cross-Platform;
- Multi-Thread;
- Run on specified time;
- Print periodic at specific intervals.

The basic link consists of an *Iperf Client* and an *Iperf Server*, not necessarily running the same operating system. In the example from [18], the client works on a system running Linux, while the server is on a Windows machine, as shown in Figure 3.4. The first thing to do is to set the server side and then the client side. The basic code to activate the server side is:

```
iperf -s
```

the client side:

```
iperf -c 192.168.1.100
```

where 192.168.1.100 is the Ip address of the server. By default, the client connects the server on the TCP port 5001, and the bandwidth displayed is the one from the client to the server, it runs the experiment for 10 seconds. The received information can be changed by adding some command option lines, the most important are:

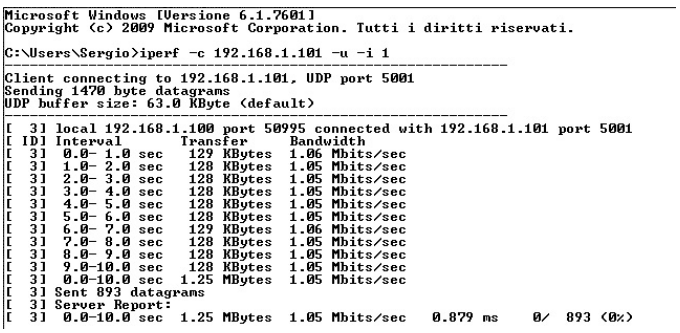
Command Line Option	Description
no arg.	Default setting
-b	Data format
-r	Bi-directional bandwidth
-d	Simultaneous bi-directional bandwidth
-w	TCP Window size
-p, -t, -i	Port, timing and interval
-u, -b	UDP test, bandwidth setting
-m	Maximum Segment Size display
-M	Maximum Segment Size settings
-P	Parallel tests
-h	help

Personal experience

In order both to calculate the available bandwidth and to understand the functioning of the tool itself, some experiments have been carried out.

1. In the first it was necessary to understand in which way Iperf was working, later the goal became to record evidence that it was actually measuring the channel capacity. In order to understand the basics of Iperf, simple experiments in UDP environment were performed. Once the server was activated, all the code was then written on the client side. Of course, the first experiment was the easiest one, already described in the previous section. The result showed that the analyzed channel has a capacity of 1.05 Mbps, this was achieved by sending 893 datagrams in 10 seconds.

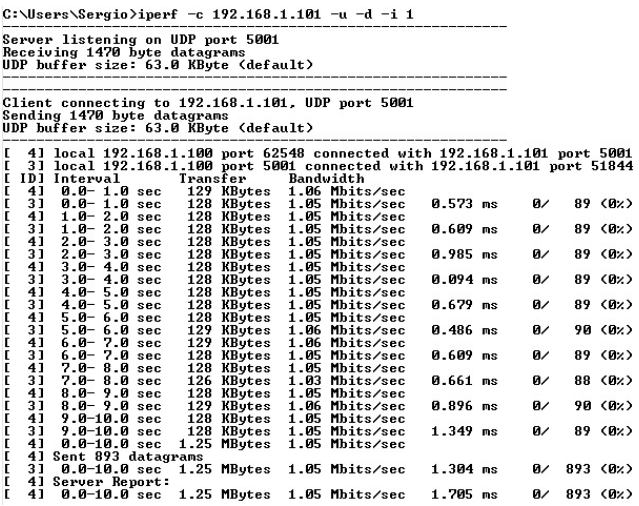
2. Later to make the things more interesting, the option *-i* was added to receive information each second. This showed each second the client was sending 128 kBytes, reporting a bandwidth estimation from each of those and, at the end, even a summary of what has been obtained by the overall simulation. This



```
Microsoft Windows [Versione 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Tutti i diritti riservati.

C:\Users\Sergio>iperf -c 192.168.1.101 -u -i 1
------------------------------------------------------------
Client connecting to 192.168.1.101, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 63.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.100 port 50995 connected with 192.168.1.101 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec   129 KBytes  1.06 Mbits/sec
[  3]  1.0- 2.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  2.0- 3.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  3.0- 4.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  4.0- 5.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  5.0- 6.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  6.0- 7.0 sec   129 KBytes  1.06 Mbits/sec
[  3]  7.0- 8.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  8.0- 9.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  9.0-10.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Sent 893 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.879 ms    0/  893 (0%)
```

Figure 3.5 *Results of Experiment 2 - Client side*



```
C:\Users\Sergio>iperf -c 192.168.1.101 -u -d -i 1
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 63.0 KByte (default)
------------------------------------------------------------
------------------------------------------------------------
Client connecting to 192.168.1.101, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 63.0 KByte (default)
------------------------------------------------------------
[  4] local 192.168.1.100 port 62548 connected with 192.168.1.101 port 5001
[  3] local 192.168.1.100 port 5001 connected with 192.168.1.101 port 51844
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0- 1.0 sec   129 KBytes  1.06 Mbits/sec
[  3]  0.0- 1.0 sec   128 KBytes  1.05 Mbits/sec   0.573 ms    0/   89 (0%)
[  4]  1.0- 2.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  1.0- 2.0 sec   128 KBytes  1.05 Mbits/sec   0.609 ms    0/   89 (0%)
[  4]  2.0- 3.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  2.0- 3.0 sec   128 KBytes  1.05 Mbits/sec   0.985 ms    0/   89 (0%)
[  4]  3.0- 4.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  3.0- 4.0 sec   128 KBytes  1.05 Mbits/sec   0.094 ms    0/   89 (0%)
[  4]  4.0- 5.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  4.0- 5.0 sec   128 KBytes  1.05 Mbits/sec   0.679 ms    0/   89 (0%)
[  4]  5.0- 6.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  5.0- 6.0 sec   129 KBytes  1.06 Mbits/sec   0.486 ms    0/   90 (0%)
[  4]  6.0- 7.0 sec   129 KBytes  1.06 Mbits/sec
[  3]  6.0- 7.0 sec   128 KBytes  1.05 Mbits/sec   0.609 ms    0/   89 (0%)
[  4]  7.0- 8.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  7.0- 8.0 sec   126 KBytes  1.03 Mbits/sec   0.661 ms    0/   88 (0%)
[  4]  8.0- 9.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  8.0- 9.0 sec   129 KBytes  1.06 Mbits/sec   0.896 ms    0/   90 (0%)
[  4]  9.0-10.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  9.0-10.0 sec   128 KBytes  1.05 Mbits/sec   1.349 ms    0/   89 (0%)
[  4]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  4] Sent 893 datagrams
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   1.304 ms    0/  893 (0%)
[  4] Server Report:
[  4]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   1.705 ms    0/  893 (0%)
```

Figure 3.6 *Results of Experiment 3 - Client side*

summary is very interesting because in addition to the information related to the bytes sent in the time interval required and the capacity of the channel, it also reported information about the packet sent, packet lost (and percentage) and jitter.

3. The third experiment was about the simultaneous bi-directional bandwidth, so the option *-d* was added. Every second two results were shown, the first one was the estimation from the client to the server, and the second one was about the opposite way. Of course the channel had the same capacity in both directions, but as the client this time was the receiver, it was possible for it to calculate the information that are usually reported as ACK: packet sent, packet loss and jitter.

4. In the fourth experiment the environment has been slightly altered by adding an additional client (running on a Linux machine). Both computers tried to

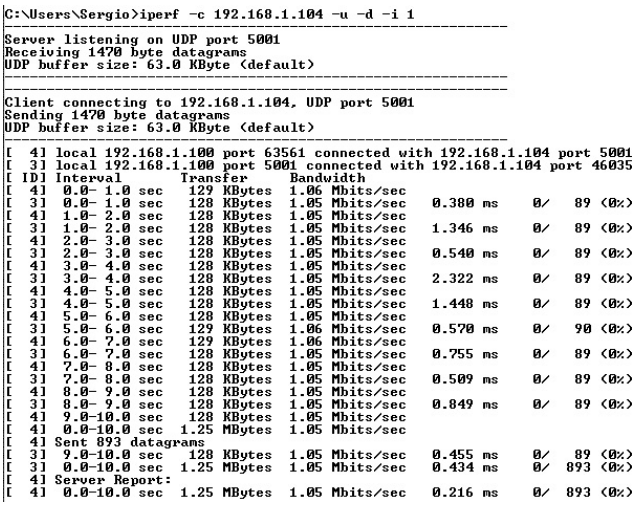
```
C:\Users\Sergio>iperf -c 192.168.1.104 -u -d -i 1
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 63.0 KByte (default)
------------------------------------------------------------
------------------------------------------------------------
Client connecting to 192.168.1.104, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 63.0 KByte (default)
------------------------------------------------------------
[  4] local 192.168.1.100 port 49603 connected with 192.168.1.104 port 5001
[  3] local 192.168.1.100 port 5001 connected with 192.168.1.104 port 39621
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0- 1.0 sec   129 KBytes  1.06 Mbits/sec
[  3]  0.0- 1.0 sec   128 KBytes  1.05 Mbits/sec   0.482 ms    0/   89 (0%)
[  4]  1.0- 2.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  1.0- 2.0 sec   128 KBytes  1.05 Mbits/sec   0.494 ms    0/   89 (0%)
[  4]  2.0- 3.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  2.0- 3.0 sec   128 KBytes  1.05 Mbits/sec   0.301 ms    0/   89 (0%)
[  4]  3.0- 4.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  3.0- 4.0 sec   128 KBytes  1.05 Mbits/sec   0.290 ms    0/   89 (0%)
[  4]  4.0- 5.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  4.0- 5.0 sec   129 KBytes  1.06 Mbits/sec   0.741 ms    0/   90 (0%)
[  4]  5.0- 6.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  5.0- 6.0 sec   128 KBytes  1.05 Mbits/sec   0.552 ms    0/   89 (0%)
[  4]  6.0- 7.0 sec   129 KBytes  1.06 Mbits/sec
[  3]  6.0- 7.0 sec   128 KBytes  1.05 Mbits/sec   0.366 ms    0/   89 (0%)
[  4]  7.0- 8.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  7.0- 8.0 sec   128 KBytes  1.05 Mbits/sec   0.686 ms    0/   89 (0%)
[  4]  8.0- 9.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  8.0- 9.0 sec   128 KBytes  1.05 Mbits/sec   1.064 ms    0/   89 (0%)
[  4]  9.0-10.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  9.0-10.0 sec   128 KBytes  1.05 Mbits/sec   0.333 ms    0/   89 (0%)
[  4]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  4] Sent 893 datagrams
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.366 ms    0/  893 (0%)
[  4] Server Report:
[  4]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.202 ms    0/  893 (0%)
```

Figure 3.7 *Results of Experiment 4 - Client 1 side*

```
C:\Users\ide>iperf -c 192.168.1.104 -u -d -i 1 -t 20
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 64.0 KByte (default)
------------------------------------------------------------
------------------------------------------------------------
Client connecting to 192.168.1.104, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 64.0 KByte (default)
------------------------------------------------------------
[  4] local 192.168.1.101 port 64754 connected with 192.168.1.104 port 5001
[  3] local 192.168.1.101 port 5001 connected with 192.168.1.104 port 35816
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0- 1.0 sec   129 KBytes  1.06 Mbits/sec
[  3]  0.0- 1.0 sec   128 KBytes  1.05 Mbits/sec   0.906 ms    0/   89 (0%)
[  4]  1.0- 2.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  1.0- 2.0 sec   129 KBytes  1.06 Mbits/sec   0.861 ms    0/   90 (0%)
[  4]  2.0- 3.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  2.0- 3.0 sec   128 KBytes  1.05 Mbits/sec   0.682 ms    0/   89 (0%)
[  4]  3.0- 4.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  3.0- 4.0 sec   128 KBytes  1.05 Mbits/sec   0.721 ms    0/   89 (0%)
[  4]  4.0- 5.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  4.0- 5.0 sec   128 KBytes  1.05 Mbits/sec   0.435 ms    0/   89 (0%)
[  4]  5.0- 6.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  5.0- 6.0 sec   128 KBytes  1.05 Mbits/sec   0.604 ms    0/   89 (0%)
[  4]  6.0- 7.0 sec   129 KBytes  1.06 Mbits/sec
[  3]  6.0- 7.0 sec   128 KBytes  1.05 Mbits/sec   1.909 ms    0/   89 (0%)
[  4]  7.0- 8.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  7.0- 8.0 sec   129 KBytes  1.06 Mbits/sec   0.408 ms    0/   90 (0%)
[  4]  8.0- 9.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  8.0- 9.0 sec   128 KBytes  1.05 Mbits/sec   0.469 ms    0/   89 (0%)
[  4]  9.0-10.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  9.0-10.0 sec   128 KBytes  1.05 Mbits/sec   0.639 ms    0/   89 (0%)
[  4] 10.0-11.0 sec   128 KBytes  1.05 Mbits/sec
[  3] 10.0-11.0 sec   128 KBytes  1.05 Mbits/sec   1.186 ms    0/   89 (0%)
[  4] 11.0-12.0 sec   128 KBytes  1.05 Mbits/sec
[  3] 11.0-12.0 sec   128 KBytes  1.05 Mbits/sec   0.607 ms    0/   89 (0%)
[  4] 12.0-13.0 sec   129 KBytes  1.06 Mbits/sec
[  3] 12.0-13.0 sec   128 KBytes  1.05 Mbits/sec   0.981 ms    0/   89 (0%)
[  4] 13.0-14.0 sec   128 KBytes  1.05 Mbits/sec
[  3] 13.0-14.0 sec   129 KBytes  1.06 Mbits/sec   0.751 ms    0/   90 (0%)
[  4] 14.0-15.0 sec   128 KBytes  1.05 Mbits/sec
[  3] 14.0-15.0 sec   128 KBytes  1.05 Mbits/sec   0.934 ms    0/   89 (0%)
[  4] 15.0-16.0 sec   128 KBytes  1.05 Mbits/sec
[  3] 15.0-16.0 sec   123 KBytes  1.01 Mbits/sec   0.913 ms    3/   89 (3.4%)
[  4] 16.0-17.0 sec   128 KBytes  1.05 Mbits/sec
[  3] 16.0-17.0 sec   128 KBytes  1.05 Mbits/sec   0.428 ms    0/   89 (0%)
[  4] 17.0-18.0 sec   128 KBytes  1.05 Mbits/sec
[  3] 17.0-18.0 sec   128 KBytes  1.05 Mbits/sec   0.645 ms    0/   89 (0%)
[  4] 18.0-19.0 sec   129 KBytes  1.06 Mbits/sec
[  3] 18.0-19.0 sec   128 KBytes  1.05 Mbits/sec   0.628 ms    0/   89 (0%)
[  4] 19.0-20.0 sec   128 KBytes  1.05 Mbits/sec
[  4]  0.0-20.0 sec  2.50 MBytes  1.05 Mbits/sec
[  4] Sent 1785 datagrams
[  3] 19.0-20.0 sec   129 KBytes  1.06 Mbits/sec   0.464 ms    0/   90 (0%)
[  3]  0.0-20.0 sec  2.50 MBytes  1.05 Mbits/sec   0.448 ms    3/ 1785 (0.17%)
[  4] Server Report:
[  4]  0.0-20.0 sec  2.50 MBytes  1.05 Mbits/sec   0.520 ms    0/ 1785 (0%)
```

