Performance Comparison of H.264/AVC and HEVC Standards over LTE Networks

by
Ravneet Sohi
Bachelor of Technology, Kurukshetra University, 2011

Project Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Engineering

in the
School of Engineering Science
Faculty of Applied Science

SIMON FRASER UNIVERSITY

SUMMER 2016

Approval

Name:	**Ravneet Sohi**
Degree:	**Master of Engineering**
Title:	**Performance Comparison of H264/AVC and HEVC Standards over LTE Networks**
Examining Committee:	**Chair:** Dr. Mirza Faisal Beg Professor

Dr. Jie Liang
Senior Supervisor
Associate Professor

Dr. Jiangchuan (JC) Liu
Supervisor
Professor

Date Defended/Approved: February 23, 2015

Abstract

Long Term Evolution (LTE) is a standard for wireless communication developed by the 3rd Generation Partnership Project (3GPP) with an aim to fulfill the requirements defined for the fourth generation (4G) wireless networks. With more than hundred service providers across the globe and around one billion subscribers predicted by the year 2016, LTE is set to become the first true global standard. With the high data rates supported by LTE, improvements like Content Distribution Network (CDN) and increase in router switching speeds, popularity of video streaming services over the mobile networks is set to touch a new high. This popularity provides opportunity to the network operators to increase their revenues, but it also challenges them to provide video streams with minimum desirable quality to their customers. This has led to the emergence of two popular single layer video coding standards namely, H.264/AVC and more recently H.265/High Efficiency Video Coding (HEVC). With LTE being projected as a candidate to fuel the future 4G services, it is desirable to evaluate the performance of these two video coding standards over the LTE networks. The first part of this project tries to evaluate the video quality offered by these two video coding standards over LTE network by studying the impact of delay, distance and number of users in the LTE cell.

In the second part of this project we implement a frame dropping mechanism which drops low priority frames of the video encoded with hierarchical B-frame structure when the channel conditions are not ideal, thus providing graceful degradation to the single layer videos. This mechanism tries to exploit the fact that in a video that is encoded using a hierarchical structure, the loss of a frame that belongs to the higher indexed temporal layer of a video has less negative impact on the video quality in comparison to the loss of a frame in the lower indexed temporal layer.

Keywords: H.264/AVC, JSVM, HEVC, HM, Video Quality, LTE, LTE-SIM Simulator

ACKNOWLEDGEMENT

I am using this opportunity to express my gratitude towards my senior supervisor, Dr. Jie Liang for his valuable guidance, motivation and support for this project. I am sincerely grateful to him for taking time out and sharing his truthful and illuminating opinion on various aspects of this project. I would also like to thank Dr. Liang for providing me with an opportunity to work as a Research Assistant for this project.

I would also like to express my warm thanks to the chair, Dr. Mirza Faisal Beg and my supervisor, Dr. Jiangchuan (JC) Liu for serving on my committee, providing me with their valuable advice and feedback. Finally, I want to thank my parents for not just funding my education, but also for their constant support throughout the duration of this project.

TABLE OF CONTENTS

List of Tables

List of Figures

LIST OF ACRONYMS

3GPP	3rd Generation Partnership Project
AMC	Adaptive Modulation and Coding
AVC	Advanced Video Codec
CABAC	Context Based Adaptive Binary Arithmetic Coding
CAVLC	Context-Based Adaptive Variable Length Coding
CQI	Channel Quality Information
CB	Coding Block
CBT	Coding Tree Block
CTU	Coding Tree Unit
DPB	Decoded Picture Buffer
EPC	Evolved Packet Core
EXP–PF	Exponential Proportional fair Scheduler
FDD	Frequency Division Duplexing
GoP	Group of Pictures
GSM	Global System of Mobile Communications
KHz	Kilo Hertz
MHz	Mega Hertz
HEVC	High Efficiency Video Coding
HSPA	High Speed packet Access
HSS	Home Subscriber Service
LTE	Long Term Evolution
MC	Motion Compensation

MCS	Modulation and Coding Scheme
MIMO	Multiple Input Multiple Output
MME	Mobility Management Entity
MV	Motion Vector
NAL	Network Abstraction Layer
OFDM	Orthogonal Frequency Division Multiplexing
OFDMA	Orthogonal Frequency Division Multiple Access
QOS	Quality of Service
PB	Prediction Block
PRB/RB	Physical Resource Block
P-GW	Packet Data Network Gateway
PF	Proportional Fair
RAN	Radio Access Network
RB	Resource Block
SAE	System Architecture Evolution
SPS	Sequence Picture Set
S-GW	Serving Gateway
TB	Transport Block
TDD	Time Division Duplexing
UE	User Equipment
UMTS	Universal Mobile Telecommunications System
VLC	Video Coding Layer

Chapter 1

LTE

1.1 Introduction

LTE (Long Term Evolution) has emerged as a next logical step for cellular operators around the world. LTE is a standard for wireless communication developed by the 3rd Generation Partnership Project (3GPP) with an aim to provide high throughput to the end-users, reduce network latency, provide seamless mobility experience, improve spectral efficiency, support flexible spectrum allocation and provide enhanced Quality of Service (QOS) [1]. In addition, LTE is optimized to provide service to the users having a vehicular speed from 0 kmph to 15 kmph, but can provide mobility support up to 500 kmph [2]. In order to achieve the desired results, LTE relies on improvements in air interface like Orthogonal Frequency Division Multiple Access (OFDMA), Multiple Input Multiple Output (MIMO) technology and smart antennas [3]. Moreover, it is designed to co-exist with the existing cellular systems such as UMTS (Universal Mobile Telecommunications System) and GSM (Global Systems for Mobile Communications). Some of the key parameters used in LTE networks are listed in Table 1.

Peak Data Rates (for 20 MHz Bandwidth)	Downlink : 100 Mbps
	Uplink : 50 Mbps
Mobility Support	Upto 500 kmph
Control Plane Latency	< 100 ms
User Plane Latency	< 5ms
Control Plane Capacity	> 200 users per cell (for 5 MHz spectrum)
Bandwidths Available	1.2 , 3 ,5 , 10 , 15, 20 MHz

Table 1: Key LTE Parameters [4]

1.2 LTE TRANSMISSION SCHEME

The work on LTE started in the year 2004 with an objective to provide new radio-access technology, which was aimed at developing a network that eliminates the use of circuit switching and provides access through packet switched domain only. The transmission scheme used in LTE is based on Orthogonal Division Frequency Multiplexing (ODFM). OFDM uses large number of relatively narrow band subcarriers instead of few wideband subcarriers that were used in the earlier schemes. In HSPA (High Speed Packet Access) for example, the overall transmission bandwidth of 20 MHz consists of four wide-band carriers of 5 MHz each. On the other hand, OFDM transmission scheme used in the LTE divides this 20 MHz bandwidth into hundreds of narrow sub-carriers. Figure 1 shows the difference between a single carrier spacing used in the earlier schemes and the subcarrier spacing employed in the OFDM. Use of the OFDM provides high degree of robustness, since narrow sub-bands in OFDM based transmission schemes make them immune to signal corruption that may occur due to the frequency selective fading in wireless mediums. Moreover, OFDM makes bandwidth allocation easier as bandwidth can be varied easily, merely by varying the number OFDM sub-carriers.

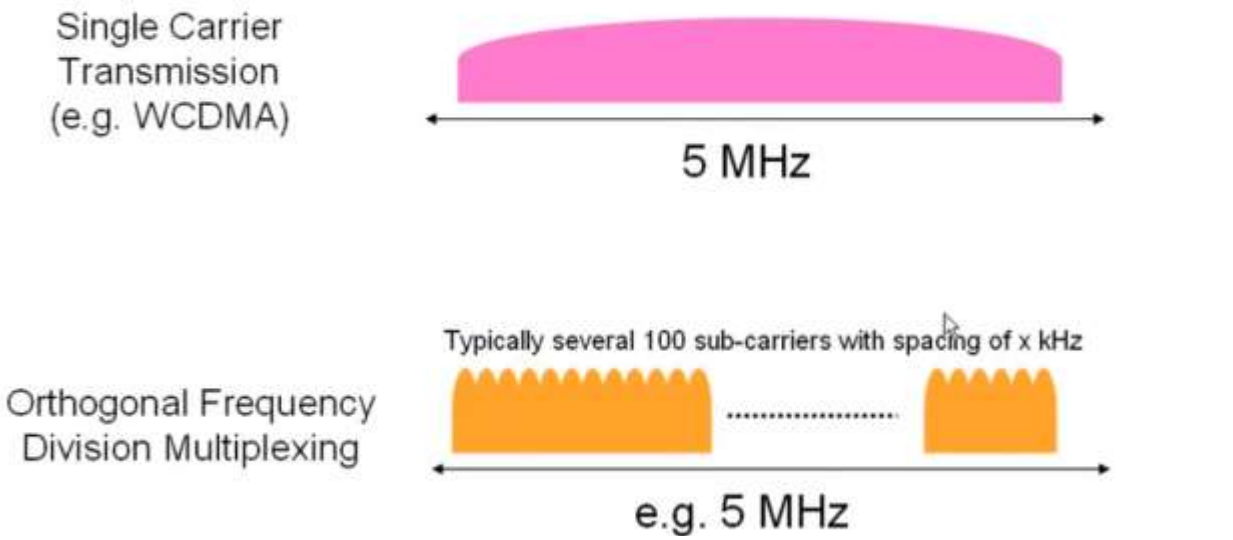


Figure 1: Difference between sub-carrier spacing in Single Carrier Transmission and Orthogonal Frequency Division Multiplexing [5]

LTE uses an extension of the OFDM called Orthogonal Frequency Division Multiple Access (OFDMA). Advantage of using OFDMA is that the scheduler has access to the frequency domain channel. Thus by using OFDMA, users can be allocated both in time as well as in frequency domain.

In order to achieve higher data rates, higher bandwidths are necessary. Thus, an important feature of LTE is the support for flexible spectrum. LTE can support different transmission bandwidths. In LTE, physical layer can have any bandwidth from 1.4 MHz and 20 MHz in steps of 180 KHz (which corresponds to one Physical Resource Block). However, for simplicity and easier implementation, current LTE specifications support a subset of 6 different system bandwidths. Bandwidths supported according to current specifications are 1.4, 3, 5, 10, 15 and 20 MHz. Table 2 shows the number of Physical Resource Blocks (PRBs) corresponding to the channel bandwidths specified for the LTE systems.

Channel Bandwidth (in MHz)	1.4	3	5	10	15	20
Number of Resource Blocks	6	15	25	50	75	100

Table 2: Channel Bandwidth and Corresponding Number of Resource Blocks

1.3 LTE Network Architecture

In addition to the development of specifications, there have been improvements in the network architecture for the LTE systems. The new architecture is called System Architecture Evolution (SAE). Development of this architecture resulted in flat Radio Access Network (RAN) architecture and new core network architecture called Evolved Packet Core (EPC). RAN is responsible to execute radio-related functionalities like scheduling, retransmission, coding and multi-antenna schemes. EPC on the hand is responsible for handling non-radio functions such as authentication, handling billing information, setup and release of end-to-end connections [3]. Figure 2 shows the various components that form the EPC and the RAN in the LTE network architecture.

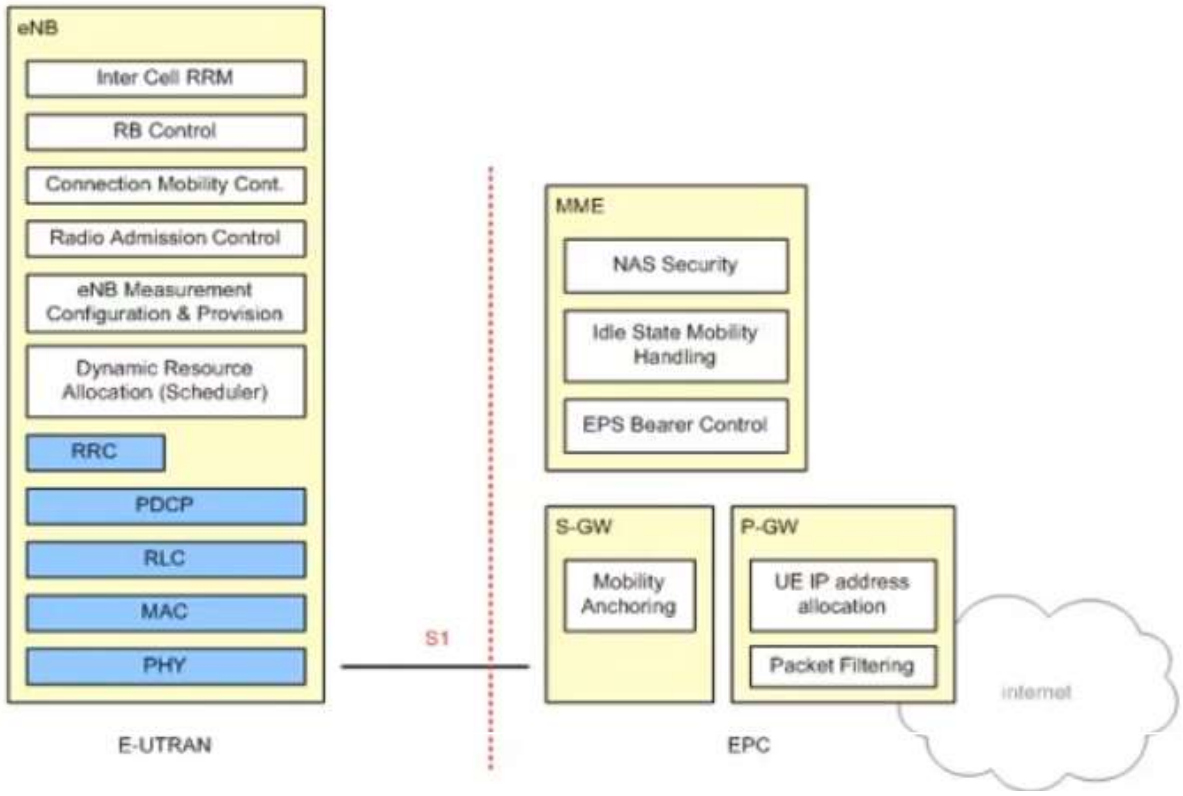


Figure 2: SAE Architecture of LTE consisting of RAN and EPC [5]

Core network in the LTE is essentially different from the core network used in previous cellular systems as it allows access through packet-switched domain only. Core network in the LTE consists of several sub-systems, some of which are discussed below.

Mobility Management Entity (MME) is a control plane sub-system. It is responsible for Bearer control (connection setup and release of bearers). It also handles idle to active transition and handles security issues related to the non-access stratum. Serving Gateway (S-GW) forms a connection between the radio access of the LTE and the core network. It also forms a mobility anchor for the UEs that move from one base station in LTE to another. Apart from this, it also handles billing information. The Packet Data Network Gateway (PDN Gateway, P-GW) is a subsystem that forms a connection between the core network and the Internet. It is responsible for allocating IP addresses to UEs. Secondly, Quality of Service is implemented by the PDN gateway. Thirdly, it forms the mobility anchor for the users subscribed to non 3-GPP radio access technologies. Home Subscriber Service (HSS) is a database responsible for maintaining information about subscribers.

Radio Access Networks in LTE consists of single type of node called eNodeB (eNodeB is similar to Base station in other system). It is responsible to implement all the radio related functionalities in the LTE cell. As a result, eNodeB has to handle a lot of functions such as inter-cell radio management, radio admission control and most importantly – scheduling algorithm is implemented at the eNodeB. ENodeBs are connected to the EPC components through a S1 interface and to each other via X2 interface. X2 interface is used for Inter-Cell Interference Co-ordination (ICIC) which helps in reducing the interference between the neighbouring cells [3].

1.4 Scheduler

In LTE systems, scheduler (both downlink and uplink) is implemented at eNodeB. Since LTE has a strong support for Quality-of Service, scheduling operations become challenging (especially in presence of real time multimedia application) and is designed to maximize the cell throughout while being fair to all users.

LTE system implements a channel-sensitive scheduler which uses the fact that the each user experiences different fast fading channel conditions than those experienced by the other users. When the number of users are large, probability to find a user with good channel conditions maximizes, which leads to good spectral utilization over a period of time. Maximum Throughput and Proportional Fair (PF) are two basic scheduling algorithms that exploit the concept of channel sensitive scheduling. Many other scheduling strategies that are modifications of these strategies have also been implemented.

Each User Equipment (UE) periodically monitors the quality of the signal sent to it by the base station. UE reports this quality back to the base station as Channel Quality Index (CQI) in the form of uplink control messages. This CQI feedback is then used by the scheduler to make a scheduling decision according to the scheduling algorithm implemented. In addition, CQI is also used by the base station to find the most suitable modulation scheme and calculate the coding rate that must be used for transmission to the UE on that sub-channel.

Basic Scheduling Algorithm - At the start of each Transmission Time Interval (TTI), the scheduler makes a list of all the flows that have data to be sent to their respective users. The scheduler then calculates a metric for each flow on each sub-channel. Let $w_{i,j}$ represent the metric for the i^{th} flow on j^{th} sub-channel. Scheduler calculates this metric based on CQI feedbacks using algorithm specific to that scheduler. Next, the scheduler assigns j^{th} sub-channel to the flow with the highest metric for that TTI. This process is repeated for all the sub-channels in each TTI. Then scheduler maps the CQI feedback with appropriate modulation and coding scheme (MCS). MCS is further used to calculate the size of the data that will be transmitted on that sub-channel in that particular TTI. This complete procedure is repeated in each TTI for each sub-channel.

Metric Calculation – Primary difference between all schedulers is the procedure they employ and the variables they use to calculate the metric. For our simulation part, we have used Exponential Proportional Fair (EXP-PF) Scheduler [6]. LTE-SIM has this scheduler implemented in its scheduling module. A brief description of the EXP–PF Scheduler is given below.

