ANEX

1.1.- CODING SCHEMES

MPEG-4

The MPEG-4 standard (ISO/IEC 14496) has been published in 1998. It is designed to address the demands of highly interactive multimedia applications as well as traditional applications. MPEG-4 standard consists of tools for coding natural and synthetic video and audio and text. In addition, MPEG-4 standard includes a system part for synchronization of these streams. The MPEG-4 standard includes advanced functionality, for instance objectbased coding, interactivity, scalability and error-resilience. The MPEG-4 has adopted five error-resilience tools that provide some error robustness. These tools are video packet resynchronization, data partitioning, reversible variable length coding (RVLC), header extension code (HEC) and new prediction schemes. At the video decoder, these tools can be used to detect and conceal transmission errors.

MPEG-4 (Visual standard) offers three different techniques for scalable video coding: temporal scalability, spatial scalability and SNR fine granular scalability (FGS). Furthermore, the object-based scalability comes automatically with the object-based composition. These scalability schemes can also be combined to reach a finer scalability like space-temporal scalability. MPEG-4 (together with H.263) has been selected as a multimedia standard for 3rd Generation Partnership Project (3GPP).

MPEG-4 FGS

Fine granularity scalability is a coding method by which the rate as well as the quality increment changes in smaller steps. A major difference between FGS technique and other scalable and non-scalable video coding techniques is that FGS technique separates encoding from the distribution process. FGS coded video includes two layers, a base layer and enhancement layer.

- Base layer is produced to provide a low but guaranteed level of quality and a normal encoding process can be used to achieve this layer.
- The enhancement layer provides improvement to quality at rather small steps.

The reconstruction error of the base layer is encoded in the enhancement layer using a bitplane representation of the DCT coefficients. The most significant bits are included into the bitstream for all macroblocks, followed by the second most significant bitplanes and so on. Because bitplanes are ordered like this, it is possible to stop the transmission of the enhancement layer data at any point, while being able to make use of all transmitted data up to that point. Fine granularity scalability video coding technique is included in the MPEG-4 standard. The coding efficiency of the FGS method is significantly lower than efficiency in non-scalable methods. The MPEG-4 FGS could be used as a reference for adaptive video coding technique in the PHOENIX project.

H.264 / MPEG-4 AVC

Studies on H.264 began within the Video Coding Expert Group (VCEG) of ITU-T in 1999. The objectives of this standard were:

- To provide efficient compression (a reduction of about 50% for the average bitrate at equivalent visual quality when compared to the other existing standards),

- To present a reasonable ratio between complexity and coding efficiency,
- To be easily adaptable to networked applications, and in particular wireless networks and internet (hence over IP)
- To adopt a syntax easy to use, with a reduced number of profiles and options.

H.264 main target applications are:
- Duplex real-time voice services (*e.g.* visiophony) over wired or wireless (such as UMTS) networks (bitrate below 1 Mb/s and low latency),
- Good or high quality video services for streaming over satellite, xDSL, or DVD (bitrate from 1 to 8 Mb/s and possibly large latency),
- Lower quality streaming for video services with lower bitrate such as over Internet (bitrate below 2 Mb/s and possibly large latency).

H.264 specifies only the video coding aspects, while the transport problematic is dealt by MPEG-4 system specifications. Still, as illustrated in the next figure , H.264 includes above its coding layer a flexible adaptation layer that allow it to be compatible with transport technologies of both wireless and wired worlds:

- For phone networks through H.324 (circuit mode) or H.320 (fixed networks);
- IP world through RTP/UDP/IP or TCP/IP stacks.

This makes of H.264 a standard which is compatible with both mobile and fixed solutions, and can hence be a way to help the already beginning unification of those two worlds.

