De Fez Lava, I.; Fraile Gil, F.; Guerri Cebollada, JC. (2013). Effect of the FDT transmission frequency for an optimum content delivery using the FLUTE protocol. Computer Communications. 36(12):1298-1309. doi:10.1016/j.comcom.2013.04.008.

Effect of the FDT transmission frequency for an optimum content delivery using the FLUTE protocol

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

[a] Institute of Telecommunications and Multimedia Applications (iTEAM), Universitat Politècnica de València, Camino de Vera, 46071, Valencia, Spain

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

Abstract. File Delivery over Unidirectional Transport (FLUTE) is the standard protocol used in unidirectional environments to provide reliability in the transmission of multimedia files. The key element of this protocol is the use of the 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. Clients need to receive the FDT in order to start downloading files. Thus, the delivery of FDT packets and the proper configuration of their parameters have a great impact on the Quality of Experience perceived by the users of FLUTE content download services. This paper presents a complete analysis about how the FDT transmission frequency affects the download time of files. Moreover, results show which are the optimum values that minimize this download time. An appropriate configuration of the FDT transmission frequency as well as the use of AL-FEC mechanisms provides an optimum content delivery using the FLUTE protocol.

Keywords: File Delivery Table, FLUTE, Content Download Services, AL-FEC, Quality of Experience

1. Introduction

The access to multimedia content has grown significantly in the last years and nowadays transmission networks carry all kind of multimedia traffic. The visualization of multimedia streaming through video portals and photo sharing on social networks are two good examples of the great importance that multimedia content has on existing networks. Although a large proportion of this traffic is sent through unicast IP networks, the use of multicast networks has a great importance. For instance, multicast networks are used by TV channels or radio stations to broadcast their contents.

The utilization of multicast networks allows sending content to several clients using a single transmission operation. Thus, depending on the service, multicast networks result more efficient than unicast ones, which generate more traffic in the network. This way, multicast networks are very recommendable when there is sufficient number of users interested in receiving the same content.

Apart from broadcasting video, multicast networks can be used to transmit files. Note that in file downloads it is necessary to receive correctly all the packets that compose a file, so channel losses should not occur. However, multicast transport does not guarantee, generally, error free communication and the application layer protocol needs to be able to provide protection against errors. Additionally, some multicast file transmissions lack of a feedback channel that can be used by the clients to report the servers about the state of their downloads. Therefore, clients are not able to ask for packets lost during the transmission. Thus, file transmissions over unidirectional channels require a protocol that provides reliability. Among the different existing protocols, one of the most used by the different

standardization organisms is FLUTE (File Delivery over Unidirectional Transport). FLUTE is highly used for delivering multimedia content in unidirectional environments in a reliable manner.

The main characteristic of the protocol is the use of an in-band signaling mechanism, i.e. the File Delivery Table (FDT). The FDT describes the main properties of the files that are being transmitted, such as the file name, the content length or the encoding type. Once the clients receive the FDT, they know the files the server is delivering and can start downloading them.

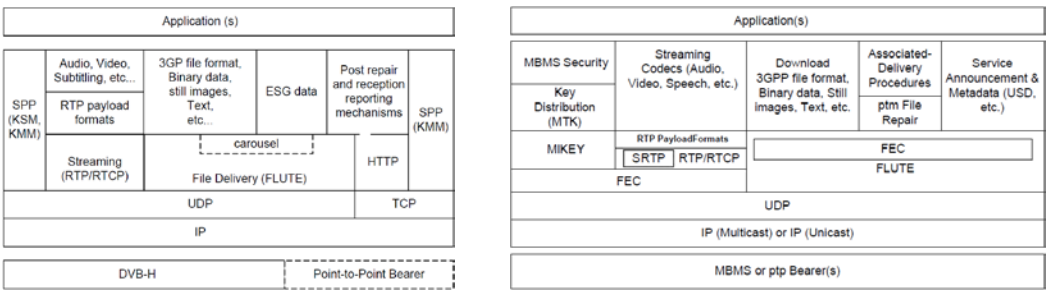
On the other hand, the Quality of Experience (QoE) perceived by the users is an important parameter in the evaluation of both video streaming and file transmission services. In the first, the delay, the losses and the video quality are key aspects for the QoE. Regarding file transmissions, a good QoE is obtained when the file is received correctly with the minimum download time. This is accomplished by sending files with a high transmission rate in channels with minimum losses. But these conditions cannot always be controlled and therefore it is essential to send the content in the most efficient way.

Regarding this, file transmissions through FLUTE can be optimized to improve the efficiency of the content delivery. In this sense, the FDT is a key element in FLUTE sessions and how often the FDT is transmitted has an important impact on the QoE of FLUTE services. On the one hand, if the FDT is sent with low frequency, clients have to wait a long time until they can start downloading the contents. On the other hand, if the FDT is received too frequently, clients will have to wait less time to receive the FDT (and therefore to start the download) at the expense of receiving more useless packets afterwards, thus reducing the efficiency of the transmission.

In this sense, this paper analyzes how the transmission of the FDT affects FLUTE file delivery sessions and evaluates which are the optimum values of the FDT transmission frequency that minimize the download time and, therefore, improve the Quality of Experience perceived by the users.

2. Related work

The FLUTE protocol, defined initially in RFC 3926 [1] and updated in RFC 6726 [2], has been established as the multicast file delivery protocol for different standards, such as DVB-H (Digital Video Broadcasting – Handheld) [3] and DVB-IPTV (DVB – Internet Protocol TV) [4]. Also FLUTE is used by the 3GPP Multimedia Broadcast Multicast Service (MBMS) [5] and evolved MBMS (eMBMS) [6] to download multimedia content. Fig. 1(a-b) show, as an example, the protocol stack of DVB-H and MBMS. As mentioned, both technologies use FLUTE to send different types of files, such as multimedia files, metadata or the Electronic Service Guide in DVB.



(a) IP datacast over DVB-H (b) MBMS IP multicast

Fig. 1 Protocol stack of DVB-H and MBMS.

In this regard there are research works that analyze the use of FLUTE in DVB and MBMS. For instance, [7] presents FLUTE as a multicast content delivery protocol to be used by DVB-H and MBMS, making a comparison

among different transmission protocols. On the other hand, [8] proposes a hybrid transmission network where FLUTE is used by DVB-H to broadcast multimedia content, whereas [9] develops a portable middleware for DVB-H clients. Also, other papers explore the use of FLUTE in MBMS, for instance [10].

But the use of FLUTE is not limited only for DVB and MBMS. For instance, [11] proposes the integration of FLUTE and TESLA (Timed Efficient Stream Loss-Tolerant Authentication) over satellite networks, whereas [12] presents an architecture for scalable DTN (Delay-Tolerant Networking) communication in sparsely populated areas based on FLUTE, similar to that proposed in [13].

Apart from file distribution, other papers propose FLUTE to provide video on demand services, such [14]. Also, [15] introduces an efficient progressive downloading over multimedia broadcast multicast service using FLUTE.

Among the research work related to FLUTE, the paper that best analyses the behaviour of the protocol is, probably, [16]. That paper makes a complete analysis of FLUTE, evaluating the different configuration parameters of the protocol and how these parameters affect the transmission. Among the several contributions to the FLUTE protocol from the authors of [16], one of the most important has been the publication of an open source implementation of FLUTE, available in [17]. In fact, that implementation has been used in different research works, for instance in some papers that analyze AL-FEC (Application Layered – Forward Error Correction) codes. Another research work of these authors worth highlighting is [18], which presents a bandwidth-efficient file delivery system using the server file format for FLUTE; and [19], where the use of congestion control protocols in FLUTE is studied.

However, despite that the FDT is one of the most characteristic elements of FLUTE, there are no papers where the FDT is analyzed in detail. This paper is intended to fill this gap.

3. FLUTE and FDT

FLUTE works over ALC (Asynchronous Layered Coding) [20], which is built over three building blocks: LCT (Layered Coding Transport) [21], a control congestion block and a FEC block [22]. Both ALC and LCT building blocks provide basic transport to FLUTE, which inherits the requirements of these blocks.

Moreover, one of the main error protection mechanisms of FLUTE is the use of a FEC block, which provides protection to the files sent. In this sense, FLUTE supports different FEC codes: Compact No-Code [23], Reed-Solomon [24], Raptor [25], RaptorQ [26] and LDPC (Staircase and Triangle) [27] codes. The use of one coding or another depends on the application but, in general, Raptor and RaptorQ work more efficiently at the expense of more complexity. Compact No-Code is recommended only on reliable channels (with very low losses), whereas Reed-Solomon is convenient when the amount of data to code is not very high. Also, LDPC codes are very efficient and their coding/decoding complexity is low.

