De Fez Lava, I.; Fraile Gil, F.; Belda Ortega, R.; Guerri Cebollada, JC. (2012). Performance evaluation of AL-FEC LDPC codes for push content applications in wireless unidirectional environments. Multimedia Tools and Applications. 60(3):669-688. doi:10.1007/s11042-011-0841-y.

Editorial Manager(tm) for Multimedia Tools and Applications
Manuscript Draft

Manuscript Number:

Title: Performance evaluation of AL-FEC LDPC codes for push content applications in wireless unidirectional environments

Article Type: Manuscript

Keywords: LDPC; FLUTE; AL-FEC; Push content download service; Wi-Fi; Broadcast.

Corresponding Author: Juan C. Guerri, Dr.

Corresponding Author's Institution: ETSI Telecomunication

First Author: Ismael de Fez, Engineer

Order of Authors: Ismael de Fez, Engineer;Francisco Fraile, Engineer;Roman Belda, Engineer;Juan C. Guerri, Dr.

Performance evaluation of AL-FEC LDPC codes for push content applications in wireless unidirectional environments

Ismael de Fez[a], Francisco Fraile[a], Román Belda[a], Juan C. Guerri[a]

[a] *Institute of Telecommunications and Multimedia Applications (iTEAM), Universidad Politécnica de Valencia, Camino de Vera, 46071, Valencia, Spain*

e-mail: isdefez@iteam.upv.es, ffraile@iteam.upv.es, robelor@iteam.upv.es, jcguerri@dcom.upv.es
phone: 34-963879588

Abstract. FEC (Forward Error Correction) mechanisms improve IP content transmission reliability through the recovery of packets lost in transmission. Opposite to ARQ (Automatic Repeat Request), FEC mechanisms are especially suited to unidirectional environments or to multicast environments where multiple receivers perceived different channel losses, thus making difficult the implementation of mechanisms based on feedback information. Among the different types of FEC codes, this paper presents a thorough performance evaluation of LDPC (Low Density Parity Check) codes, based on an implementation developed by the authors, according to the specifications defined by RFC 5170 for the usage of LDPC codes by push content applications based on the FLUTE protocol. LDPC codes provide a good trade-off between performance and complexity, hence, they are appropriate for mobile applications. Contributions of this paper include tests conducted with commercial mobile phones connected to the push content download server over a Wi-Fi network. The evaluation highlights the advantages of using packet level FEC encoding in file transmission over unidirectional networks and provides with a comparison between two kinds of LDPC structures: Staircase and Triangle. This is accomplished by calculating the inefficiency ratio of these LDPC structures in different environments. Results show that the implemented LDPC codes can provide inefficiency ratios close to one when the different coding parameters (as the code rate or the number of blocks) are configured to an optimal value that depends on the packet loss rate.

Keywords: *LDPC, FLUTE, AL-FEC, Push content download service, Wi-Fi, Broadcast.*

1. Introduction

In the last years, the use of wireless networks has increased dramatically, to the point that wireless technologies, such as Wi-Fi, DVB, 3G or Bluetooth, play an important role in our society. One of the main aspects on the design of these

technologies is the reliability provided in the transmission since, unfortunately, dealing with the problems caused by error-prone communication networks is also part of modern everyday life. In order to improve communications in the presence of errors, the use of error protection mechanisms is needed, since there are losses in wireless networks. In a transmission medium, wired or wireless, there are errors in the channel, so the use of tools that detect and correct these errors is something essential in a communication system.

Generally, there are two main error correction techniques, ARQ (Automatic Repeat Request) and FEC (Forward Error Correction). The former consists of retransmitting data that are missed in the communication, whereas FEC allows to reconstruct the original data without retransmissions, through error correction encoding. There are different categories of FEC codes: convolutional codes, block codes, fountain codes and hybrid systems. FEC is mainly used in unidirectional environments, where a return channel does not exist.

Error correction is generally applied in the lower layers of a communication system, although it can be used at higher layers. Specifically, AL-FEC (Application Layer FEC) provides additional robustness to certain services without any modification in the lower layers of a system, through applying FEC coding at transport packet level. Thus, the use of AL-FEC is particularly interesting for provisioning new services over communication networks already in place, since AL-FEC can increase the native reliability of the network to meet the requirements of a specific service, without additional infrastructure. Moreover, AL-FEC may improve the performance of content transfer through wireless communication networks, as it can decrease download times as well as network traffic, since it avoids the request of lost packets.

This paper is focused on the analysis, implementation and evaluation in unidirectional environments of Low Density Parity Check (LDPC) codes AL-FEC for push content download services over broadcast wireless networks. The rest of the paper is structured as follows: the next section describes the different technologies involved in file transfer services over wireless networks. Section 3 includes a brief overview of different AL-FEC codes, focusing on LDPC codes. Section 4 is dedicated to the implementation of the LDPC library. Section 5 presents the results of the evaluation and lastly, section 6 contains some final conclusions.

2. Push content download services over wireless networks

2.1 Wireless networks standards

Nowadays, there are a lot of standardized wireless networks that support IP broadcast and/or multicast from standardization bodies such as DVB, 3GPP or IEEE.

DVB (Digital Video Broadcasting) is an organization which its main objective is the creation of digital television standards and data broadcasting services. DVB has produced several standards useful to broadcast IP datagrams to mobile devices from terrestrial (DVB-H) [1] or hybrid terrestrial/satellite networks (DVB-SH). DVB is currently developing the second generation of mobile broadcast standards.

On the other hand, 3GPP has developed several specifications to broadcast services via existing cellular networks, such as MBMS (Multimedia Broadcast and Multicast Services) [2] and IMB (Integrated Mobile Broadcast) [3]. Moreover, multicast is also regarded in the LTE (Long Term Evolution) [4] specifications, which support MBMS and E-MBMS (Evolved MBMS).

Lastly, the IEEE 802.11 standards for WLAN (Wireless Local Area Network) [5] and the IEEE 802.16 standards for BWA (Broadband Wireless Access) [6] also support IP broadcast and multicast. To date, WLANs are by far the most widespread wireless technology with support for IP broadcast and multicast.

In order to reconstruct a file successfully, a client must receive all packets that compose it. Therefore, file transfers must not experience any errors at the topmost application layer of the protocol stack. The main problem is that, as explained, WLAN (and the rest of wireless networks) lacks of mechanisms to guarantee error free broadcast file transfers. Thus, in push content download services [7] it is necessary to use a protocol that guarantees that receivers can recover the original data in the event of errors. The most used protocol in broadcast content download services to mobile devices is FLUTE.

