Evaluation of background push content download services to mobile devices over DVB networks

Francisco Fraile, Ismael de Fez and Juan Carlos Guerri

Abstract— **This paper proposes a multicast content download service based on the use of residual network capacity to push multimedia content to available local storage in personal multimedia devices. The service under study is based on the FLUTE protocol. Specifically, FLUTE packets fill the spare capacity in the IP tunnels reserved for the primary streaming service (opportunistic insertion). The paper also evaluates the use of Application Layer – Forward Error Correction (AL-FEC) parity to overcome transmission errors, object multiplexing to send the most popular multimedia contents more frequently and cache management policies that consider user preferences in order to keep in storage the most useful items. The service has been evaluated through simulations and measurements performed with an application prototype based on the DVB-H standards. The results show that AL-FEC enables the use of residual capacity for background content download services. In turn, AL-FEC, as well as object multiplexing, improves the relation between the number of content items and the overall access time. Moreover, results show that high percentages of requests can be served from the local cache of the service, provided that it is possible to estimate the popularity of content items and the user preferences.**

Index Terms— **Background services, AL-FEC, Cache management, FLUTE, Carousel scheduling**

EDICS— **2-SYSS, 8-MSAT, 8-WMMM**

I. Introduction

MOBILE television promoted the appearance of a number of technologies to provide IP multicast / broadcast access to mobile multimedia devices. This way, the last years have witnessed the release of standards such as IMB (Integrated Mobile Broadcast), to enable IP broadcast support in cellular 3G networks, or DVB-H (Digital Video Broadcasting - Handheld) and ATSC-H/M (Advanced Television Systems Committee –Handheld/Mobile) that are meant to enable the provision of IP services over terrestrial broadcast networks. The use of dedicated broadcast networks (and the associated investments) is only motivated when there is a critical mass of spectators looking at the same program and at the same time. This only happens for television content that is sufficiently popular in the service area. For the rest of the content offering, video delivery over point-to-point connections seems like a better alternative than broadcast. Obviously, in this case, the mobile network needs to be dimensioned to support peak traffic load, which is heavily dependent on IP video traffic. Without IP broadcast or multicast support, there are no means to save the limited radio spectrum resources when several users watch the same content.

Hence, both alternatives, i.e. the usage of dedicated broadcast networks (when few content items have many simultaneous viewers) and the usage of point-to-point connections (when simultaneous viewers look at different content items) require investments in network infrastructure that are only necessary during certain day parts. This is because, in both alternatives, it is assumed that content needs to be streamed, i.e. transmitted at the time that users watch it. However this is not always true, because pre-produced content is delay-tolerant, in the sense that it can be transmitted at any time after it is produced and stored in available storage space in wireless terminals until it is consumed.

This way, content items can be delivered over a push Content Download Service (CDS) and stored in local caches until users watch it. Moreover, the CDS can use network capacity between peak network loads, as a background service, so that the network is not overloaded with the addition of a new service.

The reuse of excess network capacity for the delivery of delay-tolerant multimedia content has been investigated for cellular networks [1]. In particular, cellular networks are dimensioned to support the expected peak traffic demand in the service area. Therefore, there are idle intervals when the demand is lower and the resulting excess capacity can be used to deliver delay-tolerant multimedia content through unicast connections. Exploiting this instantaneous excess of capacity for the delivery of delay-tolerant data services in mobile networks can increase both user perceived quality and service value. On the other hand, the use of prefetching and caching in broadcast networks has been investigated in the literature [2], [3], although not as background services. The results show that caching from broadcast CDSs is an efficient method for

Manuscript received October 11, 2012.

F. Fraile*, I. de Fez, and J. C. Guerri are with the Institute of Telecommunications and Multimedia Applications (iTEAM) from Universitat Politècnica de València, Camí de Vera s/n, 46022 Valencia, Spain (phone: 34-963879588; fax: 34-963879583; e-mail: ffraile@iteam.upv.es, isdefez@iteam.upv.es, jcguerri@dcom.upv.es). This work is founded in part by the PAID-05-12 program of the Universitat Politècnica de València.

delivering pre-produced multimedia content to a potentially massive audience.

This paper proposes the use of residual capacity of broadcast networks for the provision of background push CDS, as a way to reduce the traffic of unicast streaming services over cellular networks and improve the Quality of Experience (QoE) of mobile television services. The excess of capacity in broadcast networks can be used to push content to mobile terminals in any scenario where battery consumption is not constrained, for instance whenever the terminal is charging battery or in systems embedded in vehicles. Thus, the services under study in this paper push popular content to portable devices through existing broadcast connections, by means of background services that do not interfere with the primary network services. Background broadcast services reuse network investments, while at the same time consume little (and unused) network resources. Therefore, these services turn out to be inexpensive to network operators. Additionally, the use of broadcast capacity to send just popular content and the time-shifting achieved with background CDSs foster an appropriate use of broadcast spectrum. Hence, background broadcast push CDS can improve the efficiency of mobile television content delivery with very little overhead in terms of operational expenditures and no need for additional infrastructure.

This kind of background services can be provisioned over any wireless network with support for IP broadcast. Specifically, this paper proposes the provisioning of background push CDS in DVB networks through opportunistic insertion, in particular DVB-H networks. DVB networks carry a number of streaming services that are multiplexed together. Normally, operators leave some spare room in the multiplex, i.e. some marginal bandwidth for the sake of security during the multiplexing process. With opportunistic insertion, this filling capacity is used by a background push CDS to broadcast non-live content items, as an IP datacast service through the DVB network. The protocol stack of the proposed background CDS uses the FLUTE (File Delivery over Unidirectional Transport) protocol, adopted by DVB among other organizations, such as ATSC, 3GPP (Third Generation Partnership Project) or OMA (Open Mobile Alliance). FLUTE has been designed to deliver files in unidirectional environments where the network does not guarantee the delivery of data and it has been adopted for the delivery of CDS on several mobile TV standards besides DVB-H, like MBMS (Multimedia Broadcast / Multicast Services). FLUTE applies a transmission scheme known as file carousels, where files are repeatedly delivered in a continuous cycle. In order to establish the connection, clients need only to join the broadcast session to start receiving the packets of the carousel.

Background push CDS could play different roles for the primary mobile TV services, either linear TV services or content on demand services. For the latter, the service could be used to push content offered on demand to local storage, thus reducing the traffic of the on demand service for the content provider and the access time for the user. Other use cases, useful for both linear and on demand services, could be to detach the transmission of advertisements from the main transmission service or to send alternative content for playback under special circumstances, like no proper reception of the main service.

These use cases show that background push CDS could provide value for different actors in the value chain: television viewers benefit from improved Quality of Experience; content providers make more value of the resources dedicated to content delivery; and network operators make a better usage of radio spectrum resources. Moreover, it is clear that current trends in hardware capabilities, in terms of memory consumption, processing power and storage capacity make it feasible to support this kind of services.

However, the use of residual bandwidth involves difficulties for the provisioning of the service with acceptable performance. First, it is necessary to verify that the use of residual broadcast capacity in this scenario is feasible and provides sufficient bitrate for the CDS. Obviously, there is a trade-off between the number of files offered by the CDS and the time needed to access any of the files therein (access time). Thus, it is necessary to assess this trade-off for relevant number of files and their respective sizes. Last, it is necessary to manage appropriately the storage space available in user terminals in order to maximize the number of content requests served from local storage (cache hit ratio).

Taking this into consideration, this paper evaluates the combination of three different mechanisms to improve the performance of broadcast CDSs over background channels. In particular, the paper presents Application Layer Forward Error Correction (AL-FEC) mechanisms that improve the reliability of the content delivery service; object multiplexing to improve the relation between number of files and access time; and cache replacement policies that maximize the cache hit ratio. The combination of these techniques can optimize the performance of existing broadcast CDSs, allowing broadcast network operators to use the residual capacity in their networks to push mobile video programs to local storage in mobile terminals. The novelty of this research work is that it provides a framework to evaluate the performance of broadcast push CDSs combining AL-FEC, object multiplexing and caching. This framework is used to evaluate a service for mobile devices, including measurements over an application prototype in laboratory conditions. As mentioned, the application prototype is based on the DVB-H standard. However, it is worth noting that the proposal is applicable in other mobile TV standard specifications, for instance MBMS or ATSC-H/M.

The paper is structured as follows: Section II presents an overview of the relevant work in the three related areas; Section III explains the architecture of the service; Section IV provides the analytical models used to evaluate the background service; Section V presents the methodology used to evaluate the service; Section VI shows the measurement results and their corresponding analysis; Lastly, Section VII includes some final conclusions.

II. Related Work

This section presents the three main technologies under study: AL-FEC, object multiplexing and caching. Each subsection includes a description of the technology, references to related works and the contribution of this study to the state of the art.

A. AL-FEC

There are three complementary mechanisms to provide reliability in Content Download Services based on the FLUTE protocol [4]: file retransmissions, file repair sessions and AL-FEC. In unidirectional networks, content download services use carousel retransmissions in combination with AL-FEC.

