3.1.- Simulation modeling specification roll-out

To allow a fairly independent development of each simulation modelling activity (even of other workpackages), at different level, with as many elements as possible, and also carry out the validation of each newly introduced element, a basic but general simulation chain frame has been defined, that will rely on file exchanges at each interface of main blocks, with three main formats for file depending on the place in the chain they correspond to: **application** level, **network** level and **physical** level.

The emission side for this general simulation chain is presented in Figure ? and the reception side is presented in Figure 3.1.

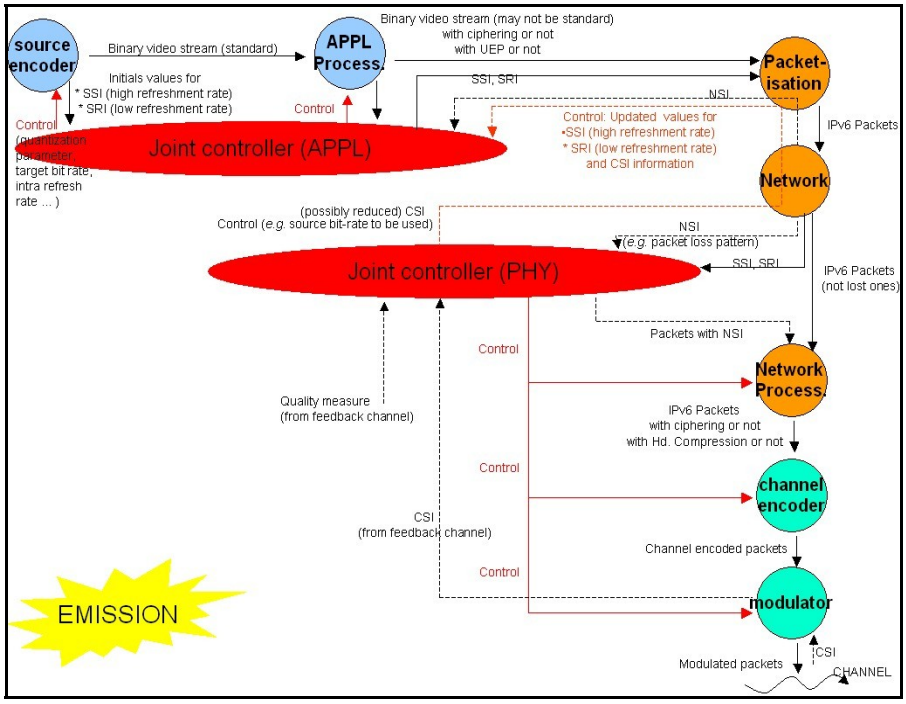

Figure 3.1: General simulation chain: emission side.

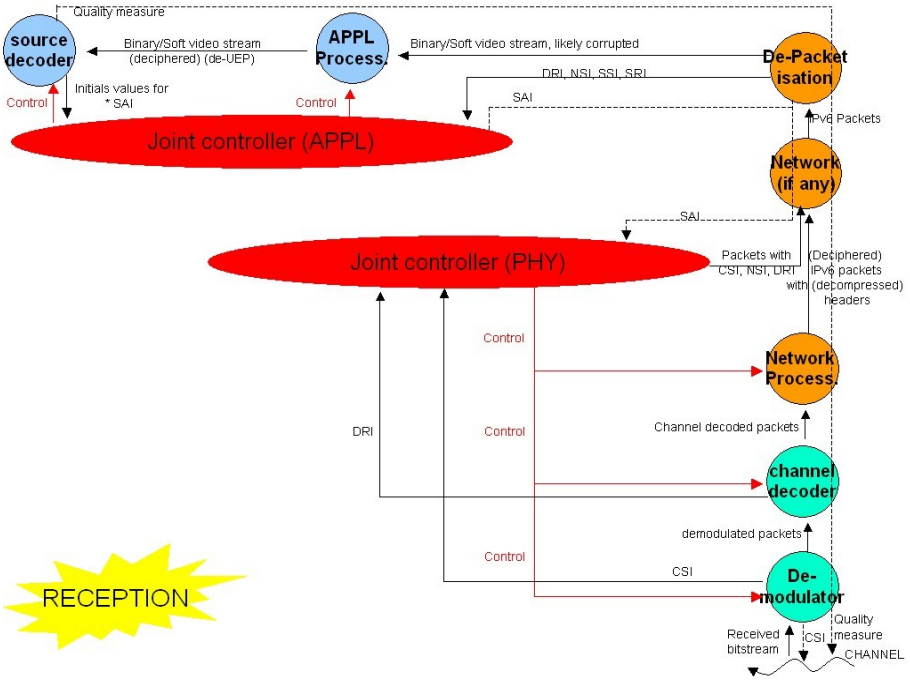

Figure 3.2 : General simulation chain: reception side.