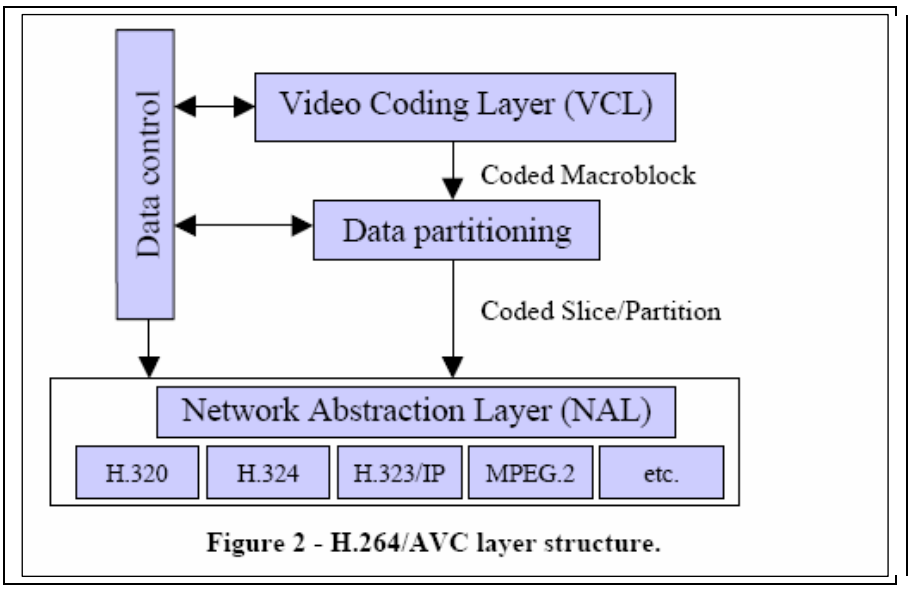
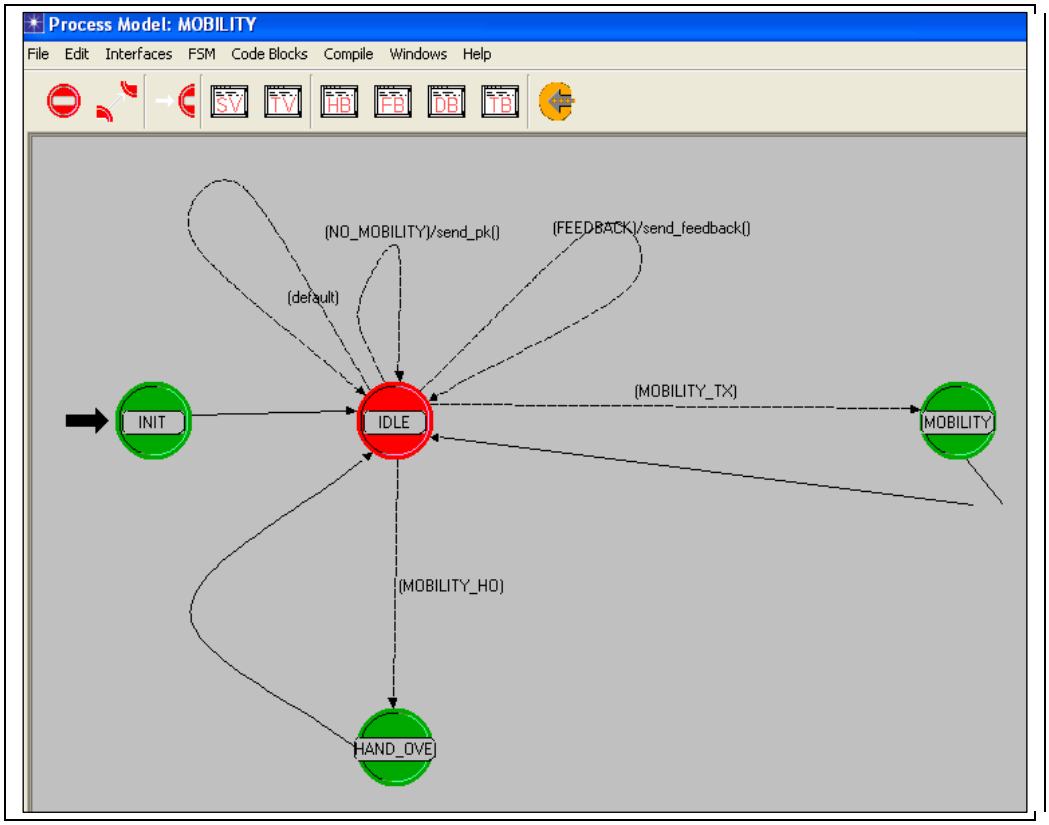


Figure 2 - H.264/AVC layer structure.