FLUTE transmissions are based on sessions. Each session is sent in a certain IP address and with an identifier called TSI (Transport Session Identifier). Also, each session contains one or more associated channels, in which files are delivered. Each channel has an associated port number and sends with a certain transmission rate. On the other hand, each file transmitted in a file delivery session is uniquely identified by its content location. Internally, also it is used a file identifier called TOI (Transport Object Identifier).

To start receiving a file delivery session, clients need to know the transport parameters of that session, mainly the IP address, the TSI and the channel port number. These parameters are obtained through the Session Description. This can be obtained by different out-band mechanisms such as SDP (Session Delivery Protocol) [28][29], XML (Extensible Markup Language) or HTTP (Hypertext Transfer Protocol).

As mentioned, in FLUTE transmissions, before downloading a file, clients need to know the files that the server is sending. In this sense, the FDT is used to inform clients about the files that are being transmitted and their associated metadata. The FDT is sent as FDT Instances, which are made of FLUTE packets with a special extension header (called EXT_FDT) to indicate they carry FDT data. FDT Instances are sent in the same session and channel that the files. Also, FDT packets have assigned a value of TOI equal to 0. Each FDT is described using XML language.

The original FLUTE RFC [1] (published in 2004) and the current FLUTE standard [2] (published in 2012) establish that the number of files described in each FDT is variable, that is, each FDT Instance contains at least a single file description entry and at most the complete FDT of the file delivery session. So, there are two types of FDT Instances: partial FDT and complete FDT. An FDT Instance can be sent in any part of the file delivery session so packets for an FDT Instance may be interleaved with data packets. Moreover, both RFCs indicate that the way FDT Instances are transmitted has a large impact on the transmission. It is recommended to repeatedly transmit FDT Instances describing files while these are being transmitted. Also, it is highly recommended to send the FDT Instances reliably using FEC. [2] suggests that mechanisms used for FDT Instances transmission should achieve higher delivery probability than the file recovery probability. Nevertheless, neither [1] nor [2] analyze how often an FDT Instance should be sent and how much FEC protection should be provided for each FDT Instance.

Regarding the transmission, in FLUTE file delivery sessions each file is fragmented into different source blocks. Each source block is fragmented into encoding symbols. There are two types of encoding symbols: source and parity symbols. Parity symbols are additional data sent to provide reliability on the transmission. The payload of a FLUTE packet contains, at least, one encoding symbol.

Moreover, it is recommended the use of file transmission carousels. In carousels, files are sent cyclically on a seamlessly endless loop, which represents another reliability mechanism, since clients can complete their downloads if they have suffered losses in previous carousel cycles.

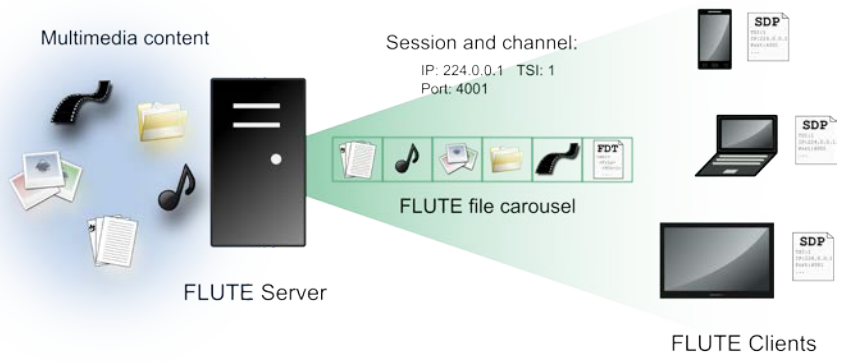


Fig. 2 File delivery using FLUTE protocol.

As a summary, Fig. 2 shows a general overview of a file transmission using the FLUTE protocol. A FLUTE server has a repository of multimedia contents. For example, it could send video, audio, images, documents… The FLUTE server broadcasts the contents in a certain session (identified by the IP address and the TSI), which contains, at least, one delivery channel (identified by the port number). On the other hand, clients will follow next steps:

- **Step 1**. Clients will obtain by an out-band mechanism the Session Description which contains the transport parameters associated to the session. The way clients obtain the Session Description is independent of FLUTE.
- **Step 2**. Once the clients have connected to a certain session they will have to wait until they receive the FDT that describes the files (and their corresponding metadata) that the server is sending.

- **Step 3**. Then clients will be able to identify the data packets they are receiving and they will be able to download the files they are interested in.

If the server wants to stop sending data, the FLUTE protocol contains a field in its header to indicate clients that the session will be closed soon.

4. Theoretical analysis

4.1 Introduction

The objective of the study is to calculate the total download time (t_T) of a file and evaluate how it is affected by the FDT delivery configuration. Throughout this section several variables appear. Table 1 shows these variables and explains their meaning.

b	Transmission rate	P	Probability of receiving a certain number of new encoding symbols for a block at a certain loop
CR	Code rate	S_i	Size of file *i*
CR_f	Code rate applied to files	S_L	Size of the file to download
CR_{FDT}	Code rate applied to FDT Instances	t_C	Cycle time
D_{FDT}	Number of data packets sent between two FDT Instances	t_D	Download time
e	Encoding symbol length of a FLUTE data packet	t_{FDT}	Delivery period of FDT Instances
e'	Encoding symbol length of a FLUTE FDT Instance packet	t_M	Time passed since the FDT is received until the client receives the first packet of the file to download
l	Number of lost packets per cycle	t_S	Time needed to send once the file to download
m	Number of times per cycle that an FDT is sent	t_T	Total download time
m_L	Number of FDT Instances of file *L* sent in each carousel cycle	t_W	Waiting time
N	Number of files in the carousel	v	Number of missing packets at the beginning of a cycle
n_{CR}	Number of packets that make up a file after applying AL-FEC	x	Expectation value of the number of new packets received at a certain carousel loop
n_{cycles}	Number of cycles needed to download a file	α	Adjustment factor of the file size
n'_{cycles}	Entire number of the n_{cycles}, defined as: ceil (n_{cycles})	α_L	Adjustment factor of the file size referred to the file *L*
$n_{cyclesFDT}$	Number of cycles needed to download an FDT	β	In the last cycle, it represents the remaining cycle percentage to complete an entire cycle, referred to the entire carousel
n_i	Number of packets that make up file *i*	β_L	In the last cycle, it represents the remaining cycle percentage to complete an entire cycle, referred to the file to download
n_L	Number of packets that make up the file to download	γ	Number of packets that compose an FDT

Table 1 Key notation.

This section analyzes five transmission configurations, each one explained in a different subsection as follows. Section 4.2 presents the general case: download of one file, use of a complete FDT and sequential scheduling. Section 4.3 analyzes partial FDTs, downloading one file and using sequential scheduling. Section 4.4 studies the scheduling model: download of one file, use of a complete FDT and interleaving scheduling. Section 4.5 considers multiple downloads, using a complete FDT and sequential scheduling. Finally, Section 4.6 analyzes the use of

prefetching with the parameters of the general case: download of one file, use of a complete FDT and sequential scheduling.

4.2 General case

As a first study case, among the two types of FDTs (partial and complete), this section analyzes the complete FDT. Each FDT Instance is transmitted in γ FLUTE packets, depending on the number of files and attributes that the FDT Instance describes. This study evaluates different cases: sending only one FDT Instance per cycle; sending as many FDT Instances as files in the carousel; and sending FDT Instances more frequently.

In this study, it is assumed that a server sends N files sequentially in a FLUTE carousel and that a user wants to download a certain file L with size S_L. Fig. 3 shows an example of a FLUTE delivery session based on carousels where three files are sent. Note that the FDT Instances are sent in-band with the files. In the example, a certain client wants to download the file *F3*.

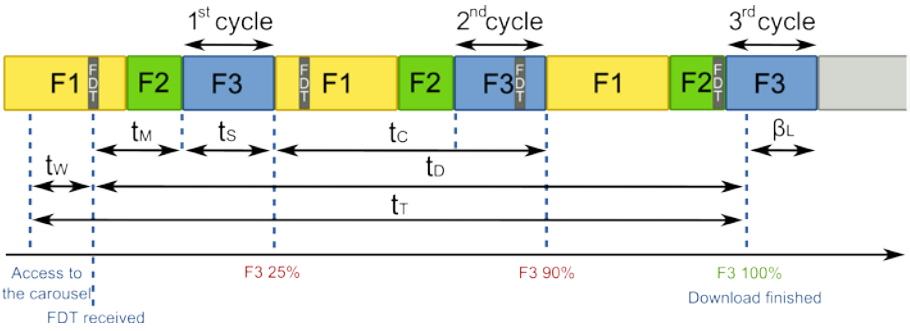


Fig. 3 Example of a carousel transmission.

