Implementation Of AL-FEC Raptorq Code Over 3GPP Embms Network

Ms. Avani U Pandya [1], Prof. Sameer D Trapasiya [2], Santhi S Chinnam [3]

[1]*PG Student,* [2]*Assistant Professor,*[3]*Telecom Engineer*

[1,2]*Department of Electronics & Communication Engg.,* [3]*4G RAN & Device Validation*

[1, 2] *G .H. Patel College of Engineering & Technology,* [3]*Rancore Technologies Pvt Ltd*

[1, 2]*Vallabh Vidyanagar - 388120,*[3]*Navi-Mumbai- 400701,India*

Abstract

Long Term Evolution (LTE) is the current standard for wireless mobile communication based on Third Generation Partnership Project (3GPP). LTE includes enhanced multimedia broadcast and multicast services (MBMS), also called as Evolved multimedia broadcast and multicast services (eMBMS) where the same content is transmitted to multiple users in one specific area. eMBMS is a new function defined in 3GPP Release 8 specification that supports content distribution and streaming to group users into LTE mobile networks. In LTE an important point of demanding multimedia services is to improve the robustness against packet losses. In this sense, in order to effectively support point-to-multipoint download and streaming delivery, 3GPP has included an Application Layer Forward Error Correction (AL-FEC) scheme in the standard eMBMS. The standard AL-FEC system is based on systematic, fountain Raptor codes. Raptor coding is very useful in case of packet loss during transmission as it recover all data back from insufficient data at receiver terminal However, the 3GPP standardized systematic fountain Raptor code is nowadays considered obsolete, since a new variation of the Raptor codes has emerged. This enhanced AL-FEC scheme, named RaptorQ, promises higher protection efficiency and superior flexibility on the provision of demanding mobile multicast services. In this work, we provide an extensive performance evaluation presenting at first a theoretical performance comparison of the newly introduced RaptorQ(IETF RFC 6330) FEC scheme with its predecessor Raptor code, examining the enhancements that RaptorQ introduces on the AL FEC protection robustness with Thereafter, to verify the enhanced performance of RaptorQ, we present several simulation results of its implementation considering the modeling of the AL-FEC protection over multicast services for next generation mobile networks.

1. Introduction

Nowadays there is a significant demand for multimedia services over wireless networks due to the explosive growth of the multimedia internet applications and dramatic increase in mobile wireless access. It is, therefore, foreseen that the wireless systems will have to support applications with increased complexity and tighter performance requirements, such as real-time video streaming. Furthermore, it is expected that popular content is streamed not just to a single user, but to multiple users attempting to access the same content at the same time. This is addressed by standardization bodies through introduction of a point-to-multipoint service - enhanced Multimedia Broadcast Multicast Service (eMBMS), a resource-efficient transmission scheme targeting simultaneous distribution of multimedia content to many user devices within a serving area, over a single set of Core Network and Radio resources. So, Multimedia Broadcast and Multicast Service (MBMS) has been standardized as a key feature in Third Generation Partnership Project(3GPP) systems to broadcast and multicast multimedia content to multiple mobile subscribers via MBMS radio bearer service.MBMS is a point-to-multipoint (PTM) Standard, whose further evolvement and enrichment attracts nowadays widespread interest.

Long Term Evolution (LTE) provides both the transmission mode single-cell MBMS, MBMS services which are transmitted in a single cell and multi-cellular evolved MBMS transmission mode, providing synchronous MBMS transmission from multiple cells , also known as multicast / broadcast single frequency network mode of transmission. To transmit the same data to multiple recipients allows network resources to be shared. In order to meet the increasing use of high bandwidth multicast services, 3GPP initially standardized MBMS in third generation mobile systems. MBMS is a unidirectional ptm service in which data are transmitted from a single source to a group of multiple mobile endpoints in a specific service area. MBMS allow for multiple mobile subscribers to share radio and core network resources and as such offer many advantages regarding system resource utilization. The MBMS provide two modes of

operation, the broadcast and the multicast mode. 3GPP defines three distinct functional layers for the delivery of MBMS services: the user service, the delivery method and the bearers. MBMS user services are built on top of the MBMS bearer service.3GPP defines a set of media codecs, formats and transport/ application protocols to enable the deployment of several MBMS user services with different requirements. Furthermore, 3GPP defines two delivery methods for the MBMS user services, namely download and streaming. The delivery of software upgrades is an example of application using the download delivery method, while the delivery of real-time video is an example of the streaming delivery. MBMS delivery methods make use of the MBMS bearer service in order to distribute an application to multiple subscribers. Finally, bearers provide the mechanism by which IP data is transported. A MBMS bearer is an IP-multicast packet flow between a multicast gateway and the mobile MBMS subscribers.

