QoE Evaluation for Video Streaming over eMBMS

Utsaw Kumar, Ozgur Oyman, and Apostolos Papathanassiou
Intel Corporation, Santa Clara, USA
Email: {utsaw.kumar; ozgur.oyman; apostolos.papathanassiou}@intel.com

Abstract—As the multicast standard for long term evolution (LTE), Enhanced multimedia broadcast multicast service (e-MBMS) was introduced by third generation partnership project (3GPP) to facilitate delivery of popular content to multiple users over a cellular network. If a large number of users are interested in the same content, for e.g., live sports, news clips, etc., multicast transmission can significantly lower the cost and make better use of the spectrum as compared to unicast transmission. In this paper, we study and analyze the quality of experience (QoE) at the end user during live video streaming over eMBMS. We consider a comprehensive end-to-end MBMS streaming frame-work based on H.264/AVC encoded video content delivered in a chunked format over multiple segments using the file delivery over unidirectional transport (FLUTE) protocol, combined with application layer (AL) forward error correction (FEC) based on Raptor and RaptorQ codes. Specifically, our study involves QoE evaluation in terms of startup delay, rebuffering percentage and peak signal-to-noise ratio (PSNR) metrics and provides performance evaluations to characterize the impact of various MBMS streaming, transport and AL-FEC configurations on the end user QoE. For simulating media access control-protocol data unit (MAC-PDU) losses, we propose a new Markov model and show that our model captures the coverage aspects of eMBMS in contrast to the RAN endorsed model, which assumes the same Markov model for all the users. We also propose a decoding strategy that takes into consideration the systematic structure of AL-FEC codes to enhance performance when a source block decoding fails.

Index Terms—multicast, e-MBMS, video streaming, ALFEC, QoE metrics

I. Introduction

Video communication over wireless is a unique and challenging problem, owing to the scarcity of bandwidth and the low-latency, high reliability constraints of video. The growth of multimedia applications and increased mobile internet access has created the need for enhancing video delivery over wireless systems. In fact, mobile video traffic has been projected to occupy more than two-thirds of the total mobile traffic by 2015 [1]. eMBMS allows multimedia content to be broadcasted and received by many users in a scalable fashion. Delivery of popular Youtube clips, live sports events, news updates, advertisements, file sharing, etc. are relevant use cases for eMBMS. This can be seen as a viable alternative to unicast when the same content is of interest to a large number of users. eMBMS would utilize the network bandwidth more efficiently by using the inherent broadcast nature of wireless channels.

Wireless channels are lossy and reliability mechanisms are implemented to make up for losses over the wireless media. For unicast transmissions, retransmissions based on Automatic Repeat Request (ARQ) and/or hybrid ARQ (HARQ) are used to ensure reliability. However, for a broadcast transmission, implementing ARQ can lead to network congestion with multiple users requesting different packets. Moreover, each user has a different channel to the base station owing to different distance from the transmitting station, user mobility, and attenuation characteristics. Thus, different users might lose different packets and retransmission could mean sending a large chunk of the original content again, leading to inefficient use of bandwidth as well as increased latency for some users. AL-FEC is an error correction mechanism in which redundant data is sent to facilitate recovery of lost packets. Such increased reliability through AL-FEC help to reliance on recovery procedures that involve retransmission of lost packets over broadcast. For this purpose, Raptor codes [2] were adopted in 3GPP TS 26.346 [7] as the AL-FEC scheme for MBMS delivery, and were also adopted in Internet Engineering Task Force (IETF) RFC 5053 [3]. Recently, improvements on the Raptor codes have been developed and an enhanced code called RaptorQ codes have been specified in RFC 6330 [5] and proposed to 3GPP. Decoding probability and AL-FEC overhead comparisons between Raptor and RaptorQ codes were presented in [6]. Streaming delivery (based on the H.264/AVC video codec and real-time tranport protocol (RTP)) over MBMS was studied in [4].