2.2 FLUTE

FLUTE (File Delivery over Unidirectional Transport), defined in RFC 3926 [8], is a protocol for the unidirectional delivery of files over the Internet, which is particularly suited to multicast networks. Its main characteristic is that this protocol offers reliability in the transmission. Moreover, it provides massive scalability, management and congestion control, and it is useful to send metadata. In fact, DVB-H uses FLUTE to send the Electronic Service Guide (ESG).

FLUTE file transfers [9] 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 in a port number and with a given transmission rate. The files sent through the channels of a session are identified by the TOI (Transport Object Identifier), a numeric identifier.

To start receiving a file delivery session, the receiver needs to know the transport parameters associated with the session. This information can be sent out-of-band, through methods such as HTTP/Mime headers or SAP protocol. But the most used form is through the SDP (Session Description Protocol) protocol [10]. Session Description must include the parameters that identify a session: source IP address and TSI. Moreover, the Session Description can include additional information such as the number of channels or congestion control algorithm used.

Once the clients have the necessary information to join to a session and have established the connection, they can receive files. But before, they must know which files are being transmitted within the session and their characteristics. This information is obtained by means of the File Delivery Table (FDT). FDT provides a means to describe various attributes associated with the files sent through the session. The most important attributes are: object identifier (TOI), location and file name (specified as URI), file length and the encoding, among others. FDT is written in XML language and is delivered through FDT Instances, which are FLUTE packets with a FDT header extension. The XML is the payload of the packet to send. The value “0” of TOI is reserved to identify a packet as FDT.

Figure 1 shows the protocol stack used by FLUTE. ALC protocol [11] provides the basic transport to FLUTE. At the same time, ALC uses the LCT (Layered Coding Transport) [12] building block for session management

functionalities, the congestion control (CC) block, as well as the FEC block [13] used for error control. In the lower layers, FLUTE works over UDP (User Datagram Protocol) on transport level and IP (Internet Protocol) on network level.

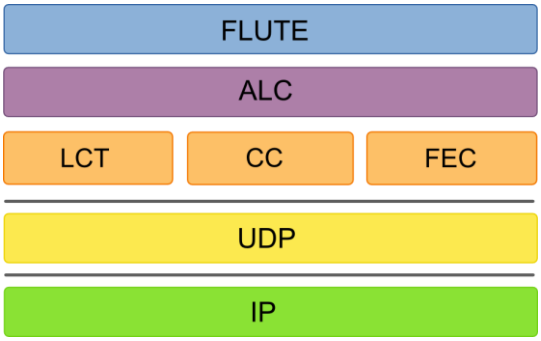


Figure 1 FLUTE protocol stack

Each file represents a transport object. As figure 2 shows, each transport object is fragmented in blocks, using an algorithm defined by FLUTE. Also, each block is composed of encoding symbols. There are two types of encoding symbols: source and parity symbols. The first ones conform the original data of the file, whereas the parity symbols are created from a combination of source symbols, through FEC encoding, to provide reliability on the transmission. Thus, each block contains *n* encoding symbols, *k* of which are source symbols. If encoding is employed, the number of parity symbols per block will be *n-k*. Finally, each FLUTE packet contains an integer number of encoding symbols from a source block as payload.

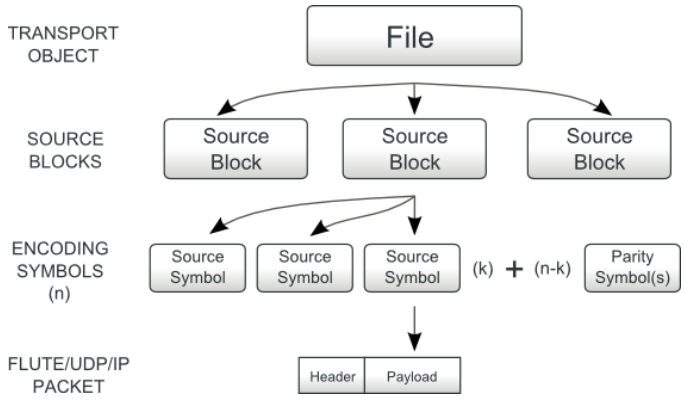


Figure 2 FLUTE packet construction

As for the organization of file transmissions, there are two different kinds of FLUTE delivery sessions: file transmission sessions and file carousels. In the latest, files are sent cyclically on a seamlessly endless loop. Furthermore, sessions can be static or dynamic, depending on whether the contents of the session change

during its lifetime. The most used kind of sessions are file carousels. In fact, the use of carousels, together with FEC mechanisms, is what provides reliability on the transmission, the main characteristic of FLUTE protocol.

3. FEC Codes

The FEC codes supported by FLUTE are: Compact No-Code, Raptor, Reed-Solomon over GF (2^m), Reed-Solomon over GF (2^8), LDPC Staircase and LDPC Triangle. Compact No-Code [14] implies not using any coding mechanism, i.e. only source packets are sent. In the next sections the other codes are briefly explained.

3.1 Raptor

Raptor codes [15] were created in 2001 by "Digital Fountain Inc." company. These codes belong to the fountain codes category, which allows generating as many symbols as needed on the fly from the source symbols of a block, that is, a fixed code rate is not needed. Despite being a proprietary implementation, these codes have been adopted by several technologies. One of these is the DVB-H standard. Their main characteristic is that these codes are able to generate infinite parity information. Moreover, receivers need only few packets more than the number of packets that makes up the file for reconstructing it, independently of the type of the received packets. That is the reason why these codes are very efficient. Furthermore, the encoding and decoding are very fast, so their implementation in software is easy.

The encoding process is divided in two steps: first, a precoding is done, in which l output packets are created through k input packets ($l>k$). The second step consist of the creation of the n source symbols through the l precoded symbols ($n>l$), using LT (Luby Transform) codes [16], a kind of fountain codes. Each symbol is generated independently, and it is possible to create an unlimited number of symbols. RFC 5053 [17] describes deeply the creation of the symbols and the header format related to Raptor.

3.2 Reed-Solomon

Reed-Solomon codes, invented in 1960, are used for a lot of applications, such as data storage (for instance in CD or DVD), in wireless networks (mobile phones) o by satellite, in wired communications (ADSL) and in digital television (DVB uses Reed-Solomon to correct errors in the physical layer).

Reed-Solomon is an error corrector block code based on polynomials, and creates symbols by means of *m*-bits sequences. Each code word is composed of *n* symbols, which *k* are source symbols and *r* are parity symbols. The relation between the code word length and the number of symbols is defined by: $n=2^m-1$. These codes are able to correct errors even in *r/2* symbols.

RFC 5510 [18] defines the FEC schemes for Reed-Solomon codes over GF (2^8) and over GF (2^m). In both cases, the creation of the *n* symbols through the *k* symbols that make a block is produced by means of a generation matrix. That matrix uses a polynomial which depends on the length of the m-finite field elements.

