EVALUATION OF UNIDIRECTIONAL BACKGROUND PUSH CONTENT DOWNLOAD SERVICES FOR THE DELIVERY OF TELEVISION PROGRAMS

Author: Francisco Fraile Gil

Supervisor: Juan Carlos Guerri Cebollada

Departamento de Comunicaciones,

Universitat Politécnica de València

Abstract

This thesis dissertation presents background push Content Download Services as an efficient mechanism to deliver pre-produced television content through existing broadcast networks. Nowadays, network operators dedicate a considerable amount of network resources to live streaming live, through both broadcast and unicast connections. This service offering responds solely to commercial requirements: Content must be available anytime and anywhere. However, from a strictly academic point of view, live streaming is only a requirement for live content and not for pre-produced content. Moreover, broadcasting is only efficient when the content is sufficiently popular.

The services under study in this thesis use residual capacity in broadcast networks to push popular, pre-produced content to storage capacity in customer premises equipment. The proposal responds only to efficiency requirements. On one hand, it creates value from network resources otherwise unused. On the other hand, it delivers popular pre-produced content in the most efficient way: through broadcast download services.

The results include models for the popularity and the duration of television content, valuable for any research work dealing with file-based delivery of television content. Later, the thesis evaluates the residual capacity available in broadcast networks through empirical studies. These results are used in simulations to evaluate the performance of background push content download services in different scenarios and for different applications. The evaluation proves that this kind of services can become a great asset for the delivery of television content.

Resumen

Este trabajo de tesis presenta los servicios de descarga de contenido en modo push como un mecanismo eficiente para el envío de contenido de televisión pre-producido sobre redes de difusión. Hoy en día, los operadores de red dedican una cantidad considerable de recursos de red a la entrega en vivo de contenido televisivo, tanto sobre redes de difusión como sobre conexiones unidireccionales. Esta oferta de servicios responde únicamente a requisitos comerciales: disponer de los contenidos televisivos en cualquier momento y lugar. Sin embargo, desde un punto de vista estrictamente académico, el envío en vivo es únicamente un requerimiento para el contenido en vivo, no para contenidos que ya han sido producidos con anterioridad a su emisión. Más aún, la difusión es solo eficiente cuando el contenido es suficientemente popular.

Los servicios bajo estudio en esta tesis utilizan capacidad residual en redes de difusión para enviar contenido pre-producido para que se almacene en los equipos de usuario. La propuesta se justifica únicamente por su eficiencia. Por un lado, genera valor de recursos de red que no se aprovecharían de otra manera. Por otro lado, realiza la entrega de contenidos pre-producidos y populares de la manera más eficiente: sobre servicios de descarga de contenidos en difusión.

Los resultados incluyen modelos para la popularidad y la duración de contenidos, valiosos para cualquier trabajo de investigación basados en la entrega de contenidos televisivos. Además, la tesis evalúa la capacidad residual disponible en redes de difusión, por medio de estudios empíricos. Después, estos resultados son utilizados en simulaciones que evalúan las prestaciones de los servicios propuestos en escenarios diferentes y para aplicaciones diferentes. La evaluación demuestra que este tipo de servicios son un recurso muy útil para la entrega de contenido televisivo.

Resum

Aquest treball de tesi presenta els serveis de descàrrega de contingut en mode ***push*** com un mecanisme eficient per a l'enviament de continguts de televisió produïts amb anterioritat sobre xarxes de difusió. Hui en dia, els operadors de xarxes dediquen una quantitat considerable de recursos de xarxa al lliurament en viu de contingut televisiu, tant sobre xarxes de difusió com sobre connexions unidireccionals. Aquesta oferta de serveis respon únicament a requisits comercials: disposar dels continguts televisius en qualsevol moment i lloc. No obstant, des d'un punt de vista estrictament acadèmic, l'enviament en viu és únicament un requeriment per al contingut en viu, no per a continguts que ja han estat produïts amb anterioritat a la seua emissió. Més encara, la difusió no més és eficient quan el contingut és suficientment popular.

Els serveis baix estudi en aquesta tesi utilitzen capacitat residual en xarxes de difusió per enviar contingut produït amb anterioritat per a que s'emmagatzeme en els equips de l'usuari. La proposta es justifica únicament per la seua eficiència. Per una banda, genera valor de recursos de xarxa que no s'aprofitarien d'altra forma. Per altra banda, realitza el lliurament de continguts produïts amb anterioritat i populars del mode més eficient: sobre serveis de descàrrega de continguts en difusió.

Els resultats inclouen models per a la popularitat i la duració de continguts, valuosos per a qualsevol treball d'investigació basats en el lliurament de continguts televisius. A més, la tesi avalua la capacitat residual disponible en xarxes de difusió, mitjançant estudis empírics. Després, aquests resultats són utilitzats en simulacions que avaluen les prestacions dels serveis proposats en escenaris diferents i per a aplicacions diferents. L'avaluació demostra que aquest tipus de serveis són un recurs molt útil pel lliurament de contingut televisiu.

Acknowledgement

I would like to express my gratitude to my supervisor, professor Juan Carlos Guerri Cebollada, for all the interest and the support that he has provided through the years. This is extensible to all the members of the Multimedia Communication Group and my former colleagues at Interactive TV Arena, which have dedicated a great deal of resources to making this thesis possible. I would like to thank specially Ismael de Fez for all the feedback and the help provided in the writing and the simulations, and Pau Arce and Román Belda for their work in the development of the applications needed to perform the measurements.

I also have to thank all the industrial partners that have participated in the different projects that have financed this thesis work. Thanks Vicente Plá, from Televisió Autonòmica Valenciana, for all the feedback necessary to make the thesis results relevant from an industrial point of view.

I also would like to thank my family for their unconditional support. The pride in the eyes of my parents, Manuel and Pascuala, has been the strongest encouragement for developing this thesis. Also, the feedback from my sister, Celia, her husband, Francisco, and my uncle Miguel Angel, has been really helpful. Thanks also to all my friends in Jumilla, Valencia and Gävle, for being so supportive during these years.

To my brother Manuel in loving memory.

Contents

List of Tables

List of Figures

Acronyms

3DTV	Three-dimensional Television
AL-FEC	Application Layer Forward Error Correction
API	Application Programming Interface
AWG	Arbitrary Waveform Generator
CAPEX	Capital Expenditures
CBR	Constant Bit Rate
CCDF	Complementary Cumulative Density Function
CDF	Cumulative Density Function
CDN	Content Delivery Network
CDS	Content Download Service
CNR	Carrier to Noise Ratio
DPLN	Double Pareto Lognormal
DVB	Digital Video Broadcasting
ES	Elementary Stream
FDT	File Delivery Table
FFQ	Fluid Fair Queuing
FLUTE	File Delivery over Unidirectional Transport
GOP	Group of Pictures
HD	High Definition
HP	High Priority
IMDB	Internet Movie Database
IPTV	Internet Protocol Television
LN	Lognormal
LP	Low Priority
LPF	Low Pass Filter
MIP	Mega-frame Information Packet
MLE	Maximum Likelihood Estimates
MME	Method of Moments Estimates
MPE	Multi-Protocol Encapsulation
MPEG-TS	Moving Pictures Expert Group – Transport Stream
ODI	Opportunistic Data Insertion
OFDM	Orthogonal Frequency Domain Multiplexing
OPEX	Operational Expenditures
P2P	Peer to Peer

PCR	Program Clock Rate
PDF	Probability Density Function
PLL	Phased Lock Loop
PLP	Physical Layer Pipe
PSI/SI	Program Specific Information / Service Information
QAM	Quadrature Amplitude Modulation
QoE	Quality of Experience
QoS	Quality of Service
QP	Quantifier Parameter
QPSK	Quadrature Phase Shift Keying
RTP	Real-time Transport Protocol
SD	Standard Definition
T-STD	Transport – System Target Decoder
TDM	Time Domain Multiplexing
TOI	Transport Object Identifier
TSI	Transport Session Identifier
UDP	User Datagram Protocol
UGC	User Generated Content
VBR	Variable Bit Rate
VNI	Visual Networking Index
VoD	Video on Demand

Notation

b_{CDS}	bitrate of Content Download Service
b_s	burst size
b_{VBR}	VBR encoding bitrate
c	number of cycles
c_{ts}	time slicing compression parameter
f_{psize}	packetized frame size
f_{size}	frame size
I^k	knapsack algorithm order in iteration k
k	transmitted packets
k_3	third order cumulant
k_4	fourth order cumulant
l	lost packets
m	missing packets
n	number of encoded symbols
p_e	error rate
p_j	probability of access of file j
P_N	popularity ranking function
r	correctly received symbols
s	duration sample
s_A	storage size
s_j	size of file with index j
t_A	access time
t_C	carousel time
t_D	download time
v_j	value of file with index j
α	parameter of power law distribution
β	parameter 2 of DPLN distribution
χ	parameter of the exponential cutoff component
Γ	Gamma function
δ	parameter 1 of DPLN distribution
Δt	real time between two consecutive timestamps
Δt_B	time calculated with byte count between two consecutive timestamps
Δt_{PCR}	time difference between two consecutive timestamps
ε	error
λ	requests per unit of time

μ	scale of distribution
σ	location of distribution
τ	parameter 4 of DPLN distribution
υ	degree of T-location distribution
υ	parameter 3 of DPLN distribution

Chapter I

Introduction

The main purpose of this initial chapter is to describe the objectives and the methodology of the thesis. The chapter starts with a classification of the most widely used television content delivery technologies. This classification is included to provide a better understanding of the purpose and characteristics of unidirectional background push content download services. Later, the chapter includes a description of the proposed service, presenting its architecture and describing its main features in order to facilitate the presentation of the objectives and the methodology.

I.1 Classification of television content delivery technologies

Traditionally, television service providers have used dedicated broadcast platforms for the delivery of television services. These broadcast platforms -satellite, cable or terrestrial- migrated to digital technology to cope with the increasing requirements for quality, content offer and interactivity. In the mean time, IP networks have emerged as a delivery channel for video services, fostering the appearance of different IP based video delivery technologies.

Due to these developments, nowadays television service providers have many different options to deliver their content. Clearly, choosing the best technology depends on different aspects and it is important to classify them according to their characteristics. A simple

classification can be made based on three characteristics: Infrastructure requirements, delivery method and service type.

First, content delivery technologies can be classified according to the infrastructure resources they require. There are content delivery technologies that use dedicated infrastructure and network resources, e.g. Digital Video Broadcast (DVB) or IP Television (IPTV) technologies. Similarly, other delivery technologies use dedicated infrastructure, but share network resources with other services. This is the case of video technologies based on Content Delivery Networks (CDNs) over the Internet. Lastly, other technologies use neither dedicated infrastructure nor network resources, as television services delivered over Peer-to-Peer (P2P) networks.

A second characteristic to take into consideration is the delivery method used – unicast, multicast or broadcast. This way, unicast technologies send the video to each viewer on a separate IP connection. Oppositely, broadcast technologies establish a single connection from the source to all hosts connected to the network, regardless if the video is being watched or not at any particular host. Moreover, multicast technologies use a single connection, as in broadcast, but send the video only to interested viewers, as in unicast.

The third characteristic of content delivery technologies regarded in this simple classification is the type of service, that is, whether they are based on content download or streaming services. Content download consists of sending the programs to the users as files. The video is presented after the download process is completed. Alternatively, video streaming technologies maintain a constant delay between the ingestion and the consumption of every video sample, so that the presentation of the video can be initiated at any point of the video timeline, without downloading the entire video. In order to compensate for non-constant delays (e.g. network delay or decoding delay), it is necessary to introduce intermediate buffers. The size of these buffers should be sufficiently high to guarantee a constant delay, but at the same time, it should be kept as small as possible to minimize end-to-end delay. An alternative between streaming and content download is pseudo-streaming, where the presentation can start at any point of the video timeline as in

streaming, except that before starting the playback, it is necessary to download a piece of the video file as in content download.

Figure 1 shows this classification applied to popular technologies used to deliver television content. The figure also highlights two commercial stages in the life cycle of television content. The first phase, *content released* represents the time when the content is (first) aired in a television channel. The second phase, *content downloadable* represents the time when the content is available for download to viewers. As shown in the figure, there are different technologies used in each one of these phases. It is also worth noting that the figure does not make a distinction between network layer multicast – when the underlying network supports multicasting – and application layer multicast – when the network does not provide multicast support (P2P) and multicasting is implemented in the application layer.

