An adaptive mechanism for optimal content download in wireless networks

Ismael de Fez and Juan Carlos Guerri

***Abstract*—This paper presents an adaptive mechanism for improving the content download in wireless environments. The solution is based on the use of the file delivery over unidirectional transport (FLUTE) protocol in multicast networks, which reduce considerably the bandwidth when there are many users interested in the same contents. Specifically, the system proposed reduces the average download time of clients within the coverage area, thus improving the Quality of Experience. To that extent, clients send periodically feedback messages to the server reporting the losses they are experiencing. With this information, the server decides which is the optimum application layer – forward error correction (AL-FEC) code rate that minimizes the average download time, taking into account the channel bandwidth, and starts sending data with that code rate. The system proposed is evaluated in various scenarios, considering different distributions of losses in the coverage area. Results show that the adaptive solution proposed is very suitable in wireless networks with limited bandwidth.**

***Index Terms*—Adaptive codes, application layer – forward error correction (AL-FEC), file delivery over unidirectional transport (FLUTE), low density parity check (LDPC), multicast wireless networks.**

***EDICS*— 8-WMMM, 8-ERCO**

I. Introduction

Wireless networks are part of our daily lives for many years. Since the first half of the last century, people listened to the news or music through the waves that arrived to their radios. Some decades later, the use of the television became popular in all households. Today, former analogical broadcasting systems have been replaced by modern digital ones, such as DVB-T (Digital Video Broadcasting – Terrestrial) [1] or ATSC (Advanced Television System Committee) [2].

In contrast to radio and television, other technologies were intended for wired scenarios and have evolved to wireless systems. There are two representative examples: telephone and Internet. Nowadays, approximately three quarters of the world population has access to a mobile phone [3]. This percentage exceeds the 90% in developed countries. In the last years, the appearance of the smartphones has revolutionized the industry. Through these intelligent devices, users connect to the Internet using new technologies such as Wi-Fi, HSPA (High-Speed Packet Access), WiMAX (Worlwide Interoperability for Microwave Access) or LTE (Long Term Evolution).

In this regard, the Internet consumption through wireless networks has exploded in the last years due to the fact that not only the number of users has increased but also the amount of data consumed. Nowadays, a large number of the contents consumed by the users require a high bandwidth, for instance the visualization of multimedia streaming through video portals.

In this framework, the exponential growth of the traffic in wireless networks is becoming a problem. Thus, as users require more bandwidth and the frequency spectrum is a limited resource, it is necessary to find several mechanisms that allow users to consume contents without getting worse their Quality of Experience.

A good mechanism for reducing the bandwidth is the use of multicast networks. Through these networks it is possible to send multimedia content to different users within a single connection [4]. Regarding data communications, multicast file transmission is very useful in crowded environments where users have similar interests, for instance in malls, popular festivals or sport events. As a possible use case, in a basketball match users could receive on their mobile devices different information such as the retransmission of the best moments of the match or information related to the shops within the court (e.g., publicity or discount vouchers). Another good example is multicast streaming based on file multicasting. In this field, there are different related works in the literature that propose the use of file delivery to provide multicast streaming. For instance, [5] proposes the use of the FLUTE file delivery protocol to send DASH (Dynamic Adaptive Streaming over HTTP) segments over MBMS (Multimedia Broadcast /Multicast Service) [6] and eMBMS (Evolved MBMS) [7]. DASH [8] is a novel ISO standard for the transmission of on-demand and live streaming.

In this sense, FLUTE [9] is an IP multicast protocol widely used for multicast file download services. In fact, FLUTE has been established as the multicast file delivery protocol for different standards, such as DVB-H (DVB – Handheld) [10], DVB-IPTV (DVB – Internet Protocol TV) [11], or the aforementioned MBMS and eMBMS. FLUTE is based on the

Manuscript submitted January 3, 2013. This work is supported in part by the Ministerio de Economía y Competitividad of the Government of Spain under project COMINN (IPT-2012-0883-430000).

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

use of a File Delivery Table (FDT), which is the in-band mechanism used by FLUTE to inform clients about the files (and their characteristics) transmitted within a FLUTE session [12]. The main characteristic of FLUTE is that it provides reliability to the transmission.

In that regard, due to the transmission losses, typical in wireless environments, multicast networks should be protected against errors. To that end, there are different error control mechanisms for multicast transmissions [13], in wired and wireless networks [14]. One of the most used is Forward Error Correction (FEC), which allows to recover packets lost by adding redundancy to the transmission. Specifically, using FEC at the application layer (AL-FEC) avoids further investments in infrastructure [15]. Precisely, AL-FEC is one of the protection mechanisms used by FLUTE.

In AL-FEC, the amount of protection provided to the transmission is defined by the code rate. As shown in a previous study carried out by the authors [16], there is an optimum code rate that minimizes the download time of a certain file by a client. This download time depends, among other parameters, on the channel losses perceived by each client. Therefore, it is possible to transmit a file at an optimum code rate for each client in order to minimize the download time. To that extent, clients must be able to inform the server about the losses they are experiencing.

Nevertheless, in environments with limited bandwidth it may not be possible to send a file with different code rates, since the bandwidth would increase considerably. Thus, it would be very useful to send data at an optimum code rate that benefits the major part of the users. Since the losses perceived by the users could change quickly, this code rate should be chosen dynamically.

In this sense, the present work presents an adaptive system where the server sends data at an optimum code rate for each time interval, according to the losses detected by the clients within the coverage area.

Specifically, the objective of this paper is to reduce the average download time that clients need when downloading contents. It should be noted that, one of the main goals of any service is to provide a good Quality of Experience (QoE) to the users. In streaming services a good QoE is provided when users receive the video without interruptions, with high quality, and with the minimum waiting time. With regards to file transmission to multiple receivers, users have a good QoE when they receive files correctly and the download time is minimal. In this sense, the download time is a well-known QoE metrics for evaluating file multicast download. For instance, it is used in IP Datacast for DVB-H [10] for evaluating the effect of Raptor AL-FEC codes for a FLUTE multicast session.

The rest of the paper is structured as follows. Next section provides an overview of the adaptive system proposed. Section III explains the evaluation methodology used to obtain the results presented in Section IV, where the adaptive system is analyzed and evaluated. Finally, the last section of the document includes some final conclusions.

II. System overview

This paper presents an alternative to the study presented by the authors in [16], taking into account the channel bandwidth. In the adaptive system presented in [16] a server sends multimedia content in a multicast network to several users, which inform the server about the losses they are experiencing through the delivery of feedback messages. Using this information, the server makes forward error correction parity symbols available to the clients at an optimum code rate. All parity symbols belonging to a code rate are inserted on a separate transmission channel. Thus, clients join to the channel that minimizes their losses. 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. Nevertheless, that scenario does not take into account the channel bandwidth limitations. Therefore, in scenarios where the bandwidth is limited, it is necessary to find another solution that benefits all users. Unlike [16], in the scenario hereby proposed, as only one code rate will be used in a given time, only one transmission channel will be needed. Thus, the file delivery session only contains one channel, in which both source and parity symbols are sent, which represents an easy solution for the server and the clients. In contrast to [16], the main objective of this paper is to optimize the bandwidth usage, but without getting worse the download time of clients. In this sense, this paper presents new proposals, algorithms, evaluation scenarios, results and improvements compared with [16].

It should be mentioned that it is used a random transmission model, where source and parity packets are sent in a fully random order, since this model is more efficient than the sequential model, as it is proved in [17] and [18].

Furthermore, this paper considers that the losses perceived by the clients can change over time. We propose an adaptive mechanism to assign an optimum code rate based on that used by RTP/RTCP (Real-time Transport Protocol / RTP Control Protocol) for dynamic adjustment of the bandwidth requirements of multimedia applications [19]. As in [19], our algorithm increases, holds or decreases a certain parameter (in this case the code rate, instead of the bandwidth) according to the feedback received by the clients.

Fig. 1 shows an overview of the system proposed. There is a certain number of clients within the coverage area in a multicast wireless network. Clients perceive different losses depending on how far they are from the server (the further the more losses). Also, clients are continuously moving so the channel losses they perceive are changing. Initially, the server sends data with a certain code rate. After a while clients start sending feedback messages informing the server about the losses they are perceiving. In this paper we consider that the feedback messages always arrive to the server. The way clients send these losses reports is not analyzed in this paper. Different mechanisms to provide this feedback are: [20] and [21], based on RTCP; and the reporting mechanisms used by DVB-H [10] and MBMS [6], which support FLUTE, the protocol used in this proposal. Once the losses reports are

received by the server, it decides which is the optimum code rate that minimizes the average download time of all clients. Then, the server starts sending data at this optimum code rate. This process is carried out periodically: the clients are repeatedly sending feedback messages and the server is analyzing them at a certain time intervals.