Let $r_{i,j}$ be the instantaneous data rate for the i^{th} flow on j^{th} subchannel and let the average transmission rate $\overline{Ri}$ (k) for i^{th} flow after the k^{th} interval be defined as [14]

$$\overline{Ri}\,(k) = 0.8\ \overline{Ri}\,(k) + 0.2\ R_i\,(k)$$

where $R_i\,(k)$ is data rate achieved in k^{th} TTI for the i^{th} flow

$\overline{Ri}$ (k) is average data rate calculated after k^{th} interval for the i^{th} flow

$\overline{Ri}$ $(k\text{-}1)$ is average data rate calculated after k-1 th interval for the i^{th} flow

Then metric calculation for the real time flow in EXP– PF scheduler is done by following equation [14]:

$$W_{i,J} = \exp\left(\frac{\alpha_i D_{HOL,i} - \chi}{1+\sqrt{\chi}}\right)\left(\frac{r_{i,j}}{\overline{Ri}}\right)$$

where

$W_{i,j}$ is the metric associated with j^{th} sub-channel for i^{th} flow

$\overline{Ri}$ is the average data rate for the i^{th} flow

$r_{i,j}$ is instantaneous data rate associated with j^{th} sub-channel for the i^{th} flow (which is computed by AMC module based on the CQI reports sent back by the user to base station)

The term α_i is defined as [14]

$$\alpha_i = \frac{-\log \delta_i}{\tau_i}$$

where τ_i is packet delay threshold and

δ_i is maximum probability that Head of Line Packet Delay $D_{HOL,i}$ will exceed τ_i for a real time packet flow i.

The term $D_{HOL,i}$ is Delay of the Head of Line Packet (First packet to be transmitted in the queue associated with the flow i) .

Further term χ is defined as [14]

$$\chi = \frac{1}{N_{rt}} \sum_{i=1}^{N_{rt}} \alpha_i D_{HOL,i}$$

where N_{rt} represents the number of active downlink real time flows .

For non-real time flow, metric used is [14]

$$W_{i,j} = \frac{r_{i,j}}{\overline{Ri}}$$

where $\overline{Ri}$ and $r_{i,,j}$ have same meaning as described earlier.

EXP Scheduler has been designed to increase the priority of the real time flows like video traffic and voice traffic in comparison to the non-real time flows like simple web browsing traffic. This scheduler is extension of Proportional Fair scheduler.

1.5 Duplex Schemes in LTE

Spectrum flexibility is one of the important characteristics of LTE. LTE not only provides flexibility in terms of bandwidth but also provides support for operations in both Frequency Division Duplex (FDD) mode and Time Division Duplex (TDD) mode.

FDD uses two carrier frequencies simultaneously, one for uplink and one for downlink. In each frame, there are 10 downlink sub-frames and 10 uplink sub-frames. Duplex filter is used to enable simultaneous downlink and uplink operations. Although base station in LTE system must support the full-duplex capability, specific UE in the cell may not have full duplex capability.

In Time Division Duplex (TDD) operation, downlink and uplink operation is done on same carrier frequency. Downlink and uplink operation in this case is not simultaneous. Each frame is divided into 10 sub-frames of 1 millisecond each, with sub-frame number 0 reserved for downlink and sub-frame number 5 reserved for uplink. Rest of the sub-frames can be allocated for either downlink or uplink, depending upon the configuration used. There are seven such configurations that can be used for sub-frame allocation. A switch from the downlink to uplink operation and the vice versa in TDD is made using special sub-frame [3].

Chapter 2

H.264/AVC Video Standard

2.1 Introduction

Recent Internet traffic trends suggest that video traffic has emerged as the dominant form of traffic and will remain so in the future. There has been a steady increase in real time traffic on mobile networks. Mean monthly mobile data usage in North America in year 2013 was 443.5MB [8] and real-time traffic consisting of applications such as video and audio streaming was the dominant traffic in this usage. In fact, 29.5% of upstream traffic, 39.91% of downstream traffic and 37.53% of aggregate traffic during peak periods actually came from real-time streaming applications [8].

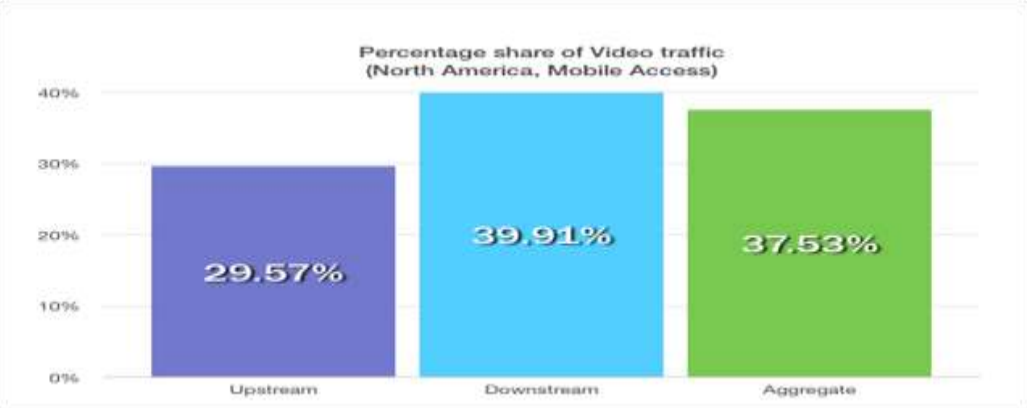


Figure 3: Percentage share of Video in Mobile Traffic in North America for year 2013

H.264/AVC is a video compression standard that is widely used for compression of High Definition (HD) videos and videos of other formats for the purpose of storage and transmission over the network. Main aim behind development of this standard was to develop a video compression standard that increases coding efficiency of the video and provides substantial bit rate reduction in comparison to the previous standards (like H.263) when encoding videos of similar video quality.

2.2 H.264/AVC Coding Technique Highlights

In H.264/AVC video standard, video data is primarily encoded using two layers – Network Abstraction Layer (NAL) and the Video Coding Layer (VCL). VCL is the layer that basically encodes the video source data according to the encoding mechanism that has been defined in the H.264/AVC standard. NAL formats the VCL data in such a way that transmission of this data over the network becomes easier and provides a header that contains the information that enables the encoded video to serve wide categories of application. NAL also maps the data encoded by the VCL to the transport layers of the network.

In H.264/AVC, video bitstream is organised as Network Abstraction Layer (NAL) units. NAL units consist of a one byte header (which indicates the type of data) and the payload itself. Payload of NAL units is interleaved with bits that prevent the accidental generation of a code that may represent start of NAL unit. Payload of the NAL units can either be the video data encoded by the VCL or it can be a non – VCL payload.

Non VCL payload usually contains additional information about a VCL payload in form of parameter sets or it can be Supplementary Enhancement Information (SEI) that gives additional information (such as timing information) about the video bit-stream but is not necessary to decode the data [10]. Parameter is a non-VLC data that gives general information that is applicable to decoding of large number of NAL units that contain VCL payload.

A set of related consecutive NAL units form an Access Unit. Successful decoding all the NAL units composing an access unit results in one decoded picture. Start of an access unit is indicated by a special prefix called access unit delimiter. This prefix is followed by a sequence of primary NAL units that contain slice data that represents samples of the video picture. This is followed by redundant information about the coded picture that may help the decoder reconstruct a picture if some data related to primary NAL units has been lost.

VCL produces an encoded source data which is obtained by block based hybrid video-encoding approach. Similar approach was used in previous standards, but new features that have been added to the H.264/AVC that allow it to have better coding efficiency. Video picture in H.264/AVC is represented as YCbCr colour space. Component Y is called the Luma component and it represents the brightness. Cb is called the Chroma component and it measures the deviation of the color gray from the color blue in the sample. Component Cr is also called Chroma component and it gives the measure of the deviation of the color gray from the red in the sample. Sampling resolution of the Cb and Cr components in H.264/AVC is reduced in comparison to the sampling resolution of the Y component to take advantage of the fact that human eye is more sensitive to brightness than it is towards colour.

Pictures in the H.264/AVC standard are coded (and decoded) using basic building blocks of fixed size called macroblocks. Macroblocks are rectangular regions of 16 x 16 samples in case of luma component and 8 x 8 in case of chroma components. Related macroblocks are grouped together to form slices. Slices are a set of macroblocks that are encoded in the order of raster scan. A picture can be divided into several slices as shown in figure 4.

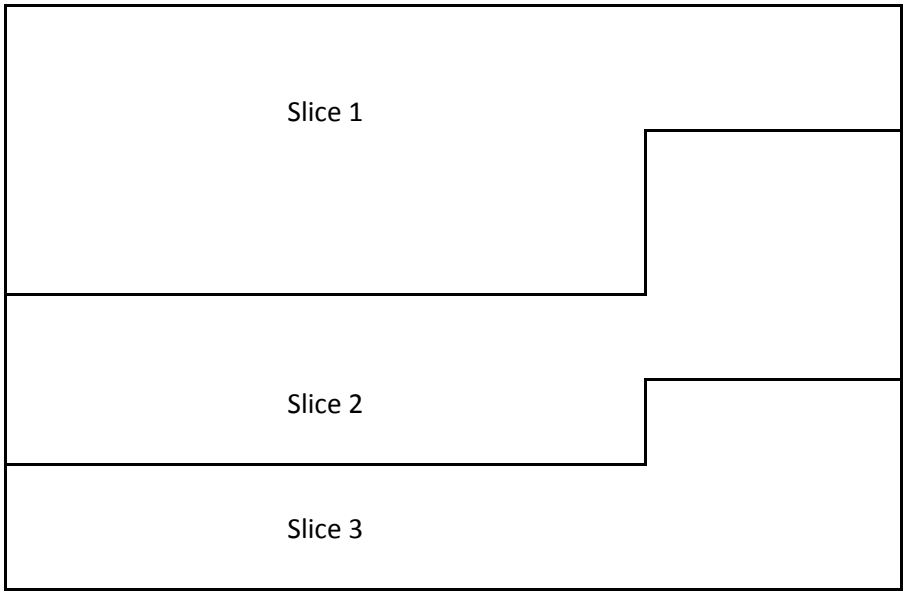


Figure 4: Division of a picture into multiple slices

Samples of the macroblocks can be spatially or temporally predicted. Depending on the type of prediction, H.264/AVC supports five type of slice coding

1) I-Slice : Utilizes intra-picture coding.

2.) P-Slice : Employs Intra and Inter predictive coding with one predicted signal.

3.) B-Slice : Employs Intra and Inter predictive coding with two predictive signals.

4.) SP Slice : SP Slice is a new type of slice that has been introduced in the H.264/AVC that allows efficient switching from pre-encoded pictures.

5.) SI Slice : SI slice is also new slice that has been introduced in H.264/AVC that facilitates random access and error recovery process [10].

Intra prediction of I-slices is done using one of the several directional-spatial modes that are defined in H.264/AVC specifications. Intra-prediction is done by using one of these modes to create a prediction signal by comparing the samples of the current block with the samples of the neighbouring blocks that have already been encoded. In case of luma blocks, size of 8 x 8 samples or 16 x 16 samples can be used for intra- prediction. However, in the case of chroma components only option available is to use complete macroblock for the intra prediction.

For prediction of P-slices and B-slices, block size is indicated by the macroblock type and block size of 16 x 16 luma samples, 8 x 16 luma samples, 16 x 8 luma samples and 8 x 8 luma samples are allowed. If block size indicated is 8 x 8 luma samples, the block can be further divided. For P-slice, one motion vector for prediction is used for each block. In case of B-slices, three prediction methods are defined. Either of the three methods can be used for prediction of each block. These prediction methods are simply called list 0, list 1 and bi-predictive method. When list 0 prediction method is used, prediction signal is generated using a picture from reference picture list 0. Similarly, when list 1 prediction method is used, prediction signal is generated using a picture from reference picture list 1. When bi-predictive method is used, prediction signal

is generated using weighted sum of the two signals that are generated using pictures from the reference picture list 0 and reference picture list 1.

H.264/AVC coding standard produces blocking artifacts due to the use of macroblocks. As a result of this, picture that is reconstructed has inferior quality at the block edges. In order to detect blocking artifacts, the difference between the samples near the block edges is calculated. If this difference is high, it is considered to have been arisen due to a blocking artifact. For reducing blocking artifacts, H.264 employs an adaptive de-blocking filter. In loop deblocking filter used in this process helps to achieve bit rate reduction in the range of 5% to 10 % in comparison to the video encoded without the use of deblocking filter.

H.264 supports two type of entropy coding methods – first is Context-Based Adaptive Variable Length Coding (CAVLC) which uses variable length coding. Other entropy coding method called Context Based Adaptive Binary Arithmetic Coding (CABAC), which employs more sophisticated coding mechanism and thus is more bit rate efficient than CAVLC. Usually, CABAC provides bit rate savings in the range of 5% to 15% over CAVLC [9].

H.264/AVC video standard allows much more flexibility in terms of Reference Picture Memory Control in comparison to previous standards. Reference picture list that contain the pictures that are used as reference pictures for prediction of B and P slices can be controlled by the use of Reference Picture List Reordering (RPLR) commands. Decoded Picture Buffer (DPB) has capacity to hold up to 16 frames.

H.264/AVC defines a set of profiles that specify a set of algorithms that can be used to create conforming bit-streams that are useful to facilitate interoperability among certain applications. H.264/AVC standard defines three such profiles – Baseline, Main and Extended profile. Decoder that conforms to a particular profile must support all the features that have been defined for that particular profile.

Chapter 3

HEVC / H.265 Video Standard

3.1 Introduction

High Efficiency Video Coding (HEVC) is designed as a successor to H.264 /AVC video standard. It is jointly developed by the ISO/IEC Moving Picture Experts Group and ITU-T Video Coding Experts Group (VCEG). Older video encoding standards such as H.264 AVC served well till now, but increase in diversity of the video applications being offered in the last few years (for example increase in the popularity of HD video) may lead to increase in the frequency of network congestions in commercial access networks in the near future. Thus, HEVC video codec was developed with an aim to provide bit rate reduction of approximately 50% in comparison to the existing H.264/AVC standard for equal levels of video quality. HEVC is designed to address four key issues – support increased resolution (resolution up to 8192 x 4320), make integration with transport protocols easier, support parallel processing architecture and support data loss resilience [12].

HEVC supports multiple profiles like the Main profile that supports bit depth of 8 bits, Main 10 profile that supports bit depth of 10 bits, a Main Still Picture profile for still digital pictures and a group of profiles that are known as Range Extensions (RExt).

3.2 HEVC Coding Design and Features

Each picture during HEVC encoding process is split into blocks. Map of the block partitioning is fed to decoder at the other end. First picture of the each video sequence and the first picture at the clean random access point (CRA) are coded with intra-picture prediction. For the remaining pictures, hierarchical inter-picture prediction is used. Inter-picture prediction involves choosing the reference pictures to be used and the Motion Vector (MV) to be applied for predicting the samples of the each block. Motion Compensation (MC) data is obtained from the MV and mode decision data (which is present in the encoded bit stream) [12].

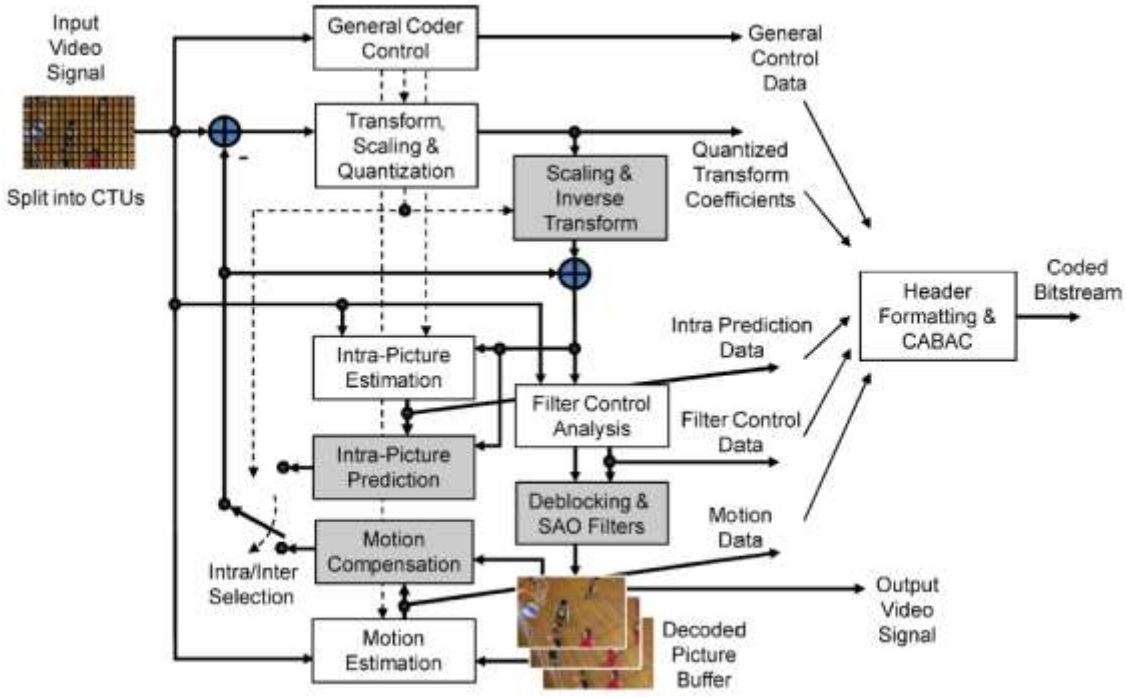


Figure 5: Block Diagram of HEVC Video Encoder [12]

Video Layer Coding (VLC) in HEVC employs basically same approach of using combination of inter and intra prediction along with the 2D (2-Dimension) transform coding, as employed in other video coding standards before HEVC. Residual signal goes through scaling, quantisation and entropy coding. Quantisation transform coefficients are then calculated by inverse scaling and inverse transformation. This is done so that encoder and decoder have same residual signal approximation [12]. Then residual signal is added to the prediction signal and resulting picture is smoothed by the use of a loop filter to reduce the artifacts that may arise due to the block processing. The resulting picture is then stored in Decoded Picture Buffer (DPB) and can be used as a reference for predicting other pictures in the Group of Pictures (GoP).

3.3 HEVC Video Coding Technique Highlights

For representing colour video signals, HEVC employs colour space with 4:2:0 sampling. This means that sampling resolution of Luma component is twice as sampling resolution of Chroma component. Video of pictures with resolution W x H is scanned progressively, where W is the width and H is the height.

Each picture in the HEVC standard is partitioned into Coding Tree Unit (CTU). Each CTU consists of one Luma Coding Tree Blocks (CTBs) and two Chroma CTBs. CTUs are the basic processing unit in the HEVC decoding process (like macroblocks in H.264). Luma CTBs cover an area of L x L Luma components and Chroma CTBs cover an area of L/2 x L/2 Chroma components, where L can be 16, 32 or 64 samples. In comparison to H.264 which employed 16 x 16 size CTBs (macroblocks of fixed size), HEVC employs variable size CTBs. Larger size CTBs make encoding efficient when encoding higher resolution videos.