In this paper, we focus on live streaming over eMBMS, where the H.264/AVC encoded video content is chunked at the server into multiple segments of a fixed duration and each segment is then pushed to the client by the use of MBMS download delivery methods, i.e., through the FLUTE protocol [13]. With an end-to-end eMBMS system design approach focused on user QoE, we analyze the effect of different design parameters on startup delay, rebuffering percentage and PSNR experienced at the MBMS client. We also provide comparisons between Raptor and RaptorQ codes to characterize their impact to the user QoE in terms of PSNR and rebuffering percentage metrics. The novel contributions of this paper include user QoE evaluations for video streaming over eMBMS, a new multi-tiered Markov model for simulating MAC layer losses and a

Manuscript received March 15, 2013; revised May 25, 2013.
Corresponding author email: utsaw.kumar@intel.com.
doi:10.12720/jcm.8.6.352-358

decoding algorithm that intelligently exploits the systematic nature of Raptor/RaptorQ codes. An earlier version of results in this paper appeared in [10]

The rest of the paper is organized as follows. In Section II, we briefly discuss the Raptor and RaptorQ codes. In Section III, we present different components of the eMBMS streaming framework, for e.g., video traffic modeling, AL-FEC design parameters, transport protocols, and loss models for the radio layer. In Section IV, we present our simulation results. We conclude in Section V.

II. Background on AL-FEC Schemes

In this section, we discuss the 3GPP standardized Raptor code and the more recent RaptorQ codes, the latter being proposed to provide improvements over the former. Both Raptor and RaptorQ codes are fountain codes, thus as many encoding symbols can be generated on-the-fly by the encoder from the source symbols. As mentioned earlier, for broadcast applications, it is important to provide some redundancy at the application layer, as retransmissions become very frequent if ARQ is used for reliable delivery of packets. For this purpose, the AL-FEC framework decomposes each file into a number of source blocks of approximately equal size. Each source block is then broken into K source symbols of fixed symbol size T bytes. The Raptor/RaptorQ codes are used to form N encoding symbols from the original K source symbols where $N>K$. Both Raptor/RaptorQ are systematic codes which means that the original source symbols are transmitted unchanged as the first K encoding symbols. Both Raptor/RaptorQ codes have linear encoding/decoding time complexity. The encoding symbols are then used to form IP packets and sent. At the decoder, it is possible to recover the whole source block from any set of encoding symbols only slightly greater than K with a very high probability. We now provide an overview of Raptor codes.

A. Raptor Codes

Raptor codes first appeared in [2] and has been standardized by 3GPP in the MBMS service specification TS 26.346 [7]. A block diagram showing the encoding process is given in Fig. 1. We describe the encoding and decoding process very briefly. More details can be found in [3]. A Raptor code is formed by a concatenation of a very high-rate block code and a non-systematic Luby Transform code [8]. The first encoding step is a pre-coding step to generate L intermediate symbols. The L intermediate symbols satisfy a set of pre-coding relationships, defined such that the last $L–K$ intermediate symbols are expressed in terms of the first K symbols. The linear operations in this step can be represented by a matrix **A**. The second encoding step is LT encoding to generate the repair symbols and form the overall code with N encoding symbols. Note that Raptor code is systematic as ensured by the pre-coding relationships. The pre-coding is useful for the following two reasons. First, since LT codes are in general non-systematic, pre-coding ensures that the overall code is systematic. Secondly, the complexity of LT codes is $O(K\log K)$, whereas for Raptor codes is $O(K)$, where K is the number of source symbols. Decoding is performed at the receiver by checking if an appropriate set of encoded symbols is available such that the matrix **A** has full rank. If the matrix **A** is rank deficient, an error is declared and the whole source block is assumed to be lost. The decoding failure probability decays exponentially in the number of repair symbols received [9]. Thus, smaller the code rate K/N, more is the redundancy and lower is the decoding failure probability, but the smaller failure probability comes at the cost of more overhead.

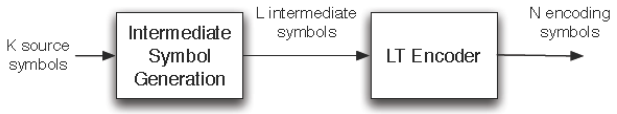

Figure. 1. Encoding overview of Raptor codes.