3.2.- Input and Output Interfaces

At the **source node** the input interface (directly derived from the video data file) is stored in a file with readable information.

The format of the lines in the interface files is given as:

[transmission time][PACKET#][packet#][SIZE][size#][SSI][SRI][SRI][DRI][DRI]...[PAYLOAD][payload][newline]

hereafter an example is reported:

12[PACKET#]0001[SIZE]38[PAYLOAD]%^m??$_$Wwd4G~(=/%!+"">#&@{}<f??]?m??

000253[PACKET#]2[SIZE]3[SSI]1,9[SSI]3,15[PAYLOAD]...

000254 [PACKET#] 3 [SIZE]0[PAYLOAD]

765[PACKET#]0001[SIZE]3[DRI]%^m??$_$W[PAYLOAD]f]?

While, **at the radio receiver(s**) the input interface is stored in a binary with 0 corresponding to no error, and 1 to an error. This type of file is deployed in order to interface the architectural level simulator (OPNET), with the lower levels (Data-link and Physical layers) simulators.

For what concerns the output interfaces, the format is analogous to the input one specified at the source node, providing essentially the same information at IP level, either to the radio transmitter()s or to the destination node(s).

3.3.- Stima of Video Quality

The main problem of the application controller, that is implemented in the Basic chain and described above, is the use of PSNR as a metric for evaluating the video quality.

PSNR is a metric that quantify how differs the encoded video from the original one considering each frame. The following formula gives the PSNR value:

$$PSNR = \log_{10}\frac{255}{\sqrt{MSE}} \quad \text{and} \quad MSE = \frac{\sum_{i,j=0,0}^{I,J} f(i,j) - F(i,j)^2}{I * J}$$

where I and J are the size of the frame and $f(i,j)$ e $F(i,j)$ are the pixel of the reconstructed and original frame respectively. In a video sequence PSNR is calculated as an average of the PSNRs evaluated for each video frame.

PSNR has usually values between 20 and 40 and is used to evaluate how good an encoding algorithm is, how noisy a radio channel is or how full a packet network is, because higher the PSNR, higher the similarity between the original and the transmitted video. Furthermore this metric is related with the quality perceived by the user, expressed by the Mean Opinion Score (MOS) and reported in the Table 0-1.

PSNR	MOS	Quality
>37	5	Very good
31-37	4	Good

25-31	3	Medium
20-25	2	Low
<20	1	Very low

Table 0 1 - Relation between PSNR and MOS.

However, PSNR has two main problems:

- It requires a high computational overhead in the case of a video sequence and can only be used in simulation or in off-line evaluations, but not for real time adaptations.
- In a real transmission system the original video isn't available at the receiver, or, vice versa, the transmitted video isn't available at the transmitter. This means that the PSNR cannot be calculated.

PROPOSED PSNR ESTIMATION

In order to properly evaluate the video quality, there are essentially two patterns that can be followed. The first one is to create a new metric using the following parameters :

- coding type and parameters.
- statistics on the errors introduced by the radio channel. ([7],[8]).

and the second is to estimate by the same data, an existing and well-known metric, i.e. the PSNR.

Using the basic simulation chain, and setting fixed source encoding parameters, we collect a lot of data regarding a single second of video (a time slot of the application controller):

- PSNR evaluation by definition
- Bit Error Rate (BER) on the radio link
- Packet Error Rate (PER) on the radio link
- distribution of errors in the header and payload
- packet loss rate in the overall IP network

On the basis of this data, we were able to design an estimation model that can evaluate the PSNR.

To validate the model, we evaluated the variance of the estimation error, which is a measurement of the difference between the estimate value and the real one, for each set of parameters.