Assuming that clients connect to the channel after the server starts sending content, a client will access to the carousel in a certain moment and will not be able to download the file until that client receives the FDT. So, the client needs to wait a certain time to receive the FDT, called waiting time (t_W). After receiving the FDT, the client needs a time to download the file (t_D). This way, the total download time needed to complete a download (t_T) is:

$$t_T = t_W + t_D. \quad (1)$$

Regarding the calculation of the download time (t_D), this is composed by three different times, as Fig. 3 shows. Firstly, there is a time t_M that indicates the time passed since the FDT is received until the client receives the first packet of the file to download. In that example, when the client accesses to the carousel, *F1* is being transmitted. Therefore, the client will have to wait until the server finishes the transmission of *F1* and then the server sends *F2* completely. Secondly, t_S indicates the time needed to send once the file to download, that is, the time passed from the transmission of the first packet until the transmission of the last one. As Fig. 3 shows, it is very likely that, if there are losses during the transmission, the client needs more than one carousel cycle to download a certain file. In the example, after the first cycle, the client has downloaded a 25% of the file, whereas after the second cycle, the client has downloaded 90%. The download is completed during the third cycle. Thus, the download time also depends on the number of cycles needed to complete the download. Therefore, the download time is calculated as:

$$t_D = t_M + t_S + t_C \cdot (n_{cycles} - 1). \quad (2)$$

Hence, the cycle time (t_C) has a great impact on the download time. The use of AL-FEC mechanisms allows to reduce the number of carousel cycles. In the case of considering AL-FEC, the code rate must be taken into account when calculating the cycle time:

$$t_C = \frac{\sum_{i=1}^{N} ceil\left(\frac{S_i}{CR_f}\right) + ceil\left(\frac{m \cdot e' \cdot \gamma}{CR_{FDT}}\right)}{b}, \quad (3)$$

where N is the number of files in the carousel; S_i is the size of the file i; m is the number of FDT Instances sent during a carousel cycle; e' is the size of an FDT Instance packet; γ is, as mentioned, the number of packets that compose an FDT Instance; CR_f and CR_{FDT} represent the code rate used to send the files and the FDT Instances, respectively; and b is the transmission rate. Note that S_i can be calculated as the product of the number of packets of the file i (n_i) and the size of FLUTE packets (e). Also, the use of AL-FEC encoding generates more packets. The total number of packets after coding is calculated by ceiling the division of the number of packets by the code rate. Assuming that both the files and the FDT will have the same protection (and thus the same code rate) and not considering the effect of the ceiling round, formula (3) is simplified:

$$t_C \cong \frac{\sum_{i=1}^{N} S_i + m \cdot e' \cdot \gamma}{b \cdot CR}. \quad (4)$$

Note that this formula and the following ones are valid when no FEC is used, simply assigning CR = 1.

Returning to (1), with regard to the waiting time (t_W), this is a random variable with expected value $E[t_W]$. In this study, it is assumed that the instant of time when the client accesses the carousel is uniformly distributed in the interval $[t, t+t_{FDT}]$, where t_{FDT} is the delivery period of the FDT Instances, that is:

$$t_{FDT} = \frac{t_C}{m}. \quad (5)$$

This way, the expected waiting time will be:

$$E[t_W] = \frac{t_C}{2m}. \quad (6)$$

The delivery period of FDT Instances is directly related with the parameter D_{FDT}, which indicates how many packets are sent between two FDT Instances:

$$D_{FDT} = \frac{\sum_{i=1}^{N} n_i + m \cdot \gamma}{m} \cong \frac{t_{FDT} \cdot e}{b}. \quad (7)$$

The approximation of the latter expression is due to the fact that the encoding symbol length of the data packets (e) and the FDT Instances (e') can be different.

Moreover, if there are losses in the transmission, it must be taken into account that FDT Instances can be lost, so several FDT cycles ($n_{cyclesFDT}$) –considering that an FDT cycle is the period between the delivery of two FDT Instances (t_{FDT})- can be necessary to obtain the FDT. Thus, formula (6) becomes:

$$E[t_W] = \frac{t_C}{2m} + \frac{t_C}{m} \cdot (n_{cyclesFDT} - 1). \quad (8)$$

Simplifying this expression yields:

$$E[t_W] = \frac{t_C}{m} \cdot \left(n_{cyclesFDT} - \frac{1}{2} \right). \quad (9)$$

As (9) shows, the expected value of the waiting time will decrease as the number of FDT Instances sent in the carousel is higher and as the number of cycles is lower.

In order to calculate this number of cycles, the methodology explained in [16] and [30] can be used. The first considers uniformly distributed losses whereas the latter models the channel losses using the Markov model. This paper considers bursty losses, thus the Markov model [31] is used. In this case, when FEC mechanisms are not employed, the expected number of new packets received in a certain cycle can be calculated using a hypergeometric distribution. Formula (10) calculates the probability of receiving *x* new encoding symbols for a block at a certain loop.

$$\mathrm{P}(x,v,n,l) = \frac{\binom{v}{x}\binom{n-v}{(n-l)-x}}{\binom{n}{n-l}}, \quad (10)$$

where *n* is the number of packets that make up a file or an FDT Instance; *l* is the number of lost packets per cycle; and *v* is the number of missing packets at the beginning of the cycle. The expectation value of the number of new packets at loop *i*, will be:

$$x(i) = \sum_{\xi=0}^{v} \xi \cdot \mathrm{P}(\xi, v, n, l). \quad (11)$$

The number of cycles needed to download a file or an FDT Instance can be estimated using (12). This formula is calculated through an iterative process, using Algorithm 1 and 2, explained later.

$$n_{cycles} = \min\{ j : \sum_{i=1}^{j} x(i) = n \}. \quad (12)$$

Similarly, if AL-FEC is used, the probability of receiving *x* new packets at any given loop is calculated using the next equation:

$$\mathrm{P}(x, n_{CR}, r, l) = \frac{\binom{n_{CR}-r}{x}\binom{r}{(n_{CR}-l)-x}}{\binom{n_{CR}}{n_{CR}-l}}, \quad (13)$$

where n_{CR} is the number of encoding symbols (source symbols plus parity symbols) after applying AL-FEC; *r* is the number of received symbols at the beginning of the loop; and *l* is the number of lost packets per loop. In this case, the expectation value is defined by:

$$x(i) = \sum_{\xi=0}^{n_{CR}-r} \xi \cdot \mathrm{P}(\xi, n_{CR}, r, l). \quad (14)$$

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

$$n_{cycles} = \min\{ j : \sum_{i=1}^{j} x(i) \geq n * inef\_ratio \}. \quad (15)$$

As in (10), n represents the number of packets that make up a file or an FDT Instance. The inefficiency ratio represents the relation between the number of packets needed to decode a file and the number of source packets that make up the file. The value of the inefficiency ratio depends on the coding used. The codes of type Minimum Distance Separable (MDS) have an inefficiency ratio equal to 1, whereas in the rest of codes this value is higher.

On the other hand, in regard to the calculation of the expected download time (t_D), as explained before this is composed by three terms: t_M, t_S and the number of cycles needed to complete the download.

$$E[t_D] = E[t_M] + E[t_S] + E[t_C \cdot (n_{cycles} - 1)] \tag{16}$$

Concerning t_M, it is possible to obtain an analytical expression considering m possible cases, each one corresponding to an FDT Instance in the file carousel. It is assumed that the client accesses the carousel at any instant $[t, t+t_C]$ and therefore, the m possible cases have equal probabilities, yielding to the following approximation:

$$E[t_M^L] \approx \frac{t_C}{2} - \frac{S'_L}{2b}, \tag{17}$$

where S'_L is the size of the file to download adjusted by the number of FDT Instances sent during the delivery of that file per carousel cycle. That adjustment is represented by the variable α. Also, S'_L is adjusted by the additional packets transmitted because of the use of AL-FEC. Thus,

$$S'_L = \frac{S_L \cdot \alpha}{CR}, \tag{18}$$

$$\alpha = 1 + \frac{m \cdot \gamma}{\sum_{i=1}^{N} n_i}. \tag{19}$$

Returning to (16), the calculation of t_S is more intuitive:

$$E[t_S^L] = \frac{S'_L}{b}. \tag{20}$$

On the other hand, in (16) the number of cycles is calculated using again formulas (10)-(15). This number of cycles has a great impact on the t_T and, as mentioned, is directly related to the losses in the transmission channel. Note that the client, in order to complete the download, needs actually an entire number of cycles (n'_{cycles}) minus a percentage of the last cycle (β_L), where β_L represents the download percentage referred to the file. In this sense, β is defined as the download percentage of the entire carousel. In the example of Fig. 3, the client needs 3 cycles to complete the download minus a little percentage of the last cycle.

$$n_{cycles} = n'_{cycles} - \beta. \tag{21}$$

Algorithm 1 and Algorithm 2 show how to calculate both n'_{cycles} and β, for the case of no FEC and AL-FEC, respectively. Both algorithms use the formulas presented in (10)-(15), with the following input parameters: the number of packets that make up a file (n), the percentage of channel losses and the maximum number of iterations used to calculate β. Also, Algorithm 2 receives as input parameters the number of packets of the file after decoding (n_{CR}) and the inefficiency ratio. The algorithm calculates the number of new packets received in each cycle and checks if enough packets have been received to rebuild the file. In that case, it adjusts the percentage of the last cycle that completes the download. This is calculated through an iterative process with a precision determined by the maximum number of iterations. For instance, with 10 iterations there is a precision smaller than 10^{-3} cycles.