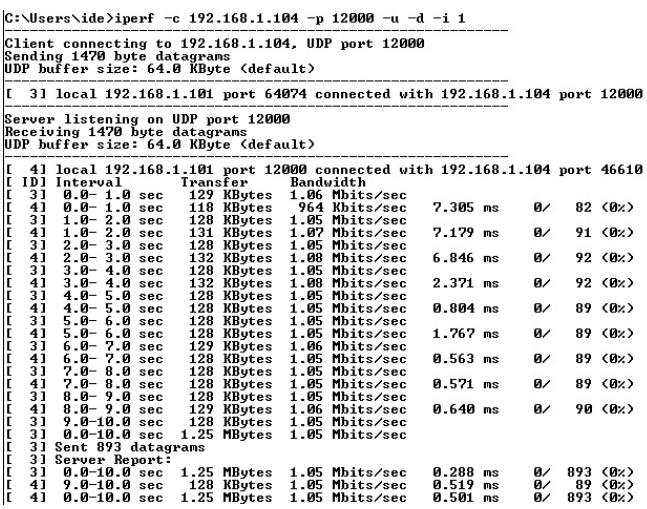
Figure 3.8 *Results of Experiment 4 - Client 2 side*

send a stream of data at the same server, the result was that both of them were able to send the stream and receive ACK from the server. What is very interesting is the delay at second 6.0-7.0 in Figure 3.8, moment in which the first client began sending the stream. The results, as expected, were the same for both clients measuring the same capacity for the same experiments.



```
C:\Users\Sergio>iperf -c 192.168.1.104 -u -d -i 1
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 63.0 KByte (default)
------------------------------------------------------------
------------------------------------------------------------
Client connecting to 192.168.1.104, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 63.0 KByte (default)
------------------------------------------------------------
[  4] local 192.168.1.100 port 63561 connected with 192.168.1.104 port 5001
[  3] local 192.168.1.100 port 5001 connected with 192.168.1.104 port 46035
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0- 1.0 sec   129 KBytes  1.06 Mbits/sec
[  3]  0.0- 1.0 sec   128 KBytes  1.05 Mbits/sec   0.380 ms    0/   89 (0%)
[  4]  1.0- 2.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  1.0- 2.0 sec   128 KBytes  1.05 Mbits/sec   1.346 ms    0/   89 (0%)
[  4]  2.0- 3.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  2.0- 3.0 sec   128 KBytes  1.05 Mbits/sec   0.540 ms    0/   89 (0%)
[  4]  3.0- 4.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  3.0- 4.0 sec   128 KBytes  1.05 Mbits/sec   2.322 ms    0/   89 (0%)
[  4]  4.0- 5.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  4.0- 5.0 sec   128 KBytes  1.05 Mbits/sec   1.448 ms    0/   89 (0%)
[  4]  5.0- 6.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  5.0- 6.0 sec   129 KBytes  1.06 Mbits/sec   0.570 ms    0/   90 (0%)
[  4]  6.0- 7.0 sec   129 KBytes  1.06 Mbits/sec
[  3]  6.0- 7.0 sec   128 KBytes  1.05 Mbits/sec   0.755 ms    0/   89 (0%)
[  4]  7.0- 8.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  7.0- 8.0 sec   128 KBytes  1.05 Mbits/sec   0.509 ms    0/   89 (0%)
[  4]  8.0- 9.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  8.0- 9.0 sec   128 KBytes  1.05 Mbits/sec   0.849 ms    0/   89 (0%)
[  4]  9.0-10.0 sec   128 KBytes  1.05 Mbits/sec
[  4]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  4] Sent 893 datagrams
[  3]  9.0-10.0 sec   128 KBytes  1.05 Mbits/sec   0.455 ms    0/   89 (0%)
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.434 ms    0/  893 (0%)
[  4] Server Report:
[  4]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.216 ms    0/  893 (0%)
```

Figure 3.9 *Results of Experiment 5 - Client 1 side*



```
C:\Users\ide>iperf -c 192.168.1.104 -p 12000 -u -d -i 1
------------------------------------------------------------
Client connecting to 192.168.1.104, UDP port 12000
Sending 1470 byte datagrams
UDP buffer size: 64.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.101 port 64074 connected with 192.168.1.104 port 12000
------------------------------------------------------------
Server listening on UDP port 12000
Receiving 1470 byte datagrams
UDP buffer size: 64.0 KByte (default)
------------------------------------------------------------
[  4] local 192.168.1.101 port 12000 connected with 192.168.1.104 port 46610
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec   129 KBytes  1.06 Mbits/sec
[  4]  0.0- 1.0 sec   118 KBytes   964 Kbits/sec   7.305 ms    0/   82 (0%)
[  3]  1.0- 2.0 sec   128 KBytes  1.05 Mbits/sec
[  4]  1.0- 2.0 sec   131 KBytes  1.07 Mbits/sec   7.179 ms    0/   91 (0%)
[  3]  2.0- 3.0 sec   128 KBytes  1.05 Mbits/sec
[  4]  2.0- 3.0 sec   132 KBytes  1.08 Mbits/sec   6.846 ms    0/   92 (0%)
[  3]  3.0- 4.0 sec   128 KBytes  1.05 Mbits/sec
[  4]  3.0- 4.0 sec   132 KBytes  1.08 Mbits/sec   2.371 ms    0/   92 (0%)
[  3]  4.0- 5.0 sec   128 KBytes  1.05 Mbits/sec
[  4]  4.0- 5.0 sec   128 KBytes  1.05 Mbits/sec   0.804 ms    0/   89 (0%)
[  3]  5.0- 6.0 sec   128 KBytes  1.05 Mbits/sec
[  4]  5.0- 6.0 sec   128 KBytes  1.05 Mbits/sec   1.767 ms    0/   89 (0%)
[  3]  6.0- 7.0 sec   129 KBytes  1.06 Mbits/sec
[  4]  6.0- 7.0 sec   128 KBytes  1.05 Mbits/sec   0.563 ms    0/   89 (0%)
[  3]  7.0- 8.0 sec   128 KBytes  1.05 Mbits/sec
[  4]  7.0- 8.0 sec   128 KBytes  1.05 Mbits/sec   0.571 ms    0/   89 (0%)
[  3]  8.0- 9.0 sec   128 KBytes  1.05 Mbits/sec
[  4]  8.0- 9.0 sec   129 KBytes  1.06 Mbits/sec   0.640 ms    0/   90 (0%)
[  3]  9.0-10.0 sec   128 KBytes  1.05 Mbits/sec
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Sent 893 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.288 ms    0/  893 (0%)
[  4]  9.0-10.0 sec   128 KBytes  1.05 Mbits/sec   0.519 ms    0/   89 (0%)
[  4]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.501 ms    0/  893 (0%)
```

Figure 3.10 *Results of Experiment 5 - Client 2 side*

5. In the fifth experiment with the second client the option *-p* has been added, so the code was:

   ```
   iperf -c 192.168.1.100 -p 12000 -u -d -i 1
   ```

   and at the server another terminal instance was started. The same results as the previous experiments were achieved.

6. From the previous experiments it was clear that all streams were received (almost) without any loss. So it was interesting to send a larger number of multiple parallel streams. In order to do this the code at the client was:

```
[ 18] Server Report:
[ 18]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.656 ms    3/  893 (0.34%)
[ 18]  0.0-10.0 sec  5 datagrams received out-of-order
[ 15] Server Report:
[ 15]  0.0-10.0 sec  1.25 MBytes  1.04 Mbits/sec   1.206 ms    2/  892 (0.22%)
[ 15]  0.0-10.0 sec  3 datagrams received out-of-order
[ 21] Server Report:
[ 21]  0.0-10.0 sec  1.25 MBytes  1.04 Mbits/sec   0.662 ms    3/  893 (0.34%)
[ 14] Server Report:
[ 14]  0.0-10.0 sec  1.26 MBytes  1.06 Mbits/sec   1.374 ms    0/  893 (0%)
[ 14]  0.0-10.0 sec  6 datagrams received out-of-order
[  9] Server Report:
[  9]  0.0-10.0 sec  1.24 MBytes  1.04 Mbits/sec   0.917 ms    7/  892 (0.78%)
[  9]  0.0-10.0 sec  1 datagrams received out-of-order
[ 11] Server Report:
[ 11]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   1.321 ms    1/  892 (0.11%)
[ 11]  0.0-10.0 sec  1 datagrams received out-of-order
[ 17] Server Report:
[ 17]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.866 ms    0/  892 (0%)
[ 17]  0.0-10.0 sec  2 datagrams received out-of-order
[ 19] Server Report:
[ 19]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   1.859 ms    4/  892 (0.45%)
[ 19]  0.0-10.0 sec  5 datagrams received out-of-order
[  7] Server Report:
[  7]  0.0-10.0 sec  1.24 MBytes  1.04 Mbits/sec   1.189 ms    4/  892 (0.45%)
[  7]  0.0-10.0 sec  2 datagrams received out-of-order
[  3] Server Report:
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   1.546 ms    0/  892 (0%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
[ 13] Server Report:
[ 13]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   1.567 ms    2/  892 (0.22%)
[ 13]  0.0-10.0 sec  2 datagrams received out-of-order
[  5] Server Report:
[  5]  0.0-10.0 sec  1.24 MBytes  1.04 Mbits/sec   2.145 ms    4/  892 (0.45%)
[  5]  0.0-10.0 sec  1 datagrams received out-of-order
[  4] Server Report:
[  4]  0.0-10.0 sec  1.24 MBytes  1.04 Mbits/sec   0.321 ms    8/  893 (0.9%)
[ 12] Server Report:
[ 12]  0.0-10.0 sec  1.26 MBytes  1.05 Mbits/sec   2.964 ms    2/  893 (0.22%)
[ 12]  0.0-10.0 sec  6 datagrams received out-of-order
[ 22] Server Report:
[ 22]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.769 ms    1/  893 (0.11%)
[ 22]  0.0-10.0 sec  2 datagrams received out-of-order
[  6] Server Report:
[  6]  0.0-10.0 sec  1.26 MBytes  1.06 Mbits/sec   0.316 ms    0/  893 (0%)
[  6]  0.0-10.0 sec  5 datagrams received out-of-order
[  8] Server Report:
[  8]  0.0-10.0 sec  1.24 MBytes  1.04 Mbits/sec   0.586 ms    6/  893 (0.67%)
[ 10] Server Report:
[ 10]  0.0-10.0 sec  1.26 MBytes  1.05 Mbits/sec   1.314 ms    1/  893 (0.11%)
[ 10]  0.0-10.0 sec  6 datagrams received out-of-order
[ 16] Server Report:
[ 16]  0.0-10.0 sec  1.25 MBytes  1.04 Mbits/sec   2.094 ms    1/  893 (0.11%)
[ 20] Server Report:
[ 20]  0.0-10.0 sec  1.25 MBytes  1.04 Mbits/sec   2.686 ms    2/  893 (0.22%)
```

Figure 3.11 *Report of Experiment 6 - 20 connections*

```
[ 10] Server Report:
[ 10]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.686 ms    2/  892 (0.22%)
[ 10]  0.0-10.0 sec  2 datagrams received out-of-order
[  3] Server Report:
[  3]  0.0-10.0 sec  1.26 MBytes  1.05 Mbits/sec   0.528 ms    1/  892 (0.11%)
[  3]  0.0-10.0 sec  6 datagrams received out-of-order
[ 12] Server Report:
[ 12]  0.0-10.0 sec  1.25 MBytes  1.04 Mbits/sec   1.689 ms    3/  893 (0.34%)
[  4] Server Report:
[  4]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.640 ms    0/  892 (0%)
[  4]  0.0-10.0 sec  1 datagrams received out-of-order
[  5] Server Report:
[  5]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.574 ms    2/  892 (0.22%)
[  5]  0.0-10.0 sec  1 datagrams received out-of-order
[  9] Server Report:
[  9]  0.0-10.0 sec  1.24 MBytes  1.04 Mbits/sec   0.819 ms    4/  892 (0.45%)
[  9]  0.0-10.0 sec  1 datagrams received out-of-order
[ 11] Server Report:
[ 11]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.678 ms    1/  892 (0.11%)
[ 11]  0.0-10.0 sec  1 datagrams received out-of-order
[  7] Server Report:
[  7]  0.0-10.0 sec  1.25 MBytes  1.04 Mbits/sec   1.197 ms    3/  892 (0.34%)
[  7]  0.0-10.0 sec  1 datagrams received out-of-order
[  6] Server Report:
[  6]  0.0-10.0 sec  1.26 MBytes  1.05 Mbits/sec   0.929 ms    0/  892 (0%)
[  6]  0.0-10.0 sec  7 datagrams received out-of-order
[  8] Server Report:
[  8]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   1.330 ms    1/  892 (0.11%)
[  8]  0.0-10.0 sec  1 datagrams received out-of-order
```

Figure 3.12 *Report of Experiment 6 - 10 connections*

```
iperf -c 192.168.1.100 -p 12000 -u -i  1 -P 20
```

This resulted in twenty parallel connections. The results showed that for each connection it was possible to receive an ACK, but no one of those resulted in a "perfect stream", i.e. no one had no loss or no out-of-order datagram. The experiment lately was repeated with ten connections, showing the same results of the previous experiment.

From the simulation it was clear that this tool is perfect to inject traffic in a well-

known channel and even to estimate the capacity of this. Moreover, if information about the packet sent and loss are necessary, the tool can provide it through the ACK received from the server. From my personal experience I can say that the algorithm which was used requires that the packets are sent in order and with a constant flow; but in order to have a comprehensive knowledge of the channel, an ACK from the receiver is required. This is why Iperf can be described as an active probing measurement tool, using a technique based on the packet-pair algorithm.

3.2.5 NEWT

Network Emulator for Windows Toolkit is a software-based emulator. It is capable of emulating the behavior of networks using a reliable physical link and set values for certain parameters including: round-trip time across the network (latency), the amount of available bandwidth, packet loss and error propagation. It also provides flexibility in filtering network packets for protocols like TCP and UDP.

NEWT has been employed in this work in the first and second set of simulations. In particular, after setting the network, it has been used for filtering the UDP channel used by Skype. The parameters that have been changed the most were: packet loss rate and the available bandwidth. In particular, it must be emphasized that the first does not provide a constant value, but on the contrary produces an average value in a short interval.

3.2.6 AviSynth

Avisynth, as suggested by the logo, is: "A powerful non linear scripting language for videos" [19]. Among its main advantages there is the one to operate as a frameserver, i.e. any changes made when writing the code provides a direct result without the need for generating any temporary files. It is a defined scripting language as it is not based on any graphical user interface, but instead makes the most of a script system which allows advanced non-linear coding. This feature is the main one provided by Avisynth as the Non-Linear (Video) Editing (NLE), unlike the normal linear one, it allows for advanced editing on clips like trimming, even if the duration of the video has been modified. More information on this feature can be found in [20]. Avisynth works by generating a new AVI file, it is thus possible to use it with any application capable of running AVI files; the reason of this resides in its nature of working as a

framework, i.e. the new video is generated by following the "instructions" provided by the script which is created by the user.

In this work AviSynth has been used for PSNR calculation. In particular, a script was generated in order to provide the value of the PSNR for each frame starting from certain input values. Most of those input values were the same during the different simulations, the only one which was constantly changed was the bit rate. This made it possible to achieve the value of the PSNR for each frame for different values of the bit rate; moreover, by means of these it was possible to obtain an average value of the PSNR for the whole video. It was therefore possible to generate a conversion table to calculate the PSNR starting from the bit rate.

3.3 OpenWrt

OpenWrt is a GNU/Linux distribution for embedded devices. It is designed to be a full-featured easy modifiable operating system for routers. Instead of a single static firmware, OpenWrt offers a writable filesystem where it is possible to install optional packages. It allows for custom configuration after installing the firmware and the chance to use it differently than imposed the seller. Moreover, for developers, it provides a framework to build an application without the requirement of creating a static and complete image, and distribution, around it [21]. Main features:

- Free and open-source: it is a completely open-source project licensed under GPL. It is intended to be always accessible through a website, with full source code which is easy build.
- Easy and free access: as it is an open-source anyone can contribute. The current developers always grant write access to anyone who is interested in having it. Their motto is:

  > "We believe people are responsible when given responsibility. Just ask and you will be able to acquire the access rights you need."

- Community driven:

  > "It is not about us offering you something, it is about everyone coming together to work and collaborate towards a common goal."