At first, using the Kolmogorov-Wiener estimation theory [9], we found out that the process was very dynamic, and it was not useful to take into account of the PSNR value estimated in the previous time slot. Therefore, our estimation model uses only the data concerning the last transmitted video fragment.

With the different type of data, we calculated the correlation between them and the real PSNR excluding each one at one time, in order to establish a hierarchy of importance. As a result of a first analysis, we

decided not to use the packet loss rate: its contribution to the correlation was very small, because the impact on the video quality strongly depends on the lost packet content. However, we opted to still use this kind of information directly into the application controller algorithm, and employing the PSNR estimation only when there isn't congestion in the IP network. On the other hand the two error rates were the parameters with the highest correlation: BER is a global measurement of the errors, and the PER is an effective way to classify the distribution of errors among different packets.

The first model investigated was a linear interpolation calculated with the least squares method. Even if the obtained results were fairly good, considering the real value of PSNR, it must be noted that there is an exponential relationship between error and PSNR, therefore we have built a very simple exponential model that uses only the BER and the PER as exponent of two parameters evaluated also in this case with the least squares method. The resulting formula is:

$$PSNR = CoeffBER^{AvgBER} * CoeffPER^{PER} * Cost$$

The *CoeffBER, CoeffPER* and *Cost* are multiplicative coefficients that have been estimated for different encoding settings, i.e. different application controller states. The *AvgBER* is the average of the Bit Error Rate and *PER* is the Packet error Rate. The Table 3.1. lists the values of the multiplicative coefficients used in the formula for each possible encoding setting and shows the corresponding variance of the estimation error.

This second model has been adopted, because its variance of the estimation error is lower than in the first case. Although it entails a higher computational overhead, no particular constraints arise from this point of view, because the PSNR estimation is made with a low frequency (every time slot, e.g. every 1 sec.).

To validate the proposed model we ran again simulations by the Basic Simulation Chain with several coder settings and different random seeds. Then we determined the PSNR in such conditions, and calculated the variance of the estimation error. The resulting values are very close to each other (Figure 3.3), confirming that the designed model has a general validity at least for MPEG based encoding.

Coding state	Coeff PER	Coeff BER	Cost	Variance
A	0,756382501	0,045728972	30,51411804	6,152992409
B	0,536339427	0,152505049	29,2249529	9,352558685
C	0,483247178	0,310011959	28,9191338	12,98070871
D	0,491059349	0,269933645	30,30351096	8,398609429
E	0,4570908	0,471651664	30,0691574	16,14259077
F	0,429847492	0,565076836	29,38965322	12,10636812
G	0,515808963	0,214080168	30,62655963	12,55157101

Table 3.1 : Values of the multiplicative coefficients and variance of estimation error for each possible encoding settings.

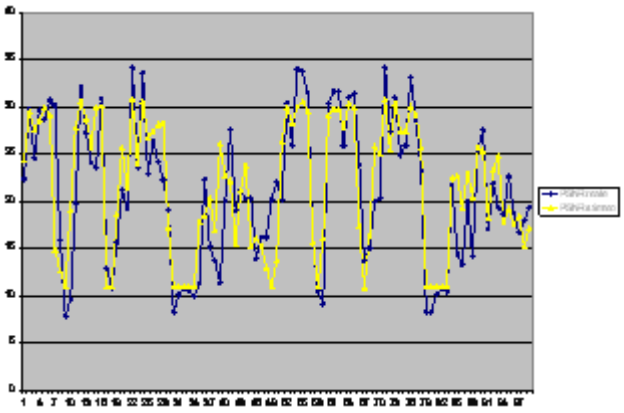

Figure 3.3 : Real and estimated PSNR.

3.4.- Network Transparency Support

We will see now different possible methods to support the Network Transparency. Each method is more or less appropriate with respect to a specific type of information that can be transported either by an in-band or an out-of band signalling. Once define the nature and size of the concerned information one or more mechanisms can be adopted and analysed in order to highlight the related pros and cons. For example, as a rule of thumb, control/signalling information that is strictly coupled with the multimedia data is more likely to be in-band delivered, while the others by an out-of-band protocol; furthermore, ad-hoc or extensions to already existing standard protocols are to be employed to provide communication between different nodes, while inter-layer signalling should require the development of new APIs.