Algorithm 1. No FEC.

INPUT: n, losses, max_iter
OUTPUT: n'_{cycles}, β_L

```
Initialize (n'cycles =1, βL=0, packets_received=0)
Calculate number of losses per cycle (l=n*losses) and
    number of packets received at first loop
while (not all packets have been received)
    n'cycles = n'cycles +1;
    Calculate new packets received (P) in the current
    loop using (11)
    if (packets_received+P=n)
        Initialize (bottom=0, top=1, thres=0, iter=1)
        while (iter<max_iter)
            thres =(bottom+top)/2
            Calculate new packets received (P) with
            (11) and input parameters:
             thres*(n,packets_received,l)
            if (packets_received+P=n)
                top= thres;
            else
                bottom= thres;
            endif
            iter=iter+1;
        endwhile
        βL=1-thres;
        BREAK
    else
        packets_received = packets_received+P;
    endif
endwhile
```

Algorithm 2. AL-FEC.

INPUT: n, n_{CR}, losses, inef_ratio, max_iter
OUTPUT: n'_{cycles}, β_L

```
Initialize (n'cycles =1, βL=0, packets_received=0)
Calculate number of losses per cycle (l=nCR*losses)
    and number of packets received at first loop
while (not enough packets have been received)
    n'cycles = n'cycles +1;
    Calculate new packets received (P) in the current
    loop using (14)
    if (packets_received+P≥n*inef_ratio)
        Initialize (bottom=0, top=1, thres=0, iter=1)
        while (iter<max_iter)
            thres =(bottom+top)/2
            Calculate new packets received (P) with
            (14) and input parameters:
            thres *(nCR,packets_received,l)
            if (packets_received+P≥n*inef_ratio)
                top= thres;
            else
                bottom= thres;
            endif
            iter=iter+1;
        endwhile
        βL=1-thres;
        BREAK
    else
        packets_received = packets_received+P;
    endif
endwhile
```

Both algorithms return the entire number of cycles (n'_{cycles}) and the percentage of the last cycle that completes the download referred to the file to download (β_L). In order to calculate the β referred to the entire carousel, formula (22) is used:

$$\beta = \beta_L \cdot \frac{S'_L \cdot CR}{\sum_{i=1}^{N} S_i + m \cdot e' \cdot \gamma}. \tag{22}$$

This way, taking into consideration expressions (17)-(22), the expected value of t_D of a file L is obtained by replacing in (16):

$$E\left[t_D^L\right] \approx \frac{t_C}{2} - \frac{S'_L}{2b} + \frac{S'_L}{b} + t_C \cdot \left(n'_{cycles} - \beta - 1\right). \tag{23}$$

That expression can be simplified:

$$E\left[t_D^L\right] \approx t_C \cdot \left(n'_{cycles} - \beta - \frac{1}{2}\right) + \frac{S_L \cdot \alpha}{2b \cdot CR}. \tag{24}$$

Finally, once the expected waiting time and the expected download time are calculated, the estimated value of the expected total download time of a file L is calculated replacing in (1):

$$E\left[t_T^L\right] \approx t_C \cdot \left(\frac{n_{cyclesFDT}}{m} + n'_{cycles} - \beta - \frac{1}{2} - \frac{1}{2m}\right) + \frac{S_L \cdot \alpha}{2b \cdot CR}. \tag{25}$$

For the sake of clarity, we sum up the main formulas presented. The expected download time of a certain file is calculated using (25), where t_C is obtained with formula (3) or (4), α is calculated with (19), and the number of cycles (both the file and the FDT) is obtained by means of Algorithm 1 (if No FEC is used) or by means of Algorithm 2 (if AL-FEC is used).

4.3 Partial FDT

The theoretical analysis presented in previous section assumes that the FDT sent by the server is complete, that is, it describes all files of the carousel. The calculation of the expected total download time of a file with partial FDT is very similar. The main difference regarding (25) is the value of m, which must be replaced by a partial m (m_L), which indicates how many FDT Instances describing file L are sent in each carousel cycle.

$$E\left[t_T^L\right] \approx t_C \cdot \left(\frac{n_{cyclesFDT}}{m_L} + n'_{cycles} - \beta - \frac{1}{2} - \frac{1}{2m_L} \right) + \frac{S_L \cdot \alpha_L}{2b \cdot CR}. \quad (26)$$

Supposing that each partial FDT only carries information of one certain file, the sum of all m_i must be equal to m:

$$m = \sum_{i=1}^{N} m_i. \quad (27)$$

Moreover, the value of α_L could be different depending on the way partial FDT Instances are sent. If they are sent throughout the carousel, α_L will be equal to expression (19), whereas if partial FDT Instances are only sent during the transmission of the file they describe, then α_L will be calculated using the next expression:

$$\alpha_L = 1 + \frac{m_L \cdot \gamma}{n_L}. \quad (28)$$

Note that in partial FDT the number of cycles needed to rebuild an FDT ($n_{cyclesFDT}$) will be lower or equal to the ones needed in the previous case since FDT Instances will be made up by less packets (in general, γ=1 if instances only describe one file).

4.4 Interleaving

On the other hand, server may interleave the packets belonging to different files, instead of using a sequential transmission. The waiting time is not affected by the transmission model (only by the transmission frequency of FDT Instances and the cycle time), so formula (9) remains valid. Nevertheless, when calculating the download time –formula (16)– both t_M and t_S change. The term referred to the number of cycles is equal. Specifically, since packets are interleaved, the time passed since the FDT is received until the first packet of the file to download is received, that is t_M, is reduced. When calculating t_M, considering that packets are interleaved according to a periodic sequence, it is taken into account that files with higher sizes will have a lower t_M. This time is calculated by multiplying the time needed to transmit a packet by the size of the whole carousel and divided by the size of the file to download. Thus, the expected t_M is:

$$E\left[t_M^L\right] = \frac{e}{b} \cdot \frac{\sum_{i=1}^{N} S_i + m \cdot e' \cdot \gamma}{2S'_L \cdot CR} = \frac{t_C}{2} \cdot \frac{e}{S'_L}. \quad (29)$$

On the contrary, the value of t_S increases considerably. This value is different depending on whether FEC is used or not. Without FEC, on average, the client must wait until almost the end of the carousel to get the last packet of the

file to download. Thus, following a similar reasoning that in formula (28), it is possible to calculate the expected value of t_S:

$$E\left[t_S^L\right] = t_C \cdot \frac{\frac{S'_L}{e} - 1}{\frac{S'_L}{e}}. \tag{30}$$

If AL-FEC is used, the value of t_S is in the range:

$$E\left[t_S^L\right] = \left[t_C \cdot \frac{e}{S'_L}(n \cdot inef\_ratio - 1), t_C \cdot \frac{e}{S'_L}(n_{CR} - 1) \right]. \tag{31}$$

The best case is when there are no losses, so the client has to receive *n*inef_ratio* packets to rebuild the file. As the losses increase, t_S increases too, with a maximum value of almost t_C. Thus, the total download time is obtained as in the general case, that is, considering t_W, t_M, t_S and the number of cycles necessary to complete the download. In any case, the download time using interleaved scheduling will always be higher than the one obtained using sequential transmission. Nevertheless, the use of interleaving could be very recommended in order to minimize the effect of burst losses.

