De Fez Lava, I.; Fraile Gil, F.; Belda Ortega, R.; Guerri Cebollada, JC. (2012). Analysis and evaluation of adaptive LDPC AL-FEC codes for content download services. IEEE Transactions on Multimedia. 60(3):641-650. doi:10.1109/TMM.2012.2190392.

# Analysis and Evaluation of Adaptive LDPC AL-FEC Codes for Content Download Services

Ismael de Fez, Francisco Fraile, Román Belda and Juan Carlos Guerri

***Abstract*— This paper proposes the use of Adaptive LDPC AL-FEC codes for content download services over erasure channels. In Adaptive LDPC codes, clients inform the content download server of the losses they are experiencing. Using this information, the server makes FEC parity symbols available to the client at an optimum code rate. This paper presents an analytical model of the proposed Adaptive LDPC codes. The model is validated through measurements realized with an application prototype. Additionally, results show the performance of these codes in different scenarios, compared to the performance of non-adaptive AL-FEC, Optimum LDPC AL-FEC codes and an almost ideal rateless code. Adaptive LDPC AL-FEC codes achieve download times similar to almost ideal rateless codes with less coding complexity, at the expense of an interaction channel between server and clients.**

***Index Terms*—Adaptive codes, AL-FEC, FLUTE, LDPC**
***EDICS*— 5-HIDE, 5-SEND, 5-WRLS**

## I. Introduction

FOR a while now, the demand for wireless broadband bandwidth has being increasing very rapidly. Users demand greater access capacity to utilize an increasing number of services and applications. In turn, these applications become more and more hungry for bandwidth, especially video services and related applications. As a result, wireless broadband demand has experienced a hundredfold growth in the last years and one can only expect a similar expansion in the years to come.

Unfortunately, wireless broadband exploits a limited resource: frequency spectrum. For this reason, new wireless technologies improve spectrum efficiency and regulators reshape radio spectrum allocation, adapting it to satisfy the changing needs of society.

However, this approach faces important challenges to cope with user demands. Modern wireless standards perform very close to theoretical limits. Without a major breakthrough, it is very unlikely that the efficiency of next generation telecommunication systems would be orders of magnitude greater than that of current technologies. Also, new wireless technology needs time to become established in markets and even more time to become economically profitable for operators. Policymakers face difficulties to reshape radio spectrum as any change requires extensive negotiations.

Since the frequency spectrum is a limited resource, it is necessary to search for underexploited features of current wireless technologies that could optimize the usage of spectrum efficiency. Modern wireless standards provide wireless multicast access. On the other hand, Consumer Electronic Devices integrate higher storage capacity available for content download. The combined usage of multicast transmissions and content caching can be useful to improve the performance of wireless communications, since content items requested by many different users can be sent with a single transmit operation and cached in storage memory of clients, prior to a download request [1].

In this framework, content transmissions need to be protected against errors. To avoid further investments in infrastructure, additional protection needs to be provided at the application layer, either by AL-FEC (Application Layer – Forward Error Correction), by retransmissions or by a combination of both. AL-FEC significantly improves the performance of multicast content download services, but its performance is somewhat dependent on the complexity of the algorithm used to protect the information. In this sense, the most advanced algorithms fall in the category of rateless codes and perform very close to ideal FEC codes: no matter what is the erasure rate of the channel, receivers need only to acquire an amount of data equivalent to the size of the original file to be able to restore it.

Nevertheless, rateless codes require more processing to generate the parity data for a specific file than other AL-FEC codes, such as LDPC. In environments where the multicast content selection is dynamic, it may be impossible to generate the parity data and insert it in the network in time [2].

On the other hand, LDPC AL-FEC codes provide a good trade-off between performance (download time) and complexity (time required to generate parity and time required to do the decoding process) [3]. LDPC AL-FEC parity can be generated nearly in real time, but unlike rateless codes, the optimum code rate depends on the erasure rate of the channel. When the code rate of LDPC AL-FEC is set to the optimum rate for an erasure channel, LDPC AL-FEC codes perform very closely to rateless codes.

In this paper, the code rate of an AL-FEC LDPC code is

Manuscript received October 1, 2011. This work was supported by the Spanish Ministry of Industry, Tourism and Trade, under project MIQUEL (TEC2007-68119-C02-01/TCM).

I. de Fez*, F. Fraile, R. Belda and J. C. Guerri are with the Institute of Telecommunications and Multimedia Applications (iTEAM) from Universitat Politècnica de València, Camí de Vera s/n, 46022 Valencia, Spain (phone: 34-963879588; fax: 34-963879583; e-mail: isdefez@iteam.upv.es, ffraile@iteam.upv.es, robelor@iteam.upv.es, jcguerri@dcom.upv.es).

adapted to the erasure rate of the channel as perceived by a particular user. During the transfer process, clients report on the erasure rate they perceived. If needed, the server generates additional FEC rate and inserts it in a multicast channel. Receivers are notified about the availability of additional FEC parity data and start processing it. The performance of these proposed Adaptive LDPC AL-FEC codes is compared to rateless codes, static LDPC AL-FEC codes and Optimum LDPC AL-FEC codes. The Optimum LDPC AL-FEC codes are an ideal implementation of Adaptive LDPC codes where the feedback received by the clients is instantaneous.

The rest of the paper is structured as follows. Next section provides an overview of related technologies. Section III presents the test case scenario that sets the basis for the measurements. Section IV analyses mathematically Adaptive LDPC codes, whereas section V describes the methodology used for the measurements. Section VI includes the theoretical and the experimental results and their corresponding analysis. Finally, the last section of the document includes some final conclusions about the study and the future work.

## II. Background

This section describes the main technologies related to IP multicast content download services. The proposal described in this paper is based on the FLUTE (File Delivery over Unidirectional Transport) protocol [4], which provides reliability on the transmission through the usage of AL-FEC, among other mechanisms. FLUTE implements an AL-FEC building block [5] that allows the integration of virtually any FEC code. In this sense, this paper is focused on the standardized LDPC codes.

### A. FLUTE

FLUTE is an IP multicast protocol widely used for multicast file download services to mobile devices. Multicast protocols improve the scalability of file transfer services, since files can be delivered to massive numbers of receivers through a single transmission operation. However, in the absence of retransmissions in the transport layer, it is necessary that multicast protocols provide reliability at the application layer to overcome possible packet loss in the communication channel.

FLUTE file transfers are organized into file delivery sessions. A session is uniquely identified by the multicast source IP address and by a session identifier called TSI (Transport Session Identifier). Each session contains one or more delivery channels. Each channel sends FLUTE packets with a different UDP destination port number and with a given transmission rate.