Luma and Chroma CTBs can directly be encoded as Coding Blocks (CB) or can further be divided into multiple CBs. This partitioning is done with help of quad tree signalling that allows partitioning of CTBs of appropriate sizes based on characteristics of the picture being encoded. This partitioning can go on iteratively until it reaches lowest allowed value.

Prediction in HEVC can be intra or inter prediction. If prediction mode is intra, the PB size is same as CB for all block sizes.

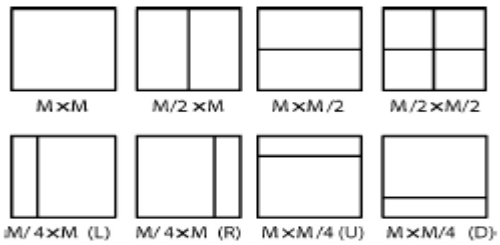


Figure 6: Modes for splitting a CB into PBs, subject to certain size constraints. [12]

In case of inter picture prediction mode, it is specified whether Luma and Chroma CB will be partitioned into one, two or four PB. Splitting of CB into four PBs of equal size is allowed only when CB size is equal to minimum size that is allowed. When CB is split into two PBs, six type of splitting is possible. Figure 6 illustrates partitioning possibilities for CBs. For intra picture prediction, partitioning of only L x L and L/2 x L/2 types is allowed. Lower section of the figure 6 shows four partitions that are together part of a group called Asymmetric Motion Partitioning (AMP) and these can be used only when L is equal to 16 or larger for Luma samples. The Luma and Chroma PBs, together with the associated corresponding syntax, form one Prediction Unit (PU) [12].

For residual coding, CB is iteratively partitioned into quadrants. This partitioning is mapped in the residual quad-tree. Only square shape partitions of residual CBs are allowed. Given a Luma CB of L x L Luma samples, samples can be divided into four L/2 x L/2 block, by setting corresponding flag. These blocks can be further split iteratively into four quadrants using another flag, if minimum allowed size for partitioning is not reached which is indicated by Sequence Picture Set (SPS).

Slices are the sequences of CTUs that are encoded in the order of the raster scan. A picture can be partitioned into one or more slices. Slices are independent coding units such that if certain picture parameters are provided, values of samples of the picture that a slice represents can be successfully decoded without the use of any data from other slices. As in H.264 video standard, slices can be coded using three different coding types-

1.) I-Slice – All CUs in the slice are coded using intra-picture prediction.

2.) P-Slice – In addition to coding techniques applied to coding CUs in I slice, some CUs can be uni-predicted, which is inter-prediction that allows one motion compensated prediction signal per Prediction Block (PB).

3.) B-Slice - Similarly B-Slices allow CUs to be bi-predicted, that is they allow two motion compensation prediction signals per PB.

Intra-prediction in HEVC is done in accordance to TB size by making comparison with previously decoded boundary samples from neighbouring TBs. For square TBs with size in range of 4 x 4 to 32 x 32, 33 directional orientations can be used in addition to planar and DC predictions [12]. In case of Chroma samples, prediction modes can be signalled separately or can be indicated to be same as that of Luma samples.

HEVC defines various intra-picture predictive coding methods such as Intra_Angular, Intra_Planar and Intra_DC. Intra_Angular prediction takes advantage of regions with strong directional edges. Intra_Angular prediction used in HEVC has even better performance than that used in H.264 because of the use of 33 direction orientations in HEVC instead of just 8 such orientations used in H264 and partly due to increase in size of TBs. Intra_DC prediction mode uses average value of reference samples for prediction. Intra_planar prediction is useful when dealing with discontinuities along block boundaries since it employs average value of predictions that use corner reference samples.

Samples of PB for inter-picture predicted CB are obtained by first determining the reference picture indicated by the reference picture index and then determining the corresponding block of this reference picture using the horizontal and vertical component of the motion vector. If the motion vector does not have integer value, then technique called fractional sample interpolation is used to generate the prediction signals. In comparison to H264, which used two stage processes for interpolation (which involved rounding off the intermediate results), interpolation process in HEVC is a single stage process which improves precision and thus makes the implementation of interpolation process easier [12].

CABAC is used for entropy coding in HEVC. CABAC was also used in H.264 AVC, but CABAC used in H.265 has undergone improvements to support parallel-processing, minimize memory requirements and overall compression performance.

In-loop de-blocking filter used in HEVC is similar to one used in H.264, but it has been designed for easier decision making and has been optimised for parallel processing.

HEVC also defines the concept of tiles. Tiles are rectangular regions of the picture that are helpful in parallel processing, thus making implementation of encoding and decoding process faster. One tile may contain several slices. Conversely, multiple tiles will share same header if they are a part of same slice.

As in H.264/AVC, HEVC also specifies concept of Profile and Levels. Profile defines a set of algorithms that are used to encode conforming bit stream. Some of the profiles defined in HEVC standard are Main, Main 10 and Main Still Picture Profile. Main and Main still picture profile support bit depth of 8 bits per sample, whereas when using Main_10 Profile, 10 bits per sample are supported. When using Main Still Picture Profile, entire video is encoded as one picture. Level on the other hand, defines certain parameters that put certain restrictions depending on decoder capabilities that may come in the shape of maximum resolution supported, maximum bit rate supported, minimum compression ratio supported or size of DPB. Apart from Profiles and Levels, in order to cater demands of applications that differ in terms of maximum bit rate and Coded Picture Buffer (CPB) capabilities, two Tiers are specified in HEVC – Main Tier that is used for normal applications and High Tier for demanding applications.

Chapter 4

Frame Skip Mechanism

4.1 Frame Skip Mechanism

For second part of this project, we work on a simple frame skip mechanism that intentionally skips frames of less priority that belong to the topmost temporal layer (of the video encoded using hierarchical structure) when the channel quality received by a UE is poor. Proposed mechanism makes use of the properties of hierarchical structure of the video coding used in the recent video encoding standards such as H.264/AVC and new H.265/HEVC video standard. Frames are skipped when Channel Quality Index (CQI) reports indicate that channel conditions for a particular UE are not appropriate for sending video of a high bit rate and thus sending a video composed of frames of only lower indexed temporal layers provide graceful degradation to the video received by the user.

In LTE Systems, available subcarriers are stacked together to form a Physical Resource Block (PRB) and bandwidth resources are allocated to the users in units of PRBs. Each PRB represents a bandwidth of 180 KHz which is allocated to the user for one TTI (Transmission Time Interval).

Each UE after receiving downlink signal sent by the base station, measures the Signal-to-Noise Ratio (SNR) being received on the sub channels. These measurements are reported back to the base station in the form of CQI reports. CQI can be classified as wideband CQIs and sub-band CQIs. In wideband CQIs, report contains the average of the channel quality received throughout the spectrum. Whereas, sub-band CQIs report contains channel quality for each sub-band. Additionally, CQI reports can be classified on the basis of how periodically CQI reports are sent back by the UE to the base stations as aperiodic and periodic CQIs reports.

CQI reports are further used by the Base Station to make scheduling decisions and select modulation and coding rate (MCS) that is best suited for the UE on the sub channels that may be allocated to the UE depending on the scheduling decision, in such a way that Block Error Rate (BLER) does not exceed 10%. CQI index used in LTE have values from 0 to 15, with value of 0 meaning no signal is received and value of 15 corresponds to best channel conditions. Table 3 shows the modulation and the coding schemes used in the LTE systems.

CQI Index	Modulation	Code Rate x 1024	Efficiency
0	no signal		
1	QPSK	78	0.1523
2	QPSK	120	0.2344
3	QPSK	193	0.377
4	QPSK	308	0.6016
5	QPSK	449	0.877
6	QPSK	602	1.1758
7	16QAM	378	1.4766
8	16QAM	490	1.9141
9	16QAM	616	2.4063
10	64QAM	466	2.7305
11	64QAM	567	3.3223
12	64QAM	666	3.9023
13	64QAM	772	4.5234
14	64QAM	873	5.1152
15	64QAM	948	5.5547

Table 3: Modulation and Coding Scheme used based on CQI index

Mechanism that we used to test the effectiveness of frame skipping works simply by skipping the transmission of the frames of the topmost temporal layer on which quality of no other layer depends. This can be simply done by communicating to the video server the wideband CQI of the user and instruct it to skip frames depending on the MCS value as shown in the figure 7.

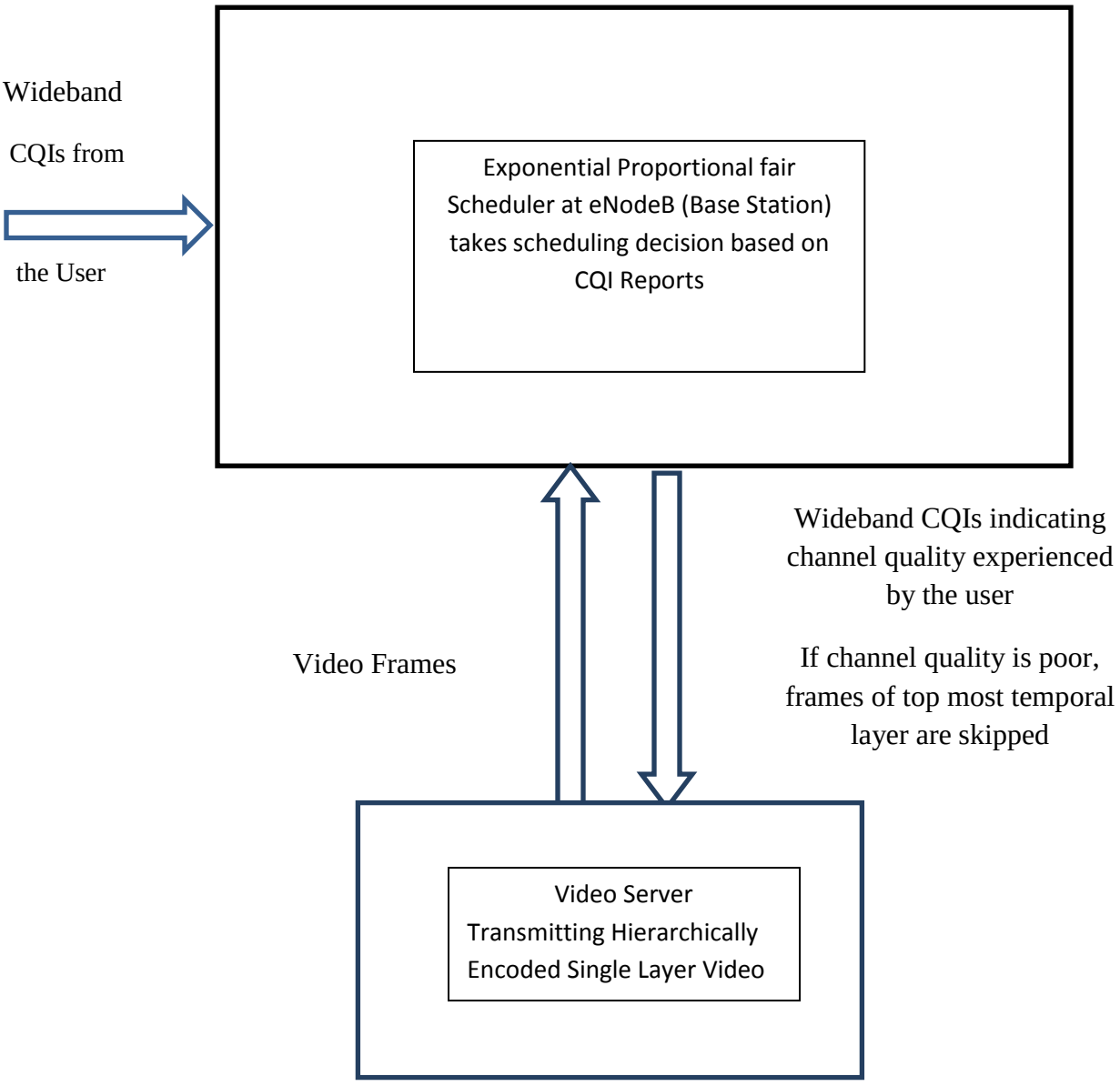


Figure 7: Frame Skip Mechanism based on Channel Quality Reports used in our Simulation

Conditions used in the HD Video Transmission for our simulation are:

1. Transmit all the temporal layers (T=0 to T=4) of the HD video if channel conditions are good and modulation scheme used is 64 QAM (MCS value is more than or equal to 10).

2. Drop the topmost temporal layer (T= 4) and transmit only temporal layers (T=0 to T =3) of the HD video if channel conditions are mediocre or poor and modulation scheme used is 16 QAM or QPSK (MCS value if MCS value is less than 10).

Software code used to implement the frame skip mechanism defined above is given Appendix C.

NOTE: Frame skip mechanism we employ is an off-shot of the Cross–Layer Mechanism designed for transmission of the Scalable Video Coding(SVC) [15], as we use channel quality reports to skip frames in a single layer video instead of choosing appropriate layers of SVC video to transmit over the LTE in the original design. Frame skipping conditions in our simulation are not optimised and can be changed depending on the bit rate of the video encoded and other parameters. Ideally, conditions for skipping frame should take all relevant parameters such as bandwidth available for a particular video flow, bit rate for the video and the queue size of a particular video flow into account. Moreover, frame skipping conditions could be further segmented to drop some more temporal layers. However, given the time constraints attached with the project, aim was to see the effectiveness of the frame skip mechanism rather than developing and optimising a full-fledged scheduler.

Chapter 5

Source Code to Skip Decoding a Frame in the HEVC Bitstream

In order to measure the PSNR value of the video received after LTE-SIM Simulation, video trace file is sent over the LTE network to the user in the simulator and the modified trace file (with certain frames dropped over wireless medium) is obtained. In order to find the PSNR value of the bit-stream thus obtained, we need to delete corresponding frames from bit-stream when decoding the bit-stream and replace them with the last successfully decoded frame (frame copy). There was no method implemented in HM Reference software by which we could skip the decoding of a particular frame (frame that has been dropped or corrupted during transmission) when decoding the encoded bit-stream. Moreover, although frame copy has been implemented in the HM Reference software, it does not work when we artificially skip decoding of particular frame as in our case. Thus in order to tackle above problem, code was developed to first skip a specific frame and call the method that implements frame copy in the HM. After implementation of code, we can simply input the frames we want to skip during decoding process from the console/command prompt. Software code used to implement the frame skip mechanism defined above is given in Appendix A and the results of the tests done to validate the successful implementation of the code are shown in Appendix B.

Chapter 6

Simulation and Results

6.1 Introduction

For Simulation, single layer bit-streams of HD sequence named old_town [13] were encoded according to HEVC/H.265 video standard and H.264/ AVC video standard. Open Source HM reference software was used for encoding videos conforming to HEVC/H.265 video standard. Open Source JSVM reference software was used for encoding videos conforming to H.264 AVC video standard. Videos were encoded to meet different video quality standards (different average PSNR).

Traces for the videos encoded were generated and processed to make them compatible for transmission over the LTE-SIM Simulator [14]. LTE–Sim Simulator is an open source simulator widely used to test various aspects of LTE systems. LTE –Sim has in-built support for various schedulers such as Exponential Proportional Fair (Exp-PF), Proportional Fair, Maximum Throughput Scheduler, etc. Apart from this, it supports various aspects of the LTE systems such as the bandwidths defined for the LTE systems, Quality of Service standards, channel quality feedback modes, user mobility, multi cell environment, handover procedures, etc. It also implements all the necessary protocols used in the LTE systems - from the physical layer to the application layer. At the application layer, it supports trace based traffic, Voice over IP (VoIP) traffic, web traffic, Constant Bit Rate (CBR) traffic and Infinite Buffer traffic.

Figure 8 shows the curve illustrating the average PSNR v/s Bit Rate comparison for H.264/ AVC and HEVC/H.265 video encoding standards for the videos encoded. Table 4 shows the corresponding data. It can be seen from the curve that the videos encoded with HEVC/H.265 standard show appreciably better video quality for similar bit rates, thus displaying better rate-distortion characteristics.

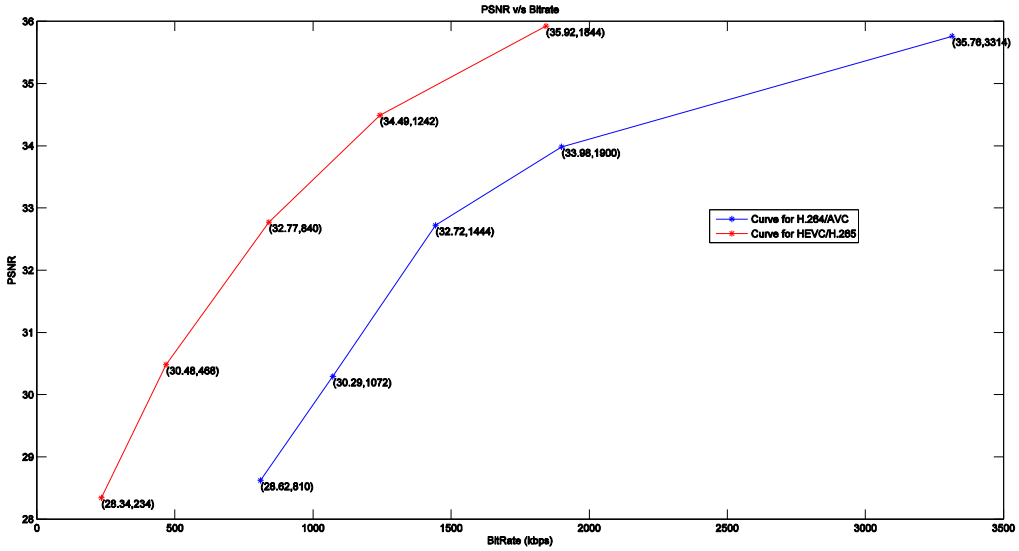


Figure 8: Rate – Distortion Curves comparing H.264/AVC and H.265 for tested sequences

H.264/AVC	
Bit Rate (in kbps)	PSNR
810	28.62
1072	30.29
1444	32.72
1900	33.98
3314	35.76

H.265/HEVC	
Bit Rate (in kbps)	PSNR
234	28.34
468	30.48
840	32.77
1242	34.49
1844	35.92

Table 4: Rate – Distortion Comparison for H.264/AVC and H.265 for tested sequences

6.2: Simulation - Comparison of Bandwidth Requirements

In order to compare the bandwidth requirements for the videos encoded with the H.264/AVC and H.265/HEVC video encoding standards over the LTE system, we transmitted the encoded videos to a single user at different distances from the base station (eNodeB). Aim of the simulation was to compare the bandwidth resources required for the transmission of videos encoded with video standards under study over LTE system without any loss of frame at different distances. Thus full LTE system bandwidth of 20 MHz was allocated to the base station so that no packet loss occurs for any of the bitrates or distances (between the base station and the user) under study.