3.3 LDPC

LDPC (Low Density Parity Check) codes were invented by Gallager in 1960 [19]. But until 30 years later they were not used, thanks to MacKay and Neal [20]. The original specification has suffered some improvements that make easy their utilization in different environments. For instance, they are the base of Tornado, LT and Raptor codes, all these proprietary implementations. LDPC belongs to the large block codes category, in which it is needed to receive more of the *k* packets that make up a file for reconstructing it. The codes included in this category are advisable when large files are encoded, since computational cost does not grow excessively.

Low Density Parity Check codes are systematic lineal block codes based on a parity check matrix used in the encoding and decoding processes. This matrix defines the relations between the different encoding symbols (source symbols and parity symbols). The matrix consist of some elements with values “0” and “1”, and it is disperse, since the most part of the elements are null. By means of the matrix the encoder generates the parity symbols through the source symbols (and other parity symbols generated). Also, in reception, the matrix is used to reconstruct the symbols that have not been received, through the encoding

symbols already received. Figure 3 shows an example of a parity matrix, and establishes the relations between source and parity symbols.

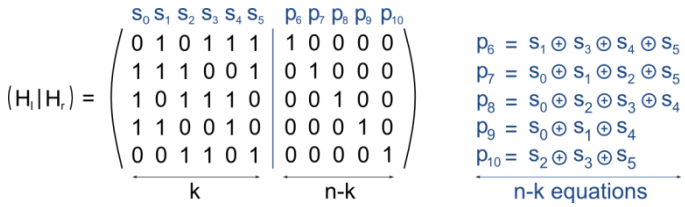


Figure 3 LDPC parity check matrix (k=6, n=11)

The figure depicts a matrix with values *k*=6 and *n*=11, which generates 5 parity symbols per block. The size of the matrix is *(n-k)* x *n*, so there are *n–k* rows, each one representing an equation. The columns are related to the symbols of the block. Each element of the matrix with the value "1" ($h_{ij} = 1$) indicates that the *j*-th symbol takes part in the *i*-th equation. Thus, for instance, the first parity symbol (identified as p_6) is composed of the XOR sum of the s_1, s_3, s_4 and s_5 symbols. Receivers are able to recover a symbol from an equation once they have successfully received all other symbols that take part in the given equation.

Moreover, a parity symbol can take part in the creation of other parity symbols. In general, each source symbol takes part in a fixed number of equations, that is, the number of 1s that contains the corresponding column. That parameter is called *N1*. The number of non-null elements of a row or column is called degree.

On the other hand, the matrix is divided in two sub-matrixes: the left and the right ones. The first refers to source symbols, whereas the right sub-matrix refers to parity symbols. 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 (*N1*) and seed used to generate the pseudorandom numbers. The sender signals all these parameters in an extension of the LCT header so that receivers can generate the exact same matrix used by the encoder.

Depending on the parity check matrix structure there are two kinds of LDPC codes: regular codes and irregular codes. In the first ones, all the rows of the matrix have the same degree and all the columns have the same *N1* value,

while irregular LDPC codes do not fulfill either condition. Gallager and Mackay codes are example of LDPC regular codes, whereas LDPC Staircase and Triangle are irregular codes.

In this sense, this paper is focused on the implementation and analysis of LDPC Staircase and LDPC Triangle codes, which only differ on the right sub-matrix generation, as figure 4 shows. In the LDPC Triangle structure, the degree of each row is equal or higher than that of the LDPC Staircase structure.

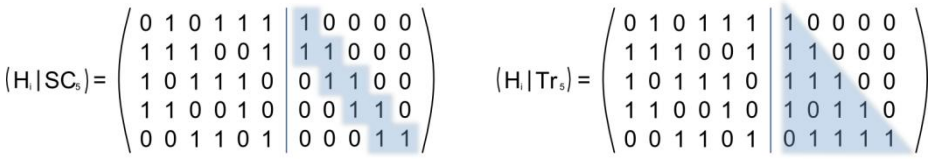


Figure 4 Example of LDPC Staircase and LDPC Triangle matrix (k=6, n=11)

4. Implementation

4.1 RFC 5170 specifications

RFC 5170 [21], *LDPC Staircase and Triangle FEC*, from June 2008, introduces the LDPC-Staircase FEC codes and the LDPC-Triangle FEC codes. Both schemes belong to the broad class of large block codes, according to the definition of FEC RFC [14].

RFC 5170 defines the parity matrix generation in both structures. For that, it provides an algorithm that creates the parity matrix using certain input parameters. In both schemes, the algorithm is the same for the left sub-matrix but different for the right sub-matrix. For the creation of the matrix, the RFC proposes the use of the pseudorandom number generator algorithm of Park-Miller [22]. The RFC, as well, defines the fields of LCT header extension EXT_FTI for LDPC, which includes the coding parameters.

There is an open-source implementation of the LDPC Staircase and LDPC Triangles codes for FLUTE applications [23], developed by the INRIA research institution, whose authors also participated in the development of the RFC 5170.

In this sense, this paper presents an own implementation of a content broadcast architecture that uses LDPC codes for data protection. That implementation fulfills the requirements of the RFC 5170. In contrast to [23], the

library hereby presented is specifically developed for mobile devices. The next sections present the developed LDPC Staircase and Triangle codecs.

4.2 File server structure

Figure 5 shows the architecture of the push content download server implementation, based on the FLUTE protocol. The FLUTE session and channels management and their delivery through ALC protocol is done by means of the corresponding classes. The packet delivery is carried out with a rate fixed by "RateControl" class, using a transmission model managed by the "Scheduler" block.

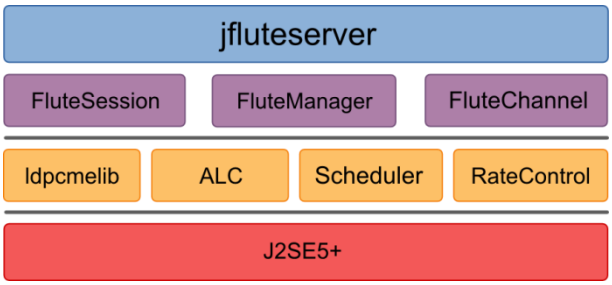


Figure 5 Push content download server structure

The "ldpcmelib" library implements the LDPC encoder and decoder (both schemes Staircase and Triangle). It is developed in J2ME, in order to be used by mobile devices. This library creates the parity matrix, which defines the relation between source and parity symbols. Also, the transmitter creates the header including the coding parameters. This way, the receiver can generate the same parity matrix and do the decoding.

4.3 File client structure

The FLUTE client structure, shown in figure 6, is very similar to the structure of the server and share most of the code.