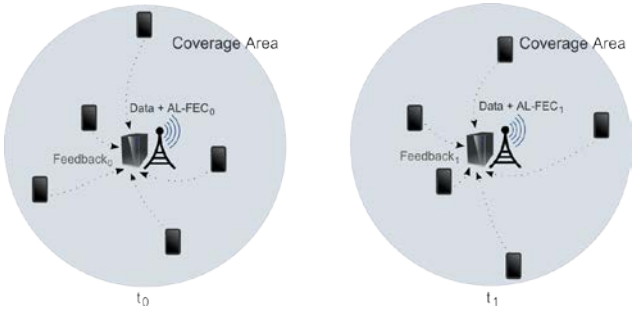


Fig. 1. System overview.

When choosing the code rate that best suits all clients it must be taken into account that using insufficient protection for clients with high losses has more impact on the download time than using an excessive protection for clients with low losses. Therefore, feedback messages of clients with high losses will have more weight when the optimum code rate is chosen. To that extent, in this proposal we consider that the server classifies the losses perceived by each user into three different regions: low losses, medium losses and high losses region. Hence, each one of the *n* clients is classified in each instant of time in a certain region according to their loss rate, as Fig. 2(a) depicts. The server calculates how many clients (n_L) are in the low losses region (clients who have less than λ_L losses), how many clients (n_M) are in the medium losses region (those who have between λ_L and λ_H losses) and how many clients (n_H) have high losses (those who have more than λ_H losses). Then, the code rate is chosen according to the percentage of clients in each region.

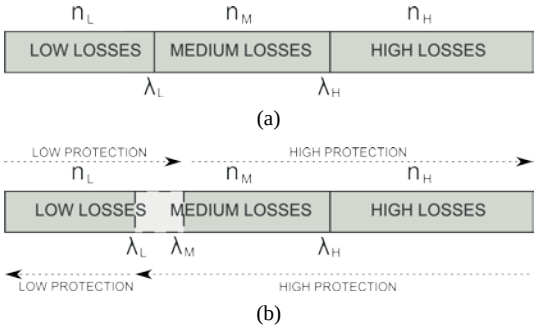


Fig. 2. Losses region classification. (a) Without hysteresis. (b) With hysteresis.

It is worth noting that, according to [16], a given AL-FEC code rate performs well in a wide interval of packet loss rates around the value for which it provides a minimum download time. Based on this conclusion, a priori, it could be considered the use of only three different code rates (as many as losses regions). Nevertheless, at the time of choosing the code rates for each region it must be taken into account the bandwidth increase associated to each code rate. Using a high protection (low code rates) increases considerably the bandwidth, therefore there is a trade-off between the bandwidth and the download time. According to the studies presented in Section IV.A, this paper only considers two different code rates: one code rate for low losses and another one for medium-high losses. Using two code rates instead three provides very good results regarding the download time, and improves considerably the channel bandwidth.

In this sense, the use of a high number of losses regions would increase the complexity of the system and could become the system inefficient, since there would be a lot of changes of code rates and clients would have to create continuously the decoding parity matrix associated to that code rate and discard continuously parity packets previously received. Additional studies carried out by the authors and not included in this paper prove that considering six losses regions and therefore using six different code rates (instead two) slightly reduces the download time (less than a 5% in the scenarios considered in this paper, which are shown in Section IV.A) at the expense of increasing considerably the number of changes of code rate and the bandwidth (more than four times of overhead). Specifically, in scenarios 1 and 2, there is a download time reduction over a 2% considering six different code rates, but the bandwidth overhead increases from a 20% until a 93%. In scenario 3 the bandwidth overhead considering six losses regions is a 133% but the average download time is only 5% lower.

Hence, in our proposal there will be two protection states: a state of low protection (which will use a high code rate) and a state of high protection (which will use a medium code rate). Furthermore, the encoding (and decoding) process is easier as less different code rates are used. The server calculates the optimum code rate in each instant of time according to the following algorithm, based on the one shown in [19] for bitrate (as aforementioned):

Algorithm 1

1: **if** $n_H/n \geq N_h$ then new_state=HIGH_PROTECTION
2: **else if** $(n_M+n_H)/n \geq N_m$ then new_state=HIGH_PROTECTION
3: **else** then new_state=LOW_PROTECTION

N_h and N_m are the thresholds for clients with high and medium losses, respectively. The use of these thresholds allows to give more priority to clients with higher losses. The design of an efficient adaptive system depends greatly on the values of N_h and N_m. Other key parameters are λ_L and λ_H. In this sense, since it would not be very efficient to change the code rate too frequently when losses do not vary excessively, the system performance can be improved by using hysteresis [22]. Fig. 2(b) proposes the use of three thresholds: λ_L, λ_M, and λ_H. Therefore, when the server calculates the number of clients in each region, it takes into account the current protection state. Thus, if the server changes their state from the low protection state to the high protection state, a client in the low losses region will change to the medium losses region when their losses are higher than λ_M, whereas it will come back to the low losses region when their losses are lower than λ_L, as Fig. 2(b) depicts.

On the other hand, in order to avoid an erroneous estimation

of the losses, the server can smooth, for each feedback message received, the instantaneous losses (L_{inst}) with the previous average losses (L_{avg}), using a low-pass filter, calculating the new loss rate (L) as: $L=(1-\alpha)\cdot L_{avg}+\alpha\cdot L_{inst}$, where α is the influence factor of the new value, ranged between 0 and 1.

III. Evaluation Methodology

A. Calculation of the download time

This section presents a methodology to calculate analytically the download time. As mentioned, the objective of this proposal is to reduce the average download time when clients download contents. The download time of a certain file L is defined as the time passed since the transmission starts until the file is completely downloaded. This occurs when clients have received enough packets to rebuild the file.

In this study we suppose that the server is sending files during a certain time, then it receives feedback messages from clients and then it continues sending files with the new code rate. In this theoretical study for simplicity we consider that, in each instant of time the server sends all files available in their repository. That is, this involves considering the use of file carousels to send content. The use of carousels as a delivery mechanism is employed by some standards that use FLUTE as delivery protocol, such as [10]. In this way, we will consider that each cycle of the carousel corresponds to an instant of time.

Thus, as a first approximation, we are going to study the case where the server sends only one file. If a client is not able to download it during a certain instant of time, the client will need T instants of time to download the file. In each instant of time the server will send contents with a certain code rate, so the duration of each instant of time i ($t_T(i)$) could be different. Therefore the download time (t_D) is calculated as:

$$t_D = t_S \cdot \beta + \sum_{i=1}^{T-1} t_T(i), \qquad (1)$$

where t_S is the time needed to send all packets (source plus parity packets) that compose the file in the cycle that completes the download. Due to the use of AL-FEC encoding, clients can download a file before the last packets have been received so, actually clients could need a time lower than t_S to complete their downloads. To consider this, we define a factor $0<\beta\leq 1$. Remember that, in reception clients need to receive an amount of packets equal to the product of the number of source packets of the file to download by a factor called inefficiency ratio, which depends on the coding algorithm. The value of the inefficiency ratio is equal to 1 in codes that belong to the Maximum Distance Separable (MDS) category, and in the rest of codes this value is greater than 1.

Developing expression (1):

$$t_D = \frac{ceil(\frac{S_L}{CR_T})}{b}\cdot\beta + \sum_{i=1}^{T-1}\frac{ceil(\frac{S_L}{CR_i})}{b}, \qquad (2)$$

where S_L is the size of the file L to download, CR_i is the code rate of the instant of time i, and b is the transmission rate. Moreover, in order to calculate the number of packets that compose a file after decoding (and thus the transmission size) it is necessary to ceil the division between the number of packets that form a file by the code rate.

In the case of sending several files within the carousel, the calculation of the download time is slightly different. Fig. 3 shows an example of a transmission using file carousels. In the example we suppose that a client is connected to the channel at the start of a certain instant of time and that they want to download the file F3.

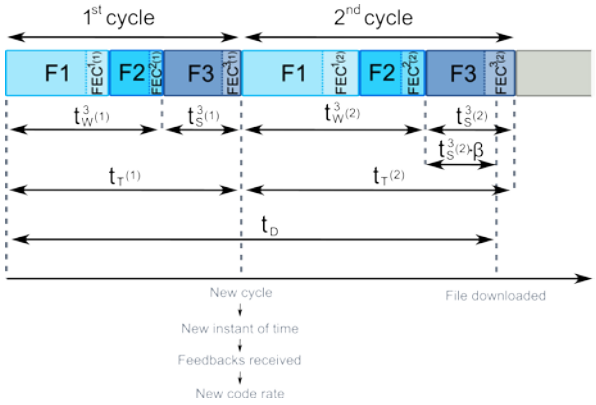


Fig. 3. Example of a carousel transmission.

First, if the client only needs one carousel cycle to download the file, the client will have to wait a time t_W to start receiving packets of the file L to download. After a time $t_S\cdot\beta$, the client will have downloaded the file. If the client needs more than one cycle to complete the download, it will be necessary to consider the transmission time of the entire carousel (t_T).