3GPP focuses on the provision of reliability control over the MBMS delivery. A crucial point in achieving this objective is the introduction of a Forward Error Correction (FEC) mechanism on the application layer for both MBMS delivery methods. FEC is a method used for ‘‘forward’’ error control in data transmission over unreliable channels, such as radio transmission channels. The ‘‘forward’’ concept of FEC is justified by the redundant data transmission in advance the source information, unlike the common methods for error control (i.e. ARQ, Carousel) that are based on lost or corrupted packets retransmission to obtain the recipients the ability to overcome packet losses. The application of FEC on ptm reliability protocols, such as the MBMS environment, provides particular advantages since the redundancy introduced in the multicast transmission can overcome the common methods limitations [2]. The most important property of FEC codes is the ability to use the same FEC packets to simultaneously repair different independent packet losses at multiple receivers. However, FEC comes with its own cost since FEC protection must be carefully applied with respect to the current network conditions so as to avoid channel bandwidth wastage and achieve an efficient and reliable multicast delivery. 3GPP recommends the use of the systematic, fountain Raptor code as an Application Layer FEC (AL-FEC) protection mechanism exclusively for MBMS [1]. Raptor FEC [3] was selected due to the higher performance compared with existing AL-FEC codes. However, in the meantime a new very promising variation of Raptor codes has emerged, named RaptorQ [4]. RaptorQ is the most recent member of Raptor codes family, providing exceptional protection performance and enhanced encoding parameters

The rest of this paper is organized as follows: in Section 2 we provide an overview of the 3GPP AL-FEC eMBMS delivery framework and Section 3 presents a detailed description of the examined AL-FEC scheme with IETF RFC 6330 .In Section 4 we present the implementation simulation environment and the conducted experimental results. Finally, in Section 5 we draw our conclusions and we describe some possible future steps.

2. EMBMS Protocol Stack

2.1 3GPP AL-FEC eMBMS DELIVERY

3GPP defines two delivery methods namely, downloading and streaming. eMBMS user plane stack of these delivery methods is illustrated in Fig. 1.

2.1.1MBMS Streaming Delivery Protocols Stack:

The purpose of the MBMS streaming delivery method is to deliver continuous multimedia data (i.e. speech, audio, and video) over an MBMS bearer. MBMS makes use of the most advanced multimedia codecs such as H.264 for video applications and enhanced *Advanced Audio Coding* (AAC) for audio applications. Real-time transport protocol (RTP) is application layer transport protocol for MBMS streaming delivery. RTP provides means for sending real-time or streaming data over user datagram protocol (UDP), the resulting UDP flows are mapped to MBMS IP multicast bearers. Furthermore RTP provides RTP Control Protocol (RTCP) for feedback about the transmission quality.

3GPP recommends the use of an AL-FEC mechanism by the sender before RTP flows are mapped onto UDP. The MBMS AL-FEC streaming framework operates on RTP/UDP flows. A copy of the source packets is forwarded to the Raptor encoder and arranged in a source block with row width T bytes with each packet occupying a new empty row. The source block is filled up to k rows, where the value of k can be different for each source block and depends on the variable streaming services constraints. After forming a FEC source block from the packets to be protected together, the Raptor encoder generates the desired repair symbols. These generated Raptor repair symbols are then sent using the FEC repair packet format.

2.1.2 MBMS Download (File) Delivery Protocols Stack: MBMS download delivery method aims to distribute discrete objects (e.g. files) by means of a MBMS download session. Download method uses the

File deLivery over Unidirectional Transport (FLUTE) protocol when delivering content over MBMS bearers. FLUTE is built on top of the Asynchronous Layered Coding (ALC) protocol instantiation. ALC combines the Layered Coding Transport (LCT) building block and the FEC building block to provide reliable asynchronous delivery of content to an unlimited number of concurrent receivers from a single sender. A detailed description of the FLUTE building block structure can be found in [1]. Thereafter, FLUTE is carried over UDP/IP, and is independent of the IP version and is forwarded to the Packet Data Convergence Protocol (PDCP) layer. The packets are then sent to the Radio Link Control (RLC) layer. The RLC Layer functions in unacknowledged mode. The RLC layer is responsible for mapping IP packets to RLC SDUs. The Media Access Control (MAC) Layer adds a 16 bit header to form a PDU, which is then sent in a transport block on the physical layer.