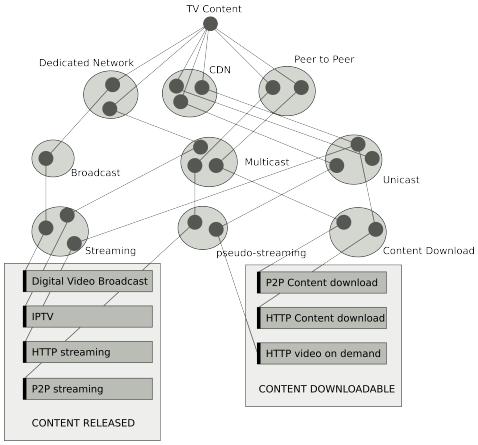

Figure 1. Classification scheme of popular content delivery technologies used for television programs.

In recent years, the amount of network resources consumed by these technologies has experienced an increasing growth and the weight of video traffic is expected to continue rising in the years to come. As an example, according to [1], in 2010 Internet video (IPTV, Internet video and Video on Demand) represented 40% of consumer Internet traffic, reaching 62% by 2015, not including P2P video file exchange. Accounting for video P2P

file exchange, this percentage goes up to 90%. Moreover, according to [2], in the first quarter of 2012, 15% of all television viewership happened on Internet connected devices. This growth is forcing network operators to reconsider how video services are provided today and how video traffic is treated across the networks.

Indeed, these streaming technologies were not deployed because of their network resource utilization efficiency, but mainly because other commercial requirements. For instance, HTTP and P2P streaming respond to a commercial requirement: allowing users to watch television content *anywhere*, through the Internet. Similarly, HTTP Video on Demand (VoD), HTTP and P2P content download respond to the demand for television content to be available at *anytime*. On the other hand, IPTV appeared to allow Internet Service Providers to bundle telephony, Internet access and television together in their service offer. Note that none of these requirements are related to resource utilization. However the core business of a network operator is to monetize its network resources and therefore, it is of capital importance that content delivery technologies use the network in an efficient manner.

Nevertheless, each category in the classification above has very different characteristics when it comes to network resource utilization.

First, dedicated infrastructure and network resources ensure that content is delivered with sufficient quality to all potential viewers inside the service area, at the expense of high costs in network resources. Obviously, the decision about using dedicated infrastructure should be made considering the economic viability of the service, whether the service will generate enough economic resources to sustain the initial investments -Capital Expenditures (CAPEX) - and the Operational Expenditures (OPEX) associated to the infrastructure resources. In this sense, sharing network resources with other services reduces both CAPEX and OPEX, but at the expense of decreasing (or at least compromising) the Quality of Service (QoS).

Second, broadcast and multicast are more efficient than unicast when there are several simultaneous clients of the service, since broadcast and multicast use a single connection to

deliver the content to all viewers. In theory, the use of broadcast (or multicast) is only motivated when there is a critical mass of spectators looking at the same program at the same time. This only happens for television content that is sufficiently popular. For the rest of the content offering, it is more efficient to send the programs over unicast connections.

Last, the provision of streaming services with sufficient Quality of Service (QoS) imposes more strict requirements to the network than content download services. In this sense, streaming is only a requirement for live content. Pre-produced content can be delivered any time after it is produced. Actually, from a network operator point of view, rather than just delivering the content, the challenge is to stream television content to the largest audience affordable, keeping a good QoS.

Current television services do not always take these technological aspects into consideration. As mentioned, there were other aspects rather than economic viability behind the appearance of dedicated broadcast networks for television content delivery. The actual popularity of a program is not considered - or just in very basic terms - in order to decide if that program should be delivered over broadcast/multicast or unicast. Finally, there is no distinction between pre-produced content and live content in video streaming services. All these factors can be exploited by innovative content delivery technologies that actually take them into account, as the content download services under study in this thesis.

The next subsection describes unidirectional background push Content Download Services (CDS), a content delivery technology that can be provisioned over the residual bandwidth of a network with broadcast or multicast support. First, the service is conceived to make the most of broadcast infrastructure, using the bandwidth left by streaming services to push content to local storage capacity in client devices. Second, the service only deals with popular content, well suited for broadcasting. Third, being a background content delivery service, it does not impose any instant bandwidth requirements on the network. Therefore, this kind of services can become a great asset for network operators wanting to make a more efficient use of broadcasting resources.

I.2 Unidirectional background push Content Download Services

Popular CDSs (highlighted in Figure 1 for downloadable content) use either unicast (i.e. HTTP content download and HTTP video on demand) or application layer multicast (i.e. P2P content download) protocols to deliver the content. This means that these services do not benefit from the efficiency inherent to broadcast or network multicast, when the same content is delivered to several clients concurrently. This is because, these services have been designed to work across the Internet, where there is no end-to-end support for broadcasting or network multicasting.

On the contrary, CDSs that use broadcast or multicast connections are referred to as *unidirectional* CDSs and they allow all clients interested in the same content item to download it from a single connection, thus benefiting from the efficiency inherent to broadcasting. Broadcast CDS can be delivered over television broadcasting networks, like DVB networks. Additionally, any network delivering IPTV services provides end-to-end support for IP broadcast and content download services can benefit from this feature.

There are two different kinds of CDSs: *push* content download, where the delivery is initiated by the server; and *pull* content download, where the delivery is initiated by the viewer. Thus, pull CDSs generate traffic in the network after a client request, whereas the traffic generated by push content download services can be controlled on the server side. Another characteristic common to the different CDSs in Figure 1 is that they are pull CDSs and therefore, the bandwidth they use depends to a great extent on the user demand. Contrarily, unidirectional push CDSs are managed by the service provider, who is in control on the generated traffic.

Finally, the concept of *background* CDSs is used in this thesis to refer to services that use network resources not used by any other service. In video broadcasting networks, video streaming services are delivered over reserved network resources of fixed capacity. However, video traffic is by nature variable with time and it follows that the reserved network resources are not used to their full extent at all times. This excess of reserved capacity can be used to provision other services with no instant capacity requirements, as

long as they do not compromise the QoS of the streaming services. Clearly, unidirectional background push CDSs can benefit from this excess of reserved capacity, since they do not have any instant capacity requirements and the traffic generated can be controlled at the server side. Hence, *Unidirectional background push CDSs* send multimedia content to customer premises devices through existing broadcast connections, without interfering with the primary live streaming services.

At the client side, the service stores the files delivered by the unidirectional background push CDS in local storage. The download process is transparent to the user: there is no explicit indication from the user on which files to keep in storage. Moreover, the service works as a prefetching cache, downloading files in the background before they are offered to the user.

Unidirectional background push CDSs reuse the network infrastructure of television streaming services, while at the same time consume little (and unused) network resources. Therefore, these services turn out to be inexpensive to network operators. Hence, background broadcast push CDS can improve the efficiency of 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 network with broadcast support and can play different roles in combination with other television services, either Linear TV services or VoD services. For the latter, the service could be used to push content to local storage. This way, the background service works as a prefetching cache, thus reducing both the traffic of the on demand service and the access time. Another use case, 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 unidirectional background push CDS could provide value for different actors in the value chain. Content providers make more value of the resources

dedicated to content delivery; network operators make a better usage of network resources; and television viewers benefit from improved Quality of Experience (QoE).

I.2.1 Architecture

Figure 2 shows the architecture of the unidirectional background push CDS proposed in this thesis. A CDS delivers content from a content repository, together with metadata descriptions of the content (Content Descriptions). The CDS implements a Scheduler, establishing the order at which files are delivered. The main role of the scheduler is to optimize the service performance, by changing the order in which files are transmitted, according to properties like their size or their popularity. Later, the CDS is delivered over a background virtual channel by means of Opportunistic Insertion. Opportunistic Insertion inserts packets from the Content Download Service whenever there is capacity available in the network. This way, the CDS is delivered over a virtual channel, the background channel, made of the residual transmission capacity in the reservations of a television service (the primary service).

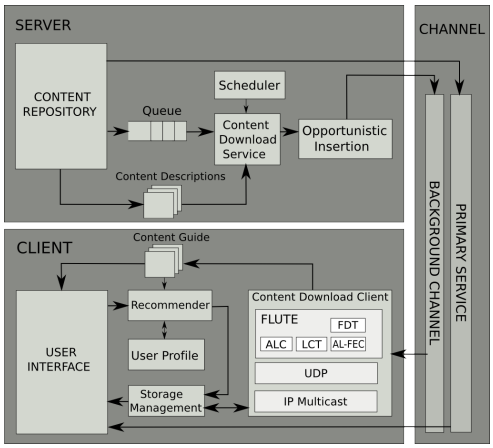

Figure 2. Unidirectional background push CDS architecture.

The client needs to implement the corresponding broadcast CDS client. The figure presents the protocol stack of the one implemented and under study in this thesis, the FLUTE (File Delivery over Unidirectional Delivery) protocol, described in the next section. The storage

memory is managed by a Storage Management policy, to ensure that the client uses only the storage capacity reserved for the background service. In this proposal, a Recommender uses feedback from user interaction and the metadata in the Content Guide to model the user preferences (User profile). With this, the Recommender is able to provide the Storage Management with an estimation of the usefulness of each content item. In turn, the Storage Management uses this information to filter the contents offered in the CDS, in order to keep in storage only the items that better fit the user preferences. This introduces some level of personalization to the service, thus improving the QoE.

This brief overview highlights the main features of the service: Use of residual capacity to deliver CDSs and storage management to introduce personalization. Nonetheless, in order to better understand the objectives of this thesis, it is necessary to introduce the key concepts behind two of its main components: File delivery over unidirectional transport and storage management for unidirectional file delivery.

I.2.2 File delivery over unidirectional transport

Multicast protocols improve the scalability of data transfer services. In unidirectional environments, where the network does not guarantee the delivery of data without errors, it is necessary to provide additional mechanisms to improve the reliability of the file transmission. FLUTE [3] is a protocol for the reliable provisioning of content download services over the unidirectional IP transport protocol UDP.

FLUTE transmissions are organized in sessions, identified by a unique Transport Session Identifier (TSI) and addressed to a single IP multicast group. The traffic of a FLUTE session can be organized in several channels, each one using a different UDP (User Datagram Protocol) destination port and transmission rate. In FLUTE, each file has a unique identifier – Transport Object Identifier (TOI) –, included in the FLUTE header of every packet so that clients can filter the packets of the file they want to download. The list of files delivered in a session and their corresponding TOIs are indicated in the File Delivery Table (FDT), which is a text file delivered in band (with TOI equal to 0). There are two different ways to schedule transmissions within a FLUTE session: file carousels

and scheduled file transmissions. In the latter, the session has a limited duration, bounded by a start time and an end time. On the other hand, in file carousels, files are transmitted one after another on an endless loop.

Moreover, there are three complementary mechanisms to provide reliability in FLUTE file delivery sessions: retransmissions, Application Layer Forward Error Correction (AL-FEC) and file repair sessions. File retransmissions allow the clients to recover the missing packets of a file by transmitting the same file again in the session. Clearly, retransmissions are a reliability mechanism inherent to file carousels. Additionally, AL-FEC consist of generating redundancy to the transmission of a file, so that clients are able to recover missing packets from the redundant information. Finally, file repair sessions provide an off-line bidirectional service to download missing packets on request.

Regarding AL-FEC, FLUTE implements a FEC building block that allows to generate FEC parity from file data and send the parity over the session together with the original contents of the file. FLUTE divides the original file into source blocks, each consisting of n encoding symbols: k source symbols and n-k parity symbols generated by a FEC code applied over the original symbols. Each encoding symbol conforms the payload of a FLUTE packet. The relation k/n, or *code rate*, establishes the parity introduced in the file (the less k/n, the more protection). Sometimes, this relation is expressed as the ratio n/k, referred to as the *FEC ratio*.

In order to generate parity symbols, the FEC encoder performs mathematical operations over other symbols (source symbols or previously computed parity symbols). The decoder performs the opposite mathematical operations to recover a missing packet from correctly received packets. The efficiency of an AL-FEC code is expressed as the ratio between the number of packets needed to recover a block and the number of source symbols k. This coefficient is known as the *inefficiency ratio* (*inef_ratio*).

I.2.3 Storage management for unidirectional file delivery

At the receiver, the service client downloads files from the FLUTE session. The storage size is limited and may not be sufficient to store all the files that a user is interested in.