Since Physical Resource Block (PRB) assigned to a particular flow gives a measure of bandwidth resources required for transmission of that particular flow, curve between number of physical resource blocks (PRB) used to transmit complete video (all 320 frames) without any frame loss for various distance between the base station and user is plotted. In order to minimize the number of variables, only the case where maximum allowed delay is 500ms is considered. Some important parameters used during the simulation are given the table below.

Number of UEs	1
Distance of UE from the Base Station	Variable
Downlink Bandwidth	20 MHz
Number of Usable PRBs	100
Video Used	old_town
Resolution	HD (1280 x 720)
Frame Rate	50 Frames / Second
GoP (Group of Pictures)	G16B15
Frames Encoded	320
Downlink Scheduler	Exponential Fair Scheduler
Speed of UE	0 km /hr
Frame Structure	FDD
Maximum Delay	500 milliseconds
Flow Duration	20 seconds
Simulation Duration	25 seconds

Table 5: Parameters used for the simulation for comparison of bandwidth resources required for the transmission of videos conforming to H.264/AVC and H.265 standards

In this simulation, we performed bandwidth requirement comparisons for the videos conforming to the H.264/AVC and H.265 video standards by placing a user at the distances of 200m, 400m, 600m, 800m, 1000m, 1200m and 1400 m from the base station. Data for all these comparisons is displayed in table 6. However to maintain conciseness, graph for comparisons of only three cases have been plotted - 200m (when the user is close to the base station), 800m (when the user is at a relatively medium distance from the base station) and 1400m (when the user is far from the base station). **Results of all the three cases show that videos encoded with H.265/HEVC video standard consume appreciably less bandwidth resources than those required for the transmission of videos encoded with H.264/AVC Single Layer Standard. For example, when distance between the user and base station is 800m, bandwidth resources required to transmit a video conforming to H.265 standard and encoded with PSNR value of 32.77 is 31% less than the bandwidth resources required to transmit same video conforming to H.264/AVC standard encoded to attain PSNR value of 32.72. Similarly, where video conforming to H.264/AVC standard and encoded with PSNR value of 35.76 requires 161391 RBs to transmit 320 frames of a video, same video conforming to H.265 standard and encoded with PSNR value of 35.92 required mere 88598 RBs.** Figure 9, 10 and 11 shows the comparison of bandwidth requirements at distances of 200m, 800m and 1400m respectively.

	Distance of the User from Base Station						
	200 m	400 m	600 m	800 m	1000 m	1200 m	1400 m
avc_ 810 kbps	16465	28822	36507	37666	44781	54937	68203
avc_1072 kbps	19006	34508	49843	50499	68415	69234	94167
avc_1444 kbps	24153	47853	62217	68412	84259	95729	130965
avc_1900 kbps	35904	65425	82217	91850	113703	123106	152393
avc_3314 kbps	55011	110971	146983	161391	187306	220290	275956
hevc_234 kbps	4879	8054	10378	11398	13709	16002	19451
hevc_468 kbps	8383	17765	21129	25119	25119	30134	41167
hevc_840 kbps	16714	28717	36836	47006	48522	56021	64993
hevc_1242 kbps	25578	43986	55487	59207	59207	82062	100529
hevc_1844 kbps	32455	59819	82655	88598	88598	129067	142154

Table 6: Comparison of Bandwidth Requirements (Number of Resource Blocks) for videos conforming to H.264/AVC and H.265 Standards at various distances between the user and the base station

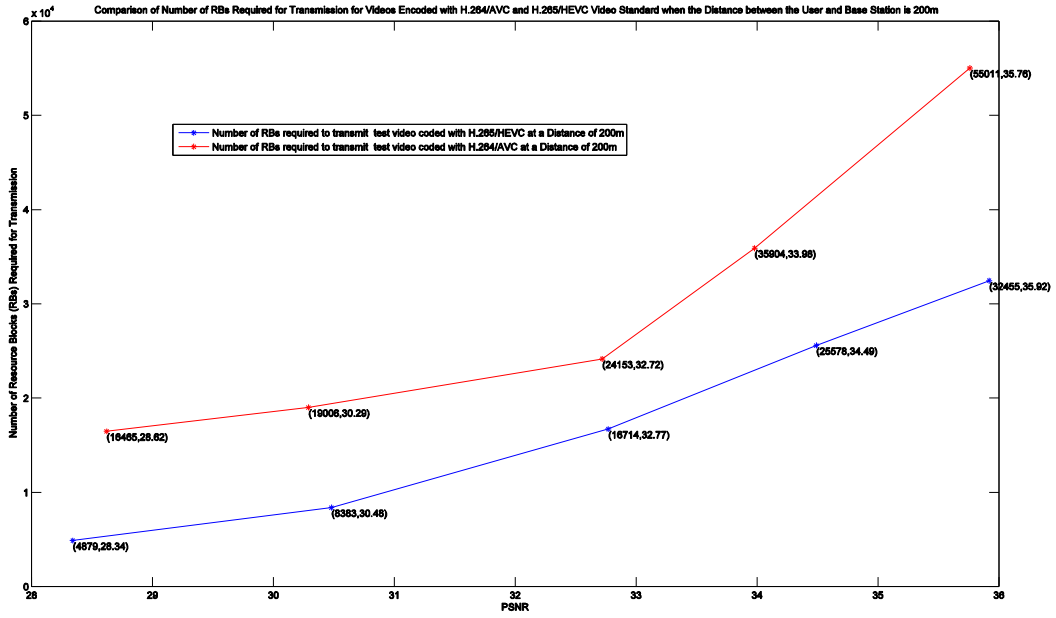


Figure 9: Comparison of Bandwidth Requirement (Number of RBS required) for videos conforming to H.264/AVC and H.265 Standards when distance between the user and the base station is 200m

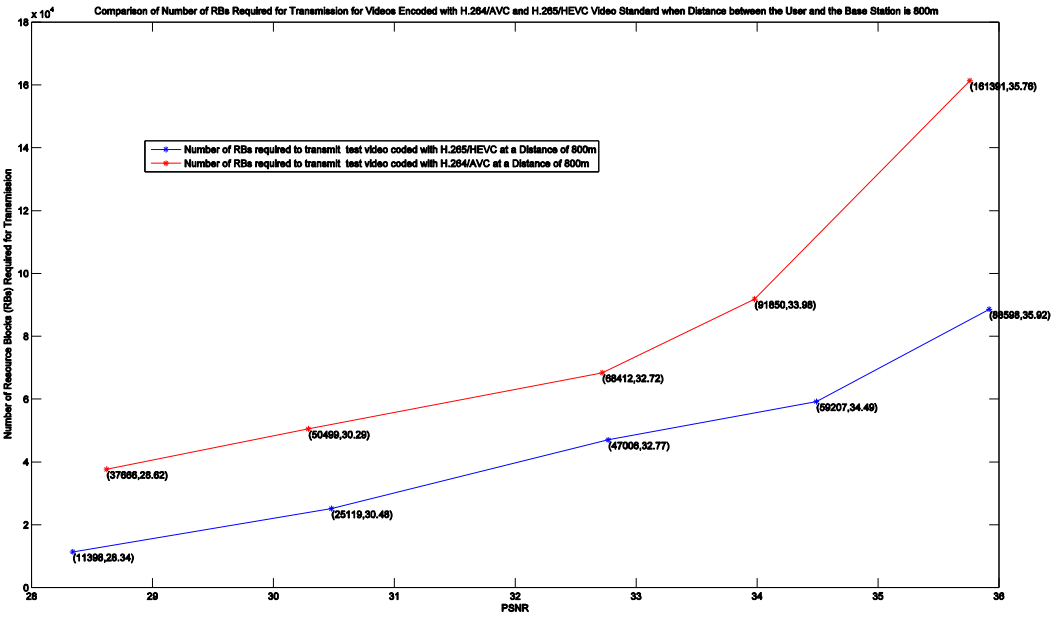


Figure 10: Comparison of Bandwidth Requirement (Number of RBS required) for videos conforming to H.264/AVC and H.265/HEVC Standards when distance between the user and the base station is 800m

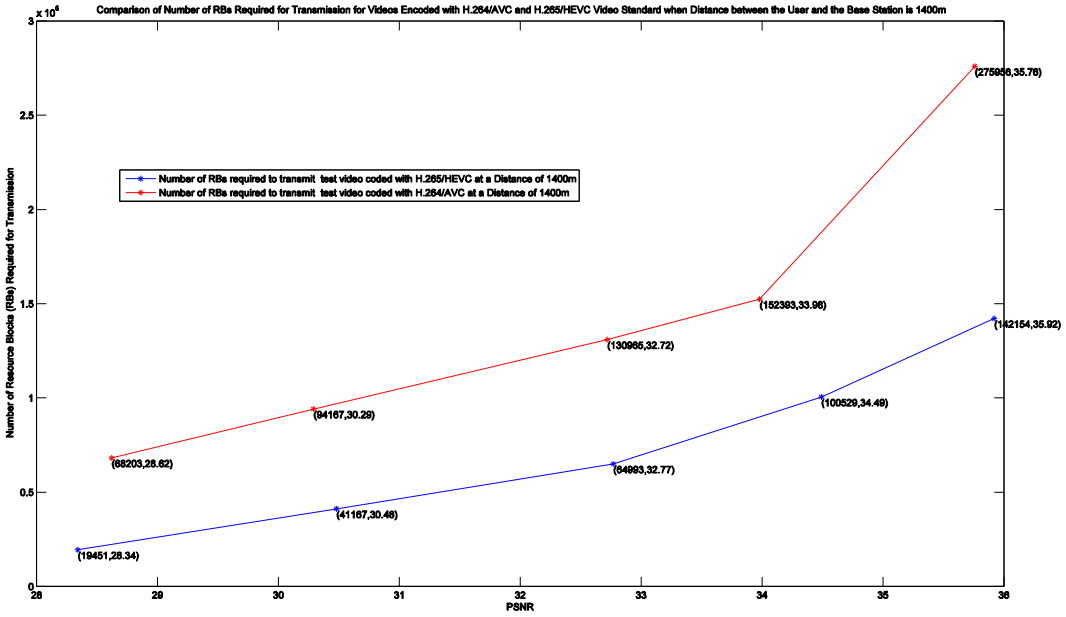


Figure 11: Comparison of Bandwidth Requirement (Number of RBS required) for videos conforming to H.264/AVC and H.265 Standards when distance between the user and the base station is 1400m

6.3: Simulation 2 - Effect of Maximum Allowed Delay and Number of Users in the Cell

In order to assess the effect of Maximum Delay (and thus Quality-of-Service constraints) and the Number of Users in the cell (Effect of Background Traffic) on the video quality received by a user in the LTE networks, we transmitted the traces of the videos encoded to a stationary user at 650 metres from the base station. Then we measured the quality of video received by measuring its PSNR value by varying maximum allowed delay and number of users (in steps of five users). A packet arriving after experiencing a delay of more than value set for Maximum Delay is discarded even if it is received successfully. The delay in LTE-Sim simulator is the queuing delay that a packet has to undergo at the base station and thus depends on number of factors like the size of the queue when the packet arrives for transmission at the base station, channel conditions and bandwidth available for a particular flow. Some of the important parameters that were used for this simulation are listed in the table below.

Number of UEs	Variable (Increased in steps of 5)
Transmission Radius of Each Cell	2 Kilometers
Downlink Bandwidth	20 MHz
Number of Usable PRBs	100
Video Used	old_town
Resolution	HD (1280 x 720)
Frame Rate	50
GoP (Group of Pictures)	G16B15
Number of Frames	320
Downlink Scheduler	Exponential Fair Scheduler
Speed of UE	0 km /hr.
Frame Structure	FDD
Maximum Delay	0.2, 0.3 ,0.4,0.5,0.6, 0.7
Flow Duration	20 seconds
Simulation Duration	25 seconds
Path Loss Model	Macro Cell Urban Area
Distance of the user from BS	650 meters

Table 7: Some Important Parameters used for this Simulation

Result 6.3.1: Effect of Maximum Allowed Delay When Number of Users are Less Than 20

In order to access the effect of Maximum Delay (and thus Quality-of-Service constraints) and Number of Users (Effect of Background Traffic) on the video quality received by a user in LTE networks, we started with background traffic of 5 users to whom foreman video sequence encoded with 128 kbps were transmitted. Trace of this video sequences is already implemented in LTE –SIM Simulator. To the 6^{th} user (user under observation), we transmitted the trace of the videos that we encoded using JSVM and HM reference software and calculated the PSNR value of the video thus received. Results remain the same when number of users which comprise the background traffic were increased to 10, 15 and 20 users. Our simulations show that at no packet is dropped for the videos encoded with the bit rates under study (for our project) at any considered value of maximum allowed packet delay and thus there is no decrease in the PSNR value of the received video in comparison to the original video (video before transmission) under these background traffic conditions. **As a result, curves for all the videos (H264/AVC videos and H.265 videos) shown in figure 12 are straight lines, indicating that the change in maximum allowed delay (and thus change in LTE Quality-of-Service parameters) have no effect on the PSNR value of the videos received over the LTE network for this background traffic (number of users in the cell) and for the range of maximum delays used in our study.**

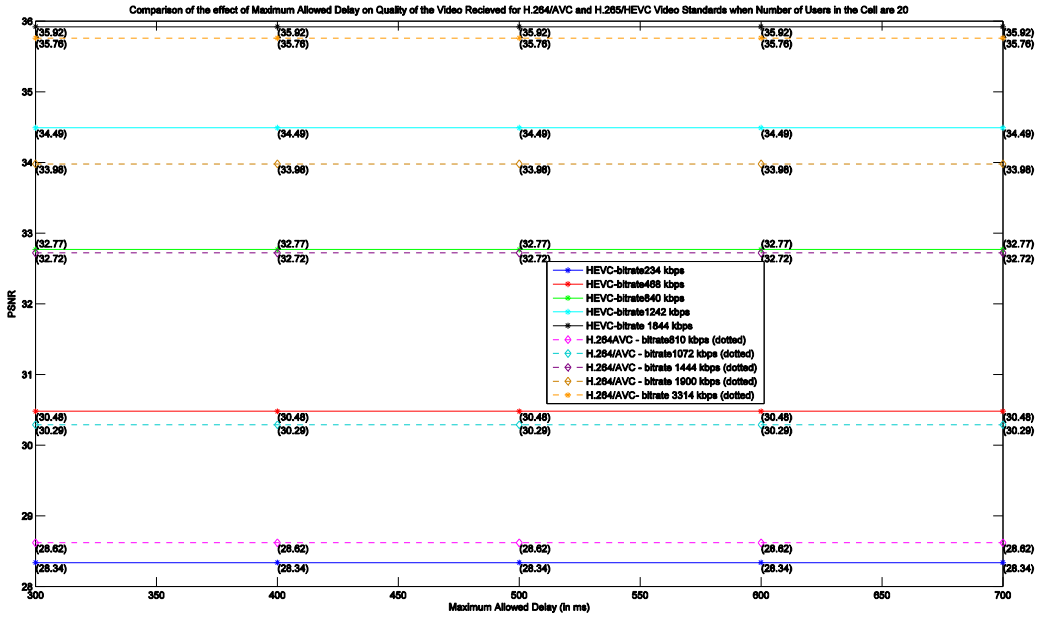


Figure 12: Comparison of PSNR of the videos encoded with H.264/AVC and H.265/HEVC standards at various maximum delays when number of users in the cell are less than or equal to 20.

	Delay				
	300ms	400ms	500ms	600ms	700ms
avc_ 810 kbps	28.62	28.62	28.62	28.62	28.62
avc_1072 kbps	30.29	30.29	30.29	30.29	30.29
avc_1444 kbps	32.72	32.72	32.72	32.72	32.72
avc_1900 kbps	33.98	33.98	33.98	33.98	33.98
avc_3314 kbps	35.76	35.76	35.76	35.76	35.76
hevc_234 kbps	28.34	28.34	28.34	28.34	28.34
hevc_468 kbps	30.48	30.48	30.48	30.48	30.48
hevc_840 kbps	32.77	32.77	32.77	32.77	32.77
hevc_1242 kbps	34.49	34.49	34.49	34.49	34.49
hevc_1844 kbps	35.92	35.92	35.92	35.92	35.92

Table 8: Comparison of the PSNR values of the videos encoded with H.264/AVC and H.265/HEVC when number of users in the cell are less than 20.

Result 6.3.2: Effect of Maximum Allowed Delay When Number of Users is Equal to 25

As it can be seen in figure 13 on the next page, when the background traffic is increased to 25 users in the cell, there is no loss of frame and thus no effect on the quality of the videos received when maximum allowed delay in LTE system is set to 600 milliseconds (ms) or more. As we decrease the maximum allowed delay to 500 ms, there is a decrease in quality of video received by the user from the original PSNR value of 35.76 to 34.77 for the video conforming to H.264/ AVC standard (encoded with bit rate of 3314 kbps). PSNR value for this video drops to 33.72 when maximum delay allowed for the video packets is set to 400ms. **Video quality for the H.264/AVC video encoded at 3314 kbps (PSNR = 35.76) deteriorates to 32.15 and becomes even lower than H.264/AVC videos encoded with bit rate of 1900 kbps (PSNR = 33.98) and 1444 kbps (PSNR= 32.72) when maximum allowed delay is reduced to 300 milliseconds due to excessive frame loss.**

Slight decrease in the video quality of H.264/AVC video encoded with 1900 kbps is also observed when the maximum allowed delay is reduced to 300ms. Quality of this video drops from PSNR value of 33.98 to 33.78. There is no deterioration for any of the videos encoded with H.265/HEVC Video Standard when number of users in the cell are 25. Yellow boxes in table 9 indicate the deterioration in video quality.