4.5 Multiple download

Returning to the case of sequential scheduling, if the client decides to download all files of the carousel instead of only one, the total download time is calculated in a similar way. In this case, the waiting time would be the same that in the general case, calculated using formula (9). On the other hand, the term t_M disappears. In this case, the download time is determined by the file that takes longer to download. In general, this file will be the largest file in the carousel, because its download requires more carousel cycles. The expression of the total download time of the entire carousel is:

$$E\left[t_T\right] \approx \frac{t_C}{m} \cdot \left(n_{cyclesFDT} - \frac{1}{2} \right) + t_C \cdot (n'_{cycles} - \beta). \tag{32}$$

4.6 Prefetching

In this regard, it would be interesting to start storing packets before the FDT is received, in order to reduce the total download time. In this sense, the latest FLUTE RFC [2] indicates that, although generally a receiver needs to receive an FDT Instance describing a file before it is able to recover the file itself, when packets are received before the FDT, the system performance might be improved by caching such packets within a reasonable time window and storage size. Therefore, the client can save packets although the client does not know which file the packets belong to until the client receives the FDT. This can be useful in environments where the size of the data carousel is not very high and the client does not have limited resources or storage problems. In this case, the total download time can be reduced considerably. Regarding the waiting time, this will be the same that in the previous analysis, that is:

$$E\left[t_W\right] = \frac{t_C}{m} \cdot \left(n_{cyclesFDT} - \frac{1}{2} \right). \tag{33}$$

If the client starts saving data before the FDT is received, the expression of the download time does not change, considering that, in this situation, t_M represents the time passed since the client connect to the channel until the client receives the first packet of the file to download (although the client is storing all the packets that it receives), so the formula that defines the download time is:

$$E\left[t_D^L\right] \approx t_C \cdot \left(n'_{cycles} - \beta - \frac{1}{2} \right) + \frac{S_L \cdot \alpha}{2b \cdot CR}. \quad (34)$$

As clients need to receive the FDT to assign the properties to the file, the total download time will be specified by the waiting time if all packets are received before the FDT does, or by the download time otherwise:

$$E\left[t_T^L\right] \approx \max\{\frac{t_C}{m} \cdot \left(n_{cyclesFDT} - \frac{1}{2} \right), t_C \cdot \left(n'_{cycles} - \beta - \frac{1}{2} \right) + \frac{S_L \cdot \alpha}{2b \cdot CR}\}. \quad (35)$$

Finally, recall that all presented formulas remain valid when no FEC is used, by fixing the code rate to 1.

5. Evaluation

5.1 Methodology

This section presents the evaluation of the FDT transmission frequency, first for the general case (Section 5.2) and then for some particular cases. Specifically, Section 5.3 analyses the interleaving scheduling and Section 5.4 evaluates the advantages of using prefetching.

The main evaluation parameter is the total download time (t_T). This section shows various results for different values of the number of files in the carousel (N), different AL-FEC protection, and for different sizes and file distributions of packets (S_i).

The evaluation of the FDT transmission frequency has been done through different simulations, each comprised of as many iterations as needed to provide narrow (99%) confidence intervals. The studies consider different values of the number of times an FDT Instance is sent in each carousel cycle (that is, different values of m, which imply different values of D_{FDT}). Specifically, the values of m considered have been: 1, N, 2*N, 5*N, 10*N, 50*N and 100*N. These values of m provide the following approximated values of D_{FDT}: n_i*N, n_i, n_i/2, n_i/5, n_i/10, n_i/50 and n_i/100 packets respectively.

In the evaluation, a packet size (e) of 1428 bytes has been used, according to the results presented in [16], and taking into account the maximum transfer unit (MTU) of Ethernet (1500 bytes). With that size, and with typical file parameters in an FDT (such as content location, TOI or file size), it is assumed that one packet can contain over six file descriptions. This way, γ=ceil(N/6). Also, it is supposed that e'=e. Moreover, a transmission rate (b) of 5 Mb/s has been used, although the value of this parameter does not affect the conclusions of this study. Regarding channel losses, the different studies consider errors from 0 to 50% packet losses, in steps of 5%.

In relation to AL-FEC, it is considered that LDPC codes are used. These codes provide a good tradeoff between performance (download time) and complexity (time necessary to generate parity and time required to do the decoding process) [32]. Nevertheless, the studies presented here remain valid independently of the codes used. The only difference in the results presented is the value of the inefficiency ratio. In this case, it is considered a value of inefficiency ratio equal to 1.07, according to [30] and [33].

Finally, mention that in the studies the file L to download will always be the largest of the carousel.

5.2 General case

As a first study case, it is considered that a server sends a data carousel with N=100 files of n_i=1000 packets (that is, S_i=1 428 000 bytes). These values of N and n_i, according to (7), provide the following values of D_{FDT}: 100 000, 1000, 500, 200, 100, 20 and 10 packets.

Fig. 4 shows that there are high differences regarding the total download time depending on the FDT transmission frequency when no FEC is applied. These differences are higher as the losses increase. As the figure reflects, the values of m that provide minimum download times are N, $2N$ and $5N$ for all percentage of losses. In this case, the value of m=N is better since less traffic is generated in the network. On the other hand, the download time gets worse if a lot of FDT Instances (m=$50N$, $100N$) are sent. Furthermore, sending only one FDT Instance per cycle is not the best option either.

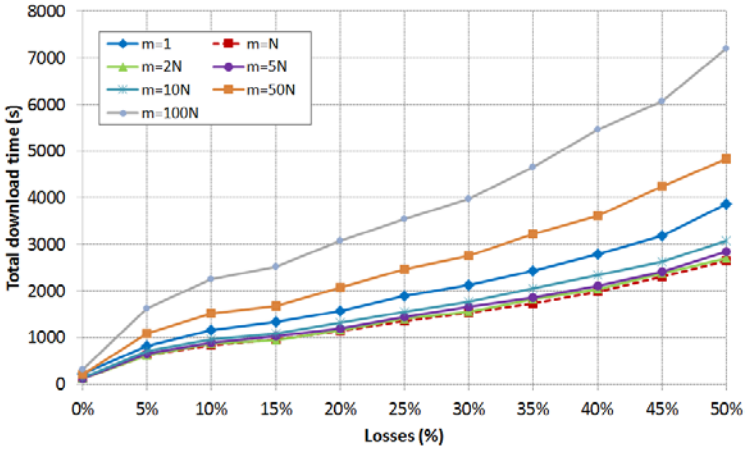
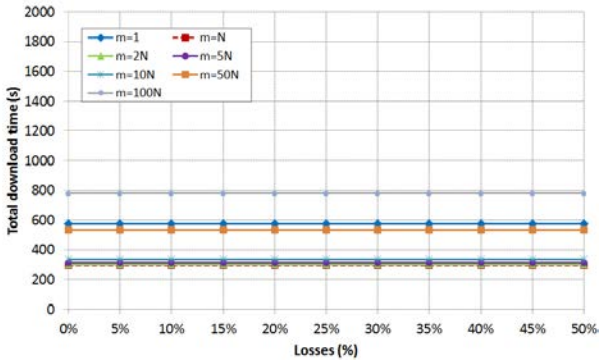


Fig. 4 Total download time evaluation for N=100 files and constant file size of n_i=1000 packets, without AL-FEC.

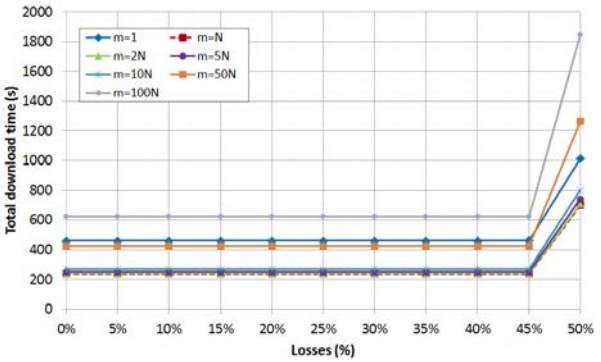
The system performance can be improved by using AL-FEC. In this sense, Fig. 5(a-f) show the total download time for different values of the code rate (CR). It should be noted that the scale of the total download time is different comparing with Fig. 4, as well as Fig. 5(f).

As expected, the use of AL-FEC reduces considerably the total download time. Fig. 5(a-f) reflect that, for any code rate, the total download time regarding the losses percentage has a stepped behavior. This is due to the fact that the AL-FEC features are homogeneous for a wide range of losses, as shown in [30]. Thus, when the amount of AL-FEC applied is not enough (high values of code rate) the download time starts increasing (for instance, for losses higher than 10% when the code rate is 0.8) because more carousel cycles are needed to complete the download. Although using a high code rate guarantees a good protection, this does not always entails a minimum value of download time, since using too much parity produces that a lot of parity packets are sent, thus reducing the efficiency of the transmission. Furthermore, a high protection increases considerably the bandwidth. In this example, as the figures show, a code rate that provides a good tradeoff between total download time and bandwidth is 0.7 – Fig. 5(d)-, so this code rate will be used in the rest of studies.

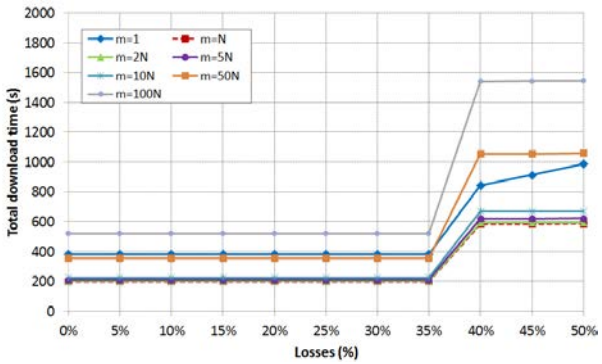
Regarding the FDT frequency, as in Fig. 4, values of m around N provide minimum download times for all percentage of losses and code rates, whereas sending only one FDT Instance or a lot of FDT Instances is not a good solution. It can also be noted that the effect of the FDT delivery frequency in the total download time is independent of the AL-FEC parity.