Therefore, it is necessary to implement storage management policies to maximize the value of the files kept in memory.

In the literature, these kinds of client applications are referred to as broadcast caches [4]. Previous studies indicate that the performance of broadcast caches is subject to the estimation of a parameter: the usefulness, that is, the probability that a file is requested in the near future. In this proposal, the usefulness is derived from the historical usage of the television service of a particular viewer, i.e. the user profile.

As indicated in Figure 2, the service delivers a metadata description of each file in the form of a Content Guide. Content-based recommendation systems [5] calculate the usefulness of a file by comparing its metadata description to the user profile. Basically, a content-based recommender describes every content item on a specific semantic space. Similarly, a user profile expresses the user preferences in the same semantic space. Then, the usefulness of a content item is estimated with a similarity function between its description and the user profile. Hereby, the storage management uses this estimation to decide which files to keep in memory.

I.3 Problem definition

As mentioned above, in convergent television scenarios, there are multiple transport options for television content delivery: unicast or multicast, streaming or (push / pull) content download. For a network operator, there are many parameters that determine what is the best alternative for a given program (e.g. content audience, popularity or network conditions). Therefore, it is of interest for network operators to look into appropriate transport schemes combining seamlessly different methods to better utilize their network infrastructure.

In this context, the motivation of this thesis work is to:

> Evaluate the use of background capacity -network capacity otherwise unused- to prefetch pre-produced television content over unidirectional background push content download services.

Unidirectional background push services reuse network investments meant for other services, while at the same time consume little (and unused) network resources. Therefore, these services turn out to be inexpensive to network operators. Hence, unidirectional background push CDS can improve the efficiency of television content delivery with very little overhead in terms of operational expenditures and no need for additional infrastructure. With this motivation, the main objective of this thesis work is to evaluate the benefits of the provision of background content download services for different stakeholders in the television content value chain: television content providers, network operators and consumers.

I.4 Scope and contributions

This thesis provides different contributions to the study of this kind of services, which are listed below:

i) Unidirectional background push Content Download Services

The first contribution of this thesis work regards the modeling of the key system aspects of content download services for television content delivery, taking into consideration:

What are the quality metrics that characterize the performance of background Content Download Services? What models are needed to evaluate these metrics for television content?

The next section, Methodology, and the Chapter II are dedicated to these questions. The Methodology section provides a description of the quality metrics of CDSs, together with a brief introduction of the models used to characterize key aspects of the system, like the

content source, the communication channel and the user requests. Later, Chapter II describes the different models used.

ii) Opportunistic insertion of background television services

Next, this thesis is dedicated to verify the viability of the service proposal and provide an answer to the questions:

Is it possible to provide unidirectional background push content download services over television networks? What mechanisms can be used to insert data of a content download service together with a streaming service without compromising its QoS? What is the bandwidth available for the background service?

In this sense, Chapter III describes Opportunistic Data Insertion in first-generation broadcast networks and second-generation broadcast networks, as the technology enabler to use residual capacity for the insertion of background services. Moreover, the chapter provides an empirical assessment of the capacity available for opportunistic insertion in broadcast networks. These measurements are used in Chapter IV to obtain models for opportunistic insertion in different scenarios and to estimate the performance of background CDS over background channels.

iii) File carousel AL-FEC and scheduler optimization.

Later, the research work is focused on the server side, especially in the AL-FEC building block and the file scheduler:

What configurations of the FLUTE AL-FEC block optimize the service performance? And what file scheduling policies? What is the relationship between AL-FEC and file scheduling?

In this sense, Chapter V shows the improvements on the service performance brought by AL-FEC encoding, object multiplexing and the combination of both techniques. The

chapter presents a novel algorithm for object multiplexing, based on the Fluid Fair Queuing (FFQ) algorithm, comparing its performance to a previously proposed algorithm.

iv) Storage management, QoE and personalization

Finally, the last section of the thesis deals with the process at the client side, especially with the storage management and its relationship to the user experience:

What are the properties of a file that should be considered by storage management policies? What is the optimum size of a cache for a background CDS? How does the CDS affect the QoE of television services?

Chapter VI analyzes the performance of the storage management policies. The results assess the effect of the recommender in the service performance and the relationships between the storage size and the file carousel size.

I.5 Methodology

The methodology applied in this thesis is mainly based on the evaluation of two parameters that determine the performance of the service under study, the *overall access time* and the *average cache hit ratio*. The evaluation is performed through simulations and experimental measurements to validate the results of the simulations.

The access time is regarded as the most important quality metric for content download services. Back to the architecture in Figure 2, at one point, the storage management discovers a new content item in the content guide and decides that it should be kept in local storage. The access time is defined as the time elapsed between the instant when the storage management makes the decision until the content is completely downloaded in local storage. Furthermore, the overall access time is defined as the average access time among all the different content downloads in the service area.

On the other hand, the cache hit ratio is defined as the proportion of user requests to an on demand service that are successfully served from local storage, providing to the user the files that have been previously downloaded (pre-fetched) with the background push CDS.

As with the average access time, the average cache hit ratio is an average of the cache hit ratio among all users in the service area.

The evaluation of the overall access time and the average cache hit ratio relies on a system model for unidirectional background CDSs for television content delivery. Figure 3 shows a schematic representation of the system model:

Figure 3. Diagram of the system model

The parameters of the model are set so as to simulate different service scenarios. Each scenario is characterized by the nature of the content, the average bit rate available for the background CDS and the amount of losses in the channel.

In the content model, each program grid item –programs, commercials- is regarded as a separate file in the Content Download Service. In this study, television programs are characterized by their content duration, file size and popularity. The models used in this thesis are based on parametric statistical distributions, obtained from commercial television content.

As for the server model, one of the main characteristics of the proposed service architecture is the use of FLUTE file carousels to deliver the content. Consequently, the server model simulates the generation of FLUTE file carousels, taking as input the properties of the files defined in the content model. As indicated, the FLUTE protocol implements an AL-FEC building block. Adding AL-FEC parity to the carousel improves the download time in channels with packet losses, at the expense of increasing the carousel cycle (i.e. the time needed to broadcast the different files). The simulations are aimed at finding optimal configurations of the AL-FEC building block for FLUTE file carousels.

Additionally, related literature [4], [6] shows that file carousels for television content may be weighted, so that files do not necessarily have the same carousel cycles, for instance, to foster the download of popular content. The scheduler is the system block in charge of shaping the file carousel. Although there are several studies about scheduling policies for weighted carousels, none regard packet losses and AL-FEC parity. This thesis work addresses the problem with a different perspective than previous works, by taking into account both network and application layer requirements under the same study. The simulations show how AL-FEC and weighted carousels affect the overall access time in the presence of channel losses.

The channel model provides a model for the packet losses in the communication channel. More specifically, in FLUTE carousels, the channel losses determine the expected number of carousel retransmissions needed to download a certain program. In order to validate the channel model, a series of experimental results are conducted using a service prototype in laboratory conditions.

At the receiver side, the storage management needs to decide which files to download from the file carousel and manage the storage space available for the service. Moreover, this decision-making process should be transparent for the user, but at the same time, it should introduce some degree of personalization in the service, in order to adapt to the user preferences. Like the AL-FEC and the scheduler building blocks, the storage replacement policy is subject to optimization. Specifically, this thesis defines a generic model to evaluate different cache management policies based on the heuristics of the well-known knapsack problem [7]. This model is referred to as the cache model.

Note that the decisions made by the storage management policy are transparent to the user. For this reason, there are separated models for the cache application and the user model. The user model becomes necessary in the evaluation of the cache hit ratio, when the background CDS service is used as a prefetching cache for a VoD service. For this case study, it is necessary to model how the users generate requests for the different content items over time. In this thesis, QoE and personalization are related to the ability of the

service to successfully push programs of interest for the user. Consequently, QoE is related to the access time and the cache hit ratio. It is worth highlighting that the cache of the background content download service saves bandwidth for the primary service on every cache hit. Thus, the cache hit ratio is also related to the bandwidth savings brought by the background service.

The following chapter is dedicated to describe in detail each of the system model blocks mentioned above.

Chapter II

System models

In order to evaluate the performance of background push CDSs through simulations, it is necessary to develop models for the content sources, the communication channel, the storage management and the user. This chapter is divided in three sections. The first section, server models, addresses the models at the server side, that is, the content sources. On the other hand, the second subsection describes the models used to characterize the communication channel. Finally, the client models section describes the models at the client side, that is, the models for the storage management and the user.

Regarding the content sources, current communication network services deal with all sorts of data, from lightweight messages to very large files. In many cases, the evaluation of new service proposals relies on source models to simulate the generation of service traffic. In this sense, the background CDSs under study in this thesis deliver multimedia files containing encoded video. Therefore, in order to evaluate its performance through simulations, it is necessary to obtain meaningful models for the most relevant parameters of video files: The file size and the popularity.

Thus, section II.1 includes parametrical statistical models for the content duration and the popularity of television programs. These models are obtained by fitting their curves to the statistics of a popular online television content repository, the Internet Movie Database (IMDB) [8]. Such models can be very valuable for studies dealing with television content

delivery to different kinds of devices. To our understanding, there is no previous work modeling the duration of television programs (professional content appearing in traditional television channels). Alternatively, the section includes models for the content duration of user generated content sources (YouTube) available in previous research works.

Similarly, in order to simulate the performance of a service in a channel with packet losses, it is necessary to obtain models for the communication channel in the different reception conditions under study. Section II.2 presents the channel model used in this thesis to simulate channel losses, specifically designed to deal with carousel retransmissions in broadcast CDSs. The channel model has been validated through measurements carried out with an application prototype in laboratory conditions.

Finally, after presenting the models used for the content and the communication channel, II.3 introduces the models used for the storage management application and the users. In a general sense, it can be stated that any storage management application deals with a combinatorial optimization problem, because it needs to decide which is the subset of the files in the carousel best kept in local storage, without overpassing the storage space limit. In combinatorial optimization, this problem is known as the knapsack-problem [7]. Section II.3.1 presents the model obtained for the storage management policy used in this thesis, which is based on a well-known heuristic that solves the knapsack problem. The cache model allows to simulate the decisions made by different storage management policies and in extension, different implementations of the client application. Later, section II.3.2 describes the model used to simulate the generation of user requests, that is, the user model. Based on the well-known Poisson distribution, the user model makes it possible to generate user requests for different files over time.

This way, the different service models presented in this section are used in the simulations carried out in later chapters to produce the results.

II.1 Server models

The increasing success of IP video delivery services technologies poses several challenges to both network operators and service providers. Services like Video on Demand and IPTV are becoming the most important sources of Internet traffic and for this reason, a lot of on-going research studies focus on technologies that deal specifically with video traffic. New cache replacement strategies [9], hybrid multicast-unicast transport [10], P2P systems [11] or CDN networks [12] are few of the technologies conceived to alleviate the pressure that video traffic puts on IP networks.

In this context, the duration and the popularity of video items are important parameters for file-based video delivery technologies, like HTTP streaming [13] or FLUTE [3]. In the evaluation of these technologies, it is important that these parameters are accurately modeled, providing statistical properties as similar as possible to real video sources.

Ultimately, the file size of video items depends on two fundamental characteristics of media files: The duration and the encoding rate. While the content duration is intrinsic to the content itself, the encoding rate is dependent on the application. As an example, a service targeting mobile devices does not apply the same encoding rate as a service that targets High Definition television sets, but the duration of a particular program is the same in both cases. As Internet video is becoming more ubiquitous, it seems more convenient to use separate models for the content duration and the encoding rate. This is the approach taken in this thesis. Nevertheless, to our understanding there is no previous work modeling the duration of television programs (professional content appearing in traditional television channels). Such models can be very valuable for studies dealing with television content delivery to different kinds of devices.

On the other hand, there are several use cases where a model of the popularity of television content becomes necessary. For instance, the design of new caching strategies based on content popularity [9] or the optimization of hybrid multicast-unicast transport [10] -using multicast for the most popular programs and unicast for the less popular programs- require models for the content popularity. There are several research works that analyze the

popularity of Internet video sharing sites (e.g. YouTube), but to our understanding, there is no previous work modeling the popularity of television content. Currently, television content is delivered through different technologies (broadcast, IPTV, Internet video) and it seems crucial to characterize its popularity.