B. RaptorQ Codes

Since the introduction of Raptor codes in 3GPP for MBMS, there has been a lot of progress in improving the performance of Raptor codes. A superior form of Raptor called RaptorQ codes have been introduced in IETF RFC 6330 [5]. RaptorQ is more efficient than Raptor in coding efficiency, coding flexibility, etc. The encoding process for RaptorQ is given in Fig. 2. Most of the encoding steps in RaptorQ are identical to those in Raptor, albeit a few important differences. Firstly, the algebra in RaptorQ is over GF(256) as compared to GF(2) for Raptor codes. Operating over larger fields allows the recovery of source blocks with lesser overhead. Secondly, the first step of encoding in RaptorQ is to construct an extended source block of K' symbols by padding the original source block with zeros, such that K' is one of the values listed in the specification [5], (Section 5.6). Padding enables faster encoding and decoding, while at the same time minimizing the amount of information that needs to be stored. Note that the encoder does not need to send the padded symbols because padding can also be done at the decoder once the symbols have been received. Thirdly, in the next encoding step, an enhanced two-step pre-coder is used. Fourthly, a superior algorithm for LT encoding is used in the final step of the encoding process. Lastly, RaptorQ codes support a wider range of the number of source symbols and encoding symbols. Specifically, RaptorQ can encode up to 56403 source symbols, whereas Raptor codes can only encode upto 8192 symbols. Because of the above properties, RaptorQ codes are much more superior to Raptor codes in terms of performance as well as flexibility. RaptorQ can deliver huge chunks of data at once, owing to the fact that it can support large source block sizes. File delivery decoding efficiency is

thus superior, since protection is distributed over the larger chunks of the file. For delay sensitive real-time applications too, RaptorQ has superior performance because of the crucial functional differences listed above. RaptorQ combines operations over GF(2) and GF(256) to keep the complexity low, such that majority of the operations are over GF(2), and only a small fraction of the operations are over GF(256). Decoding is similar as in Raptor codes, where the decoder checks for an appropriate set of encoding symbols such that the matrix **A** has full size.

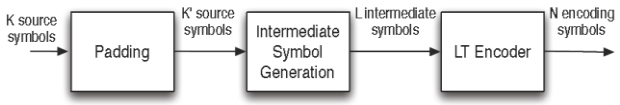

Figure. 2. Encoding overview of RaptorQ codes.

III. Streaming Framework

A. Video Traffic Modeling

Simulation of video traffic based on packet traces of actual encoded video streams has been considered a viable approach for evaluation of various video delivery systems. While the video bit streams give the actual bits carrying the video information, the video traces only give the number of bits used for the encoding of the individual video frames. Each video frame is either intra-coded (*I*), forward predictive coded (*P*) with motion compensated prediction from the preceding *I* or *P* frames, or bi-directionally predictive coded (*B*) according to its position in a group of pictures (GoP) structure. One GoP consists of a fixed number of *I*, *B*, and *P* frames.

TABLE I: Video Trace Details

Video source	Quantization parameter	Raw bitrate (kbps)
Sony-1080	34	164.72
NBC News	34	190.66

Frame size video traces are files mainly containing video frame time stamps, frame types (e.g., *I*, *P*, or *B*), encoded frame sizes (in bits), and frame qualities (e.g., PSNR). Thus, in comparison with video traffic models, simulation with trace-based video traffic sources can provide much better information about the quality of the received video stream. Publicly available video traces can be used for the simulation (http://trace.eas.asu.edu) [11], [12]. For the purpose of our simulations, we use the H.264 SVC single layer video traces with encoding type: Main (Level 2.1). The H.264 SVC is an attractive choice for video streaming simulations due to their superior compression efficiency and network friendly video representation. The resolution is CIF 352×288. The videos are variable bit rate and encoded with a fixed quantization parameter. The video details used for our simulations are given in Table I The H.264 standard also defines a network abstraction layer (NAL) which contains functions for mapping the coded video data to a network transport layer. For the purpose of our simulations, we assume that each video frame is preceded by a 10 byte NAL prefix.

