Application Layer Forward Error Correction for Mobile Multimedia Broadcasting

Thomas Stockhammer, Amin Shokrollahi, Mark Watson, Michael Luby, Tiago Gasiba

Abstract

Application Layer Forward Error Correction (AL-FEC) is an innovative way to provide reliability in mobile broadcast systems. Conventional data such as multimedia files or multimedia streams are extended with repair information which can be used to recover lost data at the receiver. AL-FEC is integrated into content delivery protocols (CDPs) to support reliable delivery. Several standardization committees such as 3GPP and DVB have recognized the importance of AL-FEC and have standardized Raptor codes as the most powerful AL-FEC codes to be used for such applications. The major characteristics of Raptor codes are channel efficiency, low-complexity, and flexibility. An important consideration when using AL-FEC is system integration. With the right system design including AL-FEC, the efficiency and/or quality of delivery services can be significantly enhanced. This work shows these benefits in selected use cases, specifically focusing on 3G based multimedia broadcasting within the MBMS standard.

I. Introduction

The primary objective of Mobile TV is to bring TV-like services to mobile phones. However, handheld devices in use today differ significantly from traditional TV equipment. For example, mobile phones integrate two-way communication network connections and flexible operating systems as well as powerful hardware platforms which enables the use of smart and powerful software applications and tools. With these valuable additions, mobile TV users can enjoy personalized and interactive TV with content specifically adapted to the mobile medium. In addition to traditional live TV channels, mobile TV delivers a variety of services including video-on-demand and video downloading services, and the content may delivered to a mobile user either on-demand or by subscription.

From a delivery perspective there are to date two different approaches to delivering mobile TV services. It is noteworthy that currently more than 90% of commercially deployed Mobile TV services run over two-way cellular network such as UMTS, CDMA2000, WiMAX, or extensions of those. However, more recently, unidirectional broadcast technologies such as DVB-H, DMB/DAB and MediaFLO are attracting significant attention. Furthermore, two-way cellular networks are currently being extended with IP multicast transport, e.g., with 3GPP Multimedia Broadcast/Multicast Services (MBMS) or 3GPP2 BroadCast MultiCast Services (BCMCS), which provide the possibility to distribute IP multicast data over point-to-multipoint radio bearers and therefore increase efficiency and allow delivery of more revenue-generating services.

The increasing demand of mobile users for multimedia information in many different application scenarios leads to significant challenges. End users having experience with TV-grade video signals also expect high quality from mobile applications. Furthermore, network and service operators expect high efficiency and low costs in terms of infrastructure and hardware, while still providing the highest customer satisfaction. Whereas personalized services can be handled by improving point-to-point distribution, popular content requires more efficient broadcast distribution. However, the reliable delivery of large files in podcasting or clipcasting-like service or simultaneous live

Thomas Stockhammer, Mark Watson, Michael Luby, and Tiago Gasiba are with Digital Fountain Inc., Fremont, California. Amin Shokrollahi is with EPFL, and with Digital Fountain. Emails: {stockhammer, amin, mark, luby, gasiba}@digitalfountain.com, amin.shokrollahi@epfl.ch

video broadcasting to many users over unreliable and bandwidth-limited networks is an extremely challenging task.

Many of the challenges arise on the physical and medium access control layers. However, there is a general tendency to reuse as much as possible of the existing network infrastructure to avoid huge initial investments. For example, DVB-H relies heavily on DVB-T infrastructure, and 3GPP MBMS reuses existing signal processing and network infrastructure from point-to-point UMTS. Therefore, the optimization potentials on the layer below IP are limited and IP multicast transmission is in general neither reliable nor completely optimized. Therefore, content delivery protocols (CDPs) play an important role in the successful service delivery over wireless and mobile channels via IP multicast.

A full specification of a CDP usually consists of a collection of different tools. Many commercial standards bodies look first for standardized protocols in the Internet Engineering Task Force (IETF). The IETF has defined protocols which provide delivery of files or streaming content to many users, and these have become important components in commercial standards. The most popular protocol for file delivery over IP multicast is the File Delivery over Unidirectional Transport (FLUTE) protocol [1]. The IETF is in the process of specifying a similar framework for the delivery of streaming media [2]. In both frameworks, the integration of application layer forward error correction (AL-FEC) [3], [4] is the primary technology used to overcome IP packet losses for the provisioning of a reliable service. The guidelines on how to use AL-FEC to provide support for reliable delivery of content within the context of a CDP are provided in [3]. The IETF is in the process of defining several AL-FEC schemes, e.g. [5], whereby especially Raptor codes [6] have been selected in different CDP definitions recently by commercial standards. For a detailed description on Raptor codes we refer to [7], but a summary of the codes can be found along with some implementation details and performance results are provided in Section II.

3GPP MBMS uses IETF standardized protocols, including the Raptor code, for streaming and file delivery. MBMS extends the existing architecture by the introduction of an MBMS Bearer Service and MBMS User Services [8]. The former is provided by the packet-switched domain to deliver IP multicast datagrams to multiple receivers using minimum radio and network resources. The Bearer service re-uses many existing components of the UMTS system such as radio access including physical layer coding based on Turbo codes. The end user is provided with two MBMS user services [9], download delivery for reliable multicasting of files as well as streaming delivery for real-time multicasting of multimedia streams. Those two services make use of different CDPs. The end-point of the CDP on the network side is a new architectural component, the Broadcast Multicast Service Center (BM-SC). It provides the MBMS User Services to the User Equipment (UE). Both delivery methods in MBMS mandate that the UE supports Raptor codes. We discuss the integration of AL-FEC into the two delivery methods and provide an overview over the MBMS system as an exemplary multimedia broadcast system in Section III.

The assessment of mobile multimedia broadcasting services is quite complex. This is especially the case because features which can provide reliability are separated in the overall protocol stack, for example physical layer forward error correction (PHY-FEC), power control, and AL-FEC. Also, mobility aspects are fairly complex to handle as long-term signal variations significantly influence the performance of a system. Furthermore, the criteria for performance evaluation are quite different for different services. Whereas file delivery services usually have relaxed timing constraints, for real-time streaming delivery and live Mobile TV aspects such as delay, latency, or channel switching times are very important. Therefore, in Section IV we introduce a realistic system level approach that allows assessing the performance of mobile multimedia broadcast applications. Selected simulation results for different services are discussed in Section V. These show the benefits of AL-FEC in

mobile broadcasting services. Some discussions and optimization potentials are presented. The final section summarizes our results and provides further conclusions.

II. Standardized Raptor Code

Raptor codes were introduced by Shokrollahi in 2001 [10] and and a comprehensive overview is provided in [7]. They are an extension of LT-codes, introduced by Luby [11]. Raptor codes have been standardized to address the needs of compliant implementations in many different environments for efficiently disseminating data over a broadcast network. The major standardization work has been done in 3GPP and the standardized Raptor specification is provided in the MBMS specification [9, Annex B], which is identical to the Raptor specification in [6] and [12, Annex B]. Raptor codes provide improved system reliability, while also enabling a large degree of freedom in the choice of transmission parameters. Raptor codes are *fountain codes*; therefore, as many encoding symbols as desired can be generated by the encoder on-the-fly from the source symbols of a source block of data. The decoder is able to recover the source block from any set of encoding symbols only slightly more in number than the number of source symbols. As a result, Raptor codes operates very closely to an ideal fountain code which would require only exactly the number of source symbols for recovery.

The following subsections are designed to familiarize the reader with the main concepts behind Raptor codes, their operational use, and efficient encoding and decoding algorithms. To fix notation, we assume that we send a piece of content consisting of k *symbols* over an unreliable channel in which symbols may get lost. In our context a symbol is a collection of bits; it can be as small as one bit, or as large as a transmission packet over the Internet. We denote the vector of symbols by $x = (x_1, x_2, \ldots, x_k)$, and we assume that all the symbols in this vector have the same size (in bits). We call vector x the *source block*, the *vector of source symbols*, or simply the *source symbols*. The encoding procedures we outline below use the simple procedure of *XOR* on the symbols; the XOR of two symbols x_i and x_j is a symbol whose ℓth bit is the XOR of the ℓth bit of x_i and the ℓth bit of x_j, respectively. We denote the XOR of x_i and x_j by $x_i \oplus x_j$. If a is in $\mathrm{GF}(2)$, then we denote by ax the symbol in which the ℓth bit is the binary AND of a and the ℓth bit of x. Using this notation, if $a_1, \ldots, a_k$ are elements of $\mathrm{GF}(2)$, then the expression $\oplus_{i=1}^{k} a_i x_i$ is a well-defined symbol.

A. Fountain Codes

Fountain codes are a novel and innovative class of codes designed for transmission of data over time varying and unknown erasure channels. They were first mentioned without an explicit construction in [13], and the first efficient construction was invented by Luby [14]. A fountain code designed for k source symbols is specified by a probability distribution $\mathcal{D}$ on the set of binary strings of length k. Operationally, a fountain code can produce from the vector x a potentially limitless stream of symbols $y_1, y_2, y_3, \ldots$, called *output symbols*, satisfying several fundamental properties:

1) Each output symbol can be generated according to the following probabilistic process: the distribution $\mathcal{D}$ is sampled to yield a vector $(a_1, \ldots, a_k)$, and the value of the output symbol is set to be $\oplus_{i=1}^{k} a_i x_i$. This process is referred to as *encoding*, and the vector $(a_1, \ldots, a_k)$ is called the *mask* corresponding to the output symbol.
2) The output symbols can be independently generated.
3) The source symbols can be recovered from any set of n output symbols, with high probability. The recovery process is usually called *decoding*, and the number $n/k - 1$ is called the *overhead* of the decoder. The probability that the decoder fails is called the *error probability* of the code.

The third condition shows that fountain codes are robust against erasures, since only the number of received output symbols is important for decoding. Different fountain codes differ in terms of their overhead for a given error probability. But they also differ in terms of the computational efficiency of the encoding and decoding processes. To fix notation, we call the expected number of XORs that is required to produce an output symbol the *encoding cost* of a fountain code. The expected number of XORs required to decode the source symbols from the received output symbols is called the *decoding cost*. In terms of the computational complexity the best type of fountain codes one can envision have a constant encoding cost (independent of k), and a decoding cost which grows linearly with k. As a caveat, we would like to mention that considering the computational complexity in isolation does not make much sense; generally one has to look at all the parameters of a fountain code, i.e., overhead, computational complexity, and the error probability of the decoder. We briefly elaborate on this issue later when comparing LT-codes and Raptor codes.

In operation the output symbols need to contain indications that allow the receiver to recover the mask of each of these symbols. This is accomplished by equipping output symbols with Encoding Symbol ID's (ESI's). In the standardized Raptor code, an ESI is a 16-bit integer which facilitates the creation of the mask associated to an output symbol. Details are described in [9].

The conceptually simplest form of decoding a fountain code is the following: the receiver recovers for every received symbol y_i its corresponding mask $(a_{i1}, \ldots, a_{ik})$, and sets up the following system of linear equations:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & an2 & \cdots & a_{nk} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}. \tag{1}$$

In effect, all decoding methods for fountain codes try to solve this system of equations, either implicitly or explicitly. The task of the code designer is to design the fountain code in such a way that a particular (low-complexity) decoding algorithm performs very well. The following subsections give examples of such codes.

B. LT-Codes

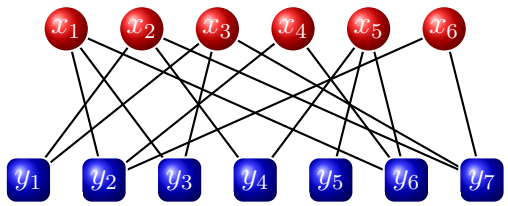
LT-codes, invented by Luby [11], are the first realization of fountain codes. LT-code exhibit excellent overhead and error probability properties. For LT-codes the probability distribution $\mathcal{D}$ has a particular form which we describe by outlining its sampling procedure. At the heart of LT-codes is a probability distribution Ω on the integers $1, \ldots, k$. This distribution is often called the *weight* or *degree* distribution of the LT-code. To create an output symbol, the following procedure is applied:

1) Sample from Ω to obtain an integer $w \in \{1, \ldots, k\}$. The number w is called the *weight* or *degree* of the output symbol.
2) Choose a binary vector $(a_1, \ldots, a_k)$ of Hamming weight w uniformly at random.
3) Set the value of the output symbol to $\oplus_{i=1}^{k} a_i x_i$.

An LT-code as described above is determined by its parameters (k, Ω). As outlined above, the output symbol is given an ESI which enables the recreation of its mask.

As with other fountain codes, LT-codes can be decoded by solving the system (1). However, in many applications, straightforward solution of this system using, e.g., a naive Gaussian elimination, is prohibitively expensive. It is therefore imperative to employ faster elimination algorithms, and design the distribution Ω such that these decoding algorithms have low overhead while maintaining

$$
\begin{aligned}
y_1 &= x_2 \oplus x_3 \\
y_2 &= x_1 \oplus x_4 \oplus x_6 \\
y_3 &= x_1 \oplus x_3 \\
y_4 &= x_2 \oplus x_5 \\
y_5 &= x_5 \\
y_6 &= x_1 \oplus x_4 \oplus x_5 \\
y_7 &= x_2 \oplus x_3 \oplus x_6
\end{aligned}
$$



y_i	x_i
5	5
4	2
1	3
3	1
6	4
2	6

Fig. 1. Toy example of an LT-code. The collected output symbols are shown on the left. In the middle part, these symbols are transformed into a graph. The array on the right side gives a "mini-schedule" for recovering the source symbols: output symbol y_5 recovers x_5; thereafter, y_4 recovers x_2, y_1 recovers x_3, etc.

stringent bounds on their error probabilities. One of the simplest elimination algorithm one can envision is the greedy one. We describe it using a graph terminology.

