

Apéndice D. Código fuente de scripts bash y parámetros de codificación

Este apéndice corresponde al código fuente de los scripts bash que automatizan parte del *framework* propuesto para la evaluación de la transmisión de flujos de video codificados de acuerdo con el estándar HEVC/H.265. Igualmente, se presentan los comandos para la captura de trazas y Streaming de video, estando dentro de la emulación con Mininet. Al finalizar, se presentan los parámetros de codificación utilizados en el codificador HM (*HEVC test Model*).

1. Código Fuente script *Encoding.sh*

Este script permite la ejecución automática de las etapas 1, 2 y 3 del framework propuesto en este proyecto, estas etapas corresponden a: codificación, empaquetado y formato. Antes de iniciar la ejecución de este script, se debe tener correctamente instalado y configurado las herramientas HM, MP4BOX, ffmpeg, y el archivo “*encoding_config.cfg*” con los parámetros correctos del video y su codificación.

```
#!/bin/bash

#Este script corresponde a la etapa de codificación, empaquetado y formato.
#antes de iniciar su ejecucion, se debe configurar correctamente el archivo .cfg
#el cual contiene todos los parametros del video y los parametros de la codificacion.

echo " "
echo "Introduzca la tasa fps del video (Ej: 25,60,120)"
echo " "
read fpsrate #tasa fps del video
echo " "
#La codificacion HEVC se realiza con HM.
echo "Etapa de codificacion HEVC con HM"
echo " "
sleep 3
TAppEncoderStatic -c encoding_config.cfg
echo " "
echo "Empaquetando el video codificado en MP4"
echo " "
sleep 3
MP4Box -add outencoded.bin:fps=$fpsrate:FMT=HEVC -new outencoded_hevc.mp4
echo " "
echo "Convirtiendo a formato .ts (transport stream)"
echo " "
sleep 3
ffmpeg -i outencoded_hevc.mp4 -map 0:v -c copy forTX.ts
echo " "
echo "A continuacion se prepara el mininet iniciando los servidores de OVS"
```

```
echo "Esta operacion requiere log in como SUDO "  
echo " "  
sleep 1  
  
#el siguiente comando es para preparar la simulacion con mininet  
sudo /usr/local/share/openvswitch/scripts/ovs-ctl start
```

Una vez ejecutado el script mencionado anteriormente, se genera el video *forTX.ts* el cual corresponde al video a transmitir en la red SDN.

2. Comandos para la captura de trazas y Streaming de video

Una vez iniciada la red SDN en la herramienta MININET, se procede a realizar el streaming de video desde el host transmisor hacia el host receptor, capturando al mismo tiempo las trazas de envío y recepción. Para esto, se abren instancias *xterm* (terminal) de cada host, y se lanzan los siguientes comandos de la herramienta *TCPdump* para la captura de trazas y *FFMPEG* para realizar el streaming:

- En el host receptor (h2): *tcpdump -i h2-eth0 -nn -v -tt "src host 10.0.0.1 and dst host 10.0.0.2" > received*. Se habilita la captura tcpdump en host2, y se realiza filtro de contenido "*src host 10.0.0.1 and dst host 10.0.0.2*", la traza resultante es almacenada en el archivo *received*.
- En el host transmisor (h1): *tcpdump -i h1-eth0 -nn -v -tt "src host 10.0.0.1 and dst host 10.0.0.2" > sent*. Se habilita la captura tcpdump en host1, y se realiza filtro de contenido "*src host 10.0.0.1 and dst host 10.0.0.2*", la traza resultante es almacenada en el archivo *sent*.
- En el host receptor (h2): *ffmpeg -i rtp://10.0.0.2:5003 -c copy rec_rtp.ts* , donde *rec_rtp.ts* corresponde al video recibido en el extremo del receptor.

- En el host transmisor (h1): `ffmpeg -re -i forTX.ts -f rtp_mpegts rtp://10.0.0.2:5003`, donde *forTX.ts* es el video a transmitir, el formato del streaming es *rtp_mpegts*, el protocolo de transmisión es RTP, y la dirección y puerto destino corresponde al host *h2*.