B. AL-FEC Framework and Transport Layer Processing

The eMBMS-based live video streaming is over the FLUTE protocol, which allows for error-free transmission of files via unidirectional MBMS bearers. Since we are looking at live streaming, there are strict time constraints for content delivery. Each video session is delivered as a FLUTE transport object as depicted in Fig. 3. Transport objects are created as soon as packets come in. The IPv4/UDP/FlLUTE header is a total of 44 bytes per IP packet. The size of an IP packet is 1333 bytes, thus the maximum FLUTE payload size is (1333–44) = 1289 bytes. The AL-FEC increases the number of parameters that have to be chosen. The choice of the AL-FEC parameters is made at the Broadcast Multicast Service Center (BMSC). For example, the BMSC has to select the number of source symbols *K*, the code rate *K/N* and the source symbol size *T*. Larger the value of *K*, better is the efficiency and performance of the AL-FEC scheme. On the other hand, larger value of *K* leads to higher latency for a fixed symbol size *T*, since there is a need to wait a long time for sufficient number of frames to come in order to do the encoding. The code rate choice also affects the end-to-end performance because, smaller the value of *K/N*, more is the redundancy added to the source symbols as more repair symbols are generated, and thus better is the performance. However, more repair symbols come at a cost and thus the code rate needs to be chosen intelligently. In addition, it is beneficial to choose the symbol size *T* as a power of 2 for the sake of complexity. Smaller values of *T* for a fixed symbol size can mean large values of *K*, resulting in better performance, but at the same time, resulting in higher complexity because of the larger dimensions of the encoding and decoding matrices. On the contrary, larger value of *T* results in small *K* and thus poor performance.

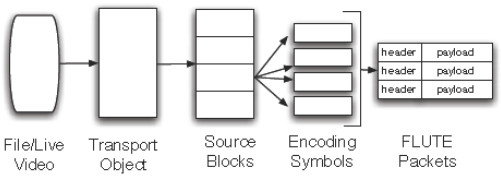

Figure. 3. Transport layer processing overview.

For our simulations, the source block parsing is done as follows. We simulate a live service and thus, a long waiting time for encoding is not desirable. However, to ensure good Raptor/RaptorQ performance, we need to choose a high enough *K*. Thus, we have a minimum value of $K=K_{min}$ as a design parameter. The larger K_{min} is, larger is the start-up delay. One source block consists of one or more GoPs, which is decided

based on the size of the GoP. The size of a source block is chosen as follows. For this algorithm, it is assumed that the gth GoP needs to be transmitted.

Algorithm: Choosing the source block size

1. source block size ← 0
2. while source block size<K_{min} T
3. do source block size ←source block size+size of gth GoP
4. g= g+1
5. end while

Note that a source block always consists of integer number of GoPs unless size of a GoP is greater than 8192, the maximum allowed K for Raptor. Furthermore, K is computed as

$$K = \left\lceil \frac{\text{source block size}}{T} \right\rceil$$

where $\lceil x \rceil$ rounds x to the nearest integer greater than x. Note that the value of K varies depending to the size of GoPs. The resulting source block is padded with ($K_{min}T$ – source block size) zeros to ensure that there are exactly K symbols. Now N encoding symbols are generated from these K symbols using the AL-FEC scheme. IP packets are then formed using these encoding symbols as payloads. The FLUTE packet is generated from the FLUTE header and payload containing the encoding symbols.

C. RLC (radio link control)/MAC (multiple access control) layer processing

1) RAN Endorsed Model: IP packets (RLC-SDUs (service data unit)) are mapped into fixed-length RLC-PDUs (protocol data unit). A 3GPP RAN1 endorsed two-state Markov model [14] is used for the simulation of LTE RLC-PDU losses as shown in Fig. 4. The Markov model satisfies the following two properties:

(1) each state persists for 10 ms, and

(2) a state is good if it has

- less than 10% packet loss probability for the 1% and 5% BLER simulations,
- less than 40% packet loss probability for the 10% and 20% BLER simulations.

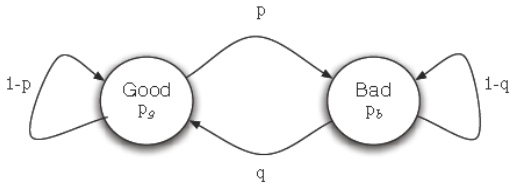

Figure. 4. Markov model for simulating LTE RLC-PDU losses.

The parameters in the figure are as follows:

- p: transition probability from Good state to Bad state. 1/p will be the average length of the Good state segment.
- q: transition probability from Bad state to Good state. 1/q will be the average length of the Bad state segment.
- p_g: BLER in Good state.
- p_b: BLER in Bad state.