In the transmission, each file is fragmented in source blocks. Also, each block is composed of *n* encoding symbols: *k* source symbols and *n* - *k* parity symbols. Generally, each symbol represents the payload of a FLUTE packet, although one FLUTE packet can contain several encoding symbols. Obviously, when no AL-FEC is used *n=k*. For the sake of simplicity, in this paper *n* refers to the total number of symbols of a file for both no FEC and AL-FEC. Figure 1 shows the division of a file in blocks and symbols. Segmentation of files is provided by a blocking algorithm (which calculates blocks from files) and a symbol encoding algorithm (which calculates encoding symbols from blocks). These algorithms are defined in the FLUTE RFC [4].

In reception, clients will be able to rebuild the file when they receive a number of packets equal to *k*inefficiency_ratio* [6] . The value of the inefficiency ratio depends on the coding algorithm. In codes that belong to the Minimum Distance Separable (MDS) category this value is equal to 1 [6], whereas in the rest of codes the inefficiency ratio is greater than 1.

In this sense, FLUTE supports different AL-FEC codes: Compact No-Code (that is, no coding is applied – No-FEC) [7], Reed-Solomon [8], Raptor [9] [10], RaptorQ [11] and LDPC codes [12].

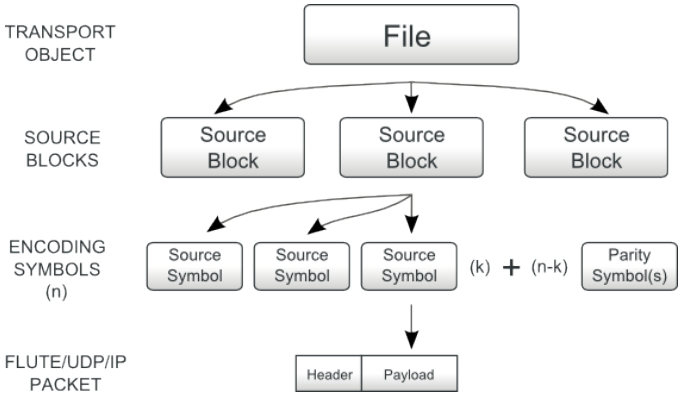


Fig. 1. FLUTE packet construction.

On the other side, there are two types of FLUTE delivery sessions: file transmission and file carousels. In carousels, files are sent cyclically on a seamlessly endless loop, which represents another reliability mechanism. This way, the clients can receive in the next cycles the packets lost in the current cycle. This paper uses carousels as delivery mechanism. Each carousel contains all the files to send.

### B. LDPC

Low Density Parity Check (LDPC) codes [12] [13] are lineal block codes based on a parity check matrix used in the encoding and decoding processes. In AL-FEC, this matrix defines the relations between the source and the parity symbols. The matrix, which is by definition disperse, is divided in two sub-matrixes: the left one (which establishes the relations between the source symbols) and the right sub-matrix (which refers to parity symbols).

By means of the matrix, the encoder generates the parity symbols through XOR operations on the source symbols and other parity symbols previously generated. Similarly, receivers use the matrix to reconstruct the symbols that have not been received by performing XOR operations on the encoding symbols already received.

Obviously, receivers must use the same parity matrix as the sender in order to successfully decode each source block. Sender and receivers obtain the parity check matrix via a predefined algorithm (depending on the type of LDPC

structure). The algorithm generates the matrix using some input parameters: number of source symbols (*k*), number of encoding symbols (*n*), number of equations to which a source symbol belongs to and seed used to generate the pseudorandom numbers. All these parameters are sent in the FLUTE header extension EXT_FTI.

The RFC 5170 [14] defines two LDPC structures: Staircase and Triangle. These are the only LDPC structures supported by FLUTE. Both structures only differ on the right sub-matrix generation: one has a shape like a staircase, and the other like a triangle. A complete comparison between both structures can be found in [15], which analyses, among other parameters, the inefficiency ratio and the encoding/decoding time.

## III. System Overview

The system proposed in this paper uses a hybrid unicast / multicast content delivery mechanism to provide content to users within the boundaries of the service area. It is assumed that the overall capacity of the wireless access is shared between unicast and multicast connections and that there is a limited bandwidth for multicast connections. It is also assumed that users experience a slowly varying channel. Moreover, it is assumed that files in the carousel are only a few popular files that may change with time. This system may be provided on top of any wireless network technology with multicast support, such as Wi-Fi.

During the delivery process, clients and server use a reporting mechanism through which the server obtains an estimation of the erasure rate perceived by every user. If required, the server generates AL-FEC parity and inserts it in the wireless media. All parity symbols belonging to a code rate are inserted on a separate FLUTE channel. Therefore, every client receives the base ALC (Asynchronous Layered Coding) layer, with encoding symbols belonging to the base FEC rate and, after some time, they also subscribe to a second ALC channel in which they receive additional AL-FEC parity at a rate adapted to the erasure rate that the user experiences.

Obviously, all multicast channels share the overall maximum bitrate allocated for multicast in the wireless access. Also, multicast and unicast traffic compete for network resources. For this reason, there is no multicast traffic until there are a sufficient number of requests for a content item. Similarly, there are no AL-FEC parity channels until users need it. In order to upper bound the maximum number of channels, all users that experienced similar losses are prompted to join the same multicast channel for additional parity data. By separating the parity packets in different channels according to their encoding rate, receivers only need to process AL-FEC packets at an optimal rate for their channel losses. This multicast scheme achieves lower resource consumption in clients, which is appropriate for mobile devices.

Figure 2 shows a general overview of the proposed scenario.

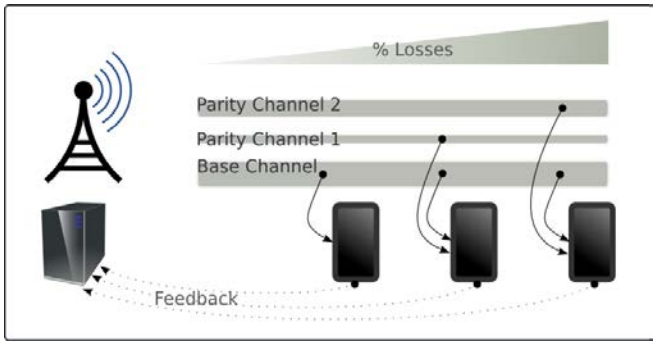


Fig. 2. System overview.

## IV. Theoretical Analysis

This section analyses mathematically Adaptive LDPC codes. The main goal is to calculate analytically the download time for multicast FLUTE file transfer services using Adaptive LDPC codes. The download time is defined as the time elapsed from a download request until the file is completely downloaded to storage memory. The last encoding symbol received establishes this download time. In this study, each FLUTE packet contains exactly one encoding symbol.