The objective of AL-FEC is to reduce the number of cycles needed to download a file in the presence of errors, by applying error-correcting encoding to the file. The AL-FEC encoding algorithms work with pieces of the files known as source blocks, operating over smaller fragments known as encoding symbols. There are two kinds of encoding symbols: source and parity symbols. In FLUTE transmissions, first, the file is divided into source blocks, made of *k* source symbols. Later, the error-correcting code generates *n - k* parity symbols, out of the *k* symbols in the source block. Thus, *n* is the number of symbols after FEC encoding, as depicted in Fig. 1. The relationship between *n* and *k* is expressed either as the code rate *k/n* (CR) or its inverse, the FEC ratio (FR), *n*/*k*.

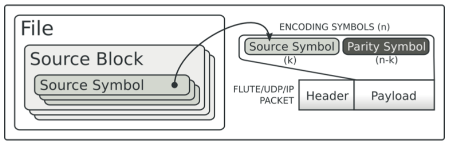

Fig. 1. FLUTE packet construction.

This way, by applying AL-FEC, the size of the file will increase by a factor equal to the FEC ratio. However, receivers will only need to download an amount of symbols slightly higher than the number of symbols that compose the file without AL-FEC encoding. The ratio between the number of symbols needed to successfully decode a file and the original number of symbols of the file is known as the inefficiency ratio (*inef_ratio*) and it depends of the AL-FEC code used. In this sense, an ideal AL-FEC code would provide an *inef_ratio* equal to one, regardless of the packet loss rate of the communication channel. [5] provides a comparison of the encoding efficiency of relevant AL-FEC codes for content download services. Besides the inefficiency ratio, there are other application aspects, like the content size or the memory requirements, which should be taken into consideration when deciding which AL-FEC code is better suited for a given application. There are different codes supported by FLUTE, such as Compact No-Code, Reed-Solomon, Raptor, RaptorQ or LDPC (Staircase and Triangle) [6].

In file carousels, the amount of AL-FEC parity added to a file and the channel losses affect the time needed for a client to recover a file completely, i.e. its download time. As explained in [7], there is an optimal code rate that minimizes the download time for clients of a file carousel. In this context, it is also important to bear in mind that different clients on a service area will experience different losses and consequently, the amount of AL-FEC parity needed should be optimized so as to minimize the average download time over all clients in a service area, instead of being optimized for the losses experienced by a client alone. [8] and [9] regard this problem from a network planning perspective, yielding to FLUTE file recovery configurations that minimize infrastructure costs or energy consumption.

There are other application parameters that affect the download time in file carousels, such as the distribution of file sizes, the relative popularity of the files added to the carousel or aspects related to file scheduling and cache management. One of the contributions of this study is to show how AL-FEC improves the download time of CDS taking into consideration these application parameters.

B. Object Multiplexing

Several research works focus on the optimization of broadcast data carousels in communication channels without errors [2], [3], [10]. These studies assume that clients do not issue the same number of requests to each item in the carousel, since some items are more popular than others. The results show how the average download time in a service area can be reduced by weighting the bitrate used to deliver each file, so that more popular items are transmitted more frequently. They propose different scheduling policies to accomplish this. [2] allocates the available rate in broadcast disks: sub-carousels with independent queues in which files are placed according to their popularity. Moreover, [10] considers the economic cost for the operator related to each data item. The results show that the proposed scheduling policy -based on the broadcast disk proposal- minimizes the overall cost of the service for the operator.

On the other hand, [3] states that it is not necessary to send several files concurrently, since the download time depends on the long-term average transmission rate assigned to each file. Thus, files are sent at the maximum available download rate one after the other, but more popular files appear more often in the carousels. Specifically, [3] derives an optimum value for the long-term average rate assigned to each file, which depends on its size and its popularity. Once the optimum long-term average rate is obtained, the carousel can be shaped by adapting fair queuing algorithms to implement scheduling policies for file carousels. In this paper, the optimum long-term average rates are obtained taking into account the effect of AL-FEC and the losses in the channel.

C. Cache Management

While object multiplexing aims at reducing the average download time by sending more often the most popular content items, caching aims at reducing the access time by storing files locally. In its most basic form, the problem of finding the optimum set of files that better utilize the available cache space reduces to the 0-1 knapsack optimization problem [11]: there is a set of items with certain weight and value and there is a need to determine which combination of items maximize the value, given a constraint on the total weight for the selection. For cache management, the size of files can be understood as the weight of items and the total size of the cache as the constraint on the total weight of the knapsack problem. However, the concept of value is open for interpretation and there are many different aspects that can be taken into account for its definition, like the size of the file or the time it was used for the last time. [12] presents a good compilation of cache replacement policies in the context of web caches.

The concept of value needs to be redefined for broadcast caches, quite different from web caches. [2] presents a cache replacement policy, named PIX, for caches of non-uniform accessed broadcast data in which the value of storing a file in memory is directly proportional to the future access probability of a file and inversely proportional to its relative transmission frequency (i.e. the inverse of its carousel cycle period). [13] introduces the size of the file in the calculation of its value in cache. In this paper, this policy is referred to as PIXS (Probability inverse frequency and size).

It is worth noting that the results in the aforementioned works are oriented to on demand applications and the content is cached after a request for a file. In this study, the cache works as a prefetching cache, downloading files from the broadcast carousel before the user issues a request for it.

III. Service Architecture

Fig. 2 shows the architecture of the multicast push CDS proposed in this paper. A broadcast *Content Download Service* will push contents over a background transmission channel, made of the residual transmission capacity in the reservations of a television service (the primary service). The server implements a *Scheduler* that decides which file to transmit at each time, to optimize the service performance. Also, the service inserts packets from the *Content Download Service* whenever there is capacity available (*Opportunistic Insertion* in the diagram).

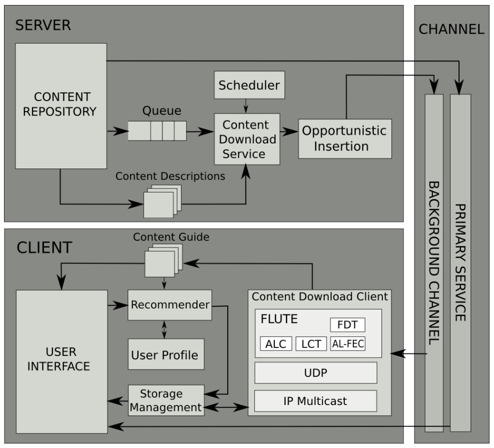

Fig. 2. Background multicast push CDS architecture.

The client will need to implement the corresponding broadcast *Content Download Client*. The figure presents the protocol stack of the one implemented and under study in this paper. The background service will push content to storage memory, managed by some *Storage Management Policy*, to ensure that the client uses only the storage capacity reserved for the background service. In this proposal, the *Recommender* uses feedback from user interaction and the metadata in the *Content Guide* to build a *User Profile* that models user preferences. The Recommender then provides the Storage Management with an estimation of the usefulness of each content item. In turn, the Storage Management uses this information to decide which contents to keep in storage.

As mentioned in the introduction, DVB broadcast networks transport streaming services over a constant bitrate MPEG Transport Stream (MPEG-TS). This study proposes opportunistic insertion of content download services based on the DVB-H IP protocol stack [14]. IP datagrams need some encapsulation protocol to be transmitted on top of the MPEG-TS. [15] defines a mechanism, time slicing, aimed at reducing the battery consumption of MPE decapsulation by sending the datagrams in bursts and shutting down the receiver at idle times.

Due to time slicing, the multiplex consists of bursts, which are made up of a considerable amount of video and audio packets belonging to the same mobile TV service. These bursts do not have a constant size. On the other hand, CDS services do not have real time requirements like the zapping time. Therefore, it is possible to provide a background content download service together with every video service so that the resulting burst size and burst period are kept constant, as shown in Fig. 3. Clearly, this simplifies the multiplexing process, making it easier to use the whole capacity of the multiplex.

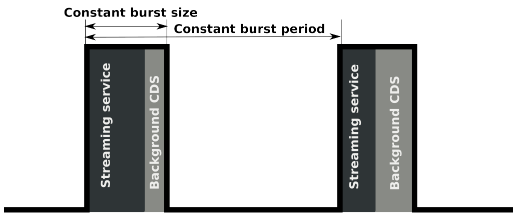

Fig. 3. DVB-H bursts with time slicing.

Some of the building blocks have already been introduced in the previous section. Regarding AL-FEC, the service uses LDPC coding, since they provide a good trade-off between performance (download time) and complexity [16]. As for the *Scheduler*, the carousel uses the whole available capacity to send the files as units, without interleaving packets belonging to different files. The operator needs to configure the weight of each file in the carousel, so that more popular files are transmitted at higher long-term bitrates. The calculation of the weights and the algorithm for the *Scheduler* are presented in the next section. Similarly, at the receiver, the cache management downloads files from the carousel, so as to keep in local storage the files that better fit the user needs. The size of the cache is limited and therefore, the cache management applies a cache replacement policy to maximize the value of the files kept in memory.