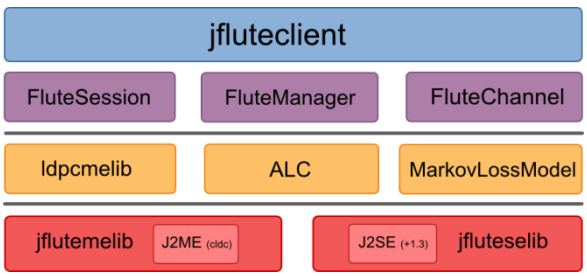


Figure 6 File client structure

The client is designed to support two different scenarios: mobile phone devices and a wired environment that emulates losses in the channel. In order to simulate these losses, a two state Markov Model has been implemented. We have chosen this model because it simulates well the burst losses (typical in wireless networks) and because it is widely used in literature [24].

In the decoding process, the algorithm used is a key factor that affects the decoding efficiency and the energy consumption on the receiver. In this sense, some studies can be found in [25]. In our study, the decoding is performed using the iterative decoding algorithm, as the flow chart of figure 7 shows. When a new packet arrives, if the packet has not been received previously, the client obtains the parity matrix associated to the block which the symbol belongs to and checks the rows related to that particular symbol. In our algorithm, the decoding is based on partial sum buffers in each row of the parity matrix. Each buffer contains the XOR sum of the received symbols of a row. When all packets of a row except one have been received, the data of the non-received symbol is the partial sum of the buffer of that row. This way, it is possible to reconstruct a symbol that has not been received yet.

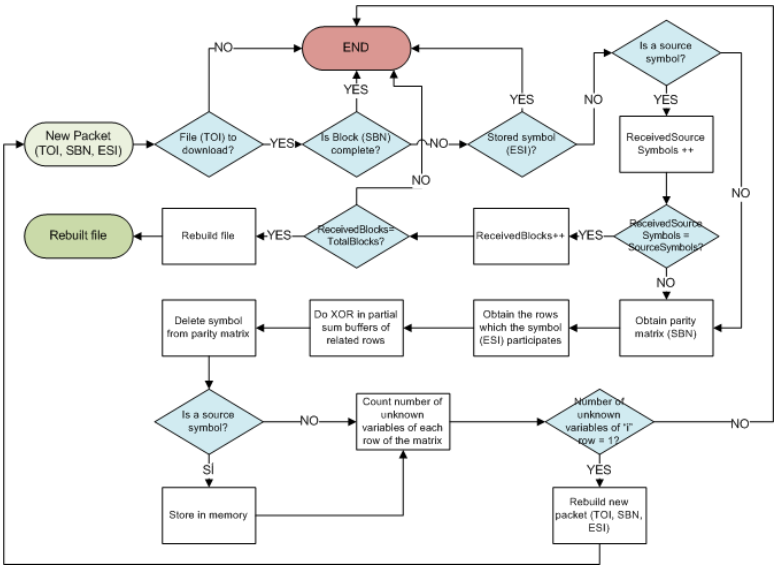


Figure 7 Packet reception flow chart

5. Performance evaluation

The performance of the implementation of LDPC codes has been assessed through several tests that are explained in the following sections. The parameters to evaluate are:

- *Inefficiency ratio*: represents the relation between the number of packets needed to decode a file and the number of source packets that make up the file. The less inefficiency ratio the more efficient is the coding. Ideally this value is 1.

$$inefficiency\_ratio = \frac{n_{necessary\_for\_decoding}}{k}$$

- *Number of carousel cycles* needed to rebuild the file.

In the tests carried out two different scenarios have been proposed, as figure 8 shows. In the first scenario the server and the client are in the same machine to avoid uncontrolled packet loss in the network. In order to simulate packet losses in the channel, a two state Markov model has been implemented in the FLUTE client.

In the second scenario, the FLUTE client is a mobile phone and connects to the server through a Wi-Fi channel, since it is one of the most used wireless networks. In this sense, one of the main contributions of this paper is the evaluation of LDPC codes with mobile devices through wireless networks. For these tests Nokia E90 has been used, since this device is a representative of Smartphones' family with operative system Symbian S60, which has the most number of smartphone devices in the market.

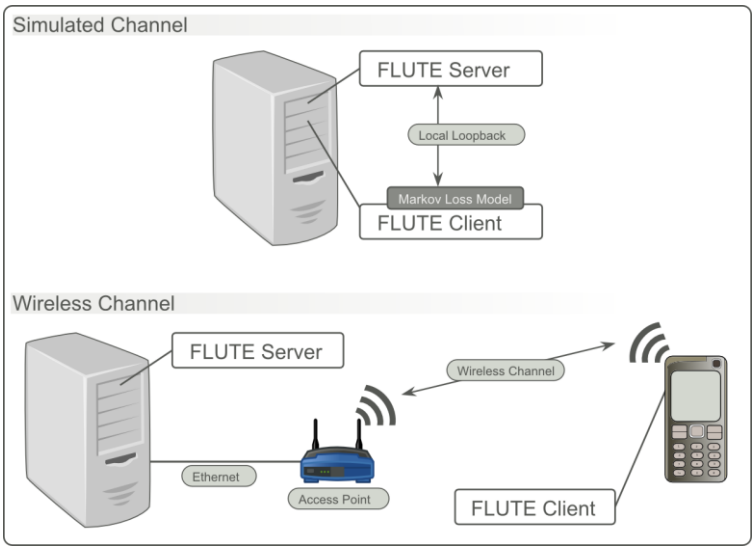


Figure 8 Testbed

In order to see the losses detected in the wireless channel in our scenario, a study over the Wi-Fi multicast losses detected is presented. The study has been carried out in a typical laboratory indoor environment, in which there are several computers and access points. The measurements assessed the number of packets per cycle received by the mobile terminal, so it is possible to calculate the percentage of lost packets. The figure 9 shows the results of a study made between 9.30 am and 1.00 pm.

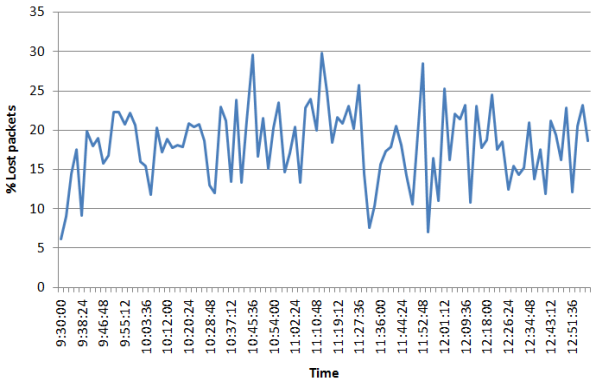


Figure 9 Evaluation of losses in Wi-Fi

As figure shows, the percentage of losses is time-dependent. In general, the percentage of losses in our trial environment is between 15 and 25%. In order to obtain accurate measurements, the tests carried out (which are presented in the next sections) have been done in different days and hours, using different transmission rates.