It is to be noted that while being now part of MPEG-4 standard, H.264 is not directly compatible with the other MPEG-4 video parts, as the compression tools greatly differ. While not being completely different from the other existing video standards, the video coding layer of H.264 is characterized by the following new key aspects:
- Better motion compensation,
- Smaller blocks in the coding transform,
- A new coding transform (inversible),
- Improved deblocking filter
- More efficient entropy coding

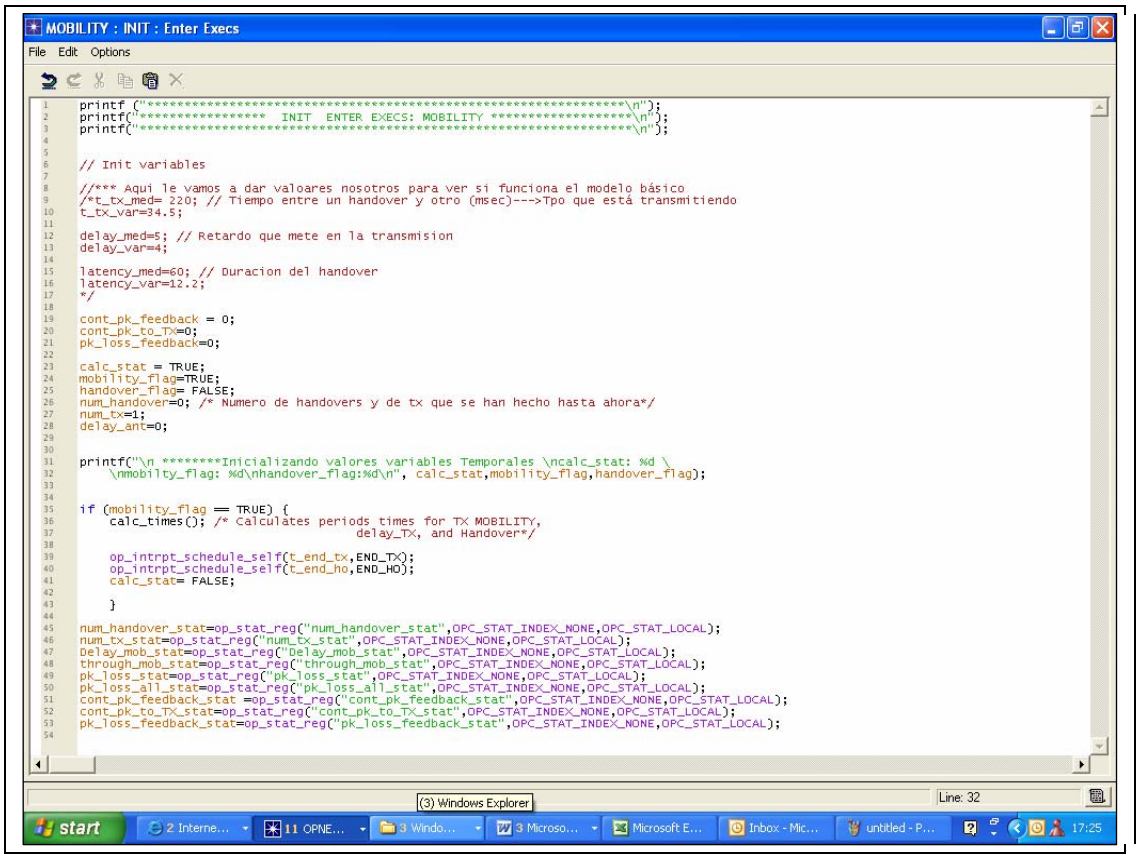
1.2.- MOBILITY MODULE

1.2.1.- Process Model

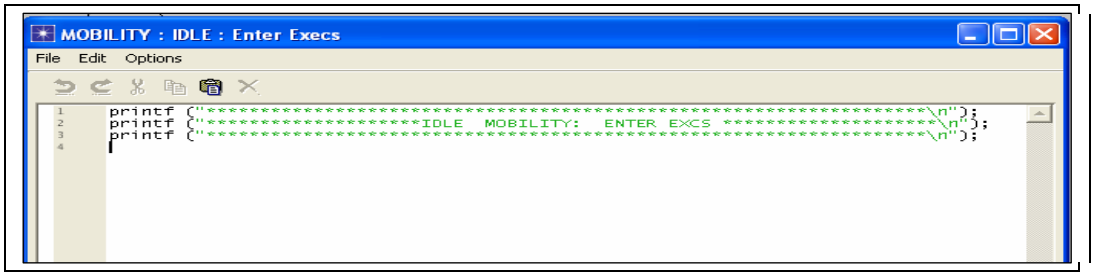


1.2.2.- C Code

INIT:



IDLE:



MOBILITY:

```
printf ("*****************************************************************************\n");
printf ("*****************TX MOBILITY: Enter EXECS  ***************************\n");
printf ("*****************************************************************************\n");


// Si es una interrupcion programada de END_HO

if ((op_intrpt_type() == OPC_INTRPT_SELF) && (op_intrpt_code() ==END_HO)&& (handover_flag==TRUE)){

    op_prg_odb_bkpt("con_MOBILITY");

    calc_times();
    handover_flag=FALSE;
    op_intrpt_schedule_self(t_end_tx,END_TX);
    num_tx++;
    delay_ant+=delay_tx;
    op_stat_write(Delay_mob_stat, delay_ant/num_tx);
    op_stat_write(num_tx_stat,num_tx);
    printf("Number of tx until now : %d\n",num_tx);
    printf(" Programamos la interrupcion para el periodo de HO \n");
    printf("El valor de handover_flag es : %d\n",handover_flag);


    }

if((op_intrpt_type() == OPC_INTRPT_STRM)&&(handover_flag== FALSE)){
// Si es una interrupcion de llegada de paquete

    op_prg_odb_bkpt("con_MOBILITY");
    rcv_pck_ptr_mob = op_pk_get(op_intrpt_strm());
    //rcv_pck_ptr_mob = op_pk_get(STREAM_FROM_IP);/* We recive  the packet*/