In order to determine the value of each file in cache, the *Recommender* uses the descriptions of the files in the *Content Guide* to apply content-based recommendation techniques [17]. As indicated in the figure, a content-based recommender will determine the usefulness of a file by comparing its description with the user profile.

IV. Theoretical analysis

This section provides an analytical model of the techniques under study in this paper: AL-FEC, object multiplexing and cache management. The main objective of these techniques is to improve the performance of the CDS proposed in this paper. Following the structure presented in Fig. 2, the section is organized describing the models at the server, the channel and the client needed to characterize these techniques. For the sake of clarity, Table I summarizes the main notation used:

TABLE I
Notation

b_j	Long-term bitrate of file *j*
$\bar{c}_j$	Average number of cycles to download file *j*
inef_ratio	Inefficiency ratio of AL-FEC encoding
k	Number of transmitted packets of a file
l	Number of missing packets on a cycle
m	Total number of missing packets at the beginning of a cycle
n	Number of transmitted packets of a file after AL-FEC encoding
N	Number of files in the carousel
p_j	Access probability of file *j*
$\tilde{p}_j$	Estimated probability of access of file *j*
r	Number of correctly received packets at the beginning of a cycle
s_A	Local storage size
s_j	Size of file *j*
s_j^{FEC}	File size of file *j* after AL-FEC encoding
t_A^j	Access time of file *j*
T_C^j	Long-term carousel cycle time of file *j*
$T_{C,FEC}^j$	Long-term carousel cycle time of file *j* after AL-FEC encoding
$t_C^j(k)$	k^{th} sub-carousel cycle time of file *j*
t_D^j	Download time of file *j*
t_W^j	Waiting time of file *j*
v_j	Value of file *j* in cache
x(i)	Expected number of received packets on cycle *i*

A. Server Model

1) Optimal scheduling

The first step in the analysis is to model the broadcast carousel transmission of the CDS, including the effect of AL-FEC, object multiplexing and opportunistic insertion. Later, the model is used to estimate the optimal long-term bitrates for the files in the carousel. Thus, the service will broadcast N files of sizes $s_1, s_2, \ldots, s_N$ that have a certain access probability in the service area. The access probability is represented by $p_1, p_2, \ldots, p_N$ where p_j is defined as the number of accesses to a file *j* divided by the overall number of accesses to all files. Client applications will need a given amount of time to fetch a certain file of the carousel and store it. In this study, this time is referred to as the *access time*. Connecting with the architecture (Fig. 2), the access time is the time between a request from the cache management to download a certain file and the instant when that file is completely downloaded to cache.

The objective of optimal scheduling is to minimize the *overall access time*, that is, the average of the access time of all file downloads in the service area. The access time to file *j*, t_A^j, is a random variable with expected value $\mathrm{E}[t_A^j]$. Thus, taking into account the access probability to files in the service area, the overall access time is calculated as:

$$\mathrm{E}[t_A] = \sum_{j=1}^{N} \mathrm{E}[t_A^j] \cdot p_j \qquad (1)$$

Therefore, the objective is to find a minimum of expression (1), accounting for the effect of AL-FEC, object multiplexing and opportunistic insertion. These techniques can affect the access time to files, but not the access probabilities. With this in mind t_A^j will depend on the *waiting time,* t_W^j, and the *download time*, t_D^j. The waiting time is the time between the instant when the client application joins the broadcast carousel and the time when it starts receiving packets of that file. On the other hand, the download time is the time needed to download the remainder of the file, after the first packet is received. Please note that t_D^j depends on the error rate:

$$t_A^j = t_W^j + t_D^j \qquad (2)$$

The next step is to provide an expression for t_W^j and t_D^j in carousel transmissions. For the sake of clarity, Fig. 4 shows an example of a transmission of three files, which are sent with different frequencies (for instance, F1 is sent more frequently than the others). The transmission frequency of each file *j* determines the *sub-carousel cycle time* of that file, $t_C^j(k)$, that is, the maximum time containing the k^{th} transmission of object *j*.

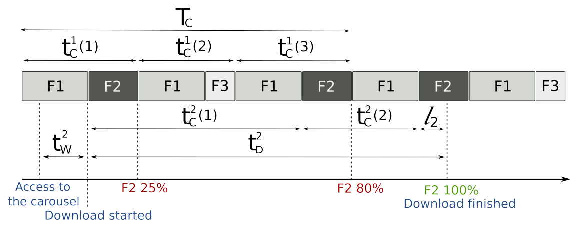

Fig. 4. Time model of the access time.

Regarding t_W^j, let us assume a client application joins the channel at an instant of time *t*, within sub-carousel cycle *k*. Assuming that the client application can join the carousel at any time with the same probability within $t_C^j(k)$, it is clear that the expected value for $t_W^j(k,t)$ is $\mathrm{E}[t_W^j(k,t)] = t_C^j(k)/2$.

Now, as noted in the picture, $t_C^j(k)$ is not necessarily the same for every index *k*. In the following equation, it is assumed that the scheduler generates a periodic carousel, with a certain *carousel period*, T_C. As Fig. 4 shows, each carousel cycle contains the same sequence of files (F1-F2-F1-F3-F1-F2 in the example). Inside the *carousel period* any element *j* appears an integer number of times, K_j. If $\rho_k = t_C(k)/T_C$ is the probability that the client joins the channel at sub-cycle *k*, the expectation value of t_W^j is:

$$\mathrm{E}[t_W^j] = \sum_{k=1}^{K_j} \rho_k \frac{t_C^j(k)}{2} = \frac{1}{2}\sum_{k=1}^{K_j} \rho_k t_C^j(k) = \frac{T_C^j}{2} \tag{3}$$

In the equation, T_C^j is defined as the long-term carousel cycle time of file *j*. For the calculation of the download time, t_D^j, if there are no losses, the download time is a deterministic variable equal to s_j/b, where s_j is the size of the file *j* and *b* is the bitrate of the datacast service. However, if there are losses in the reception of the datacast carousel, the download time becomes a statistical variable dependent of the number of cycles needed to complete the download. Specifically, to download a file *j*, clients need an entire number of cycles (c_j-1) plus a fraction l_j of the transmission of the file. For instance, in Fig. 4, in order to download file F2, the client needs 2 entire sub-carousel cycles (c_2-1=2) plus a fraction of the last sub-carousel cycle (l_2). The average number of cycles needed to download file *j* is defined as $\mathrm{E}\left[c_j\right] = \overline{c}_j$ and depends on the size of the file and the losses on the channel. If the receiver starts the download at cycle $t_C^j(k)$, after a time equal to $\sum_{l=k}^{k+\overline{c}_j-1} t_C^j(l)$, the receiver will begin the downloading of the last portion of the file. In order to fetch this last portion, the receiver will wait a time equal to $s_j l_j / b$, where l_j is a fraction of the transmission of file *j*. Note that, with channel losses, l_j is not necessarily equal to the portion of the file missing. In the example, in the last cycle after c_2, the receiver recovers 20% of F2, but since there are losses, it needs to wait a longer fraction of the time of the file in the carousel. Thus, the average download time can be divided in two terms, the first of them dependent on the carousel cycle:

$$\mathrm{E}[t_D^j(k)] = \sum_{i=k}^{k+\overline{c}_j-1} t_C^j(i) + \mathrm{E}\left[\frac{s_j \cdot l_j}{b}\right] \tag{4}$$

Now, the long-term download time of file *j* is defined as:

$$\mathrm{E}[t_D^j] = \sum_{k=1}^{K_j} \rho_k \cdot \mathrm{E}[t_D^j(k)] = \sum_{k=1}^{K_j} \rho_k \cdot \left(\sum_{i=k}^{k+\overline{c}_j-1} t_C^j(i) + \mathrm{E}\left[\frac{s_j \cdot l_j}{b}\right] \right) \tag{5}$$

note that both summations add consecutive cycles of the same file *j*. For instance, coming back to the example above where c_2-1=2 and K_2=2, with *k=1*, the inner summation results in $t_C^2(1) + t_C^2(2) + t_C^2(3)$. Similarly, with *k=2* the inner summation yields $t_C^2(2) + t_C^2(3) + t_C^2(4)$. Since the carousel is periodic, it turns that $t_C^j(1) = t_C^j(3)$ and $t_C^j(2) = t_C^j(4)$. Hence, it can be noted that when the outer summation is applied, every $t_C^j(k)$ is repeated exactly $\overline{c}_j$ times. Therefore, the terms of the summations can be re-arranged to obtain the relationship between $\mathrm{E}\left[t_D^j\right]$ and T_C^j:

$$\mathrm{E}[t_D^j] = \overline{c}_j \sum_{k=1}^{K_j} \rho_k \cdot t_C^j(k) + \mathrm{E}\left[\frac{s_j \cdot l_j}{b}\right] = \overline{c}_j T_C^j + \mathrm{E}\left[\frac{s_j \cdot l_j}{b}\right] \tag{6}$$

Combining the expected values for the waiting time and the download time, the expected value for the access time of a file *j* becomes:

$$\mathrm{E}[t_A^j] = \mathrm{E}[t_W^j] + \mathrm{E}[t_D^j] = T_C^j \cdot \left(\overline{c}_j + \frac{1}{2}\right) + \mathrm{E}\left[\frac{s_j \cdot l_j}{b}\right] \tag{7}$$

In the last expression, the average number of cycles and the portion of the file downloaded in the last cycle are mainly related to the losses in the communication channel and in turn to AL-FEC. On the other hand, the sub-carousel cycles of each file *j* are determined by the scheduler and related to object multiplexing. First, let us introduce the effect of AL-FEC. AL-FEC encoding reduces the average number of cycles $\overline{c}_j$ at the expense of increasing the file sizes by the FEC ratio (FR>1). Thus, AL-FEC will increase the size of all files to $s_j^{FEC} = s_j \cdot FR$. If AL-FEC is applied, long-term carousel cycle of each file is noted as $T_{C,FEC}^j$.