In order to apply AL-FEC protection on the MBMS download delivery, the transmitted file is partitioned in one or several source blocks. Each source block consists of k source symbols, each of length T except for the last source symbol, which can be smaller. Through the Raptor encoding, for each source block, redundant repair symbols are generated according to the desired amount of protection. A unique ID is assigned on each resulting encoding symbol, which can be a source or a repair symbol, in order to identify the type of the symbol according to the assigned ID. Subsequently, one or more FEC encoding symbols are placed in each FLUTE packet payload and the resulting packets are encapsulated in UDP and distributed over the IP multicast MBMS bearer.

Furthermore, 3GPP defines a post-delivery procedure to provide file repair features for the MBMS download delivery. The purpose of the file repair procedure is to repair lost or corrupted file fragments from the MBMS download data transmission. A MBMS client is able to determine, for each source block of each file, which source symbols should have been received but have not and is also able to determine the number of symbols it has received. Therefore, each MBMS client is able to determine the number of further symbols required and send a file repair request message to a file repair server for unreceived symbols which will allow the MBMS FEC decoder to recover each protected block of the file. Thereafter, the MBMS client can receive the requested repair data through a point-to-point (ptp) or a ptm repair data delivery.

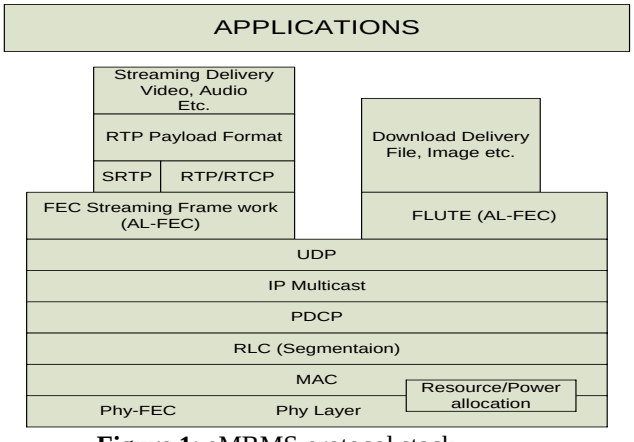


Figure 1: eMBMS protocol stack

3. AL-FEC Schemes

In general, AL-FEC codes can be considered as correcting codes for an erasure channel. In an erasure channel a transmitter sends a symbol i.e., a fragment of the source data, with the receiver either receiving or not the transmitted symbol. AL-FEC aims to cope with these symbol erasures by adding some redundancy in the transmitted data. Raptor codes were firstly introduced as a FEC erasure code in [5]. In this section we provide an analytical description of the two examined members of the Raptor codes family. We focus on the improvements that the newer member of Raptor codes, named RaptorQ, has emerged and we further provide a theoretical performance

3.1RAPTORQ CODE (IETF RFC 6330)

Since the systematic fountain Raptor code was adopted from 3GPP as the standardized AL-FEC scheme for MBMS, there has been significant progress in the design of erasure codes. The outcome of this progress is the emergence of an enhanced Raptor code at Internet Engineering Task Force (IETF) [4] in order to address the drawbacks of the standardized Raptor code on the recovery properties described in A. This newer member in Raptor codes family is known as RaptorQ code. RaptorQ is also a fountain and systematic AL-FEC code. RaptorQ is a significantly more efficient AL-FEC code than the older Raptor code, in terms of superior flexibility and higher protection and coding efficiency. The encoding process of RaptorQ code is mostly identical with that of Raptor code described in the previous subsection. However, RaptorQ code introduces certain design selections, analyzed below, that ensure superior performance compared with that of Raptor code. A key differentiation between the two schemes is that the standardized Raptor code operates over Galois field GF (2) [3], while the enhanced RaptorQ code uses symbol operations over GF(256) [4]