The time in good state T_g can be calculated by multiplying the average length of a good segment by the sampling period. Similarly, one can calculate the the time in bad state T_b. Table II gives the Markov model parameters used for simulations.

TABLE II: MARKOV PARAMETERS FOR SPEED 3 KM/H

Para.	BLER=1%	BLER=5%	BLER=10%	BLER=20%
p	0.58%	1.80%	2.79%	4.61%
q	36.13%	24.01%	20.90%	16.80%
p_g	0.03%	0.06%	0.56%	1.16%
p_b	59.47%	70.54%	82.30%	89.20%
BLER	0.97%	5.02%	9.93%	19.92%
T_g (ms)	1724	555	359	217
T_b (ms)	28	42	48	60

It can be seen that the RAN model described above does not capture the cell phone coverage aspect of a cell. For a more comprehensive end-to-end analysis we propose the following RAN model.

2) Proposed coverage-based Markov model: for generating MBSFN subframe loss pattern, we simulate the 3GPP Case 1 channel configuration with 10 MHz carrier bandwidth. Users are picked randomly from a user population of 210 users dropped uniformly over the cell. The received SINR data is then used to generate MBSFN subframe loss pattern. Such data is collected for different MCS (Modulation and Coding Scheme) values. It was observed that the RAN endorsed model does not capture the coverage variations for the cellular layout. For the results in this paper, we assume the MCS to be 21. The rest of the LTE simulation parameters are given in Table III.

TABLE III: LTE EMBMS SIMULATION PARAMETERS

Parameters	Assumption
Cellular layout	Hexagonal grid, 19 cell sites, 3 sectors per site
Inter-site distance	1732 meters
Channel model	Typical urban
User distribution	Users dropped uniformly in cell
Total BS tx power	46 dBm
Distance-dependent path loss	L=I+ 37.6 log10 (.R), R in kilometers I = 128.1 − 2 GHz, I = 120.9 − 900 MHz
Lognormal shadowing	Similar to UMTS 30.03, B 1.141
Shadowing std. dev.	8 dB
Penetration loss	20 dB
UE speed	3 km/h
Synchronization error	None

Using the subframe loss pattern for a given MCS, we generate separate Markov models for all the 210 users (rather than one model as in Section III-C1) to capture the variations and use that to generate capacity results later in the paper. Note that this model is not a fundamentally different from the RAN endorsed model, but that it accounts for the varying BLER distribution across users

in a cellular environment. In other words, for each user we deal with a Markov model, but a different BLER is instantiated based on the coverage aspect.

For example, based on our results, we derived the following BLER distribution for a user in a cellular environment.

$$\text{BLER} = \begin{cases} 0, & \text{w.p. } 0.7206 \\ 1, & \text{w.p. } 0.0717 \\ \text{uniform}[0,1], & \text{otherwise} \end{cases}$$

Note that the BLER distribution would depend on the specific deployment models and assumptions and could be different subject to different coverage statistics.

D. Quality of Experience (QoE) Metrics

Different QoE metrics can be considered for multimedia delivery to mobile devices. In the case of file download or streaming of stored content, on user request, there is a initial startup delay after which streaming of video occurs and QoE can be measured by the initial startup delay and fraction of the time rebuffering occurs. Rebuffering can be defined as the state of streaming invoked when the playback buffer is empty and buffer is being filled up for video playback [15]. However, for live streaming, when the playback buffer is empty, the client cannot wait for the buffer to be filled up. Thus, it treats the video content corresponding to when the playback is stalled to be lost and moves ahead. We evaluate the following metrics.