With this motivation, we have obtained parametric statistical models for the content duration and the content popularity of television content, obtained from the Internet Movie Database (IMDB) [8]. Quoting its web site, the IMDB is "the world´s most popular and authoritative source for movie, TV and celebrity content". The IMDB gathers information about television programs aired all over the world and, with several hundred thousand entries, it represents a very comprehensive set of data. The parameters of the models are adjusted to fit the statistics of the empirical data in the database. Later, the accuracy of the models is numerically assessed and example applications illustrate how the models can be used for the analysis of the performance of video delivery technologies. Finally, the models are compared to other models presented in previous work dealing with UGC content.

II.1.1 Parametric models for file sizes and content durations

The file size is an important parameter in the evaluation of content download services. However, in many study cases, the sizes of all the files available through the service are not known a priori. This is particularly true when the evaluation has a general character, when the number of possible files to take into consideration is very large or when their characteristics are not completely determined. In these cases, it is convenient to model the size of files by means of statistical distributions that provide approximated values to the actual file sizes in a given application.

For this reason, different research papers have investigated the statistics of web files (Hypertext and image files) available on the Internet, in order to obtain statistical models for the file sizes of web content. In [14], the file sizes of four different datasets are approximated by a lognormal (LN) distribution. Later, the study analyses the different datasets in order to find evidences of *long tails* in their Probability Density Functions (PDF).

Statistical distributions where one can expect very large files or very small files, compared to the average size, are referred to as long tail distributions. This name is due to the fact that the presence of large files or small files in the samples provides the PDF with an asymptotic behavior. As shown in Figure 4, in a long tail distribution, the components of the PDF that are far away from its body, or central part, have a significant weight. On the other hand, a file size distribution where the appearance of very large or very small files is a rare event is categorized as short tail distribution, as it is generally the case for lognormal distributions.

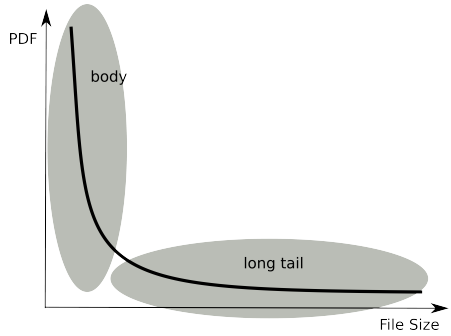

Figure 4. Example of long tail file size distribution.

The conclusion of [14] is that these long tails are not found in the datasets under analysis, unlike other previous studies on the workload of web servers [15], which stated that the PDF of web file sizes has a normal body but long tails. Testing that there are no long tails is important for the hypothesis behind the lognormal model: new web files are modifications of previous web files and therefore, their size can be determined by multiplying the size of previous web files by a random factor to a previous file size, as depicted in Figure 5. This assumption, known as the *law of proportionate effect*, is the structural cause of the LN file size distribution, that is, its generating model. Furthermore, the main objective of [14] is to prove that this generating model yields to a LN distribution.

On the other hand, [16] presents a different generating model, where new files are not necessarily variations of the same file, but of a set of files. This new generating model,

which appears to be more flexible than the one presented in [14], yields to a Double Pareto Lognormal (DPLN) distribution [17]. That is, the structural cause of the DPLN file size distribution is the combination of applying the law of proportionate effect to several files in parallel, as shown in Figure 5.

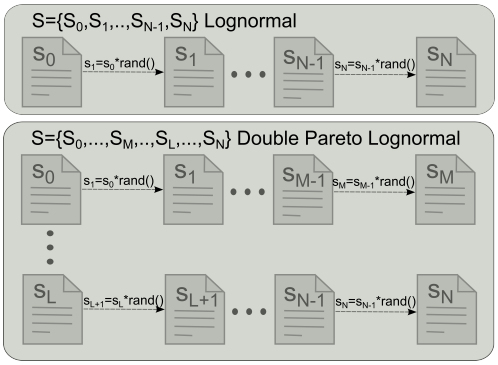

Figure 5. Generating models for the Lognormal and the Double Pareto Lognormal distributions.

Regarding video content, there are different research works investigating the workload of video sharing Internet sites. [18] analyses the requests to YouTube from a University campus network during a long period of time, providing values for the mean and the covariance of the file size obtained from the requests. Similarly, [19] develops a lognormal model of the traffic generated by requests to YouTube from a campus network. The file size of YouTube content is estimated by randomly sampling the payload size of the requests. On the other hand, [20] uses a monitoring tool that issues automatic requests using the YouTube Data Application Programming Interface (API). The paper provides two fitting functions for the encoding rate and the content duration of YouTube content.

In summary, previous related works regarding web files or videos from YouTube provide statistical models for the file size in their respective services. Most of the related studies obtain parametric statistical models that follow either the LN distribution or the DPLN distribution. At this point, it is worth highlighting that the importance of the LN distribution and the DPLN distribution lies in their underlying generating models, which can be used to

extrapolate future behavior of the distributions. It is important to emphasize that it is harder to motivate such extrapolations just by fitting statistical distributions to empirical data.

Clearly, the models obtained in the aforementioned studies are not valid for television content, mainly for two reasons. The most important reason is that the nature of the content is different. Obviously, neither web content nor User Generated Content (UGC) have the same characteristics as television content, which mainly consists of television shows and movies. The second reason is that the file size of videos depend on the encoding rate, which is application dependent – a video for an Internet VOD site is not encoded in the same way as a video for a service targeting television sets.

However, the same methodology can be applied to obtain the LN and DPLN distributions that best fit the duration of television programs. In order to do this, it is necessary to replace file size for content duration in the assumptions made to apply the underlying generating models.

This way, according to the generating model of the lognormal approximation, the duration of television programs can be obtained by applying a random factor to the duration of previous television programs. This generating model yields to a lognormal distribution for the content duration. The PDF of the lognormal distribution is given by:

$$\mathrm{pdf}(s) = \frac{1}{\sigma\sqrt{2\pi}s}\exp\left[-\frac{1}{2}\left(\frac{\ln(s)-\mu}{\sigma}\right)^2\right] = \mathrm{LN}(\mu,\sigma) \qquad (1)$$

where s is a data sample. The parameter μ is referred to as the location and it is related to the expected value of the distribution. The parameter σ, i.e. the shape, is related to the way that the variation of the distribution is shaped around the location. Given a set of data samples, s, the estimation of the parameters of the LN distribution is very straightforward: μ is approximated by the mean of the logarithm of s and σ is estimated as the standard

deviation of the logarithm of *s*. On the contrary, the mean of *s* is given by $e^{\left(\mu+\sigma^2/2\right)}$ and its variance by $\mathrm{e}^{2\mu+\sigma^2}\left(e^{\sigma^2}-1\right)$.

Regarding the Double Pareto-Lognormal (DPLN) distribution, the generating model also assumes that television programs are modifications of previous programs. The main difference with the lognormal generating model is that there can be several different initial programs in the generating model. The PDF of the DPLN distribution is:

$$\begin{aligned} pdf(s) &= \frac{\delta\beta}{\delta+\beta} s^{-\delta-1} A(\theta,\nu,\tau)\Phi\left(\frac{\log(s)-\nu-\delta\tau^2}{\tau}\right)+ \\ &+\frac{\delta\beta}{\delta+\beta} s^{\beta-1} A(-\beta,\nu,\tau)\Phi^c\left(\frac{\log(s)-\nu+\beta\tau^2}{\tau}\right) = DPLN(\delta,\beta,\nu,\tau) \end{aligned} \tag{2}$$

where $A(\theta,\nu,\tau)=e^{\theta\nu+\delta^2\tau^2/2}$, while $\Phi\left(x\right)$ and $\Phi^{\mathrm{c}}\left(x\right)$ are the PDF and the Cumulative Density Function (CDF) of the standard normal distribution. The parameter δ is related to the right tail of the distribution, whereas the parameter β is related to the left tail of the distribution. On the other hand, ν and τ are related to the location and the shape of the distribution. In general, the DPLN has long tail behavior in both tails. Furthermore, this behavior can be observed in the CDF and the Complementary CDF (CCDF) of the distribution:

$$\begin{aligned} pdf(s) &\sim \delta A(\delta,\nu,\tau)s^{-\delta-1} \quad (s\to\infty) & pdf(s) &\sim \beta A(-\beta,\nu,\tau)s^{\beta-1} \quad (s\to 0) \\ ccdf(s) &\sim A(\delta,\nu,\tau)s^{-\delta} \quad (s\to\infty) & cdf(s) &\sim A(-\beta,\nu,\tau)s^{\beta} \quad (s\to 0) \end{aligned} \tag{3}$$

Moreover, these equations show how the parameters δ and β determine the shape of the tails of the distribution. On the other hand, the estimation of the parameters is more complicated for the DPLN distribution than for the LN distribution. [17] presents two different methods to obtain the parameters of the DPLN distribution, the Method of Moments Estimates (MME) and the method of Maximum Likelihood Estimates (MLE).

Both are well known methods in statistics. The MME method provides an estimation of the parameters of the DPLN distribution from the statistics of the set of data samples, *s*, as:

$$\begin{aligned} &\kappa_3 = 2/\delta^3 + 2/\beta^3 \qquad \kappa_4 = 6/\delta^4 + 6/\beta^4 \\ &\mathrm{E}(\log(s)) = \nu + 1/\delta - 1/\beta \qquad \mathrm{var}(\log(s)) = \tau^2 + 1/\delta^2 + 1/\beta^2 \end{aligned} \tag{4}$$

In the equation, κ_3 and κ_4 are the third order and fourth order cumulants of the logarithm of the set of data samples, *s*. Depending on the dataset, the MME may not provide a valid solution. Nevertheless, this is not a problem because the use of the MME method is only recommended to provide initial values for the MLE method.

On the other hand, the MLE method is an iterative method that obtains the parameters that maximize the likelihood function, which is the likelihood that the empirical data is obtained from the distribution. The convergence of the method depends on its initial values. In the absence of a valid solution of the MME method, [17] indicates that the initial parameters for the MLE method should be find by trial and error. Moreover, it is recommended to check the PDF, CCDF and CDF in order to verify the asymptotic behavior in both tails of the distribution to avoid convergence issues.

II.1.2 Modeling the running length of television (IMDB) content

The last section proposed the LN and the DPLN distributions as two parametric statistical distributions that can be used to model the content duration of television content. In this section, these distributions are fitted to the distributions of the content duration of items in the IMDB database. The IMDB provides alternative interfaces to use the database for non-commercial purposes. Through them, it is possible to download the content duration (running length) information as a plain-text file that has 741,281 entries. Figure 6 shows the CDF (top-left), the CCDF (bottom-left) and the PDF (right) of the content duration of television programs obtained from the IMDB in logarithmic scale, together with the asymptotic CCDF and CDF approximations in eq. 3 that evidence the power law behavior of the distribution. The PDF shows program durations from few seconds and up to a maximum length of almost a week. The PDF has its central body from approximately zero

to 100 minutes of duration. There are clear peaks at 30 minutes, 60 minutes and 90 minutes, which are very typical durations for television programs. The mean duration of the distribution is 55.6 minutes and the standard deviation is 53.7 minutes. The third and fourth order cumulants are 0.82 and 0.26. Regarding the tails of the distribution, the CDF and the CCDF are rather linear, which may indicate long tail behavior of the distribution for both short and long programs.

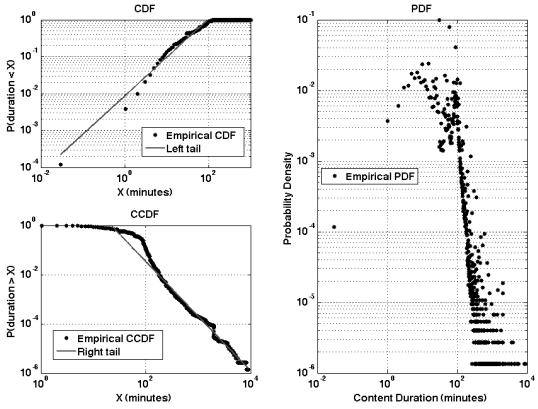

Figure 6. Cumulative Density Function, Complementary Cumulative Density Function and Probability Distribution Function of the running length in the IMDB database.