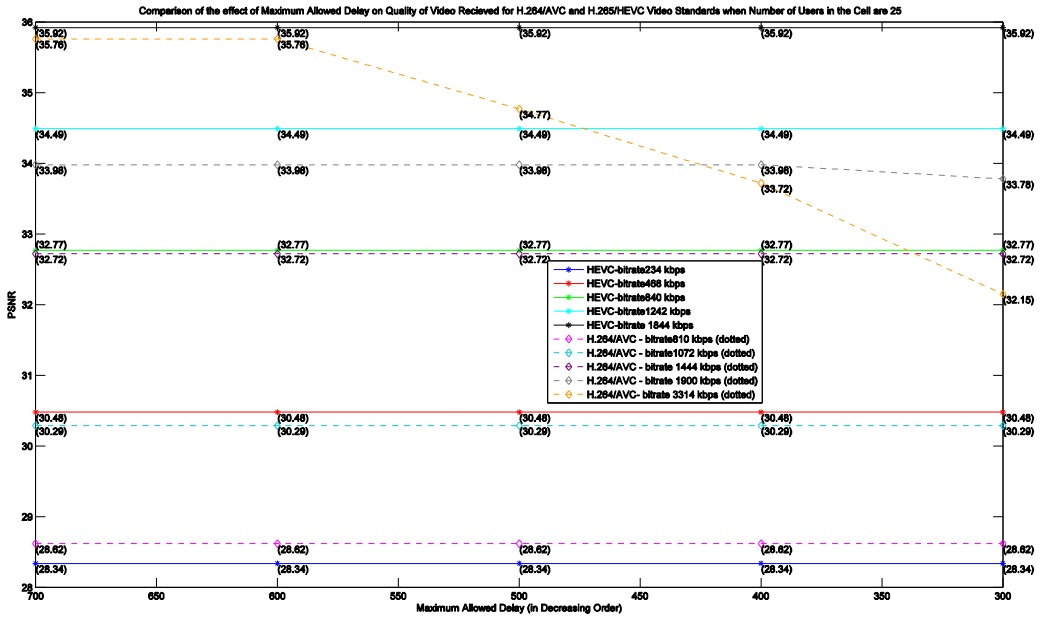


Figure 13: Comparison of PSNR of the videos encoded with standards H.264/AVC and H.265/HEVC at various maximum delays when number of users in the cell are equal to 25.

	Delay				
	300ms	400ms	500ms	600ms	700ms
avc_ 810 kbps	28.62	28.62	28.62	28.62	28.62
avc_1072 kbps	30.29	30.29	30.29	30.29	30.29
avc_1444 kbps	32.72	32.72	32.72	32.72	32.72
avc_1900 kbps	33.78	33.98	33.98	33.98	33.98
avc_3314 kbps	32.15	33.72	34. 77	35.76	35.76
hevc_234 kbps	28.34	28.34	28.34	28.34	28.34
hevc_468 kbps	30.48	30.48	30.48	30.48	30.48
hevc_840 kbps	32.77	32.77	32.77	32.77	32.77
hevc_1242 kbps	34.49	34.49	34.49	34.49	34.49
hevc_1844 kbps	35.92	35.92	35.92	35.92	35.92

Table 9: Comparison of the PSNR values of the videos encoded with H.264/AVC and H.265/HEVC when number of users in the cell are equal to 25.

Result 6.3.3: Effect of Maximum Allowed Delay When Number of Users is Equal to 30

When the cell traffic is increased to 30 users, H.264/AVC video having bit rate of 3314 kbps (golden dotted curve) and encoded with a PSNR value of 35.76 deteriorates to 35.58 when maximum allowed delay for the LTE system is 700ms, it becomes 32.24 at maximum allowed delay of 600ms and reduces further to 30.80 at maximum allowed delay of 300ms. **It can be observed that this AVC video encoded at high bit rate of 3314 kbps performs even worse than the AVC videos encoded with bit rates of 1444 kbps and 1900 kbps when maximum allowed delay is reduced. This loss in video quality can be attributed to the loss in large number of frames for the videos encoded at higher bit rates when bandwidth is constrained due to increase in number of users in the cell.** Slight loss in quality of the video received is also observed for the H.264/AVC video encoded with bit rate of 1900 kbps when maximum delay is 500ms or less. The PSNR value for this video is reduced from 33.98 to 33.07 when maximum delay for LTE system is set to 300 ms.

On the other hand, no loss in quality is observed for any of the videos encoded with HEVC/H.265 video standard for corresponding PSNR values.

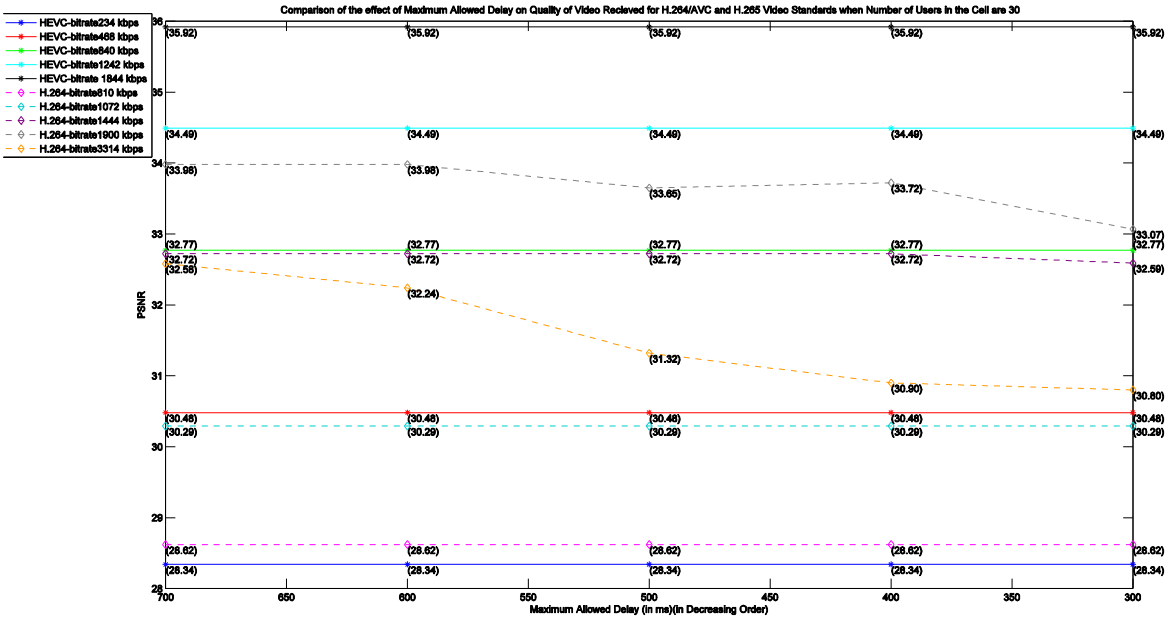


Figure 14: Comparison of PSNR of the videos encoded with standards H.264/AVC and HEVC at various maximum delays when number of users in the cell is equal to 30

	Delay				
	300ms	400ms	500ms	600ms	700ms
avc_ 810 kbps	28.62	28.62	28.62	28.62	28.62
avc_1072 kbps	30.29	30.29	30.29	30.29	30.29
avc_1444 kbps	32.59	32.72	32.72	32.72	32.72
avc_1900 kbps	33.07	33.72	33.65	33.98	33.98
avc_3314 kbps	30.8	30.9	31. 32	32.24	32.58
hevc_234 kbps	28.34	28.34	28.34	28.34	28.34
hevc_468 kbps	30.48	30.48	30.48	30.48	30.48
hevc_840 kbps	32.77	32.77	32.77	32.77	32.77
hevc_1242 kbps	34.49	34.49	34.49	34.49	34.49
hevc_1844 kbps	35.92	35.92	35.92	35.92	35.92

Table 10: Comparison of PSNR of the videos encoded with standards H.264/AVC and HEVC at various maximum delays when number of users in the cell is equal to 30.

Result 6.3.4: Effect of Maximum Allowed Delay When Number of Users is Equal to 35

When number of users in the cell are increased to 35, quality of the H.264/AVC video encoded with the bit rate of 3314 kbps and PSNR value of 35.76 deteriorates to the 29.25 when maximum delay is 700ms, further reduces to 29.22 when maximum delay is 600ms and deteriorates to 27.53 when maximum delay is 300ms. HEVC video encoded to provide similar video quality with a bit rate of 1844 kbps, deteriorates to PSNR value of 31.78 from original PSNR value of 35.92 when maximum delay is set to 300ms. AVC video encoded with bit rate of 1900 kbps and PSNR value of 33.98 deteriorates to 32.17 when maximum delay is set to 400ms and reduces further to 30.75 when maximum delay is reduced to 300 ms. **It can be seen that when Quality of Service parameter become stringent, videos encoded with high bit rates deteriorate to a video quality that is lower than videos encoded with lower bit rates.** No loss of frames is observed for any HEVC videos except HEVC (1844 kbps) and thus no loss in quality is observed for their transmission over LTE network.

Figure 15 shows the comparison of PSNR of the videos encoded with standards H.264/AVC and HEVC at various maximum delays when number of users in the cell are equal to 35.

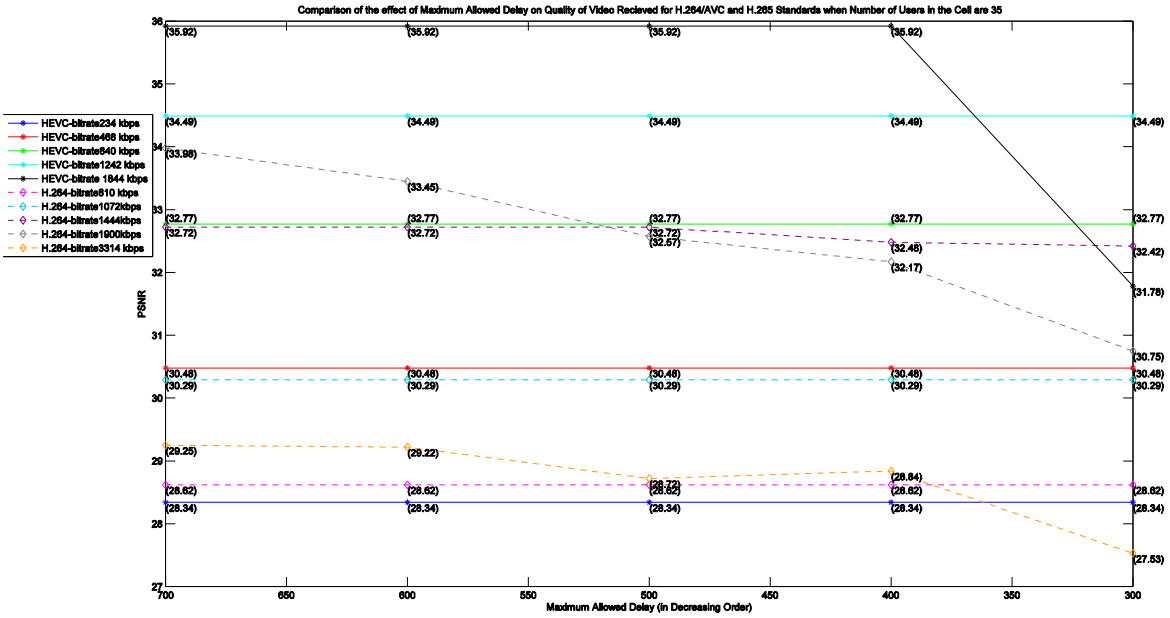


Figure 15: Comparison of PSNR of the videos encoded with standards H.264/AVC and HEVC at various maximum delays when number of users in the cell is equal to 35

	Delay				
	300ms	400ms	500ms	600ms	700ms
avc_ 810 kbps	28.62	28.62	28.62	28.62	28.62
avc_1072 kbps	30.29	30.29	30.29	30.29	30.29
avc_1444 kbps	32.42	32.48	32.72	32.72	32.72
avc_1900 kbps	30.75	32.17	32.57	33.45	33.98
avc_3314 kbps	27.53	28. 84	28.72	29.22	29.25
hevc_234 kbps	28.34	28.34	28.34	28.34	28.34
hevc_468 kbps	30.48	30.48	30.48	30.48	30.48
hevc_840 kbps	32.77	32.77	32.77	32.77	32.77
hevc_1242 kbps	34.49	34.49	34.49	34.49	34.49
hevc_1844 kbps	31.78	35.92	35.92	35.92	35.92

Table 11: Comparison of PSNR of the videos encoded with standards H.264/AVC and HEVC at various maximum delays when number of users in the cell is equal to 35

6.4: Simulation 3- Effect of Distance between the User Equipment and the Base Station

In order to access effect of distance between the user equipment and the base station on the video quality received by a user in LTE networks, we transmitted the traces of the videos encoded to a stationary user and varied their distances from the base station. Then we measured the quality of video received by measuring its PSNR value. Data indicating the PSNR values of the videos received at these distances is shown in table 12. In order to set appropriate and consistent background traffic for all simulations, 30 users were randomly distributed in the LTE cell and a trace of foreman video series (pre-implemented in LTE-Sim Simulator) was transmitted to them. Some of the parameters used for this simulation are listed in the table below.

Number of UEs	30 (randomly distributed in the cell) + 1 (node under consideration)
Transmission Radius of Each Cell	2 Kilometers
Downlink Bandwidth	20 MHz
Number of Usable PRBs	100
Video Used	old_town
Resolution	HD (1280 x 720)
Frame Rate	50
GoP (Group of Pictures)	G16B15
Number of Frames	320
Downlink Scheduler	Exponential Fair Scheduler
Speed of UE	0 km /hr.
Frame Structure	FDD
Maximum Delay	400 milli-seconds
Flow Duration	20 seconds
Simulation Duration	25 seconds
Path Loss Model	Macro Cell Urban Area
Distance of the user from BS	Variable

Table 12: Some important parameters used for the simulation

	PSNR of the lossless video	Distance of the User from Base Station				
		200 m	400 m	600 m	800 m	1000 m
avc_ 810 kbps	28.62	28.62	28.62	28.62	28.62	28.62
avc_1072 kbps	30.29	30.29	30.29	30.29	30.29	29.82
avc_1444 kbps	32.72	32.72	32.72	32.72	32.72	27.98
avc_1900 kbps	33.98	33.98	33.98	33.71	31.34	27.66
avc_3314 kbps	35.76	35.76	35.76	31.12	27.84	24.69
hevc_234 kbps	28.34	28.34	28.34	28.34	28.34	28.34
hevc_468 kbps	30.48	30.48	30.48	30.48	30.48	30.48
hevc_840 kbps	32.77	32.77	32.77	32.77	32.77	32.77
hevc_1242 kbps	34.49	34.49	34.49	34.49	34.49	33.53
hevc_1844 kbps	35.92	35.92	35.92	35.92	32.65	30.04

Table 13: This table shows the PSNR values of the videos received at various distances.

When the user equipment was placed at a distance of 200 meters and then 400 meters from the base station, no loss of the frames was reported for transmission of any video (in case of both H.264/AVC and H.265 Video Standard). **This is due to the fact that when a user is close to the base station, quality of signal is good and thus Signal –to-Noise ratio is comparatively high. Therefore, data from the base station can be transmitted at a modulation and coding rate that can support higher rate of data transmission**. As a result, loss-less transmission of the videos encoded at high bit rates can be supported. Since there is no loss of frames reported, comparison of the PSNR value of the videos received at these distance (shown in figure 16) is replica of the original rate-distortion curve that was plotted in figure 8.

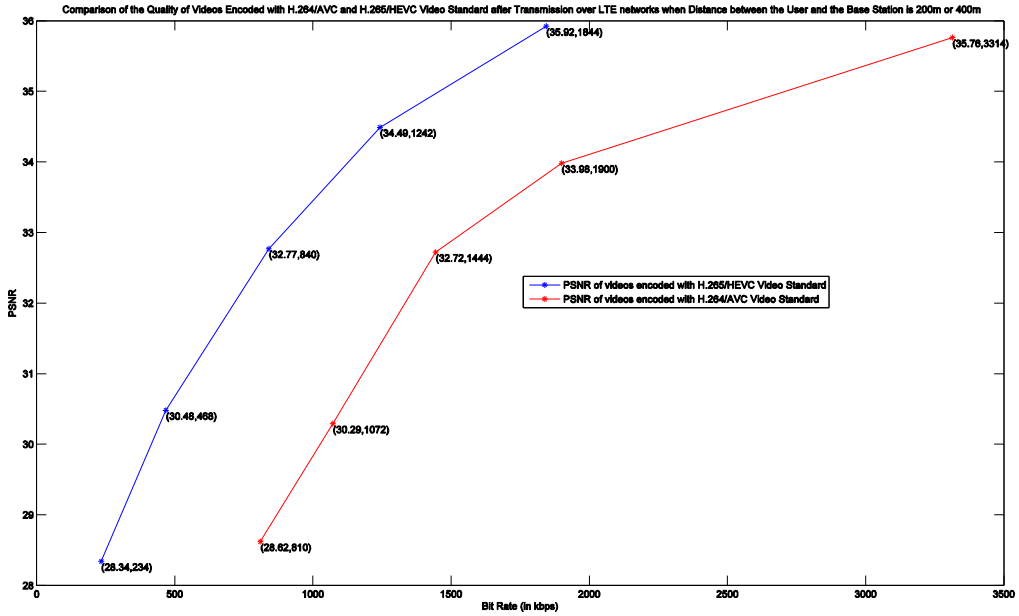


Figure 16: Comparison of PSNR of the received videos (encoded with standards H.264/AVC and HEVC) when distance between the user and base station is 200 meters or 400 meters.

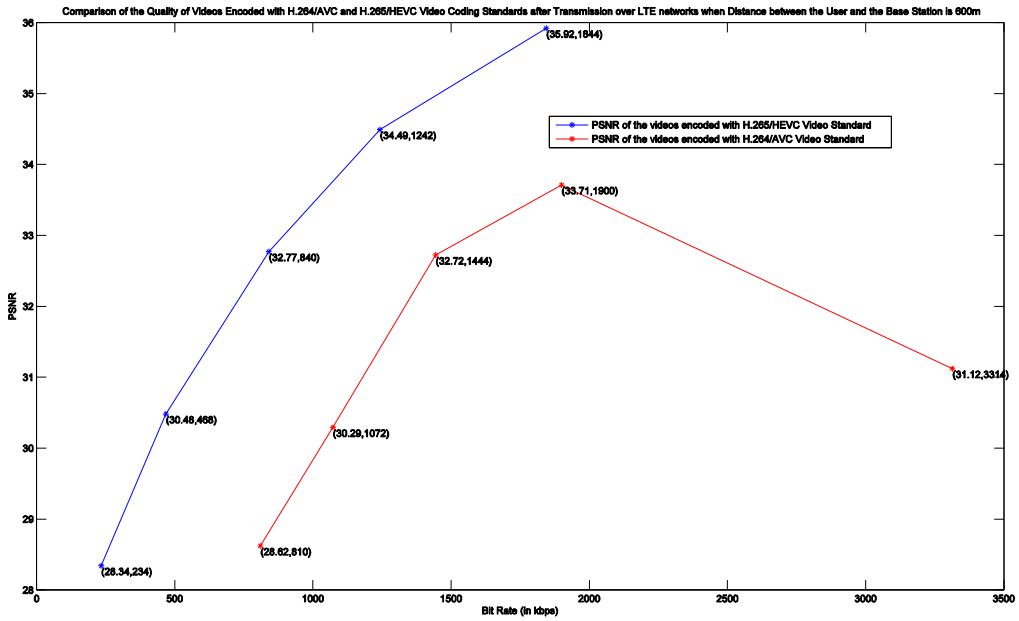


Figure 17: Comparison of PSNR of the received videos (encoded with standards H.264/AVC and HEVC) when distance between the user and base station is 600m

When the user equipment was placed at a distance of 600 meters from the base station, no loss of the frames was reported for transmission of videos with bit rates up to 1844 kbps. Thus for H.264/AVC videos encoded with bit rates less than 1844 kbps and all the HEVC videos, no loss in quality of video is observed at this distance.