Upon reception of output symbols $y_1, \ldots, y_n$, we arrange them in a bipartite graph with the output symbols forming one side, and the source symbols $x_1, \ldots, x_k$ the other side. We connect an output symbol y to all the input symbols of which y is the XOR. So, if for example $y = x_1 \oplus x_5 \oplus x_9$, we connect y to the source symbols x_1, x_5, and x_9.

The decoding algorithm is a modification of the one presented in [15] and proceeds in rounds. At each round, we search for an output symbol of degree one, and copy its value into the value of its unique neighbor among the source symbols. We then XOR the value of the newly found source symbol into all the neighbors of the source symbol among the output symbols, and delete all edges emanating from the source symbol. We continue the procedure until we cannot find an output symbol of degree one. If at this point not all the source symbols are recovered, then we declare a decoding error.

In applications it is often advantageous to not perform the XOR operations in this algorithm immediately. Instead, one would use the decoding algorithm outlined to create a "schedule" (as proposed in [9, Annex C]) which stores the order in which the XORs are performed. Such a schedule has a number of advantages. For example, when interleaving is used to create multiple symbols with the same mask to be packed into a transmission packet, scheduling needs to be done only once, amortizing the cost of scheduling over the interleaving depth.

Figure 1 provides a toy example of an LT-code giving its associated graph, and a schedule which provides an algorithm for recovering the source symbols from the received output symbols.

It is by no means certain that the greedy decoding algorithm succeeds. In fact, in a well-defined sense, almost all choices for the distribution Ω would lead to algorithms with very large error probabilities even with large overheads. It can be very easily seen that if the decoding algorithm is to have an error probability that decays inversely proportional to k, then the encoding cost associated with the distribution Ω has to be of the order $O(\log(k))$ [7], [11], and the average decoding cost of a successful algorithm is of the order $O(k \log(k))$. It is remarkable that this bound can be matched with a specific design, called the "robust soliton distribution", which asymptotically guarantees small error probabilities with an overhead of the order $O(\log^2(k)/\sqrt{k})$ [11].

$$
\begin{aligned}
y_1 &= x_1 \\
y_2 &= x_1 \oplus x_5 \\
y_3 &= x_1 \oplus x_3 \oplus x_4 \\
y_4 &= x_1 \oplus x_3 \oplus x_4 \oplus x_6 \\
y_5 &= x_4 \oplus z_2 \\
y_6 &= x_5 \oplus z_2 \\
y_7 &= x_4 \oplus x_5 \oplus x_6 \oplus z_1 \\
\hline
0 &= x_1 \oplus x_3 \oplus x_4 \oplus x_6 \oplus z_1 \\
0 &= x_1 \oplus x_2 \oplus x_4 \oplus x_5 \oplus x_6 \oplus z_2
\end{aligned}
$$

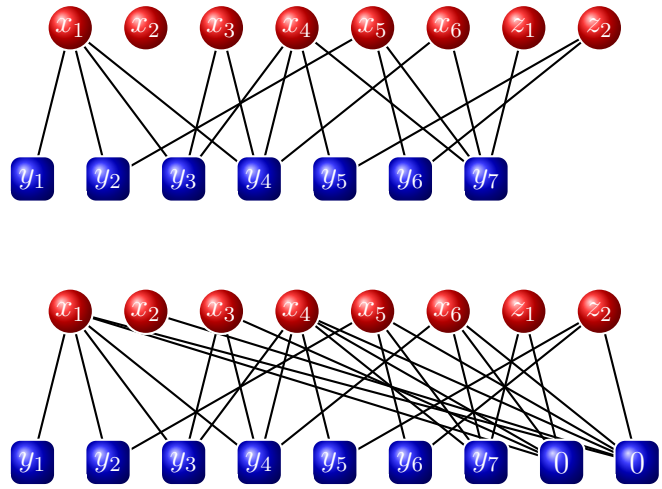


Fig. 2. Toy example of a Raptor code. The received output symbols are shown on the left, together with the relations among the input symbols dictated by the precode. The top graph is the one between the dynamic output symbols and the input symbols. The input symbols are divided into the source symbols $x_1, \ldots, x_6$ and the redundant symbols z_1, z_2. As can be seen, node x_2 is not covered and cannot be recovered. In the lower graph the static output symbols are added to the graph. The node x_2 is covered now.

C. Nonsystematic Raptor Codes

Despite the excellent performance of LT-codes, it is not possible to give a construction with constant encoding and linear decoding cost without sacrificing the error probability. In fact, a simple analysis shows that to obtain constant encoding cost with reasonable overheads, the error probability has to be constant as well.

An extension of LT-codes, Raptor codes are a class of fountain codes with constant encoding and linear decoding cost. Compared to LT-codes, they achieve their computational superiority at the expense of an asymptotically higher overhead, although in most practical settings Raptor codes outperform LT-codes in every aspect. In fact, for constant overhead ε one can construct families of Raptor codes with encoding cost $O(\log(1/\varepsilon))$, decoding cost $O(k \log(1/\varepsilon))$, and a decoding error probability that asymptotically decays inversely polynomial in k [7].

Raptor codes achieve their performance using a simple idea: the source x is *precoded* using a linear code $\mathcal{C}$ of dimension k and block-length m. The encoding of x with $\mathcal{C}$ produces a vector $z = (z_1, \ldots, z_m)$ of symbols called *input symbols*. Often a systematic encoding is used for $\mathcal{C}$, in which case $z = (x_1, \ldots, x_k, z_1, \ldots, z_{m-k})$, where $z_1, \ldots, z_{m-k}$ are redundant symbols. A suitably chosen LT-code of type (m, Ω) is then applied to z to create output symbols $y_1, y_2, \ldots$. The characterization of a Raptor code can be determined by its parameters $(\mathcal{C}, k, \Omega)$.

A toy example of a Raptor code is provided in Figure 2. In this example the check matrix of the precode $\mathcal{C}$ is equal to

$$
H = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}.
$$

Note that LT-codes form a special subclass of Raptor codes: for these codes the precode $\mathcal{C}$ is trivial. At the other extreme there are the *precode-only* (PCO) codes [7] for which the degree distribution Ω is trivial (it assigns a probability of 1 to weight 1, and zero probability to all other weights). All Raptor codes in use are somewhere between these two extremes: they have a nontrivial (high-rate) precode, and they have an intricate (though low-weight) degree distribution.

Raptor codes can be decoded in a variety of ways. The conceptually simplest decoder sets up a system of linear equations and solves the system using Gaussian elimination. The system to set up has the following shape: suppose that the code $\mathcal{C}$ has a check matrix H with m columns and $m-k$ rows. Moreover, suppose that each collected output symbol y_i has mask $(a_{i1}, \ldots, a_{mi})$, recovered using the ESI of the output symbol. In addition, let $(z_1, \ldots, z_m)$ denote the input symbols of the LT-code. Recovering these input symbols is tantamount to the recovery of the source symbols. (This is obvious if $\mathcal{C}$ is systematic, and is very easy to see in general as well.) The input symbols can be recovered by solving the system of linear equations

$$\left(\begin{array}{cccc} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & an2 & \cdots & a_{nm} \\ \hline & & H & \end{array}\right) \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{pmatrix} = \left(\begin{array}{c} y_1 \\ y_2 \\ \vdots \\ y_n \\ \hline 0 \\ \vdots \\ 0 \end{array}\right). \tag{2}$$

One can employ the Gaussian elimination algorithm to decode. This decoder is optimal as far as the success of the recovery procedure is concerned: decoding (by means of *any* algorithm) fails if and only if the Gaussian elimination decoder fails. However, the running time of this decoder is prohibitively large.

A different decoder with much lower complexity operates in the same manner as the greedy algorithm for LT-codes: The matrix in (2) is interpreted as the connection matrix between the m input symbols, and $n+m-k$ output symbols. There are n *dynamic* output symbols corresponding to the collected output symbols. The last $m-k$ *static* output symbols correspond to the precode, and the values of these symbols are set to zero. The greedy algorithm of Section II-B can be applied to this graph to recover the values of the input symbols. A modification of this algorithm has been completely analyzed in [7] and designs have been presented which show that the failure probability of the algorithm is very small even for small overheads, if k is in the range of tens of thousands.

The superior computational performance of the greedy decoding algorithm comes at the expense of large overheads for small values of k. This can be explained by the fact that for small k the variance of the decoding process is too large compared to k, and hence decoding fails more often than for large k. It seems hard to be able to control the variance for small values of k. To remedy this situation, a different decoding algorithm has been devised [16]. Called *inactivation decoder*, this decoder combines the optimality of Gaussian elimination with the efficiency of the greedy algorithm.

Inactivation decoding is useful in conjunction with the scheduling process alluded to in Section II-B and outlined in [9, Annex C]. The basic idea of inactivation decoding is to declare an input symbol as *inactivated* whenever the greedy algorithm fails to find an output symbol (dynamic or static) of weight 1. As far as the algorithm is concerned, the inactivated symbol is treated as decoded, and the decoding process continues. The values of the inactivated input symbols are recovered at the end using Gaussian elimination on a matrix in which the number of rows and columns are roughly equal to the number of inactivations. One can view Gaussian elimination as a special case of inactivation decoding in which inactivation is done at every step. Successful decoding via the greedy algorithm is also a special case: here the number of inactivations is zero.

If the number of inactivations is small, then the performance of the algorithm does not differ too much from that of the greedy algorithm; at the same time, it is easy to show that the algorithm is

optimal in the same sense as Gaussian elimination.

The design problem for Raptor codes of small length which do not exhibit a large number of inactivations is tough, but solvable to a large degree. An application of the theoretical tools used for such a design is the standardized Raptor code which is discussed in the next section, along with a description of the systematic version of these codes.

D. The Systematic Standardized Raptor Code

In a variety of applications it is imperative to have the source symbols as part of the transmission. A systematic fountain code is a fountain code which, in addition to the three conditions given in Section II-A satisfies the following properties:

1) The original source symbols are within the stream of transmitted output symbols. The output symbols not belonging to the set of source symbols are called *repair symbols*.
2) For all $0 \leq \ell \leq m$ all the source symbols can be recovered from any set of ℓ of the source symbols and any set of $n - \ell$ repair symbols, with high probability.

The straightforward idea of sending the source symbols alongside the normal output symbols of a nonsystematic Raptor code fails miserably. This is because there is large discrepancy between the statistics of the source symbols and that of the repair symbols. Instead, what is needed is a method which makes the source symbols indistinguishable from the other output symbols. With such a method, the distinction between the two disappears, and it does not matter which portion of the received symbols is source.

Such a method has been outlined in [7] and in [17]. The main idea behind the method is the following: we start with a nonsystematic Raptor code, and generate k output symbols. We then run the scheduling algorithm to see whether it is possible to decode the input symbols using these output symbols. If so, then we identify these output symbols with the source symbols, and decode to obtain a set of m *intermediate symbols*. The repair symbols are then created from the intermediate symbols using the normal encoding process for Raptor codes.

An example of a systematic Raptor code together with its encoding procedure is provided in Figure 3.

The crux of this method is the first step in which k output symbols need be found which are "decodable." This corresponds to decoding with zero overhead. A variety of methods can be employed to do this. The output symbols generated by these methods differ in terms of the error probability and complexity of the decoder. The computations corresponding to these symbols can be done offline, and the best set of output symbols can be kept for repeated use. What is then needed is an efficient method to re-produce these output symbols from a short advice, for example a 16-bit integer. The standardized Raptor code [9, Annex B] does exactly this, and provides for any length k between 1 and 8192 a 16-bit integer, and a procedure to produce the k output symbols from this integer.

Figure 4 gives a brief description of the standardized Raptor code in terms of the precode and the probability distribution Ω.