instead of over GF(2). Operating over larger finite fields allows RaptorQ to overcome the performance limitations of Raptor code since utilizing larger finite fields offers the potential of achieving recovery with lower reception overhead than the existing Raptor code. Moreover, additional important aspects of the enhanced properties of RaptorQ code are the increased number of possible source symbols and the increased number of generated encoding symbols. More precisely, RaptorQ can encode up to 56,403 source symbols into a source block in contrast to 8,192 of the Raptor code and furthermore can generate up to 16,777,216 encoding symbols, 256 times more than the older Raptor code. The expanded range of these two parameters simplifies the application of the AL-FEC protection and offers higher flexibility to RaptorQ. Based on the properties of RaptorQ code, it is obvious that can perform better and more flexible both for file delivery and streaming services. Since RaptorQ can deliver files up to 3.4 GB as a single source block maximizes the decoding efficiency and protection due to the spreading of protection across the whole file, particularly for very large files. On the delay-sensitive real-time applications, the flexible range of the block size parameter allows to determine a QoS trade-off between protection and latency considering the delay constraints of the transmitted application. At the same time RaptorQ achieves lower computational complexity [8] than the older Raptor code. Concerning the performance of RaptorQ, as already mentioned, the key property of a Raptor codes member is the probability of a successful decode as a function of the received symbols. The decoding failure probability of RaptorQ code can be modeled by (1) [8]:

$$p_{f_{RQ}}(n,k) = \begin{cases} 1 & \text{if } n < k \\ 0.01 \times 0.01^{n-k} & \text{if } n \geq k \end{cases} \quad (1)$$

In (2), p_{fRQ} (n,k) denotes the probability of a failed decode of a RaptorQ protected block with k source symbols if n encoding symbols have been received The study presented in [11] describes the decoding failure probability of Raptor code as a function of the source block size and the received symbols. In fact, the inefficiency of the Raptor code can accurately be modeled by (2) [7]

$$p_{f_R}(n,k) = \begin{cases} 1 & \text{if } n < k \\ 0.85 \times 0.567^{n-k} & \text{if } n \geq k \end{cases} (2)$$

Comparing (1) with (2), the performance superiority of RaptorQ code is unambiguous.

3.1.1 RaptorQ (RFC 6330) Encoding Process: For RaptorQ Encoder source object is divided into Z>=1 number of source blocks. Each source block has K source symbols of size T bytes each. Each source block is encoded independently from the next Block. The source block construction is specified in IETF RFC 6330 [4]. The systematic RaptorQ codes generate encoding symbols which contain K source symbols plus repair symbols.

Step 1:- The first step in encoding can be performed by constructing an extended source block by adding zero or more padding symbols such that the total number of symbols be K'. Each padding symbol consists of T octets where the value of each octet is zero.

Let T be the input of *K* source symbols that are to be encoded ($1 \leq K \leq 56{,}403$). An arbitrary vector T of size *K* is padded to a vector T of size *K'*. This operation is performed by the "Padding" block in figure. 3. *K'* can take a subset of 477 source block size values, uniformly distributed in the range from 1 to 56,403, for which GF (256) Raptor code is defined. The mapping of *K* to *K'*, which is unique to GF (256) Raptor, minimizes the amount of table information that needs to be stored and enables faster encoding and decoding. Additionally, the padding zero symbols are not transmitted but they are *a-priori* known at the decoder and act as an additionally available parity information.

Step 2:- The second step in encoding can be performed by generating an LxL encoding matrix, A to calculate L intermediate symbols as [4].

$$C = A^{-1} D \quad (3)$$

Where, C represents column vector of the L intermediate symbols, D represents column vector of S+H zero symbols followed by the K' source symbols.

The precode matrix A_{LxL} consists of several submatrices as shown in figure 3. In this matrix A, the top S+H rows represent the constraints of the pre-code on C, and the bottom K' rows, each corresponding to a extendend source symbol of D. A three stage pre-coding algorithm is used for RaptorQ. $\mathbf{G_{LDPC1}}$ and $\mathbf{G_{LDPC2}}$ are the generator submatrices of two regular low density parity check codes (LDPC), defined over GF (2). They constitute the LDPC stage of the pre-coding algorithm and are responsible for generating most of the precode redundant symbols or LDPC symbols. The number of columns *B* in $\mathbf{G_{LDPC1}}$ denotes the number of intermediate symbols that are LT symbols, excluding the LDPC symbols. From (*U* + *H*) number of columns in $\mathbf{G_{LDPC2}}$, number *U* in particular denotes the number of non-HDPC intermediate

symbols, which are permanently inactive. That is these symbols are not to be decoded with belief-propagation (BP) algorithm .Submatrix $\mathbf{G_{HDPC}}$ is a high density parity check code (HDPC) matrix, defined over GF (256).