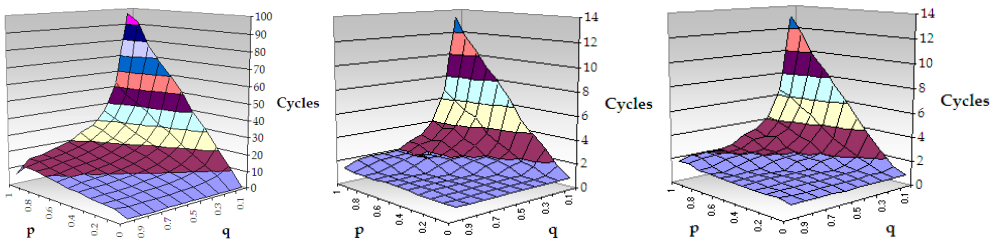
Table 1 shows the coding parameters used in each study, emphasizing in italics the ones evaluated in each case. The file size is expressed in packets with a payload size of 1428 bytes. The number of measurements accomplishes good 99% confidence intervals in all scenarios.

Study	**Section 5.1**	**Section 5.2**	**Section 5.3**	**Section 5.4**	**Section 5.5**	**Section 5.6**
Evaluation parameter	Number of cycles	Inefficiency ratio	Inefficiency ratio	Inefficiency ratio	Inefficiency ratio	Inefficiency ratio
Tx. Model	Sequential	*Sequential, Random*	Random	Random	Random	Random
Code rate	2/3	2/3	*[0.2, 0.9]*	2/3	2/3	2/3
File size	1500	1500	1500	*[10, 10000]*	1500	1500
Blocks	1	1	1	1	*[1, 150]*	1
N1	3	3	3	3	3	*[3, 8]*
Emulations	100	100	100	100	100	100
Channel	Simulated	Simulated	Simulated, Wireless	Simulated, Wireless	Simulated, Wireless	Simulated, Wireless

Table 1 Study parameters

5.1 Number of rebuilding cycles

The first study shows the number of cycles that one client needs to rebuild a file based on the channel losses, which are simulated with a two state Markov model. Remember that, in this model, *p* indicates the probability that a packet is lost when the previous was received, and *q* indicates the probability of the opposite transition. The number of cycles is directly related with the download time of a file, so it represents a key parameter. The graphs of figure 10 capture the results obtained.



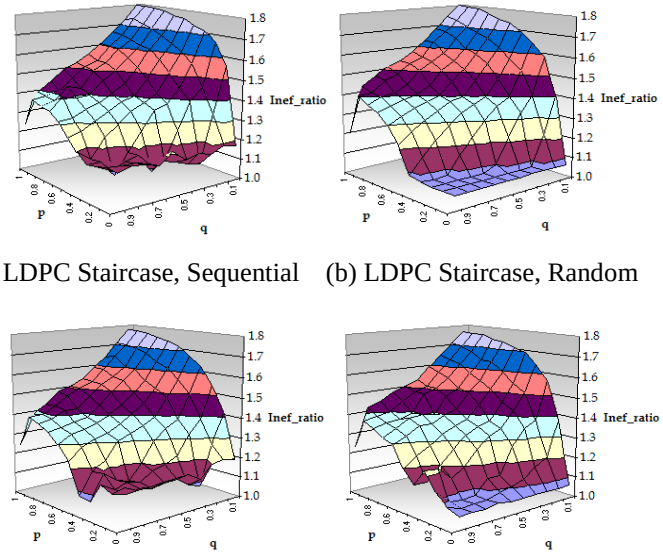
(a) No-FEC (b) LDPC Staircase (c) LDPC Triangle

Figure 10 Number of cycles depending on coding

Seeing the scale of each graph, we can clearly see the convenience of using coding (in LDPC 15 cycles are not exceeded, whereas in No-FEC it arrives until almost 100 cycles with high losses). The tendency is the same in the three codes, but the difference between them is higher when the losses increase. In low-loss environments (that is, when *p* is low and *q* is high), the graphs show that LDPC codes (both Staircase and Triangle) presents a more stable behavior and close to 1, whereas when no coding is used the number of cycles grow fast with a slight increase of the losses.

5.2 Transmission model

This study shows how the transmission model affects the coding efficiency. To that effect, two models are analyzed: a sequential model, which packets are sent in order (first source symbols and then parity symbols); and a random model, where packets are transmitted randomly (inserting source and parity symbols). The measure parameter used is the inefficiency ratio. The result is shown in the graphs of figure 11.



(a) LDPC Staircase, Sequential (b) LDPC Staircase, Random

(c) LDPC Triangle, Sequential (d) LDPC Triangle, Random

Figure 11 Transmission model evaluation

The figures show that in typical lossy environments (low *p* and high *q*), the random transmission model has a better behavior and proves more efficient than the sequential one. That is logical if we consider that losses are usually produced in bursts and that in LDPC codes a parity symbol depends on the previous symbol, so the loss of consecutive packets prevents the rebuild of the source symbol. With high losses the behavior of both models is similar.

5.3 Code rate

The code rate is a basic parameter of the push content download service. It is defined as *k/n*, that is, it represents the relation between the number of source symbols and the number of encoding symbols of a file. The number of parity symbols is, hence, *k-n*. This way, higher code rates imply less information protection. Another parameter used is the FEC ratio, defined as *n/k*, which is the inverse of the code rate.

Figure 12 shows how the code rate affects the inefficiency ratio in a lossless channel (the code rate axis has been expanded in order to see in detail the behavior of each structure).

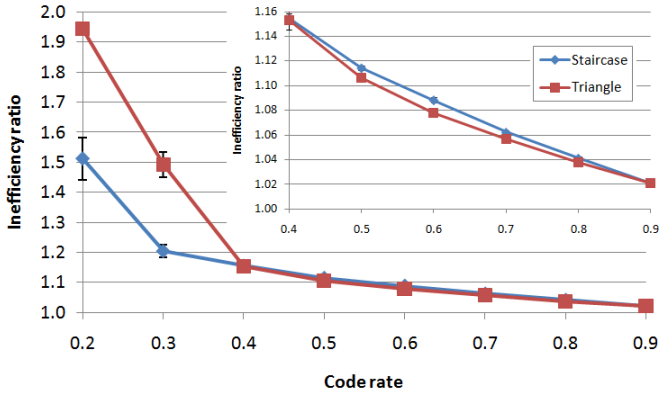


Figure 12 Code rate evaluation in a lossless channel

The higher the code rate, the lower (and better) the inefficiency ratio. Figure shows that LDPC Staircase structure is more efficient when the code rate is lower than 0.4, whereas LDPC Triangle provides better results for code rates higher than this value. Although for code rate values larger than 0.4 the difference between both structures appears to be small, it could be very meaningful when big files are sent.