E. Performance of Standardized Raptor Codes

Several aspects need to be considered in the assessment of the power of Raptor codes. These include coding performance, complexity (especially at the decoder), and the flexibility to different use cases. A reasonable insight is obtained by comparing the standardized Raptor code to an ideal fountain code. An ideal fountain code has the property that for any number k of source symbols

$$
\begin{aligned}
z_1 &= x_1 \\
z_5 &= x_2 \oplus z_1 \\
z_8 &= x_6 \oplus z_5 \\
z_4 &= x_5 \oplus z_8 \\
z_3 &= x_3 \oplus z_1 \oplus z_4 \\
z_6 &= x_4 \oplus z_1 \oplus z_3 \oplus z_4 \\
z_2 &= s_2 \oplus z_1 \oplus z_4 \oplus z_5 \oplus z_6 \oplus z_8 \\
z_7 &= s_1 \oplus z_3 \oplus z_4 \oplus z_6
\end{aligned}
$$

$$
\begin{aligned}
y_1 &= z_2 \oplus z_4 \oplus z_5 \oplus z_7 \\
y_2 &= z_2 \oplus z_3 \oplus z_6 \\
y_3 &= z_3 \oplus z_5 \oplus z_8
\end{aligned}
$$

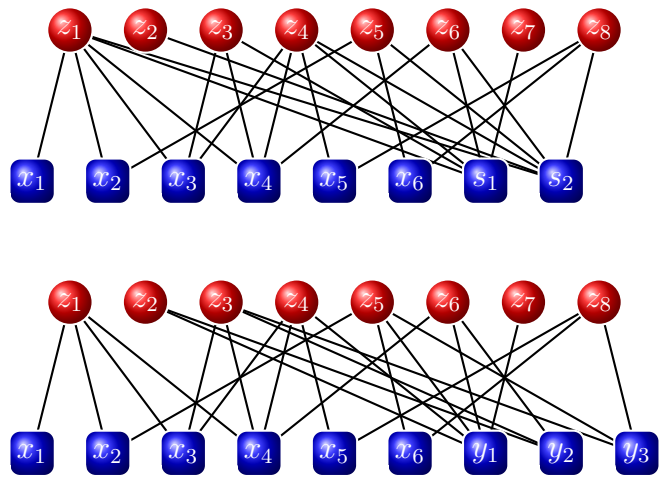
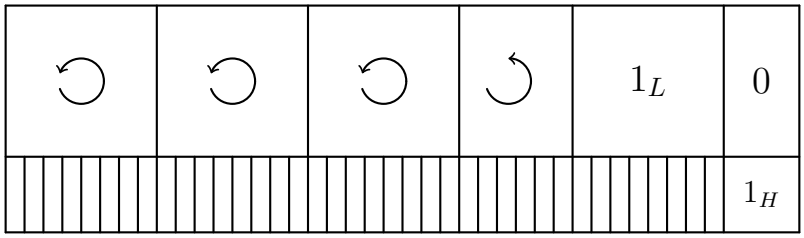


Fig. 3. Toy example of a systematic Raptor code. The source symbols are $x_1, \ldots, x_6$. The nodes with labels s_1, s_2 are obtained from the relations dictated by the precode, and their values are 0. In a first step, the intermediate symbols $z_1, \ldots, z_8$ are obtained from the source symbols by applying a decoder. The sequence of operations leading to the z_i is given on the left. Then the output symbols are generated from these intermediate symbols. Examples for three output symbols y_1, y_2, y_3 are provided. Note that by construction the x_i are also XORs of those z_i to which they are connected.



i	Ω_i
1	0.00971
2	0.4580
3	0.2100
4	0.1130
10	0.1110
11	0.0797
40	0.0156

Fig. 4. The check matrix of the precode and the LT-degree distribution Ω for the standardized Raptor code. The check matrix consists $L+H$ rows, where L is the smallest prime greater than or equal to $X + \lceil 0.01k \rceil$ where X is the smallest integer such that $X(X-1) \geq 2k$. H is the smallest integer such that $\binom{H}{\lceil H/2 \rceil} \geq L+k$. The check matrix is composed of an $L \times (k+L+H)$ matrix consisting of block-circulant matrices of row-weight 3, and block size L, an $L \times L$-identity matrix 1_L, and an $L \times H$-matrix consisting of zeros. The last circulant matrix appearing before the identity matrix may need to be truncated. The lower $H \times (k+L+H)$-matrix consists of binary vectors of length H and weight $\lceil H/2 \rceil$ written in the ordering given by a binary reflected Gray code, followed by an $H \times H$-identity matrix 1_H. The distribution Ω for the LT-code is given on the right. Ω_i is the probability of picking the integer i.

it can create any number m of repair symbols with the property that *any* combination of k of the $k+m$ source and repair symbols is sufficient for the recovery of the k source symbols. Thus, an ideal fountain code has zero reception overhead: the number of received symbols needed to decode the source symbols is exactly the number of source symbols independent of which symbols are received. Simulation results for Raptor codes provided for example in [18], [19] show that Raptor codes have reception overhead very close to ideal fountain codes.

For a file delivery session using AL-FEC, the *transmission overhead* is defined as $100*(N/K-1)$, where N is the number of encoding packets transmitted in the file delivery session and K is the number of source packets in the original file (all packets are equal size). Thus, the transmission overhead is the amount of repair data sent for the file delivery measured as a percent of the file size. During the standardization phase of MBMS, 3GPP extensively tested different alternatives to provide reliability for download delivery in 3GPP systems and measured transmission overheads. An exemplary result is provided in Figure 5: which shows the decoding failure probability versus transmission overhead when transmitting a 3MByte file encoded with the Raptor code over a MBMS UMTS bearer at different link layer loss rates p compared to an ideal fountain code. The recommended parameter settings according to [9] have been used. It is clear from the results that for these conditions the Raptor codes perform basically as good as ideal fountain codes for all loss rates. It is worth noting that the encoding symbol loss rates in general are higher than the link layer loss rates as the mapping of IP-packets to link layer packets is in general not aligned.

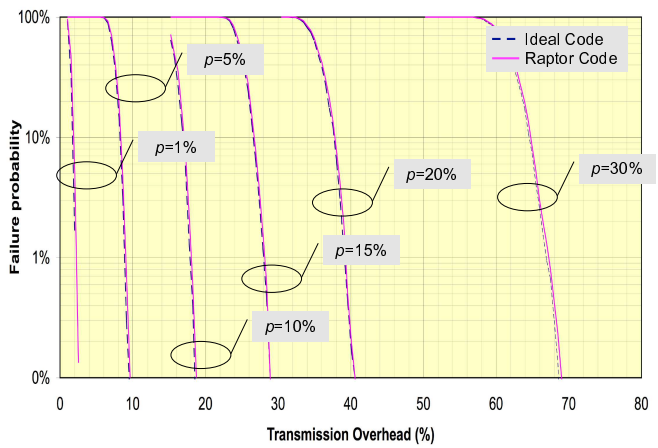


Fig. 5. Failure probability versus transmission overhead when transmitting a 3MByte file encoded with the Raptor code over a MBMS UMTS bearer at different link layer unit loss rates p compared to an ideal fountain code.

The performance of Raptor codes compared to an ideal fountain code has been investigated further. The reception overhead performance of an AL-FEC code can be expressed by the decoding failure probability $P_f(n,k)$ as a function of the source block size k and the number of received symbols n. An interesting and quite powerful estimation for the reception overhead of standardized Raptor codes has been determined as

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

Figure 6 plots the failure probability $P_f(n,k)$ versus $n-k$, where $n-k$ is the reception overhead. The figure also contains selected simulation results, which verify that the above reception overhead performance estimate is quite accurate. An experiment has been carried out where 40% of the source

data was dropped randomly and then a random $0.4k + (n - k)$ repair symbols have been chosen and the decoding failure probability is evaluated. It is observed that for different values of k, the equation almost perfectly emulates the actual reception overhead performance. For the Raptor code the failure probability for $n \geq k$ decreases exponentially with an increasing number of received symbols. The increase is so fast, that for only about 12 additional symbols the failure probability is 0.1% and for 24 additional symbols the failure probability is 0.0001%. For a typical source block sizes of $k \geq 1000$ symbols then the overhead for a 0.1% failure probability is below 1.2% and the overhead for a 0.0001% failure probability is below 2.4%.

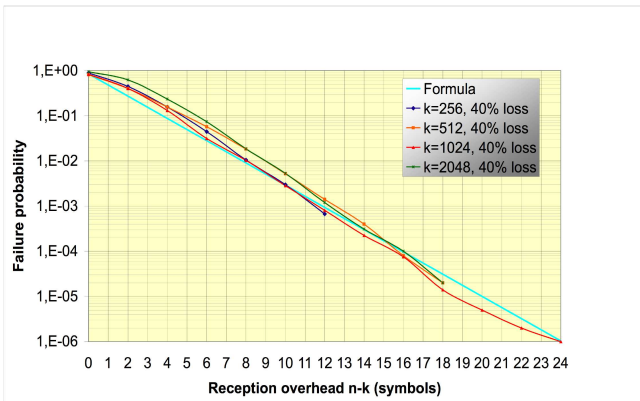


Fig. 6. Raptor source block loss rate for different source block size k, reception overhead $n - k$, and symbol loss rate 40%.

The average reception overhead of an erasure code, ε, is the average amount of additional data necessary to recover a source block. In a practical scenario, this would correspond to a receiver that requests encoding symbols as long as decoding is not successful. Based on the function $P_f(m, k)$, the average reception overhead, ε, depending on k results in

$$\varepsilon(k) = \frac{1}{k} \sum_{i=0}^{\infty} i \left(P_f(k + i - 1, k) - P_f(k + i, k) \right) = \frac{0.85}{k} \sum_{i=0}^{\infty} i \left(0.567^{i-1} - 0.567^{i} \right) = \frac{0.85}{(1 - 0.567)k}.$$

The final expression is approximately $2/k$. Therefore, the number of additional symbols is on average 2, independent of k. The average reception overhead decreases with increasing k, and for example for typical values of $k \geq 1000$ it is at most 0.2%.

In terms of complexity, the standardized Raptor codes are quite attractive. The complexity has been evaluated in the 3GPP MBMS standardization effort. For example, on a 206MHz ARM platform decoding speeds of more than 25Mbps can be supported. Compared for example to Reed-Solomon codes which operate on non-binary symbols the computational complexity of Raptor codes is orders of magnitude less. Further advantages of the Raptor codes are that the complexity is linear in the size of source data and the complexity remains linear for any packet loss rate. The memory requirements for Raptor codes are also very attractive, as for both encoding and decoding only slightly more memory is needed than the source block size.

III. Multimedia Delivery Services in MBMS

A. MBMS Architecture

MBMS is a point-to-multipoint service in which data is transmitted from a single source entity to multiple recipients. Transmitting the same data to multiple recipients allows network resources to

be shared. The MBMS bearer service offers a Broadcast Mode and a Multicast Mode. The MBMS architecture enables the efficient usage of radio-network and core-network resources, with an emphasis on radio interface efficiency. In the bearer plane, this service provides delivery of IP Multicast datagrams to User Equipments (UEs). A new functional entity, the Broadcast Multicast Service Center (BM-SC) provides a set of functions for MBMS User Services. The system architecture is shown in Fig. 7.

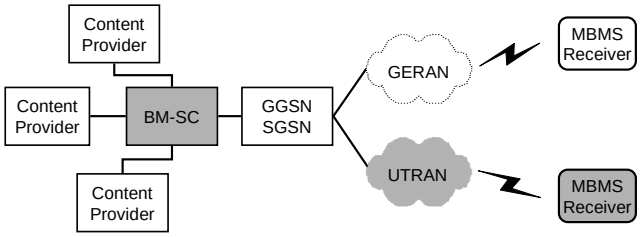


Fig. 7. Simplified MBMS system architecture.

MBMS User Service architecture is based on an MBMS receiver on the UE side and a BM-SC on the network side. Reception of an MBMS multicast service is enabled by different phases such as subscription, joining, data transfer, and leaving. In this work we focus on the data transfer phase exclusively. Furthermore, we concentrate on MBMS delivery over UTRAN and specifically focus on the mobile radio efficiency.

B. MBMS Protocol Stack

MBMS defines two delivery methods - download and streaming delivery. MBMS delivery methods make use of MBMS bearers for content delivery but may also use the associated delivery procedures for quality reporting and file repair. A simplified MBMS protocol stack focusing on data delivery aspects for streaming and download delivery is shown in Figure 8.

Streaming data such as video streams, audio programs or timed text are encapsulated in RTP and then transported over the streaming delivery network. In this case, AL-FEC is applied on UDP flows, either individually or on bundles of flows. The streaming framework provides significant flexibility in terms of code rates, protection periods, etc. [9]. Discrete objects such as still images, multimedia streams encapsulated in file formats, or other binary data are transported using the FLUTE protocol (RFC 3926 [1]) when delivering content over MBMS bearers. In both delivery services the resulting UDP flows are mapped to MBMS IP multicast bearers.

The MBMS Bearer services reuses most of the legacy UMTS protocol stack in the packet-switched domain. Only minor modifications are introduced to support MBMS. The IP packets are processed in the Packet Data Convergence Protocol (PDCP) layer where for example header compression might be applied. In the Radio Link Control (RLC) the resulting PDCP- Protocol Data Units (PDUs), generally of arbitrary length, are mapped to fixed length RLC-PDUs. The RLC layer operates in unacknowledged mode as feedback links on the the radio access network are not available for point-to-multipoint bearers. Functions provided at the RLC layer are for example segmentation and reassembly, concatenation, padding, sequence numbering, reordering and out-of-sequence and duplication detection. The Medium Access Control (MAC) layer maps and multiplexes the RLC-PDUs to the transport channel and selects the transport format depending on the instantaneous source rate. The MAC layer and physical layer appropriately adapt the RLC-PDU to the expected

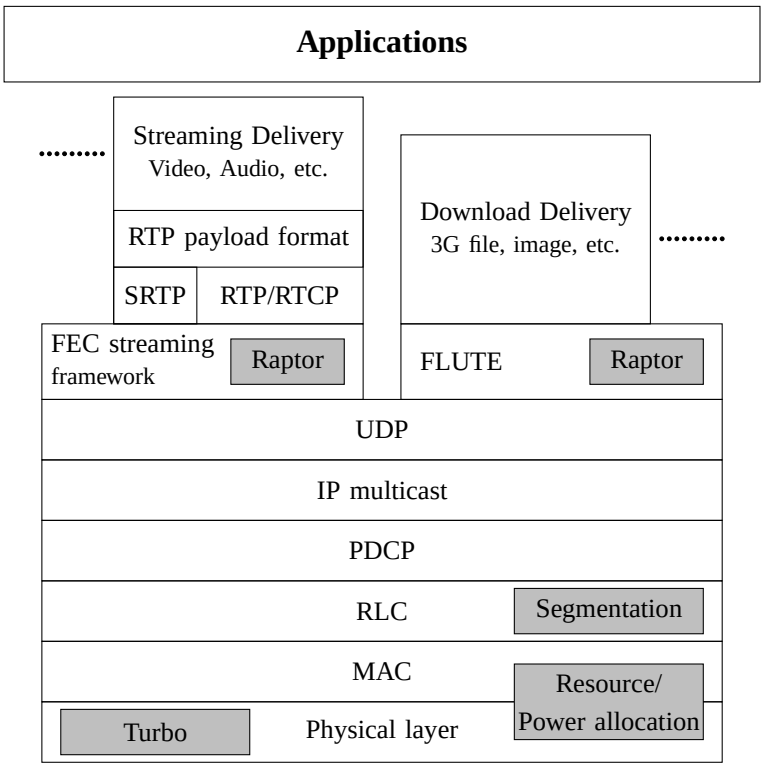


Fig. 8. MBMS protocol stack.

transmission conditions by applying, among others, channel coding, power and resource assignment, and modulation.