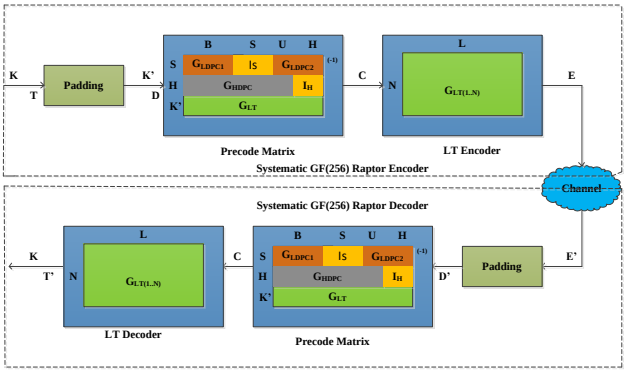


Figure 2: Block diagram of Raptor encoder and decoder

This submatrix constitutes the second stage of the pre-coding algorithm and generates a small number of additional HDPC pre-code redundant symbols. The HDPC code matrix is the main difference that sets apart RaptorQ code from Raptor code and, to a greater extend, any other linear block code in use today. Submatrix $\mathbf{G_{LT\ (1...K')}}$ formed from the first K' rows of the Luby Transform (LT) [6] matrix is included in matrix **A** to render the whole RaptorQ code systematic. Submatrix $\mathbf{G_{LT\ (1...K')}}$ constitutes the third stage of the precoding algorithm and generates LT pre-code redundant symbols. Submatrices $\mathbf{I_S}$ and $\mathbf{I_H}$ are identity matrices. Note that $K' = B + U$ to make $\mathbf{A_{L\times L}}$ a square matrix. All the precode constants are tied to the value of K' through algebraic relations [4].

Step 3:-The third encoding step is to generate the repair symbols from L intermediate symbols using LT encoding process as Enc[K',(C[0],C[1],...,C[L-1]),(d, a,b,d1,a1,b1)] [4] where *(d, a, b,d1,a11,b1)* represents tuples for each symbol.

The number of rows N in the LT encoder matrix $\mathbf{G_{LT\ (1...N)}}$ is set according to the desired rate and the expected block erasure probability. To avoid transmitting the K'-K zero padding symbols, the rows with K to K'-1 in $\mathbf{G_{LT\ (1...N)}}$ matrix do not participate in the generation of encoded symbols E.

At the end of step 3 final transmitted streams E are generated ($N \geq K'$).Where the last $N - K$ symbols in E are repair symbols.

Thus the RaptorQ encoding process is performed according to

$$E_{N\times1} = G_{LT(N..1)} . C_{L\times1} = G_{LT(N..1)} . A^{-1}_{L\times L} . D_{L\times1} \quad (4)$$

where $G_{LT\ (N..1)}$ and $E_{N\times1}$ represent $G_{LT(1..N)}$ matrix and output vector $E_{N\times1}$, respectively, with rows K to (K' - 1) removed. Therefore, $E_{N\times1}$ contains K source symbols plus the desired amount of repair symbols, resulting in total of N - (K' - K) encoded symbols that are transmitted through the channel. The most time consuming operation is the inversion of matrix **A**.

3.1.2RaptorQ (RFC 6330) Decoding Process:

The decoding process of GF (256) Raptor code exchanges the positions of the pre-code matrix and the $\mathbf{G_{LT}}$ encoder (to be used as LT decoder) matrix, as illustrated in Fig. 3. Note that both matrices are appropriately sized. This allows for successful decoding when only the first K encoded symbols have been received and no errors are detected in the channel. The input symbol from the channel contains the encoded symbols E' (which may be nonconsecutive). This are padded with (K' - K) zeros to produce symbol E' of size N' ($K' \leq N' \leq N$). Symbols E' is further augmented with $S + H$ additional zeroes to form D' of size M. Starting with $N' = K'$ the value of N' is iteratively incremented, to the maximum of N, to make the matrix **A** invertible. The difference (N' - K') is equal to or greater than the number of received encoded symbols lost in the channel. The decoding is performed according to

$$T'_{K\times1} = G_{LT(1..K)} . A^{-1}_{M\times L} . D'_{M\times1} = G_{LT(1..K')} . C_{L\times1} \quad (5)$$