3. Código fuente script QoS_Qvid.sh

Una vez se obtienen las trazas de envío y recepción como se indicó en el apartado anterior, se ejecuta el script *QoS_Qvid.sh* el cual permite procesar dichas trazas y obtener resultados de evaluación de calidad de servicio (QoS) como *packet delay*, *packet loss*, *thorughput*; y calidad del video recibido como PSNR y SSIM. Los resultados son mostrados en gráficas para un mejor análisis de los mismos.

```
#!/bin/bash

# Script para procesar trazas de TX y RX y obtener
# parámetros de QoS como: Packet Delay, Throughput y Packet Loss
# Trabaja con fecha tt en tcpdump

echo " "
echo "Introduzca nombre de video de entrada (incluya .yuv)"
echo " "
read yuv_in #corresponde al video original YUV
echo " "
echo "Introduzca tamaño en X"
echo " "
read size_x
echo " "
echo "Introduzca tamaño en Y"
echo " "
read size_y
echo " "
echo "Introduzca submuestreo (Ej: 420)"
echo " "
read subsampling
echo " "
echo "Introduzca nombre del video recibido (incluya .ts)"
echo " "
read vid_rec #corresponde al video recibido
echo " "
echo " "
echo "Obteniendo trazas..."
sleep 1
```

```

# se omiten las lineas ARP y obtenemos sólo IP con el Packet ID y
# su tiempo de envío:
grep -v ARP sent | grep IP | awk '{print $8,$1}' | tr "," " " > sent_A
echo "-"

# se obtiene traza de envío: orden de envío, Packet ID y tiempo de envío
nl sent_A | awk '{print $1, $2, $3}' > sent_1
sleep 1
echo "-"

# se obtiene traza de recepción: Packet ID, tiempo de recibo y tamaño de paquete.
grep -v ARP received | grep IP | awk '{print $8,$1,$17}' | tr "," " " | tr ")" " " > received_1
sleep 1
echo "-"

# script para obtener traza completa de RX y TX asi:
# orden envío, Pkt ID recibido, tiempo TX, tiempo RX, tamaño
perl trazas.pl sent_1 received_1 > TrazaTXRX
sleep 1
echo "-"

# se agrega linea para orden de RX y se calcula Pkt Delay
# el Pkt Delay está en orden de recibo:
echo "- Calculando Packet Delay."
nl TrazaTXRX | awk '{print $1, $5-$4}' > delay_rx
sleep 1
echo "-"

# Script para generar timestamp de cada pkt recibido desde Tinicial = 1er envío.
# resultado: tiempo de recibo y tamaño recibido.
perl GenTimeStamp.pl sent_1 received_1 > timestampRX
sleep 1
echo "-"

# script calcula throughput: acumula tamaño de pkt por segundo
echo "- Calculando Throughput."
perl throughput.pl timestampRX > thRX
sleep 1
echo "-"

# script calcula pktloss: enviados, recibidos, pktloss, %pktloss
echo "- Calculando PacketLoss"
perl pktloss.pl sent_1 received_1 > PER_packetLoss
sleep 1
echo "-"
sleep 1
echo "-"
echo "-"
echo " A continuacion se reconstruye el video en YUV"
echo "-"
echo "-"
echo "-"
echo "-"
echo "-"
echo "-"

# Obtenemos YUV recibido para calcular PSNR
ffmpeg -i $vid_rec vid_rec.yuv
echo "-"
echo "-"
echo "-"
echo "-"
echo " A continuacion se calcula PSNR y SSIM"

```

```

echo "-"
echo "-"
echo "-"
echo "-"
echo "- PSNR"
sleep 1

# Obtenemos PSNR
psnr $size_x $size_y $subsampling $yuv_in vid_rec.yuv > AAA
nl AAA | awk '{print $1, $2}' > psnr_final
rm AAA
echo "-"

echo "- SSIM"
sleep 1
# Obtenemos SSIM
psnr $size_x $size_y $subsampling $yuv_in vid_rec.yuv ssim > AAA
nl AAA | awk '{print $1, $2}' > ssim_final
rm AAA
echo "-"
echo "-"
echo "-"
echo "-"
sleep 1
echo " A continuacion se obtienen graficas de PacketDelay, thoroughput, PSNR y SSIM "
sleep 1
echo "-"
sleep 1
echo "-"
sleep 1
echo "-"
sleep 1
echo "-"

# A continuacion se obtienen gráficas de delay, thoroughput, psnr y ssim:
gnuplot -c Plots delay_rx thRX psnr_final ssim_final