In the example, each instant of time or cycle implies a different code rate so, as the protection in the second cycle is higher than in the first, the transmission size of every file in the second cycle is higher than in the first. Therefore, the transmission size of the entire carousel is bigger in the second cycle and thus the duration of that instant of time ($t_T(2)>t_T(1)$). The same applies to the value of t_W and t_S.

In this way, in the general case, clients will have to wait *T-1* entire cycles, plus a time t_W and t_S from the cycle that completes the download. So, the download time of a file L is calculated as:

$$t_D^L = t_W^L(T) + t_S^L(T)\cdot\beta + \sum_{i=1}^{T-1} t_T(i). \qquad (3)$$

Analyzing each term, the value of t_W depends on the carousel size and the transmission schedule. Considering that the probability of downloading a certain file is equal to the probability of downloading another file in the carousel, the waiting time can be calculated using the following approximation:

$$t_W^L(T) \approx \frac{t_T(T)}{2} - \frac{t_S^L(T)}{2}. \qquad (4)$$

The calculation of t_S is similar to the equation (2). In order to simplify the final expression, we do not consider the effect of the ceiling round:

$$t_S^L(T) \cong \frac{S_L}{b \cdot CR_T}. \quad (5)$$

Regarding t_T, this term is calculated as:

$$t_T(i) \cong \frac{\sum_{j=1}^{N} S_j}{b \cdot CR_i}, \quad (6)$$

where N is the number of files in the carousel and S_j is the size of the file j.

In this way, the download time is calculated as:

$$t_D^L \approx \frac{\sum_{j=1}^{N} S_j}{2b \cdot CR_T} - \frac{S_L}{2b \cdot CR_T} + \frac{S_L}{b \cdot CR_T} \cdot \beta + \sum_{i=1}^{T-1} \frac{\sum_{j=1}^{N} S_j}{b \cdot CR_i}. \quad (7)$$

Simplifying that expression:

$$t_D^L \approx \frac{S_L}{b \cdot CR_T} \cdot \left[\beta - \frac{1}{2} \right] - \frac{\sum_{j=1}^{N} S_j}{2b \cdot CR_T} + \sum_{i=1}^{T} \frac{\sum_{j=1}^{N} S_j}{b \cdot CR_i}. \quad (8)$$

Analyzing the previous formula, values of S_j, N and b depend on each particular implementation. On the other hand, the value of the code rate for each instant of time (CR_i) is calculated using the Algorithm 1, which will be analyzed in Section IV. The parameter β depends on the remaining packets needed to complete the download. This parameter is directly related to T, which is the number of instants of time (or number of cycles) that a user needs in order to download a certain file.

In order to calculate T, in each instant of time the client will receive a certain number of packets of the file to download. This number will depend on the losses of the channel. Thus, the probability of receive x new packets at any given loop can be modeled by using a hypergeometric distribution:

$$\mathrm{P}(x,m,r,l) = \frac{\binom{m-r}{x}\binom{r}{(m-l)-x}}{\binom{m}{m-l}}, \quad (9)$$

where m 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. The latter probability yields the following expression for the expectation value of the number of packets correctly received at loop i:

$$x(i) = \sum_{\xi=0}^{m-r} \xi \cdot \mathrm{P}(\xi,m,r,l). \quad (10)$$

Then, an estimation of the value of T is provided by the following expression:

$$T = \min\{h : \sum_{i=1}^{h} x(i) \geq k * inef\_ratio\}, \quad (11)$$

where k is the number of source symbols that compose a file. In this way, the value of T is calculated iteratively, checking for each cycle if the client has received enough packets to rebuild the file. A methodology used to calculate T and β can be found in [16][23].

B. Evaluation parameters

As Section II has shown, there are several parameters to configure when evaluating the system proposed. Firstly, it will be analyzed the download time of a file in a channel with different losses using different code rates. In the studies hereby presented, in order to calculate which is the code rate that minimizes the download time for each percentage of losses two different methodologies will be used: first, the download time will be calculated using the analytical model explained in the previous subsection; second, we will calculate the download time by carrying out measurements in a real environment. To do this, an implementation developed by the authors [18] of a file server and client based on FLUTE is used. It should be highlighted that the major part of the results presented in this paper are obtained through this implementation. The file server/client also includes a module that implements LDPC (Low Density Parity Check) codes [24] in order to provide reliability to the transmission. In this study, LDPC Staircase codes [25] will be used to evaluate the optimum code rate. LDPC codes have been proved to be very efficient, with a performance very close to Raptor and ideal maximum distance separable codes [16][26]. This good performance and their lower complexity make these codes very recommendable in multicast file delivery.

It should be noted that both in the analytical and in the experimental results, several measurements are made. In the experimental ones, a server sends a file in a multicast channel and a client downloads it. Apart from getting the best code rate for each losses region, the previous study also will calculate the suitable values for λ_L, λ_M, and λ_H.

In the scenario proposed, there are many clients within the coverage area that are continuously moving (sometimes they are getting closer to the server and sometimes are moving away), so the losses they perceive are continuously changing. As mentioned, the losses they perceive are directly related to the distance to the server. In [27] it is shown the relation between the distance to the server and the PRR (Packet Reception Ratio) in a particular wireless network. Based on that study, Fig. 4 allows to match the distance from clients to the server with the percentage of losses that clients have. These results have been carried out through simulations using ns-3, considering a transmission rate of 5.5 Mb/s and an output power of 17 dBm. This transmission rate is supported, among other standards, by 802.11b [28] (and subsequent versions), one of the Wi-Fi reference standards. Assuming that, in practice, the effective transmission rate is lower than the one specified in the standard (due to overheads or routers features), in the rest of studies an effective transmission rate of 5 Mb/s is used, the same considered in [16]. All these studies only consider clients within the area lower than 110 meters, since distances higher than 110 meters provide unreasonable percentage of losses (higher than 60%), so it is considered that these clients are out of the coverage area.

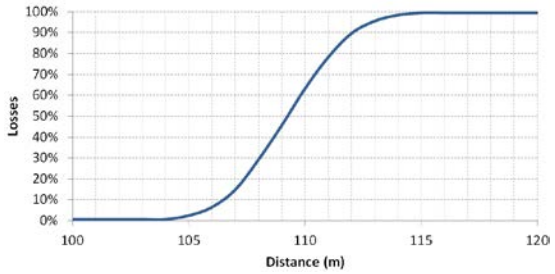


Fig. 4. Losses perceived depending on the distance to the server for a transmission rate of 5.5 Mb/s.

The percentage of losses according to the distance to the server depends on the transmission rate [29] as well as the transmission power. Thus, if the transmission rate increases (or the output power decreases), the coverage area decreases. Taking this into consideration, we will suppose that clients send their feedback messages using a lower transmission rate (for example, 1 Mb/s) with an appropriate output power through a channel that guarantees that the packets arrive to the server.

In order to analyze the behavior of the adaptive system proposed, the losses perceived by the clients change in every instant of time. Thus, there will be some moments when the average losses of all clients will increase whereas in other moments will decrease. The initial position of clients (and thus, losses) is generated randomly. Later positions of clients are calculated by using a simplified version of the Gauss-Markov mobility model [30]. This model, widely used in the literature, takes into account the previous position and speed of clients to estimate the future position of clients. In our case, the future position of clients is obtained by choosing a random value in a normal (Gaussian) distribution, which mean is the previous position.

Different losses scenarios are presented, as Section IV.B explains. Those scenarios have a losses distribution completely different. Also, in all scenarios, clients will be moving (and thus changing their losses). The system behavior will be evaluated, in most cases, during ten time intervals.

Then, parameters N_h and N_m will be analyzed, as well as the smooth factor α and the hysteresis effect. Once all the configuration parameters are chosen, it is possible to calculate the download time for each client in each instant of time. This download time is calculated using the results obtained by means of the FLUTE server/client and by the analytical results. In order to do a more accurate measurement, the download time of a certain client in the instant *i* is calculated as the average of the download time between the instant *i-1* and the instant *i*, using the code rate obtained in the instant *i-1*. In this paper, ten intermediate values in a time interval have been used to calculate the average download time. Also, for simplicity we have not considered co-channel interferences from other users.

IV. Results and Analysis

A. Analysis of the code rate

First, the optimum code rate for each percentage of losses is analyzed. To that extent, we calculate the download time of a file of approximately 4 MB (3000 FLUTE packets with length 1428 bytes), which is sent in a multicast channel using a transmission rate of 5 Mb/s. The download time is calculated by measuring the time passed since a client starts downloading a file until the file is completely downloaded. Note that, apart from the losses, the download time depends greatly on the values of the file size and the transmission rate. However, as both parameters have a linear behavior regarding the download time [16], the conclusions arisen in this study remain valid independently of the value of the file size and the transmission rate.