The matrix, A is a bit matrix that satisfies A x C = E using matrix multiplication in GF (256). Intermediate symbols C can then be decoded if the bit matrix A is square (L x L) and invertible. Since the number of received encoding symbols, N > K' in most cases, so following steps should be taken for decoding.

Step 1:- The first step in decoding is to convert (M x L) matrix A to an (L x L) matrix using Gaussian Elimination method [4].

Improved Gaussian elimination, consisting of row/column exchange and row Ex-OR, is used in the 3GPP Raptor decoding algorithm. In the decoding process, the original matrix ***A*** will be converted into an identity matrix. Besides, vector ***C*** and ***D*** change concurrently. Let （$N \geq K$） be the number of received

encoding symbols and *M=S+H+N*. The vector *D=(D[0],… ,D[M-1])* is the column vector of *M* symbols with values known to the receiver, where *D[0]*, … ,*D[S+H-1]* are zero-valued symbols that correspond to LDPC and HDPC symbols, *D[S+H]*, … ,*D[M-1]* are the received encoding symbols for the source symbols. When the original matrix ***A*** is converted into identity matrix successfully, we can get the intermediate symbols from ***D***.

Before Gaussian elimination, we assume *C[0]=0* , *C[1]=1*, … , *C[L-1]=L-1* and *D[0]=0*, *D[1]=1*, … , *D[M-1]=M-1* .initially. In the process of Gaussian elimination, the vectors ***C*** and ***D*** change concurrently with the changes of matrix ***A***. The process abides by the rules as follows:

- If each time a multiple, beta, of row i of A is added to row i', then the symbol beta*D[d[i]] is added to symbol D[d[i']]
- If the row *i* of ***A*** is exchanged with row *i'*, then the value *d[i]* is exchanged with the value *d [i']*;
- If each time a row i of A is multiplied by an octet beta, then the symbol D [d[i]] is also multiplied by beta.
- If the column *j* of ***A*** is exchanged with column *j'*, then the value c*[j]* is exchanged with the value *c [j']*.

It is clear that *C[c[0]]*, *C[c[1]]*, … , *C[c[L-1]] =D[d[0]]*, *D[d[1]]*, … , *D[d[L-1]]* at the end of successful decoding.

The process of converting ***A*** into identity matrix consists of five phases: *Phase I* and *Phase III* involve the use of Belief Propagation, while *Phase II* performs Gaussian elimination. *Phase IV* and *Phase V* involve zeroing and forward elimination of submatrices of **A**, respectively.

Phase I: At the beginning of Phase I matrix A is duplicated into matrix X with the same dimensions. Let the pre-code matrix A have M rows and L columns. To achieve full rank, the dimensions of matrix A should be such that M ≥ L. Phase I of uses two column indices i and u. Index i is initialized to zero, while the u is equal to a predefined number of permanently inactive symbols (columns), that are left for decoding with GE in Phase II. At each step, the algorithm goes through the matrix and selects row j (j ≥ i), which has the least (at least one) nonzero element (r) between column i and column L - u. Rows with elements defined over GF (2) are searched in priority over ones with GF (256), in order to reduce the decoding complexity. Moreover, if r = 2 builds graphs of all connected rows and columns with 2 nonzero elements and chooses a row from the graph with the most components in it. In this way, it ensures that subsequently there will be the largest set of rows with just one nonzero element. Using those rows keeps the number of symbols (columns) that turn inactive to a minimum. When found, row j and row i are exchanged. Next, if element $a_{ii} = 0$ and there is a nonzero element on row i at position k, columns k and i of matrix A are exchanged. If there are other nonzero elements on row i, the columns containing them are exchanged with columns L - u, and u is incremented with each exchange, (r - 1 increments in total). Then, row i is XOR-ed with every row k (M > k > i), which has a nonzero element at position i. At the end of each step, i increments by one. The process repeats until Phase I completes successfully, when i + u = L. If i + u ≠ L at the end of Phase I the decoding fails. Note that Phase I may involve one additional row division operation in the case when main diagonal elements are greater than 1 [4].