3.4.1.-IPv6 Extension Headers

IPV6 [10] is characterized by an obligatory header (IPv6 base header) and some optional extension headers as Figure 3.4. shows. Nowadays there are six optional possible headers, two of which can be used to exchange control/signaling information according to Phoenix demands: hop-by-hop option and destination option.

Figure 3.4. : IPv6 Datagram

Hop-by-hop option is used for data that must be controlled by all traversing nodes along its path. Nevertheless, if we use *destination option* the system can transport adding information that is only analyzed in destination node. This particularity that made them different is useful for some types of signal.

3.4.2.-ICMPv6

ICMPv6 (Internet Control Message Protocol v6) [11] is a simple protocol that relays directly over IPv6. It is part and parcel of IPv6 and so it must be supported all of it in all the nodes.

IPv6 nodes usually use ICMPv6 to highlight errors that have been encountered processing the packet and to implement other functionalities of IP level. ICMPv6 messages are classified in two types:

- Error messages
- Information messages.

The first ones are used to one-way communication, usually for error notification while the second ones, also

named query/replay messages, are utilized to request or receive information

The Figure 3.5 shows a generic ICMPv6 message.

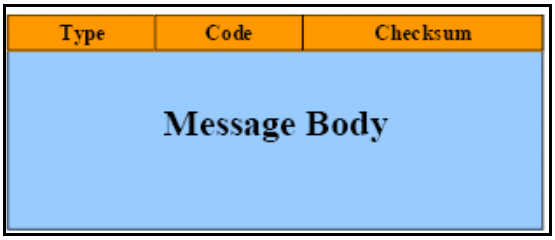

Figure 3.5 : ICMPv6 message format.

In Phoenix system, ICMPv6 can be exploited to transfer control and signalling signals between JSSC/D entities, defining new types of messages based on this protocol.

3.4.3.-Signalling protocols ad-hoc

This approach (problem) is based on already existing signaling/control protocols, conceived to send generic information and destined in this case to transport information to JSSC/D entity. Structure and transport characteristics are already defined while the higher level must be implemented in a generic manner inserting an adapted level.

As an example, a protocol with this characteristics that can be used in Phoenix is “real-time-transfer-protocol”/ “ real-time-transfer-control-protocol” (RTP/RTCP)[12], that includes two well defined protocol for real-time transport (RTP) and control and signaling information (RTCP).

Figures 3.6. and 3.7 show packet formats

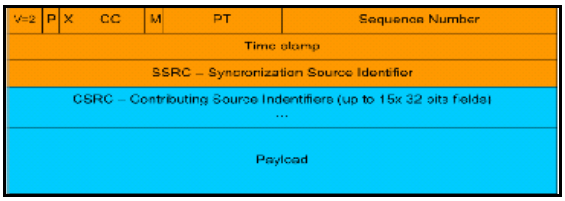

Figure 3.6 : RTP packet format

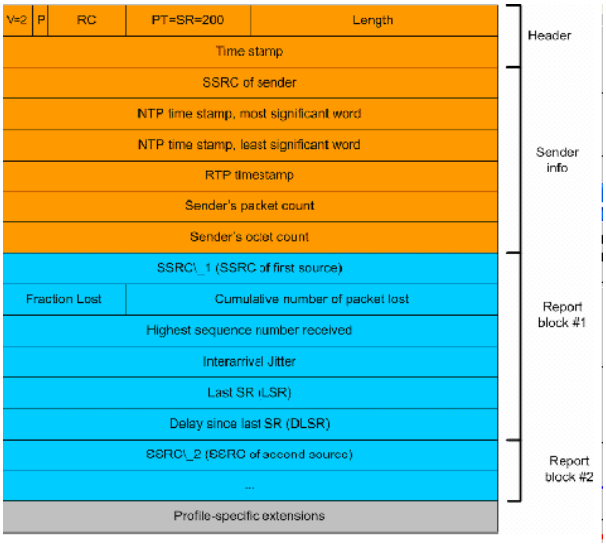

Figure 3.7 : RTCP packet Format.