Table I shows the download time (in milliseconds) for different code rates and percentage of losses for the experimental measurements. The table reflects that there is a code rate (highlighted in italics) that minimizes the download time depending on the losses. Based on the results of Table I, Table II shows the average download time for the three losses regions: low, medium and high. We have defined the low losses region as the area where losses are between 0 and 10%; the medium losses region corresponds with the area where losses are between 10 and 30%; and the high losses region is the area where losses are higher than 30%. Table II also shows the bandwidth increase for each code rate. It is worth recalling that a low value of the code rate increases considerably the bandwidth.

TABLE I
DOWNLOAD TIME (IN MILLISECONDS) FOR DIFFERENT CODE RATES

CR/ Loss	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
0%	8926	7826	7556	7946	8514	7557	6751	*6225*
5%	9388	8228	7927	8362	8723	7572	*6752*	18555
10%	9776	8704	8408	9349	8714	*7582*	9318	25622
15%	10600	9102	8923	9652	8595	*7625*	10847	29802
20%	11174	9596	9631	10034	*8739*	10129	12463	32598
25%	11396	10296	10467	*10045*	10196	11640	13399	37966
30%	12534	11033	11557	*11461*	11509	12818	15321	45880
40%	14560	13625	*12362*	13679	14463	15724	20252	60090
50%	16859	*15310*	15876	17502	18026	20656	25208	74105
60%	22182	*20225*	20880	22364	24403	27858	34538	101534

TABLE II
AVERAGE DOWNLOAD TIME (IN MILLISECONDS) FOR REGION AND BANDWIDTH INCREASE

CR/ Loss	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Low	9363	8253	7964	8552	8650	*7570*	*7607*	16801
Med.	11096	9746	9797	10108	*9551*	9959	12270	34374
High	16534	*15048*	15169	16252	17100	19264	23830	70402
ΔB	233%	150%	100%	67%	43%	25%	11%	0%

As expected, if no coding is used (that is, the code rate is equal to 1), the download time increases drastically when losses increase. When losses are low, code rates 0.8 and 0.9 provide the minimum download time. Calculating the average download time for low losses (Table II), both 0.8 and 0.9 provide a similar value (only a difference of 0.5%). Therefore, as the bandwidth increase for code rate 0.9 is lower, this code rate will be chosen for the low protection state.

Regarding the medium losses region, the code rate with lowest average download time is 0.7. Finally, with very high losses, the code rate that minimizes the download time is 0.4.

Nevertheless, since in the scenario proposed in this paper

the bandwidth is a limited resource (and that is why only one transmission channel is used), it is not acceptable to use a code rate that increases excessively the bandwidth. In this sense, the code rate 0.4 provides the best results for high losses at the expense of increasing the channel bandwidth a 150%. Comparing the results obtained for high losses with code rates of 0.4 and 0.7, the download time with a code rate of 0.7 is only 15% higher than the one obtained with a code rate of 0.4 when losses are 50%, whereas when losses are 30%, the download time of 0.7 is barely 4% higher than the one obtained with a code rate of 0.4. The difference regarding the bandwidth increase is clear: 43% with a code rate of 0.7 against 150% with 0.4. Therefore, the code rate of 0.7 will be chosen when losses are both medium and high, as mentioned in Section II.

Summarizing, in the low protection state, the server will send data using a code rate of 0.9, whereas in the high protection state, the server will use a code rate of 0.7.

In order to calculate the threshold values of the low and medium regions shown in Fig. 2(b), that is, λ_L and λ_M, it is needed to compare the behavior of the code rates (CR) 0.7 and 0.9. In this sense, Fig. 5 depicts the download time of both code rates for different percentage of losses.

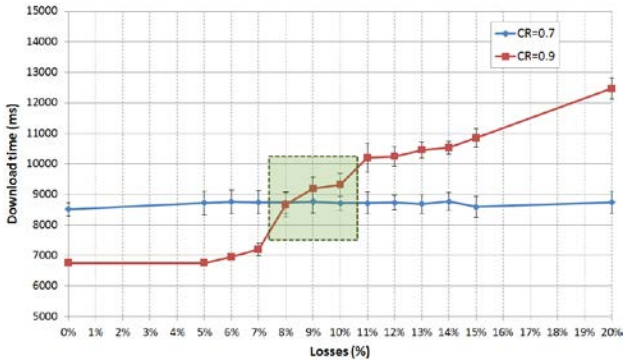


Fig. 5. Download time comparison between CR=0.7 and CR=0.9.

As Fig. 5 shows, when losses are equal or lower than 8%, the code rate 0.9 provides lower download times, whereas from 9% of losses the code rate of 0.7 is more suitable. Also, there is an area where the difference regarding the download time among the two code rates is very low, as Fig. 5 stresses. This area delimits the hysteresis zone. Initially, in order to provide a reasonable (not very tight) value of hysteresis, λ_L and λ_M will differ a 5%. Therefore, according to Fig. 5 and Table I, the values of the thresholds will be: λ_L=7%, λ_M=12% and λ_H=30%. These values will be analyzed in Section IV.E.

B. Losses model

The studies here presented consider that there are n=100 clients in the coverage area. In order to analyze the behavior of the system proposed, five different scenarios with different distributions of losses are defined. In all scenarios, clients are continuously moving within the coverage area, with the aim of analyzing how the system works when losses change.

The distribution of the instantaneous losses of all clients for each instant of time of the first scenario is represented in Fig. 6. This scenario considers 10 instants of time. The figure shows the percentage of clients per losses region as well as the hysteresis region. Clients in the hysteresis zone will be in the low or in the medium losses region depending on the state protection. Moreover, the figure also shows the average losses perceived by all clients for each instant of time. On the other hand, Fig. 7 depicts, for the same losses distribution, the distance from clients to the server for three certain instants of time (when losses are medium, high and low respectively), and it can be clearly seen how clients are distributed along time. That figure also shows three circles that represent the losses thresholds λ_L, λ_M and λ_H. As mentioned, the amount of clients in each one of these circles will determine the state of the system and therefore the code rate.

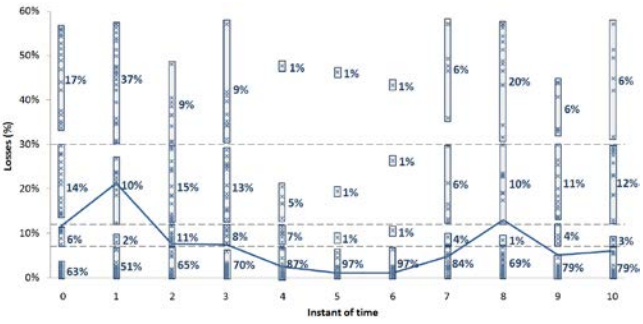


Fig. 6. Losses distribution for scenario 1.

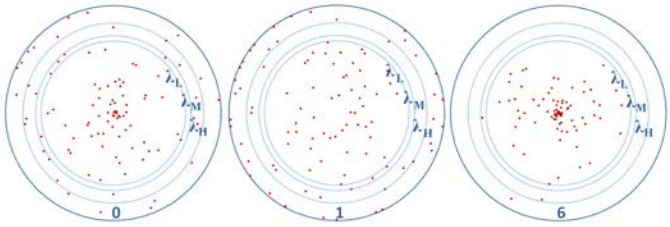


Fig. 7. Distance to the server in scenario 1 for instants of time 0, 1 and 6.

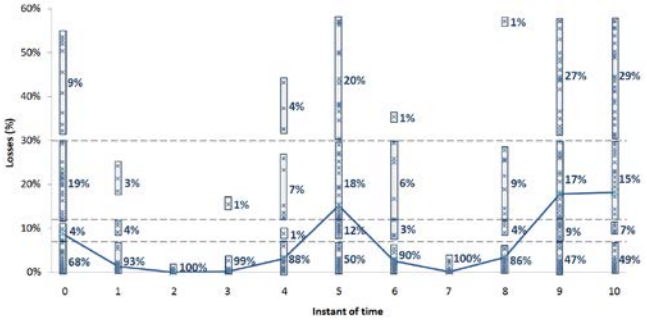


Fig. 8. Losses distribution for scenario 2.

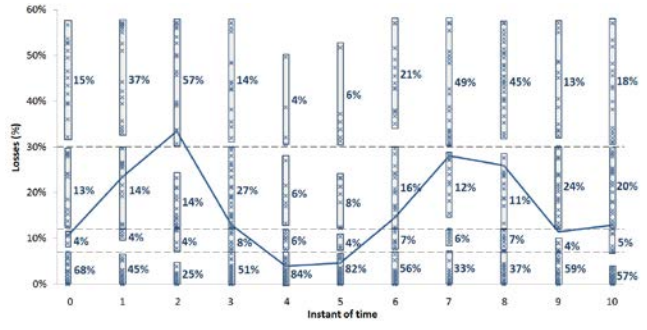


Fig. 9. Losses distribution for scenario 3.