Now, object multiplexing will send some files more often, in order to adjust the long-term carousel cycle of each file. For instance, in the figure, F1 appears in the carousel more often than F2 or F3. The optimum configuration for the long-term carousel cycles provides a minimum of the overall access time, as defined in (1). Therefore, in order to determine the optimization problem, it is necessary to substitute $\mathrm{E}\left[t_A^j\right]$ in (1) for the expression derived in (7):

$$\mathrm{E}[t_A] = \sum_{j=1}^{N} \left(T_{C,FEC}^j \cdot \left(\overline{c}_j + \frac{1}{2}\right) + \mathrm{E}\left[\frac{s_j^{FEC} \cdot l_j}{b}\right] \right) \cdot p_j \tag{8}$$

With object multiplexing, files that are sent more often have shorter cycles and, in turn, shorter access times. This way, since files have different transmission frequencies, files are not transmitted at the same rate in the long run. Therefore, files will have different *long-term bitrates* and files with shorter average download times can be seen as files with higher long-term bitrates. The long-term bitrate assigned to file *j* (b_j) is defined as:

$$b_j = \frac{s_j^{FEC}}{T_{C,FEC}^j} \tag{9}$$

Note that b_j accounts for the effect of the scheduler but also for the effect of opportunistic insertion, since the long-term carousel cycle depends on the available bitrate *b*. Nevertheless, it is worth noting that the scheduler does not need to be aware of the available bitrate, since the set of long-term bitrates are constraint by:

$$\sum_{j=1}^{N} b_j = b \tag{10}$$

The scheduler will only need to know the ratios b_j/b that define the optimum share of the available bitrate among the different files in the carousel. Taking this into account, (8) can be rewritten as:

$$\mathrm{E}[t_A] = \sum_{j=1}^{N} \frac{s_j^{FEC}}{b_j}\left(\bar{c}_j + \frac{1}{2}\right)\cdot p_j + \sum_{j=1}^{N} \mathrm{E}\left[\frac{s_j^{FEC}\cdot l_j}{b}\right]\cdot p_j \tag{11}$$

Now, optimizing the carousel, i.e. finding the optimal sequence of files that minimizes the access time, is equivalent to finding the set of relative bitrates that minimizes the first term in (11). The only relationship between the different long-term bitrates is the boundary condition in equation (10), therefore they are independent variables, but subject to that condition. Thus, the following auxiliary function is used to solve the optimization problem:

$$\mathrm{f}(b_1,...,b_N) = \sum_{j=1}^{N} \frac{s_j^{FEC}\left(\bar{c}_j + 1/2\right)\cdot p_j}{b_j} + \sigma\left(b - \sum_{j=1}^{N} b_j\right) \tag{12}$$

In order to find the minimum of (12), its derivate is equaled to zero:

$$\frac{\delta \mathrm{f}(b_j)}{\delta b_j} = -\frac{s_j^{FEC}\left(\bar{c}_j + 1/2\right)\cdot p_j}{b_j^2} - \sigma = 0 \tag{13}$$

Providing the set of optimal bitrates:

$$b_j = \frac{b\sqrt{s_j^{FEC}\left(\bar{c}_j + 1/2\right)\cdot p_j}}{\sum_{i=1}^{N}\sqrt{s_i^{FEC}\left(\bar{c}_i + 1/2\right)\cdot p_i}} \tag{14}$$

2) *Object Multiplex algorithm*

The expression obtained in the previous section provides an optimum share of the available bandwidth between the different files. As explained, the scheduler will need to create a carousel that provides such long-term average bitrates, by multiplexing the different objects in the time domain. The problem can be regarded as a form of service discipline in which objects must be scheduled for transmission in a shared medium of limited capacity. This problem has been thoroughly studied in literature related to data packet scheduling. In fact, the algorithms originally proposed to deal with packet scheduling, such as WFQ (Weighted Fair Queuing) or VC (Virtual Clock), can be adapted to work with file scheduling. This is the case of the Modified Virtual Clock (MVC) algorithm proposed in [3], hereby adapted to account for the channel losses.

The MVC algorithm is divided into two phases. In the initialization phase the algorithm assigns to each file a delay, which is calculated as $\left(\sum_{j=1}^{N} b_j\right)/b_i$ and sorts the files according this value. In the multiplexing phase, the algorithm tries to adjust the cycle period of each data element to the delay value, placing data elements in a multiplexing queue ordered by increasing delay values.

The algorithm tries to adjust the long-term bitrates of files to the optimal values calculated in the previous section. The service operator needs an estimate of the access probabilities and of the channel losses. These parameters can be estimated from operational data of the primary services and it is worth noting that all these metrics are familiar to service providers and network operators. Specifically, the access probability can be estimated from audience measurements or access statistics on the primary services. On the other hand, the channel losses are estimated in the wireless network planning phase and are used to configure the optimal FEC rate and compute the average number of cycles. The amount of losses depends on the channel, as next subsection explains.

```
Phase 1: Initialization
1:   for i=1 to N
2:        object(i).delay= $\left(\sum_{j=1}^{N} b_j\right)/b_i$
3:        object(i).tag = object(i).delay
4:        object(i).count = 0
5:        object(i).enabled = TRUE
6:        QueueObjectsByOrderIncreasingTag()
7:   end

Phase 2: Multiplexing
8:   while (not_exit)
9:        i = FindObjectWithLeastTagInActiveQueue()
10:       SendObject(i)
11:       object(i).tag = object(i).tag + object(i).delay
12:       object(i).enabled = FALSE
13:       for j=1 to N
14:            object(j).count = object(j).count + 1
15:            if (object(j).count >= object(j).delay)
16:                 object(j).count = object(j).count-object(j).delay
17:                 object(j).enabled = TRUE
18:            end
19:       end
20:  end
```

B. *Channel Model*

For the system hereby proposed, the transmission channel is simulated using the two-state Markov model [18], also known as Gilbert model. This model is widely used in literature ([19], [20]), since it simulates well the error bursts typical in wireless networks. This model is based on two states: on/off. Each state indicates whether the last packet has been received or not. Depending on the state, there are different probabilities of losing the upcoming packet. In general, the probability of losing a packet is higher if the previous packet was lost.

As mentioned, if there are losses it is very likely that clients need several cycles in order to download the file. [21] performs a complete analysis of data carousels in channels with packet losses. In order to calculate the number of cycles needed to download a file, first it is necessary to know how many new packets are received per cycle. Formula (15) models the probability of receiving exactly x new packets in a loop using a hyper-geometric probability distribution. In the equation, k is the number of transmitted packets (source symbols) of the file, l is the number of lost packets in the loop and m is the total number of missing packets at the beginning of the loop.

$$P(x,m,k,l) = \frac{\binom{m}{x}\binom{k-m}{(k-l)-x}}{\binom{k}{k-l}} \tag{15}$$

The numerator expresses the possibilities of receiving exactly x new packets of the m missing packets out of the k - l packets received in a carousel cycle. Similarly, the denominator expresses the possible combinations of k - l packets out of the transmitted k packets. Applying this hyper-geometric probability distribution, the expected number of packets received at loop i is:

$$x(i) = \sum_{\xi=0}^{m} \xi P(\xi, m, k, l) \tag{16}$$

Finally, the number of cycles needed to download a file is calculated using an iterative process, according to:

$$c = \min(i) \mid x(i) \geq k \tag{17}$$

If AL-FEC is used, the probability of receiving x new packets in a new loop can be modeled in a similar way, although there are some changes in the equations, due to the benefits of error protection. Thus, formula (18), describing the probability of receiving x new packets at cycle i with AL-FEC is slightly different to formula (15). Here, r is the number of correctly received symbols at the beginning of the loop, n is the total number of encoding symbols (k source symbols plus n - k parity symbols) of the file and l is again the number of lost packets in the loop:

$$P(x, n, r, l) = \frac{\begin{pmatrix} n-r \\ x \end{pmatrix} \begin{pmatrix} r \\ (n-l)-x \end{pmatrix}}{\begin{pmatrix} n \\ n-l \end{pmatrix}} \tag{18}$$

In this case, the numerator expresses the possibilities of receiving x new packets of the n - r packets that have not been received correctly in previous cycles, out of the n - l packets that are received in the current cycle. The denominator expresses the total number of possible combinations of n - l packets in the n transmitted packets. Then, the expectation value is defined as:

$$x(i) = \sum_{\xi=0}^{n-r} \xi P(\xi, n, r, l) \tag{19}$$

Finally, recalling that the AL-FEC decoder needs to receive at least the number of source symbols k times the inefficiency ratio *inef_ratio*, the number of cycles needed to download a file when AL-FEC is applied is:

$$c = \min(i) \mid x(i) \geq k \cdot inef\_ratio \tag{20}$$

Therefore, with these formulas it is possible to calculate the number of cycles needed to download a file, depending on whether AL-FEC is used or not: the expected number of new packets received per loop is calculated iteratively until there are enough packets to recover the file.