3.4.4.- Socket-to-Socket direct Communication

Socket-to-Socket communications are end-to-end communications in which the packets, at operating system level, are reserved for specified protocol and uses. This method can be considered as a possible solution to transfer part of JSCC/D signaling. This solution, direct communication socket-to-socket, does not need modifications of Internet protocols nor the definition of new options, but it needs opening new additional sockets to route control information. Management and programming this occurs at application level and Network control is assigned to operating system.

3.4.5.-Proposed mechanism for each type of information

After the simulation made in this Project at CEFRIEL center, it has been specified the best mechanism for each signal to transport the information, according to the exigencies of control or information signals and to the overhead that is produced by them.

3.4.5.1.- SSI and SRI

These signals are quite similar to each other. Therefore they are considered and studied together. Both signals must be sent synchronized with the video frame. Therefore the most suitable mechanisms for sending this information are those which encapsulate it inside the frame that contains the video information. As SSI and SRI information are used in different points of the Network (particularly in CHANNEL CODER/DECODER and Destination node), the most suitable header is the **HOP_BY_HOP header**. The added overhead is nearly related to video code (medium rate) in the sense that each packet is joint to this information.

3.4.5.2.- CSI

CSI information is generated each period of a set TIMER inside the system, and logically is sent periodically. There are no special requirements to the mechanism of sending for this type of information, this is not necessary to search a synchronized mechanism. The mechanisms though for CSI are: ICMPv6 and IP

ad-hoc. There is no packet flow in which encapsulate the information in the sense that the CSI flow goes in the other direction in respect to the video flow. Utilizing **ICMPv6** packets we obtain a good result of overhead.

3.4.5.3.- NSI

Also, NSI information comes generated and sent periodically, based on a timer. Therefore**, ICMPv6** packets are without a doubt a good mechanism of sending. If temporal requirements are higher, we could thing about RTP/RTCP protocol that guarantees QoS inside the IP Network. But the problem to be in account is that RTP/RTCP protocol introduces a high overhead, higher that this introduced by ICMPv6.

3.4.5.4.- SAI

This type of information is generated by the video Decoder in Destination Node, and is analyzed in RX Wireless node. In spite of it goes in the opposite sense of the video flow, it is nearly related to the received and decoded frames. Sending Frequency is quite the same as received flow and therefore is desirable generating a new IP flow from the Destination Node to the RX wireless node. Also it can be utilized an **ICMPv6** flow or to encapsulate the information with an **IP extension hop-by-hop** in case an opposite video flow exits. But there would be great fragmentation problems cause the size usually exceed the MTU size scheduled for the extension header.

3.4.5.5.- DRI

DRI information is related to “fuzzy” decoding or “soft” type decoding and contains information about the certain value that the Channel Decoder has attached to a particular bit. As it is, the information is of completion or of replacement. This flow must be logically matched to the video frame that is being transmitted. Due to the fact that the dimension is much more higher, it has been though about the DRI information to be inserted in an **ICMPv6** flow, in which we will insert also the timestamp or the frame that DRI refers to. For what we see, the impact over the system features between RX wireless and Destination Node must be analyzed because of the information size.

3.5.- The Simulation in PHOENIX Project

The aim of the simulations related to the system proposed in Phoenix is to support with a quantitative analysis the validity and the full operation of the proposed solution and the improvements that can be obtain with a JSCC/D system in a 4G Network. When possible, new proposals are searched in order to improve the present architecture implementation and the policies to possible future improvements. The simulation analysis of several modules let provide us clearly the requirements of each module, both functional and time ones that are admitted inside the system. To reach these objectives is needed to simulate the operation of all the system, considering all the aspects that are going to form it. From MPEG coding to Network Transparency aspects, from secure aspects to IP Network bottleneck. Consequently all the modules that form the Project must be modeled and simulated joint to each other with the aim of aggregate them in an only distributed system. This distributed system must operate in a converged manner and must simulate the transferring of multimedia video flow between both end systems. Due to the fact that the dimensions of the work place turn hard to manage, we have to suppose to eliminate some of the choices that we know that are not going to give us good results, remaining in the work space those that can give us good results.