The parameters of the Lognormal distribution are obtained from the empirical data just by setting μ to the mean value and σ to the standard deviation [8] of the logarithm of the samples. This method yields to the distribution LN(3.7,1.0). Regarding the DPLN distribution, the MME method (eq. 4.) does not have a solution for the IMDB sample data. Therefore, the starting values for the MLE method must be found by trial and error [17]. We have initiated the search with the approximations in eq. 3. ($\delta = 2.22, \beta = 1.04, \nu = 3.55, \tau = 0$), but the MLE did not converged. However, with the initial values of ($\delta = 2.22, \beta = 1.04, \nu = 8.29, \tau = 1.26$) the MLE method converges to the distribution DPLN(4.36, 1.26, 4.22, 0.63). Additionally, the distribution DPLN(8.96, 2.87, 3.65, 1.20) has been obtained by curve fitting the PDF of the IMDB data.

Figure 7 shows the empirical PDF of the IMDB dataset, together with the PDFs of the different analytical models presented above: The lognormal (LN) distribution, the DPLN approximation according to eq. 3 (DPLN AP), the DPLN distribution obtained with the MLE method (DPLN MLE) and the curve fitted DPLN (DPLN CF). As shown in the figure, the LN distribution, the DPLN MLE and the curve fitted DPLN distribution provide good approximations for the central part of the distribution. However, there are significant differences between the model and the empirical data at the peaks highlighted in Figure 6, as well as in the left tail of the PDF. For these reasons, none of the distributions provide a perfect match from a goodness-of-fit point of view, although they provide density functions close to the empirical data.

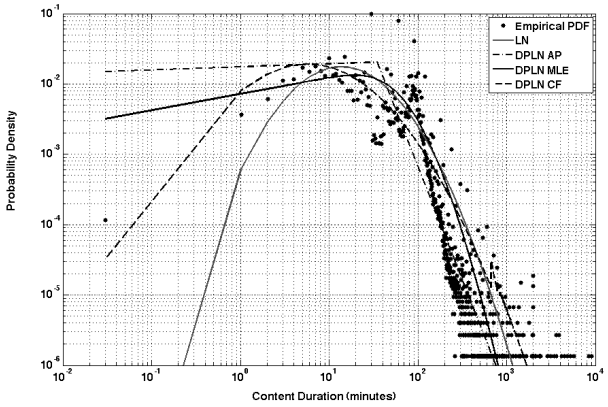

Figure 7. Probability Density Function (PDF) of the IMDB dataset together with the Lognormal distribution and the Double Pareto Lognormal Distributions.

In order to show the behavior at the left tail of the distribution, Figure 8 shows the empirical CDF of the short duration programs (from 6 seconds to 100 minutes) in the dataset and the parametric models. The comparison of the CDF in Figure 8 shows that the LN distribution and the DPLN CF distribution approximate the CDF of the dataset with a similar accuracy for the short duration samples in the dataset. The DPLN MLE distribution provides a better fit to the central part of the distribution, but the DPLN CF approximates better the shortest duration samples. Moreover, it is worth noting that the DPLN is more

likely to provide very small program durations, whereas the LN distribution is less likely to provide very small program durations, both compared to the empirical CDF.

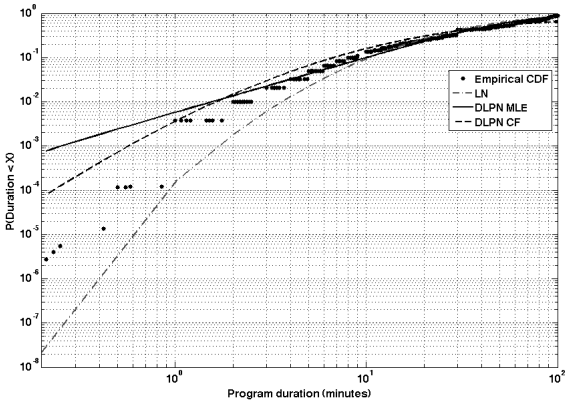

Figure 8. Cumulative Density Function (CDF) of the IMDB dataset together with the lognormal distribution and the Double Pareto Lognormal Distribution.

On the other hand, in order to see clearly what happens at the right tail of the distribution, Figure 9 presents the CCDF of long duration programs, from 10 minutes and up to 1000 minutes.

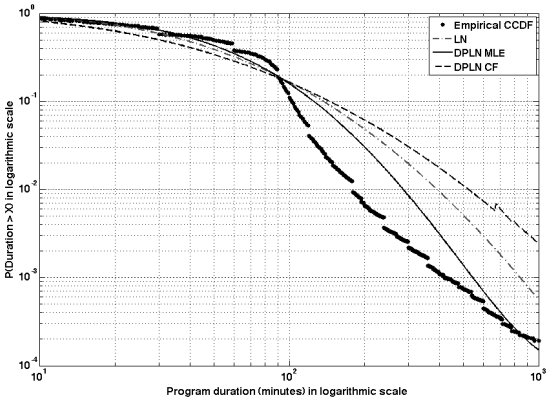

Figure 9. Complementary Cumulative Density Function (CCDF) of the IMDB dataset together with the lognormal distribution and the Double Pareto Lognormal Distribution.

The DPLN MLE distribution provides the best approximation to the empirical CCDF and in general, a better fit to statistics of the empirical data. On the other hand, the DPLN CF distribution provides an upper bound for both tails of the empirical data. This can be convenient in simulations verifying how the long tails affect the service performance.

Looking at the CDF, CCDF and PDF, it is clear that the parametric distributions have density functions that are close to the statistics of the running length in the IMDB dataset. At this point, it is interesting to evaluate the expected error in a simulation. With this in mind, let us define the set of N durations selected randomly from the database $X = [x_1, x_2, ..., x_N]$ and the set of N durations $Y = [y_1, y_2, ..., y_N]$, generated with a probability distribution. Both sets are sorted so that they are monotonically decreasing. Then, the error of sample i is defined as $\varepsilon_i = \text{abs}(x_i - y_i) / x_i$.

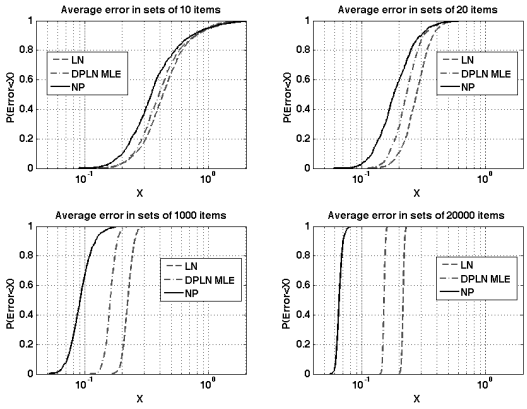

Figure 10. Cumulative Density Function of the differences between sets of files generated with statistical models and randomly selected from the database.

Figure 10 shows the CDF of the mean error found in 1,000 simulations of sets of 10, 20, 1,000 and 20,000 random samples. The error found for the LN and the DPLN MLE distributions is compared to the error of a non-parametric (NP) distribution (generated with Matlab fitting tools) that matches exactly the statistics of the database. The DPLN CF

approximation is obviated because it provides worst results than the DPLN MLE distribution.

The error of the DPLN MLE distribution is lower than the error of the LN distribution in all cases. Note that, due to the difficulty of modeling small sample sets, the mean error is higher for sets of 10 and 20 files. Moreover, the differences with respect to the error of the NP distribution increase with the number of files in the set. However, in every case, the error of the two parametric distributions is rather close to the error of the NP distribution. For instance, the mean error found in sets of 20 files is lower than 0.37 for 90% of the simulations with the LN distribution, lower than 0.33 with the DPLN MLE distribution and lower than 0.30 with the non-parametric distribution. Likewise, the error of the LN distribution is smaller than 0.22 in 90% of the simulations for sets of 20,000, whereas the error of the DPLN MLE distribution is smaller than 0.15 and the error of the non-parametric distribution is smaller than 0.07. It is worth noting that the size of current Internet movie catalogs is in the order of thousands (e.g. around 60,000 in Netflix or 1,800 in Amazon), so the values for sets of 1,000 and 20,000 are representative of the error expected when emulating streaming services.

In conclusion, provided the great range of durations found in the database, it can be stated that the DPLN MLE distribution provides a simple and relatively accurate model to generate sets of programs from the IMDB, when the number of files required by the simulations is sufficiently large. On the other hand, both the LN and the DPLN models provide simple and accurate models for the duration of television programs when the number of files in the simulations is small. In any case, whether the model is valid or not for a particular application depends on the accuracy required in the simulations and on how the errors in the model affect the results of the simulation.

Regarding the objective of this thesis, it is not so important to work with very accurate models, but with simple models that provide similar program durations as real television programs, allowing to draw general conclusions about the performance of background CDS. On the other hand, the number of files used in the simulations is relatively small. For

these reasons, the LN model is used in the rest of this work to obtain the program durations of the television programs in the simulations. On the other hand, the DPLN model is used in some studies, in order to assess the effect of the long tails (very long programs) in the results.

It is worth noting that, since the models provide values for the program duration, it is possible to use them for different applications (e.g. High Definition video or video to portable devices) by adjusting the encoding rate. Additionally, for better accuracy, it is possible to use either the non-parametric model or the values from the dataset, at the expense of computational power.

Finally, as mentioned, the IMDB collects data from television productions and movies. However, there are other sources of content not accounted for in the database, like user-generated content. In order to provide a broader perspective to the study, we will use another set of program duration distributions from measurements performed over the VoD service YouTube in [18]. By the time the study was performed, Youtube had limited the maximum size of the content uploaded to the site. Thus, the contents in the video service were mainly short user-generated videos and promotional videos like advertisement, movie trailers or music video clips. Therefore, the dataset under analysis represents another kind of television content not regarded in the IMDB, but still relevant for this study. The data analyzed provides a mean duration of only 162 seconds and a coefficient of variation of 0.55 seconds, which is the ratio between the standard deviation and the mean and consequently, the standard deviation is 89.1 seconds. Hence, the distribution LN(0.86, 0.51) is used in following studies to model YouTube content.

II.1.3 Parametric models for the content popularity

The content popularity is a crucial parameter in the evaluation of the workload of content delivery services. In this study, content popularity is a measure of the audience of television programs. Therefore, the most popular program of a service is the program that has more views. Hence, content probability is related to the relative access probability of each

television program offered in the service. As with the file sizes, it is necessary to obtain probability distributions that can model the popularity of television content.

[21] describes how the ZIPF distribution (and in general power law distributions) provides a good model for the long term, stationary behavior of complex network systems, as long as they have two characteristics: a) they are *open networks*, in the sense that new nodes are added continuously to the network and b) they exhibit *preferential attachment*, or preferential connectivity, meaning that better connected nodes are more likely to be connected to other nodes than nodes with less established connections.

From this perspective, [22] finds evidence of the ZIPF distribution in the popularity of web content. Web pages are regarded as nodes and hyperlinks as links between nodes in the graph model of the network and the number of links to pages can be model with a ZIPF distribution. Furthermore, the parameter of the ZIPF distribution can be obtained from the underlying preferential attachment model. Thus, it is possible to simulate future states of the network. This model for the size of web files has been used in numerous publications.

In a similar way, the popularity of video content can be modeled as a network. This way, content and users are regarded as nodes of a complex and dynamic network. A connection between a content item and a user is established whenever a user watches (or ranks) a movie. Therefore, the content popularity is determined by the number of connections to a content item, which can be modeled by a certain ZIPF power law distribution. This way, assuming preferential attachment and no information filtering, given a set of N content items ordered by their popularity ranking $i=1,..,N$, so that the item with $i=1$ is the most popular item and the item with $i=N$ is the least popular item, the popularity of program i, $P_N(i)$ is defined as:

$$P_N(i) = \frac{\Omega}{i^{\alpha}}, \quad \Omega = \left(\sum_{i=1}^{N} \frac{1}{i^{\alpha}} \right)^{-1} \tag{5}$$

Note that the ZIPF distribution has only one parameter, α, which determines how fast the popularity decreases with the ranking index. Given a set of empirical data, it is easy to

estimate α, just by linear interpolation of the logarithm of $_{PN(i)}$ against the logarithm of i. obviously, the value of α depends on the nature of the ranked items. For instance, [22] finds that its value is between 0.5 and 0.9 for web content, whereas [23] states that $\alpha=1$ fits well the popularity of rented videos on a movie rental service in the United States.