    if(op_intrpt_strm()==STREAM_TX_RX_a){
        printf("\n\nReceiving packet from the IP interface. We are in mobility module introducing delay\n\n\n");

        // Vamos a probar a enviarlo aqui
        op_pk_stamp(rcv_pck_ptr_mob);
        op_pk_print(rcv_pck_ptr_mob);
        op_pk_send_delayed(rcv_pck_ptr_mob,STREAM_TX_RX_b,delay_tx);


        cont_pk_to_TX++;
        op_stat_write(cont_pk_to_TX_stat, cont_pk_to_TX);

        op_stat_write(pk_loss_all_stat, pk_loss_all);
        printf("\n\n *****Num de paquetes que se envian con mobilidad al TX: %d*******\n\n",cont_pk_to_TX);
        printf("\n\n ****  PAquete enviado al modulo TX *************** \n\n");
        printf("\n\nSimulation time while sending packet with MOBILITY := \t %.16f\n",(double)op_sim_time()+delay_tx);

        }
    else{ // PAquete recibido del modulo TX
        op_prg_odb_bkpt("nodo_from_TX");
```

```
        printf("\n\nReceiving packet from the TX interface. We are in mobility module introducing delay\n\n\n");

        // Vamos a probar a enviarlo aqui
        op_pk_stamp(rcv_pck_ptr_mob);
        op_pk_print(rcv_pck_ptr_mob);
        op_pk_send_delayed(rcv_pck_ptr_mob,STREAM_TX_RX_a,delay_tx);


        //op_pk_deliver_delayed (rcv_pck_ptr, mod_objid, instrm_index, delay)

        cont_pk_feedback++;
        op_stat_write(cont_pk_feedback_stat, cont_pk_feedback);

        op_stat_write(pk_loss_all_stat, pk_loss_all);
        printf("\n\n *****Num de paquetes que se envian con mobilidad a la IP: %d*******\n\n",cont_pk_feedback);
        printf("\n\n ****  PAquete enviado al modulo Network IP *************** \n\n");

        printf("\n\nSimulation time while sending packet with MOBILITY := \t %.16f\n",(double)op_sim_time()+delay_tx);

        }
    }
```