C. MBMS Bearer Service over UMTS

The UMTS bearer provides services with different QoS which are fundamental to support the MBMS broadcast mode. Radio bearers are specified among others by the data throughput, the data transport format, PHY-FEC, rate matching, power allocation, and many other things. MBMS uses the *Multimedia Traffic Channel* (MTCH), which enables point-to-multipoint distribution. This channel is mapped to the *Forward Access Channel* (FACH) which is finally mapped to the *Secondary - Common Control Physical Channel* S-CCPCH physical channel [20]. Among others, an MBMS radio bearer is defined by the transport format size and number of transport blocks that are to be protected by PHY-FEC at every transmission time interval (TTI). The TTI is transport-channel specific and can be selected from the set $\{$10 ms, 20 ms, 40 ms, 80 ms$\}$ for MBMS. Thereby, higher values are accomplished by longer interleaving and/or longer codeword sizes of channel code, but at the expense of higher latencies.

Refer again to Fig. 8. After RLC layer processing the resulting RLC/MAC blocks are mapped into the transport blocks according to the specified radio bearer settings and a 16 bit CRC is appended. The resulting blocks might be concatenated and then further segmented into code blocks such that the maximum length of a code block does not exceed 5114 bits [21], [22]. This limitation comes from the restrictions on complexity, memory and power consumptions of Turbo decoders in handheld devices. After Turbo coding is applied, the resulting blocks are concatenated, interleaved and eventually rate matching is performed such that the Turbo code rate, r_{inner}, can be set in the

TABLE I

BEARER PHYSICAL LAYER PARAMETERS

Channel bit rate	SF	bits per slot	data bits per slot
120 kbps	64	80	72
240 kbps	32	160	152
480 kbps	16	320	312
960 kbps	8	640	632

range $[\frac{1}{3}; 1]$. By repetition, even lower code rates than $r_{\text{inner}} = \frac{1}{3}$ can be supported. The resulting codeword is mapped to one or more transmission slots that are finally mapped into radio frames, as shown in Fig. 9. A radio frame consists of 15 slots, whereby the number of bits per slot depends on the applied spreading code. Finally, radio frames are transmitted every 10 ms using a chip-rate of 3.84 Mcps and QPSK modulation.

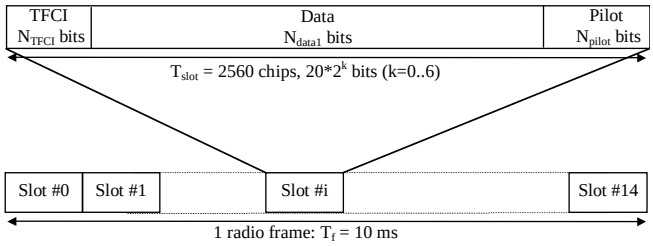


Fig. 9. Structure of the UMTS radio frame.

For interoperability and testing purpose a number of MBMS reference radio bearers have been defined as a preferred configuration [23] for a specific data rate. The configurations define a single default radio parameter set including Turbo code rates, transport format, transmission time interval, symbol rate, etc. However, for our purposes to leave some flexibility in investigating the trade-off between the Turbo code and the Raptor code we only fix a subset of parameters of selected MBMS bearers in Table I. In contrast to specified bearers, we allow the Turbo code rate, r_{inner}, to be adjusted so that trade-offs between the Turbo code and the Raptor code can be analyzed. Note that MBMS allows use of configurations other than those specified in [23], and in particular all configurations we apply in the following fully comply with the specification.

To appropriately compare different settings, we specify an MBMS bearer not by its data rate at the RLC layer, but instead by the symbol rate at the physical layer. The considered bearers with their respective settings are provided in Table I. The variation of the Turbo code rate results in varying RLC bit rates: higher Turbo code rates offer higher data rates but also result in higher RLC-PDU loss rates, whereas lower Turbo code rates result in lower loss rates, but also offer higher RLC data rates. A primary purpose is to investigate combinations of code rate settings for Raptor codes used at the application layer and Turbo codes used at the physical layer that optimize the overall use of the network.

At the receiver side, the inverse operations are applied. Specifically, if CRC for a transport block fails after Turbo decoding, the block is considered erased. All correct data is delivered to the upper layers. At the RLC layer, only correctly received RLC-SDUs (usually containing an entire IP packet) are delivered to the higher layer. Therefore, any IP packet which is partly contained in an erroneous transport block is lost and is not available at the receiver.

MBMS is generally applied in a multi-cellular environment. If the same MBMS services is offered

synchronously in not just a single cell, but in an entire area, then the receiver performance especially at cell edges can be improved. A mobile terminal can perform combining of different signals, either on the physical layer in the form of *soft combining* or on the RLC layer referred to as *selective combining*. In these cases an MBMS UE listens to more than one Node-B signals simultaneously, and for selective combining, it individually decodes the streams in the hope that for at least one signal passes passes the CRC such that the correct RLC-PDU can be forwarded to the upper layer. However, any combining scheme increases the complexity and costs of a receiver and therefore might not be used in initial deployments of MBMS.

Mobile radio bearers can be configured in a quite flexible manner. For the IP multicast bearers in MBMS, the parameters in Table I provide insight into the configuration. On top of this, as already mentioned, also the Turbo code rate r_{inner} can be modified. The quality of such bearer configurations can basically be evaluated by their supported bitrate on RLC layer and the observed RLC-PDU loss rate. Whereas the bitrate is a transmitter configuration, e.g. a combination of the bearer parameter in Table I and the Turbo code rate, the observed loss rate significantly depends on the position and mobility of the user under investigation. Furthermore, the observation window for the loss rate measurement is quite important in the interpretation of the loss rate. The long-term loss rates might be quite different than those being observed over a shorter period.

With the use of Raptor codes, higher loss rates can be compensated by the transmission of additional repair symbols, ensuring that all transmitted data that is useful for recovery of the original source data. In this case a good measure for the overall network performance is the so called *goodput*, defined as the supported bitrate multiplied by one minus the packet loss rate of the user measured over some window of time. A goodput measurement represents the average received amount of data over a window of time, and a sequence of goodput measurements can be continually varying depending on the changing average loss rate within different windows of time. Variations of goodput measurements depend not only on the transmitter configuration and the user mobility, but also on the observation window for measuring the goodput: In general, the smaller the observation window for measuring the goodput the higher the variance of the measured goodputs. Some selected measurements of goodput distributions for different MBMS bearer settings are provided in Section V.

D. MBMS Download Delivery Service

To deliver a file[1] in a broadcast session, FLUTE provides mechanisms to signal and map the properties of a file to the Asynchronous Layered Coding (ALC) [24] protocol such that receivers can assign these parameters to the received files. The 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. Both parameters T and k are signaled in the session setup and are fixed for one session. For each source block, additional repair symbols can be generated by applying Raptor encoding as explained in detail in Section II. Each encoding symbol, i.e. a source symbol or a repair symbol, is assigned a unique encoding symbol ID (ESI) to identify the symbol and its type. With respect to the symbol type, if the ESI is less than k then it is a source symbol, otherwise it is a repair symbol. Let us denote the total number of encoding symbols to be transmitted as n and define the resulting Raptor code rate as $r_{\text{outer}} \triangleq k/n$. One or more encoding symbols of the same type with consecutive ESIs are placed in each FLUTE packet payload. The source block number as well as ESI of the first encoding symbol in the packet are signaled in the FLUTE header. FLUTE packets are encapsulated in UDP and distributed over the IP multicast MBMS bearer.

[1]For simplicity we continue with the notion of a file in the following though the ALC/LCT concept uses the more general terminology *transport object*.

Receivers collect received FLUTE packets containing encoding symbols, and with the information available in the packet headers and the file session setup, the structure of the source block can be recovered. If no more encoding symbols generated from the source block are expected to be received, the Raptor decoder attempts to recover the source block from all received encoding symbols. Due to heterogeneous receiving conditions in a broadcast session, the amount as well as the set of received encoding symbols differs among the receivers. If all source blocks belonging to the file are recovered at a receiver, the entire file is recovered. If file recovery fails, a post-delivery repair phase might be invoked. With the download delivery protocols in place, different services can now be realized.

Scheduled Distribution without Time-Limits: In a scheduled broadcast service, files are distributed once within a session and all users join the session at the very beginning. The costs of such a service in mobile cellular systems is appropriately measured by the consumed resources on the physical layer, which comprise of the bandwidth share, the transmit power, and duration of the session. For simplicity, we consider the case of distributing a single file. Assuming that we fix the bandwidth share, a suitable single measure for the costs associated to transmitter is the product of the assigned transmit power for such services and the "on air time" for the distribution of the file. The product results in the necessary energy, E, to distribute the file. Secondary aspects such as the experienced "download time", i.e., how long it takes to receive the file, are generally not essential in this use case as it is assumed that the distribution is not time-critical.

In terms of user perception, file download delivery is to a large extent binary, i.e. for each user the file is either fully recovered and the user is satisfied or the file is not fully recovered and the user is unsatisfied. Clearly, not all users can always be satisfied, and file distribution services are usually operated such that a certain percentage of users are satisfied. Unsatisfied users are not necessarily excluded in the MBMS download service, and may rely on post-delivery methods to complete the file recovery. We evaluate the necessary system resources in terms of the required energy to satisfy at least a certain percentage of the user population in the MBMS service area. As a reasonable number, the support of 95% of the user population is the objective.

Time-Constrained Distribution: In a second service scenario we consider scheduled distribution with the additional constraint that files of a certain aggregate size need to be distributed within a certain amount of time. For simplicity, we consider the case of distributing a single file. In this case it is of interest to evaluate the radio resources required to transport a file of a certain size, and/or the maximum size file that can be transmitted in a certain amount of time. In the latter case, the ratio of the maximum supported file size and the allowable delivery duration also expresses the maximum supported bitrate within the time and resource constraints.

An example for such as service is the following: Assume that a provider offers the possibility to purchase a song which is played on a regular analog or digital radio program right after the song is played. To enable this, the song is distributed via a download delivery bearer, for example within an MBMS system, and is available to all users. The user can then select to purchase the song, i.e. unlock it, or not. In this case the song must be delivered within the on-air time. For the case that radio resources are restricted to a certain maximum, the efficiency of the system determines the maximum bitrate of the compressed song which relates to the quality of the media stream.

Carousel Services: File delivery using carousel is a possibly time-unbounded file delivery session in which a fixed set of files are delivered. Two types of carousel services are distinguished, static and dynamic. Whereas the former delivers only fixed content within the file, in dynamic file carousels individual files may change dynamically. When using FLUTE the file delivery carousel is realized as content delivery session whereby file data tables and files are sent continuously during

a possibly time-unbounded session. In case no AL-FEC is available, the data must be repeated. In the case that the Raptor code is used, the data transmitted for a given file generally includes repair symbols generated by Raptor encoding in addition to the original source symbols. In particular, file reception time is minimized if symbols are never repeated until all 65,536 possible symbols (source and repair) have been sent. With this, the fountain property of the Raptor codes can be optimally exploited.

In terms of system configuration, the transmitter has only limited options, basically only the transmit rate can be selected. However, of interest for the receiver is the amount of time it takes to acquire the file. The objective is to minimize the time that it takes for a receiver to acquire all the files, with a given probability, when joining the stream at some random time. Typically, the acquisition times for 95% of the users is a reasonable measure, but in this case also the average reception time provides an interesting service quality measure. The performance of carousel services and also more advanced carousel services which allow Video-on-Demand-like services over broadcast channels have for example been introduced in [25].

E. MBMS Streaming Delivery Service

Real-time MBMS streaming services mainly target classical Mobile TV services. For these services the MBMS FEC streaming framework including the Raptor codes [2], [9] plays an important role, see Figure 8. The FEC streaming framework operates on RTP packets or more precisely on UDP flows, incoming on the same or different UDP ports. In video streaming applications, these RTP packets generally include H.264 Network Abstraction Layer (NAL) [26] units and/or audio packets. It has been proven beneficial to apply NAL unit fragmentation such that RTP packets do not exceed the size of the underlying RLC frame size. As shown in Figure 8, the FEC streaming framework is based on top of the UDP layer. The legacy RTP packets and the UDP port information are used in order to generate Raptor repair symbols. Original UDP payloads become source packets by appending a 3 byte FEC source payload ID field at the end of each UDP payload. These packets are then UDP encapsulated and transported on the IP multicast bearer.