As mentioned above, the server sends files cyclically in a file carousel. In order to calculate the download time, it is necessary to know the minimal number of times a carousel is sent, that is, the minimal number of cycles or loops needed by a client to download a file. This number will in turn depend on the packet losses of the communication channel between server and client. [16] presents a mathematical model valid for channels with uniform channel losses. In the methodology applied in this paper, a Markov model models the channel packet losses, in order to account for the characteristic burstiness of wireless communication channels. The next subsection describes the model for carousel retransmissions, where the expected number of cycles is derived from the expected number of packets received per cycle. The following subsection describes how the Markov model is applied to the calculation of the expected number of packets received per cycle. Finally, the last subsection presents an algorithm to calculate mathematically the download time.

### A. Analysis of carousel retransmissions

The probability of receiving new packets is different depending on whether AL-FEC is applied or not. When No-FEC is used, the probability of receiving *x* new packets at any given loop can be modeled by a hypergeometric distribution:

$$P(x,m,n,l)=\frac{\binom{m}{x}\binom{n-m}{(n-l)-x}}{\binom{n}{n-l}} \quad (1)$$

where *n* is the number of symbols or packets (as there is no FEC, *n*=*k*) of the file, *l* is the number of lost packets per loop and *m* represents the number of missing packets at the beginning of the loop. The denominator expresses the probability of receiving *n* - *l* packets in a carousel cycle.

Similarly, the numerator expresses the probability of receiving exactly $x$ new packets of the $m$ missing packets out of the received $n$ - $l$ packets.

The range of all possible values for $x$ is [0, $m$]. Thus, the latter probability yields the following expression for the expectation value of the number of packets correctly received at loop $i$:

$$x(i) = \sum_{\xi=0}^{m} \xi \cdot \mathrm{P}(\xi, m, n, l). \tag{2}$$

The number of cycles needed to download a file can be estimated using the following equation:

$$n_{cycles} = \min\{k : \sum_{i=1}^{k} x(i) \geq n\}. \tag{3}$$

Similarly, if AL-FEC is used, the probability of receive $x$ new packets at any given loop is defined by the next equation:

$$\mathrm{P}(x, n, r, l) = \frac{\binom{n-r}{x}\binom{r}{(n-l)-x}}{\binom{n}{n-l}}. \tag{4}$$

where, in this case, $n$ is the number of encoding symbols (source symbols plus parity symbols), $r$ is the number of received symbols at the beginning of the loop and $l$ is the number of lost packets per loop. Note that this expression is equal to (1), making the substitution $m$=$n$ - $r$. In this case, the expectation value is defined by:

$$x(i) = \sum_{\xi=0}^{n-r} \xi \cdot \mathrm{P}(\xi, n, r, l). \tag{5}$$

Then, an estimation of the number of cycles is provided by the following expression:

$$n_{cycles} = \min\{k : \sum_{i=1}^{k} x(i) \geq k * inef\_ratio\}. \tag{6}$$

### *B. Markov model channel*

The Markov model [17], widely used in the literature, simulates well the burst losses, typical in wireless networks. Specifically, the two-state Markov model (also known as Gilbert model) establishes that the probability of losing a packet depends on whether the previous packet has been received or not, as Figure 3 shows. Thus, in a bursty wireless channel, it is more likely to lose a packet if the previous packet is lost ($1$-$q$>$p$).

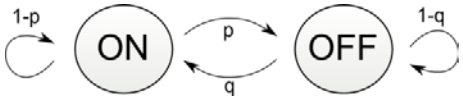


Fig. 3. State transition diagram for an example simplified Gilbert model.

The main parameters that characterize a lossy communication channel are the average loss probability ($P_{loss}$) and the average burst size ($b$). An appropriated configuration of the parameters $p$ and $q$ allows to model a channel with a given loss probability and burst size:

$$P_{loss} = \frac{p}{p+q}. \qquad b = \frac{1}{q}. \tag{7}$$

Figure 4 depicts a state transition diagram defined by applying this model to the FLUTE transmission with carousels and considering that in each cycle of the carousel the server sends $n$ packets. Each state in the diagram contains a pair ($x$, $y$) of numbers, where $x$ is the number of packets received in the current loop, and $y$ indicates if the last packet was received (0-ON) or not (1-OFF). Thus, there will be 2$n$+1 possible states in the transition diagram:

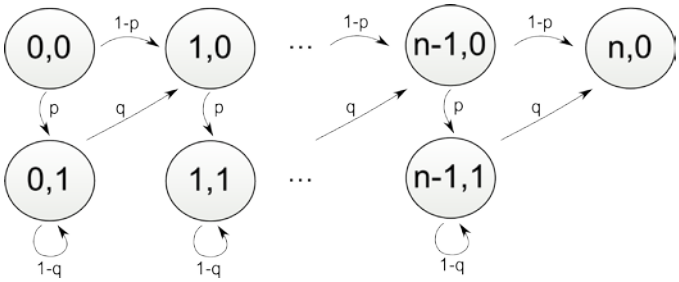


Fig. 4. State transition diagram for the Markov model in the transmission of $n$ packets.

The transition matrix associated, with dimensions [(2$n$+1) x (2$n$+1)], will be:

$$T = \begin{pmatrix} 0 & p & 1-p & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1-q & q & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & p & 1-p & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1-q & q & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & p & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 1-q & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1-p \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & q \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 \end{pmatrix} \tag{8}$$

The probabilities vector is a [2$n$+1] vector that indicates the probability of being in each state:

$$\Pi(i) = \left(P_{0,0}\ P_{0,1}\ P_{1,0}\ P_{1,1}\ P_{2,0}\ \cdots\ P_{n,0}\right). \tag{9}$$

Therefore, the probabilities vector in the iteration $i$ (that is, after $i$ packets have been sent) will be:

$$\Pi(i) = \Pi(0) T^{i}. \tag{10}$$

where $T$ is the transition matrix and $\Pi(0)$ is the initial probabilities vector. Considering that initially the system is in the state ON, since at the beginning no packets have been received, $\Pi(0)$ will be:

$$\Pi(0) = (1\ 0\ 0\ 0\ \cdots\ 0). \tag{11}$$

This way, the average number of packets received after $i$ iterations is calculated adding up the number of packets of

each state multiplied by the probability of being in each state:

$$\overline{N}(i) = \sum_{\eta=0}^{i<n} \eta \left[ P_{\eta,0}(i) + P_{\eta,1}(i) \right] + n P_{n,0}(i). \quad (12)$$

The total number of packets received in a loop is provided by applying ($i$=$n$) in (12). Hence, the estimated number of lost packets per cycle will be equal to:

$$l = n - \overline{N}(n). \quad (13)$$

With this value, the number of cycles needed to download a file is calculated using formulas (1)-(6).