As the code rate is higher (and therefore FEC ratio lower), less parity packets are sent, so in lossless environments the inefficiency ratio will be lower (since less "useless" packets are received). Ideally, in a lossless channel, if the code rate is 1 (that is, no coding is used) the inefficiency ratio is 1. But, unfortunately, most of the channels have losses. Before seeing the analysis in a wireless environment, we study the behavior in a loss environment, using the two state Markov model. The results in figure 13 show the evaluation of the inefficiency ratio of LDPC codes in an emulated channel with a packet loss rate of 25% and parameters p=0.1 and q=0.3.

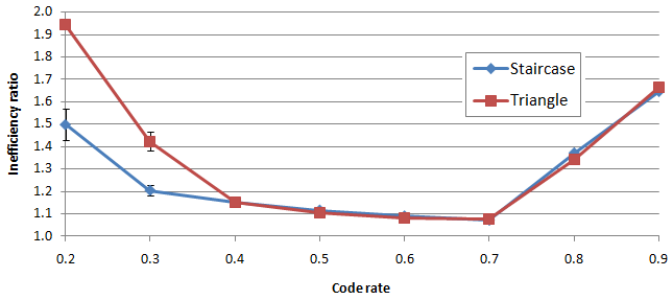


Figure 13 Code rate evaluation in a loss channel (p=0.1, q=0.3)

We could conclude that for choosing a suitable code rate it is necessary to bear in mind the losses of the channel. Using high code rates could cause that the information is not protected appropriately, hence increasing the inefficiency ratio. For instance, for the channel evaluated in figure 13, the best code rate is 0.7.

The behavior of the code rate has been tested in a Wi-Fi wireless environment with a mobile device. Figure 14 gathers the results of this study, where the conclusions reached in the previous studies still hold. LDPC Staircase is more efficient with code rates lower than 0.4, whereas LDPC Triangle, in general, is better in the other cases. Depending on the channel, there are code rates that minimize the inefficiency ratio. The values of the inefficiency ratio are rather higher than in the figure 12, due to the losses of the channel.

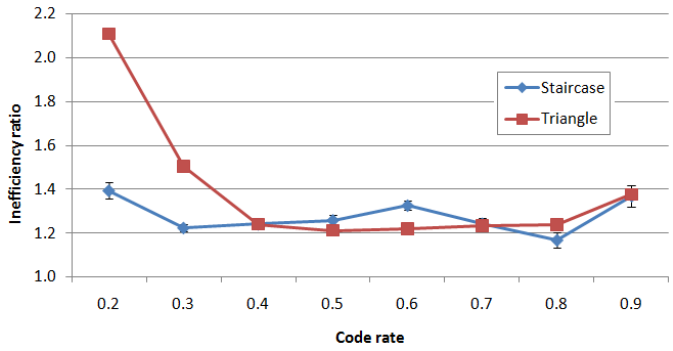


Figure 14 Code rate evaluation with a mobile device in Wi-Fi environment

5.4 File size

As we have seen, using any coding mechanism makes the communication more efficient. This improvement depends on the size of the information that is sent. Figure 15 shows the inefficiency ratio measured in a wireless channel with No-FEC and LDPC (an average of LDPC Staircase and LDPC Triangle codes) depending on file size.

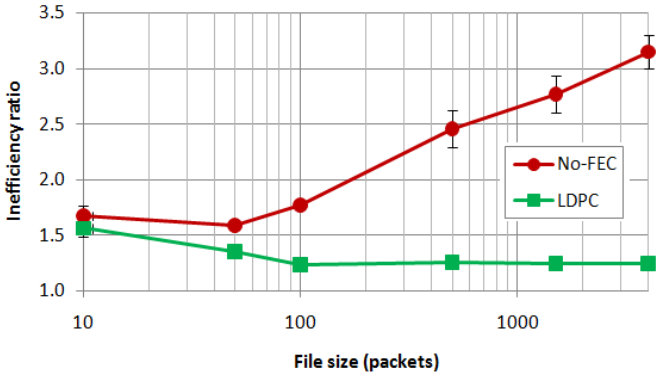


Figure 15 Comparison between No-FEC and LDPC depending on file size in a Wi-Fi channel

The behavior of the two coding mechanisms is completely different. In No-FEC the larger the file size, the higher (and worse) the inefficiency ratio, whereas in LDPC is the opposite. The advantages of using FEC coding are more evident when large files are sent.

A deeper study of LDPC depending on file size will be explained next. First, using a channel with no losses. The results are shown in figure 16.

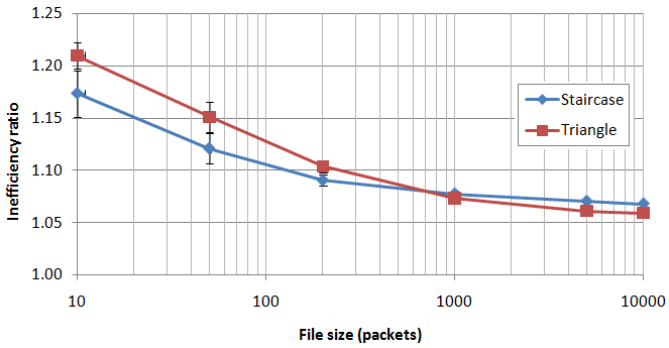


Figure 16 File size evaluation in a lossless channel

LDPC codes are more efficient when large files are sent, as graph shows. For instance, with files of 10000 packets size (over 14 Mbytes), for Triangle structure the inefficiency ratio is 1.0593. This means that it only is needed to receive a 5.93% more of the packets which make up a file to rebuild it. That is, reliability is being provided to the communication but without increasing the rebuild time in reception excessively.

With regard to the LDPC structure, the graph's tendency proves that Staircase offers better results than Triangle with small files, whereas with large files LDPC Triangle has a better inefficiency ratio.

The study in a wireless environment reflects the same behavior of both structures, as figure 17 shows.

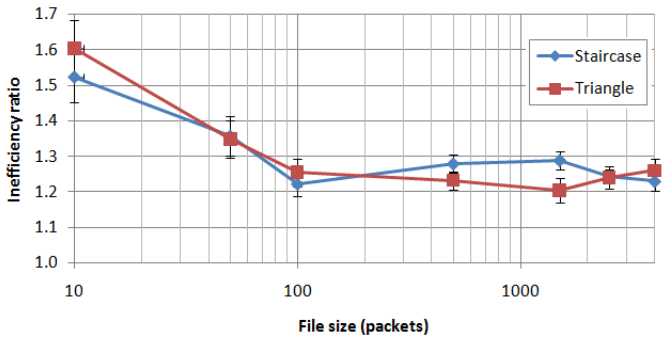


Figure 17 File size evaluation with a mobile device in a Wi-Fi channel

The conclusions reached regarding file size and code rate are in accordance with those found in [26].