However, for the videos encoded with the higher bit rates, quality of the video received starts deteriorating at this distance due inability of the LTE system to support higher bit rate (signal quality at this distance is medium). Thus, some frames are lost for videos encoded with higher bit rate. AVC video encoded with the bit rate of 3314 kbps, which was originally encoded to provide video quality of PSNR value of 35.76 degrades to the value of 31.12 and the quality of video received is even less than the quality of the H.264/AVC videos encoded with the bit rates of 1900 kbps and 1444 kbps. H.264/AVC video with the bit rate of 1900 kbps also encountered losses of some frames and thus PSNR value comes down slightly to 33.71 from lossless value of 33.98.

Comparison of PSNR of the received videos (encoded with standards H.264/AVC and HEVC) when distance between the user and base station is 600m is shown in figure 17.

When the user equipment was placed at a distance of 800 meters from the base station, no loss of the frame was reported for transmission of the videos with bit rates up to 1444kbps. Thus, similar trends are observed that we started observing when the distance was 600 meters. However, frame loss at this distance is much severe and thus decrease in PSNR value for the videos encoded with bit rates of more than 1444 kbps is much higher than it was when the distance was 600 meters. PSNR value of the H.264/AVC video encoded with the bit rate of 3314 kbps and original PSNR of 35.76 drops to 27.84, which is even lower than the case of H.264/AVC video encoded with 810 kbps. Although, HEVC video with bit rate 1844 kbps and original PSNR of 35. 92 performs better than its AVC counterpart (here counterpart means videos encoded with approximately same video quality and not similar bit rate), it too deteriorates to 32.65, which is lower quality than quality offered by HEVC videos encoded at bit rate of 1244 kbps and 840 kbps.

An interesting result is observed here. As discussed above, HEVC video with bit rate of 1844 kbps and original PSNR value of 35.92 deteriorates to 32.65 at this distance (which is decrease of 9.10 %). On the other hand, AVC video with bit rate of 1900 kbps and originally encoded with PSNR value of 33.98 deteriorates to 31.34 (which is a decrease of 7.76 %). Although PSNR value of the video received is more for HEVC video, it seems that effect of distance in this case is more pronounced for HEVC video (as percentage decrease in PSNR value of the received video is more in case of HEVC), although it is encoded at slightly lower bit rate. After looking at the output of the LTE-SIM Simulator for these cases, the reason seems to be the ratio of the sizes of the I-frames to that of B-frames. On analysing the output of LTE-SIM for these traces, 16 consecutive frames were lost (including I frame) at a particular section of the output for the HEVC video and only 15 frames were lost for corresponding section in case of AVC (I frame was not lost). Assuming channel conditions when both these traces were transmitted are same (since distance of the user from the base station in both the cases is the same and same seed is used for the simulation), we can assume that same amount of data that can be sent to the UE for this section of video transmission. In case of HEVC video, this Group of Pictures (all 16 frames) had combined size of 77115 bytes and size of the I-frame was 63839 bytes. For AVC on the

other hand, although total size for this Group of Pictures (all 16 frames) was 81392 bytes, size of the I-frame was 50053 bytes (which is was less than that of HEVC video). Further from the output trace of the LTE-Sim Simulation, it was observed that base station had capacity to transmit 52150 bytes for this flow when this particular GoP arrived at base station for the transmission. Due to this, I-frame of AVC video got transmitted and rest of the 15 frames were dropped. However, all 16 frames were lost in case of HEVC video, since size of I frame is more than the transmission capacity of base station for this flow at that particular instant. Since in our study, we assume a frame to be received correctly only when all the bits of the frame are received correctly, this partially received I-frame is considered lost in the case of HEVC video. This loss of I-frame frame can be thus be blamed on the higher ratio of I-frame size to B-frame size in case of HEVC encoding standard in comparison to H.264/AVC encoding standard (Note : similar configurations were used for encoding the videos of both video standards).

Comparison of the PSNR of the received videos (encoded with standards H.264/AVC and HEVC) when distance between the user and base station is 800m is shown in figure 18.

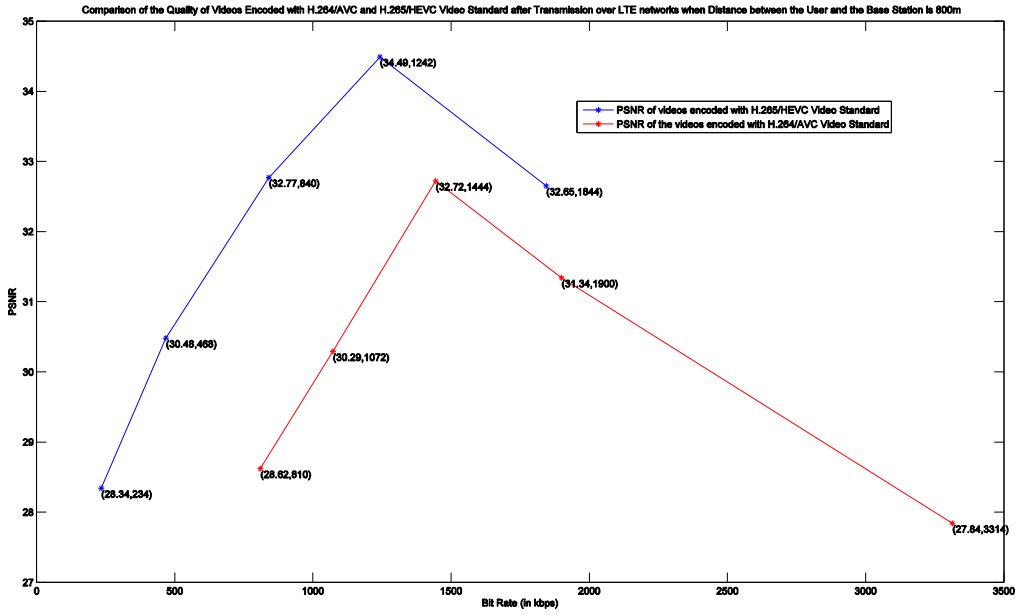


Figure 18: Comparison of PSNR of the received videos (encoded with standards H.264 AVC and HEVC) when distance between the user and base station is 800m

When the user equipment was placed at a distance of 1000 meters from the base station, no loss of the frame was reported for transmission of videos with bit rates up to 840 kbps. In case of H.264/AVC videos at this distance, it is observed that videos encoded with higher bit rates show lower quality at the receiving end. AVC video with bit rate 3314 kbps and PSNR value of 35.76 deteriorates to 24.69 at the receiver side, due to excessive loss of frames. Similarly, AVC video with bit rate 1900 kbps deteriorates from PSNR value of 33.98 to 27.66. There is slight decrease in the PSNR value of AVC video encoded with bit rate of 1072 kbps from 30.29 to 29.82.

HEVC video encoded with the bit rate 1844 kbps deteriorates due to loss of large number of frames, from the PSNR value of 35.92 to 30.04, which is even lower than the HEVC video encoded with the bit rate of 468 kbps. Slight decrease in PSNR of the HEVC video with bit rate 1242kbps from 34.49 to 33.53 is also observed.

Comparison of PSNR of the received videos (encoded with standards H.264/AVC and H.265/HEVC) for the case when distance between the user and base station is 1000m is shown in figure 19.

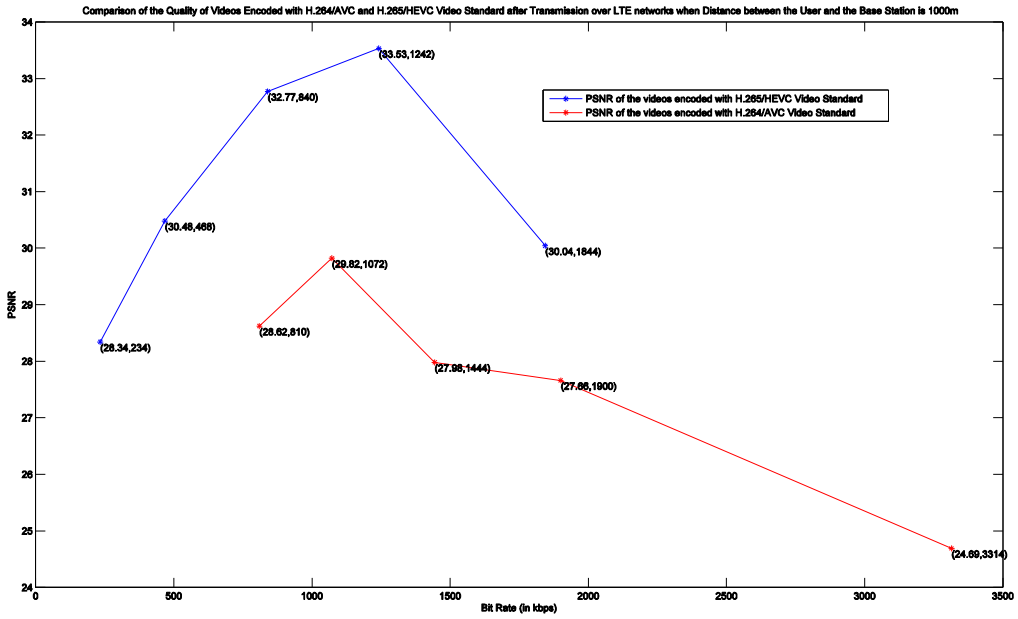


Figure 19: Comparison of PSNR of the received videos (encoded with standards H.264 AVC and HEVC) when distance between the user and base station is 1000m

6.5: Simulation 4 - Performance Evaluation of Frame Skip Mechanism

In order to evaluate the effectiveness of the proposed frame skip mechanism, we transmitted a HD video encoded with bitrate of 1072 kbps and video quality of PSNR value 30.28 to a stationary user at a distance of 1200m from the base station. In order to constrain the bandwidth, we gradually increased the number of users in the cell in steps of one and compared the performance of the proposed frame skip mechanism (that skips less important frames in the hierarchical structure when channel conditions for a particular user are poor) with the normal video transmission (that transmits all the frames) on three parameters: video quality received (PSNR value of the video received) by the user, Packet Loss Ratio (PLR) of the cell and Spectral Efficiency achieved in the cell. Some of the important parameters used for this simulation are listed in the table below.

Number of UEs	Variable
Transmission Radius of the Cell	2 Kilometers
Downlink Bandwidth	5 MHz
Number of Usable PRBs	25
Video Used	old_town
Resolution	HD (1280 x 720)
Frame Rate	50
GoP (Group of Pictures)	G16B15
Number of Frames	320
Downlink Scheduler	Exponential Fair Scheduler
Speed of UE	0 km /hr
Frame Structure	FDD
Maximum Delay	500 ms
Simulation Duration	25 seconds
Distance Between Base Station and User	1200 meters
Delay	300 ms
Path Loss Model	Macro Cell Urban Area

TABLE 14: Important Simulation Parameters

Result 6.5.1: Comparison of the PSNR of the video received through Normal Video Transmission and Frame Skip Mechanism with respect to Number of Users

Results in figure 20 show that when the number of users in cell are less, PSNR value of the video received by user using traditional transmission mechanism is better than the quality of the video received by the user using proposed frame skip mechanism. This is because, although CQI sent by the user back to the base station indicates that channel quality for the user is poor, bandwidth available at base station for the user is still high (when number of users are less, bandwidth resources are not constrained). However, in the frame skip mechanism that we employ, frames are skipped depending on the CQI value sent by the user to the base station. Since bandwidth is not considered in this mechanism, bandwidth resources in the case of transmission with the frame skip mechanism are under-constrained when number of users are less. However, when the number of users in the simulation are increased, quality of the video received by the user is better when frame skip mechanism employed in comparison to the case when traditional transmission scheme is employed. The difference in PSNR values of video received between two cases increases as number of users increase, showing effectiveness of the frame skip mechanism when the channel conditions are mediocre and bandwidth resources are constrained.

This better video quality in case of frame skip mechanism can be attributed to the two reasons. First, the frame skip mechanism degrades the video gracefully by skipping the less important frames. Although traditional video transmission may have better throughput for a particular user, but frames of the higher temporal layer may be transmitted at the cost of frames of lower temporal layer. Secondly, when the channel conditions are mediocre, frame skip mechanism avoids the accumulation of large queue for the video flow, which can lead to dropping of frames in the traditional video transmission.

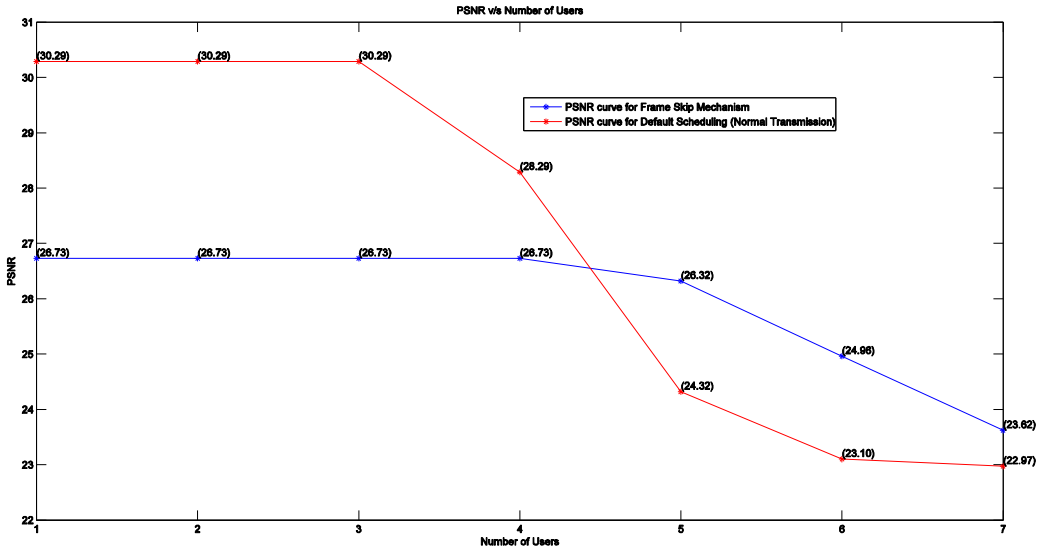


Figure 20: Comparison of quality of video received by Normal Video Transmission and Frame Skip Mechanism with respect to varying number of users

Number of Users	1	2	3	4	5	6	7
PSNR (Frame Skip Mechanism)	26.73	26.73	26.73	26.73	26.32	24.96	23.62
PSNR (Normal Video Transmission)	30.29	30.29	30.29	28.29	24.32	23.1	22.97

Table 15: Comparison of quality of video received by Normal Video Transmission and Frame Skip Mechanism with respect to varying number of users

Result 6.5.2: Packet Loss Ratio of the Cell versus Number of Users in the Cell

Packet Loss Ratio (PLR) for the whole cell (packet loss for all users in the cell) for the two mechanisms under consideration was compared next. Figure 21 shows the curves that display the comparison of PLR between the two mechanisms. Table 16 shows the corresponding data.

Performance of proposed frame skip mechanism is appreciably better than traditional video transmission method when bandwidth resources become constrained. This shows that proposed frame skip mechanism can not only provide better video quality to the individual users that face poor channel conditions, but can also help in improving performance of the LTE scheduler by vacating bandwidth resources for the use by the other users, thus reducing packet loss ratio of the cell, in general.

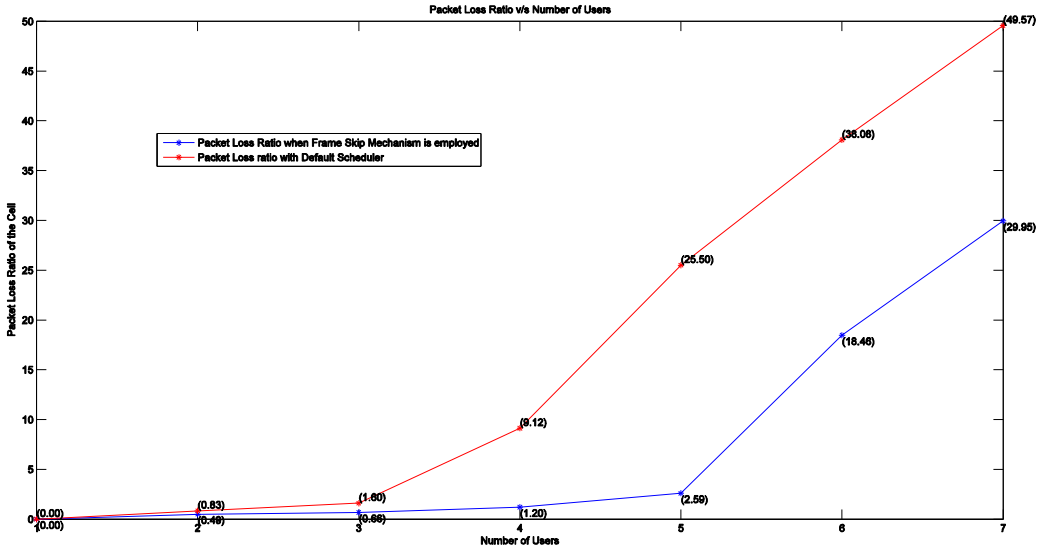


Figure 21: Comparison of Packet Loss Ratio in the LTE cell when using Normal Video Transmission and Frame Skip Mechanism with respect to varying number of users

Number of Users	Packet Loss Ratio (Frame Skip Mechanism)	Packet Loss Ratio (Default)
1	0	0
2	0.49	0.83
3	0.66	1.6
4	1.2	9.12
5	2.59	25.5
6	18.46	38.08
7	29.95	49.57

Table 16: Comparison of Packet Loss Ratio in the LTE cell when using Normal Video Transmission and Frame Skip Mechanism with respect to varying number of users

Result 6.5.3: Spectral Efficiency of the Cell with respect to the Number of Users

Figure 22 shows the comparison the spectral efficiency achieved by eNodeB (Base Station) of the cell for the two cases (frame skip mechanism and traditional video transmission). Data corresponding to the figure is given in table 17. Results show that spectral efficiency is lower for the frame skip mechanism than tradition video transmission mechanism. This is not a negative result, as word 'efficiency' can be a misnomer in our case. In general, spectral efficiency is used to compare the efficiency of two schedulers. However, since same (EXP- PF Scheduler) scheduler is being used for both the cases, lower spectral efficiency in our analysis only verifies that less bandwidth resources are consumed in case when frame skip mechanism is employed for video transmission.