When simulating, we can choose implementation configurations and choose the level of detail without increasing a lot the global simulation time. There are lot of objectives in this project and above all the types of results that it can be obtained are heterogeneous because of the largeness of the Project. Because of that, two types of simulations have been chosen, with different modalities, purposes and performances in order to be able to observe different main aspects. On one hand, a serial of modules in a "Closed Chain" has been created. This Closed Chain simulates the actual packet transferring from the source to the destination. It has been utilized a C++ or C software and it has been compiled in a Linux Platform. This environment, called "Basic Simulation Chain" has been utilized as a Source of Input dates for the other type of simulation that has been carried out. On the other hand, as we have just said, there is another type of simulation, made in Network environment: OPNET Modeler (We have centered our efforts in this type of simulation) . The basic differences between both types of simulation are shown in the following table.

	"BASIC SIMULATION CHAIN"	**"OPNET MODELER"**
INPUT	1.- Parameters that characterize the Network 2.- Random seed 3.- Video File to transmit	1.- Group of Files to transmit and its characteristics
SIMULATION	All the dates that must be sent to the following module are elaborated in each module.	The packets are sent without payload and in each module information is collected.
RESULTS	1.- Decoded video 2.- Information from several modules	1.- Statistics over errors, delays, losses, traffic…

Table 3.2 : Main differences between both types of simulation.

3.6.- Application Controller

APPLICATION CONTROLLER ALGORITHM

Using the scheme proposed for the Basic Simulation Chain, we tried to build a more flexible and complete application controller that can work in real time, using the data actually transmitted through the network.

The controller collects all the feedback information from the overall network and from the radio link, after a fixed time slot, evaluates the PSNR using the model proposed, and then decides the encoding parameters to be used for the next transmission slot. The choice of the encoding parameters is made among a fixed set of possible settings, as shown in the table below.

State	QI, QP	GOV	Frame rate	Source bit rate
A	(14,16)	8	7.5	189 kbps
B	(14,16)	15	15	198 kbps
C	(14,16)	30	30	231 kbps
D	(14,16)	15	30	271 kbps
E	(8,12)	15	15	269 kbps
F	(8,12)	30	30	318.44 kbps
G	(8,12)	15	30	382.68 kbps

Table 3.3 : Application Controller states in terms of QI, QP, GOV, Frame Rate and Source bit rate.

GOV: (Group of Video Objects) That is, the size of the group of frames that will be encoded. It is composed by **I** frames and **P** frames.

QI,**QP**: Quantization parameters for the different types of frame.

The algorithm is defined by a simple state machine (see Fig. 3.8), essentially based on the piece of code below (Table 3.4).

The Application Controller has five states, but the possible sets of parameters are seven (A to G, see Table X), because the states 3 and 5 have two different options for the GOV length.

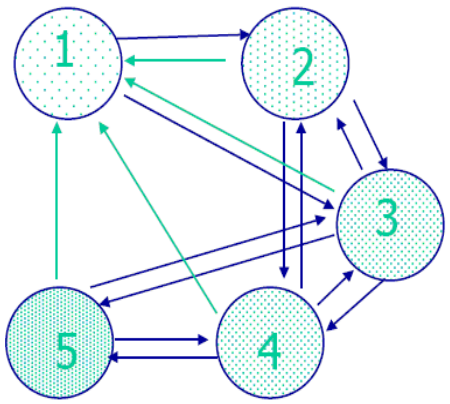

Figure 3.8 : Application controller state machine

Each state represents an encoding rate and a transition happens when the video quality or the channel state or the network state change. The idea is that if the channel is noisy or the network is congested, the encoding rate will be decreased in order to improve the perceived quality, the same applies when the PSNR is decreasing.

```
APP CONTROLLER ALG:

if PLR > PLR threshold

state=1

else

STATE SELECTION;

End APP CONTROLLER ALG



STATE SELECTION:

If PSNRi<PSNRi-1 (PSNR is decreasing)

// move towards more robust states

        if 25 dB<PSNRi<29 dB

        STATE=STATE - 1

       else if PSNRi <25 dB

       STATE=STATE - 2



If PSNRi>PSNRi-1 (PSNR is increasing)

// move towards less robust states

if PSNRi>31 dB

STATE=STATE + 2

else if 28 dB<PSNRi<31 dB

STATE=STATE + 1

end STATE SELECTION
```

Table 3.4 : The pseudo-code describing the application controller algorithm.