As shown in Figure 10 a copy of these packets is forwarded to the Raptor encoder and arranged in a source block with row width T bytes with each consecutive packet starting at the first empty row. The source symbol starts at the beginning of a new row, but it is preceded by a 3-byte field containing the UDP flow ID (1 byte) and the length field (2 bytes), both of which are part of the source symbol. If the length of the packet plus the 3-byte field is not an integral multiple of the symbol length then the remaining bytes in the last row are padded out with zero bytes. The source block is filled up to k rows, where the value of k is flexible and can be changed dynamically for each consecutive source block. The selection of k depends on the desired delay, the available terminal memory and other service constraints.

After collecting all packets to be protected as a single source block, the Raptor encoder generates $n - k$ repair symbols of size T as described in Section II, where the selection of n depends on how much loss is to be protected against. The generated Raptor repair symbols can be transmitted individually or as blocks of symbols in the payloads of UDP packets, called repair packets. Each source and repair packet contains sufficient information such that a receiver can use Raptor decoding to recover a source block if enough encoding symbols are received for that source block.

There are a large number of system parameters which can be adjusted to fulfill certain utility functions. There are a significant number of options for resource and Quality-of-Service optimized system configuration for an operator that runs Mobile TV services using MBMS. In terms of *system resources*, an operator can choose radio bearer configurations as discussed previously.

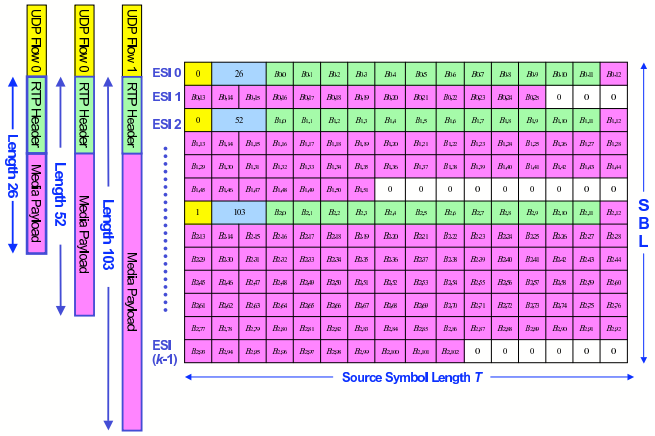


Fig. 10. MBMS FEC streaming framework.

In addition to the radio parameters for the IP multicast bearer, within the streaming delivery service, one can basically select the following parameters:

- The settings of the Raptor code parameters, mainly (i) the Raptor code rate,$r_{\text{outer}} \triangleq k/n$, which determines, together with the physical layer settings, the available bitrate for the application, and (ii) the protection period T_{PP} which determines the efficiency of the code, but also influences the end-to-end and tune-in delay.
- The video coding parameters, mainly determined by the bitrate and quality tradeoff Q_{enc}, as well as the error-resilience and tune-in properties determined by the random access point frequency, and in case of H.264/AVC determined by the instantaneous decoder refresh (IDR) frame distance T_{IDR}.

The selection of the parameters should be such that the user satisfaction is maximized whereas the usage of system resources is minimized. The target for an operator is user satisfaction for as many users as possible in the serving area, whereby the environment as well as the user behavior such as mobility also influences the reception quality.

In contrast to the download delivery service, the definition of user satisfaction is more complex for mobile TV services. The service quality from the user perspective is mainly determined by the video quality, whereby it is essential to understand that both, the *error-free video quality*, Q_{enc}, as well as the degradation due to errors matter. A reasonable service quality is only achieved if the encoded video has at least a certain encoded video quality $Q_{\text{enc,min}}$ and if errors only occur infrequently, i.e. if the *video quality degradation*, $\overline{D}_{\text{dec}}$, due to errors does not exceed a certain value $\overline{D}_{\text{dec,max}}$.

If only a single service is offered, the user perception might be slightly influenced by the tune-in time, but this aspect is usually of less relevance. However, in case MBMS is used for Mobile TV services with multiple channels, then an important service parameter is also the time how long it takes to tune into a program or how long it takes to switch between different channels of the mobile TV service. For our system configurations, tune-in and zapping times are identical and therefore we focus on the notion of tune-in time, $T_{\text{tune-in}}$.

IV. MBMS System-Level Simulation

A. Motivation

As discussed in Section III, mobile broadcast services just as any mobile multimedia service allow for a significant amount of system parameter settings. However, in p-t-p transmission systems the concept of Quality-of-Service (QoS) provision of the lower layers for the higher layers is quite established and also reasonable as the quality on the lower layers can be controlled by frequent feedback messages, adaptation to changing channel conditions, retransmissions, acquisition control, or other means. Therefore, it is quite reasonable to optimize each layer individually, or at least it is not necessary to do a full end-to-end and across layer evaluation and optimization. In contrast, mobile broadcast systems do not provide any of these fast QoS control mechanisms. The overall performance depends significantly on the settings on different layers, and efficient and optimized parameter configurations in different layers can only be obtained by understanding the service from end-to-end perspective and across all layers from the physical layer up to the media coding layer. Mobile broadcast systems require cross-layer evaluation and optimization to fully exploit their potentials.

In addition to the comprehensive end-to-end approach, user behavior and mobility primarily resulting in varying channel conditions in various time scales need to be taken into account. As in general many user consume a multimedia broadcast service in parallel, the heterogeneity of the reception conditions of different users influences the service quality. To meet the requirements and expectations of this rather complex system design, comprehensive end-to-end system level simulations are necessary. We have taken this approach to motivate the benefits of Raptor codes in mobile multimedia broadcast systems, specifically in MBMS. The basic concepts, the applied simulation framework, as well individual components of the system level simulation are introduced in more detail in the following.

B. Modeling and Simulation of MBMS IP Multicast Bearer

For the modeling of the IP multicast bearer a comprehensive approach on propagation, interference, multiuser, physical layer, as well as protocol stack modeling is proposed. Figure 11 provides insight into this approach. The simulator is composed of different modules which simulate and/or model different components of the entire system. It is divided in two blocks, the mobile cellular channel model and the radio protocol stack including the Turbo code. The cellular channel model generates traces for the carrier, interference and noise present at the mobile terminal and also the observed orthogonality factor (required to compute self-interference), with a resolution of 2 ms and for as many as N different users with different random initial position, whereby each trace corresponds to values captured over 10 minutes. The number of users in our case is $N = 500$.

These traces are generated for normalized transmit power and no spreading gain and are subsequently modified to obtain an effective SINR for each TTI by applying some appropriate combining, referred to as *Equivalent SNR Method based on Convex Metric* (ECM), power assignment and spreading code assignment. A resulting SINR is obtained for each TTI, which is converted to a sequence of RLC-PDU loss traces by applying a suitable table lookup for the Turbo code. The resulting RLC-PDU loss traces for each individual user are then applied to an IP multicast stream.

Mobile Radio Channel in Multicellular Environment: The mobile radio channel places fundamental limits on the performance of a wireless communication system. Unlike wired channels which are more stationary and predictable, radio channels show extremely varying behavior. In fact,

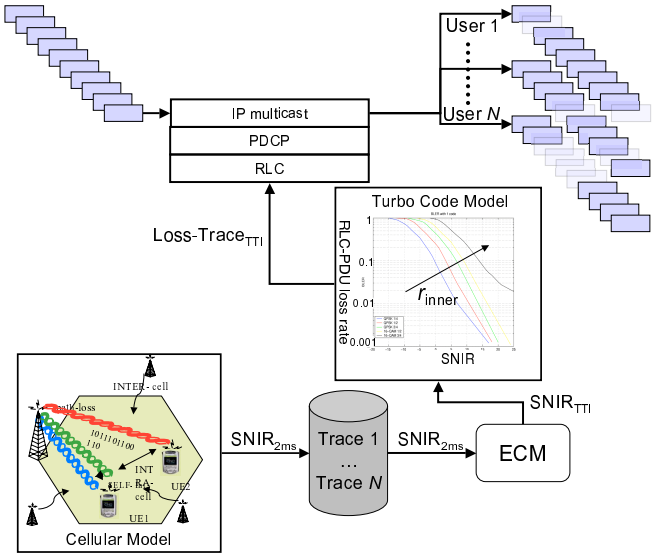


Fig. 11. Concept of simulation setup for MBMS IP Multicast Delivery over UMTS.

the radio transmission path between the transmitter and the receiver can vary from a simple line-of-sight to one that is severely obstructed by buildings, mountains, foliage, etc. Also the speed of the mobile terminal has a great impact on the received radio signal. In order to evaluate the impact of the mobile radio in a multicellular environment, a channel model that uses standard models and techniques has been defined and developed in 3GPP [27], partly based on real measurements. In particular, effects such as *path loss*, *Doppler spreads*, *shadowing*, *antenna radiation pattern* and *interference* are taken into account in this simulation setup.

This channel model allows to simulate *pedestrian* and *vehicular* mobile users within a cell, whereby the main difference between these users is the speed at which they move within the cell, but also their power-delay spectra. Fig. 12 shows examples of movements of users in a cell for different speeds and different starting points. The applied movement model is based on random walk with high directional correlation. The users do not leave the area, but bounce at the cell edges. However, possible handover effects are simulated as the signal is not necessarily received from the base station assigned to the hexagonal cell, but from the strongest one. The figure shows the position of four different users for a time of 10 min. Notice that the vehicular user undergoes a much larger distance due to his speed of 30 km/h, while the pedestrian users at 3 km/h covers less distance in the same amount of time. The right hand side shows the *signal to interference and noise ratio* (SINR) for each of the users. The SINR is varying due to large scale effects such as attenuation and shadowing on both, useful signals and interferers, as well as due to short-term effects such as fading and Doppler. Note also that for the vehicular user, the SINR shows faster variations than for pedestrian.

ECM Method: The generic mobile radio simulator computes the signal to interference and noise ratio every 2 ms which is for example the TTI in HSDPA. However, the investigated MBMS bearers use transmission time intervals of up to 80 ms. Therefore, an appropriate conversion of the effective SINR for every TTI is required. This is achieved using a *link error prediction method* called *Equivalent SNR Method based on Convex Metric* (ECM), as defined in [27]. This technique allows to combine several SINR values into a single effective SINR which is equivalent to the

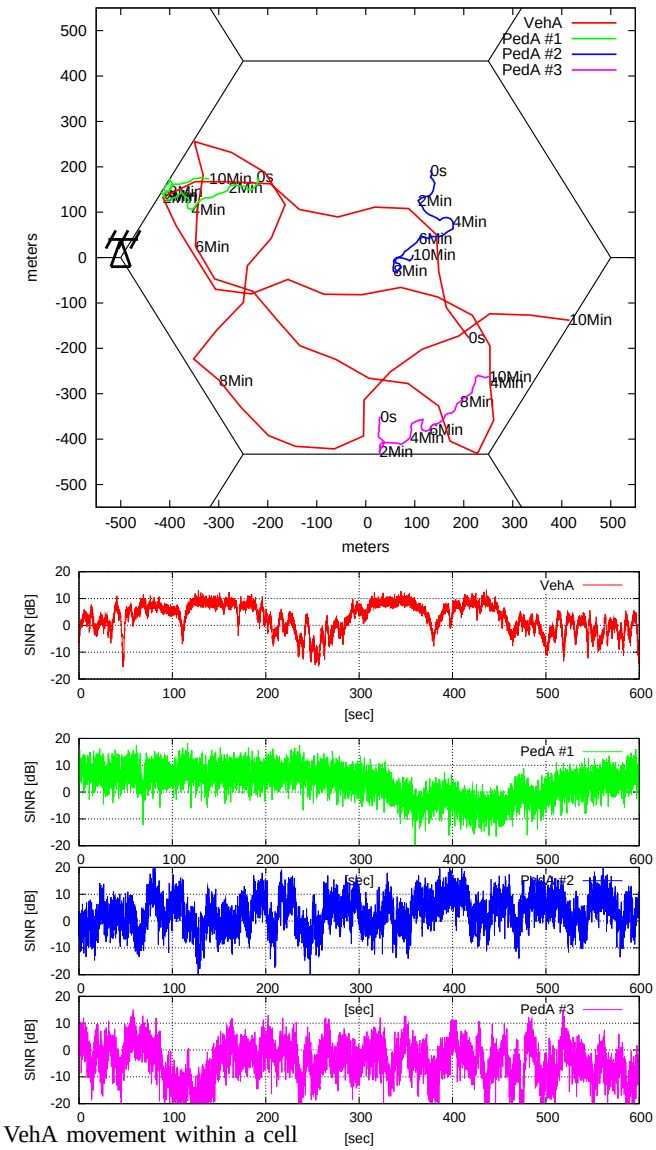


Fig. 12. Example of PedA and VehA movement within a cell

channel decoder in case that interleaving over these multiple channel access slots is provided. The method is based on Shannon's channel capacity formula and is processed in the following steps:

1) Compute the channel capacity C_i for every TTI ($i = 1, 2..., n$)
2) $\bar{C} = \frac{\alpha}{n} \sum_{i=1}^{n} C_i$
3) Compute SNR_{eff} such that $C(\text{SNR}_{eff}) = \bar{C}$

whereby the factor α is a *correction factor* that depends on the mobile speed, the interleaver, etc. For low mobile speeds and the almost ideal UMTS interleavers it has been found [27] that $\alpha = 1$ is an appropriate value.

Power and Spreading Code Assignment: As already mentioned, the traces are generated for a normalized transmit power. However, the MBMS bearer might get assigned different power resulting in different SINR values. Appropriate transmit power adjustments results in increase or decrease of the effective SINR. Furthermore, changing the MBMS radio bearer parameters, in particular the

spreading factor, leads also to different SINR values. Since the individual values for the carrier, the interference and the noise are stored in the trace files, it becomes quite easy to update the carrier power and re-compute the effective SINR by

$$\text{SINR}_{\text{eff}}\left(C, I, N, OF, SF\right) = 10\log_{10}\frac{C \times SF}{C \times (1 - OF) + I + N},$$

whereby C represents the carrier power, I the interference power, N the noise power, OF the orthogonality factor and SF the spreading factor.