```

En los siguientes subcapítulos se presentan los códigos fuente de los scripts que se utilizaron en esta etapa:

3.1. Código fuente Perl *trazas.pl*.

Este script se utiliza para generar una única traza de envío y recepción para posteriormente realizar el cálculo de *Packet Delay*.

```

# Script para obtener una traza completa de paquetes recibidos
# con sus respectivos tiempos de recepcion, tamaño y tiempo de envío.
#
# el comando es:
#
# perl trazas.pl sent_1 received_1 > TrazaTXRX
#
#

```

```

# El argumento [0] es la traza de envío y tiene la sgte forma:
#
# Col 1: orden de envío
# Col 2: Id de paquete
# Col 3: tiempo de envío
#
# El argumento [1] es la traza de recepción y tiene la sgte forma:
#
# Col 1: Id de paquete
# Col 2: tiempo de recepcion
# Col 3: tamaño del paquete
#
# La traza resultante tiene la forma:
#
# Col 1: Orden de envío
# Col 2: Id de paquete
# Col 3: Tiempo de envío
# Col 4: Tiempo de recepcion
# Col 5: Tamaño de paquete
#
#####

$enviado=$ARGV[0];
$recibido=$ARGV[1];

open (RX,"<$recibido")|| die "Can't open $recibido $!";

while (<RX>) {

    @r = split(' ');

    open (TX,"<$enviado")|| die "Can't open $enviado $!";

    while (<TX>) {

        @t = split(' ');

        if ($t[1] eq $r[0] )
        {
            print "$t[0] $r[0] $t[2] $r[1] $r[2]\n";
        }
        else
        {
            {
            }
        }
    }
    close TX;
}
close RX;

```

3.2. Código fuente Perl *GenTimeStamp.pl*.

Este script se utiliza para generar un *timestamp* que registra el delta de t de los paquetes recibidos y su tamaño. Con esta información es calculado posteriormente el *Throughput*.

```

# Generador de TimeStamp en RX
# a partir de las trazas sent_1 y received_1
#
# El comando es:
# perl GenTimeStamp.pl sent_1 received_1 > timestampRX

```

```

#
# La salida es utilizada para calcular el throughput en RX

$sent=$ARGV[0];
$received=$ARGV[1];

open (DATA,"<$sent")|| die "Can't open $sent $!";
@y = split(' ',<DATA>);

$Tini = $y[2];# el tiempo inicial del timestamp es el tiempo
#en el que se envía el primer paquete
#luego se restan todos los tiempos de recepcion menos este tiempo inicial
# y se obtiene el delta de t transcurrido desde el tiempo inicial, y
# se adiciona el tamaño recibido en ese instante.

#print "$Tini\n";

close DATA;

open (DATAr,"<$received")|| die "Can't open $received $!";

while (<DATAr>) {

    @x = split(' ');

    $t = $x[1]-$Tini;
    print "$t $x[2]\n";

}

close DATAr;
exit(0);

```

3.3. Código fuente Perl *throughput.pl*.

Este script se utiliza para calcular el *Throughput* en el host destino.

```

# Este script calcula el throughput en nodo destino
# a partir del timestamp calculado en la etapa anterior.
#
# El comando es: perl throughput.pl timestampRX > thRX
#
#

$timestamp=$ARGV[0];
$interval=1;# define el intervalo de tiempo para acumular bytes transmitidos

$tamano=0;#suma de tamaño paquete
$tiempo=0;#contador de tiempo

open (DATA,"<$timestamp")|| die "Can't open $timestamp $!";

while (<DATA>) {

    @x = split(' '); #separa DATA en columnas por espacio y el contenido lo almacena en la matriz @x

    if ($x[0]-$tiempo <= $interval)
    {

```

```

        $tamano=$tamano+$x[1];#se acumula cantidad de bytes recibidos
    }
    else
    {
        if ($tamano ne 0)# cuando el acumulador contiene bytes acumulados para throughput
        {
            $th=$tamano/$interval;# se calcula el throughput
            print "$x[0] $th\n";#se imprime
            $tiempo=$tiempo+$interval;# se aumenta un intervalo
            $tamano=0; #se resetea el acumulador de bytes
        }
        else
        {
            $tiempo=$tiempo+$interval; #se incrementa el acumulador de tiempo
        }
    }
}

#throughput del último intervalo de tiempo
$th=$tamano/$interval;
#se imprime
print "$x[0] $th\n";

close DATA;
exit(0);