In the second scenario, the distribution of losses is different, existing time intervals when clients get closer to the server and others when clients move further away from it, as Fig. 8 shows. In addition, in the third scenario (Fig. 9) the losses are, in general, rather higher.

On the other hand, the fourth scenario is rather different from the previous ones, since losses change more abruptly. Fig. 10 shows the distribution of losses for each instant of time as well as the average losses along the time.

Finally, the fifth scenario, shown in Fig. 11, will be used to evaluate the effect of the hysteresis. To that extent, the scenario considers many time intervals, specifically 100, instead of 10 used in the previous scenarios. Moreover, the average losses will be all around the hysteresis zone.

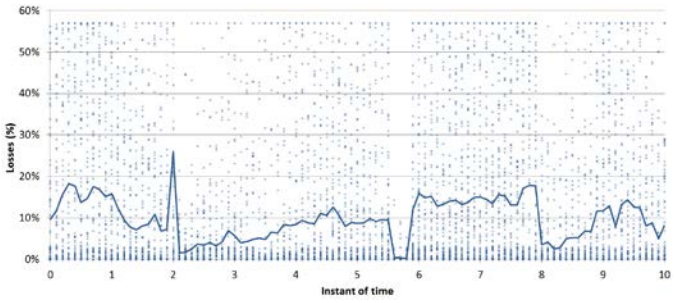


Fig. 10. Losses distribution for scenario 4.

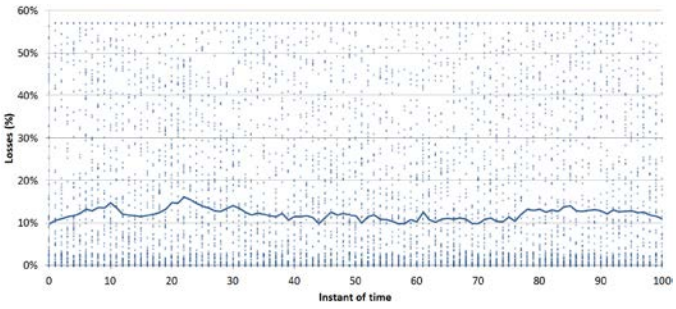


Fig. 11. Losses distribution for scenario 5.

C. Adaptive code rate

The value of parameters N_h and N_m has a great influence on the performance of the system proposed. As mentioned, it is necessary to give more priority to those clients who perceive high losses, therefore if a low percentage of clients has high losses the protection must be increased. In the following study, we will consider N_h=17% (1/6) and N_m=33% (1/3), and then we will analyze other values. Therefore, if more than a sixth of clients have high losses or more than a third of clients have medium or high losses, the server will be in the high protection state. Also, initially we will consider that α=1, so the server will not smooth the losses perceived by the clients. In the three scenarios used in this subsection this value seems appropriate since there are not excessively abrupt changes in losses.

As mentioned previously, this paper presents both analytical and experimental measurements. In the figures shown, the analytical results are presented as an upper error bar that represents the difference regarding the download time obtained in the experimental results.

The performance of the adaptive system in scenario 1 is shown in Fig. 12. The figure shows the average download time per instant of time for different code rates: a fixed code rate of 0.7, a fixed code rate of 0.9, the adaptive code rate and the ideal case. This ideal case considers that each client connects to a channel which transmits with the optimum code rate according to their losses. That is, this case corresponds with the proposal presented in [16] after all parity channels (with code rates 0.4, 0.5, 0.6, 0.7, 0.8 and 0.9) plus the base channel have been created. In this way, the ideal case provides the minimum average download time possible.

Initially (in the instant of time 0) the adaptive system is in the low protection state (CR=0.9), and the adaptation begins in the next instant of time. Fig. 12 reflects how the adaptive system changes its protection state according to the client losses in each instant of time. As figure shows, the behavior of the adaptive system is rather good, since the server is sending data, in most cases, with the code rate that minimizes the download time. Apart from the initial instant of time, in the instant of time 8 the adaptive system is not transmitting with the optimum code rate. Nevertheless, in this case, the difference regarding the download time between code rates 0.7 and 0.9 is minimal, therefore in this scenario the adaptive system works almost perfect. Comparing with the ideal case, the adaptive system provides, on average, download times over 20% higher, which is a rather good result.

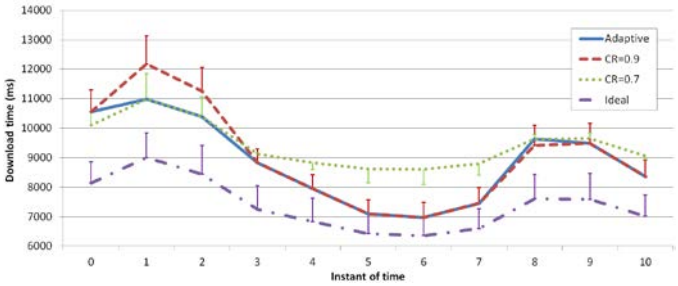


Fig. 12. Download time evaluation in scenario 1 for N_h=1/6, N_m=1/3.

The good behavior of the adaptive system is proven in scenarios 2 and 3, as Fig. 13 and Fig. 14 reflect.

Comparing the experimental and the analytical results in these three scenarios, we see that, in general, analytical results provide higher download times, but the difference is not very meaningful (it is not higher than a 10% in all cases). In fact, both experimental and analytical results provide the same adaptive protection state in all instants of times, so the analytical model works rather well. The differences among two models are due to the precision of transmission rate module of the implemented file server when calculating the experimental results and due to the value of the inefficiency ratio used in the analytical results. Note that the value of the inefficiency ratio can not be calculated analytically in some codes (those which do not belong to the MDS category), since the inefficiency ratio depends on the order of the packets upon arrival. In this study we have used a specific value of the inefficiency ratio for each code rate, according to the results presented in [18].

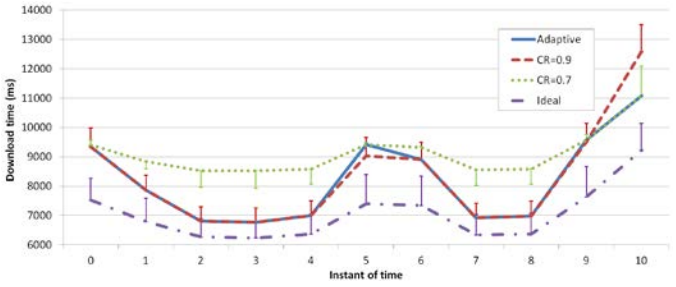


Fig. 13. Download time evaluation in scenario 2 for N_h=1/6, N_m=1/3.

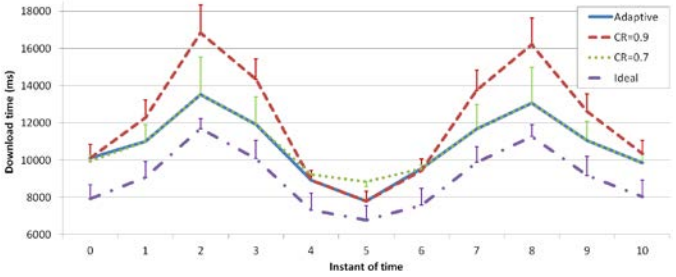


Fig. 14. Download time evaluation in scenario 3 for N_h=1/6, N_m=1/3.

As mentioned, one of the most important parameters to take into account is the bandwidth increase due to the use of AL-FEC. As Table II has shown, the bandwidth increases over 11% with a code rate equal to 0.9 (in low protection state), whereas the increase of bandwidth for 0.7 (in high protection state) is over 43%. Thus, the average bandwidth increase in the adaptive system will depend on the protection state. In scenario 1 the average bandwidth increase is 19.8%, in scenario 2 is 19.8% too, and in scenario 3 is 34.2%. The bandwidth distribution per time intervals in the different scenarios is shown in Fig. 15. In this point, it is worth comparing the results obtained with the adaptive case regarding the ideal case [16]. As mentioned, the ideal case provides, on average, a download time over a 20% lower than the adaptive case. Nevertheless, the ideal case provides an overhead of 396% (considering six parity channels plus the base channel).

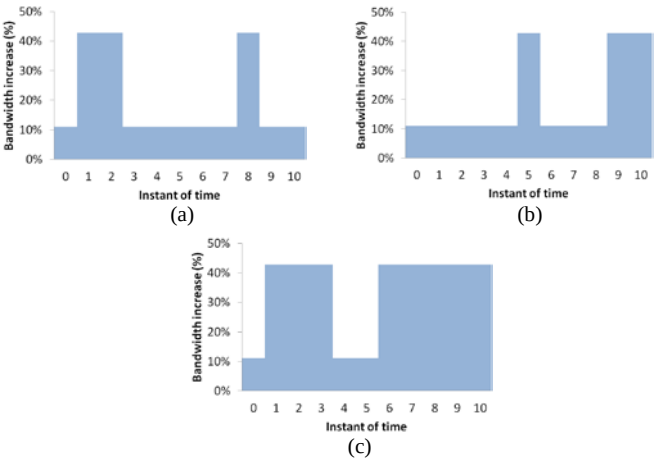


Fig. 15. Bandwidth increase in different scenarios. (a) Scenario 1. (b) Scenario 2. (c) Scenario 3.