### C. Analysis of Adaptive LDPC

As explained above, when using Adaptive LDPC, clients will receive the file with no FEC parity until they are able to join their corresponding parity channel. Thus, in order to model Adaptive LDPC there is a need to combine the two methods described above, corresponding to the cases where AL-FEC is used and where it is not. This section presents an algorithm that performs such combination to calculate the average download time using Adaptive LDPC codes.

Basically, the algorithm hereby proposed calculates the number of file packets downloaded applying the method presented above for No-FEC for every cycle up to feedback time. If the file download is not finished at feedback time, then the algorithm applies the method for AL-FEC.

The algorithm uses the formulas presented in the previous sections, using the following input parameters: the values $p$ and $q$ from the Markov loss model, the number of packets that make up a file without FEC ($k$), the transmission rate ($b$), the packet size ($S$) and the feedback time ($t\_fd$).

Moreover, in order to implement Adaptive LDPC, the algorithm has also as input parameters the code rate used to do the coding process and the inefficiency ratio derived from this code rate. These are two key parameters in the performance of the algorithm. Optimum values of code rate will provide the minimum values for the download time. In this sense, the value of the inefficiency ratio is strongly dependent on the code rate. As mentioned, the inefficiency ratio depends on the type of coding, and low values of inefficiency ratio will reduce the download time.

Note that it is necessary to take into account that the download can finish during a cycle, that is, before all packets of a file have been sent. For this reason, dichotomy algorithms 1 and 2 adjust the download time, obtaining the percentage of the last cycle in which the download has finished. This adjustment can have a great impact on the download time if the cycle time is very high. Specifically, Algorithm 1 adjusts the download time when No-FEC is applied (used in part 1 of the main algorithm) and Algorithm 2 adjusts it when Adaptive LDPC is applied (used in part 2 of the main algorithm). Algorithms 1 and 2 have as input parameters those that appear in equations (1) and (4), respectively. Algorithm 1 has also as input parameter the value of $m$ from the previous cycle to the cycle where the download has finished, whereas Algorithm 2 has the $r$ of the previous cycle. Moreover, Algorithm 2 has the inefficiency ratio as input parameter.

It is worth mentioning that the expectation value of the number of new packets calculated using equations (2) and (5) provides a decimal value. As the number of received packets is an integer number, in order to round in a realistic way, the Monte Carlo method has been used in all three algorithms.

**Algorithm Adaptive LDPC**

```
INPUT: p, q, k, S, b, t_fd, coderate, inef_ratio
OUTPUT: download_time
1:    Initialize (num_cycles1=0, num_cycles2=0)
2:    Calculate in which cycle the feedback message (c_fd) arrives
Part 1: No-FEC
3:    while (not all packets have been received and num_cycles1+1<c_fd)
4:        Calculate number of losses per cycle according to Markov model
          (p,q) using (13)
5:        Calculate new packets received (P) in the current loop using (2)
           and update the total number of packets received
6:        num_cycles1=num_cycles1+1
7:    end
8:    if (all packets have been received)
9:        Obtain the percentage of the last cycle using Algorithm_1
10:       download_time=(num_cycles1-1+percentage1)*k*S/b
Part 2: Adaptive LDPC
11:   else
12:       while (not all packets have been received)
13:           Calculate number of losses per cycle using (13)
14:           Calculate new packets received (P) in the current loop with (5)
              and update the total number of packets received
15:           num_cycles2=num_cycles2+1
16:       end
17:       Adjust download time obtaining percentage2 using Algorithm_2
18:       download_time=(num_cycles2-1+percentage2)*k*S/b/coderate+
          + t_fd
19:   end
```

**Algorithm 1**

```
INPUT: k, m, l, last_m
OUTPUT: percentage1
1:    Initialize (bottom=0, top=1)
2:    while (true)
3:        percentage1=(bottom+top)/2
4:        Calculate new packets received (P) with (2) with input parameters:
          percentage1*(k,m,l) and update total packets not yet received
5:        if (last_m-P<0)
6:            top=percentage1
7:        else if (last_m-P>0)
8:            bottom=percentage1
9:        else
10:           BREAK
11:       end
12:   end
```

**Algorithm 2**

```
INPUT: k, r, l, last_r, inef_ratio
OUTPUT: percentage2
1:    Initialize (bottom=0, top=1)
2:    while (true)
3:        percentage2=(bottom+top)/2
4:        Calculate new packets received (P) with (5) with input parameters:
          percentage2*(k,r,l) and update total packets received
5:        if (last_r+P<k*inef_ratio)
6:            bottom=percentage2
7:        else if (last_r+P>k*inef_ratio)
8:            top=percentage2
9:        else
```

```
10:             BREAK
11:         end
12:     end
```

Note that, when several blocks are used it is necessary to modify the proposed algorithm. In the modified algorithm, it is needed to take into account that the feedback message will arrive when a specific block is being received. Hence, some blocks can be received without FEC, another blocks can be received without FEC and then with Optimum LDPC, and another blocks can be received only with Optimum LDPC. So, the last block downloaded will determine the download time.

## V. Evaluation Methodology

This section describes the methodology used to evaluate the performance of the proposed Adaptive AL-FEC codes. The goals of the evaluation are to validate the analytical model presented above and to compare the performance of Adaptive LDPC codes for content download services with other proposals. The metric selected for the evaluation is the download time. Thus, it is necessary to identify the system parameters that could affect the download time and define values for them that are relevant for the case under study, as subsection V.A describes. Furthermore, in order to compare analytical and experimental results, it is necessary to setup a valid scenario for conducting trials with the Adaptive LDPC implementation, which is explained in subsection V.B.

Since Optimum LDPC is an ideal implementation of Adaptive LDPC, it is necessary to obtain the AL-FEC rate that minimizes the download time in the evaluation scenario for every packet loss rate. These values are later used to compare the developed implementation with the lower bounds established by Optimum LDPC.

Once all environment conditions are set and Optimum LDPC is modeled, the evaluation will consist of comparisons of the download time achieved under different configurations of the system parameters.

### A. Evaluation parameters

As mentioned, the parameter used in the evaluation is the average download time. The study consists of measurements of this time obtained by applying different AL-FEC codes to FLUTE file delivery sessions: No-FEC, Optimum LDPC (Staircase and Triangle), Adaptive LDPC and rateless codes.

The comparison between these codes is done analyzing their behavior in different environments. Specifically, this paper evaluates these codes for different file sizes, different number of blocks, transmission rates and feedback times.

In this sense, the measurements consider two different file sizes: 3000 and 6000 packets file size (i.e., over 4 and 8 Mbytes, as each packet contains 1428 bytes). These are typical sizes of multimedia contents played in mobile devices, such as music files or short videos [18].