HANDOVER:

```
printf ("*************************************************************************\n");
printf ("******************HANDOVER: EnTER EXECS    ***************************\n");
printf ("*************************************************************************\n");


/* Aqui no se hace nada , solo DESTRUIR  paquetes y recoger estadisticas de
paquetes destruidos durante este periodo, y los destruidos hasta el momento de
ejecución que estamos*/


//Si es una interrupcion programada de END_TX
if((op_intrpt_type ()==OPC_INTRPT_SELF)&&(op_intrpt_code()==END_TX)&&(handover_flag==FALSE)){
    op_prg_odb_bkpt("HO");
    handover_flag= TRUE;
    op_intrpt_schedule_self(t_end_ho,END_HO);
    num_handover++;
    op_stat_write(num_handover_stat,num_handover);

    printf("Number of HANDOVER until now : %d\n",num_handover);
    }




//Si es  una interrupcion normalde llegada de paquete
if ((op_intrpt_type()== OPC_INTRPT_STRM)&&(handover_flag== TRUE)){
    op_prg_odb_bkpt("HO");
    rcv_pck_ptr_mob = op_pk_get(op_intrpt_strm());


    if(op_intrpt_strm()==STREAM_TX_RX_b){
        cont_pk_feedback++;
        pk_loss_feedback++;
        op_stat_write(cont_pk_feedback_stat, cont_pk_feedback);
        op_stat_write(pk_loss_feedback_stat, pk_loss_feedback);
        }

    op_pk_destroy(rcv_pck_ptr_mob);
    pk_loss++;
    pk_loss_all++;
    op_stat_write(pk_loss_stat,pk_loss);

    printf(" PAquetes perdidos en HANDOVER en este periodo:\n\
        Coming from Feedback = % d\n\
        Coming from Source and Feedback packets = %d\n ", pk_loss_feedback,pk_loss);

    printf(" PAquetes perdidos en HANDOVER en total       :%d\n", pk_loss_all);

    printf("\n\n ****** DESTRUIMOS PAQUETE DE HANDOVER *********\n\n");


\
```