In the calculation, the number of packets lost in every loop, l, is obtained from a two state Markov model. The Markov model determines how many of the transmitted packets (k when no AL-FEC is applied and n when AL-FEC is applied) are correctly received. The parameters of the Markov model are adjusted to match the statistical properties of error traces measured in the cases under study.

C. *Client Model*

Up to this point, the analysis has covered the models for the server and the channel. This section will describe the models used at the client side. Back to the architecture in Fig. 2, the client implements the CDS client, the cache and the storage management. Moreover, the storage management uses information from the recommender to decide which files to keep in cache. Initially, the cache is empty. The CDS client will fetch a file from the carousel and store it in cache as requested by the cache management. It is worth noting that, since the service under study is a background service, the user does not implicitly requests the CDS client to download a file. Instead, it is the recommender that initiates the download process.

As explained in the introduction, recommenders calculate the *utility* (or usefulness) of a content item for a particular user through a given utility function [17]. The design of the recommender and the details of such utility functions are out of the scope of this thesis. For the purpose of this study, it is enough to acknowledge that a recommender will determine how useful each of the content items are for the user. However, for the sake of clarity, let us explain briefly how a content filtering recommender [17] would work inside the client application.

Thus, the recommender analyses the content descriptions to determine the utility of file j, $\tilde{p}_j$. Later, the cache management will calculate the value of file j, v_j, as a function of $\tilde{p}_j$. The cache has a storage capacity equal to s_A. As this storage capacity is, in general, smaller than the sum of the size of all files in the carousel, the storage management must decide the set of files that maximize the overall value of the files in cache. In this study, the cache management only keeps entire files. Therefore, the cache management needs to find the decision vector $Y=\{y_1,y_2,\ldots,y_N\}$ that maximizes the value of files in cache, where y_j=1 if the storage management decides that file j should be kept in memory, or 0 otherwise. This problem can be expressed as:

$$\text{Find } Y=\{y_1,y_2,\ldots,y_N\},\ y_i \in \{0,1\}\ /$$
$$\text{maximize } \sum_{i=1}^{N} y_i v_i,\ \sum_{i=1}^{N} y_i s_i \leq s_A \tag{21}$$

Clearly, this is an instance of the 0-1 knapsack problem, thoroughly studied in the literature [11]. The problem is NP-complete, but there are many algorithms that solve it in polynomial time, each one optimized for a particular kind of instance of the problem.

The algorithm used in our proposal can model the decisions made by algorithms for cache management policies based on the branch-and-bound algorithms, which is the most basic approach to solve the 0-1 knapsack problem. The cache management algorithm decides which files should be stored in memory every time t_k when the recommender provides a new estimation of the utility of a file. Note that the recommender may have not estimated the value of all files in the carousel at t_k. Let I^k be the subset of files of the carousel with a value estimation up to the beginning of t_k. I^0 is initially empty. The algorithm will find the decision vector $Y=\{y_1,y_2,..,y_N\}$ by ordering the files in descending value, conditioned by their sizes:

1. Sort I^k such that $\frac{v_j}{s_j} \geq \frac{v_{j+1}}{s_{j+1}}$
2. Find $i_m = \min(i : \sum_{j=1}^{i} s_j \geq s_A)$ (22)
3. $y_j = 1; j < i_m$ and $y_j = 0; j \geq i_m$

The branch-and-bound algorithm presented above will model how the storage management policy will handle the storage space, according to an estimation of the utility of a file provided by the recommender. At every time t_k, the cache management will calculate the decision vector Y. If y_j changes from 0 to 1, the storage management will issue a download request for file j to the CDS client. Contrarily, if y_j changes from 1 to 0, it will remove the data of file j stored in the cache.

Regarding the definition of value, as explained in the related work section, there are different definitions for the value of files in broadcast caches. Table II provides the definitions under study in this paper:

TABLE II
DEFINITION OF VALUE FOR DIFFERENT CACHE REPLACEMENT POLICIES

Algorithm	Value of object *j* in cache
P	$v_j = \tilde{p}_j$
PIX	$v_j = \tilde{p}_j \cdot t_C^j$
PIXS	$v_j = \tilde{p}_j \cdot t_C^j / s_j$

Each cache replacement policy will estimate v_j using different parameters as a function of the utility. Note that, in this study the utility is seen as an estimation of the future probability of access to file j, as defined in the broadcast cache literature. The PIX and PIXS policies also account for t_C^j, the carousel cycle time of file j. Note that, in order to use t_C^j in the calculation of the value, the recommender needs to know the scheduling of the files beforehand. Finally, the PIXS policy also accounts for s_i, which is the size of file i.

V. EVALUATION METHODOLOGY

The main purpose of this study is to validate the performance of background push CDS over terrestrial DVB networks. The case under study is a background push CDS associated to a DVB-H mobile service, although it is worth noting that the background CDS service can be provisioned over any hybrid unicast / multicast TV platform [23].

The background push CDS implementation has been tested in laboratory conditions. Specifically, the performance of the background CDS channel has been measured experimentally, as described in section V.B. The objective of the measurements has been to characterize the long-term bitrate and the average number of cycles with opportunistic insertion in this scenario ($E[b]$ and $\overline{c}_j$, respectively). Additionally, the measurement results show configurations of the AL-FEC layer of the background CDS service that minimize the download time of a single file.

Later, the overall average access time is evaluated for different probability distributions of the popularity and the file sizes. The results are generated through simulations according to the analytical model and applying the empirical values obtained from the measurements. The overall access time is regarded as the most important quality metric for the background CDS as a standalone service. Even though the operational costs of the service for the operator are really low, the service will only be viable if it can provide sufficiently large catalogues of content at reasonable rates.

Finally, the study includes a performance analysis of different storage management strategies for broadcast caches, using the channel model described in Section III and the results from the measurements. The parameters selected to evaluate the performance of the storage management policies are two. First, the *cache hit ratio*, which is the ratio between the requests served from cache to the total number of requests generated. The second parameter taken into consideration represents the time needed by a cache management policy to complete the download of the files selected to fill the available storage capacity, refer to as *delta* time. These two parameters are relevant when the background service is associated to a video on demand (VOD) service, as a utility to lower the bandwidth consumed by the primary service. After comparing the cache management policies discussed in the previous section, the results show the relationship between cache size and cache hit ratio in different reception scenarios.

A. Parameters for the evaluation

The evaluation of the *access time* needs a model for the access probability distribution and the file size distribution. The access probability distribution regarded is the ZIPF distribution [24]. This is a quite common approach to model the relative popularity of multimedia items. The ZIPF distribution is defined by the parameter α, which indicates how fast the relative popularity of files decreases. In the results below, the range of values for α (0.6 – 2) is derived from previous related studies ([24], [25], [26]). The lower values, α (0.6-0.85) are taken from previous studies on the popularity of web files and video sharing sites, while the higher values are taken from [26], which investigates the suitability of multicasting against unicasting depending on the content popularity. The main conclusion is that the benefits of multicasting are more noticeable when α is higher than 1.5. For this reason, we have analyzed this case more thoroughly in the results.

Similarly, the sizes of files used in the simulations follow a lognormal distribution, as suggested in [27]. The parameters of the lognormal distribution (μ, σ) can be adjusted to provide certain mean and variance values. In this study, the mean file size and the variance of the lognormal distribution are adjusted to match the statistics of the file size in a popular VOD service [28]. The files sent through the background CDS represent videos and therefore the mean file size is related to their mean duration. The relationship between file size and video length is given by the encoding rate. The studies assume video formats and encoding rates typical of portable devices. In particular, the video encoding rate is set to 384 kbps, which is the maximum encoding rate of a video with resolution 320x240 and 20 frames per second with H.264 (basic profile, level 1.2). The audio encoding rate is 128 kbps, providing a bitrate of 640 kbps for the video. With this, a file size of 10Mb corresponds to approximately 2 minutes of video.

In summary, the simulations do not use models for the popularity and the file size specific of any particular application. For the popularity, we use models considered relevant for broadcasting applications. Similarly, for the file size we use a lognormal model based on the statistics of a video sharing site, corresponding to short videos (2 minutes on average) suitable for mobile video applications.