Moreover, two different number of blocks have been used: 1 block and 10 blocks. In efficiency terms, it is more efficient to send the files using one block. However, the block represents the decoding unit and hence clients require less memory when they work with small blocks. Therefore, using several encoding blocks can be very recommended when clients have limited resources.

Furthermore, two different transmission rates have been used: 5 Mbps and 10 Mbps. Besides, feedback times of 1, 3 and 5 seconds have been used, since these are reasonable response values according to the transmission rates used.

The results of the study are presented in Section VI, which contains two types of results: analytical and experimental. In the first, the evaluation is done calculating the download time through the algorithm presented in the previous section. On the other hand, the next subsection explains the performance of the experimental results.

### B. Experimental scenario

The experimental results have been carried out in order to validate the analytical ones. Thus, a more exhaustive analysis of the different parameters can be made using the analytical model, since its performance is much faster and easier.

The performance of these experimental results has been carried out using an implementation of a FLUTE server and client developed by the authors, which implements all the aforementioned AL-FEC codes.

Note that the implementation of Adaptive AL-FEC codes is not specifically regarded in the FLUTE standards. In order to implement the Adaptive LDPC, this paper proposes that the FEC information (i.e., the FLUTE header extension EXT_FTI) is included in all the parity symbols, so that clients detect the code rate as soon as they join the parity channel. The FLUTE RFC [4] does not establish the frequency or the type of packets that carry the FEC information, as the only requirement is that there is one packet with the EXT_FTI extension per file in a cycle.

Furthermore, the insertion of a new parity channel does not affect ongoing downloads. When the server decides to include a new parity channel, it generates the encoding symbols for the specific file without interrupting the base channel. Therefore, parity channels will be only available after the server is able to process the file to generate the parity packets. Clearly, the complexity of the AL-FEC algorithm and the size of the block will lengthen the time needed for a server to include the parity data in the scenario.

On the other hand, when a client joins a given parity channel, they keep the source symbols successfully decoded from the encoding symbols received in the base channel. However, if there is a change in the AL-FEC code rate, client needs to discard previously received parity symbols, as these are no longer valid for the new code rate.

It is assumed that there is a feedback between server and client that provides the server with an estimation of the losses experienced by every client. A possible implementation of this feedback is described in [19].

Rateless codes are simulated according to their definition as near ideal FEC codes, which establishes that it is only necessary to receive a small additional percentage of the packets that make up a file to rebuild it, regardless the erasure rate of the channel [20].

It is also worth noting that the transmission scheme of the

packets that compose a block also affects the performance of LDPC codes. For this reason, the measurements apply a random transmission scheme (source and parity symbols are sent in a random order), which provides better results than a sequential scheme in the presence of burst losses [21].

In both analytical and experimental results, the measurements collect as many iterations as needed to provide a 99% confidence intervals. The measurements have been made in a controlled environment, simulating the losses in the channel with the two state Markov model [17]. In order to simulate a typical wireless channel, different channel losses between 0% and 30% (in steps of 5%) have been simulated. Also a 50% losses channel has been simulated to see the general tendency in the different studies. Fixing a percentage of losses and an average burst size, parameters *p* and *q* from Markov model are obtained using (7).

In the encoding process, values of code rates between 0.2 (very strong protection) and 0.9 (weak protection) have been used, with a precision of 0.1. Preliminary results show that this precision is enough to establish optimum AL-FEC values for the case under study [22]. Bear in mind that code rate represents the relation between the source symbols of a file and the total encoding symbols, that is, *k/n*. So, the less code rate, the more protection.

## VI. Results and Analysis

This section presents the results of the average download time against different parameters such as the channel packet loss rate, the transmission rate, the file size or the number of blocks used to send a file.

Two main studies have been developed: the first evaluates the optimum coding and code rates in channels with different loss rates. Once these values have been obtained, Adaptive LDPC is analyzed and compared with rateless codes and Optimum LDPC codes.

The optimum codes and code rates are measured experimentally, according to the methodology described in the previous section. On the other hand, the evaluation of Adaptive LDPC is done through analytical and experimental measurements.

### *A. Optimum codes and code rates*

This study analyzes what is the optimum coding and the optimum code rate depending on the losses of the channel and different transmission parameters: the transmission rate, the content size and the number of blocks.

Just to give an example of the results obtained, Figure 5 shows the download time of a 3000 packet file size, using 1 encoding block with a transmission rate of 5 Mbps, applying LDPC Staircase codes. The results show that for every channel packet loss rate there is an optimum AL-FEC code rate that minimizes the download time. For instance, in channels with 25% of losses, the optimum code rate for LDPC Staircase is 0.7. These results are compared with the ones obtained with LDPC Triangle and Compact No-Code (no AL-FEC used). The best codes (Compact No-Code, LDPC Staircase or Triangle) and the best code rate for each percentage of losses are chosen as optimum.

In this sense, Table I shows the optimum codes and code rates obtained for each scenario. In the table, the AL-FEC codes are identified according to the numeric identifier assigned by the IANA: 0) Compact No-Code, 3) LDPC Staircase and 4) LDPC Triangle. The parameters of the four scenarios are:

- Case 1: 3000 packets file size, 1 block, 5 Mbps
- Case 2: 3000 packets file size, 1 block, 10 Mbps
- Case 3: 6000 packets file size, 1 block, 5 Mbps
- Case 4: 3000 packets file size, 10 blocks, 5 Mbps

TABLE I
Optimum coding parameters. IANA AL-FEC codes identifiers: (0) Compact No-Code, (3) LDPC Staircase, (4) LDPC Triangle

| Losses | Case 1 | Case 2 | Case 3 | Case 4 |
|---|---|---|---|---|
| **0%** | - (0) | - (0) | - (0) | - (0) |
| **5%** | 0.9 (3) | 0.8 (3) | 0.9 (4) | 0.8 (3) |
| **10%** | 0.8 (3) | 0.8 (3) | 0.8 (3) | 0.7 (3) |
| **15%** | 0.8 (3) | 0.7 (3) | 0.8 (3) | 0.6 (3) |
| **20%** | 0.7 (3) | 0.7 (3) | 0.7 (3) | 0.6 (3) |
| **25%** | 0.7 (4) | 0.6 (3) | 0.7 (3) | 0.5 (3) |
| **30%** | 0.6 (3) | 0.6 (3) | 0.6 (3) | 0.5 (3) |
| **50%** | 0.4 (3) | 0.4 (3) | 0.5 (4) | 0.3 (3) |

Clearly, if there are no losses, the addition of AL-FEC parity penalizes the download time. For this reason, the best code for lossless channels is Compact No-Code in all scenarios. In the event of channel losses, LDPC Staircase generally provides better download times than LDPC Triangle. The optimum code rates range between 0.6 and 0.9 in most of the cases.