```

3.4. Código fuente Perl *pktloss.pl*.

Este script se utiliza para calcular el *Packet Loss*. El archivo de salida muestra el total de paquetes enviados, recibidos, perdidos, y el porcentaje de paquetes perdidos.

```

# Script para calcular el porcentaje de paquetes perdidos
#
# El comando es: perl pktloss.pl sent_1 received_1 > PER_packetLoss
#
$enviados=$ARGV[0];
$recibidos=$ARGV[1];

$env=0;#contador enviados
$rec=0;#contador recibidos

open (DATA,"<$enviados")|| die "Can't open $enviados !";
$env++ while <DATA>;
close(DATA);

open (DATA,"<$recibidos")|| die "Can't open $recibidos !";
$rec++ while <DATA>;
close(DATA);

$NumPktLoss = $env-$rec;
$PER_Pkt_Loss = ($NumPktLoss/$env)*100;

print "Enviados Recibidos PktLoss Per_PktLoss\n";
print "$env $rec $NumPktLoss $PER_Pkt_Loss";

```


3.5. Código fuente script Plots.

Este script consiste en un conjunto de líneas de comandos para obtener, con GNUplot, las gráficas de las variables obtenidas anteriormente: *Packet Delay*, *Throughput*, *PSNR*, *SSIM*.

```
plot "delay_rx" w l linecolor rgb "blue" title 'Delay'
set grid
set xlabel 'Numero de paquete'
set ylabel 'Retardo [seg]'
set title 'Retardo de paquetes'
set term png
set output 'PacketDelay.png'
replot
plot "thRX" w l linecolor rgb "blue" title 'Throughput'
set grid
set xlabel 'Tiempo [seg]'
set ylabel 'Throughput [Bytes/seg]'
set title 'Throughput en nodo destino'
set term png
set output 'Throughput.png'
replot
plot "psnr_final" w l linecolor rgb "blue" title 'PSNR'
set grid
set xlabel 'FRAME'
set ylabel 'PSNR [dB]'
set title 'PSNR'
set term png
set output 'PSNR.png'
replot
plot "ssim_final" w l linecolor rgb "blue" title 'SSIM'
set grid
set xlabel 'FRAME'
set ylabel 'SSIM [#]'
set title 'SSIM'
set term png
set output 'SSIM.png'
replot
```

4. Parámetros de codificación en el codificador HM (encoding_config.cfg).

En la etapa de codificación se utilizó el codificador HM (*HEVC test Model*). Muchos de los parámetros de configuración no fueron registrados, dejando así sus valores por defecto, sin embargo, los parámetros configurados son mencionados a continuación:

4.1. Parámetros de codificación del video

ShakeNDry_3840x2160_120fps_420_10bit_YUV.yuv

```

#===== File I/O =====

InputFile          :/home/salopezgu/Videos/4K/ShakeNDry/DogShake.yuv
InputBitDepth      : 10      # Input bitdepth
InputChromaFormat  : 420    # Ratio of luminance to chrominance samples
FrameRate          : 120     # Frame Rate per second
FrameSkip          : 0      # Number of frames to be skipped in input
SourceWidth        : 3840    # Input frame width
SourceHeight       : 2160    # Input frame height
FramesToBeEncoded  : 300     # Number of frames to be coded

#===== others

BitstreamFile      :dog.bin
ReconFile          :dog.yuv
SummaryOutFilename :true
SummaryPicFilenameBase : true
QuadtreeTULog2MaxSize :5
GOPSize            :1
Profile            :main10

#===== Coding Structure =====

IntraPeriod        : 1      # Period of I-Frame ( -1 = only first)
DecodingRefreshType : 1      # Random Accesss 0:none, 1:CRA, 2:IDR, 3:Recovery Point SEI
GOPSize            : 1      # GOP Size (number of B slice = GOPSize-1)
ReWriteParamSetsFlag : 1     # Write parameter sets with every IRAP
QP                 : 25