HEADER_BLOCK:

```
/*** Include Library ***/
/*** Libraries including declarations ***/

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>
#include <io.h>
#include <time.h>
#include <sys\stat.h>
#include <math.h>
#include <sys/types.h>
#include <sys/stat.h>


// A) TRansition Conditions without applicating MOBILITY to Feedback Information

#define MOBILITY_TX  (((op_intrpt_type() == OPC_INTRPT_STRM)&&(mobility_flag==TRUE)&&(handover_flag== FALSE)&&\
    (op_intrpt_strm ()== STREAM_TX_RX_a))||((op_intrpt_type() ==OPC_INTRPT_SELF)&&(op_intrpt_code()== END_HO)))

#define MOBILITY_HO  (((op_intrpt_type() == OPC_INTRPT_STRM)&&(mobility_flag==TRUE)&&(handover_flag==TRUE)&&\
    (op_intrpt_strm () == STREAM_TX_RX_a))||((op_intrpt_type() ==OPC_INTRPT_SELF)&&(op_intrpt_code()== END_TX)))

#define NO_MOBILITY  ((op_intrpt_type() == OPC_INTRPT_STRM)&&(mobility_flag==FALSE)&&(op_intrpt_strm () == STREAM_TX_RX_a))
#define FEEDBACK   ((op_intrpt_type() == OPC_INTRPT_STRM)&&(op_intrpt_strm ()==STREAM_TX_RX_b))


//B) Transition conditions when MOBILITY is applicated to Feedback Information
/*
#define MOBILITY_TX  (((op_intrpt_type() == OPC_INTRPT_STRM)&&(mobility_flag==TRUE)&&(handover_flag== FALSE))||\
((op_intrpt_type() ==OPC_INTRPT_SELF)&&(op_intrpt_code()== END_HO)))

#define MOBILITY_HO  (((op_intrpt_type() == OPC_INTRPT_STRM)&&(mobility_flag==TRUE)&&(handover_flag==TRUE))||\
((op_intrpt_type() ==OPC_INTRPT_SELF)&&(op_intrpt_code()== END_TX)))

#define NO_MOBILITY  ((op_intrpt_type() == OPC_INTRPT_STRM)&&(mobility_flag==FALSE))
#define FEEDBACK    0

*/


/* COnstants and flows definitions **/

#define STREAM_TX_RX_a                  0
#define STREAM_TX_RX_b                  1

#define TRUE                            1
#define FALSE                           0
#define END_TX                          0
#define END_HO                          1
#define TX_HO                           2
```

```
typedef struct {
    unsigned start;
    unsigned end;
    unsigned delay;
    struct transmission_t *next;
}transmission_t;

typedef struct {
    unsigned start;
    unsigned end;
    struct ho_t *next;
}ho_t;

typedef struct{
    transmission_t *transmission_list;
    ho_t *ho_list;
}exec_cfg_t;
```

FUNCTION BLOCK:

```
//*************************************************************************//
//**********************   FUNCTION  BLOCK  ***************************//
//*************************************************************************//


send_feedback(){
 Packet*  pk_recv;
 int stream;
 stream = op_intrpt_strm();
 pk_recv = op_pk_get(stream);// Lo recibimos por un stream

 if (stream != STREAM_TX_RX_b)
      op_prg_odb_bkpt("\n\n ******* ERROR en el stream de FEEDBACK *********\n\n");

 op_pk_send(pk_recv,STREAM_TX_RX_a);
 printf( " *************ENviando FEEDBACK a la fuente ---> sin Delay\n");
 cont_pk_feedback++;
 op_stat_write(cont_pk_feedback_stat,  cont_pk_feedback);


 }

//*************************************************************************

 send_pk(){
 Packet*  pk_recv;
 int stream;
 stream = op_intrpt_strm();
 pk_recv = op_pk_get(stream);// Lo recibimos por un stream
                                    // y lo enviamos por el otro y BASTA
 op_pk_print(pk_recv);

 op_prg_odb_bkpt("sin_MOBILiTY");

 if(stream ==STREAM_TX_RX_a){
    op_pk_send(pk_recv,STREAM_TX_RX_b);
     printf( " *************ENviando SIN MOBILITY al destino---> sin Delay\n");

    }
 else{
    op_prg_odb_bkpt("nodo_from_TX");
    op_pk_send(pk_recv,STREAM_TX_RX_a);
     printf( " *************ENviando SIN MOBILITY a la fuente ---> sin Delay\n");

    }
}
```

```
/**** Error Handling functions. ****/

static void
mgr_error (const char* msg0)
    {
    /** Print an error message and exit the simulation. **/
    FIN (mgr_error  (msg0));

    op_sim_end ("Error in JSCC source process:", msg0, OPC_NIL, OPC_NIL);

    FOUT;
    }


//*************************************************************************//

void calc_times(){

    /* CAda periodo la inicializamos a 0*/
    pk_loss=0;


    printf("***************  INICIALIZANDO   DATOS Periodo  ************\n\n\n");
    t_end_tx= op_sim_time()+ op_dist_exponential(220.0)/1000; // In msec
    delay_tx=op_dist_uniform(5.0)/1000; // In msec

    t_init_ho =t_end_tx;
    t_end_ho = t_init_ho + op_dist_uniform(60.0)/1000; // In msec

    printf("t_end_tx = %f \n",t_end_tx);
    printf("t_end_ho = %f \n",t_end_ho);

    printf( "\n\n**********FIN de Inicializar Datos Periodo ************\n\n\n\n");

}

//*************************************************************************//

int recv_stream (int intrpt_strm){

    Packet* rcv_pck_ptr;
    int stream= intrpt_strm;

    if (op_pk_get(stream)==OPC_NIL)
        return (0);
    else
        return (1);
    }
```