When row/column exchange and row division operations are performed on matrix A, the same operations are performed on matrix X, too. However, no row XOR operations are performed on matrix X.

At the end of Phase I, matrix A is reduced to the form shown in (4).

$$A_{PhaseI(M\times L)} = \begin{bmatrix} I_i & \\ Z_{(M-i)\times i} & U_{M\times u} \end{bmatrix} \quad (6)$$

Sub-matrix I_i is an identity matrix, sub-matrix $Z_{(M-i)\times i}$ is a zero matrix, and sub-matrix $U_{M\times u}$ is a nonzero matrix, which contains the inactive symbols (columns) that will be processed in the remaining phases of the algorithm.

Phase II: In the second phase algorithm converts the lower part of the sub-matrix $\mathbf{U_{M\times u}}$ into an identity matrix $\mathbf{I_u}$, through standard GE process. The forward elimination of GE starts at row *i* and continues to row *L* - 1. If GE is successful, the last *M* - *L* rows of matrix **A** are discarded and the backward substitution is performed. At this point, the success of GE determines the success of the whole process. At the end of a successful *Phase II*, matrix **A** becomes square in the form;

$$A_{PhaseII} = \begin{bmatrix} I_i & U_{i\times u} \\ Z_{u\times i} & I_u \end{bmatrix} \quad (7)$$

In *Phase II* all the entries of matrix **X** outside the first *I* columns and rows are discarded. Matrix **X** reduces to an $i \times I$ square matrix having a lower triangular form.

Phase III: At the beginning of *Phase III*, the part of the pre-code matrix **A** that remains to be zeroed is submatrix $\mathbf{U_{i\times u}}$ in (6). This submatrix is typically dense. To reduce it to a sparse form, submatrix $\mathbf{X_{i\times i}}$ is multiplied with the submatrix composed of the first *i* rows of **A**. After this matrix multiplication the first *i* rows and columns of matrix **A** form a lower triangular form which is identical to $\mathbf{X_{i\times i}}$, while submatrix $\mathbf{U_{i\times u}}$ is sparse.

Phase IV: In *Phase IV*, matrix **Iu** is used to zero matrix Ui×u through row XOR operations.

Phase V: And finally, in *Phase V* forward elimination is used to zero the first *i* rows of matrix **A** to

$$A_{PhaseV} = \begin{bmatrix} I_i & Z_{i\times u} \\ Z_{u\times i} & I_u \end{bmatrix} \quad (8)$$

The process is guaranteed to be successful, because all the

main diagonal elements are different than zero. Note that allArithmetic operations are defined over GF (256).

Once the Gaussian Elimination is complete the bit matrix, A becomes {LxL} and invertible. The intermediate symbols C can then be obtained as

$$C=A^{-1}D \quad (9)$$

Here D represents column vector of S+H zero symbols followed by the N received symbols. The intermediate symbols, C is then passed to LT decoding to regenerate the K Source symbols. [4]

4. EXPERIMENTAL EVALUATION

4.1 Simulation Set-Up Diagram of RaptorQ Code: The implementation and performance of RaptorQ codes has been evaluated as shown in figure 3 below. The encoder and decoder are designed as specified in section 3.we have used an image as a source where a group of T symbols of the image is considered as one source symbol Simulation Parameters are number of Source Symbols K=372, Each Source Symbol of Size T=86 symbols=688 Bits. With maximum Packet loss supported up to 50% but we took results for 47% loss .