```

4.2. Parámetros de codificación del video Vidyo1

```

#===== File I/O =====

InputFile          :/home/salopezgu/Videos/Vidyo1/vidyo1-300frs.yuv
InputBitDepth      : 8      # Input bitdepth
InputChromaFormat  : 420    # Ratio of luminance to chrominance samples
FrameRate          : 60     # Frame Rate per second
FrameSkip          : 0      # Number of frames to be skipped in input

```

SourceWidth : 1280 # Input frame width
SourceHeight : 720 # Input frame height
FramesToBeEncoded : 300 # Number of frames to be coded

#===== others

BitstreamFile :outencoded.bin
ReconFile :outencoded.yuv
SummaryOutFilename :true
SummaryPicFilenameBase : true
QuadtreeTULog2MaxSize :5
GOPSize :16
Profile :main

#===== Coding Structure =====

IntraPeriod : 32 # Period of I-Frame (-1 = only first)
DecodingRefreshType : 1 # Random Accesss 0:none, 1:CRA, 2:IDR, 3:Recovery Point SEI
GOPSize : 16 # GOP Size (number of B slice = GOPSize-1)
ReWriteParamSetsFlag : 1 # Write parameter sets with every IRAP

IntraQPOffset : -3

LambdaFromQpEnable : 1 # see JCTVC-X0038 for suitable parameters for IntraQPOffset, QPoffset, QPOffsetModelOff, QPOffsetModelScale when enabled

Type POC QPoffset QPOffsetModelOff QPOffsetModelScale CbQPoffset CrQPoffset QPfactor tcOffsetDiv2
betaOffsetDiv2 temporal_id #ref_pics_active #ref_pics reference pictures predict deltaRPS #ref_idcs reference idcs

Frame1: B 16 1 0.0 0.0 0 0 1.0 0 0 0 2 3 -16
-24 -32 0

Frame2: B 8 1 -4.8848 0.2061 0 0 1.0 0 0 1 2 3 -
8 -16 8 1 8 4 1 1 0 1

Frame3: B 4 4 -5.7476 0.2286 0 0 1.0 0 0 2 2 4 -
4 -12 4 12 1 4 4 1 1 1 1

Frame4: B 2 5 -5.90 0.2333 0 0 1.0 0 0 3 2 5 -2
-10 2 6 14 1 2 5 1 1 1 1 1

Frame5: B 1 6 -7.1444 0.3 0 0 1.0 0 0 4 2 5 -1
1 3 7 15 1 1 6 1 0 1 1 1 1

Frame6: B	3	6	-7.1444	0.3	0	0	1.0	0	0	4	2	5	-1
-3	1	5	13	1	-2	6	1	1	1	1	0		
Frame7: B	6	5	-5.90	0.2333	0	0	1.0	0	0	3	2	4	-2
-6	2	10	1	-3	6	0	1	1	1	1	0		
Frame8: B	5	6	-7.1444	0.3	0	0	1.0	0	0	4	2	5	-1
-5	1	3	11	1	1	5	1	1	1	1	1		
Frame9: B	7	6	-7.1444	0.3	0	0	1.0	0	0	4	2	5	-1
-3	-7	1	9	1	-2	6	1	1	1	1	0		
Frame10: B	12	4	-5.7476	0.2286	0	0	1.0	0	0	2	2	3	
-4	-12	4	1	-5	6	0	0	1	1	1	0		
Frame11: B	10	5	-5.90	0.2333	0	0	1.0	0	0	3	2	4	-
2	-10	2	6	1	2	4	1	1	1	1			
Frame12: B	9	6	-7.1444	0.3	0	0	1.0	0	0	4	2	5	-1
-9	1	3	7	1	1	5	1	1	1	1	1		
Frame13: B	11	6	-7.1444	0.3	0	0	1.0	0	0	4	2	5	-
1	-3	-11	1	5	1	-2	6	1	1	1	1	0	
Frame14: B	14	5	-5.90	0.2333	0	0	1.0	0	0	3	2	4	-
2	-6	-14	2	1	-3	6	0	1	1	1	1	0	
Frame15: B	13	6	-7.1444	0.3	0	0	1.0	0	0	4	2	5	-
1	-5	-13	1	3	1	1	5	1	1	1	1	1	
Frame16: B	15	6	-7.1444	0.3	0	0	1.0	0	0	4	2	5	-
1	-3	-7	-15	1	1	-2	6	1	1	1	1	0	

#===== Quantization =====

QP : 18 # Quantization parameter(0-51)