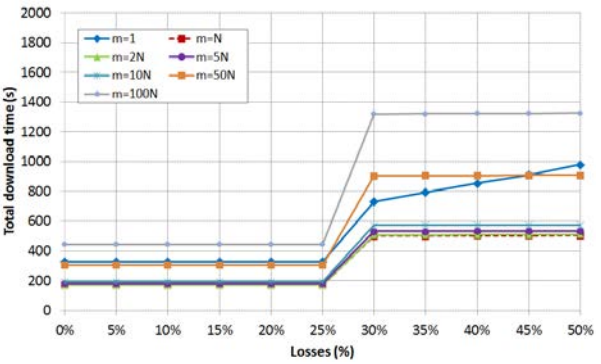
(a) CR=0.4



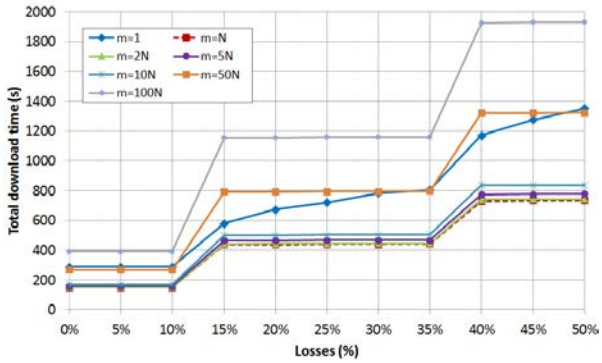
(b) CR=0.5



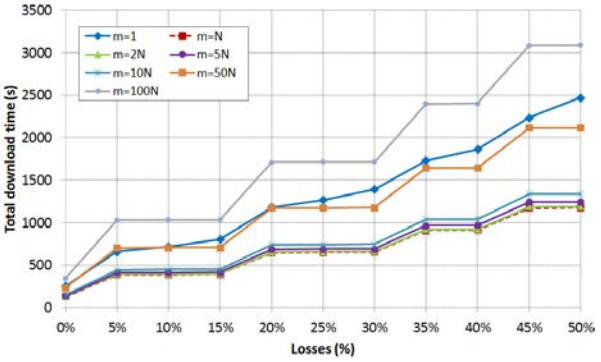
(c) CR=0.6



(d) CR=0.7



(e) CR=0.8

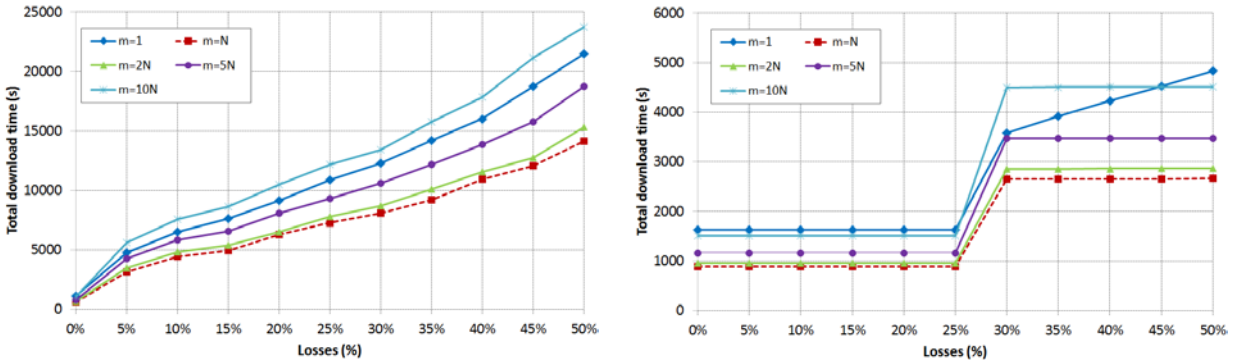


(f) CR=0.9

Fig. 5 Total download time evaluation for N=100 files, constant file size of n_i=1000 packets for different values of code rate.

5.2.1 Impact of the number of files in the carousel

Regarding the number of files in the carousel, previous conclusions are confirmed in the studies carried out with low and high values of N. The differences between the different values of m are higher as the number of files in the carousel increases. To see this, Fig. 6(a-b) show the behavior of the FDT when N=500 files. The figures show higher differences between the different values of m (note that high values of m are not shown, which are considerably higher than the rest). Once again, values of m=N and 2N provide minimum download times for all channel losses, whether AL-FEC is applied or not.



(a) No FEC **(b)** AL-FEC (CR=0.7)

Fig. 6 Total download time evaluation for N=500 files and constant file size of n_i=1000 packets.

Table 2 and Table 3 show the difference between the total download time obtained with the optimum value (m=N) and the rest of values, for different values of N and percentage of losses, without using FEC and using AL-FEC respectively. Specifically, tables show how the total download time increases (in percentage) regarding the optimum value. These percentages are calculated by dividing the total download time obtained using a certain value of m by the total download time obtained with m=N. Note that negative values indicate that a certain configuration provides lower download times than the configuration with m=N.

	***N*=10**			***N*=100**			***N*=500**		
	losses			**losses**			**losses**		
m	**5%**	**25%**	**50%**	**5%**	**25%**	**50%**	**5%**	**25%**	**50%**
1	22%	14%	14%	34%	40%	46%	51%	50%	52%
2N	1%	-3%	0%	2%	3%	2%	10%	7%	8%
5N	2%	-1%	-1%	6%	7%	7%	36%	28%	32%
10N	-1%	2%	-1%	16%	15%	16%	78%	68%	68%
50N	9%	5%	7%	78%	82%	83%	404%	377%	384%
100N	17%	15%	17%	164%	161%	172%	790%	754%	789%

Table 2. Total download time percentage referred to the case of sending m=N FDT Instances per carousel cycle for files of constant size of n_i=1000 packets without AL-FEC.

	***N*=10**			***N*=100**			***N*=500**		
	losses			**losses**			**losses**		
m	**5%**	**25%**	**50%**	**5%**	**25%**	**50%**	**5%**	**25%**	**50%**
1	78%	179%	99%	94%	94%	94%	84%	84%	82%
2N	-5%	-9%	-7%	1%	1%	1%	8%	8%	8%
5N	-7%	-13%	-9%	6%	6%	6%	31%	31%	31%
10N	-7%	-13%	-10%	14%	14%	14%	69%	69%	69%
50N	-1%	-8%	-4%	80%	80%	80%	379%	379%	379%
100N	8%	0%	5%	163%	163%	163%	765%	765%	765%

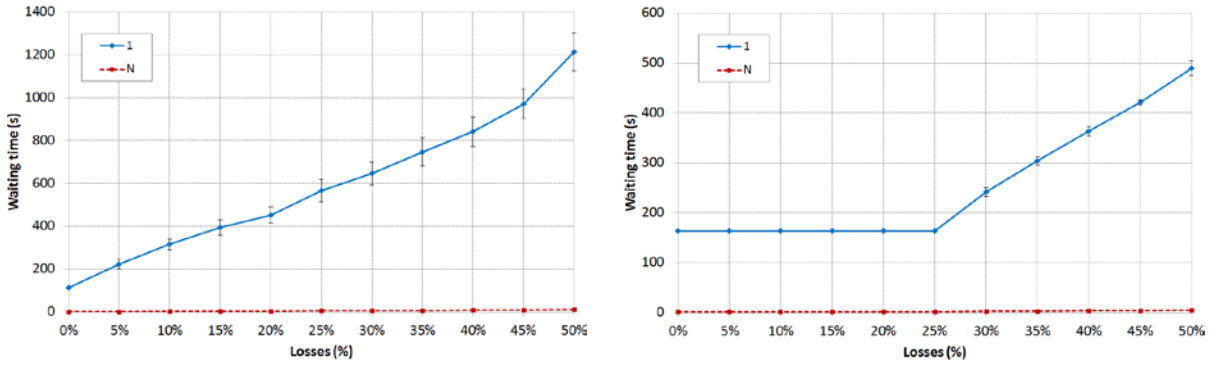
Table 3. Total download time percentage referred to the case of sending m=N FDT Instances per carousel cycle for files of constant size of n_i=1000 packets with AL-FEC.

Tables show that for all values of N and percentage of losses the total download time with m=2N is almost optimum or even optimum in some cases. Both tables show that the difference regarding the optimum value is similar for all percentage of losses. Nevertheless, as the number of files in the carousel increases, the differences with the optimum value increase drastically. For example, in Tables 2 and 3 for N=500 files and m=10N (that is,

sending an FDT Instance each 100 packets) the total download time obtained is over 70% higher than the one obtained with m=N (that is, sending an FDT Instance each 1000 packets). Moreover, as mentioned, sending FDT Instances more frequently increases the total download time considerably, as values of m=50N and m=100N show. Also, sending only one FDT per cycle provides values of total download time rather higher.