Note that the optimum coding parameters are not only dependent on the channel losses, but also on other parameters like the file size, the number of blocks or the transmission rate. For instance, in the transmission of a 3000 packet file size, using 1 block, with a transmission rate of 5 Mbps in a channel with 25% of losses, the optimum coding parameters are: LDPC Triangle with a code rate of 0.7. However, if the file is divided into 10 source blocks, the optimum coding parameters are LDPC Staircase with a code rate of 0.5.

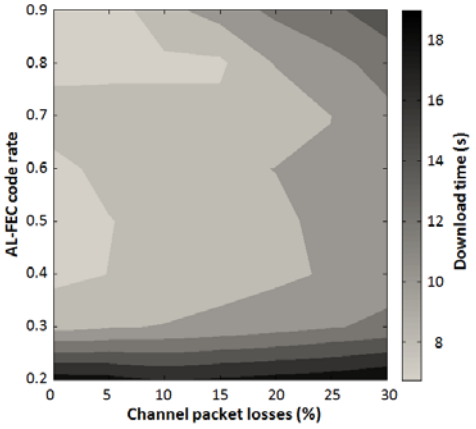


Fig. 5. Download time evaluation with LDPC Staircase codes with 3000 packet file size, 1 block and b=5 Mbps.

Fig. 5 shows how, for every loss rate, the average download time increases as the code rate moves away from its optimal

value. Nevertheless, moderate deviations of the actual channel packet loss rate from an estimated value may not increase the download time drastically. For instance, in the first scenario, using LDPC Staircase codes, the optimum code rate for a 15% packet loss is 0.8. The same code rate provides the best results for a 10% of packet loss and the results for 20% of losses are only slightly worse than for the optimum code rate of 0.7.

### *B. Evaluation of Adaptive LDPC*

In Adaptive LDPC codes, the server changes the coding parameters (coding and code rate) upon reception of a message that informs about the losses of the channel after some feedback time. Once this message arrives, the server uses the results obtained in the previous study to choose the optimum coding parameters depending on the transmission parameters (losses, file size, transmission rate and number of blocks) and continues sending the file with the new parameters. In order to minimize the download time, the server continues sending from the last block that was being transmitted before the coding change occurred.

In the different studies it is assumed that, initially, the server sends the file using Compact No-Code (No-FEC) codes, so no protection is used.

As Figure 6 shows, Adaptive LDPC (A-LDPC) codes offer very good results compared to No-FEC. As the losses are higher, the need of using AL-FEC mechanisms is more obvious.

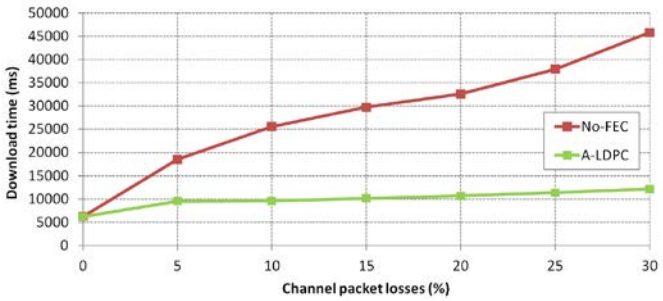


Fig. 6. Comparison between Adaptive LDPC and Compact No-Code with 3000 packet file size, 1 block, b=5 Mbps and t_fd=3 s.

As mentioned, experimental results have been carried out to validate the analytical results. In order to see the differences between the analytical and the experimental results regarding Adaptive LDPC codes, Figure 7 shows a comparison between two files of different size: 3000 packets and 6000 packets.

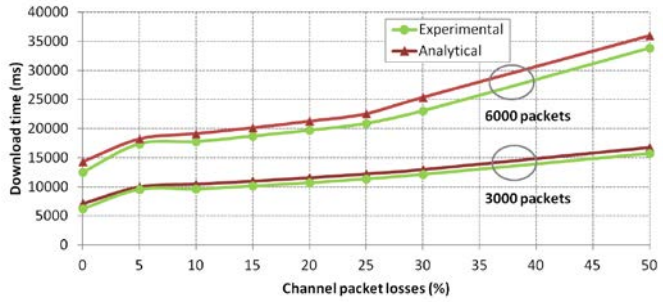


Fig. 7. Comparison between analytical and experimental results in a file size evaluation with 1 block, b=5 Mbps and t_fd=3 s.

As figure shows the values of analytical and experimental performance are very similar, although analytical results are a slightly higher than experimental ones. This is due to the fact that analytical results use a fixed inefficiency ratio of 1.07 which it is not exactly the inefficiency ratio of each percentage of losses. This value depends on each coding and different transmissions of the same file can provide different values so, in some codes it is not possible to obtain the inefficiency ratio analytically. We have chosen the value of 1.07 according to [15] and [21].

Mention that all the studies hereby presented have been carried out analytically and experimentally. Since both models provide very similar download times, only the experimental results are shown. Nevertheless, in the different graphs, the experimental results include an upper error bar that represents the difference with respect to the download time obtained in the analytical results.

Returning to the file size analysis, Figure 8 shows a comparison between the proposed Adaptive LDPC (A-LDPC) codes, Optimum LDPC (O-LDPC) and rateless codes. As expected, in all codes the download time of 6000 packet file size is approximately twice the download time of 3000 packet file size. For instance, the download time using Adaptive LDPC codes with 20% of losses is 10693 milliseconds with 3000 packet file size and 19710 milliseconds with 6000 packet file size.

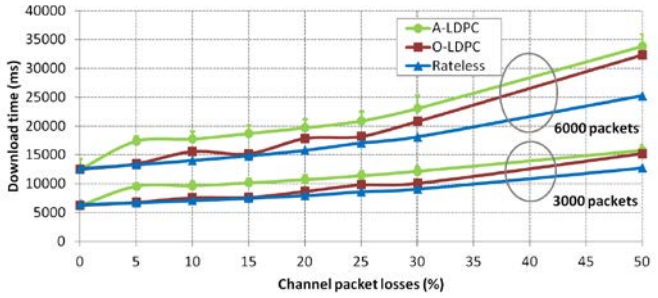


Fig. 8. File size evaluation with 1 block, b=5 Mbps and t_fd=3 s.

The difference between Adaptive LDPC and Optimum LDPC is lower as the file size is larger. On the contrary, the difference between Optimum LDPC and rateless codes is higher. Nevertheless the download time ratio (the download time of Optimum LDPC divided by the download time of rateless codes) gets better, so the larger file size, the better download time ratio.