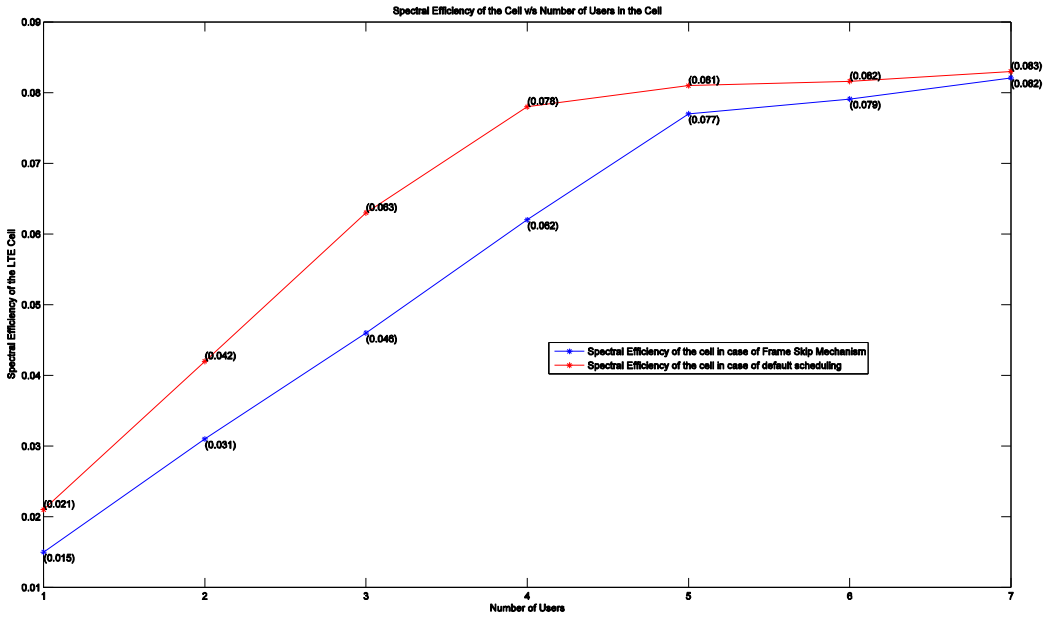


Figure 22: Comparison of Spectral Efficiency of the Cell versus Number of Users in the Cell in the cases when Normal Video Transmission is done and when Frame Skip Mechanism is employed

Number of Users	Spectral Efficiency (Normal Video Transmission)	Spectral Efficiency (Frame Skip)
1	0.021	0.015
2	0.042	0.031
3	0.063	0.046
4	0.078	0.062
5	0.081	0.077
6	0.0816	0.0791
7	0.083	0.0812

Table 17: Comparison of Spectral Efficiency of the Cell versus Number of Users in the Cell in cases when Normal Video Transmission is done and in case of Frame Skip.

Conclusion

In this project, we successfully compared the performance of the H.265/HEVC and H.264/AVC video coding standards over LTE networks through simulations. Performances of the videos conforming to these standards were evaluated by varying distances between the user and the base station, varying maximum delay for the packets and number of users. Results of the simulations prove that the videos encoded with HEVC standard provide better video quality over the LTE networks than the corresponding videos encoded with H.264/AVC standard for similar levels of video quality. Results also verify that bandwidth required to transmit videos encoded with H.265 video standard require comparatively less bandwidth resources than videos conforming to H.264/AVC video standard. It is also observed that when signal quality received by the LTE user is poor or number of users are large (and thus the bandwidth is constrained), videos encoded with lower bit rates can provide better video quality than videos encoded with higher bit rates.

Results of the second part of this project (frame skip mechanism) show that proposed frame skip mechanism provides better video quality than traditional video transmission when channel conditions are poor and bandwidth resource are constrained, but when number of users are less and bandwidth is not constrained, frame skip mechanism under performs. However, this drawback of frame skip mechanism can be improved if criteria for skipping the frames is optimized to include parameters such as available bandwidth, maximum delay allowed (before video packet is discarded) and length of the queue for a particular video flow.

Also, frame skip mechanism was found to perform better when evaluated on the basis of packet loss ratio and spectral efficiency of the LTE cell, indicating that the frame skip mechanism can not only provide better quality video to the users experiencing mediocre channel quality, but also release bandwidth for use by the other users in the cell.

References

[1] Talukdar, Anup, Mark Cudak, and Amitava Ghosh. "Streaming video capacities of LTE air-interface." In *Communications (ICC), 2010 IEEE International Conference on*, pp. 1-5. IEEE, 2010.

[2] Pande, Amit, Vishwanath Ramamurthi, and Prasant Mohapatra. "Quality-Oriented Video Delivery over LTE." *Journal of Computing Science and Engineering* 7.3 (2013): 168-176.

[3]Dahlman, E. Parkvall, S. Sköld, 1. J. & Beming, , 3G "Evolution: HSPA and LTE for Mobile Broadband". 3rd ed, Oxford, Academic Press, 2007.

[4] Motorola, Long Term Evolution. "A technical overview." *Technical White Paper*(2007).

[5]Rohde & Schwarz. "Rohde & Schwarz LTE Basics Webinar "Online video clip. *YouTube*. YouTube, Jan 28, 2010. Web. 3 January. 2015.

[6] J.-G. Choi and S. Bahk, "Cell-throughput analysis of the proportional fair scheduler in the single-cell environment," *IEEE Trans. Veh. Technol.*, vol. 56, no. 2, pp. 766 –778, Mar. 2007.

[7] R. Basukala, H. M. Ramli, and K. Sandrasegaran, "Performance analysis of EXP/PF and M-LWDF in downlink 3GPP LTE system," in *Proc. of First Asian Himalayas Int. Conf. on Internet. AH-ICI*, Kathmandu, Nepal, Nov. 2009.

[8]. "Global Internet Phenomena Report." [online] [Accessed: 20 Dec 2014] https://www.sandvine.com/downloads/general/global-internet-phenomena/2013/sandvine-global-internet-phenomena-report-1h-2013.pdf, Sandvine, pg 8, 2013.

[9]. Schwarz, Heiko, Detlev Marpe, and Thomas Wiegand. "Overview of the scalable H. 264/MPEG4-AVC extension." *Image Processing, 2006 IEEE International Conference on*. IEEE, 2006.

[10]. Wiegand, Thomas, et al. "Overview of the H. 264/AVC video coding standard."*Circuits and Systems for Video Technology, IEEE Transactions on* 13.7 (2003): 560-576

[11] Seeling, Patrick, and Martin Reisslein. "Video Traffic Characteristics of Modern Encoding Standards: H. 264/AVC with SVC and MVC Extensions and H. 265/HEVC." *The Scientific World Journal* 2014 (2014

[12] Sullivan, Gary J., et al. "Overview of the high efficiency video coding (HEVC) standard." *Circuits and Systems for Video Technology, IEEE Transactions on*22.12 (2012): 1649-1668.

[13] "Xiph.org Video Test Media [derf's collection] "[online] [Accessed: 20 Sept 2014] https://media.xiph.org/video/derf/

[14] Piro, Giuseppe, et al. "Simulating LTE cellular systems: An open-source framework." *Vehicular Technology, IEEE Transactions on* 60.2 (2011): 498-513.

[15] Radhakrishnan, Rakesh, and Amiya Nayak. "Cross layer design for efficient video streaming over LTE using scalable video coding." *Communications (ICC), 2012 IEEE International Conference on*. IEEE, 2012.

Appendix A.

HM Code Source Code

```
Bool TDecTop::xDecodeSlice(InputNALUnit &nalu, Int &iSkipFrame, Int iPOCLastDisplay )


{

 // My code Starts   part 1

  bool flag = false ;

  //  if value of this flag is true , decoding for that picture is skipped ; it is copy
       of earlier frame in the dynamic buffer is made

 static bool input_flag  = true ;
 //  flag to ensure that input of lost frames from the console screen is inputted just
       once
 static vector<int> lostframes;


  // this loop takes helps create an vector that takes the POC numbers of lost frames
during network transmission

 if (input_flag == true)
  {
       int input;

       do{

        cout << "Enter lost frames number or number greater than 50,000 to continue with
                    decoding : " << endl;
        cin >> input;
        if ( input < 50000)
        {
              lostframes.push_back(input);
        }

       } while(input < 50000);


  input_flag = false ;

  }

// my code ends   Part 1
```

```
//mycode starts Part 2

 vector<int>::iterator it = lostframes.begin();

 /*
 this loop checks whether the compares of current POC of the current PIC being decoded
with values we have inputted as lost frames during transmission and sets the flag = true;
if condition current flag needs   */

 while( it != lostframes.end())
 {

              if (m_pcPic->getPOC()==(*it))
              {

                     flag = true ;

              }


              it++;


 }


// this piece of code copies the nearest frame in the dynamic buffer to the frame that we
input as lost frame;
// this code is modified code from already existing code in HM software (but that does
not serve our purpose)

  if (flag ==  true  )
  {
       TComList<TComPic*>::iterator iterPic1 = m_cListPic.begin();
  Int closestPoc = 1000000;



  while ( iterPic1 != m_cListPic.end())
  {
    TComPic * rpcPic = *(iterPic1);


    if(abs(rpcPic->getPicSym()->getSlice(0)->getPOC()-pcSlice-
>getPOC())<closestPoc&&abs(rpcPic->getPicSym()->getSlice(0)->getPOC() - pcSlice-
>getPOC())!=0&&rpcPic->getPicSym()->getSlice(0)->getPOC()!=pcSlice->getPOC())
    {
      closestPoc=abs(rpcPic->getPicSym()->getSlice(0)->getPOC()- pcSlice->getPOC());
//iLostPoc

    }
```

```
      iterPic1++;

  }



  iterPic1 = m_cListPic.begin();
  while ( iterPic1 != m_cListPic.end())
  {

      TComPic *rpcPic = *(iterPic1++);


    if(abs(rpcPic->getPicSym()->getSlice(0)->getPOC() -pcSlice-
>getPOC())==closestPoc&&rpcPic->getPicSym()->getSlice(0)->getPOC()!=pcSlice->getPOC())
    {
      printf("copying picture %d to %d \n",rpcPic->getPicSym()->getSlice(0)->getPOC()
,m_pcPic->getPOC());
      rpcPic->getPicYuvRec()->copyToPic(m_pcPic->getPicYuvRec());

         m_pcPic->setOutputMark(true);

      m_pcPic->setReconMark(true);



         TComSlice::sortPicList( m_cListPic ); // sorting for application output
       m_cCuDecoder.destroy();

      return false;
    }
  }
  }



  // Decode a picture if PIC is not inputted as lost picture during transmission process

 else
  {
          m_cGopDecoder.decompressSlice(nalu.m_Bitstream, m_pcPic);
  }


  // my code ends part 2
```

Appendix B.

Code Testing and Validation

In order to validate the code developed for implementing frame copy (after implementing code for skipping the frame in bit stream) in HM Reference Software, first 9 frames of foreman video sequence were encoded and frames were selectively dropped from it and PSNR value of the resultant decoded videos were obtained.

Sequence Used for Test:

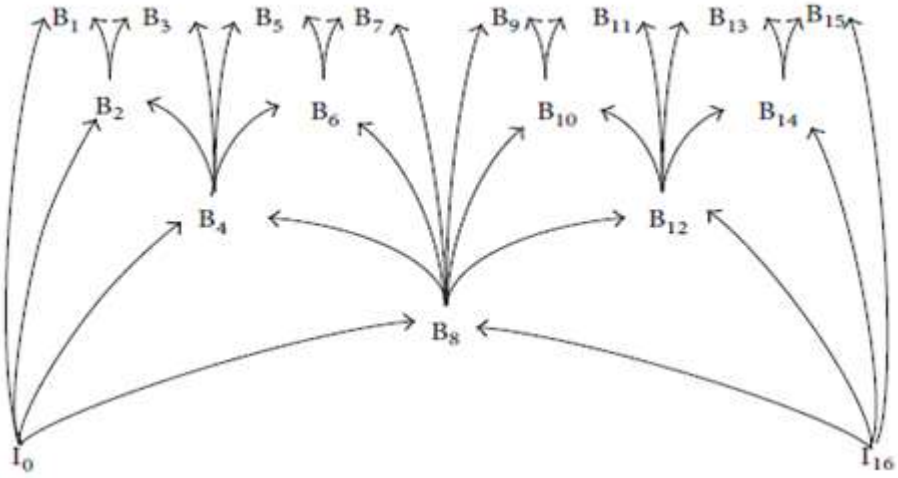
Sequence Name: Foreman

Number of Frames: 9

Format: CIF

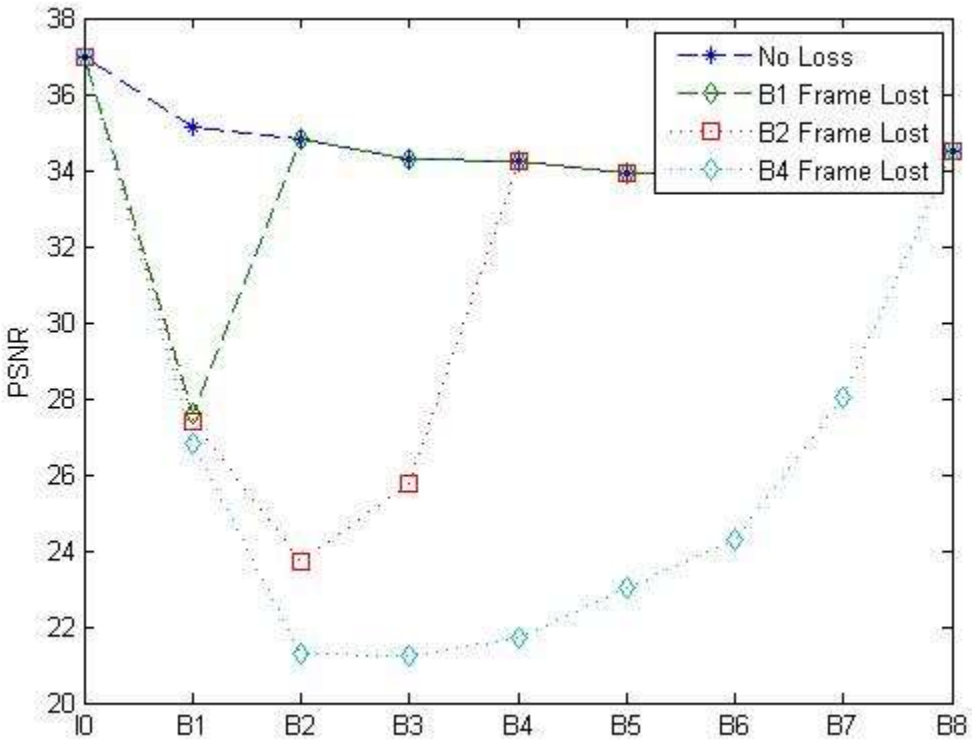
GOP Size: 16

Following figure shows the coding structure used.



Validation Test and Results

For test, we encoded 9 frames and dropped the frames B1, B2 and B4 and PSNR value of all 9 frames was compared with lossless sequence. Results are as expected, since loss of frame in lower level temporal layer also affects dependent frames in higher temporal layers. Loss of B1 frame effects just PSNR value of B1 frame since it is in highest temporal level and has no dependent frame. On the other hand, loss of B4 frame affects the PSNR value of all the other frames except I0 and B8 frame since they are in lower temporal layer than B4 itself.



Related Data

Frame Number	PSNR Value with Lost Frame	PSNR Value when B1 is lost	PSNR Value when B2 is lost	PSNR Value when B4 is lost
I0	36.99	36.99	36.99	36.99
B1	35.13	27.6	27.39	26.8
B2	34.82	34.82	23.73	21.3
B3	34.27	34.27	25.76	21.22
B4	34.24	34.24	34.24	21.73
B5	33.9	33.9	33.9	23.05
B6	33.87	33.87	33.87	24.3
B7	33.96	33.96	33.96	28.04
B8	34.5	34.5	34.5	34.5

- Frames Effected

Appendix C.