Regarding AL-FEC, the simulations use different AL-FEC code rates. The model simulates an AL-FEC with an efficiency ratio of 1.07. Note that the measurements have been carried out using an implementation of the LDPC Staircase codes developed by the authors. According to [22] and [30], the inefficiency ratio of LDPC Staircase is approximately 1.07 for sufficiently large files and encoding rates.

In the client, the knapsack algorithm decides which files to store in cache. In the simulations, the description of the file, needed to compute its associated value, is transmitted at the beginning of each file. This way, the decision vector is modified every time the client finds a new file in the carousel. The requests generated by VOD clients are modeled as a Poisson distributed event, completely independent of the status of the carousel or the cache. The expected value of requests per unit of time is defined as λ=50. Furthermore, the expected value of requests per unit of time for each file is $\lambda_i = \lambda \cdot p_i$, where p_i is the probability of access for file i, according to the ZIPF probability distribution.

It is worth noting that there could be deviations from the estimated values used by the MVC algorithm and the actual popularity in the service area, i.e. implementation losses related to the scheduler and the recommender. These implementation losses are not regarded in this study. With this in mind, the results in this paper should be regarded as upper bounds of the access times and cache hit ratios that can be accomplished with background push CDS.

B. Measurement setup

Fig. 5 shows the diagram of the measurement setup. The server implements a video streaming server, to generate the traffic of the primary video service and a FLUTE server that generates the traffic for the background CDS service. All service layers except for the physical layer are implemented in software, providing an MPEG Transport Stream to an external baseband DVB-H modulator. The radiofrequency (RF) waveform is generated from the baseband DVB-H signal with an Arbitrary Waveform Generator (AWG). To perform measurements for error free reception (0% losses), the receiver is just connected to the transmitter. In order to simulate mobile reception with losses, the AWG implements a baseband channel simulator. The channel simulator applies a TU6 channel model [29] with a Doppler speed of 50 km/h to the baseband signal, in order to simulate urban mobile reception. The CNR level of the received signal is set to two different values, to emulate two reception scenarios: good reception, defined as less than 5% of losses [31], and bad reception (50% losses). Finally, the receiver demodulates the DVB-H signal, extracts the CDS service and generates the measurement results.

The automation of the measurement procedure is achieved by allowing the client to reconfigure the parameters of the server. This way, the client can evaluate a configuration a number of times and then update the configuration of the server, thus automating the generation of results. The client software controls the server through a control channel that is not part of the service architecture, while the parameters for the measurements (number of files, AL-FEC configuration, number of iterations) are written down in test scripts. These scripts contain configuration parameters for the carousel, telling the server how many files should be included in the transmission and their parameters (size or AL-FEC encoding).

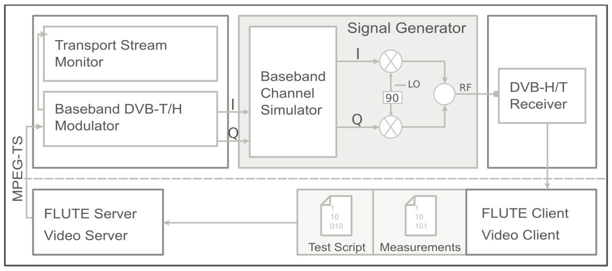

Fig. 5. Block diagram of the configuration of the measurements

Fig. 6 shows an example of a network capture of the received IP packets for both the primary streaming service (RTP) and the associated background service (FLUTE). First, the picture shows that the packets from both broadcast services arrive in bursts. The bursts have a fixed size and the CDS packets are used to fill in the spare burst capacity unused by the streaming service, as showed in Fig. 3. Therefore, on every measurement point, the number of FLUTE packets received is significantly smaller than the number of RTP packets.

The long-term bitrate for the CDS service in this scenario (E[b]) is 50kbps. Next sections present the results for AL-FEC, optimal scheduling and cache management.

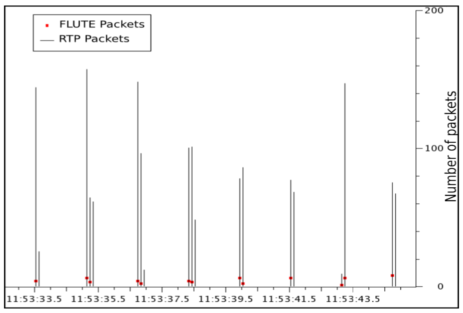

Fig. 6. Traffic generated by the primary streaming service and the associated background CDS service.

VI. Results and Analysis

A. AL-FEC

As mentioned above, the background service uses two different error recovery techniques. Apart from AL-FEC, the service uses carousel retransmissions in order to allow clients to obtain packets missed in previous carousel cycles. It is clear that the number of cycles has a substantial impact on the performance of the service, especially if the number of files present in the carousel is high.

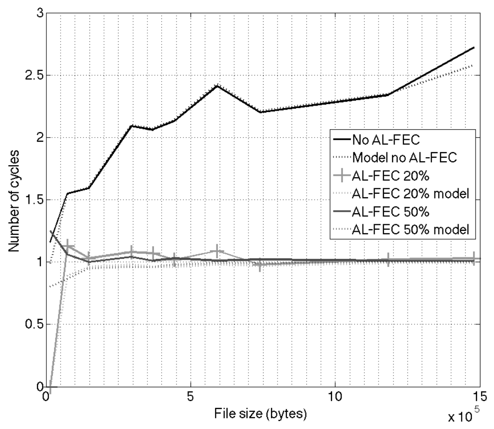

Fig. 7. Cycles needed to recover a file in good reception conditions.

Fig. 7 shows the number of cycles needed to download files of different sizes with 5% losses, while Fig. 8 shows the number of cycles needed with 50% losses for different configurations of AL-FEC: adding no AL-FEC parity, 20% of AL-FEC parity and 50% of AL-FEC parity. Both figures include the results obtained in the measurements against the results obtained with the simulation model. Clearly, the addition of AL-FEC parity reduces the number of cycles needed to recover the file. Fig. 7 shows that the average number of cycles with AL-FEC is approximately 1.

In Fig. 8, the number of cycles is significantly higher. The more parity added the less cycles are needed to download a file. Best results are provided by adding 50% of AL-FEC parity, corresponding to approximately 2 cycles. As for the channel model, the number of cycles provided is very similar to the number of cycles in the measurements in all cases.

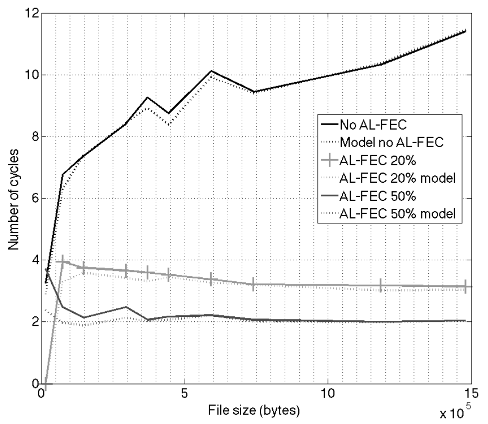

Fig. 8. Cycles needed to recover a file in bad reception conditions.

The last study evaluates the relation *k/n* (or code rate) with 0%, 5% and 50% of losses. In this study, a file of 500 kb is used. Fig. 9 shows the effective data rate, defined as the average bitrate perceived by the CDS application during the download of a file. Relating to the theoretical analysis, the effective rate is equal to the size of the file divided by the download time. This parameter is evaluated for different configurations of the LDPC AL-FEC code rate in the three scenarios.

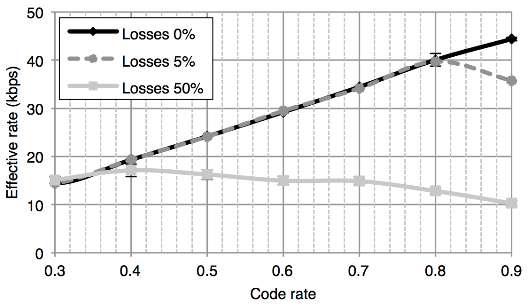

Fig. 9. Effective rate against different configurations of the AL-FEC block under three reception conditions.

With 5% losses, the addition of too many parity packets degrades the effective rate. Thus, for a given communication channel, there is an optimal AL-FEC code rate that maximizes the effective rate. However, this optimal value depends strongly on the reception conditions. For this reason, the optimum configuration of the AL-FEC module for 5% of losses is the addition of a moderate amount of FEC packets, with an optimum configuration of *k/n* over 0.8 (25% of AL-FEC) in the graph. With 50% losses, the code rate does not have such a drastic impact on the effective rate.

B. Object Multiplexing

The aim of the following results is to show the benefits of using object multiplexing. In this sense, Fig. 10 shows the ratio between the overall access time obtained with object multiplexing at the optimum long-term bitrates defined in eq. (15) and that obtained when the files are transmitted sequentially.