5.2.2 Impact of the waiting time

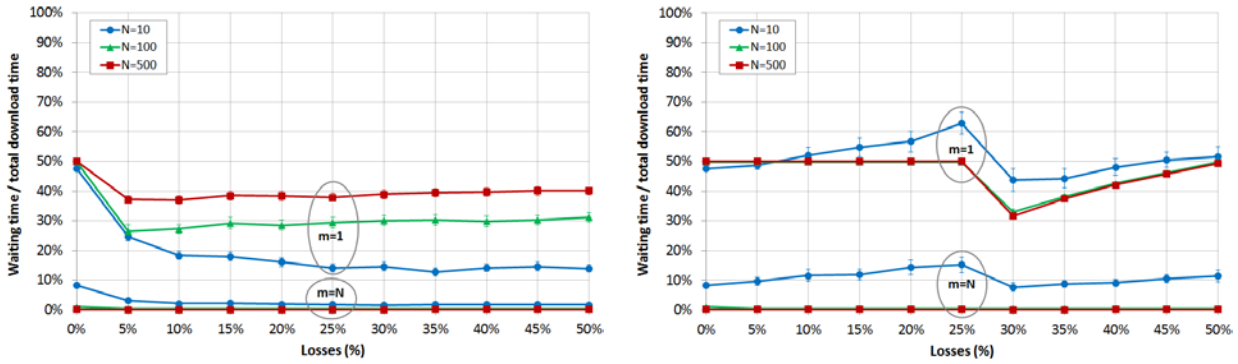
On the other hand, as mentioned in the theoretical analysis, there are two variables that make up the total download time (t_T): the waiting time (t_W) and the download time of the file (t_D). According to the theoretical analysis, the number of FDT Instances sent affects mainly the waiting time. Fig. 7(a-b) show the waiting time for only two values of m (m=1 and m=N), since values higher than m=N provide values very close to 0. Results are very clear: it is enough to send only N FDT Instances per cycle so as to minimize the waiting time.



(a) No FEC **(b)** AL-FEC (CR=0.7)

Fig. 7 Waiting time evaluation for N=100 files and constant file size of n_i=1000 packets.

In this sense, a related study is presented in Fig. 8(a-b), where it is shown how the waiting time affects the total download time. Specifically, the figures depict the percentage of t_W regarding t_T for different values of N. Also, the same two values of m of the previous figure are shown. As m is higher, the waiting time hardly affects the total download time. In contrast, when only one FDT is sent in each carousel cycle, the waiting time has a great impact on the total download time.



(a) No FEC **(b)** AL-FEC (CR=0.7)

Fig. 8 t_W/t_T relation evaluation for m=1 and m=N.

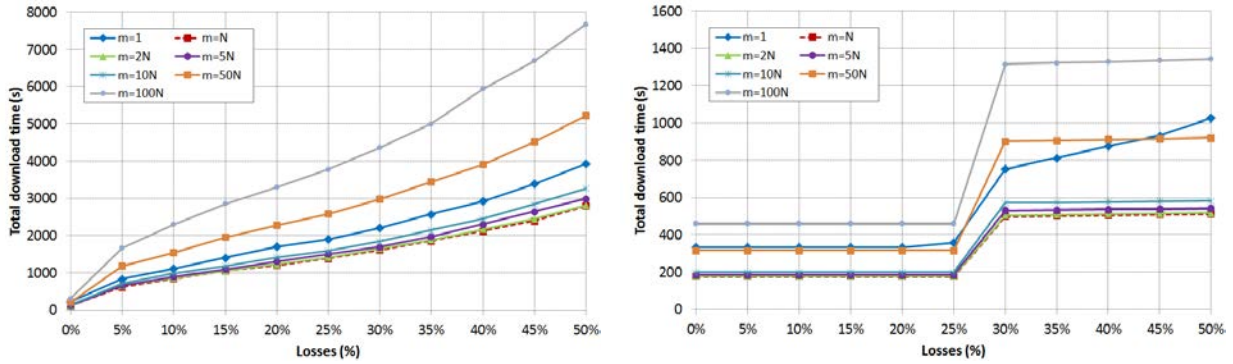
When no FEC is used, the percentage of t_W regarding t_T is higher as the number of files in the carousel increases if only one FDT Instance is sent. For instance, when N=500, t_W represents over 40% of the t_T. Moreover, Fig. 8(a) reflects that, for a fixed value of N, the percentage remains very similar independently of the losses, except in the

case of no losses. In contrast, if AL-FEC is used the waiting time has more importance on the total download time when the FDT is not sent very frequently. Fig. 8(b) shows how the relation t_W/t_T varies depending on the losses. From a certain percentage of losses, the relation t_W/t_T decreases and then increases again. This percentage of losses is delimited by the code rate used. In Fig. 8(b), this percentage is over a 25%, a result coherent with Fig. 5(d) and Fig. 6(b).

For example, if 100 files are sent in a channel with 10% losses and only one FDT per cycle carousel is sent, the waiting time represents 25% of the total download time if no FEC is used, whereas this percentage increases up to 50% when AL-FEC is used. On the contrary, if *N* FDTs are sent, the waiting time represents only the 0.5% of the total download time without FEC and the 1% with AL-FEC. This percentage is similar in all cases studied in this section.

5.2.3 Impact of the file size distribution

Regarding the file size distribution, normally the files of the carousel will not have the same size. Fig. 9(a-b) consider a carousel where the size of the files follows a lognormal distribution which mean is 1000 packets. Regarding the comparison between different values of *m*, results are very similar to those shown in Fig. 4 and Fig. 5(d). In addition, as Fig. 9(a) shows, a carousel which file size follows a log normal distribution provides slightly higher download times when no FEC is used. Nevertheless, when AL-FEC is used –Fig. 9(b), the total download time obtained using a log normal file size distribution is almost the same as the one obtained when files have the same size. Again, it should be noted the different scale of Fig. 9(a) and Fig. 9(b).

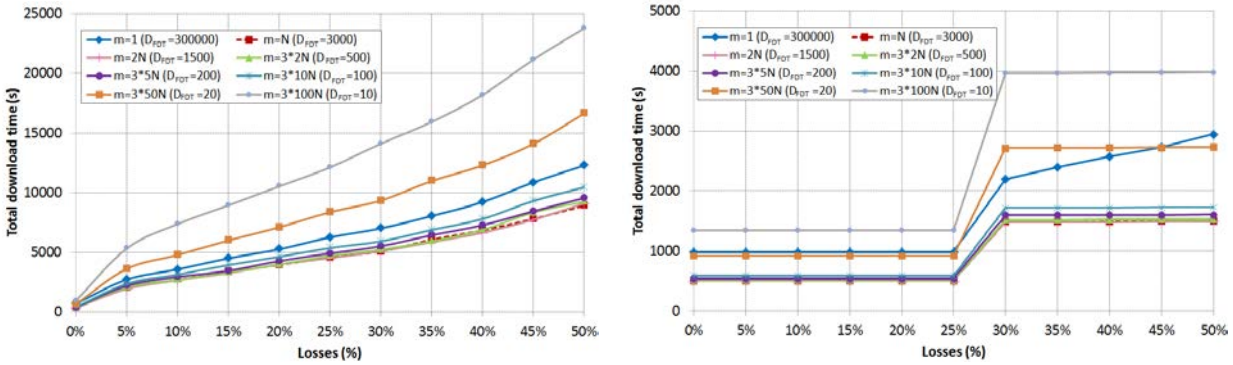


(a) No FEC **(b)** AL-FEC (CR=0.7)

Fig. 9 Total download time evaluation for *N*=100 files and log normal file size distribution with mean=1000 packets.

5.2.4 Impact of the file size

Next study considers a carousel which files are 3 times larger than in previous studies, that is, files of 3000 packets (over 4 Mbytes). In this study, the values of *m* have been modified in order to maintain the same values of D_{FDT} of the previous studies. Thus, the values of *m*: 1, *N*, 2*N*, 3*2*N*, 3*5*N*, 3*10*N*, 3*50*N* and 3*100*N* mean sending the FDT Instances every approximately 300 000, 3000, 1500, 500, 200, 100, 20 and 10 packets respectively. Fig. 10(a-b) show that the values that minimize the download time for all percentages of losses are *m*=*N* (D_{FDT}=3000 packets), *m*=2*N* (D_{FDT}=1500 packets) and *m*=3*2*N* (D_{FDT}=500 packets). These are also the optimum values in a carousel in which the file size of the carousel follows a log normal distribution (although they are not shown in the paper). Therefore, the conclusions are very similar to those reached in the previous studies.



(a) No FEC **(b)** AL-FEC (CR=0.7)

Fig. 10 Total download time evaluation for N=100 files and constant file size of n_i=3000 packets.