So far, we have considered that clients with high and medium losses have more weight than those with low losses. In order to analyze the effect of this condition, the next study gives the same priority to all losses areas, therefore the values of N_h and N_m change: N_h=33% (1/3) and N_m=67% (2/3). As a result, the server tends to be more frequently in the low protection state, as Fig. 16 (scenario 1) and Fig. 17 (scenario 2) show. Comparing with the ideal case, the difference regarding the download time among adaptive and ideal is approximately the same that in the previous studies, that is, over 20%.

Nevertheless, although in previous scenarios the adaptive system works rather well, in environments with a huge number of clients with high losses, previous values of N_h and N_m are not appropriate, such as in scenario 3, shown in Fig. 18.

Once again, both analytical and experimental results for the previous three study cases provide similar values, with a difference regarding the download time lower than a 10%.

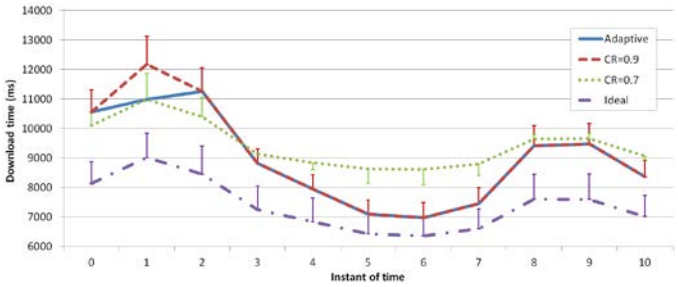


Fig. 16. Download time evaluation in scenario 1 for N_h=1/3, N_m=2/3.

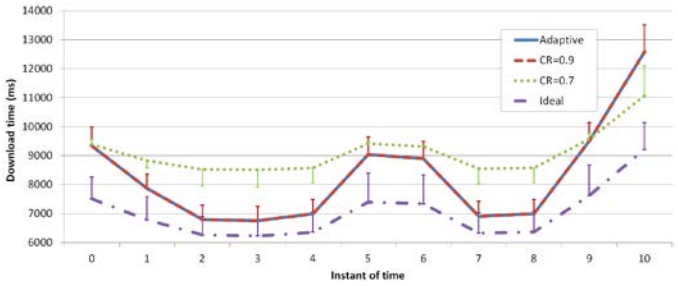


Fig. 17. Download time evaluation in scenario 2 for N_h=1/3, N_m=2/3.

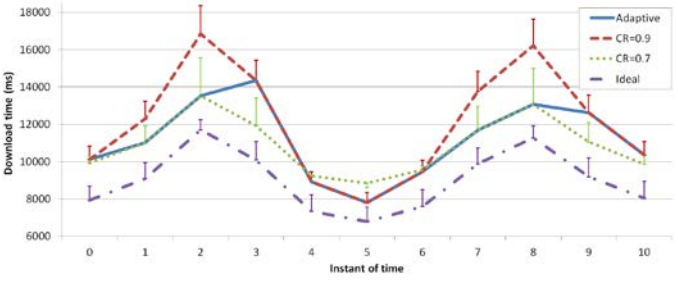
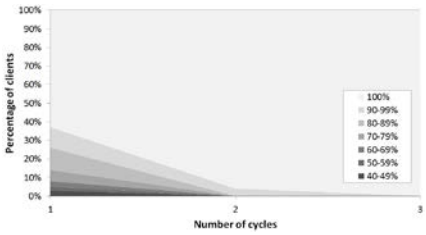


Fig. 18. Download time evaluation in scenario 3 for N_h=1/3, N_m=2/3.

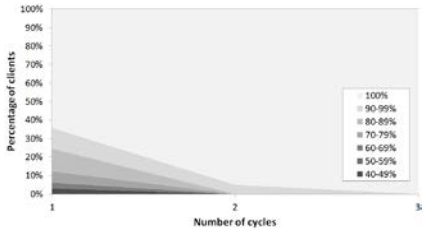
Finally, we study a particular case. We suppose that the server is sending the same file in a carousel, and each instant of time represents the moment when the server has sent the last packet of the file (that is, the end of the carousel). Thus, the time among two consecutive instants of time is the carousel period.

Initially all clients within the coverage area are interested in this file, so they start downloading it when the server begins to send data. Depending on the losses perceived by each client and on the code rate used to send the file, clients could need more than one transmission of the file (that is, various carousel

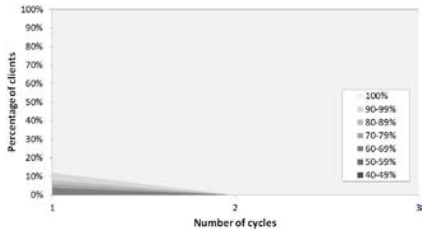
cycles) to download it. After completing the download, clients leave the channel, so do not send more reports to the server. Therefore, the server only will consider those reports received by the clients within the coverage area for each instant of time. Next study analyzes the number of clients that have completed their downloads in each carousel cycle. In this case, scenario 2 has been used.



(a)



(b)



(c)

Fig. 19. Evaluation of the number of cycles to complete the download in scenario 2 for N_h=1/6, N_m=1/3. (a) Adaptive. (b) CR=0.9. (c) CR=0.7.

Fig. 19 depicts, for each instant of time, the percentage of clients that have download a certain percentage of the file, analyzing the cases of adaptive code rate –Fig. 19(a) –, and code rates 0.9 –Fig. 19(b) –, and 0.7 –Fig. 19(c). Obviously, as the protection increases, the number of cycles decreases. It is good to recall that, although using a high protection (low code rate) always decreases the number of cycles needed to download a file, this does not entail that the download time decreases, since the carousel size and the carousel period increases, as we have seen previously. Focusing on the adaptive graph of Fig. 19(a), we can see that after the first cycle period (instant of time 1), 63% of clients have downloaded completely the file, whereas a 3% have not downloaded even the half of the file (40-49%). One carousel cycle later, 96% of clients have completed their downloads.

On the other hand, each time the server changes the code rate, clients have to discard the parity packets previously received. Even so, as Fig. 19(a) shows, the adaptive code rate provides (slightly) better results than the code rate 0.9 –Fig. 19(b). The performance can be improved if clients, instead of discarding parity symbols previously received when the code rate changes, save them in case the server sends data with the previous code rate in future carousel cycles.

It should be noted that considering that clients leave the channel once they have completed the download causes that the protection state changes. Thus, after the first carousel cycle the code rate changes from 0.9 to 0.7, whereas if consider that all clients remain within the coverage area, the code rate does not change during the first carousel cycles, as Fig. 13 showed.

D. Evaluation of the smoothing

In networks where clients have high mobility, losses are continuously changing. On some occasions, when calculating the losses for a certain instant of time it can occur that the estimation is not correct, since there are peak errors that distort the average losses. In those cases, it is very common the use of a smooth factor, through which the server considers both the current and the previous losses in order to estimate the system losses. In the three scenarios previously shown, as there were not abrupt changes regarding the losses, no smooth process was used (and thus α was equal to 1). In order to evaluate how this smooth parameter affects the system proposed, next study considers the scenario 4. Fig. 20 shows the distribution of the average losses of n=100 clients along time as well as the smooth effect for different values of α. In this scenario some bursts appear, where losses change rapidly.

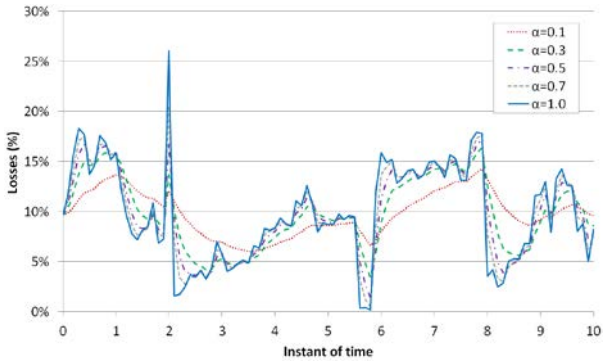


Fig. 20. Average losses distribution and smooth effect in scenario 4.

The methodology used is the same employed in the previous studies, that is, in each instant of time clients report their losses, the server smoothes them and then it chooses the optimum code rate according to this information. Note that clients only inform about the losses they are perceiving in a specific instant of time. For example, in the instant of time 1, clients only inform about the losses they perceive in that instant, and not about the partial losses perceived between the instants of time 0 and 1.