Physical Layer FEC Modeling: In system level simulation, the loss probability of the Turbo code is determined by table lookups which map the effective SINR to the loss probability. Based on this loss probability, a random generator decides whether the included RLC/MAC block is decodable or not. However, for codes with different code rates, this still requires a significant amount of link-level simulations, as each possible code rate needs to be simulated. Luckily, Turbo codes as applied in UMTS have the property that for a given SINR and a given code rate r_{inner}, the decoder is either almost always able to decode or it almost always fails. The so-called "water fall" region of long Turbo codes is rather narrow. The waterfall region for practical Turbo codes coincides quite well with the computational cutoff rate $R_0(\text{SINR}) = 1 - \log_2\left(1 + e^{-\text{SINR}}\right)$ in a sense that if the code rate of the code is below the cutoff rate for this specific SINR, decoding is successful and otherwise it fails. Note the above equation is valid for BPSK transmission as well as for each component in case of QPSK transmission. Therefore, after for each TTI, the mobile channel simulator computes the effective SINR. Based on this value and the applied code rate, the RLC-PDU are either assumed error-free or are lost.

C. System Level Simulation of MBMS Download Delivery

For the simulation of MBMS Download Delivery, the MBMS Download Delivery CDP including the Raptor code is simulated over different MBMS IP multicast bearer as shown in Figure 13. The service quality, the transport of a file using the FLUTE protocol and Raptor is simulated. Thereby, for each of the N users, and for different Raptor code rates, it is evaluated if the file can be recovered. More precisely, for each of the N users, it is evaluated, how many repair symbols are necessary to send for each of the N users to recover the file. As soon as sufficient user satisfaction is achieved, e.g. as in our case 95% of the users have recovered the file, we assume that the distribution of the file is stopped. For each of the different IP multicast bearer configurations, this value is evaluated and the necessary energy, i.e. the download time of the 95% user multiplied by the power is used as a criteria for the goodness of the configuration. An important aspect in the assessment of the service is also the file size that may vary from for example 32 kByte up to several tens of MBytes. A representative but still rather small value has been selected, namely a 512 KByte file in our simulations. This might correspond to a short multimedia clip, a still image or a reasonably sized ring tone.

D. System Level Simulation of MBMS Streaming Delivery

In a similar manner as for the download delivery, also streaming delivery over MBMS is evaluated. The concept of the simulation approach is shown in Fig. 14. The MBMS IP multicast bearer simulation is composed in the same manner as for the MBMS Download Delivery. Different is only the CDP simulation. For the evaluation of the MBMS Streaming User service, several 3GPP tools available in [28] have been used. The following procedure is applied: Initially, for a certain

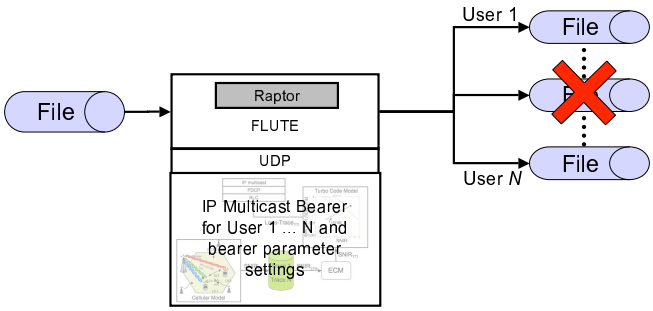


Fig. 13. Simulation setup for MBMS Download Delivery.

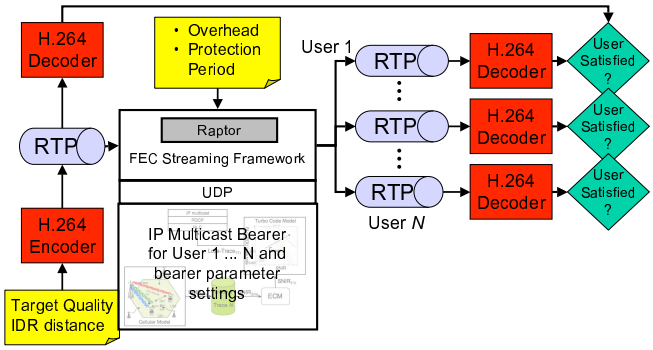


Fig. 14. Concept of simulation setup for MBMS streaming delivery over UMTS.

setting of IDR frame distance and a target quality, an encoded video stream encapsulated in RTP is generated and stored in RTP dump format. The different IDR frame distances result in different bitrates, but the streams have the same quality. The applied encoding follows a rather strict constant bit-rate rate control, but the bitrate still might fluctuate in ranges of several percent within each group-of-picture (GOP). Raptor encoding is applied to the generated RTP streams to generate a certain number of repair symbols for each source block consisting of a certain number of source symbols. Thereby, the Raptor code rate r_{outer} is selected such that together with the setting of the Turbo code rate, r_{inner}, the bearer resources are optimally used. A protection period, T_{PP}, is selected such that RTP packets within a protection period are collected in a single source block, and the source block size k may be varying slightly depending on the video statistics. After applying the IP packet loss pattern resulting from the MBMS bearer configuration, the resulting video stream is decoded and is compared to the reconstructed stream without any errors to obtain the percentage of Degraded Video Duration (pDVD) $\overline{D}_{\text{dec}}$ for this stream. The pDVD $\overline{D}_{\text{dec}}$ is used check, if the user is satisfied, the pDVD $\overline{D}_{\text{dec}}$ shall not exceed 5%. This experiment is repeated for all N users to obtain the percentage of satisfied users for the specific parameters applied. In addition, the received stream is evaluated in terms of average tune-in delay by assessing tune-in at each RLC-PDU and measuring the resulting necessary delay to display the first correct IDR frame and to ensure the display of all remaining frames without any jitter. The resulting average tune-in delay $\overline{T}_{\text{tune-in}}$ is obtained by averaging over all RLC-PDU positions and all satisfied users.

V. Selected Simulation Results

The simulation concept and details presented in Section IV allow to simulate the performance of download and streaming delivery services in MBMS. Extensive system level simulations are performed in order to evaluate system performance and specifically the trade-off between AL-FEC based on Raptor codes and PHY-FEC based on Turbo codes.

Performance Evaluation of Radio Bearer Settings: To get some insight in the performance of different radio bearer settings, we evaluate the distribution of the goodput for different system parameters assuming users being randomly placed and randomly moving in the service area. In Figure 15 we show cdf of the goodput for a bearer with 240 ksps, different transmit powers, and Turbo code rate $r_{\text{inner}} = 0.33$ and $r_{\text{inner}} = 0.67$ for an observation window of 10 minutes. The receivers do not use any receiver combining. It can be observed from the figures that the maximum value of the goodput is determined by the Turbo code rate, as expected. Higher Turbo code rates result in higher throughputs at the expense of higher error rates. However, the error rates are not that severe and comparing the two diagrams and 95% of the users satisfied, then with $r_{\text{inner}} = 0.67$ instead of $r_{\text{inner}} = 0.33$ the same goodput can be achieved with $P = 2$W instead of $P = 16$W. Therefore, if a CDP can make use of these bearer and physical properties, significant system benefits can be expected.

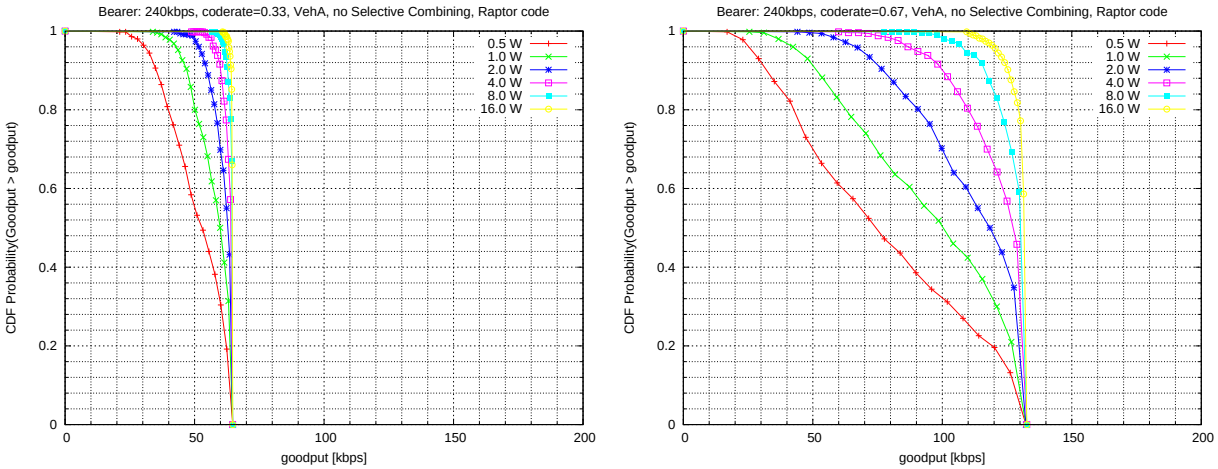


Fig. 15. CDF of goodput for a 240kpbs bearer with coder ate $r_{\text{inner}} = 0.33$ (left) and coder ate $r_{\text{inner}} = 0.67$ (right) for Vehicular A mobility model and for Raptor decoding without combining

Figure 16 shows the corresponding results when selective combining is used. It is clear that the goodput is significantly improved, but similar as for the case without combining, the same goodput can be achieved by using higher Turbo code rate of $r_{\text{inner}} = 0.67$ and the power can be reduced to $P = 1$W to have a 95% support which is as good as for a code rate $r_{\text{inner}} = 0.33$ and $P = 16$W. These findings are exploited in the crosslayer design for the delivery services in the following.

A. Performance of Download Delivery

To assess the performance of download delivery, the approach as described in subsection IV-C has been applied to a selected parameter set. The chosen bearer supports 240 kbps at the physical layer. Simulations are run for $N = 500$ users whereby their starting position is randomly and uniformly distributed over the cell area. These users are simulated for vehicular and pedestrian mobility and

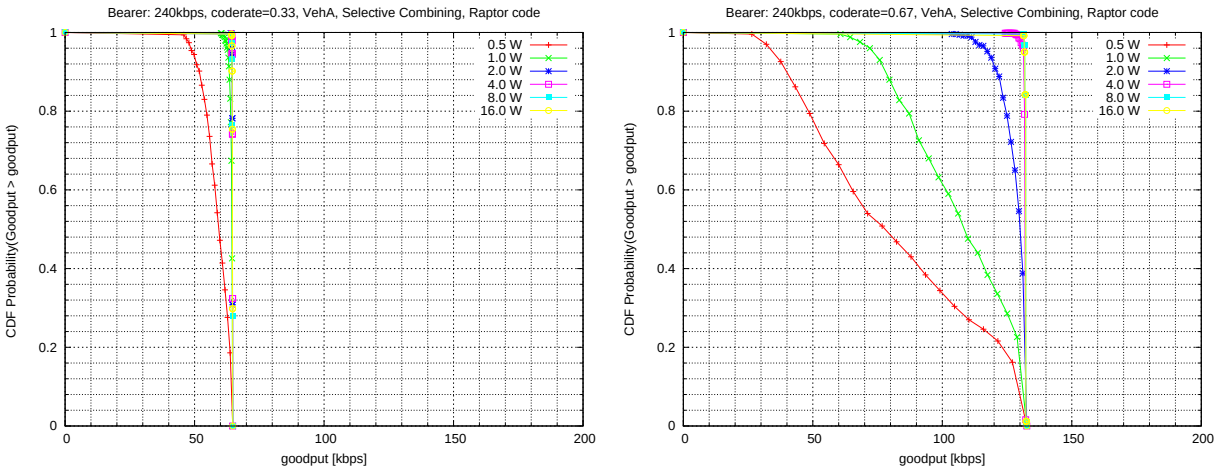


Fig. 16. CDF of goodput for a 240kpbs bearer with code rate $r_{\text{inner}} = 0.33$ (left) and code rate $r_{\text{inner}} = 0.67$ (right) for Vehicular A mobility model and for Raptor decoding with selective combining

propagation model. We also compare receiver performance with and without selective combining. Noteworthy, many more simulations than shown in this section have been carried out, and the results show a reasonable and also representative selection.

In the assessment of different system configurations, basically two aspects are of major interest, user perception of the multimedia delivery as well the resources consumed on the physical layer. The latter is most suitably expressed by the necessary energy, E, to distribute the file. We evaluate the necessary system resources in terms of the required energy to satisfy at least 95% of the user population for different system parameter configurations. We investigate different settings of Raptor code rates and Turbo code rates and transmit power assignments. For intuitive interpretation of the results, we present the RLC-PDU loss rate of the worst supported user and the necessary energy to support this user.

Fig. 17 and 18 show the necessary energy over the resulting RLC-PDU loss rate for different transmit power assignments for the MBMS service. Vehicular users only and pedestrian users only, both with and without selective combining at the receiver, are assessed. The curves are generated by applying different inner code rates r_{inner} and applying as much Raptor encoding as necessary to ensure that 95% of the users are satisfied. The curves generally terminate on the left due to the restriction on the Turbo code rate of 0.33; the leftmost point corresponds to lowest RLC-PDU loss rates and therefore to lower Turbo code rate, while the rightmost point of the curve corresponds to higher RLC-PDU loss rate and therefore to a higher Turbo code rate.