- Startup delay: we define startup delay as follows.
  Definition 1: startup delay is defined as the total time it takes to deliver live video from the moment the live event starts till the moment playback starts at the user end. The main contribution to startup delay for eMBMS live streaming is the AL-FEC encoding delay, i.e., when the service provider has to wait for sufficient number of frames to be generated to ensure large enough source block for efficient AL-FEC implementation.
- Average PSNR: the PSNR of the received video stream is calculated using the offset trace file used for simulations. When a frame is lost, the client tries to conceal the lost frame by repeating the last successfully received frame. Such concealments lead to loss in video quality at the user.
- Rebuffering percentage: rebuffering percentage for live video streaming is defined as follows.
  Definition 2: rebuffering percentage is defined as the fraction of the time video playback is stalled in the mobile device. For live streaming, rebuffering occurs whenever two or more than two consecutive frames are lost. The client repeats the last successfully received frame and the video appears as stalled to the user. Video playback resumes as soon as one of the future frames is received successfully.
  We consider two approaches for decoding.
  a) D1: when an AL-FEC block is not correctly decoded, we assume that all source symbols are lost for PSNR and rebuffering percentage calculations.
  b) D2: when an AL-FEC block is not correctly decoded, we use the fact that both Raptor/RaptorQ are systematic codes and uncorrupted symbols in the first K encoding symbols are original source symbols. This may result in a better performance because of the frame structure of the video, if for e.g., we receive an uncorrupted I or P frame in the K source symbols, it will significantly enhance the performance, as opposed to approach D1, where we simply ignore the systematic structure of the code.

IV. Simulation Results

The performance bounds for eMBMS have been evaluated under different conditions. The Bearer bitrate is assumed to be 1.0656 Mbits/sec. The length of a RLC-SDU is taken as 10 ms. The content length is set at 17000 frames for each video trace. The video frame frequency is considered to be 30 frames per second. The video frames are then used to generate source blocks and encoding symbols are generated using the AL-FEC framework (both Raptor/RaptorQ). We choose the source symbol size as T=16 bytes. We keep it small in order to decrease the initial startup delay, so that we can choose a large K for the same source block size. The frames are then packetized into FLUTE packets to appropriate lengths. The RLC-PDU losses are modeled using the Markov model discussed earlier in Section III-C. All of the users were simulated assuming same target BER 5% from Table II. The received packets at the MBMS client are then decoded to reconstruct the video frames for playback. Based on the received frames, the losses of the I–P–B frames were determined. The lost frames are concealed by repeating the last successfully received frame. The system level simulations offer beneficial insights on the effect of system level and AL-FEC parameters on the overall QoE.

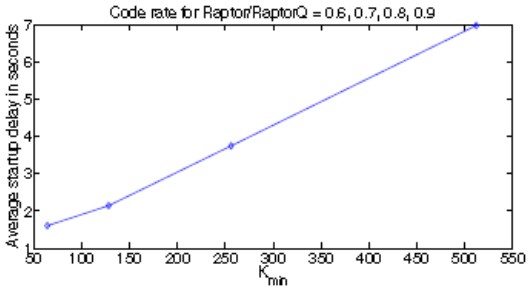

Figure. 5. Startup Delay as a function of K_{min}.

A. Startup Delay

We plot the average start-up delay (averaged over different code rates K/N=0, 6, 0.7, 0.8, 0.9) in Fig. 5 as a function of K_{min}. As expected, the startup delay increases on increasing K_{min}. It can be seen that larger the value of K_{min}, larger is the value of the startup delay.

This is a live streaming simulation and the multicast service provider has to wait for sufficient number of video frames to be generated such that the source block size is at least $K_{min}T$. Note that the startup delay for both Raptor/RaptorQ codes is same because we assume similar encoding/decoding complexity for both. Also, it was observed that the major chunk of the startup delay was because of the encoding delay. Other delays like video encoding/decoding, IP packet delivery delays, etc. are negligible as compared to the encoding delay. The decoding is assumed to be instantaneous as soon as all the encoding symbols are received.

B. Rebuffering Percentage and PSNR

1) RAN Endorsed Model: For a fixed code rate, we plot the average PSNR and rebuffering percentage in Fig. 6 and Fig. 7 respectively for the RAN endorsed model in Section III-C1. Increasing the value of K_{min} only slightly increases the performance. This is due to the fact that we treat one GoP as one source block. Most of the video traces we used have most GoPs big enough to ensure high values of K for most of the source blocks. Thus, on average, the value of K does not change. The superiority of RaptorQ as compared to Raptor can be clearly seen from the plots, especially in terms of rebuffering percentage. We also see that the second decoding approach D2 is much more efficient than the decoding approach D1. This is because in D2, we recover as many frames as we can, given the systematic structure of Raptor/RaptorQ codes. We also observed that decreasing the code rate K/N improves the performance both in terms of average PSNR and rebuffering, as expected.