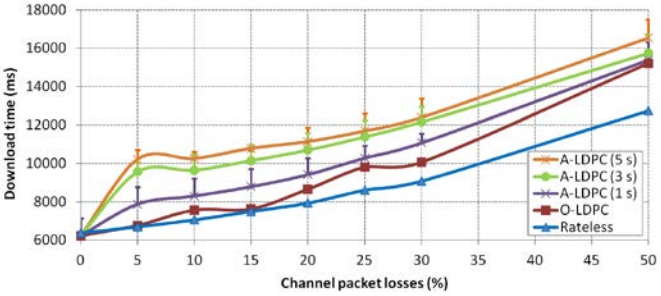


Fig. 9. Feedback time evaluation with 3000 packet file size, 1 block and b=5 Mbps.

On the other hand, Figure 9 shows the behavior of Adaptive LDPC compared with Optimum LDPC and rateless codes for different feedback times. Regarding Optimum LDPC codes, Adaptive LDPC offers a good behavior, slightly worse than Optimum LDPC if the feedback time is sufficiently short. As shown in the figure, the feedback time has a significant impact on the download time. In channel with high losses, the difference between the three feedback times (1, 3 and 5 seconds) decreases.

The graph also shows that Optimum LDPC codes perform very close to rateless codes (especially with moderate channel losses). In this sense, Figure 10 shows the download time ratio with respect to rateless codes. The graph shows that the download time for Optimum LDPC is only between 5% and 15% higher than rateless codes. This ratio is only slightly worse for Adaptive LDPC, especially when the feedback time is short (around 20% for 1 second). When the losses are higher the download time ratio is similar for the different feedback times and Optimum LDPC.

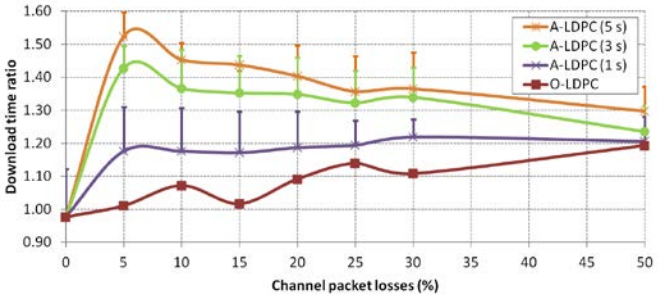


Fig. 10. Download time ratio with rateless codes with 3000 packet file size, 1 block and b=5 Mbps.

On the other hand, the transmission rate is, obviously, another parameter that affects the download time: the higher transmission rate, the lower download time. Figure 11 shows that, when the transmission rate is doubled, the download time is approximately divided by two. For instance, using Adaptive LDPC in channels with 30% of losses with a transmission rate of 5 Mbps, the download time is equal to 12155 milliseconds, whereas the download time is 6815 milliseconds when the transmission rate is 10 Mbps. With respect to Adaptive LDPC, it is worth noting that for a fixed feedback time, the difference between Adaptive LDPC and Optimum LDPC codes is lower with low losses.

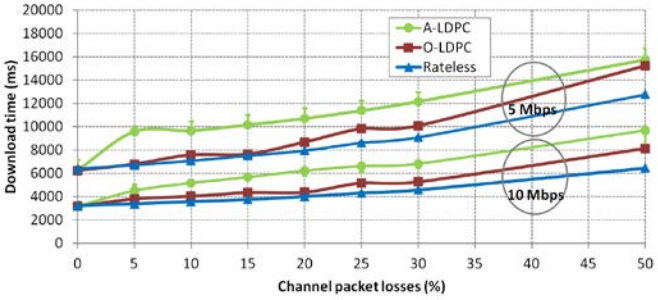


Fig. 11. Transmission rate evaluation with 3000 packet file size, 1 block and t_fd=3 s.

Finally, regarding the number of blocks, Figure 12 shows how dividing a file into source blocks affects the download time. Both LDPC and rateless codes work more efficiently with large blocks. If more blocks are used, the download time gets worse for all AL-FEC codes.

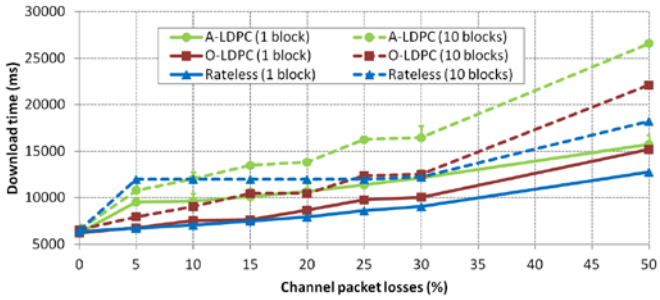


Fig. 12. Number of blocks evaluation with 3000 packet file size, b=5 Mbps and t_fd=3 s.

Note that Optimum LDPC outperforms rateless codes when more than one block is used, in cases where the channel packet loss rate is relatively low (e.g. 5%). When several blocks are used, the download time is, in general, determined by the number of cycles needed to download the file. So similar percentages of losses involve similar download times if the number of cycles is equal. In this case, the last block that has not been downloaded determines the download time. So, if a certain block has not been decoded in the current cycle, it is necessary to wait one entire cycle to try to download the complete file.

Finally, emphasize that all the graphs hereby presented have shown similar results for the analytical and the experimental model. Hence, the analytical model proposed for Adaptive LDPC AL-FEC codes is validated through experimental measurements.

## VII. Conclusions and Future Work

This paper proposes the implementation of Adaptive LDPC AL-FEC for multicast content distribution based on the FLUTE protocol. Adaptive AL-FEC codes represent a good alternative to improve the reliability of multicast connections over lossy channels, like wireless channels.

The different results show that it improves average download times to levels comparable to rateless codes, keeping the coding and decoding complexity of LDPC codes, but at the expense of an explicit feedback between clients and servers.

Despite that the ideal LDPC AL-FEC code rate depends on the amount of packets lost, a given AL-FEC code rate performs good in a wide interval of packet loss rates around the value for which it provides a minimum download time. This way, it is expected that the performance of Adaptive LDPC does not depend greatly on the accuracy of the channel packet loss estimation. However, results show that it depends considerably on the feedback time, defined as the time needed to provide clients with AL-FEC parity packets.

Although, in general, rateless codes offer better download times, different studies have shown that the ratios between

rateless codes and Optimum LDPC or Adaptive LDPC codes are not very high. On the contrary, the decoding complexity is much lower in LDPC codes, which makes these codes very recommended in receivers with limited resources, for instance, mobile devices. As mentioned, in these devices with limited resources it is recommended to send the data using several blocks, in order to reduce the decoding complexity. In that case the behavior of Adaptive LDPC codes is very similar, even better for some percentage of losses, than rateless codes, as the results have shown.