Fig. 21 compares the average download time obtained for different values of α for each instant of time. In the figure, the download time is in range [8000 ms, 12000 ms]. Due to the smooth process, in those cases where the value of α is low, the server tends to be in the same protection state. Fig. 21 shows that, in this scenario, smoothing the losses provides better results. Specifically, the system adapts better to the changing losses using α=0.3 or α=0.5. Analyzing the system behavior for α=0.3, we can see that the server sends at an optimum code rate in 8 out of 10 time intervals. This is a very good result, since in the instants of time when the server does not send with the optimum code rate, the difference between the

adaptive and the optimum code rate is pretty minimal. Therefore, we can consider that the adaptive system performs very well using an appropriate smooth factor. Obviously, an erroneous estimation of the losses can increase the average download time of the clients as well as the bandwidth.

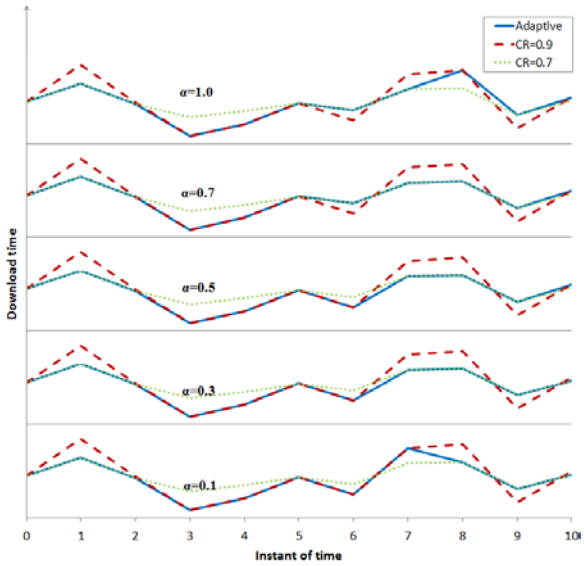


Fig. 21. Download time evaluation for different values of α in scenario 4 for N_h=1/6, N_m=1/3.

E. Delivery frequency of the feedback messages

Regarding the transmission of the feedback messages from the clients to the server, there are two main methods to send these reports: synchronously or asynchronously. In the first, clients send periodically their losses reports with a delivery frequency which depends on, for instance, the number of users and the available bandwidth. In the latter, clients inform about their losses only when it is necessary (for instance, when they detect a meaningful change of losses). As an example, it should be mentioned that RTP/RTCP [21] implement both mechanisms: a synchronous one, where users send their reports with a certain offset to avoid the feedback implosion problem; and an asynchronous mechanism (immediate feedback), where users report the server only when required.

In this way, when choosing the best policy to decide the delivery period of the feedback messages, there are several things to take into account:

- If the delivery period of the feedback messages is very short, too much traffic is introduced into the network. There would be many collisions between the feedback messages of all clients, which could lead to network congestion. Also, if losses do not change excessively it is not worth sending too many reports.
- Moreover it is not optimal to change continuously the code rate: clients have to discard continuously the parity packets previously received and generate the parity matrix. If clients do not send very frequently their reports, the server does not need to recalculate and generate a new optimum code rate constantly.
- On the other hand, sending feedback messages with a low frequency could become the system inefficient.

In this sense, we consider a good alternative that clients send the feedback messages each time they receive a FLUTE block. It should be noted that in FLUTE the coding is generated by block, so different blocks of the same file can have different coding and/or code rate. Thus, the delivery period of the feedback messages should be equal or higher than the transmission time of a FLUTE block. This solution is similar to the one used by DASH clients to request for new segments each time a segment is received.

The size of a block depends on the parameters established in the blocking algorithm (such as the code rate or the encoding symbol length). LDPC codes, used in this proposal, work more efficiently with higher block sizes [31]. But when choosing the size of the blocks it should be taken into account the computational resources of the mobile receivers (the higher the block size the more complex the decoding), so there is a trade-off between coding efficiency and computational cost.

Different tests carried out by the authors (some of them are shown in [18]) prove that a block size of 3000 FLUTE packets offers good results of the coding efficiency and the decoding time. Precisely, the experiments presented in this paper have been carried out with blocks of 3000-packet size. According to the results presented in Table I, the transmission of a block of this size takes among 7000 and 15000 ms using the optimal code rate for different channel losses. Thus, assuming an average of 10000 ms, with an average feedback delivery period of 10000 ms and 100 users in the system, the bandwidth consumed by the feedback messages will be around 8 kbps (considering that the feedback packet size is around 100 B). This is a good value taking into account that the multicast transmission rate considered is 5 Mbps. Also, following the DASH example, a feedback time of 10 s is reasonable compared to the feedback times used to request DASH segments (2 s in Microsoft Smooth Streaming or 10 s in Adobe HTTP Dynamic Streaming [32]).

These results show that the proposal of sending feedback messages after receiving each block results convenient in terms of bandwidth, which is one of the premises of this paper. In any case, an exhaustive analysis of the delivery period of the feedback messages is part of the future work.

F. Evaluation of the hysteresis

The last study evaluates how the hysteresis affects two main parameters: the average download time and the number of times that the state protection changes. If the server changes their protection state too frequently, the adaptive system could become inefficient: the server is changing the code rate continuously whereas the clients are consuming more computational resources since they have to process a lot of coding changes. Thus, there is a trade-off between minimizing the download time and the number of state changes.

Scenario 5 is used to evaluate the hysteresis, shown in Fig. 11. In that scenario, the average losses for each instant of time fluctuates in range [8%, 17%], in this way the state protection changes very frequently. With the aim of evaluating the hysteresis effect, we fix the lower threshold (λ_L) to 7% and we

increase the upper threshold (λ_M) progressively. Fig. 22 shows the results obtained regarding the number of state changes and the average download time of all clients considering all the time intervals. In the graph, the *x* label represents the threshold interval, that is, λ_M - λ_L. Thus, a value of threshold interval equal to 1 indicates that: λ_L=7% and λ_M =8%. It should be noted that, in this study we establish N_h=1/3 and N_m=2/3, so that the high losses do not mask the effect of the hysteresis.

Obviously, as the threshold interval increases, the number of state changes is lower, since the system tends to be in the same state protection. However, as the threshold interval is higher the average download time increases.

As Fig. 22 shows, the threshold interval used in the previous studies (5%, λ_L=7% and λ_M =12%) represents a good trade-off, since there are not very much changes of protection state and the average download time does not get worse considerably regarding the minimum value (only 1.5% higher).

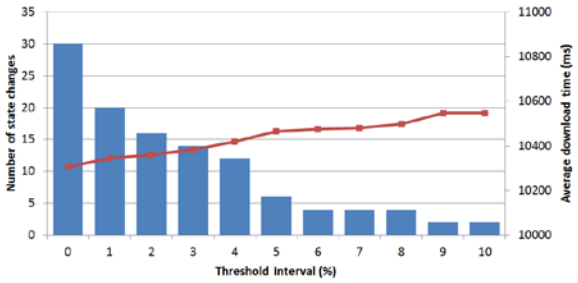


Fig. 22. Number of state changes and average download time depending on the threshold interval in scenario 5 for N_h=1/3, N_m=2/3.

G. *Server algorithm*

After evaluating all the parameters presented in the paper, this last subsection presents an algorithm that summarizes the process carried out by the server to obtain the best code rate for each instant of time so as to provide an optimal delivery.

It should be mentioned that the algorithm is heuristic, since it uses the different parameters obtained throughout the evaluation section. The fact that the algorithm works well for the five scenarios evaluated provides a certain guarantee to be valid in other scenarios.

Initially, the server fixes a conservative value of the code rate (CR=0.9) in order to save bandwidth. Also, the server considers an environment of low client mobility (so α=1) and that clients with high losses have more priority than those with low losses (N_h=1/6 and N_m=1/3). According to the previous studies the parameters that delimit each losses region are fixed to λ_L=7%, λ_M=12% and λ_H=30%.

For each instant of time the server creates a vector with the losses of those clients who have sent losses reports. If the percentage of clients with high losses is very low (≤5%), the server could give the same priority to all losses regions by assigning N_h=1/3 and N_m=2/3. Next, the server checks if the percentage of losses perceived by each client has changed a lot regarding the previous instant of time, by comparing the current losses vector with the previous. If the average increment or decrement of losses perceived by the clients is higher than a 10%, the server increases the value of α. Otherwise the server reduces α (note that $0 \leq \alpha \leq 1$). We apply the additive-increase/multiplicative-decrease algorithm (using the results obtained in the studies carried out in this paper), which is used by RTP/RTCP for dynamic adjustment of the bandwidth as well as TCP (Transmission Control Protocol) to manage network congestion. Then the server calculates the smoothed losses of all clients, generating a smooth losses vector. After that, the server counts the number of clients in each losses region and calculates the new protection state, and thus the code rate.