Moreover, [24] performs an in-depth analysis of the popularity of YouTube videos. This study confirms the power law behavior of the ZIPF distribution across several orders of magnitude of the popularity ranking in different video sharing sites (YouTube and Daim). However, the less popular videos do not fit the ZIPF probability distribution trace. It appears that the popularity of the less popular items is a lot lower than the one predicted by the ZIPF distribution and that a power law with exponential cut-off distribution fits better the experimental data.

[25] shows evidence that information filtering (users not being aware of all the contents available) can be the cause of the exponential cutoff in the distribution of connections in complex information networks. Recall that the ZIPF distribution popularity model is based on an open network model with preferential attachment. Information filtering consists of hiding some nodes of the network to the rest, so that it is not possible to establish links between any two nodes. The number of connections to any node in this new random network model is modeled by a power law distribution with exponential cutoff.

Hence, if we assume that there is information filtering in the network model, i.e. not all viewers are aware of all the television programs, the content popularity follows a power law distribution with exponential cutoff. The popularity of program i is given by:

$$P_N(i) = \frac{\Omega}{i^{\alpha}} \cdot e^{-\chi i}, \quad \Omega = \left(\sum_{i=1}^{N} \frac{e^{-\chi i}}{i^{\alpha}} \right)^{-1} \tag{6}$$

The power law with exponential cut-off distribution has two parameters, α and χ. The power law component (determined by α) determines the popularity of most popular items, up to approximately $1/\chi$. Hence, given an empirical sample, α can be obtained by fitting the popularity of most popular items with linear interpolation in a log-log scale, just as with

the ZIPF distribution. On the other hand, the parameter χ determines the point at which the exponential cut-off should override the power law component and truncate the power-law behavior of the distribution.

In this sense, the empirical data in [24], [26] and [27] exhibits a cutoff in the tail of the power law popularity distributions. In [24], $\alpha=0.84$ and $\chi = 10^{-4}$ for Youtube content, whereas [26] and [27] just provide evidence of the tail truncation but no values for the parameters of the datasets under analysis.

In the next section, the popularity in the IMDB dataset is adjusted to both parametric popularity models.

II.1.4 Modeling the popularity of television (IMDB) content

The IMDB database does not contain information about the audience that the television programs had when they were aired. However, the website allows users to rank the programs in the database. [18] shows that there is a correlation factor of 0.85 between user rankings and the number of access to video contents, meaning that an estimation of the popularity can be obtained from the user ratings. In order to process this information, anyone can download a plain file text from the IMDB alternative interfaces. The user ratings information consists of the total number of votes, the distribution of the votes in a scale from 1 to 10 points and an average rating from all the different votes. We have obtained an estimation of the popularity of each movie as the product between the average rating and the total number of votes, which is exactly the total sum of points awarded to each movie. Later, we have sorted the files in descending ranking order. At the time when this study was performed, the user ranking list in the IMDB alternative interfaces counted with 417,025 entries. Figure 11 plots the IMDB overall rankings together with two approximations that we have obtained following the methodology described in the previous section.

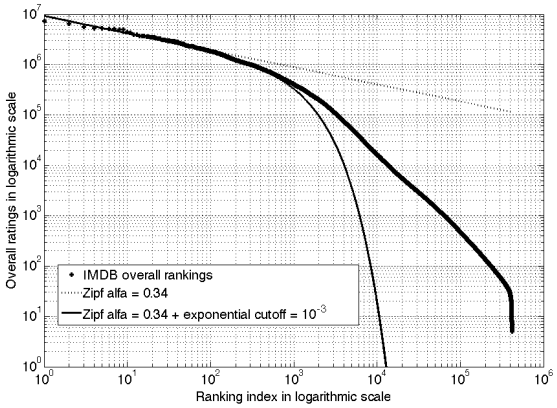

Figure 11. IMDB overall ratings together with the ZIPF approximation and the ZIPF with exponential cutoff approximation lognormal approximation.

The comparison proves that the ZIPF distribution with $\alpha=0.34$ is a good approximation to the overall ratings of the first items in the IMDB. The value of α is significantly lower than the values found in video sharing sites (YouTube) or web files in the previous studies mentioned above. On the other hand, the ZIPF distribution with $\alpha = 0.34$ and $\chi = 10^{-3}$ provides a good fit to the first 1,000 content items in the ranking. However, none of the models fit the curve of the less popular items: The ZIPF approximation provides higher popularity values than the IMDB trace, while the power law approximation with exponential cutoff provides lower popularity values than the IMDB trace.

These models of the popularity can be used to estimate the probability of access to the different television programs offered by a given service. The probability of access is the probability that a viewer requests a particular program. An estimation of the probability of access is very valuable in the evaluation of the performance of video delivery services. Specifically, the probability of access to a video is estimated by normalizing the popularity, so that the summation of the probability of access of the different files offered in the service is equal to one. At this point, it is interesting to evaluate the error of the two models in predicting the probability of access to a certain item. In this sense, Figure 12 shows the error of the probability of access of the two approximations compared to the probability of

access obtained from the IMDB dataset. If $p_{IMDB(i)}$ is the probability of access of file i estimated from the IMDB dataset and $p_{pd(i)}$ is the probability of access using a probability distribution approximation, the error is defined as $\varepsilon(i) = abs(p_{IMDB}(i) - p_{pd}(i)) / p_{IMDB}(i)$.

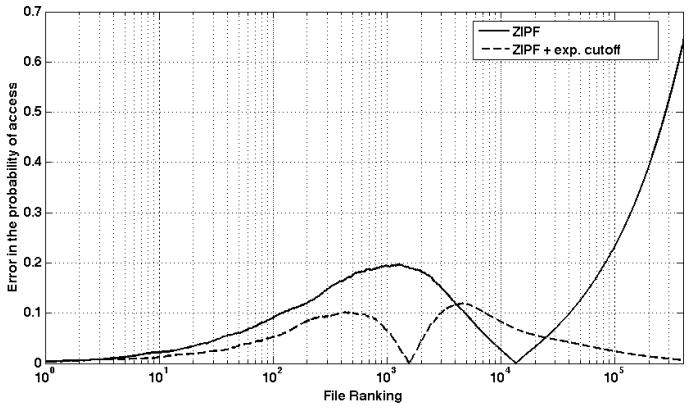

Figure 12. Error in the probability of access for the ZIPF approximation and the ZIPF with exponential cutoff approximation lognormal approximation.

The error traces bounce at zero, since they are proportional to the absolute value of the difference. In general, the power law with exponential cutoff distribution provides smaller errors than the ZIPF distribution. In any case, both provide a good fit to the probability of access of most popular items, (e.g. up to 100 files). However, the error of the ZIPF distribution becomes much larger than the error of the power law with exponential cutoff distribution for the less popular files.

The main conclusion of this section is that the power law distribution with exponential cutoff with parameters $\alpha = 0.34$ and $\chi = 10^{-3}$ models adequately the popularity of IMDB content. For this reason, it is used in future studies to model the popularity of television content.

The ZIPF distribution is used in some studies, in order to assess the effect of information filtering in the results. Additionally, the power law distribution with exponential cutoff with parameters $\alpha = 0.84$ and $\chi = 10^{-4}$ presented in [24] is used to model Youtube content.

II.2 Channel models

In this work, the channel model allows us to evaluate the performance of the CDS server in different scenarios. The file size distributions in the previous section allow us to model the input to the CDS server. As stated, the CDS creates a carousel with the input files and will start pushing FLUTE packets to the network. Since the channel does not provide any guarantees on the delivery of the packets, it is expected that the service clients can miss some packets. However, in order to recover a file successfully, a client will need to receive all the packets of a particular file. Carousel retransmissions represent an error recovery mechanism, because clients can receive the packets they have missed in the first file transmission (after they joined the channel) in the following carousel cycles. Additionally, the service uses another error recovery mechanism: AL-FEC encoding, aimed at reducing the total number of carousel cycles needed to recover a file.

In order to simulate carousel transmissions in channels with losses, first it is necessary to model the packet loss at application layer. In most communication networks, this losses are related to the robustness of the underlying physical layer against different effects of the channel, like interference or signal fading. However, for complex simulations it becomes necessary to develop simple models for the packet losses at application layer, that simulate the performance of the underlying physical layer and the communication channel.

On the other hand, the use of AL-FEC encoding affects the way the reception of new packets is modeled. The basic principle of AL-FEC encoding is to add redundant packets to the transmission of each file. Hence, the number of packets successfully received in every cycle increases if AL-FEC is applied. However, with AL-FEC it is not necessary to receive all the original and redundant packets that are sent in every cycle. Instead, it is enough to receive the number of original packets multiplied by the inefficiency ratio. For modern AL-FEC codes, the inefficiency ratio is very close to one, meaning that the number of packets needed to decode the file is slightly higher than the number of original packets. This way, the number of cycles needed is, in general terms, lower when AL-FEC is used. Below, we will present the model for packet losses and following, we will present two different models for the number of cycles, with and without AL-FEC.

II.2.1 Model for packet losses

Packet erasure channel models are communication channel models that simulate the packet losses for packet-oriented transmission services [29]. These models are widely used in wireless network simulation environments. Within packet erasure channel models, generative discrete channel models produce endless binary sequences where zeroes represent correctly received packets and ones represent erroneous (erasure) packets. Generally, these models are able to replicate the stochastic properties of packet errors in a communication channel, like the average number of packet losses and the average burst (consecutive errors) size.

For instance, a simple packet channel model consists of a random sequence generator with linear distribution in the interval [0,1]. In the sequence, a random number greater than the average packet loss will produce an erroneous packet (1) while a lower random number will produce a correctly received packet (0). This model would allow us to simulate a channel with a certain average packet loss, although it will not be possible to configure the model to provide a given average burst size.

On the contrary, the two-state Markov model is able to produce erasure sequences with a given average packet loss and average burst size [29]. This model is widely used to simulate channel losses in wireless broadcast networks and has been used in previous studies to model the burst losses in broadcast networks [30].

The two-state Markov model is based on the Markov chain. The Markov chain assumes that a system can be completely described by a finite number of states. Each state is characterized by the probability of the system being on that state and the transition probabilities to change to a different state in the chain. As its name suggest, the model is based on two states the *Good* state and the *Bad* state [29], both generating errors at state dependent rates *1-g* and *1-h* respectively, as depicted in Figure 13:

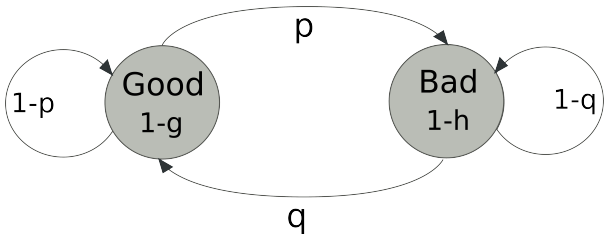

Figure 13. The two-state Markov model

Hence, the average packet loss rate, p_e and the average burst size, b_s, are provided by the following expressions:

$$p_e = (1-g)p_G - (1-h)p_B = (1-g)\cdot\frac{q}{p+q} + (1-h)\cdot\frac{p}{p+q} \qquad (7)$$
$$b_s = 1/q$$

where p_G is the probability of the model being in the good state and p_B is the probability of the model being in the bad state. This way, it is possible to model a certain erasure channel with a two-state Markov model, by adjusting the parameters of the model to match the characteristics of the channel. In this sense, there is a simplified two-state Markov model known as the Simple Gillbert model [29], where k is equal to 1 and h is either 0 or 0.5. Since the Simple Gillbert model only has two degrees of freedom, it is possible to adjust its parameters to a channel by measuring the average packet error rate, p_e and burst size, b_s of the errors of a packet trace through the transmission channel.

The main drawback of the two-state Markov model is that, despite it is adjusted to provide the average packet error rate and burst size of an error rate, it might not match other statistic properties of the error sample. In some cases, it is interesting to be able to model higher order statistical properties of the channel losses and with this motivation, the literature describes more sophisticated models that provide pattern losses better adjusted to specific wireless environments. For instance, [31] presents a comparison of the accuracy of different finite-state channel models in replicating the variance of the error burst length in mobile broadcasting systems. In their research, the authors show that such second order

statistical properties of the losses in wireless mobile television systems are not well modeled by the Gillbert model. Moreover, their research also highlights that the parameters of these models are highly dependent on system aspects like the configuration, the environment or the signal level. Another related literature shows that this conclusion can be extended to other systems. However, higher order statistical properties may be relevant or not depending on the application. For instance, the variance of the burst size is very important for real time streaming applications but, as reflected in [32] and [33] the Markov model provides accurate losses patterns to model file carousel transmissions over erasure channels.