Precisely, one of the main points of the future work is the evaluation of Adaptive LDPC in mobile devices and the validation of the results presented in a wireless channel like Wi-Fi.

On the other hand, in channels with limited bandwidth it is recommended to use few parity channels. So it is necessary to choose dynamically the optimum code rate depending on the different feedback messages received by all the clients, in order to satisfy the major part of them. The way to choose this best code rate for all users is part of the future work.

## REFERENCES


[1] W. Lam and H. Garcia-Molina "Reliably networking a multicast repository," in *Proc. of the 22nd International Symposium on Reliable Distributed Systems (SRDS)*, pp. 5-14, Florence, Italy, Oct. 2003.

[2] J. Peltotalo, J. Harju, and M. Hannuksela, "Reliable, server-friendly and bandwidth-efficient file delivery system using FLUTE server file format," in *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BSMB)*, Bilbao, Spain, May 2009.

[3] E. Paolini, M. Varrella, M. Chiani, B. Matuz, and G. Liva, "Low-Complexity LDPC Codes with Near-Optimum Performance over the BEC," in *Proc. Advanced Satellite Mobile Systems (ASMS)*, pp. 274-282, Bologna, Italy, Aug. 2008.

[4] T. Paila, M. Luby, R. Lehtonen, V. Roca, and R. Walsh, "FLUTE – File Delivery Over Unidirectional Transport," IETF RFC 3926, Oct. 2004.

[5] M. Watson, M. Luby, and L. Vicisano, "Forward Error Correction (FEC) Building Block," IETF RFC 5052, Aug. 2007.

[6] V. Roca, Z. Khallouf, and J. Laboure, "Design and evaluation of a Low Density Generator Matrix (LDGM) large block FEC codec," in *5th International Workshop on Networked Group Communication (NGC)*, Munich, Germany, Sep. 2003.

[7] M. Watson, "Basic Forward Error Correction (FEC) Schemes," IETF RFC 5445, Mar. 2009.

[8] J. Lacan, V. Roca, J. Peltotalo, and S. Peltotalo, "Reed-Solomon Forward Error Correction (FEC) Schemes," IETF RFC 5510, Apr. 2009.

[9] A. Shokrollahi, "Raptor Codes," *IEEE Transactions on Information Theory*, vol. 52, pp. 2551-2567, Jun. 2006.

[10] M. Luby, A. Shokrollahi, M. Watson, and T. Stockhammer, "Raptor Forward Error Correction Scheme for Object Delivery," IETF RFC 5053, Sep. 2007.

[11] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, and L. Minder, "RaptorQ Forward Error Correction Scheme for Object Delivery," IETF RFC 6330, Aug. 2011.

[12] R. G. Gallager, "Low Density Parity Check Codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21-28, Jan. 1962.

[13] T. J. Richardson, A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 619-637, Feb. 2001.

[14] V. Roca, C. Neumann, and D. Furodet, "Low Density Parity Check (LDPC) Staircase and Triangle Forward Error Correction (FEC) Schemes," IETF RFC 5170, Jun. 2008.

[15] V. Roca and C. Neumann, "Design, evaluation and comparison of four large block FEC Codecs, LDPC, LDGM, LDGM Staircase and LDGM Triangle, plus a Reed-Solomon small block FEC codec," INRIA Research Report RR-5225, Jun. 2004.

[16] J. Peltotalo, S. Peltotalo, J. Harju, and R. Walsh, "Performance analysis of a file delivery system based on the FLUTE protocol," *International Journal of Communications Systems*, vol. 20, no. 6, pp. 633-659, Jun. 2007.

[17] H. Bai and M. Atiquzzaman, "Error modeling schemes for fading channels in wireless communications: A survey," *IEEE Communications Surveys and Tutorials*, vol. 5, no. 2, pp. 2-9, Fourth Quarter 2003.

[18] A. Downey, "The structural cause of file size distributions," in *9th International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Cincinnati, USA, Aug. 2001.

[19] ETSI TS 102 034 v1.4.1, "Digital Video Broadcasting (DVB); Transport of MPEG-2 TS based DVB Services over IP based networks," Aug. 2009.

[20] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," in *Proc. ACM SIGCOMM 98*, pp. 56-67, Vancouver, Canada, Sep. 1998.

[21] I. de Fez, F. Fraile, R. Belda, and J. C. Guerri, "Performance evaluation of AL-FEC LDPC codes for push content applications in wireless unidirectional environments," *Multimedia Tools and Applications*, Ed. Springer Netherlands. Available online: http://www.springerlink.com/content/n53104h562685118, Jul. 2011.

[22] I. de Fez, F. Fraile, R. Belda, and J. C. Guerri, "Evaluation of adaptive LDPC AL-FEC codes for content download services," in *IEEE International Conference on Multimedia and Expo (ICME)*, Barcelona, Spain, Jul. 2011.


**Ismael de Fez** was born in Valencia, Spain. He received his Telecommunications Engineering degree and the M.S. in Telematics from the Universitat Politècnica de València (UPV), Spain, in 2007 and 2010 respectively. Currently he works as a researcher at the Multimedia Communications research group (COMM) of the Institute of Telecommunications and Multimedia Applications (iTEAM), where he is working toward his Ph. D. degree. His areas of interest are file transmission over unidirectional environments and file encoding.

**Francisco Fraile** was born in Murcia, Spain. He obtained a degree in Telecommunication Engineering from the Universitat Politècnica de València (UPV) and a M. Sc. Degree in Microwave Engineering from the University of Gävle in 2004. Since then, until 2010, he has worked as a research engineer for the Swedish company Interactive TV Arena. In 2006, Francisco joined the Multimedia Communications research group (COMM) of the iTEAM Institute, to proceed with his doctoral studies as an industrial Ph.D. student, interested in networked electronic media.

**Román Belda** was born in Alzira (Valencia), Spain. He received his Computer Science degree from the Universitat Politècnica de València (UPV), Spain in 2004 and he is currently studying his M.S. in Telematics. Currently he works as a researcher at the Multimedia Communications research group (COMM) of the iTEAM Institute. His areas of interest are mobile applications and multimedia transmission protocols.

**Juan Carlos Guerri** was born in Valencia. He received his M.S. and Ph. D. (Dr. Ing.) degrees, both in Telecommunication Engineering, from the Universitat Politècnica de València (UPV), in 1993 and 1997, respectively. He is a professor in the E.T.S. Telecommunications Engineering at the Universitat Politècnica de València, and he leads the Multimedia Communications research group (COMM) of the iTEAM Institute. He is currently involved in research and development projects for the application of multimedia to industry, medicine, education and communications.