Those are the main reasons why it is stated as the best firmware solution for routers. Moreover, it is better than other systems in regards to performance, stability, extensibility and robustness.

The firmware chosen for the antennas in the laboratory was version 14.07 codenamed Barrier Breaker, then depending on the architecture of the antennas two slightly different versions have been installed: the generic version and the `mikrotik` one. This operating system was crucial in the second part of the analysis; it allowed the creation of a network based on the 802.11p standard.

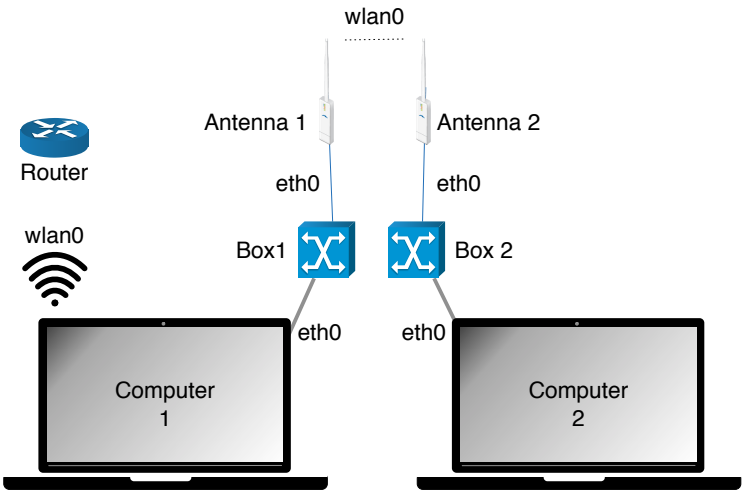


Figure 3.13 *VANET environment*

3.3.1 Setting the environment

The research laboratory in which the analysis has been carried out was composed by twelve antennas, twelve compatible boxes and eight computers. Two of those computers have been used as support for the boxes, in particular, one defined as `IDE5`, has been used to load the necessary firmware for the boxes, while the second, known as `IDE6`, to browse through the directories and install the required additional packages; the other six computers have been used to run Skype. First of all it was necessary to modify the firmware installed in the boxes in order to use the required protocol and frequencies. As previously mentioned, the firmware installed and loaded in the boxes was Barrier Breaker from OpenWrt distribution. To load it, after all the boxes have been turned on, a script named `setup-ide.sh` has been run on the command line of `IDE5`. After that, using `IDE6`, it was necessary to check if all the devices were synchronized in time. It was then possible to browse the directories in the boxes in order to modify the firmware installed.

The main challenge in this environment involved the pursuit of a possible way to obtain a connection between computers based exclusively on VANET. The reason for this lies in the very nature of Skype, the latter indeed needs, first of all, to be connected to internet, moreover, it does not allow the user to choose the connection path. When the application is lunched, it sends the IP address of the computer to its servers and receives back from those the IP addresses of the computers connected to Skype. When the call is requested, the IP address is the key to find the computer to call, i.e. a connection to the server is always required for both users as long

Table 3.1 *IP Address of the devices and interfaces*

Device	eth0 IP Address	wlan0 IP ADDRESS
Computer n.1	192.168.137.10	192.168.1.100
Box n.1	192.168.137.12	192.168.5.2
Box n.2	192.168.137.13	192.168.5.1
Computer n.2	192.168.137.101	NO

as the call is not started. The only possible way to force the path to be the one characterized by the boxes creating the VANET environment was to use one of the two computers as an access point and, through it, share the connection with the other computer using Windows ICS (Internet Connection Sharing). This service allows a *host* computer, which is connected to the Internet and has a separate connection to the other computers on the network, to share its connection with such devices. In this particular case, a computer was connected to a router via Wi-Fi (`wlan0`) and the connection shared via Ethernet (`eth0`). After ensuring the effective functioning of the setup, it was then necessary to forward the connection through the boxes. The desired environment is illustrated in Figure 3.13; the creation of which can be summarized in four steps:

1. The first step concerned the definition of the radio characteristics of the device. In order to do this the file *Wireless* in the directory `/etc/config/` needed to be completely rewritten. The Wireless configuration file is responsible for Wireless settings and WiFi network definition. Once the file was edited, it was necessary to reload the characteristics of the network to activate the configuration through the command: `\etc\init.d\network reload`.

2. The second step in the creation of the network concerned the definition of static IP addresses and routing tables. In order to enable and configure the two interfaces `lan` and `wlan0` it was necessary to modify the file *Network* located in the directory previously mentioned. The Network configuration file is responsible for defining interface configuration and network routes. Even in this case the command: `\etc\init.d\network reload` needs to be launched.

3. The third step consisted in writing the routing table for the two computers, known as `IDE4` and `IDE1`. Table 3.1 reports the IP Addresses and the interfaces of the devices.

4. The last step consisted in enabling the forwarding of the connection through

the boxes, it was possible by adding the *iptables* module to the installed firmware. For the purpose of adding a new module, the utility OPKG package manager can be used. After updating the link written in the file `opkg.conf`, placed in the `\etc` directory, it was possible to update the list of the installable modules and then install `iptables`. Once the installation is completed to forward the connection was necessary to run the commands:

```
iptables -A FORWARD --in-interface eth0 -j ACCEPT
```

```
iptables --table nat -A POSTROUTING --out-interface wlan0 -j
MASQUERADE
```

After this last step the second computer was able to connect to Internet.

3.4 Conclusions

In this chapter the materials used for the development of the project were introduced. After the materials were described the reason for their choice was also explained.

- Skype is the most important software including those used in this work, as it provides a multitude of services required. These include information about what happens in the network, as well as providing all the encoding, decoding protection and transmission of the data.
- Through the use of the tool Iperf it was possible to carry out some experiments on the channel in order to better understand how it works. From the analysis of the latter it was possible to state that it operates using a packet-pair algorithm which requires continuous ACK from the receiver.
- Network Emulator for Windows Toolkit was vital for the first and second set of simulations as it allowed the analysis of the environment based on PLR and available bandwidth parameters.
- By employing AviSynth an analysis of the PSNR based on bit rate values was possible.
- Additional softwares used were NirCmd and Manycam. The first was used to capture screenshots for the purpose of collecting data, while the second was chosen as the camera emulator.
- OpenWrt resulted essential in this work as it has provided a solid foundation for the creation of the network based on 802.11p. The process through which the vehicular network was created has also been described.

4. RESULTS AND ANALYSIS

4.1 Introduction

The study conducted by Zhang *et al.* in [22] was the starting point for this thesis. At the beginning the aim was to reproduce exactly the same environment of the mentioned paper and to conduct the same experiments. Subsequently a more specific analysis took place. In this study additional experiments were performed, for example, examining the behavior of Skype based on the amount of motion of the video.

The purpose of this work is to understand and analyze channel resource allocation in a 802.11p-based environment. On the other hand, the impossibility to perform measurements in a real environment, due to more than obvious problems in the implementation, led to having the study in a static setup in which special antennas were used. These devices have been modified to operate according to the frequencies of the standard 802.11p. In addition to the static nature of the environment, another key feature of this case has been the minimum distance between the antennas which allowed for the only loss to be due to interference (i.e. non-linear effect in the network are considered to be non-existent). Since the focus has been on the final results from the user side, this analysis has therefore been focused on the quality experienced by users in different situations.

The results shown in this chapter were derived from the analysis of the behavior of video software, in this case Skype, under different communication environments. More specifically, the chapter will be composed of three sections, the first concerning the analysis of the behavior of Skype in a communication environment having a router as access point. The second section will involve the analysis of the behavior of Skype in a 802.11p-based environment. In the third section, a new solution to the problems encountered will be proposed. Finally, the appropriate conclusions will be drawn.

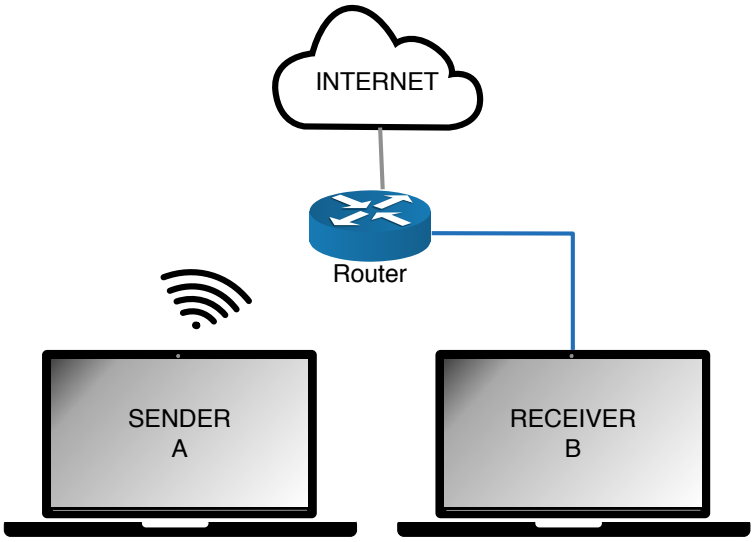


Figure 4.1 *Network Testbed*

4.2 Analysis of the Behavior of Skype Depending on Packet Loss Rate

The starting point established for this thesis was the understanding of the behavior of Skype in an environment characterized by a stable connection between users through the use of a router as an access point. Furthermore, in order not to be influenced in any way by external environment (notifications) of experimentation after several minutes the internet connection is removed. An environment similar the one described in [22] has been recreated for comparison.

Such environments can be described as follow:

- Two hosts (A and B) are connected to a router, A using a wireless connection (802.11a/n/ac mixed), while B a wired connection (Ethernet); Figure 4.1 briefly describes this process.
- After obtaining the IP address (*i.e* both computers connected to the Internet), the Sender (A) makes a video call, using Skype application, to the Receiver (B); each video call is characterized by a single-way video stream (from A to B) without any audio. However, before starting the video call, it is necessary to start the programs in the following order:
    1. Once ManyCam camera simulator is booted, the media Akiyo has to be

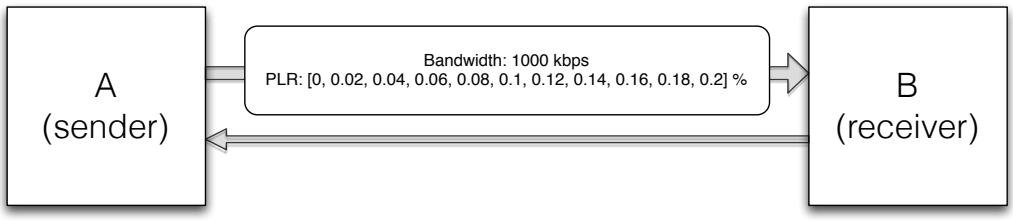


Figure 4.2 *Representation of the first set of simulations*

loaded and loop mode turned on;

2. Then NEWT is used to set the parameters necessary to recreate the network settings;
3. Skype application is thus initiated; in **Tools** > **Options** > **Video Settings**, ManyCam Camera Emulator is set as the main camera.

- The video call is thus initiated, at the moment when the connection is completely established the *Call Technical Info* window is open.
- After few minutes, required for the connection to stabilize, through the terminal, improved by NirCmd, the command for the screenshot is launched.

The Call Technical Info window has been essential in this work since it is the only way to know the characteristics of the active video calling. After the simulations were concluded, the parameters *Upload Bandwidth* (assumed equal to the Sending Rate) and *Bit rate* (assumed equal to the Video Rate) from each screenshot were manually extracted. Nevertheless, it is important to emphasize that, concerning the video rate, the pair of values acquired was considered as maximum and minimum value, for this purpose a mean was considered. This because is the software is not capable of providing a precise value in regards to the video rate while it is sending it (value certain only in reception) and Skype cannot provide any sort of explanation for the window,

In the first environment two sets of simulation have been performed: the aim of the first was to accomplish the same results as described in the reference paper, while in the second the simulations have been extended.

Table 4.1 *Parameters from first set of simulations.*

Simulation Number	Bandwidth, kbps	PLR
1)	1000	0%
2)	1000	2%
3)	1000	4%
4)	1000	6%
5)	1000	8%
6)	1000	10%
7)	1000	12%
8)	1000	14%
9)	1000	16%
10)	1000	18%
11)	1000	20%

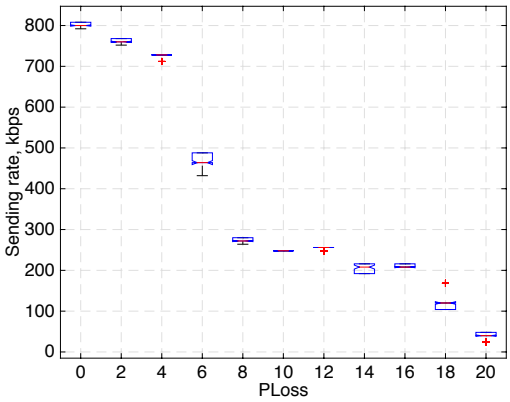


Figure 4.3 *Sending rate depending on packet loss*

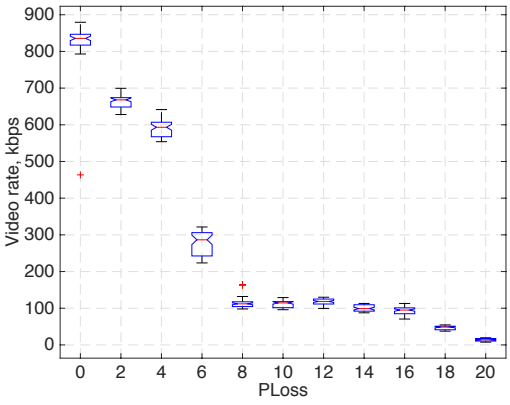


Figure 4.4 *Video rate depending on packet loss*

4.2.1 Simulations Characterized by Constant Bandwidth and Variable PLR

The first set of analysis is composed of eleven simulations, each one with a duration of one minute from the moment the script has been launched; the parameters are reported in Table 4.1.

- Figure 4.3, Figure 4.4 and Figure 4.5 show the obtained results. From the latter, in particular, it is possible to state that Skype tries to use approximately 80% of the available bandwidth, but this value drops when the number of lost packets is increased. Such behavior confirms that Skype uses video codes that

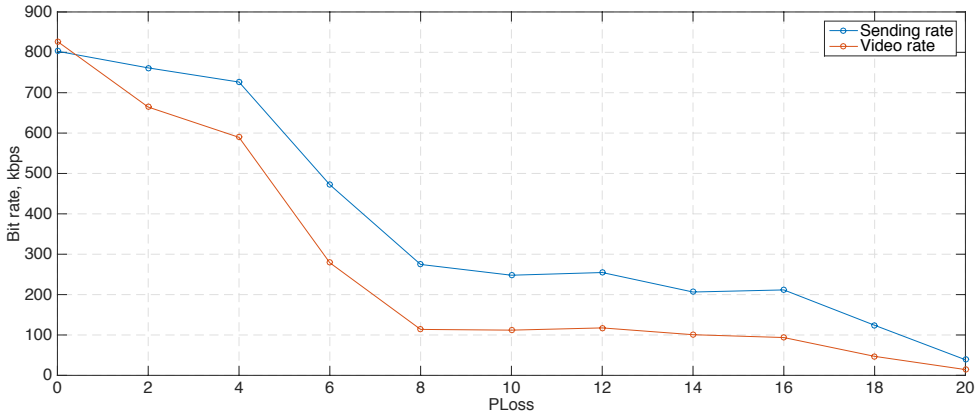


Figure 4.5 *Sending Rate and Video Rate obtained from simulation*

adapt the video rate to the current state of the network.

- Comparing the previously described results with those in [22], it is possible to notice some differences. As a matter of fact the paper states:

  > When the packet loss rate is below 10% Skype works in Normal *(NORM)* state in which its sending rate is loss-ignorant; Skype switches to Conservative *(CONS)* state whenever the packet loss rate goes over 10% [22].