From the simulation results it is apparent that there are some optimum system configurations that minimize transmit energy. Generally, the optimum is at rather high RLC-PDU losses and are not achieved when using the lowest Turbo code rate 0.33. For example, in Fig. 17, if the system allocates 4 W of transmit power for MBMS service, the optimal RLC-PDU loss rate for minimal required delivery energy is about 40%. If stronger Turbo coding is applied, the RLC-PDU loss rate decreases. However the throughput at the RLC layer also decreases as already elaborated in the goodput evaluation results. This leads to an increased download delivery time and consequently to more required energy.

If a Turbo code rate of 0.33 is chosen, the required energy for successful delivery is about 60% higher than in case of the optimum configuration. However, if the Turbo code rate is too high then the resulting higher bitrates cannot be compensated by the increasing RLC-PDU loss rate, i.e. this

leads to increased download delivery time and consequently higher required energy. These results suggest that using the Raptor code with a low code rate at the application layer and working at rather high RLC-PDU loss rates is overall very beneficial for the system resources and reduces the overall required energy for the file distribution. By the use of Raptor coding the goodput maximization can be exploited. Another interesting observation is that transmission with lower transmission power is advantageous. In all the presented results, transmission with 0.5 W always results in the minimal required energy. Although even lower transmit powers might provide even better performance, other effects such as frequent loss of synchronization or very long on air times would be counterproductive.

Selective combining, if applicable, has impact on the required energy and increases the system capacity significantly. In Fig. 17 (right) the minimum required delivery energy for 0.5 W less than half the energy required for the corresponding case without selective combining. This was also already predicted by the goodput results. The RLC-PDU loss rates for optimal energy delivery with selective combining is lower mainly due to lower download time, not due to the use of a different Turbo code rate. Therefore, receivers with and without selective combining can quite well coexist and should be operated with similar system parameters. Note, however, that the loss rates for optimal system operation points with the use of selective combining are still in the range of 15% to 25%.

When comparing vehicular and the pedestrian mobility scenarios, we conclude that less energy is required to deliver a file if the users are moving at higher speeds, i.e. use vehicular model. This can be explained as higher mobility results in higher diversity gains. When a pedestrian user is in a deep-fade, it remains in this situation for longer time than a vehicular user, which moves faster.

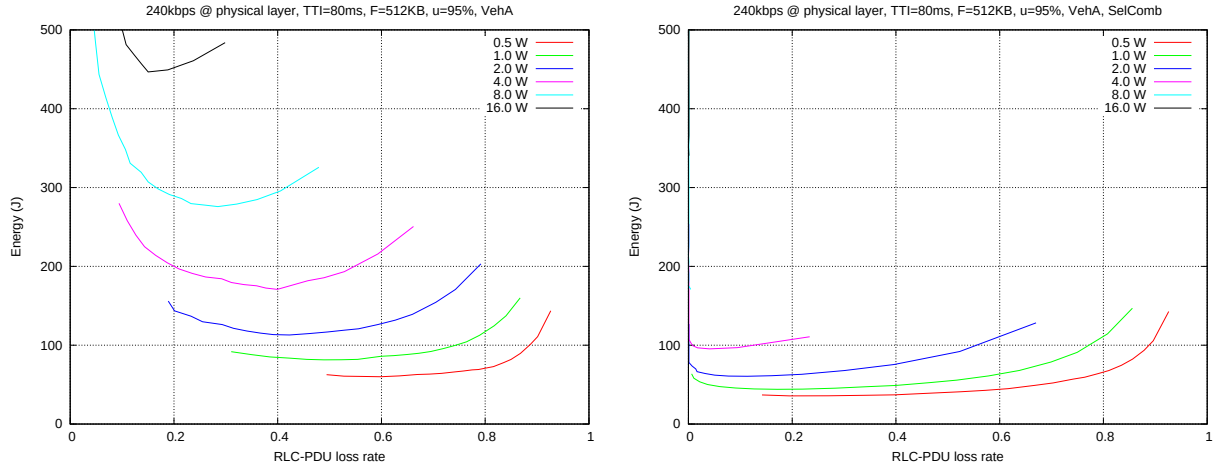


Fig. 17. Simulation Results for a 240 kbps bearer, vehicular A mobility model, without combining (left) and with selective combining (right)

B. *Time-Constrained Download Delivery*

In this section we consider the scenario where the broadcast of the file takes place over a limited amount of time. A similar setup as considered for the previous simulations is assumed. Figure 19 shows simulation results for time-constrained broadcast of a 512 KB file over a 240 kbps bearer, whereby the users follow a Vehicular A mobility model. Specifically, the figure shows the required transmission energy to deliver a 512 KB file as a function of the media bitrate for the same channel and mobility models for a conservative setting of the Turbo code rate $r_{\text{inner}} = 0.33$ and an optimized setting. To deliver a file with a certain bitrate, for the case of higher Turbo code rate, significantly

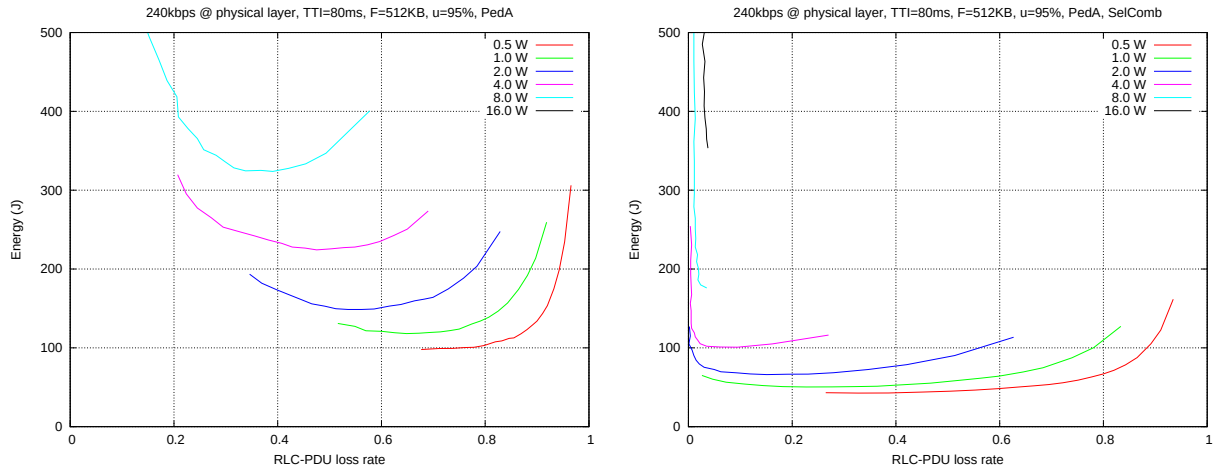


Fig. 18. Simulation Results for a 240 kbps bearer, pedestrian A mobility model, without combining (left) and with selective combining (right)

less energy is necessary. Note also that the conservative setting limits the bitrate of the file to 64 kbit/s, whereas the optimized setting can easily provide at least twice the bitrate.

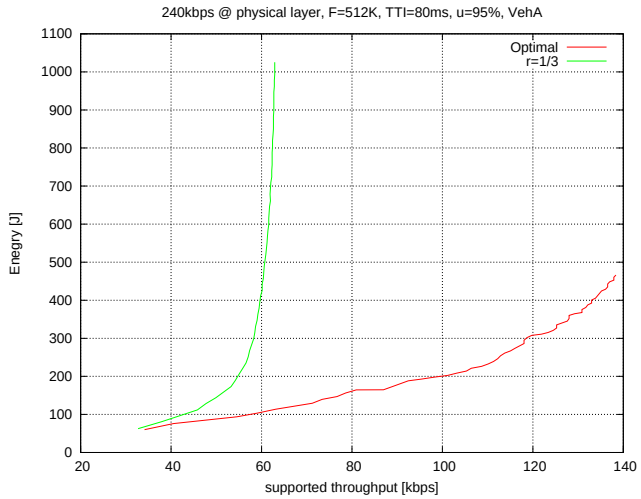


Fig. 19. Required transmission energy to deliver a 512 KB file versus the supported bit-rate over a bearer type of 240 kbps, whereby 95% of users are satisfied and follow a Vehicular A mobility model.

C. *Streaming Delivery and Mobile TV Services*

For streaming delivery, similar simulations as for the file delivery case have been performed. Tradeoffs in resource allocation have been evaluated to obtain suitable system configurations. Still, the variability of the system only allows to study selected use cases and only selected but also representative performance results are reported. In the following we briefly describe the parameters applied for the following results. For the results the bearer parameters for bearer 2 in Table I was used. The applied video sequence is the sequence *party* from the [28] in QCIF resolution and 12 fps. The 30 seconds sequence was looped 15 times such that basically the transmission

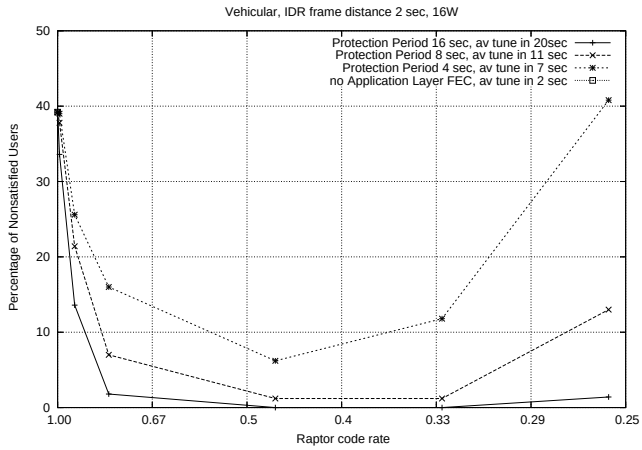


Fig. 20. Percentage of satisfied users versus Raptor code rate for constant system resources, IDR frame distance 2 seconds, and different protection periods compared to no AL-FEC. Also reported are the average tune-in delays.

of a 9 minutes video stream was simulated. The sequence was encoded with IDR frame distances $T_{\text{IDR}} = \{2\}$ seconds [2] and to achieve a target quality of average PSNR of at least 32dB. The resulting bitrate is approximately 100 kbit/s. The applied protection periods for the Raptor code were $T_{\text{PP}} = \{4, 8, 16\}$ seconds. The Raptor code rate was selected to optimally fill the IP bearer for a chosen Turbo code rate of $r_{\text{inner}} = \{0.24, 0.245, 0.26, 0.3, 0.5, 0.7, 0.9\}$ which results in Raptor code rates of of $r_{\text{outer}} = \{1.0, 0.99, 0.91, 0.79, 0.47, 0.33, 0.26\}$. Transmit powers of $P_{\text{Tx}} = \{2, 4, 8, 16\}$W were applied, but in contrast to the download delivery case, it turned out that only full power of 16W provides satisfactory results. In total, each experiment was carried out for $N = 500$ users which all are assumed to move at speed 30km/h in the serving area using the a vehicular model or with speed 3km/h using the pedestrian channel model. For the video quality evaluation, a pDVD of $\overline{D}_{\text{dec,max}} = 5\%$ was considered as satisfying quality. In any case we do not use any combining technology in the physical layer.

In a first experiment, the benefits of Raptor codes to the system is investigated along with the influence of the protection period. Figure 20 shows the percentage of satisfied users versus the Raptor code rate for constant system resources, IDR frame distance 2 seconds, and different protection periods compared to no AL-FEC. The results are for vehicular users. Along with the different configurations for the protection periods, also the average tune-in delays are reported.

Without AL-FEC and using only PHY-FEC, the performance of the system is pretty low, only 60% of the users can be supported even despite a quite low Turbo code rate is applied. With the use of Raptor codes, significantly more users can be supported. For a fixed protection period of for example 4 seconds, and using the right combination of Turbo coding and Raptor coding, the number of number of non-satisfied users decreases tremendously. A reasonably good operation point is when the Turbo code and the Raptor code use about the same code rate of 0.5. If the Turbo code rate is set higher then the Raptor code rate must be set lower and the performance decreases again. It is also clear from the results, that with longer protection periods, more and more users can be supported. With 16 seconds protection period and code rate 0.5 for each code, almost all users observe satisfying quality. However, the introduction of the Raptor code as any application layer error recovery mechanism increases the tune-in delay as can see from the values. This tradeoff

[2]larger values $T_{\text{IDR}} = \{4, 8, 16\}$ seconds have been checked, but the bitrates gains were only in the range of 5%, such that sacrificed tune-in delay is not justified and the 2 seconds value was used.

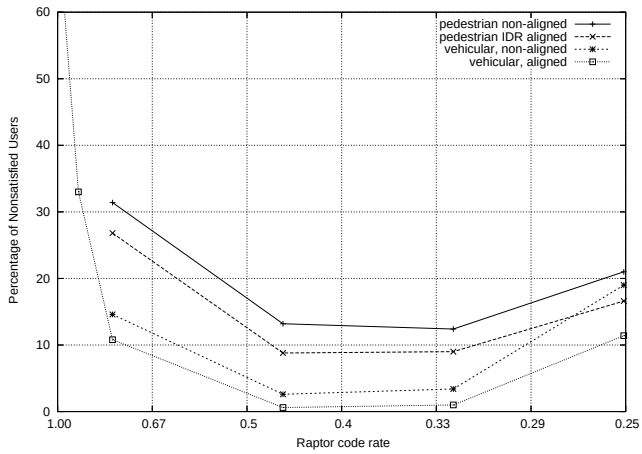


Fig. 21. Percentage of satisfied users versus Raptor code rate for constant system resources, IDR frame distance 2 seconds, 8 seconds protection period, min-buffer time 14 seconds resulting in 11 seconds tune-in delay for different IDR frame alignment and different mobility models.

needs to be taken into account in the system design.