5.5 Number of blocks

A related study is the number of blocks in which a file is divided. In this sense, figure 18 shows the inefficiency ratio measured when the number of blocks changes.

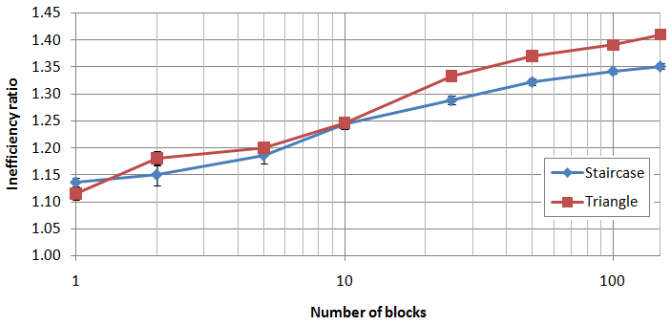


Figure 18 Number of blocks evaluation in a lossless channel

The inefficiency ratio increases with the number of blocks used and therefore, in terms of efficiency, it is better to use one block in the delivery of files. That is logical considering that, if a high number of blocks is used, each block will have fewer packets and, as we have seen before, LDPC codes are less efficient with small files. Nevertheless, it could be convenient to use more than one block in order to reduce the memory consumption.

Figure 19 shows the behavior in a mobile device using a Wi-Fi channel.

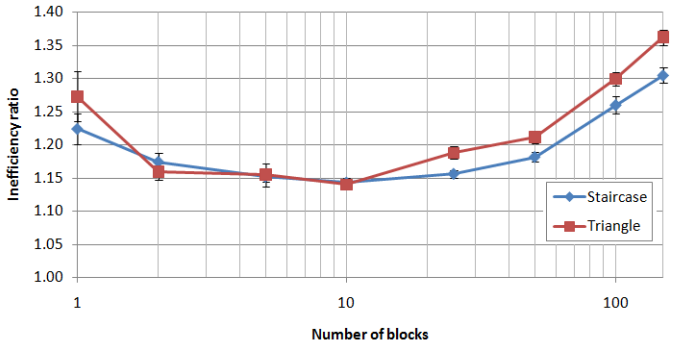


Figure 19 Number of blocks evaluation with a mobile in a Wi-Fi channel

The results are very similar to those in figure 18, except for the value of 1 block. The tendency is the same: the higher the number of blocks, the higher the inefficiency ratio. The LDPC Staircase structure has a better behavior than Triangle when the number of blocks increases.

5.6 Number of 1s in the parity check matrix

In the parity matrix creation (specifically in the left submatrix), each source symbol could be part of a certain number of equations (*N1*). This number is fixed

for each matrix and it is usually equal to 3, as LDPC RFC [21] recommends. Figure 20 shows the inefficiency ratio obtained when *N1* varies between 3 and 8 (values under 3 are not allowed), in an evaluation through a lossless channel.

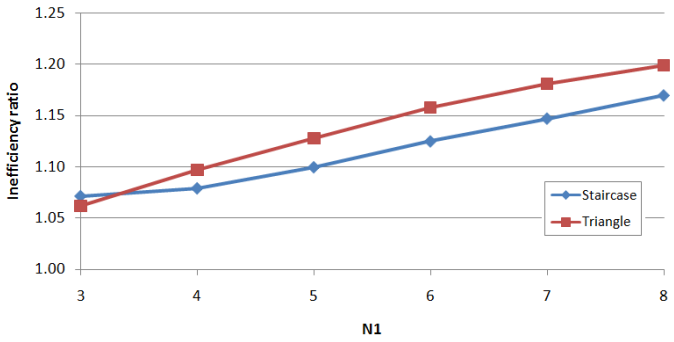


Figure 20 N1 evaluation in a lossless channel

Similar results are obtained when the same study is done in a wireless environment, as the figure 21 shows.

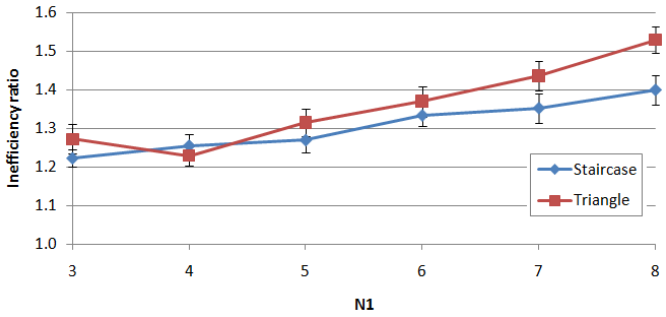


Figure 21 N1 evaluation with a mobile device in a wireless channel

Both figures show that the inefficiency ratio increases when the *N1* parameter is higher. Moreover, the Staircase structure results more efficient than Triangle when *N1* grows.

Nevertheless, the results depend on the decoding algorithm used. In our case, we have used the iterative decoding algorithm due to its simplicity and its low memory consumption. Several studies, as [27], show that using another decoding algorithm (for instance the one based on Gaussian elimination scheme) allows to reduce the inefficiency ratio. That study reflects that, using a Gaussian elimination scheme, the increase of *N1* means a lower inefficiency ratio for LDPC Staircase, at the expense of increasing the memory consumption.

Therefore, we conclude this study saying that, using the iterative decoding algorithm, an increase of *N1* does not mean an improvement in the inefficiency ratio, so the optimal value is *N1*=3.

6. Conclusions and future work

LDPC codes allow to reduce considerably the number of cycles needed to reconstruct a file and, therefore, the download time. This reduction is bigger in channels with high losses.

On the other hand, the packet delivery scheduling is a parameter that affects the efficiency of the content push download service. In environments with low losses, a random delivery model is more efficient than the sequential one, since it is more immune to the burst packet losses.

LDPC is more efficient with large files and the transmission using only one block is more efficient. Regarding the two LDPC structures, Staircase and Triangle, the first is more efficient with code rates lower than 0.4 and when short files are sent. In the experiments made with a mobile device in a Wi-Fi network, although the results of inefficiency ratio are worse, the conclusions that we have reached are the same.

The optimal coding parameters in each case (code rate, number of blocks…) depend on the transmission characteristics: channel losses, files sent or processing capacities of the receivers.

In this sense, one of the future lines is the study of the memory required by the devices to carry out the decoding process. One of the parameters that affects the memory consumption is the decoding algorithm. The use of other algorithms, such as the Gaussian elimination scheme, improves the inefficiency ratio but increases the required memory by the terminal. Regarding this, there are different methods for reducing the decoding complexity, as [28] proposes.