  In contrast, the obtained results demonstrate the existence of a third state between the two previously mentioned. This is because as long as the packet loss rate is below 8% Skype works in NORM state, then it switches to another state - that will be herein defined as Intermediate *(INT)* state - until the packet loss rate reaches 16%, after which it switches to the CONS state.

However, we must say that these reported differences may be caused by different versions of Skype used. The analysis performed in [22] results indeed to be conducted prior to the introduction of Microsoft-operated super nodes.

4.2.2 Simulations Based on the Analysis of the Frames

After the first set of experiments some questions have arisen:

1. What about other characteristics? It would be interesting to repeat similar experiments in regards to frame rate, frame resolution and frame quality adaptation depending on channel rate and PLR.
2. Is it possible to measure the objective video quality, for example Peak Signal-to-noise ratio?
3. What would happen with more dynamic video sequences, for example `Foreman`, instead of `Akiyo`? It is interesting to see the quality of the received video depending on PLR for video with high motion.

In order to go deeper in Skype behavior it was necessary to analyze the frame forming a video. The three parameters that best describe the behavior of the frame within a video are:

1. **Frame Rate:** is defined as "the frequency at which a particular device displays consecutive images known as frames. It can be equally applied to film and video cameras, computer graphics, and motion capture systems. It is usually measured in frames per second (FPS)".
2. **Frame Resolution:** is defined in [23] as "the smallest discernible detail in an image". In our experiments the three resolutions achieved are: 720p, 640x360 and 320x180.
3. **Frame Quality Adaptation:** is usually measured using PSNR metric. PSNR is defined in [24] as "the logarithmic ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation". PSNR is usually expressed in terms of the logarithmic decibel scale, this is because many signals have a very wide dynamic range. The signal in this case is the original data, and the noise is the error introduced by compression. When comparing compression codecs, PSNR is an approximation (adaptation) to human perception of reconstruction quality.

From simulations based on previous questions these conclusions are drawn:

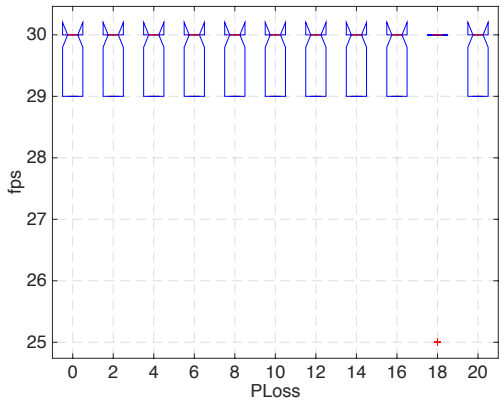


Figure 4.6 *Frame per Second depending on packet loss; fixed bandwidth of 250 kbps*

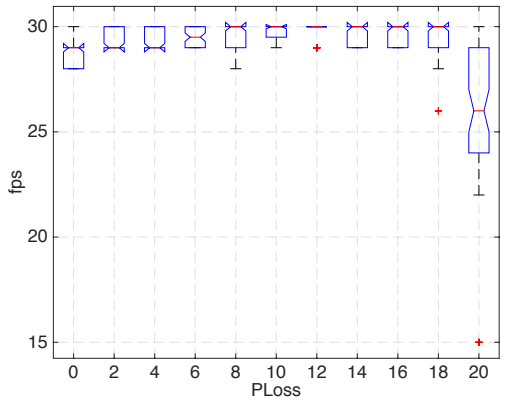


Figure 4.7 *Frame per Second depending on packet loss; fixed bandwidth of 750 kbps*

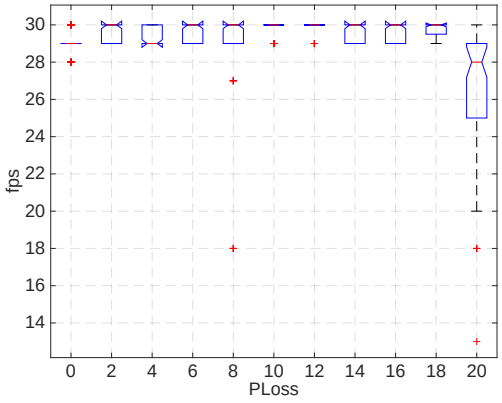


Figure 4.8 *Frame per Second depending on packet loss; fixed bandwidth of 1000 kbps*

1. - Figure 4.6, 4.7 and 4.8 show, respectively, the results obtained for the video `Akiyo` with fixed bandwidth of 250, 750 and 1000 kbps. From the analysis of these plots it is possible to state that when less bandwidth is available, Skype tries harder to maintain the FPS constant. It means that instead of varying its frame rate, when a packet is clearly corrupted, it is skipped and a copy of the previous is sent.
   - Figure 4.9, 4.10 and 4.11 are about resolution with fixed bandwidth of 250, 750 and 1000 kbps. Results are quite the same for 750 and 1000 kbps: when the PLR is null, the maximum resolution is achieved, when the PLR is increased the resolution changes to 640x360, until the PLR reaches 10% and turns in 320x180. This result was expected as in [22]

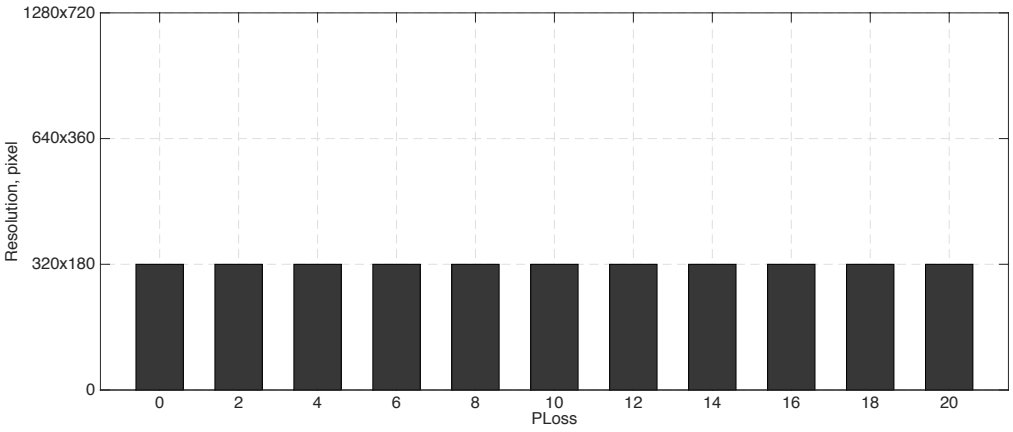


Figure 4.9 *Resolution bar graph for fixed bandwidth of 250 kbps*

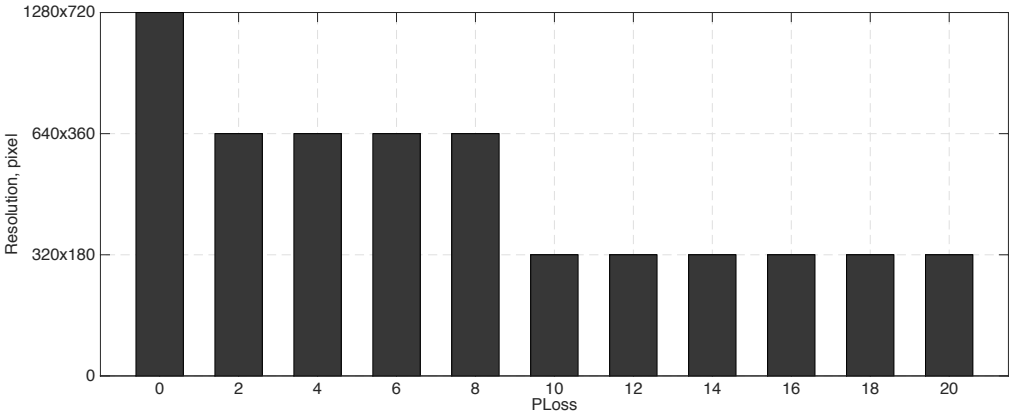


Figure 4.10 *Resolution bar graph for fixed bandwidth of 750 kbps*

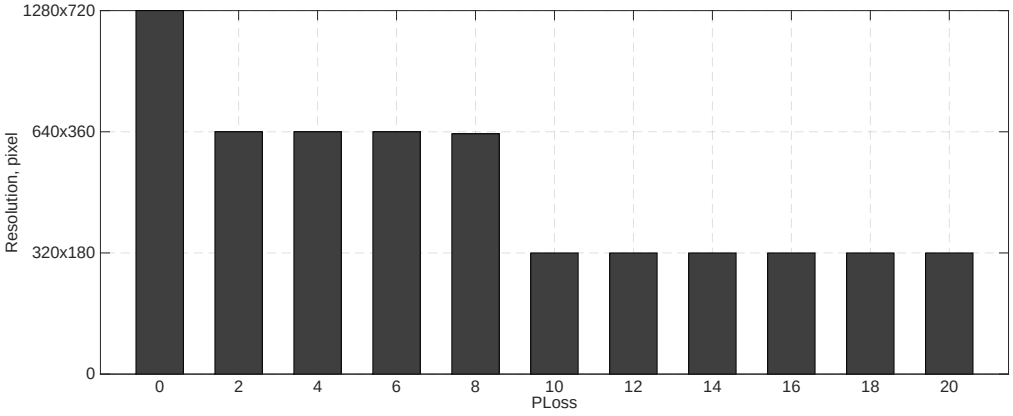


Figure 4.11 *Resolution bar graph for fixed bandwidth of 1000 kbps*

Table 4.2 *Quality experienced for a long call.*

PLR	Resolution, pixel	Quality
0%	640x360	Very good
$\leq 5\%$	640x360	Good
$\leq 10\%$	320x180	Acceptable
$\leq 20\%$	320x180	Poor
$\leq 30\%$	320x180	Unacceptable
$\leq 40\%$	NO	Disconnection

Table 4.3 *Quality experienced in the first four seconds.*

PLR	Resolution, pixel	Quality
0%	640x360	Very good
$\leq 5\%$	640x360	Good
$\leq 10\%$	320x180	Good
$\leq 20\%$	320x180	Acceptable
$\leq 30\%$	320x180	Acceptable
$\leq 40\%$	320x180	Unacceptable

it was stated: "when the packet loss rate is below 10% Skype works in Normal *(NORM)* state", when it reaches over this value it switches to Conservative *(CONS)* state. At approximately 250 kbps, it shows that Skype can recognize that the available bandwidth is not enough to allow an efficient frame quality, and for this it works immediately in *CONS* state, trying to keep as constant as possible its frame rate.

2. The analysis of the quality in the second set of simulations is based on subjective evaluation. More specifically, two tables have been created using NEWT: the maximum achievable bandwidth has been set to a value equal to 500 kbps, which led to a maximum achievable resolution equal to 640×360. Table 4.2 describes the quality experienced by the user when the call lasts long enough to make the connection stable, whereas Table 4.3 provides information about what happens in the first four seconds, describing thus what happens when a new user connects to the network.

   The results provide a description of the amount of artifacts displayed during simulations and are thereby defined:

   - *Very good:* No kind of artifact is displayed;
   - *Good:* Some frame are lost (Figure 4.12);
   - *Acceptable:* Some propagation error, visible frame compression, the lost frames result in visible freezing (Figure 4.13);
   - *Poor:* Some propagation error, poor frame compression, the lost frames result in visible freezing;
   - *Unacceptable:* Significant propagation errors, very poor frame compression, high level of freezing (Figure 4.14);

Figure 4.12 *Example of Good frame quality.*

Figure 4.13 *Example of Acceptable frame quality.*

Figure 4.14 *Example of Unacceptable frame quality.*

- *Disconnection:* After a few seconds the video source is stopped by the application.

3. The video used as a reference for the experiments: `Akiyo`, can be defined as a quasi-static video; it is very interesting to compare the results achieved with those obtained from a video with a higher level of motion. For this purpose the same experiments were also carried out on the video `Foreman`. Moreover, as in the first set of experiments the results seem to proceed linearly the maximum value of PLR considered in this comparison has changed from 20% to 30%.

    - The results have demonstrated that the main differences are in the number of frames per second sent. Indeed the trend of the sending rate in

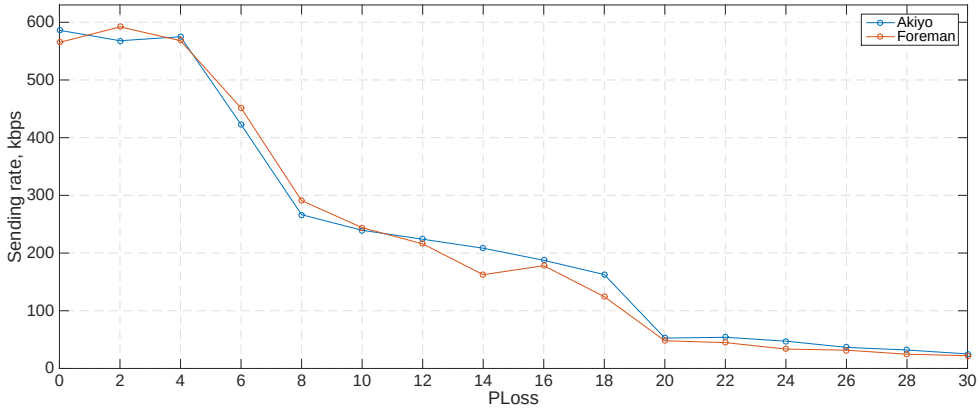


Figure 4.15 *Sending Rate Comparison Akiyo-Foreman: 750 kbps*

Figure 4.15 confirms what has been stated in the first set of simulations, but in addition, from the value 20% it is possible to notice a further state - which can be defined as *More-Conservative* - where the sending rate is kept almost constant for both `Akiyo` and `Foreman`.

- In regards to the video rate, in Figure 4.16 it is possible to analyze the video rate of both `Akiyo` and `Foreman` videos when the bandwidth is fixed at 750 kbps. It is very interesting to compare this figure with the previous one; it is possible to notice that in *CONS* state the difference between the video and sending rate is significant due to the amount of data used to protect packets, which however it is not performed in the third state *(M-CONS)*.
- Figure 4.17 shows the resolution used by Skype to send the video. The results prove that the resolution does not depend on the motion but only by the amount of data that can be sent; in this particular case it is limited by the bandwidth which is fixed at 750 kbps.
- The Frame rate is shown in Figure 4.18; concerning the latter it is possible to state that while in the semi-static video it seems to be constant, while in a video characterized by a higher level of motion it is not possible to achieve the same result.