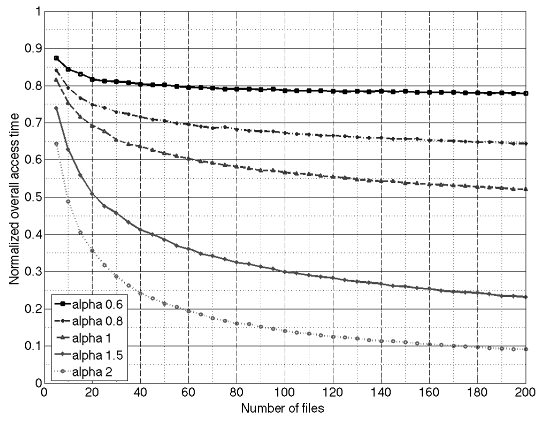

Fig. 10. Normalized overall access time without losses, mean file size = 10Mb.

As shown in Fig. 10, there are great benefits in using object multiplexing. Results show that the gain is greater as the number of files in the carousel increases and also, when the popularity distribution is sharper (greater α). For instance, for 200 files in the carousel and alpha = 2, the overall access time with object multiplexing is 10% the access time without object multiplexing.

Fig. 11 presents the same figure of merit, also against the number of files in the carousel, but this time in the presence of losses (5%) and for different amounts of AL-FEC added to each file. The graph shows that the effect of losses and AL-FEC is equivalent both for optimal scheduling and for sequential scheduling, meaning that there is no noticeable dependency between the object multiplexing gain and the configuration of the AL-FEC block or the amount of losses in the channel. This result has been validated against other simulations at different losses and AL-FEC rates.

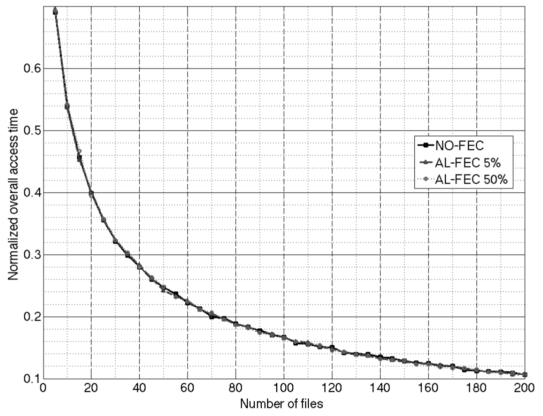

Fig. 11. Normalized overall access time with 5% of losses, ZIPF alpha = 2 and mean file size = 10 Mb.

In order to minimize the access time in the presence of losses, it is necessary to apply AL-FEC coding at an optimum rate. Fig. 12 shows the average access time achieved with different AL-FEC code rates with a 5% loss rate.

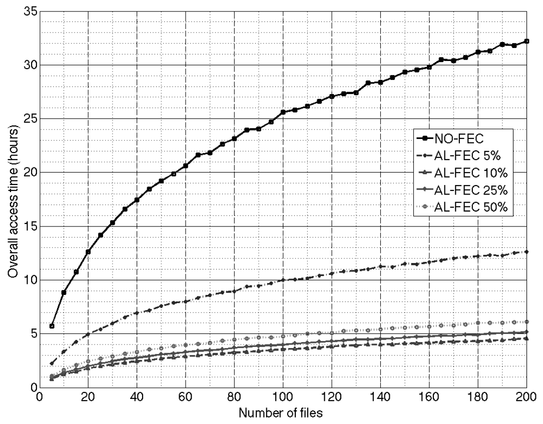

Fig. 12. Access time evaluation, with ZIPF alpha = 2, mean file size = 10Mb and 5% of losses.

Moreover, Fig. 13 provides the average access time experienced when the packet loss rate increases to 50% and, as expected, the results for the different configurations of the AL-FEC block do not keep the same order as in the previous study because, for every packet loss rate, there is an optimum FEC code rate that minimizes the download time of each file.

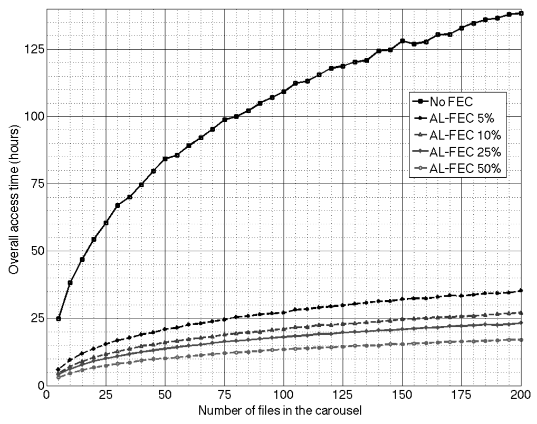

Fig. 13. Access time evaluation with ZIPF alpha = 2, mean file size = 10 Mb and 50% of losses.

At this point, it is interesting to compare the values that optimize the download of a single file (Fig. 9) and the values that optimize the download of a carousel (Fig. 12 and Fig. 13). For 5% channel losses, the optimum AL-FEC configuration for a single file is around 25% (CR = 0.8), while 10% (CR ≈ 0.9) and 50% (CR = 0.7) provide similar results. When evaluating the overall access time of carousel transmissions, it is worth noting that the size of the carousel is directly proportional to the amount of AL-FEC parity added. Therefore, the addition of AL-FEC parity has a dual effect: it lowers the average number of cycles at the expense of increasing the average waiting time. Fig. 12 shows that the minimum overall access time corresponds to 10% AL-FEC parity, although 25% and 50% AL-FEC parity provides similar overall access times, whereas the results for AL-FEC 5% are much worst. As for 50%, Fig. 13 shows that the overall access time in carousel downloads decreases gradually with the addition of AL-FEC parity, down to the minimum provided by 50% AL-FEC parity. In general, when evaluating the download time over an entire file carousel, there will be an optimum value of AL-FEC code rate that will minimize the access time. Moreover, the average access time is much higher if the AL-FEC parity is slightly below this optimum value than if the AL-FEC parity is higher. So far, the results show that object multiplexing and AL-FEC reduce considerably the overall access time. However, the overall access time obtained is very high to use the background push CDS as a standalone on demand service. Next section shows the performance of the background push CDS as a complementary service to a content on demand service.

C. Cache Management

The next results compare the cache hit ratio (CHR) achieved by the branch-and-bound algorithm of section IV.C, with different definitions of value, each one representing a different cache replacement policy. Additionally, the study compares the time needed by each policy to stabilize, referred to as delta. A cache will be stable once it has discovered all files in the carousel and it has downloaded completely those files with the best aggregated value, as determined by the branch-and-bound algorithm.

For this study, the effective rate is set to 50 kbps, while the CDS offers 100 files with sizes that follow a lognormal distribution of mean file sizes equal to 10 Mb. The average packet loss rate is 5%. As in previous studies, the probability of files follows a ZIPF distribution with α=1.5 and 2. The scheduler applies object multiplexing. The cache size is set to 4 times the mean file size. Table III shows the average percentage of files served from local storage (cache hit ratio, CHR) and the cache loading time (delta) of the different replacement policies presented:

TABLE III
CACHE HIT RATIO FOR DIFFERENT CACHE REPLACEMENT POLICIES

	α=1.5		α=2	
	CHR	Delta	CHR	Delta
P	70.8 %	65.8 hours	87.5 %	63.6 hours
PIX	60.4 %	68.1 hours	80.4 %	64.1 hours
PIXS	60.0 %	65.0 hours	74.6 %	63.4 hours

As the table shows, the best results are achieved by the *P* policy, although its results are very similar to those of the other two policies. It is worth noting that the implementation of the *P* policy is significantly simpler, because receivers do not need to know beforehand the scheduling of files or their relative transmission rates. Looking at the results, it is clear that the *PIX* and *PIXS* policies do not justify their additional complexity in terms of cache hit ratio or delta time. Regarding the delta value, the *P* policy will need more than 60 hours to have a stable cache with the files that best fit the user needs. Any update on the content carousel before the client has a stable cache could affect the cache hit ratio. Clearly, the cache in the chosen example is too small for a practical application. Fig. 14 shows the cache hit ratio for different cache sizes. Note that the cache size is normalized by the mean file size.

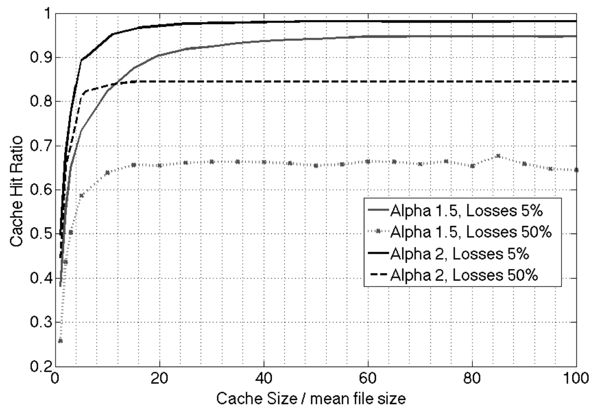

Fig. 14. Cache hit ratio against the cache size, relative to the mean file size.