Another research line is the application of LDPC codes to the video transmission, making use of the developed library, by means of LCT building block of FLUTE protocol.

Acknowledgements

This work was supported in part by the Ministry of Industry, Tourism and Trade of the Government of Spain, under project “Redes Híbridas para la Provisión de Servicios Turísticos” (TSI-020302-2010-165).

References

1. Faria G, Henriksson J, Stare E, Talmola P (2006) DVB-H: Digital Broadcast Services to Handheld Devices. Proc. of the IEEE, vol. 94, no. 1, pp. 194-209.
2. 3GPP TS 22.146 (2006) Multimedia Broadcast/Multicast Service; Stage 1 (Release 6), V6.7.0.
3. White Paper (2009) Integrated Mobile Broadcast (IMB): The Power of Predictive Broadcasting for 3G Multimedia Applications.
4. 3GPP TS 25.346 (2007) Introduction of the Multimedia Broadcast Multicast Service (MBMS) in the Radio Access Network (RAN); Stage 2 (Release8), V8.0.0.
5. IEEE (2007) Std. 802.11, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.
6. IEEE (2009) Std. 802.16, Air Interface for Broadband Wireless Systems.
7. Fraile F, de Fez I, Guerri JC (2011) Evaluation of a background push download service for personal multimedia devices. IEEE International Conference on Consumer Electronics, Las Vegas, USA.
8. Paila T, Luby M, Lehtonen R, Roca V, Walsh R (2004) FLUTE – File Delivery Over Unidirectional Transport. IETF RFC 3926.
9. Gil A, Fraile F, Ramos M, de Fez I, Guerri JC (2010) Personalized Multimedia Touristic Services for mobile Hybrid Broadband/Broadcast. IEEE Transactions on Consumer Electronics, Vol. 56, No. 1, pp. 211-129.
10. Handley M, Jacobson V (1998) SDP: Session Description Protocol. IEFT RFC 2327.
11. Luby M, Watson M, Vicisano L (2010) Asynchronous Layered Coding (ALC) Protocol Instantiation. IEFT RFC 5775.
12. Luby M, Watson M, Vicisano L (2009) Layered Coding Transport (LCT) Building Block. IEFT RFC 5651.
13. Watson M, Luby M, Vicisano L (2007) Forward Error Correction (FEC) Building Block. IETF RFC 5052.
14. Watson M (2009) Basic Forward Error Correction (FEC) Schemes. IETF RFC 5445.
15. Shokrollahi A (2006) Raptor Codes. IEEE Transactions on Information Theory no. 6.
16. Luby M (2002), LT Codes. Proc. IEEE Symposium on Foundations of Computer Science (FOCS), Vancouver, Canada.
17. Luby M, Shokrollahi A, Watson M and Stockhammer T (2007) Raptor Forward Error Correction Scheme for Object Delivery. IETF RFC 5053.
18. Lacan J, Roca V, Peltotalo J, Peltotalo S (2009) Reed-Solomon Forward Error Correction (FEC) Schemes. IETF RFC 5510.
19. Gallager R G (1962), Low Density Parity Check Codes. IEEE Transactions on Information Theory, 8(1).

20. MacKay D, Neal R (1995) Good codes based on very sparse matrices. In 5th IAM Conference: Cryptography and Coding, LNCS No. 1025.
21. Roca V, Neumann C, Furodet D (2008) Low Density Parity Check (LDPC) Staircase and Triangle Forward Error Correction (FEC) Schemes. IETF RFC 5170.
22. Park S, Miller K (1990) Random Number Generators: Good Ones are Hard to Find. Communications of the ACM, Vol. 33, No. 1, pp. 87-88.
23. INRIA Planète Research Team (2006) LDPC large block FEC codec distribution, http://planete-bcast.inrialpes.fr/article.php3?id_article=16
24. Bai H and Atiquzzaman M (2003), Error modeling schemes for fading channels in wireless communications: a survey. IEEE Communications Surveys and Tutorials, 5(2).
25. Cunche M, Roca V (2008) Optimizing the Error Recovery Capabilities of LDPC-Staircase Codes Featuring a Gaussian Elimination Decoding Scheme. Proc. of the 10th IEEE International Workshop on Signal Processing for Space Communications (SPSC), Rhodes Island, Greece.
26. Roca V, Neumann C (2004) 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.
27. Cunche M, Roca V (2008) Improving the Decoding of LDPC Codes for the Packet Erasure Channel with a Hybrid Zyablov Iterative Decoding/Gaussian Elimination Scheme. INRIA Research Report RR-6473.
28. Cunche M, Savin V, Roca V (2010) Analysis of Quasi-Cyclic LDPC codes under ML decoding over the erasure channel. IEEE International Symposium on Information Theory and its Applications (ISITA), Taichung, Taiwan.

***Author Biographies**

Click here to download Author Biographies: Biographies.doc

Ismael de Fez was born in Valencia, Spain in 1983. He received his Telecommunications Engineering degree and the M.S. in Telematics from the Universidad Politécnica de Valencia (UPV), Spain, in 2007 and 2010 respectively. His M.Sc. thesis was awarded by the Telefónica chair of UPV. Currently he works as a researcher at Multimedia Communications research group (COMM) of the Institute of Telecommunications and Multimedia Applications (iTEAM). His areas of interest are file transmission over unidirectional environments and file encoding.

Francisco Fraile was born in Murcia, Spain in 1979. He obtained a degree in Telecommunication Engineering from the Universidad Politécnica de Valencia (UPV) and a M. Sc. Degree in Microwave Engineering from the University of Gävle in 2004. Since then he works as a research engineer for the Swedish company Interactive TV Arena. In 2006, Francisco joined the Multimedia Communications research group (COMM) of the Institute of Telecommunications and Multimedia Applications at UPV, 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 in 1979. He received his Computer Science degree from the Universidad Politécnica de Valencia (UPV), Spain in 2004 and his is currently studying his M.S. in Telematics. Currently he works as a researcher at Multimedia Communications research group (COMM) of the Institute of Telecommunications and Multimedia Applications (iTEAM). His areas of interest are mobile applications and multimedia transmission protocols.

Juan Carlos Guerri was born in Valencia, Spain in 1969. He received his M.S. and Ph. D. (Dr. Ing.) degrees, both in Telecommunication Engineering, from the Polytechnic University of Valencia, in 1993 and 1997, respectively. He is a professor in the E.T.S. Telecommunications Engineering at the Polytechnic University of Valencia, and he leads Multimedia Communications research group (COMM) of the iTEAM (Institute of Telecommunications and Multimedia Applications). He is currently in research and development projects about multimedia QoS/QoE, wireless ad hoc networks and video performance evaluation.

***Author Photographs**