In a second set of experiments, also pedestrian users have been included. In addition, a second mode has been introduced, which takes care that the start of an AL-FEC source block is always aligned with an IDR frame. The results for these additional experiments are shown in Figure 21. It is observed that the alignment is beneficial in performance as the size of the source block size is less variable. However, the tune-in delay reductions are not that significant as the chosen IDR frame frequency of 2 seconds does not provide significant misalignment. Furthermore, it can be seen that the support of pedestrian users is more difficult, as the channel variations are slower and therefore less time diversity in the same time frame can be exploited. Still, the same beneficial tendencies of using Raptor codes for faster moving users still applies to slower moving users.

VI. Discussions and Optimizations

The usage of long time-diversity and AL-FEC is very beneficial and basically essential as seen from the MBMS performance results. However, the time diversity can only be fully exploited if longer protection periods are applied. If conventional sending arrangements and stringent playout strategies are applied as done for the above simulations, then the protection period also influences the channel switching times. This is of special relevance for the case of linear broadcast video delivery in mobile TV environments. Therefore, work and improvements on improved zapping times is necessary. Several methods have been proposed and discussed for this purpose, for example combinations of unicast and multicast delivery, provision low resolution fast switching channels, or smart combinations of AL-FEC and media playout, see e.g. [29]. In conjunction with AL-FEC, several aspects of improving switching times and efficiency have been proposed, e.g. in [30]. We highlight one variant in the following. The basic idea is shown in Figure 22: A continuous data stream (yellow) is partitioned into source blocks of certain size such that an AL-FEC encoding strategy can be applied. The source symbols and the generated repair symbols from a single source block are distributed over multiple transmission slots as for example typical in DVB-H because of time-slicing. Two different sending arrangements are discussed: Sending arrangement 1 distributes the source symbols and the repair symbols sequentially over the transmission slots. This scheme is applied for the results in the previous section. Sending arrangement 2 distributes the source

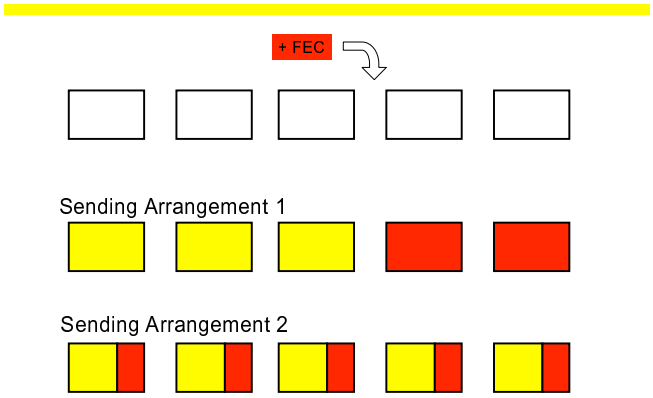


Fig. 22. Zapping-optimized sending arrangement.

symbols and the repair symbols in such a way that each burst contains a mixture of source and repair symbols. Both arrangements have advantages and drawbacks. For sending arrangement 1, in good channel conditions one might be able to ignore bursts containing only repair symbols, thus leading to power savings, see [30]. However, sending arrangement 1 can also result in increased tune in delays. For example, if the user happens to tune in to a burst of repair symbols, if there are not enough repair symbols to decode then since the corresponding source symbols were sent in earlier bursts these repair symbols are discarded and the display of the video can only commence after reception of subsequent bursts for subsequent source blocks.

Sending arrangement 2 sends source symbols interleaved with repair symbols, such that fast tuning is supported, because immediate access to source symbols is possible. For example, as soon as a burst is received without loss containing source symbols and the source symbols correspond to a random access point to the media stream, the data can be immediately decoded and displayed. By these means, the channel switching times can be reduced. However, fast switching relies on no loss in the initial received bursts. This can cause problems as once being tuned to a service and staying with the program, the AL-FEC is quite likely required at some later point of time when there is packet loss. In a simple receiver implementation, the video decoder would then just apply rebuffering, once the AL-FEC is required. However, the video and audio decoders can easily and without perceptual degradation slow down the media playout. This concept is known as adaptive media playout (AMP), see for example [31]. Therefore, it is reasonable that after switching, the media decoder slows down the playout, by for example 25%, such that buffer for AL-FEC decoding can be built up for some time. With a slow down of 25% and for a AL-FEC delay of 10 seconds, the AL-FEC can be fully exploited within 40 seconds. If the AL-FEC needs to be used faster, more aggressive strategies might be used which might lead to some small initial degradation, but losses can be compensated.

These sending strategies may also be applied for MBMS as the sending order for MBMS is not prescribed.

VII. Conclusions

In this paper we have introduced and investigated MBMS download and streaming delivery services in UMTS systems considering a comprehensive analysis by applying a detailed and complex channel model and simulation setup. A significant part of MBMS is AL-FEC based on Raptor codes, which have been standardized for MBMS for the broadcast delivery of multimedia content

and integrated in CDPs. A thorough review of the Raptor codes and some implementation guidelines are provided. Their benefits are manifold, but the use of Raptor codes for applications in mobile broadcast environments is a perfect match, mainly due to their excellent performance being close to ideal fountain codes, their low computational complexity and their flexibility. Despite the detailed analysis of Raptor codes in the MBMS standardization efforts, no full system level evaluation of AL-FEC, and especially Raptor codes has been previously done from a comprehensive and realistic system-wide perspective. Therefore, we have provided an accurate and comprehensive simulation model which takes into account the effects of different layers in the protocol stack and also evaluates the services for the two most important metrics, user experience and radio resource consumption.

Of specific interest in the evaluation is the tradeoff of code rates and resources being used in the physical layer compared to the case where the resources are spent on the application layer. The results clearly indicate that a tradeoff and thorough balancing of the overhead is necessary. In contrast to some beliefs and conjectures that all problems can be solved on the physical layer, our results clearly show that only a well designed system that considers combinations of settings of the parameters at the different protocol layers can optimize system resources and user perception. In particular it was shown that for file delivery a well-designed system should use less physical layer Turbo code protection and much more application layer Raptor code protection than considered in the MBMS standardization process. Raptor codes can spread protection over long intervals of time whereas Turbo codes only provide protection over very short intervals of time. Because channel conditions have less variance when measured over longer periods of time than shorter periods of time, the Raptor codes are more efficient at recovering losses averaged over long intervals of time than the Turbo codes are at preventing losses over short intervals of time. Thus, it turns out to be beneficial to use less Turbo code protection and accept the consequent higher RLC-PDU loss rates that can be more efficiently protected using Raptor codes. This shows that packet loss is not per se a bad thing and, counter-intuitively, high rates of packet loss can be a fundamental property of a well-designed system. The principle findings have been verified for different system parameter settings such as different power assignments, different bit rates, different mobility models, as well as advanced receiver techniques such as selective combining.

Similar results and findings have been provided for streaming delivery. However, in this case, the protection period must be lower to support the real-time delivery of the service with small channel change times. The system design in this case needs to consider not only the FEC on different layers, but also the video coding parameters. The tradeoffs of different settings have been shown, and the reported gains when using AL-FEC make the solution very attractive despite a possible increase in channel switching times. However, with smart sending arrangements and media playout schemes, these drawbacks can be to a large extent compensated.

Although details are bound to be different, we hypothesize that the system-level benefits of using AL-FEC (and in particular Raptor codes) and the system-wide trade-offs between AL-FEC and PHY-FEC shown for MBMS will also translate to other broadcast and multicast channels and services. As an example, the benefits of using Raptor codes for file delivery within the DVB-H IPDC standard have been demonstrated and the standardized Raptor codes have also been adopted by that standard.

Acknowledgments

The authors would like to thank Waqar Zia from Munich University of Technology for assisting in the streaming delivery simulation. Also the support of the staff of Nomor Research in the generation of this work, specifically Eiko Seidel for providing useful, constructive, and insightful comments on the manuscript.

REFERENCES

[1] T. Paila, M. Luby, R. Lehtonen, V. Roca, and R. Walsh, "FLUTE - File Delivery over Unidirectional Transport," IETF, RFC3926, Tech. Rep., Oct. 2004.

[2] M. Watson, "Forward Error Correction (FEC) Framework," Internet Engineering Task Force (IETF)," draft-ietf-fecframe-framework-00.txt, Feb. 2007.

[3] M. Watson, M. Luby, and L. Vicisano, "Forward Error Correction (FEC) Building Block," Internet Engineering Task Force (IETF)," RFC5052, Aug. 2007.

[4] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft, "The Use of Forward Error Correction (FEC) in Reliable Multicast," Internet Engineering Task Force (IETF)," RFC3453, Dec. 2002.

[5] M. Watson, "Basic Forward Error Correction (FEC) Schemes," Internet Engineering Task Force (IETF)," draft-ietf-rmt-bb-fec-basic-schemes-revised-03.txt, Feb. 2007.

[6] A. Shokrollahi, M. Watson, M. Luby, and T. Stockhammer, "Raptor Forward Error Correction Scheme for Object Delivery," Internet Engineering Task Force (IETF)," draft-ietf-rmt-bb-fec-raptor-object-09.txt, July 2007.

[7] A. Shokrollahi, "Raptor Codes," *IEEE Transactions on Information Theory*, no. 6, June 2006.

[8] 3GPP TS 23.246 V6.9.0, *Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Service; Architecture and functional description*, December 2005.

[9] 3GPP TS 26.346 V6.1.0, *Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Service; Protocols and Codecs*, June 2005.

[10] A. Shokrollahi, S. Lassen, and M. Luby, "Multi-stage code generator and decoder for communication systems," June 27, 2006, u.S. Patent No. 7,068,729.

[11] M. Luby, "LT codes," in *Proc. 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002.

[12] ETSI TS 102 472 v1.2.1, *IP Datacast over DVB-H: Content Delivery Protocols*, Mar. 2006, technical Specification, http://www.dvb-h.org.

[13] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," in *proceedings of ACM SIGCOMM '98*, 1998.

[14] M. Luby, "Information additive code generator and decoder for communication systems," October 23 2001, u.S. Patent No. 6,307,487.

[15] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, "Efficient erasure correcting codes," *IEEE Transactions on Information Theory*, vol. 47, pp. 569–584, 2001.

[16] A. Shokrollahi, S. Lassen, and R. Karp, "Systems and processes for decoding chain reaction codes through inactivation," 2005, u.S. Patent number 6,856,263.

[17] Shokrollahi, A., and Luby, M., "Systematic Encoding and Decoding of Chain Reaction Codes," U.S. Patent 6 909 383, June 21, 2005.

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

[19] M. Luby, T. Gasiba, T. Stockhammer, and M. Watson, "Reliable multimedia download delivery in cellular broadcast networks," *IEEE Transactions on Broadcasting*, vol. 53, no. 1, pp. 235–246, Mar. 2007.

[20] 3GPP TS 25.346 V7.0.0, *Technical Specification Group Radio Access Network; Introduction of the Multimedia Broadcast Multicast Service (MBMS) in the Radio Access Network (RAN)*, March 2006.

[21] 3GPP TS 25.212 V7.0.0, *Technical Specification Group Radio Access Network; Multiplexing and channel coding (FDD)*, March 2006.

[22] 3GPP TS 25.222 V7.0.0, *Technical Specification Group Radio Access Network; Multiplexing and channel coding (TDD)*, March 2006.

[23] 3GPP TS 25.993 V6.13.0, *Technical Specification Group Radio Access Network; Typical examples of Radio Access Bearers (RABs) and Radio Bearers (RBs) supported by Universal Terrestrial Radio Access (UTRA)*, March 2006.

[24] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, M. Handley, and J. Crowcroft, "Asynchronuous Layered Coding (ALC) Protocol Instantiation," IETF, RFC3451, Tech. Rep., Dec. 2002.

[25] T. Stockhammer, T. Gasiba, W. Samad, T. Schierl, H. Jenkac, T. Wiegand, and W. Xu, "Nested harmonic broadcasting for scalable video over mobile datacast channels," *Wiley Journal - Wireless Communications and Mobile Computing, Special Issue on Video Communications for 4G Wireless Systems*, vol. 7, no. 2, pp. 235–256, Feb. 2007.

[26] S. Wenger, T. Stockhammer, M. Hannuksela, M. Westerlund, and D. Singer, "RTP Payload Format for H.264 Video," Internet Engineering Task Force (IETF)," RFC3984, Feb. 2004.

[27] 3GPP TSG-RAN WG1 R1-030984, *Link Error Prediction for E-DCH*, PSM SWG, Seoul, South Korea, Oct. 2003.

[28] *TR26.902 Video Codec Performance*, 3GPP, June 2007. [Online]. Available: http://www.3gpp.org

[29] *TD00096, Fast channel changing in RTP, Internet Streaming Media Alliance*, ISMA, June 2007. [Online]. Available: http://www.isma.tv/technology/TD00096-fast-rtp.pdf

[30] D. Gomez-Barquero and A. Bria, "Application Layer FEC for Improved Mobile Reception of DVB-H Streaming Services," in *Proceedings IEEE VTC Fall*, Montreal, CA, Sept. 2006.

[31] M. Kalman, E. Steinbach, and B. Girod, "Adaptive media playout for low delay video streaming over error-prone channels," *IEEE Trans. on Circuits and Systems for Video Technology*, June 2004.