For this reason, in this thesis the two-state Markov model is used to provide the channel loss patterns characteristic of the scenarios under study. Recalling that the channel model is used to estimate the number of cycles needed to recover a file, the next section provides the model used to obtain the number of cycles, whereas section II.2.3 provides empirical data that shows that the channel model is valid for the cases under study.

II.2.2 Model for carousel cycles

As stated, if there are losses, it is very likely that clients need several cycles in order to download the file. 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 [34]. Eq. (8) models the probability of receiving exactly x new packets in a loop using a hyper-geometric probability distribution when no AL-FEC is used. 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.

$$P(x,m,k,l) = \frac{\binom{m}{x}\binom{k-m}{(k-l)-x}}{\binom{k}{k-l}} \tag{8}$$

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 x new packets within the m missing packets, out of the received k - l packets. Applying this hyper-geometric probability distribution, the expected number of packets received at loop i is:

$$x(i) = \sum_{\xi=0}^{m} \xi P(\xi, m, k, l) \tag{9}$$

That is, the expected number of new packets received is the expectation value of receiving x new packets out of the m missing packets. Finally, the number of cycles needed to download a file is calculated as:

$$c = \min(i) / x(i) = k \tag{10}$$

If AL-FEC is used, the probability of receive x new packets in a new loop can be modeled in a similar way, although there are some changes in the equations. Thus, eq. (11) - describing the probability of receiving x new packets at cycle i with AL-FEC- is slightly different to eq. (8). 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:

$$P(x, n, r, l) = \frac{\binom{n-r}{x}\binom{r}{(n-l)-x}}{\binom{n}{n-l}} \tag{11}$$

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 without errors 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:

$$x(i) = \sum_{\xi=0}^{n-r} \xi P(\xi, n, r, l) \tag{12}$$

Finally, the number of cycles needed to download a file when AL-FEC is applied is:

$$c = \min(i) / x(i) = k \cdot inef\_ratio \qquad (13)$$

where *inef_ratio* is the AL-FEC encoding inefficiency ratio. Therefore, with these formulas it is possible to calculate the number of cycles needed to download a file, depending if AL-FEC is used or not: the expected number of new packets received per loop is calculated iteratively - using an hyper-geometric probability distribution - 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 which of the transmitted packets (*k* when no AL-FEC is applied and *n* when AL-FEC is applied) are correctly received and which are erroneous packets. The parameters of the Markov model are adjusted to match the statistical properties of error traces measured in the cases under study. Further details about this model are provided in [33].

II.2.3 Calibration of the model with measurement data

In this section, we have used error traces collected in laboratory conditions to calibrate (and validate) the channel model. The measurements also evaluate the number of cycles needed to download a file, in order to check the accuracy of the model for content download services.

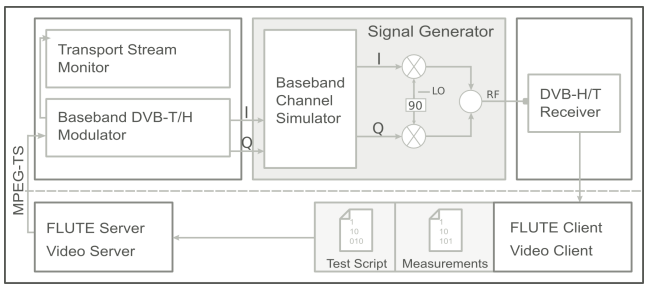

Figure 14. Measurement setup.

Figure 14 shows the setup of the measurement equipment. All service layers except for the physical layer are implemented in software, both at the server and at the client side.

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. It is worth mentioned that the FLUTE library has been implemented by the research group developing this thesis work.

Hence, the server side generates the data to be transmitted to a baseband modulator. The baseband modulator generates an output according to the DVB-H (Handheld) specifications [35].

The most outstanding aspect of the measurement setup is the usage of the baseband channel simulator of an Arbitrary Waveform Generator. The channel simulator is used to modify the received signal according to the effects introduced by wireless interfaces and mobility: noise interference, multipath and the Doppler effect typical of mobile channels. Moreover, the channel simulator allows to configure the Carrier to Noise Ratio (CNR) at the receiver. Additionally, the channel simulator allows to generate multipath taps, which consist of delayed echoes of the signal. It is possible to configure the time offset and the amplitude of the taps, as well as their Doppler phase shift, depending on the relative speed between transmitter and receiver. By configuring these parameters, it is possible to emulate many different reception conditions, such as fixed, pedestrian or mobile reception.

Later, the receiver demodulates the radiofrequency signal received from the AWG, using a DVB hardware receiver connected to a computer. The software implements the FLUTE client, customized to produce the measurement results. The automation of the measurement procedure is achieved by allowing the client to reconfigure the parameters of the server. The client software controls the server through a control channel that is not part of the service architecture, while the parameters for the measurements 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 respective characteristics.

Figure 15 and Figure 16 show the results obtained in the measurement trials. The channel simulator applies a TU6 channel model [36], 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 and bad reception. According to [37], a packet error rate of 5% is the maximum error rate that still provides an acceptable Quality of Experience for streaming services. Therefore, in this study, good mobile reception represents a TU6 mobile channel with a CINR level that provides an average error rate of 5%. Additionally, we have defined bad mobile reception condition, characterized by a 50% packet loss rate.

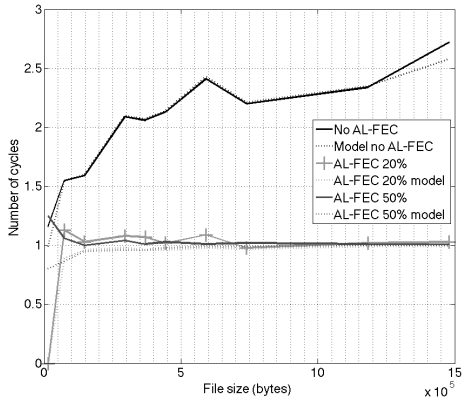

Figure 15. Number of cycles obtained under good reception conditions.

Figure 15 shows the number of cycles against the file size for different configurations of the AL-FEC block: No AL-FEC parity, 20% AL-FEC parity (CR=5/6) and 50% AL-FEC parity (CR=2/3). The figure gathers the results obtained with the measurements and with the channel model, calibrated according to the error traces under good reception conditions. As shown in the figure, model and measurements provide very similar results in most of the cases.

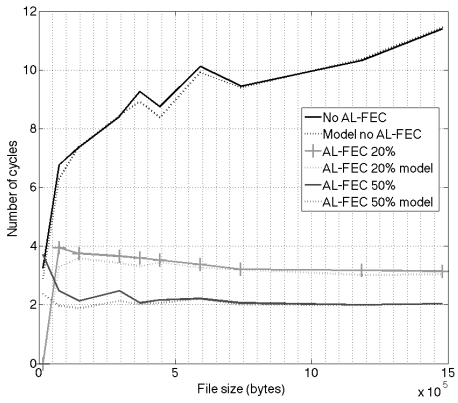

Figure 16. Number of cycles obtained under bad reception conditions.

In a similar way, Figure 16 shows the number of cycles obtained under bad reception conditions. Again, the results obtained from the measurements and the ones obtained with the channel model provide very similar results. With these results, it is clear that the models provide very similar number of cycles to those one could expect in a real DVB-H system. Hence, the models can be used in software simulations to generate results difficult to obtain in real systems (or even in laboratory conditions). This is the case of the evaluation of CDS in large service areas, with several concurrent service users. For this kind of simulations, it is necessary to use a model able to model the client population as well.

II.3 Client models

Up to this point, the analysis has covered the models for the server and the channel. This section describes the models used at the client side. The next subsection describes the models used for the background CDS client application, i.e. the application running on the client mobile device and in charge of downloading content to local storage. This thesis also analyzes how a background CDS can reduce the traffic load of a VoD service and for that purpose, a model for the user requests becomes necessary. This model is referred to as the user model and is presented in the second subsection of the client model.

II.3.1 Models for the background CDS client application

Back to the architecture in Figure 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 [5]. 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, Section VI.1.3 explains how a simple recommender 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 may be smaller than the sum of the size of all files in the carousel, the storage management must decide which file data maximizes the overall value of the files in cache. Therefore, the cache management needs to find the decision vector $Y = \{y_1, y_2, ..., y_n\}$ that maximizes the value of files in cache. Assuming that the cache management only keeps entire files, 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:

$$\text{Find } Y = \{y_1, y_2, ..., y_n\} \quad y_j \in \{0,1\} / \quad \text{maximize } \sum_{i=1}^{n} y_i v_i, \quad \sum_{i=1}^{n} y_i s_i \le s_A \tag{14}$$

Clearly, this is an instance of the 0-1 knapsack problem, thoroughly studied in the literature [7]. 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 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:

$$\begin{aligned} &1.\quad \text{Sort } I^k \text{ such that } \frac{v_j}{s_j} \geq \frac{v_{j+1}}{s_{j+1}} \\ &2.\quad \text{Find } i_m = \min(i : \sum_{j=1}^{i} s_j \geq s_A) \\ &3.\quad y_j = 1; j < i_m \text{ and } y_j = 0; j \geq i_m \end{aligned} \tag{15}$$

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

Regarding the definition of value, 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 [4], [6], [38]. Besides the utility, the definitions of value used in this thesis account for other parameters of the file, indicated in the table below:

TABLE I
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	$v_j = \tilde{p}_j \cdot t_C^j / s_j$

All the cache replacement policies estimate v_j as a function of the utility $\tilde{p}_j$. In the first policy, referred to as the *P* policy, the value of a file in cache is equal to its utility. It is important to highlight that the *P* policy has been defined in this thesis, as opposed to the other two policies. The PIX policy is defined according to [4]. This study states that, for caches of non-uniform accessed broadcast data, the value of storing a file in memory is directly proportional to the future access probability of a file (here its utility $\tilde{p}_j$) and inversely proportional to its relative transmission frequency (i.e. the inverse of its carousel cycle period t_C^j). In this policy, files with longer carousel cycles are considered more valuable because clients need more time to download them. Moreover, [38] states that, as in web caches, the size of the files should also be taken into account in the definition of its value. This policy considers that, although the utility of a large file may be higher than the utility of several smaller files, the summation of all cache hits produced by the small files may be higher than the cache hits of the large file. Therefore, smaller files should have a larger value in cache, as accounted by the PIXS.

II.3.2 User model

Section II.1.3 presented different methods to estimate the popularity of content items belonging to the same content catalogue. The models were obtained by analyzing the historical data of the requests issued to each item. This historical data is available for popular VoD services. However, historical data is not enough to model how users requests are distributed in time.

The literature presents different approaches to address this issue. In [18], the authors collected data from the requests to a popular VoD service generated across the university campus network over a three months period. This approach provided very accurate models for the requests for a large content catalogue, issued by a relatively small and homogeneous population of users. Oppositely, the authors of [23] mined the data of two content categories of popular video sites over time. Thus, they obtained a good model for requests produced by large populations over a homogeneous content catalogue. Regarding the distribution of video requests in time, both studies analyze data with a time granularity of a day. Let us provide a brief overview of their results.

The analysis of the requests generated per day in [18] shows the dependency of the activity of the VoD service on the daily habits of users – since the test population mainly consists of university students and staff, the activity is considerably lower during the weekends. Furthermore, the results also show how the number of user requests slowly increases week after week, as the VoD service becomes more popular.

[23] presents a model for the inter-arrival time of requests based on the daily number of requests per day that they measured. The study assumes that within the day, the requests are exponentially distributed. It follows that the mean inter-arrival time of requests, that is, the average time between the occurrences of two consecutive requests, is equal to $1/\lambda$, where λ is the intensity of requests per unit of time. Hence, the occurrence of requests is a discrete statistical variable that follows a Poisson distribution with parameter λt (where t represents the time period, 1 day in the study). This way, by measuring the number of content request over a period of time it is possible to measure their intensity λ.

Besides the distribution of requests, [23] analyses how the popularity of videos changes over time. The paper shows that the popularity of most new user generated content items rapidly decreases over time, while older videos are more stable in the rankings. It is worth noting that in the ranking system used in the study, videos that are not watched during one day are penalized with 2000 positions in the ranking. This means altogether that the popularity of high ranked videos is very steady over time. Additionally, the study shows

that few new videos increase their popularity very fast, indicating that a small percentage of new videos make it to the popular list. Another interesting finding is that there is a high correlation coefficient between the number of views after two or three days and in the long run.