Algorithm 2

Fix parameters (λ_L=7%, λ_M=12%, λ_H=30%)
Initialize (CR=0.9, N_h=1/6 and N_m=1/3, α=1)
for (each instant of time)
    Generate "losses_vector" and calculate n_H, n
    if ($n_H/n \leq 5\%$) then N_h=1/3, N_m=2/3
    else then N_h=1/6, N_m=1/3
    end
    if (losses variation ≥ 10%) then $\alpha_{current}=0.8 \cdot \alpha_{prev}$
    else then $\alpha_{current}=\alpha_{prev}+0.1$
    end
    Calculate "smooth_losses_vector" and n_M, n_H, n
    if ($n_H/n \geq N_h$) then CR=0.7
    else if ($(n_M+n_H)/n \geq N_m$) then CR=0.7
    else then CR=0.9
    end
end

V. Conclusions

The adaptive system presented in this paper represents a good solution for file transmission in multicast environments. The use of an adaptive code rate minimizes the average download time of all clients within the coverage area, with a reasonable use of bandwidth.

Although there is an optimum code rate per each client depending on the amount of losses perceived, a given code rate performs well in a wide interval of packet losses around the code rate that minimizes the download time. Therefore, it is possible to send using a code rate that benefits the major part of users. In order to do this, it is necessary to analyze the losses perceived by each client and decide the optimum code rate for each situation. In the studies carried out two different code rates have been considered: one code rate when the major part of clients have low losses and another one when they have medium-high losses. Clients with high losses must have more priority than those with low losses, since using insufficient protection for clients with high losses penalizes more the download time than using too much protection for clients with low losses. As the results have shown, the adaptive system proposed works very well using only two different code rates. The value of these code rates has a great impact on the system performance, as well as the thresholds that delimit the protection state of the system, which establish the code rate used to transmit. In this sense, it is recommended the use of hysteresis to avoid too much coding changes.

As a particular case, this paper has shown a carousel where the server sends the same file in each loop and clients download the file in one or several carousel cycles, depending

on the losses. In that case, the adaptive system performs rather well, despite the fact that the optimum code rate could change every carousel cycle and clients must discard the parity packets previously received.

Finally, in environments where the losses perceived by the users change very abruptly, it is recommended that the server smoothes the losses when it chooses the optimum code rate. In that case, choosing an accurate smooth factor has a great influence on the suitable performance of the adaptive system.

To sum up, the adaptive mechanism proposed in this paper represents a good trade-off between the bandwidth used by a file server and the Quality of Experience perceived by the clients, therefore it is appropriate for content download services in multicast wireless networks.

REFERENCES

[1] *Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television*, ETSI EN 300 744 v1.5.1, Jun. 2004.

[2] Advanced Television Systems Committee (ATSC) standards webpage, available at: http://www.atsc.org/cms/index.php/standards/.

[3] World Bank, "Information and Communications for Development 2012: Maximizing Mobile," available at: http://www.worldbank.org, 2012.

[4] D. Dujovne and T. Turletti, "Multicast in 802.11 WLANs: an experimental study," presented at ACM Int. Symp. on MSWIM, Málaga, Spain, Oct. 2006.

[5] T. Stockhammer and M. G. Luby, "DASH in mobile networks and services," presented at IEEE Visual Communications and Image Processing (VCIP), San Diego, CA, USA, Nov. 2012.

[6] *Universal Mobile Telecommunications Systems (UMTS); LTE; Multimedia Broadcast/Multicast Service (MBMS); Protocols and Codecs*, ETSI TS 126 346 v11.3.0, Jan. 2013.

[7] D. Lecompte and F. Gabin, "Evolved Multimedia Broadcast/Multicast Service (eMBMS) in LTE-Advanced: Overview and Rel-11 Enhancements," *IEEE Communications Magazine*, vol. 50, no. 11, pp. 68-74, Nov. 2012.

[8] ISO/IEC 23009-1, "Dynamic adaptive streaming over HTTP (DASH) – Part 1: media presentation description and segment formats, 2012.

[9] T. Paila, R. Walsh, M. Luby, V. Roca, and R. Lehtonen, "FLUTE– File delivery over unidirectional transport," IETF RFC, vol. 6726, Nov. 2012.

[10] *Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Content Delivery Protocols*, ETSI TS 102 472, v1.3.1, Jun. 2009.

[11] *Digital Video Broadcasting (DVB); Transport of MPEG-2 TS Based DVB Services over IP based Networks (and associated XML)*, ETSI TS 102 034 v1.4.1, Aug. 2009.

[12] I. de Fez, F. Fraile, and J. C. Guerri, "Effect of the FDT transmission frequency for an optimum content delivery using the FLUTE protocol," *Computer Communications*, vol. 36, no. 12, pp. 1298-1309, Jul. 2013.

[13] J. Lacan and T. Pérennou, "Evaluation of error control mechanisms for 802.11b multicast transmissions," presented at Int. Symp. on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, Boston, USA, Apr. 2006

[14] M. Kobayashi, H. Nakayama, N. Ansari, and N. Kato, "Reliable application layer multicast over combined wired and wireless networks," *IEEE Trans. on Multimedia*, vol. 11, no. 8, pp. 1466-1477, Dec. 2009.

[15] H.-T. Chiao, S.-Y. Chang, K.-M. Li, Y.-T. Kuo, and M.-C. Tseng, "WiFi multicast streaming using AL-FEC inside the trains of high-speed rails," presented at IEEE Int. Symp. on BMSB, Seoul, Korea, Jun. 2012.

[16] I. de Fez, F. Fraile, R. Belda, and J. C. Guerri, "Analysis and evaluation of adaptive LDPC AL-FEC codes for content download services," *IEEE Trans. on Multimedia*, vol. 14, no. 3, pp. 641-650, Jun. 2012.

[17] C. Neumann, V. Roca, A. Francillon, and D. Furodet, "Impacts of packet scheduling and packet loss distribution on FEC performances: observations and recommendations," in *Proc. of CoNEXT*, Toulouse, France, Oct. 2005, pp. 166-176.

[18] 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 App.*, vol. 60, no. 3, pp. 669-688, Oct. 2012.

[19] I. Busse, B. Deffner, and H. Schulzrinne, "Dynamic QoS control of multimedia applications based on RTP," *Computer Communications*, vol. 19, no. 1, pp. 49-58, Jan. 1996.

[20] *Digital Video Broadcasting (DVB); Transport of MPEG-2 TS Based DVB Services Over IP Based Networks*, ETSI TS 102 024 v1.4.1, Aug. 2009.

[21] J. Ott, S. Wenger, N. Sato, C. Burmeister, and J. Rey, "Extended RTP profile for real-time transport control protocol (RTCP) –based feedback (RTP/AVPF)," *IETF RFC*, vol. 4585, Jul. 2006.

[22] J. C. Guerri, M. Esteve, C. Palau, and V. Casares, "Feedback flow control with hysteresial techniques for multimedia retrievals," *Multimedia Tools and App.*, vol. 13, no. 3, pp. 307-332, Mar. 2001.

[23] J. Peltotalo, S. Peltotalo, J. Harju, and R. Walsh, "Performance analysis of a file delivery system based on the FLUTE protocol," *Int. J. Commun. Syst.*, vol. 20, no. 6, pp. 633-659, Jun. 2007.

[24] R. G. Gallager, "Low density parity check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21-28, Jan. 1962.

[25] V. Roca, C. Neumann, and D. Furodet, "Low density parity check (LDPC) staircase and triangle forward error correction (FEC) schemes," IETF RFC, vol. 5170, Jun. 2008.

[26] 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)*, Bologna, Italy, Aug. 2008, pp. 274-282.

[27] M. Zúñiga-Zamalloa and B. Krishnamachari, "An analysis of unreliability and asymmetry in low-power wireless links," *ACM Trans. on Sensor Networks (TOSN)*, vol. 3, no. 2, Jun. 2007.

[28] IEEE Computer Society, "Std 802.11b, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band," Sep. 1999.

[29] T. S. Rappaport, *Wireless communications. Principles & Practice*, Prentice Hall Communications Engineering and Emerging Technologies Series, 1996.

[30] B. Liang and Z. Haas, "Predictive distance-based mobility management for PCS networks," in *Proc. of INFOCOM*, vol. 3, New York, USA, Mar. 1999, pp. 1377-1384.

[31] 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, Rhône-Alpes, Montbonnot-St-Martin, France, INRIA Res. Rep. RR-5225, Jun. 2004.

[32] S. Akshsabi, S. Narayanaswamy, A. C. Begen, C. Dovrolis, "An experimental evaluation of rate-adaptive video players over HTTP," *Signal Processing: Image Communication*, vol. 27, no. 4, pp. 271–287, 2012.

Ismael de Fez 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.

Juan Carlos Guerri 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 UPV, 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.