3.7.- Improvements with Application Controller

In order to validate the performance of the proposed application controller algorithm with our novel estimation model we have built up a complex simulation scenario employing Opnet Modeler. The purpose of the analysis is to measure the benefit of the application controller in the PHOENIX JSCC/D system with an estimation of the PSNR as the indicator of the received video quality.

The Basic Simulation Chain has been employed to generate bit error patterns used to feed the simulator and emulate the behaviour of a radio channel in different condition (i.e. different SNR value).

The most interesting scenario reproduced a real channel in time-variant conditions: 0-20s good, 20-30s fair, 30-50s: good, 50-60s very poor. The different conditions were obtained using the Simulation Chain with 4 different SNR values: 8 dB for good quality, 4 dB for fair quality, 2 dB for poor quality and 1 dB for very poor quality. In Fig. 3.9. shows the Bit Error Rate during the 60 seconds of simulation.

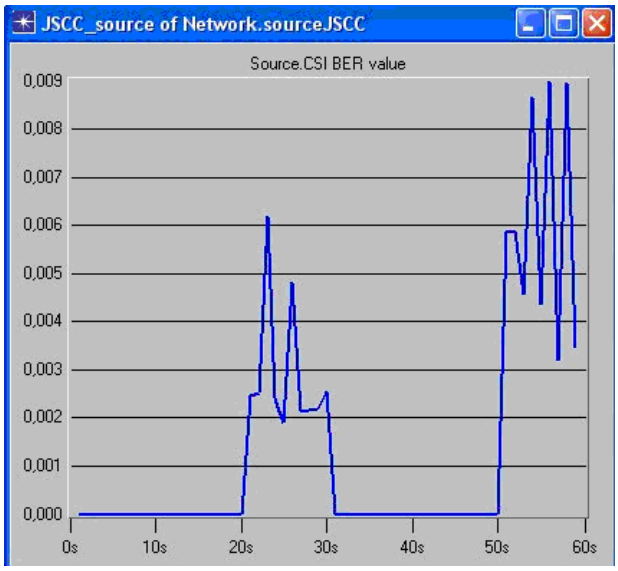

Figure 3.9 : Bit Error Rate

The application controller reacts to the change of the channel condition moving to a more efficient encoding scheme every time slot. Fig.3.10 demonstrates that when the channel quality is degrading, is more convenient to use different MPEG encoding settings. However, when the channel is good, it is possible to encode with a higher bit rate. In poor channel condition, without the application controller the PSNR of the received video is very low.

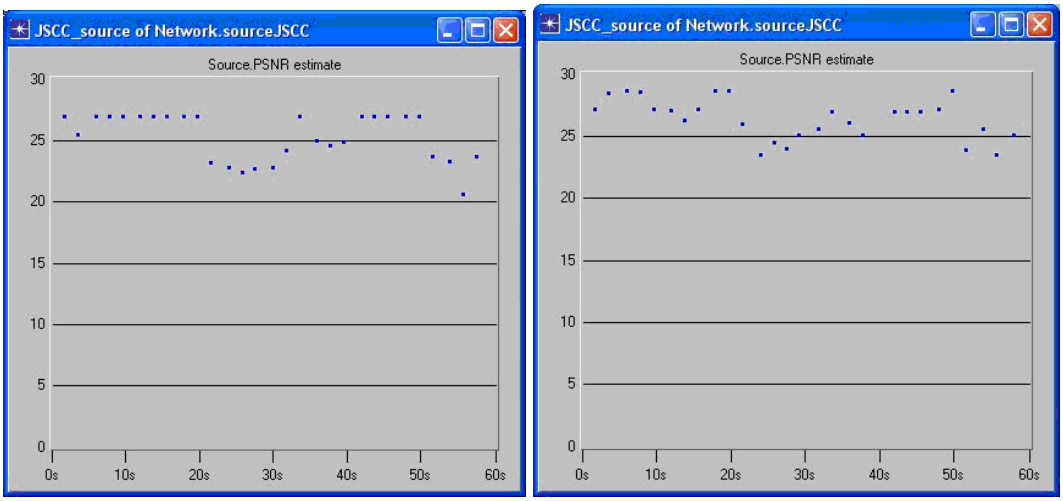

Figure 3.10 : a) PSNR without A.C. b) PSNR with A.C