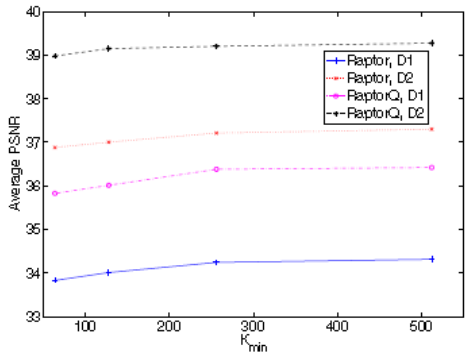

Figure. 6. Performance comparisons for K/N = 0.9: Average PSNR.

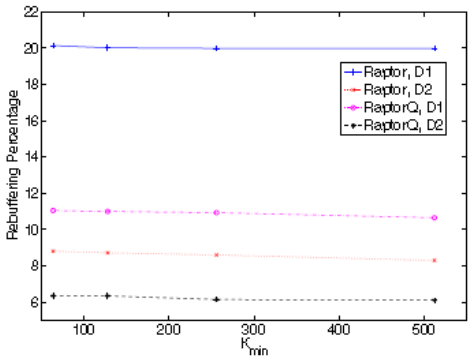

Figure. 7. Performance comparisons for K/N = 0.9: Rebuffering percentage.

2) Proposed coverage-based Markov model: using the new model proposed in Section III-C2, we plot the empirical cumulative density function (CDF) of the PSNR and rebuffering percentage for code rates 0.9 and 0.8, as shown in Fig. 8 and Fig. 9, respectively. Since increasing the value of K_{min} does not affect the PSNR and rebuffering performance much, for the simulations in this section we fix K_{min} to be 64. It can be observed that improving the code rate improves the coverage from a QoE perspective as it guarantees better PSNR and rebuffering for more number of users. It can be seen that the results in the previous section is just a point on the curve and thus, the RAN endorsed model is insufficient for analysis of coverage and resulting video streaming QoE distribution across users

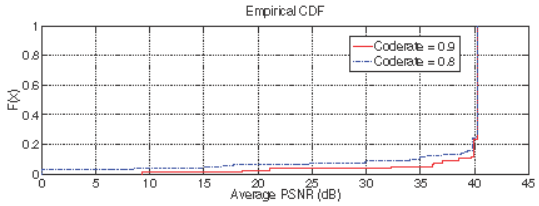

Figure. 8. Performance comparisons for K/N = 0.9: Average PSNR.

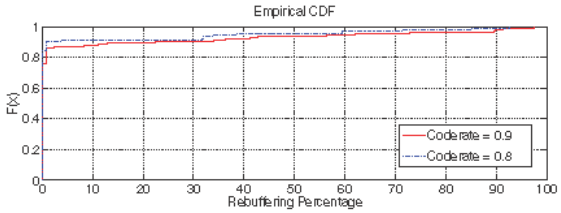

Figure. 9. Performance comparisons for K/N=0.9: Rebuffering percentage.

V. Conclusions

With an end-to-end eMBMS system design approach focused on evaluating QoE, we simulated the effect of different design parameters on startup delay, rebuffering percentage and PSNR experienced at the MBMS client. The new Markov model proposed captures the coverage aspects of MBMS as compared to the old model that assumed the same model for all users. The decoding strategy for AL-FEC decoding that takes into account the systematic code structure for Raptor/RaptorQ was shown to outperform the decoding strategy when a decoding failure leads to the loss of the whole source block.

References

[1] Cisco, *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update*, 2010-2015.

[2] A. Shokrollahi, "Raptor codes," *Digital Fountain*, Tech. Rep. DR2003-06-001, Jun. 2003.

[3] M. Luby, A. Shokrollahi, M. Watson, and T. Stockhammer, Raptor Forward Error Correction Scheme for Object Delivery, RFC 5053 (Proposed Standard), IETF, Oct. 2007.

[4] J. Afzal, T. Stockhammer, T. Gasiba, and W. Xu, "Video streaming over MBMS: A system design approach," *Journal of Multimedia*, vol. 1, no. 5, Aug 2006.