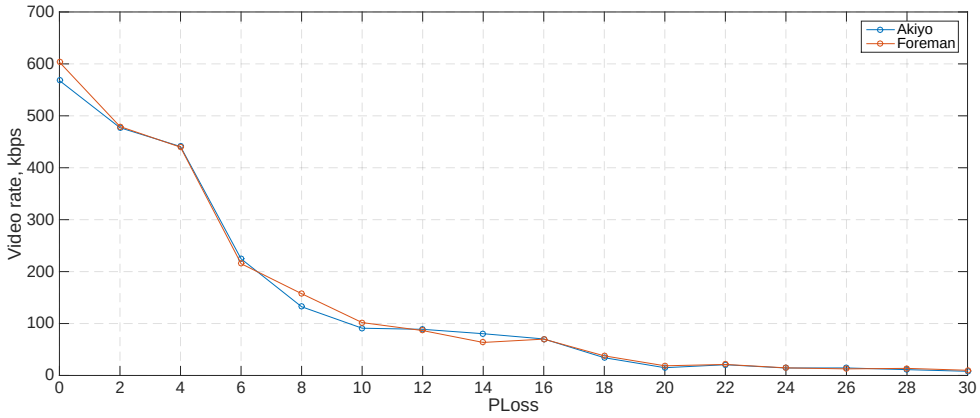


Figure 4.16 *Video Rate Comparison Akiyo-Foreman: 750 kbps*

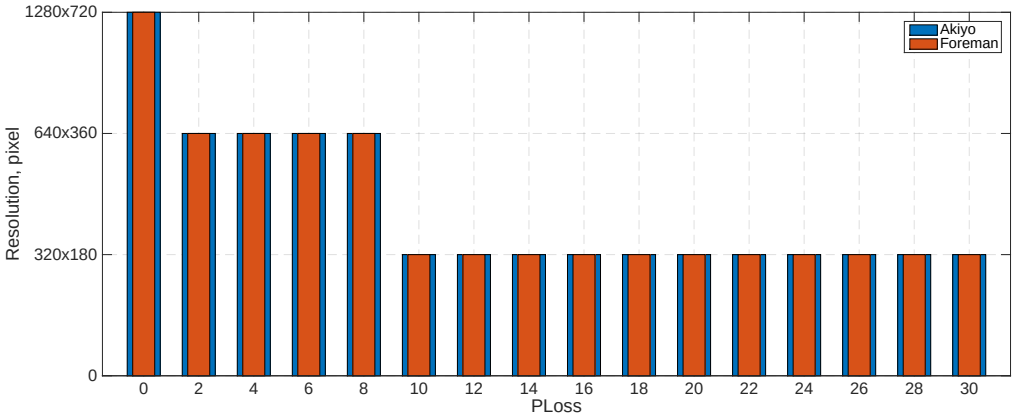


Figure 4.17 *Frame Resolution Comparison Akiyo-Foreman: 750 kbps*

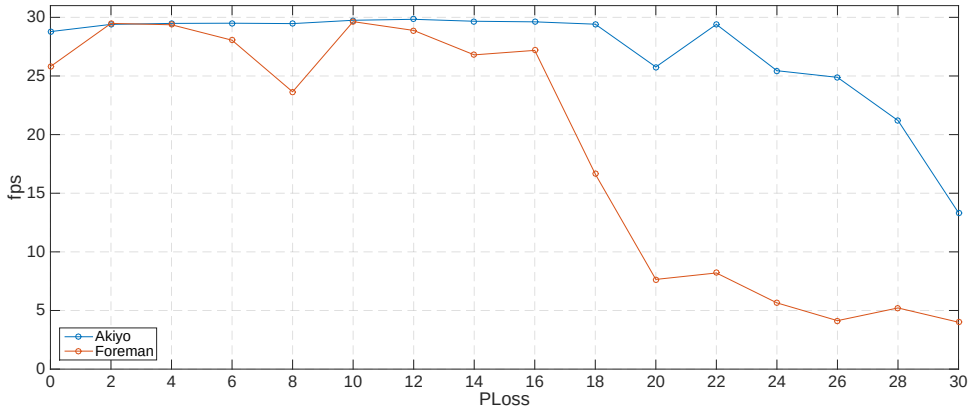


Figure 4.18 *FPS Comparison Akiyo-Foreman: 750 kbps*

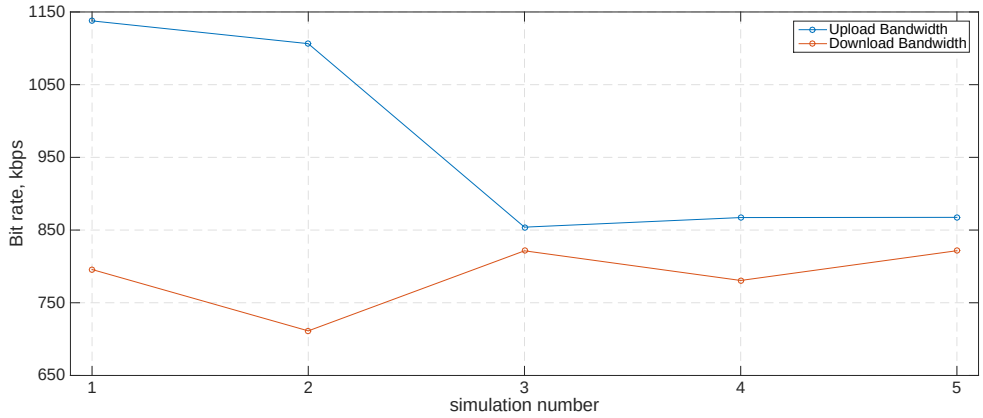


Figure 4.19 *Comparison between Upload and Download Bandwidth in the third set of simulations*

4.3 Analysis of the Behavior of Skype in a VANET Environment

After studying extensively the system based on the use of a router, the analysis has been shifted into the VANET environment where at first three sets of simulations were carried out. At a later stage additional measurements have been performed in order to further investigate the QoS experienced by the user and measured by the PSNR.

4.3.1 Simulations Characterized by Two Users

In the first set of the analysis for this new environment, five simulations have been performed. In this, two computer and two boxes have been deployed. In order to make sure the network is functioning correctly, Wireshark has been used to sniff the `eth0` interface in the first simulation. In each of those the call has been initiated and after 5 minutes the ethernet cable which was connected to the router, the one concerning the Internet connection, has been unplugged. It is very important not to disconnect the `wlan0` interface because whenever it happens the IP address of the computer on this interface is deleted and Skype is not able to properly work. After two more minutes several screenshots of the window *Call Technical Info* has been taken using a script similar to the one mentioned in the previous section. The following plots show the results achieved in this first set of simulations.

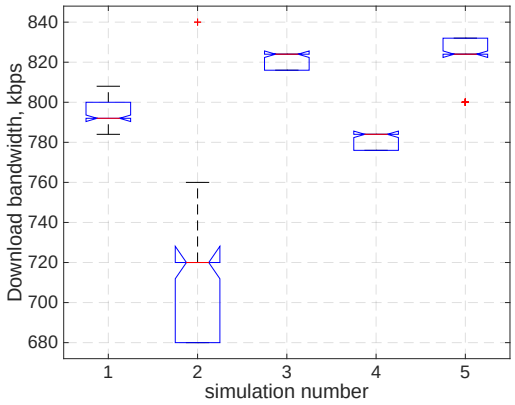


Figure 4.20 *Download Bandwidth per simulation number; third set of simulations*

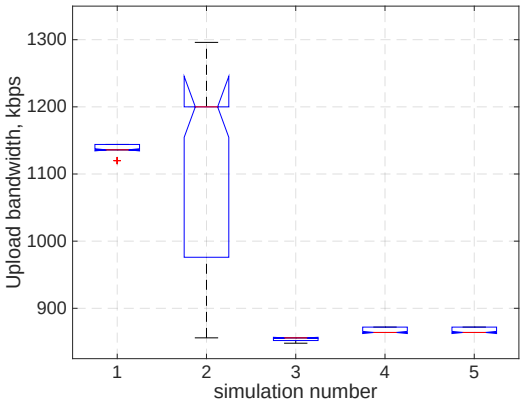


Figure 4.21 *Upload Bandwidth per simulation number; third set of simulations*

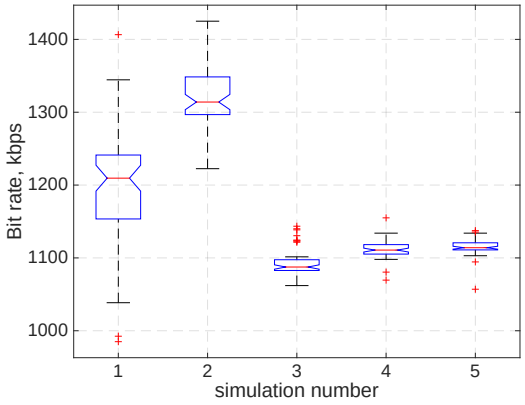


Figure 4.22 *Bit Rate per simulation number; third set of simulations*

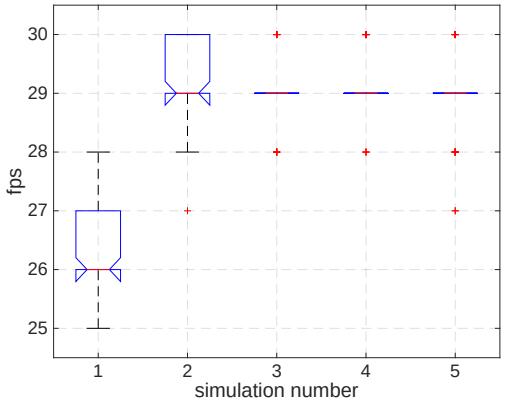


Figure 4.23 *Frame Per Second per simulation number; third set of simulations*

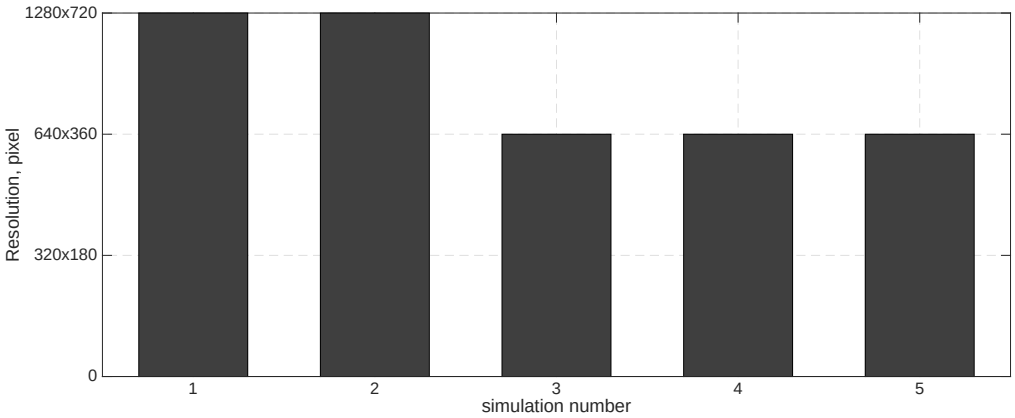


Figure 4.24 *Resolution bar graph in the third set of simulations*

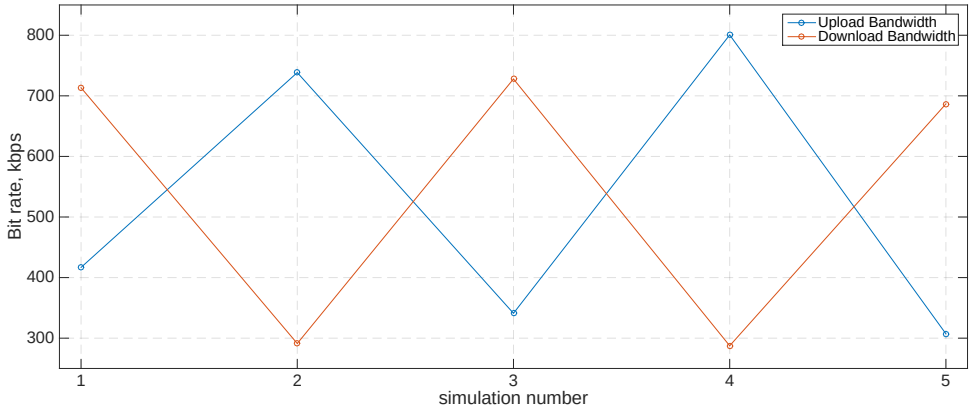


Figure 4.25 *Comparison between Upload and Download Bandwidth in the fourth set of simulations*

- Figure 4.19 shows a comparison between the average upload and download bandwidth for each simulation. It is possible to notice that the amount of bandwidth used to perform the call is not the same as the channel is used only to call, the upload bandwidth for computer n.1 represents the download bandwidth for computer n.2.

- In order to better analyze it, results of both download and upload bandwidth are shown respectively in Figure 4.20 and Figure 4.21. From these figures it is possible to see that, except for simulation n.2, the values obtained are not of great variance.

- In regards to the bit rate (Figure 4.22), simulation n.1 and simulation n.2 present highly variable values, this also affects the frames per second generated (Figure 4.23). However this can be explained by analyzing Figure 4.24: in both simulation n.1 and simulation n.2 Skype tries to work in *NORM* state, this means that it tries to achieve the best result for the video call, in the other simulation it does the opposite.

- As the simulations were taken as a sequence, it is possible to state that even if the computers were disconnected from the Internet, Skype keeps track of the information obtained in order to improve subsequent calls.

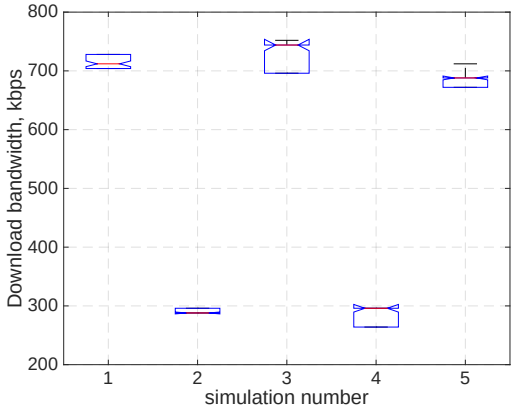


Figure 4.26 *Download Bandwidth per simulation number; fourth set of simulations*

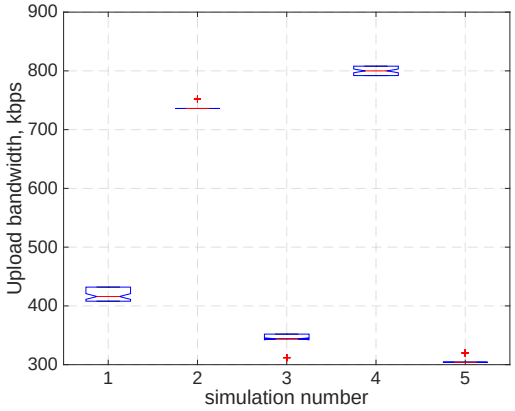


Figure 4.27 *Upload Bandwidth per simulation number; fourth set of simulations*

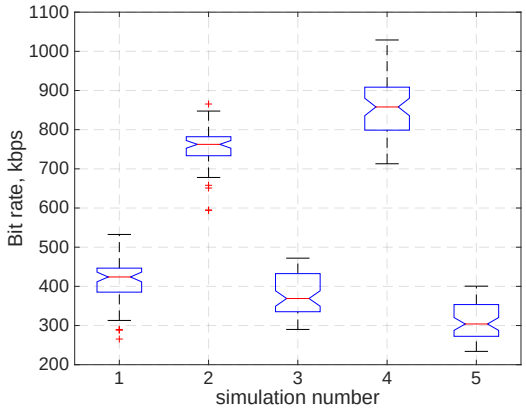


Figure 4.28 *Bit rate per simulation number; fourth set of simulations*

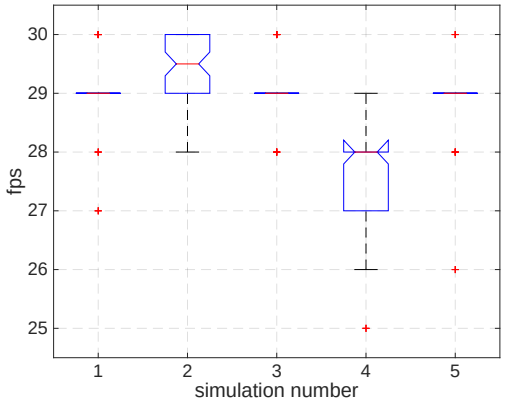


Figure 4.29 *Frame Per Second per simulation number; fourth set of simulations*

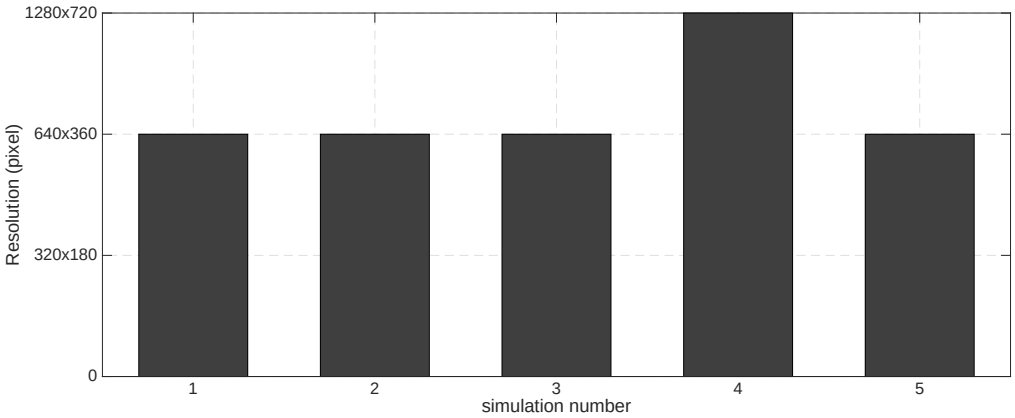


Figure 4.30 *Resolution bar graph in the fourth set of simulations*

4.3.2 Simulations Characterized by Four Users: Calls Made Non-Simultaneously

In the fourth set of simulations two more computers and boxes were added. In this scenario, the call between computer n.1 and n.2 has been initiated and after five minutes the call between computer n.3 and n.4 has taken place, after five more minutes the internet cable has been unplugged. At the twelfth minute the script for the screenshot has been launched. The results reported in the following figures have all been obtained from computer n.1.