1.3.- BASIC SIMULATION CHAIN

PHOENIX project has created a simulation tool called "Basic Simulation Chain", which we have worked and simulated with at the same time as with OPNET Modeler. The goal of this "Basic Simulation Chain" is to represent the Transmission Network by setting several modules (this transmission chain is the same as the transmission chain in OPNET), inserting the hypotized optimisations and to coordinate the work by using the "JOINT CONTROLLERS".

The chosen implementation election has been to standardize the interface files that have information about the packets to transmit and use them as an input and an output of each module. The function of each module is to get an interface file as an

input, applying the transformations that must be simulated, to save the results in a second interface file and writing all the required information to the Controller in a "feedback" file, that will be read by the Controller to optimize the system. This mechanism has been introduce instead of the real message exchange between the different modules. So the Network Transparency has been simulated by this manner, generating files for the communication of information. It has been though to keep independent all the modules and to call them by a simple global execution script in which we can set the parameters. Although the initial idea was creating a code adapted to both Windows and Linux, at the end we have worked only with Linux because of its conveniences.

The chain is shown in the following graphic, showing as well the generated and sent information of the data video flow.

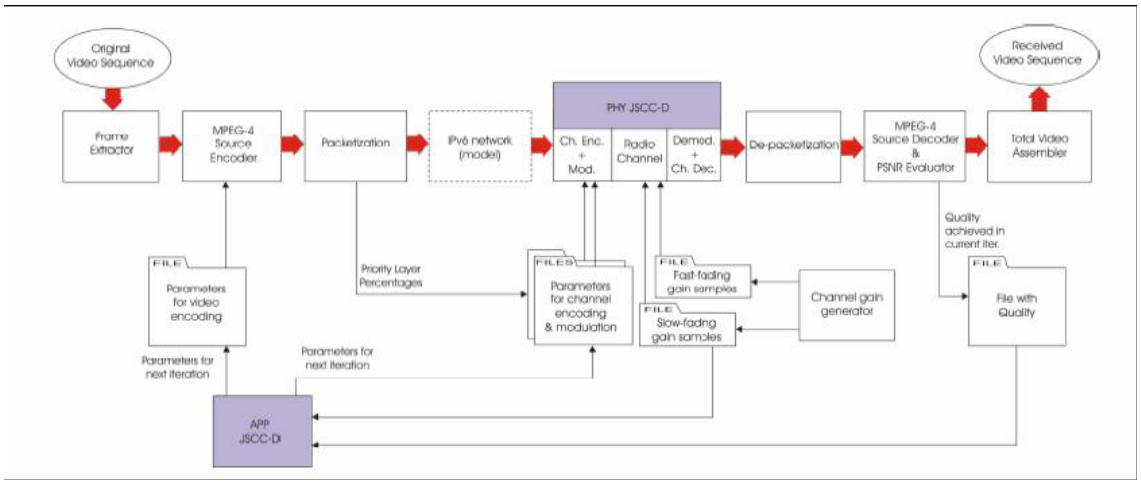


Figure 9: Basic Simulation Chain Structure

The Application Controller is invoked each cycle of transmission (1 sec), and it modifies the operation parameters. The obtained output is very attractive in the sense that the actual video quality at destination side can be observed. This is just the main objective if this Simulation Chain: Having in a reasonable short time, a system that can show tangible and observable results.