Results obtained for other values of the mean file size are equivalent to those in the figure, meaning that the important parameter to take into account is the relationship between the cache size and the mean file size. Fig. 14 shows how the CHR increases with the size of the cache for the *P* cache replacement policy. In general, the hit ratios are rather high. For instance, with a mean file size of 10Mb and a cache size of 100Mb (cache size / mean file size = 10 in the x axis of the graph), the client handles more than 80% of the requests with the files in local storage. In both scenarios, the cache hit ratios achieve a maximum value, which does not improve for larger cache sizes. With higher losses, the CDS client needs more file cycles to download the files. This penalizes the CHR to some extent under bad reception conditions. However, the client application achieves good levels of cache hits in both reception scenarios under consideration.

At this point, it is important to emphasize that the popularity and file size distributions are application dependent. Hence, in extension, the results for the optimal cache size should be regarded as application dependent. As explained, the models used for the file sizes are characteristic of short videos encoded for mobile terminals.

VII. CONCLUSIONS

We have presented a background content download service associated with a broadcast mobile TV service. Results show that the service can effectively use residual bandwidth to push multimedia content to mobile receivers. The performance of background push CDS improves with AL-FEC, object multiplexing and caching.

First, the addition of AL-FEC mechanisms improves the performance of the background CDS. AL-FEC encoding improves drastically how the service scales with the number of files in the carousel in all the cases regarded in this study. Therefore, background CDSs should always implement some AL-FEC encoding. Moreover, results show that adding extra AL-FEC parity, above the optimum value, does not degrade the overall access time as much as adding less AL-FEC parity.

The object multiplexing technique also improves considerably the scalability of the overall access time with the number of files in the carousel.

At this point, it is interesting to see the improvement in the access time achieved by the combination of AL-FEC and object multiplexing. With an average bitrate of 50 kbps, an average file size of 10Mb, a carousel of 200 files, and 5% losses, the average access time to a file is over 9 days. With object multiplexing, the average access time is reduced to around 33 hours. Adding AL-FEC at an optimum rate lowers the average access time down to 5 hours.

Still, this value is very high to offer content to users and it is necessary to add a prefetching cache to improve the QoE of the service. The paper presents an algorithm to manage the storage space and evaluates its performance. The results show that the most relevant parameter to assess the value of files in cache is the future probability of access. Apparently, taking into account the size of files or their broadcast frequency does not improve the cache hit ratio.

Finally, results show that such caches can serve a considerable amount of VOD requests using little storage memory in the client. The cache hit ratio increases rapidly with the size of the cache up to a certain value. This value can be regarded as the optimum cache size, because larger cache sizes do not improve the cache hit ratio significantly. In the case under study, cache sizes of approximately 100Mb provide cache hit ratios higher than 65%, even under bad reception conditions.

REFERENCES

[1] P. Lungaro, Z. Segall, and J. Zander, "Predictive and Context-Aware Multimedia Content Delivery for Future Cellular Networks", presented at the IEEE Vehicular Technology Conference (VTC), Taipei, Taiwan, May 2010.

[2] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, "Broadcast disks: data management for asymmetric communication environments," in *Proc. of the ACM SIGMOD Conference on Management of Data*, San Jose, California, USA, May 1995, pp. 199-210.

[3] G. Zhiqi, Y. Songyu, and Z. Wenjun, "Using object multiplex technique in data broadcast on digital CATV channel," *IEEE Transactions on Broadcasting*, vol. 50, no. 2, pp. 113-119, Jun. 2004.

[4] T. Paila, R. Walsh, M. Luby, V. Roca, and R. Lehtonen, "FLUTE – File Delivery Over Unidirectional Transport," *IETF RFC*, vol. 6726, Nov. 2012.

[5] K. Nyborn, D. Vukobratoviç, and J. Björkqvist, "Sparse-Graph AL-FEC Solutions for IP Datacasting in DVB-H", presented at the IEEE Int. Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), Bilbao, Spain, May 2009.

[6] 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.

[7] I. de Fez, F. Fraile, R. Belda and J. C. Guerri, "Performance evaluation of AL-FEC LDPC codes for push content applications in wireless unidirectional environments," *Multimedia Tools and Applications,* vol. 60, no. 3, pp. 669-688, Jun. 2012.

[8] D. Barquero and A. Bria, "Forward Error Correction for File Delivery in DVB-H," in *Proc. of the Vehicular Technology Conference*, Baltimore, USA, Oct. 2007, pp. 2951-2955.

[9] T. Lohmar and J. Huschke, "Radio resource optimization for MBMS file transmissions," presented at the IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), Bilbao, Spain, May 2009.

[10] CK. Liaskos, SG. Petridou, and GI. Papadimitrou, "Cost-Aware Wireless Data Broadcasting," *IEEE Transactions on Broadcasting*, vol. 56, no.1, pp. 66-76, Mar. 2010

[11] H. Kellerer, U. Pferschy, and D. Pisinger, "Knapsack problems," *Springer*, 2004.

[12] S. Podlipnig and L. Böszörmenyi, "A survey of Web cache replacement strategies," *ACM Computing Surveys*, vol. 35, no. 4, pp. 374–398, Dec. 2003.

[13] J. Xu, Q. Hu, W. Lee, and D.L. Lee, "Performance evaluation of an optimal cache replacement policy for wireless data dissemination," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 1, pp. 125-139, Feb. 2004.

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

[15] *Digital Video Broadcasting (DVB); DVB specification for data broadcasting*, ETSI EN 301 192 v.1.4.2, Apr. 2004.

[16] E. Paolini, M. Varrella, M. Chiani, B. Matuz, and G. Liva, "Low-complexity LDPC codes with near-optimum performance over the BEC," in *Proc. of the Advanced Satellite Mobile Systems (ASMS)*, Bologna, Italy, Aug. 2008, pp. 274-282.

[17] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734-759, Jun. 2005.

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

[19] G. Gardikis, A. Kourtis, and P. Constantinou, "Modelling TCP performance in mobile DVB-T receivers," in *Proc. of the 8th WSEAS Conference on Communications*, Athens, Greece, Jul. 2004, pp. 632-635.

[20] J. Poikonen and J. Paavola, "Error models for the transport stream packet channel in the DVB-H link layer," in *Proc. of the IEEE International Conference on Communications*, Istanbul, Turkey, Jun. 2006, pp. 1861-1866.

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

[22] 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 Transactions on Multimedia*, vol. 14, no. 3, pp. 641-650, Jun. 2012.

[23] Z. Avramova, D. De Vleeschauwer, S. Wittevrongel, and H. Bruneel, "Capacity gain of mixed multicast/unicast transport schemes in a TV distribution network," *IEEE Transactions on Multimedia*, vol. 11, no. 5, pp. 918-931, Aug. 2009.

[24] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: evidence and implications," in *Proc. of the IEEE INFOCOM*, New York, USA, Mar. 1999, pp. 126-134.

[25] M. Chesire, A. Wolman, G. M. Voelker, and H. M. Levy, "Measurement and analysis of a streaming-media workload," in *Proc. of the 3rd Conference on Usenix Symposium on Internet Technologies and Systems*, San Francisco, California, USA, Mar. 2001, pp. 1-12.

[26] J. Aaltonen, J. Karvo, and S. Aalto, "Multicasting vs. unicasting in mobile communication systems," in *Proc. of the 5th ACM International Workshop on Wireless Mobile Multimedia (WOWMOM*), Atlanta, USA, Sep. 2002, pp. 104-108.

[27] A. Downey, "The structural cause of file size distributions," in *Proc. of the 9th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, vol. 29, no. 1, Cincinatti, Ohio, USA, Aug. 2001, pp. 361-370.

[28] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: a view from the edge," in *Proc. of the 7th ACM SIGCOMM Conference on Internet measurements (ICM '07)*, vol. 20, San Diego, California, USA, Oct. 2007, pp. 15-28.

[29] COST 207, "Digital land mobile radio communications (final report)," Commission of the European Communities, Directorate General Telecommunications, Information Industries and Innovation, 1989.

[30] 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.

[31] G. Faria, J. A. Henriksson, E. Stare, and P. Talmola "DVB-H: Digital Broadcast Services to Handheld Devices," in *Proc. of the IEEE*, vol. 94, no. 1, pp. 194-209, Jan. 2006.

Francisco Fraile obtained a degree in Telecommunication Engineering from the Universitat Politècnica de València (UPV) and the M. Sc. Degree in microwave engineering from the University of Gävle in 2004. Since then, until 2010, he has worked as a Research Engineer for the Swedish company Interactive TV Arena. In 2006, he joined the Multimedia Communications research group (COMM) of the Institute of Telecommunications and Multimedia Applications (iTEAM), UPV.

Ismael de Fez received the Telecommunications Engineering degree and the M.S. degree in Telematics from the Universitat Politècnica de València (UPV), Valencia, Spain, in 2007 and 2010, respectively. Currently, he is a Researcher at the (COMM) research group of the UPV where he is working toward the 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 Universitat Politècnica de València, where he leads the COMM research group of the iTEAM Institute. He is currently involved in research and development projects for the application of multimedia to industry, medicine, education, and communications.