- From Figure 4.25 it is possible to notice that not always the same computer achieves the most bandwidth, but the average value between the two different computers remains constant throughout the simulations. It is know from previous simulations that in order to achieve a PLR almost completely equal to zero, the bit rate between the antennas must not exceed a certain value (550 kbps). In those simulations, the PLR is always approaching zero, this is because the only packets which are dropped are those that are transmitted in the process of setting up a call; i.e. when there are more than two devices simultaneously on the same VANET; in this particular environment close to the fifth minute.
- Even in this set of simulations, the results of both download (Figure 4.26) and upload (Figure 4.27) bandwidth have been carried out. Those show less variable values if compared with the previous simulations, however it is not possible to state the same in regards the bit rate (Figure 4.28).
- Figure 4.29 is related to the frame per second, which shows constant values for simulation n.1, n.2 and n.3, where the bit rate and upload bandwidth show values that are not so high: this means that it is working in a mode close to the *CONS* (conservative).
- In Figure 4.30 where, as expected, it is possible to notice that except for simulation n.4, the resolution used in transmission is 640x360 is the best compromise between the fixed FPS and not-so-high bit rate.

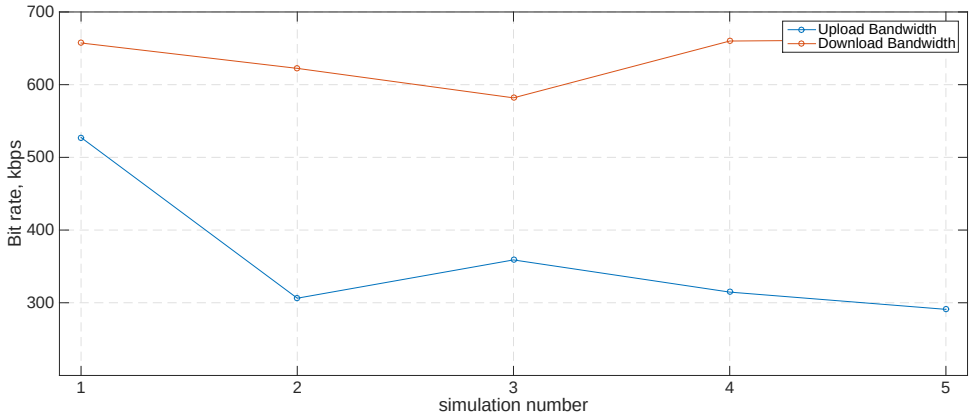


Figure 4.31 *Comparison between upload and download bandwidth in the fifth set of simulations*

4.3.3 Simulations Characterized by Four Users: Calls Made Simultaneously

The fifth set of simulations is exactly like the previous, but this time instead of starting the call with a delay, both of the calls are performed at the same moment, and then after five minutes the internet cable of the router is unplugged, and after two more minutes several screenshots of Skype's *Call Technical Info* window have been taken.

- In this case, as in the third set, one of the two computers acquires more bandwidth in every simulation as shown in Figure 4.31, although in this case the difference is not so pronounced.
- More in particular, analyzing the results for download bandwidth in Figure 4.32, it is possible to notice more variable values in the first three simulations. In Figure 4.33, related to the upload bandwidth, this is true just for the first simulation.
- Then, comparing the bit rate (Figure 4.34), it is possible to notice the complete absence of protection for the packets: all that is generated is sent, which is characteristic of the *CONS* mode.
- Further confirmation is shown also in Figure 4.35 and Figure 4.36, where the frame per second and resolution are fixed and almost constant.

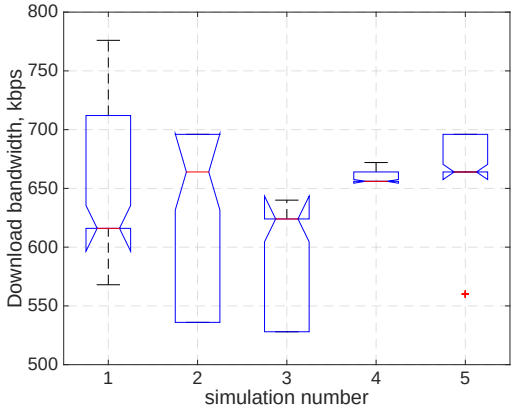


Figure 4.32 *Download Bandwidth per simulation number; fifth set of simulations*

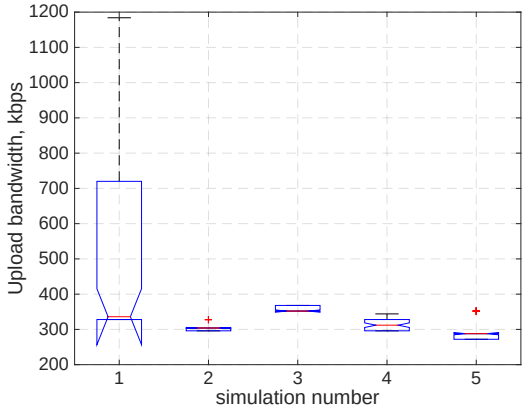


Figure 4.33 *Upload Bandwidth per simulation number; fifth set of simulations*

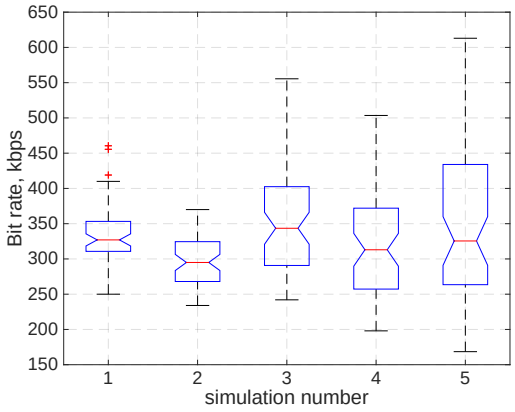


Figure 4.34 *Bit rate per simulation number; fifth set of simulations*

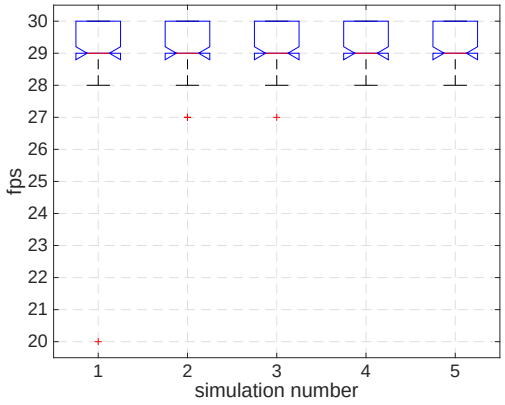


Figure 4.35 *Frame Per Second per simulation number; fifth set of simulations*

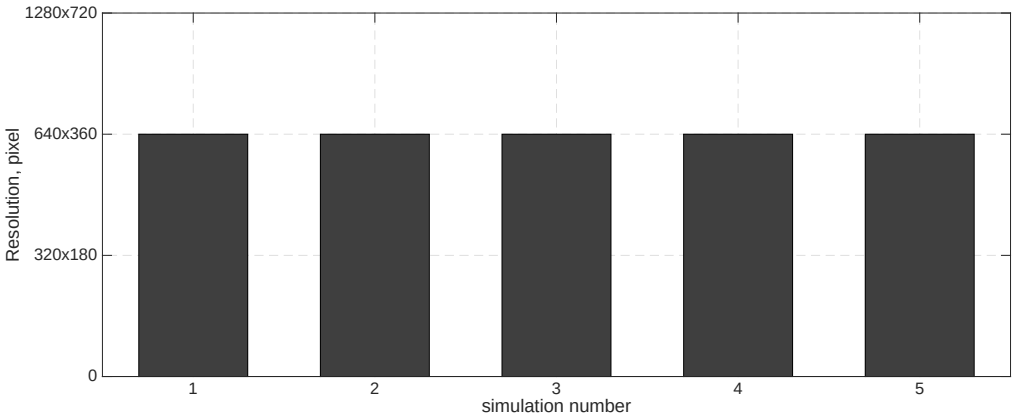


Figure 4.36 *Resolution bar graph in the fifth set of simulations*

Through the analysis of Skype in a VANET environment it has been possible to acquire further confirmations regarding most of the results previously expected. First of all, from previous simulations it has been stated that for an average transmission of less than 550 kbps in a four-device VANET environment, the Packet Loss Rate is close to zero. It is very impressive how Skype controls this value in order not to lose the packets sent. The way it manages the available bandwidth allows the transmission to have a good resolution (640x320) and an almost constant and maximum frame rate (29 FPS), also when the bit rate is not so high.

4.3.4 Simulations Based on the Analysis of Bit Rate and PSNR

Even in this case, as in the previous section, in order to acquire further information about Skype behavior at a later stage additional simulations have been carried out. Those measurements have been characterized by four factors: longer simulation, more users, intermittent users and a non-compressible video source.

In all of this, the focus was on the bit rate achieved by the different users. The goal was to demonstrate that each user independently tries to estimate the bandwidth by itself to choose the sending bit rate and protection scheme for the video bit stream.

Along with the bit rate another factor on which the focus was on is the PSNR. The analysis of it has been carried out using the AviSynth tool, previously described, in order to provide an objective value. PSNR is very important because it provides objective information about video quality for each simulation. Typical values include:

- $PSNR > 40$ dB: very good quality;
- 30 dB $\leq PSNR \leq 40$ dB: good and acceptable;
- $PSNR < 30$ dB: poor.

Each figure regarding the PSNR follows the one regarding the bit rate as it is highly correlated, however, these assist with understanding more effectively whether the result is acceptable or not in terms of quality.

Time comparison-based simulations

The first kind is characterized by three different measurements:

1. In the first, computer n.1 and computer n.2 establish the call, after five minutes computer n.3 and computer n.4 start their call (i.e. there is a five minutes delay in starting the call), the total measurement duration is ten minutes.
2. The second measurement is similar to the first, except that all computers establish the call simultaneously.
3. At the beginning of the study only two measurements were planned for but, as Figure 4.37 was particularly interesting, the idea of the third measurement came about. This is because there is a need to understand if the behavior of the users would have led to a stationary state or not. This measurement is thus similar to the first, except that the duration is twenty minutes.

Results from these measurements have shown that:

- A comparison between Figure 4.37 and Figure 4.39 show that in the case with delay, computer n.1 and n.2 have been affected from computer n.3 and n.4, in particular the higher the bit rate of computer n.3, while computer n.1 is lower. On the contrary, in the case where all the connections start at the same time, the bit rate oscillates in a 200 kbps interval presenting less peaks than the previous case.
- From Figure 4.41 it is possible to notice that even if there is not a continuous value for the bit rate, after the fifth minute the average is more or less the same.
- What has been stated in the previous point is also reflected exactly like the PSNR (Figure 4.38, Figure 4.40 and Figure 4.42). In all cases, only two users are able to ensure a level of PSNR higher than 34 dB for almost the entire duration of the simulation, but all of them exceed the threshold of 30 dB for most of the transmission.