Source Code for Frame Skip Mechanism

Modified TraceBased.h file

```
#ifndef TRACEBASED_H_
#define TRACEBASED_H_

#include "Application.h"

#include <cstdio>
#include <iostream>
#include <fstream>

//mycode
#include "../../device/ENodeB.h"


/*
 * This application sends udp packets based on a trace file could be downloaded form :
 * http://www.tkn.tu-berlin.de/research/trace/ltvt.html
 * A valid trace file is a file with 4 columns:
 * - the first one represents the frame index
 * - the second one indicates the type of the frame: I, P or B
 * - the third one indicates the time on which the frame was generated by the encoder
* - the fourth one indicates the frame size in byte
 * if no valid trace trace file is provided to the application the trace from
 * default-trace.h will be loaded.
 */

class TraceBased : public Application {
public:
       ENodeB *enb;
       TraceBased();
       TraceBased(ENodeB** pointer_enb);
       TraceBased(NetworkNode *source,
                       NetworkNode *destination,
                       int sourcePort,
                       int destinationPort,
                       TransportProtocol::TransportProtocolType protocol);

       virtual ~TraceBased();

    void SetTraceFile(std::string traceFile);
    void LoadTrace (std::string traceFile);
    void LoadDefaultTrace (void);

       virtual void DoStart (void);
       virtual void DoStop (void);
```

```
    void ScheduleTransmit (double time);
    void Send (void);

    void UpdateFrameCounter (void);
    int GetFrameCounter (void);

    void PrintTrace (void);

private:

    uint32_t m_TraceSize; //in packet
    double m_interval;
    uint32_t m_size;
    uint32_t m_sent;

    typedef struct
    {
      uint32_t TimeToSend;
      uint16_t PacketSize;
      uint32_t FrameIndex;
      char FrameType;
    } TraceEntry;

    std::vector<TraceEntry> * m_entries;
    std::vector<TraceEntry>::iterator iter;

    int m_frameCounter;
};

#endif /* TRACEBASED_H_ */
```

Modified TraceBased.cpp

```
/*
TraceBased File for Frame Skip Mechanism

Modified by Ravneet Sohi
sohi@sfu.ca
301219135
 */

#include "TraceBased.h"
#include "Trace/default-trace.h"
#include <cstring>
#include "../../componentManagers/NetworkManager.h"
#include "../radio-bearer.h"

#define MAXMTUSIZE 1490

TraceBased::TraceBased()
{
  cout << " in trace based constructor "   << endl ;
  m_sent = 0;
  m_entries = new std::vector<TraceEntry> ();
  LoadDefaultTrace ();
  m_frameCounter = 0;
  SetApplicationType (Application::APPLICATION_TYPE_TRACE_BASED);
}

TraceBased::TraceBased(ENodeB** pointer_enb)
{
 // ENodeB
       enb = *(pointer_enb);
  cout << "address of enb in Tracebased" << enb << endl;
  cout << " // in trace based constructor  //"   << endl ;
//  ENodeB::UserEquipmentRecord *ueRecord = enb->GetUserEquipmentRecord
(GetRadioBearer()->GetDestination  ()->GetIDNetworkNode ());
//  std::vector<int> cqiFeedbacks = ueRecord->GetCQI ();
  m_sent = 0;
  m_entries = new std::vector<TraceEntry> ();
  LoadDefaultTrace ();
  m_frameCounter = 0;
  SetApplicationType (Application::APPLICATION_TYPE_TRACE_BASED);
}


TraceBased::~TraceBased()
{
  m_entries->clear ();
  delete m_entries;
  Destroy ();
}

void
TraceBased::DoStart (void)
{
  Simulator::Init()->Schedule(0.0, &TraceBased::Send, this);
```

```
}

void
TraceBased::DoStop (void)
{}

void
TraceBased::SetTraceFile(std::string traceFile)
{
  LoadTrace (traceFile);

#ifdef APPLICATION_DEBUG
  std::cout << " TRACE_BASED_DEBUG: Add file trace to Trace-Based "
               "Application Client.....  DONE" << std::endl;
#endif

}

void
TraceBased::LoadTrace (std::string traceFile)
{
  if (m_entries->size() > 0)
    {
      m_entries->clear ();
    }

  uint32_t time, index, prevTime = 0;
  uint16_t size;
  char frameType;
  TraceEntry entry;
  std::ifstream ifTraceFile;
  ifTraceFile.open (traceFile.c_str (), std::ifstream::in);

  if (!ifTraceFile.good ())
    {
        std::cout << " TRACE_BASED_APPLICATION ERROR BAD FILE"<< std::endl;
      LoadDefaultTrace ();
    }
  while (ifTraceFile.good ())
    {
      ifTraceFile >> index >> frameType >> time >> size;
      if (frameType == 'B' || frameType == 'PB')
        {
          entry.TimeToSend = 0;
        }
      else
        {
          entry.TimeToSend = time - prevTime;
          prevTime = time;
        }
      entry.PacketSize = size;
      entry.FrameIndex = index;
      entry.FrameType = frameType;
      m_entries->push_back (entry);
    }
  ifTraceFile.close ();
```

```
  iter = m_entries->begin ();
}

void
TraceBased::LoadDefaultTrace (void)
{
  uint32_t time, index, prevTime = 0;
  uint16_t size;
  char frameType;
  TraceEntry entry;
  for (int i = 0; i < NB_FRAME; i++)
    {
      index = i + 1;
      frameType = frameType_Tab[i];
      time = time_Tab[i];
      size = size_Tab[i];

      if (frameType == 'B')
        {
          entry.TimeToSend = 0;
        }
      else
        {
          entry.TimeToSend = time - prevTime;
          prevTime = time;
        }
      entry.PacketSize = size;
      entry.FrameIndex = index;
      entry.FrameType = frameType;
      m_entries->push_back (entry);
    }
  iter = m_entries->begin ();

}

void
TraceBased::ScheduleTransmit (double time)
{
  if ( (Simulator::Init()->Now () + time) < GetStopTime () )
    {
      Simulator::Init()->Schedule(time, &TraceBased::Send, this);
    }
}

void
TraceBased::Send (void)
{
  ENodeB::UserEquipmentRecord *ueRecord = enb->GetUserEquipmentRecord (GetRadioBearer()-
>GetDestination  ()->GetIDNetworkNode ());
  std::vector<int> cqiFeedbacks = ueRecord->GetCQI ();
  int numberOfCqi = cqiFeedbacks.size ();
  cout << " number of CQIs = "  << numberOfCqi << endl;
  int cqi_level_for_server = cqiFeedbacks.at (0);
  cout << " value of CQI at " << cqi_level_for_server ;

  int dataOfFrameAlreadySent = 0;
```

```
static double average_CQI =10;
static int s=0 ;
cout << " s= " << s << endl;//estimated_CQI = (0.8 *cqi_level_for_server )+
(0.2*estimated_CQI);
average_CQI = ((cqi_level_for_server )+ (s*average_CQI))/(s+1);
s++ ;
//cout << "estimated CQI = " << estimated_CQI << endl;
cout << " average CQI = " << average_CQI << endl;

cout << " Destination ID = " << GetRadioBearer()->GetDestination  ()->GetIDNetworkNode ()
;


  while (true)
    {
         // base layer + layer 1 +layer 2 +layer 3
         if (cqi_level_for_server > 0 && cqi_level_for_server < 10 )
          {
               if (iter->FrameIndex % 2 == 0 )
                  {
                for (int i = 0; i < (iter->PacketSize) / MAXMTUSIZE; i++)
                      {
                            //CREATE A NEW PACKET (ADDING UDP, IP and PDCP HEADERS)
                            Packet *packet = new Packet ();
                            int uid = Simulator::Init()->GetUID ();

                            packet->SetID(uid);
                            packet->SetTimeStamp (Simulator::Init()->Now ());
                            packet->SetSize (MAXMTUSIZE);

                            Trace (packet);

                            PacketTAGs *tags = new PacketTAGs ();
                            tags-
>SetApplicationType(PacketTAGs::APPLICATION_TYPE_TRACE_BASED);
                            tags->SetFrameNumber(GetFrameCounter());
                            tags->SetStartByte(dataOfFrameAlreadySent);
                            tags->SetEndByte(dataOfFrameAlreadySent+MAXMTUSIZE-1);
                            tags->SetApplicationSize (packet->GetSize ());
                            dataOfFrameAlreadySent+=MAXMTUSIZE;
                            packet->SetPacketTags(tags);


                            UDPHeader *udp = new UDPHeader (GetClassifierParameters ()-
>GetSourcePort (),
                                                            GetClassifierParameters
()->GetDestinationPort ());
                            packet->AddUDPHeader (udp);

                            IPHeader *ip = new IPHeader (GetClassifierParameters ()-
>GetSourceID (),
                                         GetClassifierParameters ()->GetDestinationID
());
                            packet->AddIPHeader (ip);
```

```
                                PDCPHeader *pdcp = new PDCPHeader ();
                                packet->AddPDCPHeader (pdcp);

                                GetRadioBearer()->Enqueue (packet);
                        }

                        uint16_t sizetosend = (iter->PacketSize) % MAXMTUSIZE;

                        if (sizetosend > 0)
                        {
                                //CREATE A NEW PACKET (ADDING UDP, IP and PDCP HEADERS)
                                Packet *packet = new Packet ();
                                int uid = Simulator::Init()->GetUID ();

                                packet->SetID(uid);
                                packet->SetTimeStamp (Simulator::Init()->Now ());
                                packet->SetSize (sizetosend);

                                Trace (packet);

                                PacketTAGs *tags = new PacketTAGs ();
                                tags-
>SetApplicationType(PacketTAGs::APPLICATION_TYPE_TRACE_BASED);
                                tags->SetFrameNumber(GetFrameCounter());
                                tags->SetStartByte(dataOfFrameAlreadySent);
                                tags->SetEndByte(dataOfFrameAlreadySent+sizetosend-1);
                                tags->SetApplicationSize (packet->GetSize ());
                                dataOfFrameAlreadySent+=sizetosend;
                                packet->SetPacketTags(tags);


                                UDPHeader *udp = new UDPHeader (GetClassifierParameters ()-
>GetSourcePort (),

GetClassifierParameters ()->GetDestinationPort ());
                                packet->AddUDPHeader (udp);

                                IPHeader *ip = new IPHeader (GetClassifierParameters ()-
>GetSourceID (),

GetClassifierParameters ()->GetDestinationID ());
                                packet->AddIPHeader (ip);

                                PDCPHeader *pdcp = new PDCPHeader ();
                                packet->AddPDCPHeader (pdcp);

                                GetRadioBearer()->Enqueue (packet);
                               }
                         }
                       else
                                  {
                                          cout << " Frame Number"  << iter->FrameIndex << "
is skipped " << " for B "<< ((GetRadioBearer()->GetDestination  ()->GetIDNetworkNode ())-
2) << endl;
                                  }
       }
```

```
        // Base Rate + layer 1 +layer 2 +layer 3+ layer 4
        else {

                for (int i = 0; i < (iter->PacketSize) / MAXMTUSIZE; i++)
                {
                        //CREATE A NEW PACKET (ADDING UDP, IP and PDCP HEADERS)
                        Packet *packet = new Packet ();
                        int uid = Simulator::Init()->GetUID ();

                        packet->SetID(uid);
                        packet->SetTimeStamp (Simulator::Init()->Now ());
                        packet->SetSize (MAXMTUSIZE);

                        Trace (packet);

                        PacketTAGs *tags = new PacketTAGs ();
                        tags-
>SetApplicationType(PacketTAGs::APPLICATION_TYPE_TRACE_BASED);
                        tags->SetFrameNumber(GetFrameCounter());
                        tags->SetStartByte(dataOfFrameAlreadySent);
                        tags->SetEndByte(dataOfFrameAlreadySent+MAXMTUSIZE-1);
                        tags->SetApplicationSize (packet->GetSize ());
                        dataOfFrameAlreadySent+=MAXMTUSIZE;
                        packet->SetPacketTags(tags);


                        UDPHeader *udp = new UDPHeader (GetClassifierParameters ()-
>GetSourcePort (),
                                                        GetClassifierParameters ()-
>GetDestinationPort ());
                        packet->AddUDPHeader (udp);

                        IPHeader *ip = new IPHeader (GetClassifierParameters ()-
>GetSourceID (),
                                        GetClassifierParameters ()->GetDestinationID ());
                        packet->AddIPHeader (ip);

                        PDCPHeader *pdcp = new PDCPHeader ();
                        packet->AddPDCPHeader (pdcp);

                        GetRadioBearer()->Enqueue (packet);
                }

                uint16_t sizetosend = (iter->PacketSize) % MAXMTUSIZE;

                if (sizetosend > 0)
                {
                        //CREATE A NEW PACKET (ADDING UDP, IP and PDCP HEADERS)
                        Packet *packet = new Packet ();
                        int uid = Simulator::Init()->GetUID ();

                        packet->SetID(uid);
                        packet->SetTimeStamp (Simulator::Init()->Now ());
                        packet->SetSize (sizetosend);
```

```
                        Trace (packet);

                        PacketTAGs *tags = new PacketTAGs ();
                        tags-
>SetApplicationType(PacketTAGs::APPLICATION_TYPE_TRACE_BASED);
                        tags->SetFrameNumber(GetFrameCounter());
                        tags->SetStartByte(dataOfFrameAlreadySent);
                        tags->SetEndByte(dataOfFrameAlreadySent+sizetosend-1);
                        tags->SetApplicationSize (packet->GetSize ());
                        dataOfFrameAlreadySent+=sizetosend;
                        packet->SetPacketTags(tags);


                        UDPHeader *udp = new UDPHeader (GetClassifierParameters ()-
>GetSourcePort (),

GetClassifierParameters ()->GetDestinationPort ());
                        packet->AddUDPHeader (udp);

                        IPHeader *ip = new IPHeader (GetClassifierParameters ()-
>GetSourceID (),
                                                                    GetClassifierParameters
()->GetDestinationID ());
                        packet->AddIPHeader (ip);

                        PDCPHeader *pdcp = new PDCPHeader ();
                        packet->AddPDCPHeader (pdcp);

                        GetRadioBearer()->Enqueue (packet);
                       }
         }   //else ends


      iter++;
         m_frameCounter++;
         dataOfFrameAlreadySent = 0;

         if (iter == m_entries->end ())
              {
                cout << " Trace Ends" << endl;

                iter = m_entries->begin ();
              }

         if (iter->TimeToSend != 0)
              {
                ScheduleTransmit ((iter->TimeToSend)*0.001);
                break;
              }
    }
}
```

```
void
TraceBased::UpdateFrameCounter (void)
{
  m_frameCounter++;
}

int
TraceBased::GetFrameCounter (void)
{
  return m_frameCounter;
}

void
TraceBased::PrintTrace(void)
{
  std::cout << "Print Trace "<< std::endl;
  std::vector<TraceEntry>::iterator it;

  int n=0;
  for (it = m_entries->begin(); it != m_entries->end(); it++)
    {
        std::cout << it->FrameIndex << " " <<it->FrameType
                    << " " << it->TimeToSend << " " << it->PacketSize << std::endl;
        n++;
        if (n==10)
               return;
    }
}
```

Appendix D.

LTE-SIM Simulator Software Bug Detection and Fixing

While working with latest version of LTE –SIM Simulator on this project, it was detected that LTE –SIM Simulator was unable to transmit certain video traces. After careful evaluation of the output, it was observed that simulator stopped transmitting trace at first occurrence of frame with size more than 65355 bytes (which is not uncommon for I-frames of HD sequences of high resolution).

After analyzing the software code of the simulator, it was found out that this happened because two variables, namely “size" in file tracebased.cpp and variable "packetsize " in file tracebased.h were defined as uint16_t (unsigned integer of 16 bits).

Thus, this problem was occurring because range supported by the data type ‘16-bit unsigned integer ‘.

Problem was solved by declaring these variables as unsigned integers of 32 bits instead unsigned integer of 16 bits.

Issue and solution has been brought to the notice of lead developer of the simulator.

DEMO of the Bug

```
Simulation with SEED = 5
Scheduler EXP
Created UE - id 2 position -1291.91 541.592
          selected video @ 128k
CREATED VIDEO APPLICATION, ID 0
TX VIDEO ID 0 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 1 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 2 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 3 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 4 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 5 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 6 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 7 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 8 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 9 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 10 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 11 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 12 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 13 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 14 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 15 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 16 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 17 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 18 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 19 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 20 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 21 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 22 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 23 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 24 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 25 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 26 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 27 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 28 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 29 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 30 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 31 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 32 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 33 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 34 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 35 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 36 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 37 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
```

```
TX VIDEO ID 38 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 39 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 40 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 41 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 42 B 0 SIZE 1490 SRC -1 DST 2 T 1.5 0
TX VIDEO ID 43 B 0 SIZE 1465 SRC -1 DST 2 T 1.5 0

TRACE ENDS     < - Trace Ends after transmitting just first frame
                   when transmitting traces for videos encoded
                    with H.264 AVC (Bit Rate – 3314)
                    and HEVC ( bit rate – 1844 kbps)
RX VIDEO ID 0 B 0 SIZE 1490 SRC -1 DST 2 D 0.006 0
RX VIDEO ID 1 B 0 SIZE 1490 SRC -1 DST 2 D 0.01 0
RX VIDEO ID 2 B 0 SIZE 1490 SRC -1 DST 2 D 0.015 0
RX VIDEO ID 3 B 0 SIZE 1490 SRC -1 DST 2 D 0.019 0
RX VIDEO ID 4 B 0 SIZE 1490 SRC -1 DST 2 D 0.024 0
RX VIDEO ID 5 B 0 SIZE 1490 SRC -1 DST 2 D 0.028 0
```

VARIABLE DECLERATION BUG IN FILE Tracebased.h

```
class TraceBased : public Application {
public:
      TraceBased();
      TraceBased(NetworkNode *source,
                    NetworkNode *destination,
                    int sourcePort,
                    int destinationPort,
                    TransportProtocol::TransportProtocolType protocol);

      virtual ~TraceBased();

    void SetTraceFile(std::string traceFile);
    void LoadTrace (std::string traceFile);
    void LoadDefaultTrace (void);

      virtual void DoStart (void);
      virtual void DoStop (void);

    void ScheduleTransmit (double time);
    void Send (void);

    void UpdateFrameCounter (void);
    int GetFrameCounter (void);

    void PrintTrace (void);

private:

    uint32_t m_TraceSize; //in packet
    double m_interval;
    uint32_t m_size;
    uint32_t m_sent;

    typedef struct
    {
      uint32_t TimeToSend;
      uint16_t PacketSize; < - " PacketSize " declared
                           as unsigned integer of 16 bits
                           causing a bug in the software
      uint32_t FrameIndex;
      char FrameType;
    } TraceEntry;

    std::vector<TraceEntry> * m_entries;
    std::vector<TraceEntry>::iterator iter;
    int m_frameCounter; }
```

VARIABLE DECLERATION BUG IN FILE TraceBased.cpp

```
#include "TraceBased.h"
#include "Trace/default-trace.h"
#include <cstring>
#include "../../componentManagers/NetworkManager.h"
#include "../radio-bearer.h"

#define MAXMTUSIZE 1490

TraceBased::TraceBased()
{
  m_sent = 0;
  m_entries = new std::vector<TraceEntry> ();
  LoadDefaultTrace ();
  m_frameCounter = 0;
  SetApplicationType (Application::APPLICATION_TYPE_TRACE_BASED);
}

TraceBased::~TraceBased()
{
  m_entries->clear ();
  delete m_entries;
  Destroy ();
}
```

```
void
TraceBased::LoadTrace (std::string traceFile)
{
  if (m_entries->size() > 0)
    {
      m_entries->clear ();
    }

  uint32_t time, index, prevTime = 0;
uint16_t size;              < - variable size declared as
                                unsigned integer of 16 bits causing software bug
  char frameType;
  TraceEntry entry;
  std::ifstream ifTraceFile;
  ifTraceFile.open (traceFile.c_str (), std::ifstream::in);

                    ....
                    ....
            ....
SOFTWARE CODE CONTINUES
```