[5] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, and L. Minder, RaptorQ Forward Error Correction Scheme for Object Delivery, RFC 6330 (Proposed Standard), IETF, Aug 2011.
[6] C. Bouras, N. Kanakis, V. Kokkinos, and A. Papazois, "Evaluating RaptorQ FEC over 3GPP multicast services," in *Proc. 8th Int. Wireless Communications & Mobile Computing Conference* , August 27-31, 2012.
[7] 3GPP TS 26.346. (2011). Multimedia broadcast/multicast Service (MBMS): Protocols and codecs, *3rd Generation Partnership Project (3GPP).* [Online]. Available: http://www.3gpp.org/ftp/Specs/archive/26 series/26.346/
[8] M. Luby, "LT codes," in *Proc. 43rd Annual IEEE Symposium on Foundations of Computer Sci.*, pp. 271-282, 2002.
[9] T. Stockhammer, A. Shokrollahi, M. Watson, M. Luby, and T. Gasiba, *Application Layer Forward Error Correction for Mobile Multimedia Broadcasting*, Handbook of Mobile Broadcasting: DVB-H, DMB, ISDB- T and Media Flo, CRC Press, pp. 239–280, 2008.
[10] U. Kumar and O. Oyman, "QoE evaluation for video streaming over eMBMS," presented at IEEE International Conference on Computing, Networking and Communications, San Diego, USA, Jan 2013.
[11] P. Seeling and M. Reisslein, "Video transport evaluation with H.264 video traces," *IEEE Communications Surveys Tutorials*, vol. 14, no. 4, pp. 1–24, Sep 2011.
[12] G. Van Der Auwera, P. David, and M. Reisslein, "Traffic and quality characterization of single-layer video streams encoded with the H.264/MPEG-4 advanced video coding standard and scalable video coding extension," *IEEE Trans. Broadcast*, vol. 54, pp. 698-718, Sep 2008.
[13] T. Paila, M. Luby, R. Lehtonen, V. Roca, and R. Walsh, FLUTE-File Delivery Over Unidirectional Transport, RFC 3926 (Proposed Standard), IETF, Oct 2004.
[14] 3GPP Tdoc S4-111021, "Channel modeling for MBMS," *3rd Generation Partnership Project (3GPP)*, 2011.
[15] S. Singh, O. Oyman, A. Papathanassiou, D. Chatterjee, and J. G. Andrews, "Video capacity and QoE enhancements over LTE," in *Proc. IEEE International Conference on Communications*, Ottawa, Canada, June 2012.

Utsaw Kumar is a wireless systems engineer with the Standards and Advanced Technology Division of the Mobile and Communications Group at Intel. He is author or co-author of over 10 technical publications, He holds Ph.D. and M.S. degrees from University of Notre Dame, and a B.Tech. degree from Indian Institute of Technology in Kanpur, India.

Ozgur Oyman is a senior research scientist and project leader in the Wireless Communications Lab of Intel Labs. He joined Intel in 2005. He is currently in charge of video over 3GPP Long Term Evolution (LTE) research and standardization, with the aim of developing end-to-end video delivery solutions enhancing network capacity and user quality of experience (QoE). He also serves as the principal member of the Intel delegation responsible for standardization at 3GPP SA4 Working Group (codecs). He is author or co-author of over 70 technical publications, and has won Best Paper Awards at IEEE GLOBECOM07, ISSSTA08 and CROWNCOM08. His service includes Technical Program Committee Chair roles for technical symposia at IEEE WCNC09, ICC11, WCNC12, ICCC12 and WCNC14. He also serves an editor for the IEEE Transactions on Communications. He holds Ph.D. and M.S. degrees from Stanford University and a B.S. degree from Cornell University.

Apostolos Papathanassiou is a senior principal wireless architect with the Standards and Advanced Technology division of the Mobile and Communications Group at Intel. He has hundreds of technical contributions on 3G and 4G mobile broadband research and development since 1996 and on 3G and 4G standardization in ITU, 3GPP, and IEEE since 1999. His current interests include advanced multi-antenna PHY algorithms, multi-technology heterogeneous networks, and device-to-device communications.