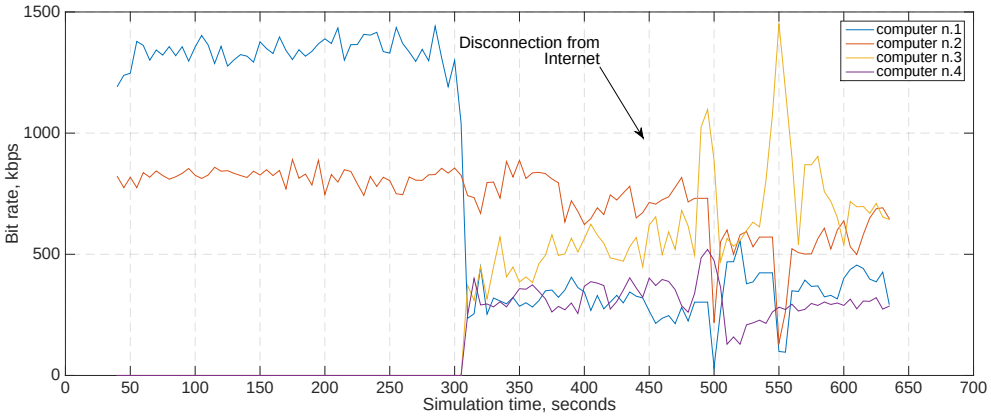


Figure 4.37 *Bit rate comparison for a 10 minutes simulation with delay*

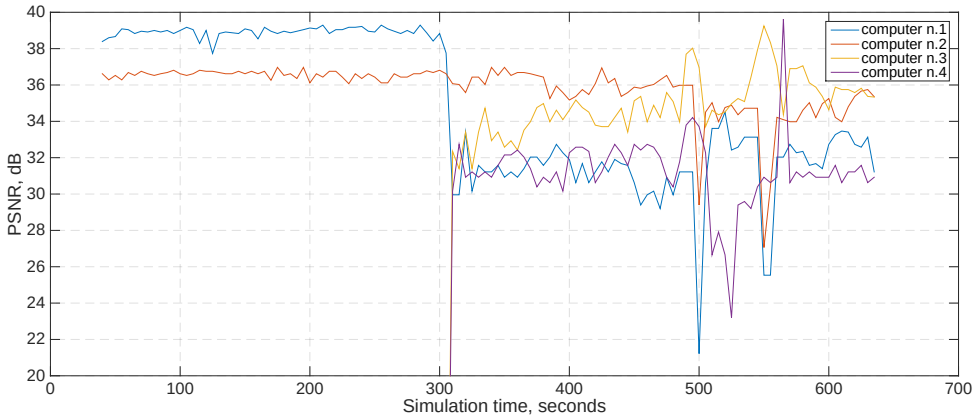


Figure 4.38 *PSNR comparison for a 10 minutes simulation with delay*

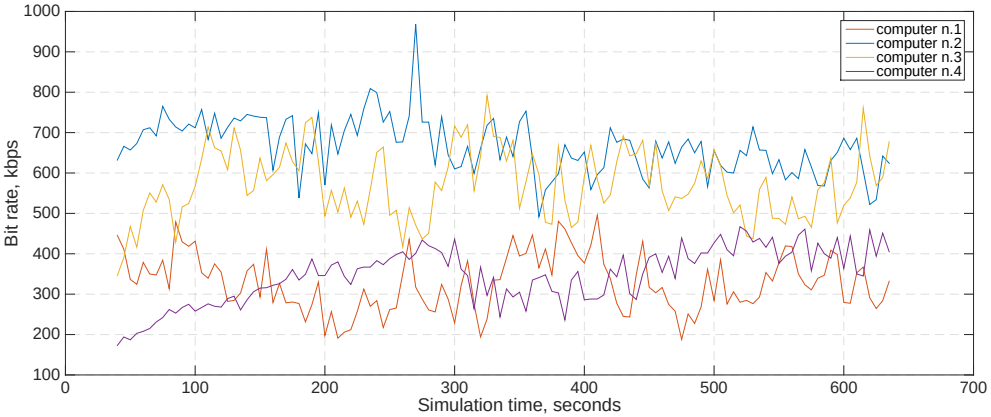


Figure 4.39 *Bit rate comparison for a 10 minutes simulation without delay*

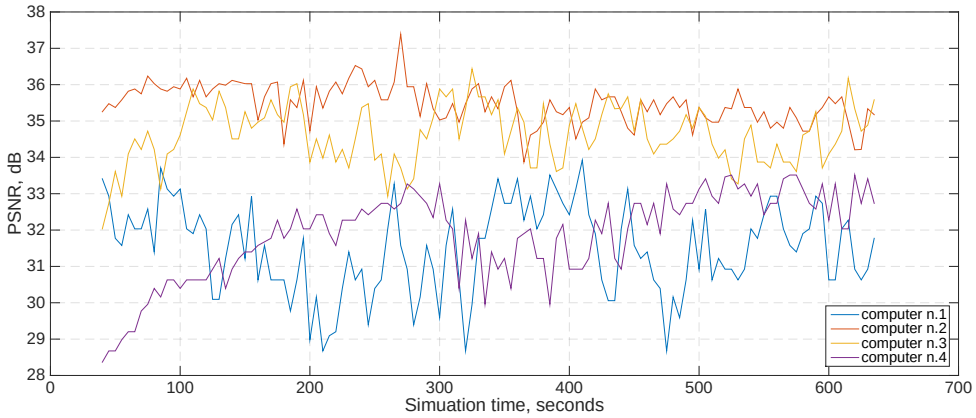


Figure 4.40 *PSNR comparison for a 10 minutes simulation without delay*

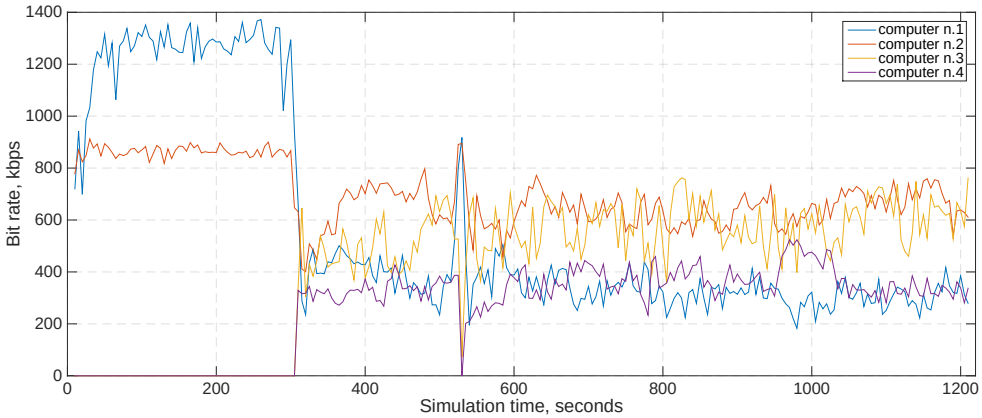


Figure 4.41 *Bit rate comparison for a 20 minutes simulation with delay*

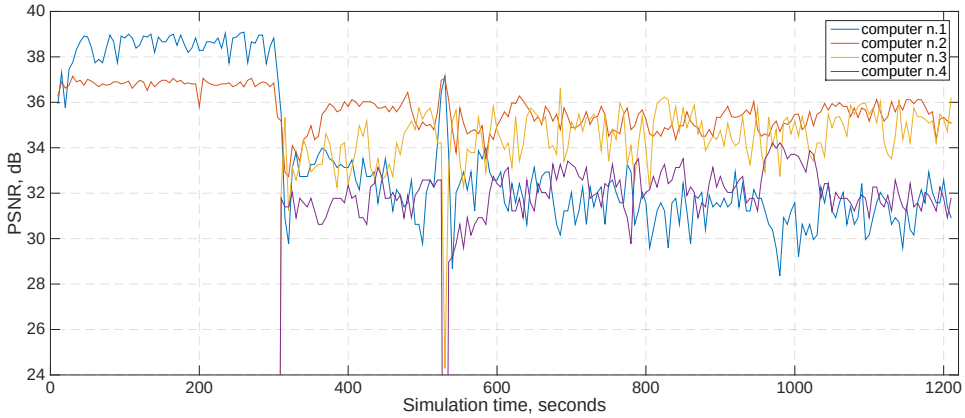


Figure 4.42 *PSNR comparison for a 20 minutes simulation with delay*

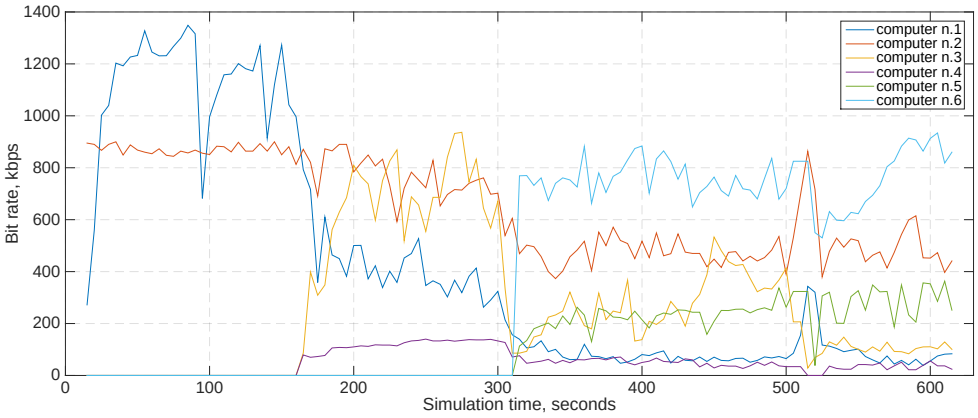


Figure 4.43 *Bit rate comparison for a 10 minutes simulation for 6 users*

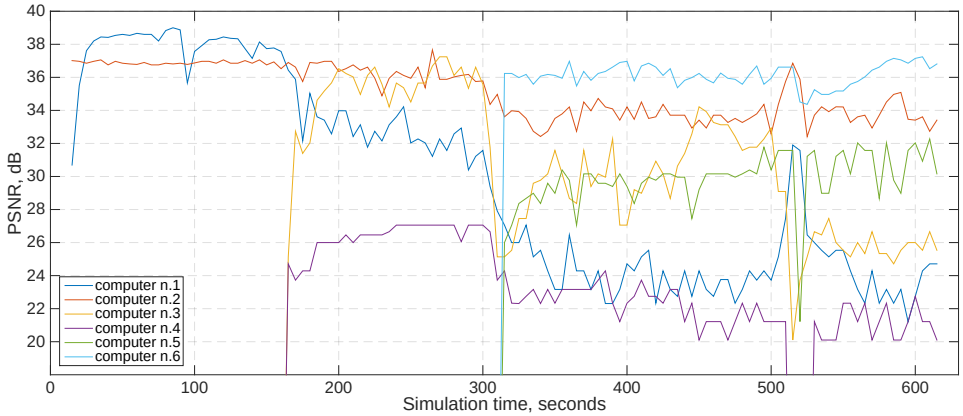


Figure 4.44 *PSNR comparison for a 10 minutes simulation for 6 users*

Six users simulations

A similar experiment has been carried out with more users. In this case the delay between the users is 150 seconds, this means that computer n.5 and computer n.6 start their conversation after five minutes.

- As shown in Figure 4.43 it is very interesting to notice how Skype tries to push the call to the limit. This is because the smallest values in bit rate are achieved after new users start the conversation, or after the data reaches the highest possible rate.
- In Figure 4.44 it is possible to notice that at the beginning of the conversation the two users reach a level of PSNR over 35 dB, however, increasing the number

of users in the simulation the values decrease, carrying the first user to a value close to 25 dB (i.e. very bad quality). The user who can achieve the best quality is n.5 which maintains a PSNR level close 36 dB: this is the best example of how, in this environment, each user chooses its own sending rate.

Noise video simulations

Afterwards the focus was on the packet loss rate. The first step was to understand if the video was responsible in any way for the lost of packets. The best possible outcome has been achieved reproducing the same experiment but using a different video source. The name of the video used is `Noise.avi` and it is a noisy Matlab-generated video; its main characteristic is that Skype requires a large amount of data to transmit it as it is not possible to compress.

- Figure 4.45 shows that, as expected, the bit rate is not stable at all as the video cannot be compressed and the amount of bandwidth necessary to transmit it is not reached by the users.
- Figure 4.46 shows that the packet loss rate reaches its peak when new users join the network, the only problem in this is in regards the values: to build the packet loss rate plot only the maximum values have been used and sometimes those values are too high for a percentage to be real (the maximum value is 295%).
- About PSNR (Figure 4.47) it is possible to notice that it never reaches the 40 dB necessary for optimum quality, but on the contrary, with the increase in the number of users it gets closer to the negative limit, and in some cases it exceeds it.

Intermittent users simulations

As previously mentioned the last kind of measurement has been characterized by intermittent users. This means that in the last simulation computer n.1 and computer n.2 begin the conversation, computer n.3 and n.4 communicate from second 40 to 60, then after 20 second they start a new conversation for 20 seconds and so on, total measurement simulation is 5 minutes.

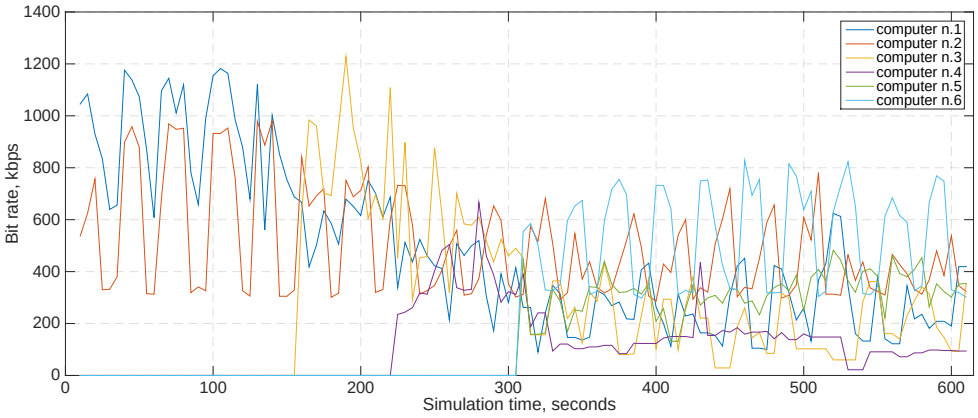


Figure 4.45 *Bit rate comparison for a 10 minutes simulation: Noise.avi*

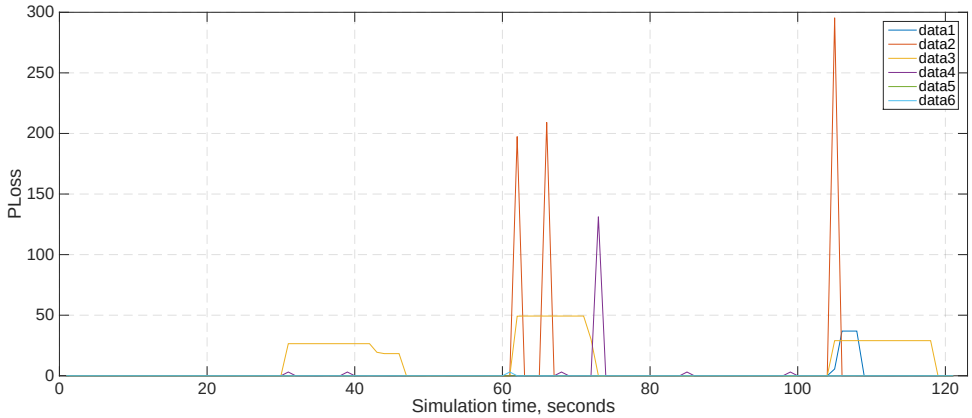


Figure 4.46 *Packet Loss Rate for a 10 minutes simulation: Noise.avi*

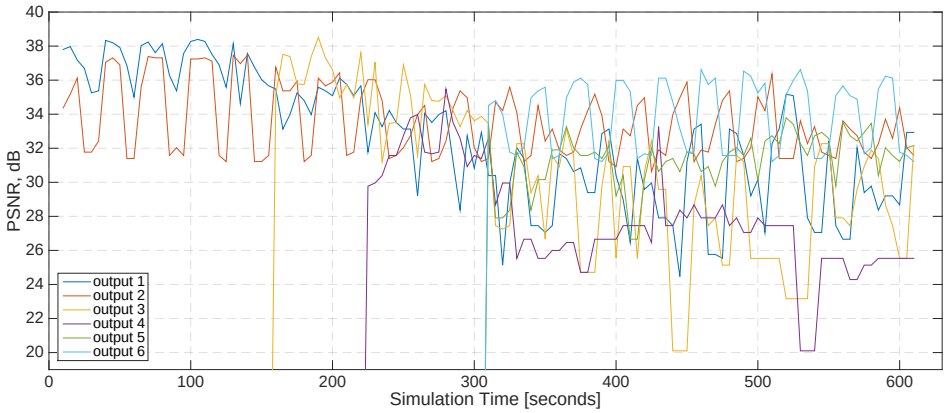


Figure 4.47 *PSNR for a 10 minutes simulation: Noise.avi*

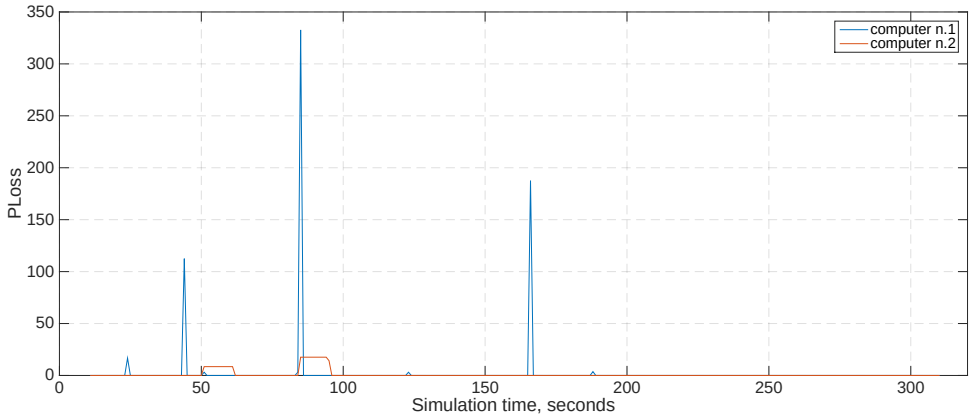


Figure 4.48 *Packet Loss Rate for a 5 minutes simulation: Foreman.avi*

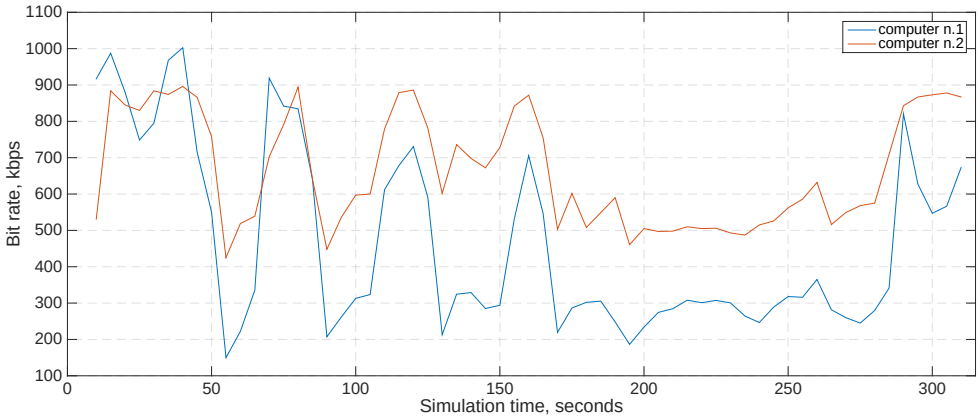


Figure 4.49 *Bit rate comparison for a 5 minutes simulation: Foreman.avi*

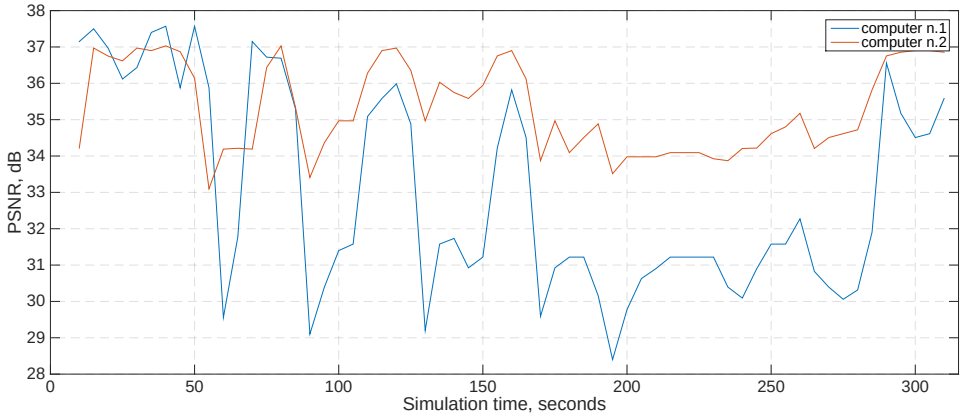


Figure 4.50 *PSNR for a 5 minutes simulation: Foreman.avi*

- From Figure 4.48, it is possible to notice that even if the maximum value of PLR is over 100%, the packets are lost mostly when computer n.3 and computer n.4 start a new call.
- What is stated in the previous point explains the drops in bit rate shown Figure 4.49. This is also reflected in the values achieved in regards to the PSNR (Figure 4.50).