One of the disadvantages is the great amount of calculations at software level and hence the number of simulations come reduced because of this limitation. The primary aim of this transmission chain has obligated us to use a particular methodology of data transport by reading/writing files, once seen that a study of network mechanism to this information was no expected. This modules execution is in sequential manner and no concurrent as it is in the real system.

LIST OF ACRONYMS

4CIF	4 times Common Interchange Format (704x576 pixels)
BER	Bit Error Rate
CIF	Common Interchange Format (352x288 pixels)
CSI	Channel state information
ETSI	European Telecommunications Standards Institute
DRI	Decision reliability information
FEC	Forward Error Correction
FER	Frame Error Rate
fps	frames per second
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HIPERLAN	High Performance Radio LAN
ICMP	Internet Control Message Protocol
IP	Internet Protocol
IPR	Intellectual Property Rights
JSCC/D	Joint source and channel coding and decoding
MAC	Medium Access Control
MOS	Mean Opinion Score
NSI	Network State Information
NT	Network Transparency
OpNet	Optimized Network Performance
PDA	Personal Digital Assistant
PHY	PHYsical layer
PSNR	Peak signal to noise ratio
QCIF	Quarter Common Interchange Format (176x144 pixels)
QoS	Quality of Service
SAI	Source a-posteriori information
SSI	Source significance information
SRI	Source a-priori Information
SPI	Source a-posteriori information
RTCP	Real Time Control Protocol
RTP	Real Time Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UEP	Unequal Error Protection
UMTS	Universal Mobile Telecommunications System
UTRA	Universal Terrestrial Radio Access
Uu	Radio Interface for UTRA
WER	Word Error Rate
WLAN	Wireless Local Area Network
WP	Work-Package

BIBLIOGRAPHY

[1] M.G. Martini, *Wireless multimedia systems-joint source and channel coding for video transmission*. Ph D.thesis, University of Bologna , Italy, Jan 2002

[2] L. Perros Meilhac and C. Lamy, *Huffman tree based metric derivation for a low complexity sequential soft VLC decoding*, in Proceedings of IEEE ICC'02, New York, USA .vol.2, pp 783-787, April-May 2002.
[3] S. Blake et al., *" An architecture for differentiated services"*, RFC 2475, Dec. 1998
[4] M. Conti, G. Maselli, G. Turi and S. Giordano, *"Cross-Layering in Mobile Ad Hoc Network Design"*, IEEE Computer Magazine, Vol. 37, Issue 2, February 2004, pp. 48-51.
[5] L. Perros-Meilhac and C. Lamy, "*Huffman tree based metric derivation for a low-complexity sequential soft VLC decoding," Proceedings of ICC'02*, vol. 2, pp. 783-787, New York, USA, April-May 2002.
[6] C. Lamy and L. Perros-Meilhac, "Low complexity iterative decoding of variable-length codes," *Proceedings of the Picture Coding Symposium (PCS'03)*, pp. 275-280, April 23rd-25th 2003, St-Malo, France.
[7] "IEEE 802.11 – 1999 edition, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Standard 802.11, 1999.*
[8] H. Kaaranen, A. Ahtiainen, L. Laitinen, S. Naghian and V. Niemi, "UMTS Networks: Architecture, Mobility and Services," John Wiley and Sons, Chichester, England, 2001.
[9] Kolmogorov, A.N. 1941. Interpolation and extrapolation. Bulletin de l'Academie des Sciences de U.S.S.R., Series Mathematics 5: 3–14.
[10] A. Conta and S. Deering, **Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification**, *RFC 2463,December 1998.*
[11] Encoded sequences with SP and SI frames *http://ivms.stanford.edu/~esetton/sequences.htm*
[12] MPEG-4 Video Group, **Overview of the MPEG-4 Standard**, *ISO/IEC JTC1/SC29/WG11 N3444, Geneva, May-June 2000*
[13] OpNet Modeler by OPNET Technologies Inc. http://www.opnet.com