5.3 Interleaving

On the other hand, next study analyzes the effect of the scheduling model. Table 4 shows the increase (in percentage) of the total download time using interleaving with respect to using sequential scheduling for different percentage of losses, number of FDT Instances sent and using AL-FEC or not. These percentages are calculated as in Tables 2 and 3. The increase of the total download time using interleaving is especially important, apart from the case of no losses without FEC, when AL-FEC is used. In these cases, the difference between sequential and interleaved transmission is approximately half cycle. For instance, in the case of AL-FEC with 25% of losses, it is necessary only one cycle to download the file but, whereas in the sequential mode the download is completed after approximately half cycle, the interleaved transmission requires almost the complete cycle, thus the difference regarding the total download time is almost the double comparing these two models. This fact and the rest of results of the table are coherent with the formulas previously presented. Independently of the number of FDT Instances sent during the carousel, the interleaving model always increases the total download time.

	No FEC				**AL-FEC**			
	losses				**losses**			
m	**0%**	**5%**	**25%**	**50%**	**0%**	**5%**	**25%**	**50%**
1	92%	29%	10%	4%	24%	28%	49%	17%
N	181%	38%	14%	8%	48%	56%	97%	33%
2N	182%	36%	15%	9%	48%	56%	97%	33%
5N	183%	34%	18%	7%	48%	56%	98%	33%
10N	183%	35%	15%	7%	48%	56%	98%	33%
50N	183%	38%	16%	6%	48%	56%	98%	33%
100N	183%	37%	18%	6%	48%	56%	98%	33%

Table 4. Relation between the total download time obtained with interleaving regarding the total download time obtained with sequential scheduling. N=100 files and constant file size of n_i=1000 packets.

5.4 Prefetching

Finally, it is considered the case of storing packets before the FDT is received. It may appear that this could reduce considerably the total download time but, according to Fig. 7(a-b) and formula (33), this assumption is only true when few FDT Instances are sent. For example, if only one FDT is sent, saving packets before the FDT is received can reduce the total download time almost a 50% for different percentages of losses. Nevertheless, the

storage space required by clients could increase by a factor equal to the number of files in the carousel, in this case by 100.

6. Conclusions

This paper has shown the great influence that the FDT transmission frequency has over the total download time of a file. A proper configuration of the FDT transmission frequency increases the bandwidth efficiency, thus improving the Quality of Experience perceived by the users. In this sense, there are certain configurations of the FDT that reduce the total download time. The results show that there is a range of values of the FDT frequency delivery that minimizes the total download time. Sending as many FDT Instances as files (N) or as the double of files ($2N$) in the carousel per cycle provide very good results, whether AL-FEC is used or not. However, if only one FDT Instance per cycle is sent, the time clients wait until they receive the FDT increases considerably, therefore the total download time increases too. Also, if several FDT Instances are sent the total download time also gets worse considerably. The difference among the total download time obtained using an optimum FDT configuration and other configurations is higher as the number of files in the carousel and its size is higher, and as the amount of losses in the channel increases.

Obviously the use of AL-FEC mechanisms improves considerably the total download time. In this sense, it is important to consider the percentage of losses of the transmission channel, in order to use an appropriate code rate that provides a good value of the total download time without increasing excessively the channel bandwidth.

Among the two parameters that make up the total download time, the waiting time has an important impact on the total download time only when few FDT Instances per cycle are sent. Sending more FDT Instances allows to minimize the waiting time. In this sense, it is not a great advantage to save packets before the FDT is received. In doing this, the waiting time would disappear, but as this waiting time would be very low with a proper value of FDT frequency, the profit would be minimal, at the expense of increasing extensively the memory and computational resources in reception.

On the other hand, although interleaved scheduling can be very useful to minimize the burst losses, sequential transmission always reduces the total download time.

To sum up, a proper configuration of the AL-FEC protection and the FDT transmission frequency provides an optimum content delivery through the FLUTE protocol.

References

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

[2] T. Paila, R. Walsh, M. Luby, V. Roca, R. Lehtonen, FLUTE – File Delivery Over Unidirectional Transport, IETF RFC 6726, 2012.

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

[4] ETSI TS 102 034, Transport of MPEG-2 TS Based DVB Services over IP based Networks (and associated XML), v1.4.1, 2008.

[5] ETSI TS 126 346, Universal Mobile Telecommunications System (UMTS); LTE; Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs (release 10), v11.3.0, 2013.

[6] D. Lecompte, F. Gabin, Evolved Multimedia Broadcast/Multicast Service (eMBMS) in LTE-Advanced: Overview and Rel-11 Enhancements, IEEE Communications Magazine, 50-11 (2012), 68-74.

[7] C. Neumann, V. Roca, R. Walsh, Large scale content distribution protocols, ACM Computer Communication Review, 35-5 (2005), 85-92.

[8] F. Fraile, I. de Fez, J. C. Guerri, Modela-TV: service personalization and business model management for mobile TV, in 7th European Interactive TV Conference (EuroITV), Leuven, Belgium, 2009.

[9] C-C. Hsieh, C-H. Lin, W-T. Chang, Design and implementation of the interactive multimedia broadcasting services in DVB-H, IEEE Transactions on Consumer Electronics, 55-4 (2009), 1779-1787.

[10] J. Peltotalo, J. Harju, M. Saukko, L. Väätämöinen, I. Bouazizi, I. Curcio, Personal mobile broadcasting based on the 3GPP MBMS System, in Proc. of MoMM, Linz, Austria, 2008.

[11] L. Liang, H. Cruichkshank, Z. Sun, C. Kulatunga, G. Fairhurst, The integration of TESLA and FLUTE over satellite networks, in Proc. of the IEEE Global Telecommunications Conference (Globecom), Miami (USA), 2010.

[12] D. Kutscher, J. Greifenberg, K. Loos, Scalable DTN distribution over uni-directional links, in Proc. of the SIGCOMM workshop on networked systems in developing regions (NSDR), Kyoto, Japan, 2007.

[13] G. Papastergiou, I. Psaras, V. Tsaoussidis, Deep-Space Transport Protocol: A novel transport scheme for Space DTNs, Computer Communications, 32-16 (2009), 1757-1767.

[14] J. Hrvoje, T. Stockhammer, W. Xu, W. Abdel Samad, Efficient video-on-demand services over mobile datacast channels, Journal of Zhejiang University, 7-5 (2006), 873-884.

[15] Z. Yetgin, T. Çelik, Efficient progressive downloading over multimedia broadcast multicast service, Computer Networks, 56-2 (2012), 533-547.

[16] J. Peltotalo, S. Peltotalo, J. Harju, R. Walsh, Performance analysis of a file delivery system based on the FLUTE protocol, International Journal of Communication Systems, 20-6 (2007), 633-659.

[17] MAD-FCL, MAD Project's home page. Online, available: http://mad.cs.tut.fi.

[18] J. Peltotalo, J. Harju, M. M. Hannuksela, Reliable, server-friendly and bandwidth-efficient file delivery system using FLUTE server file format, in Proc. IEEE Int. Symposium on BMSB, Bilbao, Spain, 2009.

[19] C. Neumann, V. Roca, Impacts of the startup behavior of multilayered multicast congestion control protocols on the performance of content delivery protocols, in Int. Workshop on WCW, Sophia Antipolis, France, 2005.

[20] M. Luby, M.Watson, L. Vicisano, Asynchronous Layered Coding (ALC) Protocol Instantiation, IEFT RFC 5775, 2010.

[21] M. Luby, M. Watson, L. Vicisano, Layered Coding Transport (LCT) Building Block, IEFT RFC 5651, 2009.

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

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

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

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

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

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

[28] M. Handley, V. Jacobson, C. Perkins, SDP: Session Description Protocol, IETF RFC 4566, 2006.

[29] R. Walsh, I. Curcio, J. Peltotalo, S. Peltotalo, H. Mehta, SDP descriptors for FLUTE, IETF Draft, 2012.
[30] I. de Fez, F. Fraile, R. Belda, J. C. Guerri, Analysis and evaluation of adaptive LDPC AL-FEC codes for content download services, IEEE Transactions on Multimedia, 14-3 (2012), 641-650.
[31] H. Bai, M. Atiquzzaman, Error modeling schemes for fading channels in wireless communications: a survey, IEEE Comm. Surveys Tuts., 5-2 (2003), 2-9.
[32] E. Paolini, M. Varella, M. Chiani, B. Matruz, G. Liva, Low-complexity LDPC codes with near-optimum performance over the BEC, in Proc. Adv. Satellite Mobile Syst. (ASMS), Bologna, Italy, 2008, pp. 274-282.
[33] V. Roca, C. Neumann, Design, evaluation and comparison of four large block FEC codecs, LDPC, LDGM, LDGM staircase and LDGM triangle, plus a Reed-Solomon small block FEC codec, INRIA Research Report, 2004.