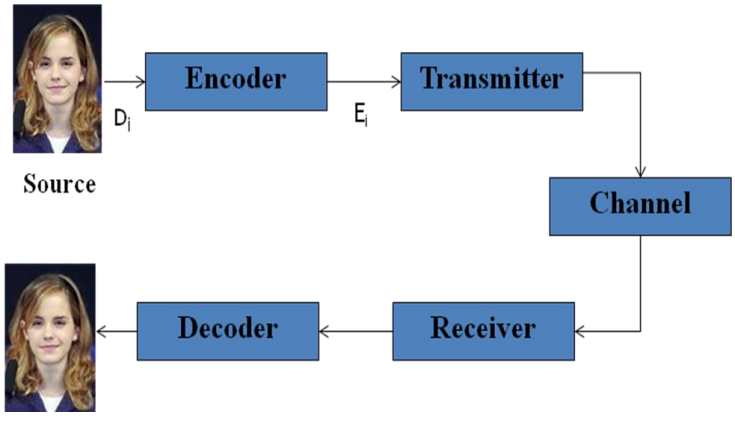
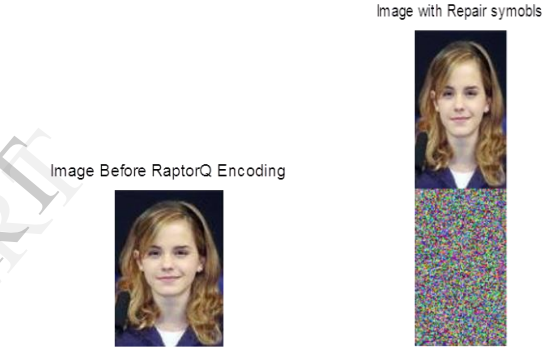
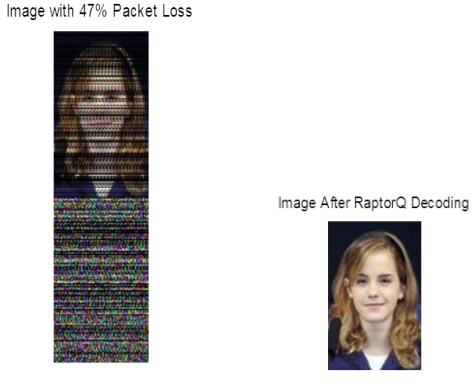


Figure 3: Block Diagram of RaptorQ Coding

4.2 Simulation Results:





5. Conclusion & Future Work

In this work we have implemented new AL-FEC RaptorQ code provided by IETF RFC 6330 to provide reliability against packet Losses. Thus, RaptorQ coding significantly improves the image quality by recovering the lost packets falling on the transmission.

Concerning some possible future steps that could follow and extend this work, providing a cross-layer design could be beneficial for the multicast transmission performance, since the interoperability between the AL-FEC with lower layers protection mechanisms could optimize the costly error protection framework in total. Furthermore, it is our belief that the introduction of an adaptive algorithm computing the optimal transmission overhead of the AL-FEC mechanism based on a sophisticated feedback-reporting scheme could further enhance the AL-FEC efficiency. Finally, since the almost ideal performance of the RaptorQ FEC addresses the shortcomings of Raptor FEC, we could evaluate the application of ALFEC protection over different transmission environments and standards and also compare both its performance in different aspects.

Acknowledgment

We would like to thank Yashesh Buch for providing such an interesting topic. We are also grateful to Harish Padi for valuable discussions and friendly support.

References

[1] 3GPP TS 26.346 V10.4.0.," Technical Specification Group Services and System Aspects: MBMS, Protocols and codecs", (Release 10), 2012.

[2] Luby, M., Vicisano, L., Gemmell, J., Rizzo, L., Handley, M., & Crowcroft, J. (2002). The use of forward error correction (FEC) in reliable multicast. RFC 3453 (Informational). URL http://www.ietf.org/rfc/rfc3453.txt.

[3] M. Luby, A. Shokrollahi, M. Watson, and T. Stockhammer, "Raptor forward error correction scheme for object delivery," September 2007, Internet Engineering Task Force, RFC 5053. Available at http://tools.ietf.org/html/rfc5053.

[4] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, and L. Minder, "RaptorQ Forward Error Correction Scheme for Object Delivery," RFC 6330, *Internet Engineering Task Force*, Aug. 2011. Available at: http://tools.ietf.org/rfc/rfc6330.txt

[5] A. Shokrollahi, "Raptor codes," Information Theory, IEEE Transactions on, vol. 52, no. 6, pp. 2551–2567, June 2006.

[6] M. Luby, "LT codes", In Proceedings of The 43rd Annual IEEE Symposium on Foundations of Computer Science, November 16-19 2002, pp. 271–282.

[7] 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, ISDBT and Media Flo, CRC Press, pp. 239–280, 2008.

[8] 3GPP."Rationale for MBMS AL-FEC enhancements."Tdoc S4-110449, 3rd generation partnership project (3GPP), (2011).