Results

The results obtained in this section have demonstrated that, in the analyzed environment, each channel without coordination suffers of two effects. These effects arise from the complete ignorance about the presence of other users in the network. Thus implies the need, for each user, to survive, leading to a completely egotistical behavior by each of them. Those effects are:

- The PSNR achieved in the simulations does not show constant values, it is therefore not possible to reach a fixed value about quality. It is also not possible to ensure the constant behavior in the same operative state, resulting in a further deterioration of the quality perceived by users.
- Adding a new user in the networks implies the necessity of a higher bandwidth capability and a higher number of simultaneous transmissions, in particular, the latter involves collisions which cause the loss of packets. This leads to a lower PSNR value, i.e. a degradation of the perceived quality.

4.4 Proposed Solution

A possible solution to the issues experienced in the previously described simulations can be achieved by using the system described in Section 2.1. The proposed solution involves a greater knowledge from users regarding their surroundings in order to avoid situations in which each user independently tries to estimate the bandwidth by itself. This prevents the latter from choosing the sending bit rate and protection scheme for the video bit stream. This can be achieved by transmitting more information in the packets sent, in order to force an equally-distributed bit rate.

To provide this, every user must send its ID number to the RSU which it is connected, the header of each packet must contain the unique ID of the transmitter. This is because it is possible that more than one user would try to connect to a single RSU. When this happens, it would be able to count the number of users through the ID value sent from each transmitter. It would therefore possible, for the RSU, to send an ACK to each user containing the maximum value of the bit rate at which their transmitter must send its data. Furthermore since stations cover adjacent areas, when the RSU senses that a user is going to switch to a subsequent station, it could warn the RSU responsible for covering such area concerning the imminent arrival of the new user; thus allowing the RSU to change the value of the bit rate in a controlled fashion and enabling further decrease of the probability of collision. The inclusion of such limitation would ensure a lower packet loss, resulting in a more stable value of PSNR.

Let us imagine that six users are trying to connect to the same system, like the case described in the previous section. Figure 4.51 shows exactly this, in addition two more curves have been added: total active and average active. The first shows the sum of all the bit rates, while the second shows the average. From the total active curve it is possible to determine the maximum load that the channel can support. As at the beginning of the conversation the users involved in the conversation are only two, each RSU should sent an ACK containing the value 1024. In this simulation, after 150 seconds, two more users start to transmit. When this occurs, the RSU, reading two more IDs, should force the bit rate to (at most) 512 kbps. In order to not lose packets, before the two new users can start a proper transmission, the RSU should check that the first two users involved in the conversation have actually reduced their bit rate.

The same thing should happen at the 300th minute when two more users join the

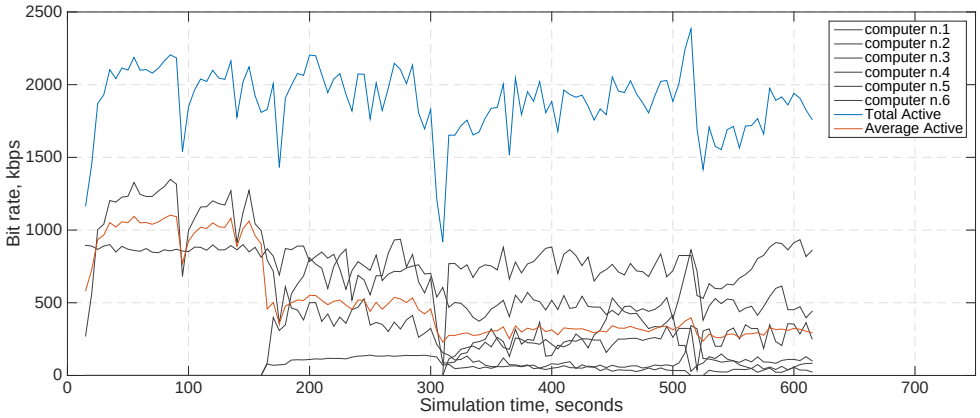


Figure 4.51 *Previous experiment with Total and Average curves.*

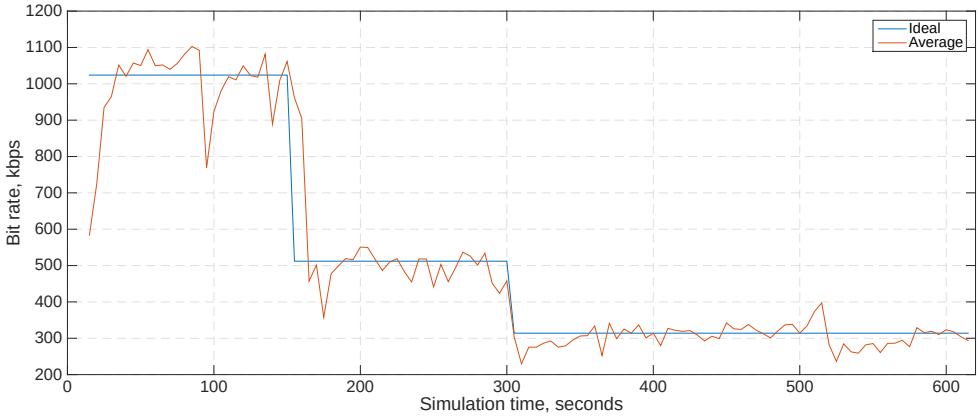


Figure 4.52 *Comparison between Ideal case and Average curve.*

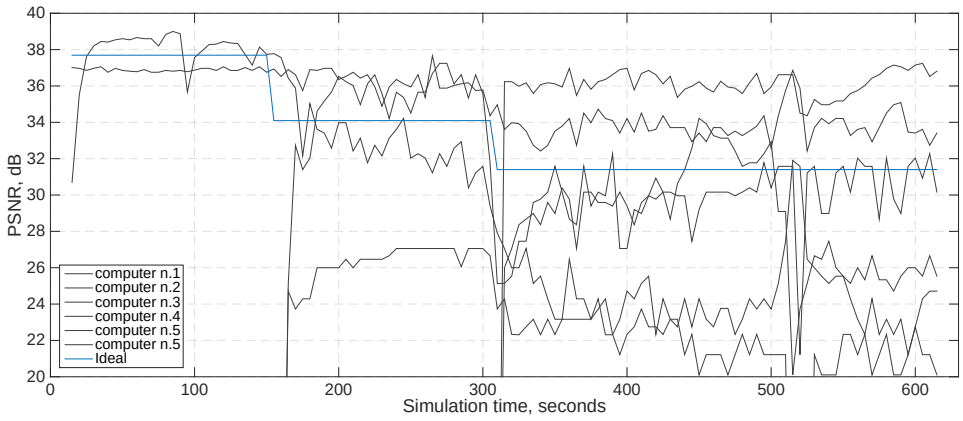


Figure 4.53 *Comparison between PSNR for 6 users and Ideal case.*

simulation. In this case, the RSU should force the the bit rate to 314 kbps. An ideal case is represented in Figure 4.52 where it is possible to notice that, the use of this technique, would lead not only to a result which follows the average of the values obtained in the simulation, but also would provide protection against packet loss (i.e. negative peaks). This implies a more stable PSNR value, as represented in Figure 4.53. In the latter it is possible to notice that the ideal PSNR would always be between 30 and 40 dB which, as previously mentioned, means not only a good and acceptable quality, but it would ensure that every user involved could take advantage of this service. On the contrary, it is not possible to ensure this in an environment where resources are allocated in a conventional way since, as demonstrated from the results, only three users can achieve this quality and with some negative peaks.

4.5 Conclusions

An examination of the behavior of Skype in different network environments it has been carried out. From the analysis has been possible to state the key points:

- By using a network emulator it was possible to demonstrate that Skype operates in four different states depending on the amount of lost packets, such states are:

  1. Normal state: $0\% \leq \text{PLR} < 8\%$;
  2. Intermediate state: $8\% \leq \text{PLR} < 16\ \%$;
  3. Conservative state: $16\% \leq \text{PLR} < 20\ \%$;
  4. More-Conservative state: $\text{PLR} \geq 20$.

- The results obtained by measurement in a 802.11p-based environment showed that the PSNR never reaches a fixed value but instead presents several fluctuations. This means that it is not possible to provide the same level of quality due to the behavior of the users, who try to estimate, by themselves, the channel required for the stream transmission parameter.

- Whenever a new user joins the network, a higher level of bandwidth is required since the number of simultaneous transmissions increases. Such an effect therefore causes an increase in the probability of collisions, which undoubtedly affects the level of quality perceived by the user.

- Increasing the knowledge of the devices involved in the network, it would be possible to eliminate the egotistical behavior discussed above. For this purpose a new solution to the problems related to PSNR encountered in this work has been proposed.

5. CONCLUSIONS

This thesis studies channel resource allocation for multi-camera video streaming in vehicular ad-hoc networks. In order to achieve this, several studies have been conducted in various environments, these have been divided in two parts and have been described in chapter 4. In the first part, an analysis of a system having a router as an access point has been conducted. The results achieved were vital for the comprehension of the behavior of Skype when the PLR is increasing. Moreover, the existence of several working modes in Skype's algorithm have been proved.

Before proceeding with the second part of the simulations, research on the standard used in the VANET environment was conducted. A study on the standard 802.11p was also conducted. From the theoretical point of view it was necessary to understand what was happening at the Physical and MAC layer. The actual development of the VANET environment required a more practical approach, which involved modifying the boxes connected to the antennas in the laboratory, to create a 802.11p-based environment.

In the second part of the work, several simulations were performed. In this case the various simulations were mostly related to the number of users, the video used and the simulation time. From this core point of the work, the most important results were achieved about PSNR. It was demonstrated that the PSNR never reaches a fixed value but instead presents several fluctuations. Such effects were associated with the behavior of each user, as each of them tries to transmit at the maximum possible rate. Furthermore, because of this behavior, it was demonstrated that whenever a new user joins the network, the level of simultaneous transmission increases while the level of quality suffered a decline due to collisions.

In the last section, a solution to the previous problem was proposed. Such a solution, based on removal of the selfish behavior from the users, represents a possible future development.

BIBLIOGRAPHY

[1] Moustafa, Hassnaa, and Yan Zhang. "Vehicular networks: techniques, standards, and applications". Auerbach publications, 2009.

[2] Vegni, Anna Maria, Mauro Biagi, and Roberto Cusani. Smart vehicles, technologies and main applications in vehicular ad hoc networks. INTECH Open Access Publisher, 2013.

[3] Hartenstein, Hannes, and Kenneth P. Laberteaux. "A tutorial survey on vehicular ad hoc networks." Communications Magazine, IEEE 46.6 (2008): 164-171.

[4] Bellalta, Boris, et al. "Performance evaluation of IEEE 802.11 p-enabled vehicular video surveillance system." Communications Letters, IEEE 18.4 (2014): 708-711.

[5] Abdelgader, Abdeldime MS, and Wu Lenan. "The Physical Layer of the IEEE 802.11 p WAVE Communication Standard: The Specifications and Challenges." Proceedings of the World Congress on Engineering and Computer Science. Vol. 2. 2014.

[6] Miao, Lusheng, et al. "A survey of IEEE 802.11p MAC Protocol." Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT) (2011): 1.

[7] Campolo, Claudia, et al. "Modeling broadcasting in IEEE 802.11 p/WAVE vehicular networks." Communications Letters, IEEE 15.2 (2011): 199-201.

[8] Krishnan, Murali E., E. Gangadharan, and Nirmal P. Kumar. "H. 264 Motion Estimation and Applications." INTECH Open Access Publisher, 2012.

[9] E.Belyaev, A.Vinel, A.Surak, M.Gabbouj, M.Jonsson and K.Egiazarian. "Robust vehicle-to-infrastructure video transmission for road surveillance applications." Vehicular Technology, IEEE Transactions on 64.7 (2015): 2991-3003.

[10] MacKay, David JC. "Fountain codes." Communications, IEE Proceedings-. Vol. 152. No. 6. IET, 2005.

[11] Shokrollahi, Amin. "Raptor codes." Information Theory, IEEE Transactions on 52.6 (2006): 2551-2567.

[12] Reed, I. S. and Solomon, G., "Polynomial Codes Over Certain Finite Fields," SIAM Journal of Applied Math, vol. 8, 1960, pp. 300-304.

[13] Dovrolis, Constantinos, Parameswaran Ramanathan, and David Moore. "Packet-dispersion techniques and a capacity-estimation methodology." Networking, IEEE/ACM Transactions on 12.6 (2004): 963-977.

[14] Luca De Cicco, S. Mascolo, and V. Palmisano. "A Mathematical Model of the Skype VoIP Congestion Control Algorithm." (2008).

[15] Baset, Salman A., and Henning Schulzrinne. "An analysis of the skype peer-to-peer internet telephony protocol." arXiv preprint cs/0412017 (2004).

[16] Vision Media Inc., "ManyCam". [Online]. *"https://manycam.com"*

[17] A. Tirumala, M.Gates, F. Qin, J.Dugan and J. Ferguson. "Iperf - The TCP/UDP bandwidth measurement tool". [Online]. Available: "https://iperf.fr".

[18] OpenManiak, "IPERF - The easy tutorial". [Online]. Available: "http://openmaniak.com/iperf.php".

[19] AviSynth: a powerful non linear scripting language for video. *"http://avisynth.nl/index.php/Main_Page"*

[20] Non Linear Editing: *http://avisynth.nl/index.php/NLE*

[21] OpenWrt Developers, "OpenWrt Wireless Freedom". [Online]. *"https://openwrt.org"*

[22] Zhang, Xinggong, et al. "Modeling and analysis of Skype video calls: Rate control and video quality." Multimedia, IEEE Transactions on 15.6 (2013): 1446-1457.

[23] R. C. Gonzalez and R. E.Woods, Digital Image Processing, 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 2007

[24] Yousuf, M. A., and M. N. Nobi. "A new method to remove noise in magnetic resonance and ultrasound images." Journal Of Scientific Research 3.1 (2010): 81.