Now, we are going to present the time model for user requests used in this thesis. The model is based on the measurement of the long-term popularity of a content catalogue and relies on the findings of the aforementioned studies. First, as in [23], it is assumed that the occurrence of requests follows a Poisson distribution. Also, it is assumed that the number of requests per unit of time can be obtained from the number of access measured over a period of time. In our approach, the total number of requests to the service over a period of time t is fixed to a given value, namely λ. Then, the rate of requests to file j, λ_j is calculated as:

$$\lambda_j = \lambda \cdot p_j \tag{16}$$

where p_j is the relative popularity of item j (as in the section Content popularity). Note that this methodology is equivalent to that in [23], due to the properties of the Poisson distribution: the superposition of different Poisson processes of rates $\lambda_1, \lambda_2, ..., \lambda_n$ is another Poisson process of rate $\lambda = \sum_{j=1}^{n} \lambda_j$. For our study, the important fact is that, with this approach, it is possible to model user requests for different files without a measurement of actual requests, but with a measurement of the content popularity. For instance, bear in mind that the estimation of the popularity obtained from the IMDB used user ratings instead of number of views. In this sense, both [18] and [23] show there is a strong cross correlation between the rankings and the number of access. Hence, the relative popularity estimated from the rankings is equivalent to that of the number of views and can be applied to eq. 16.

One of the main drawbacks of the approach followed in [23] is that, clearly, the request rate is not constant with time. For instance, there is evidence of this in the trace of user requests

per day provided in [18]. To overcome this issue, in our model, it is possible to use a non-homogeneous Poisson process for the user requests with rate $\lambda(t)$. In any case, the main drawback of our model is that it does not account for the variations in the popularity over time. This time dependency is evidenced in the results of [23], but, as stated, popularity is in general very steady, especially for the most popular items of a catalogue.

Finally, as in previous sections, we provide some results provided by the model.

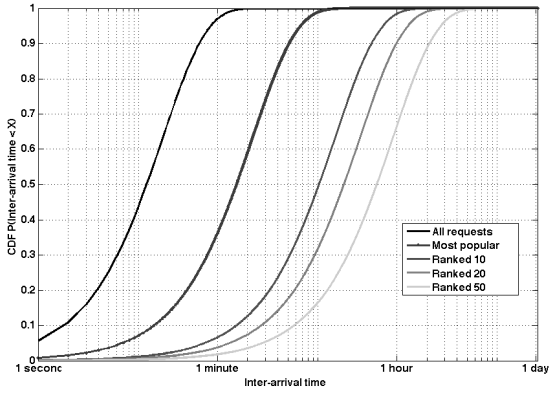

Figure 17. CDF of the inter arrival time for different content items of a catalogue of 100 files and ZIPF distribution (α=0.83).

Figure 17 shows the effect of the popularity on the inter arrival time of user requests to different files. In the figure, the number of requests over a period of one day is set to 5,000, providing an intensity of $\lambda=5{,}000/(24\cdot60\cdot60)=0.0579$. The file catalogue is consisted of 100 content items and the popularity follows a ZIPF distribution (α=0.83). The position in the ranking affects to great extent the inter-arrival times. For instance, let us compare the maximum inter arrival time of requests, which corresponds to the point in the figure when the CDF reaches 1. The maximum inter arrival time of all requests is slightly over 1 minute, meaning that the time between two consecutive request is in always less than 1 minute. On the other hand, the maximum inter arrival time of the requests to the most popular item is approximately 10 minutes. On the other hand, the maximum inter arrival times of files 10, 20 and 50 in the ranking are respectively 1 hour, 2 hours and 4,30 hours.

Hence, as expected, the inter arrival time in our model increases drastically for files with low popularity.

II.4 Conclusions

This chapter has presented the models used to characterize the content, the communication channel, the storage management and the generation of user requests. These models are used in the following chapters to conduct the simulations for the evaluation of background push CDS.

Regarding the content, we have presented models for television content and User Generated Content (UGC). The model for television content has been developed in this thesis using a database for television programs available on the Internet, the IMDB. On the other hand, the model for UGC used is presented in [18].

In order to model the file size of television content, we have used the same methodology used in previous studies to model the file size of web content. This methodology is based on adjusting the parameters of probability distributions to fit the statistics of a representative sample of file sizes. However, instead of sampling directly the file sizes, we have taken samples of program durations. This allows to generate different file size distributions depending on the encoding rate of a given application (e.g. mobile television or HD television). The results show that the same probability distributions regarded for file sizes can be adjusted to program durations. With this, we have obtained two different probability distributions from the program duration dataset, the lognormal (LN) distribution and the Double Pareto Lognormal Distribution (DPLN). The results show that the DPLN distribution is a good approximation to the actual program durations in the database, regardless of the number of files used by the simulation. On the other hand, the LN distribution is a good and relatively simple approximation program duration datasets and its performance is very close to the DPLN distribution when the number of files needed by the simulations is small (10-100). Since the number of files used in the following simulations is in this range, the LN model is used to generate file sizes of television programs for later studies.

Regarding the popularity, we have presented two different models for the popularity of the content in the IMDB. Again, we have used a methodology well established in related works, based on adjusting parametric distributions to an empirical measurement of the popularity. We have regarded two different distributions, the ZIPF distribution and the power law with exponential cutoff distribution. We have evaluated the error produced with both distributions. Although both distributions model very well the popularity of most popular items, the power law distribution with exponential cutoff provides better results and is therefore used in the simulations below.

As for the channel model, we have developed a model for the channel losses, i.e. the Markov model. Later, the model is used to calculate the number of carousel retransmissions needed to download a certain file. The model has been validated through measurements in laboratory conditions, proving its accuracy. Clearly, data collected in a particular environment (or set of environments) through field trials would produce more accurate results. Instead, laboratory measurements provide more general results, applicable to a wider range of environments, at the expense of loosing accuracy in a particular scenario. This consideration is taken into account in the analysis of the results of the simulations using the models.

Finally, we have presented models for the storage management application and the user requests. The model for the storage management allows us to simulate the decisions made by the storage management, that is, which files should be kept in local storage at anytime. The model is flexible, allowing to modify different aspects of the client application, like the popularity estimation or the definition of value used in the decision making algorithm. These are important considerations in the design of the client application and for this reason, simulations in following chapters assess the impact they have on the service performance.

Furthermore, we have presented a model for the generation of user requests. The model takes into account the relative popularity to each file, so that it is possible to split the total number of requests into the requests issued to every file. This model allows to simulate on

demand services and the ability of the background push CDS to prefetch content and store it locally.

Chapter III

Opportunistic Insertion of television services

The main objective of this chapter is to evaluate the potential use of Opportunistic Data Insertion (ODI) for the delivery of large media files using background Content Download Services (CDS) over first-generation and second-generation video broadcast networks. Some previous research works on ODI focus on the provision of IP services with instantaneous QoS requirements [39] (disregarding CDS delivery), while other related works focus on the performance of CDS services over broadcast networks [34], [40], [41], [42]. On the other hand, the use of the residual capacity for the delivery of content has been investigated for cellular networks in [43].

The multiplexers in video broadcast networks use filling packets to achieve the constant bitrate of the transmission mode used by the network. The multiplexer needs to accommodate the different video services, guaranteeing that the resulting multiplex rate is constant and that the multiplexing process does not degrade the quality nor the synchronization of the elementary media streams. In case that the summation of the rates of the different video services is lower than the multiplex rate, the constant multiplex rate is achieved by inserting NULL packets, that is packets with no useful payload. Multiplexers use different techniques to optimize the efficiency of the multiplexing process, so as to minimize the insertion of NULL packets. In this sense, ODI consists of using these NULL

packets with packets from some datacast service, which does not have such strict timing requirements as streaming services. In this thesis, ODI is used to insert the packets belonging to the background CDS services into the broadcast Transport Stream.

In order to provide a better understanding of ODI, we first describe first-generation Digital Video Broadcasting (DVB) networks in III.1. Later, III.2 presents a brief description of DVB networks and the timing model used in Moving Picture Experts Group (MPEG) systems. System components, such as an multiplexer, use this timing model to learn the timing dependencies between the different streams and it is therefore crucial to understanding why a multiplexer needs to insert filling packets.

Later, III.3 provides a brief description of MPEG encoding and Constant Bit Rate (CBR) multiplexing. After describing how ODI works in a DVB multiplex, we provide an empirical study of the rate available for ODI in different commercial MPEG Transport Streams (MPEG-TS). The section gathers measurement traces of the filling bitrate found in the different networks. Later, these measurements are used in chapter Chapter IV to develop an analytical model for the long-term bitrate available during the download of files.

Finally, sections III.4 and III.5 describe and evaluate the use of opportunistic insertion in second-generation DVB networks. In second-generation DVB networks, the overall capacity provided by the physical layer is divided into logical tunnels. Again, due to the timing requirements and the variable rate of streaming media, there is an excess of capacity in these tunnels that can be used by background services. This use case is addressed in the last section of this chapter, describing how the ODI principle can be applied to second-generation DVB networks. Again, the section includes some empirical results of the bitrate available for ODI in this scenario.

III.1 First Generation terrestrial DVB Networks (DVB-T)

First-generation networks, like Terrestrial DVB [44] (DVB-T) Networks, are used to deliver television and radio services in wide areas. Figure 18 shows the main system

components of a DVB transport network based on DVB-T. First, the video and audio samples are encoded, generating separate MPEG [45] Elementary Streams (ESs). The ESs enter the multiplexer, together with the metadata, referred to as Program Specific Information / Service Information (PSI/SI) tables [46]. Additionally, the multiplexer inserts network information in the form of Mega-frame Initialization packets (MIPs), used to synchronize the different transmitter nodes in the network [47]. The adapter connects the resulting MPEG-TS with the primary distribution network, which is the network between the multiplexer and the transmitter stations. Later, the transmitters broadcast the MPEG-TS according to the timing information indicated in the MIP packets.

At the receiver, the frontend demodulates the MPEG-TS, the demultiplexer filters the packets of the different ESs of the program and finally, the decoders generate the sequence of video and audio samples from the encoded streams.

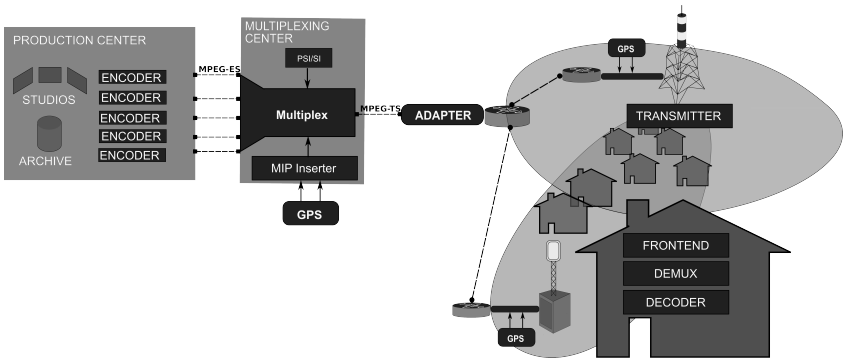

Figure 18. Main components in a DVB network.

As mentioned in the introduction of this chapter, first-generation DVB networks broadcast constant bit rate MPEG-TSs [45]. The DVB standards provide different options for the configuration of the parameters of the physical layer. However, regardless of the configuration, the system always delivers a constant rate MPEG transport stream. This way, network operators have different physical layer configurations that trade off bitrate capacity with features like coverage or mobility. This framework allows DVB networks to adapt to different scenarios and applications. For instance, Figure 19 illustrates the trade-offs in the

physical layer configuration of DVB-T networks. In the figure, GI stands for Guard Interval, which is a preamble added to every modulation symbol to improve resilience against multipath fading. Long GIs enable the reception in environments with long delay spreads, for instance portable urban reception, but as seen in the picture, at the expense of lower net MPEG-TS bitrates. On the other hand, 64-QAM, 16-QAM and QPSK stand for the constellations used in the modulation of data carriers, providing 6, 4 and 2 bits per carrier, respectively. It can be seen that low order constellations require low levels of received CINR, thus providing larger cells with the same transmission power. However, they provide significantly lower net bitrates.

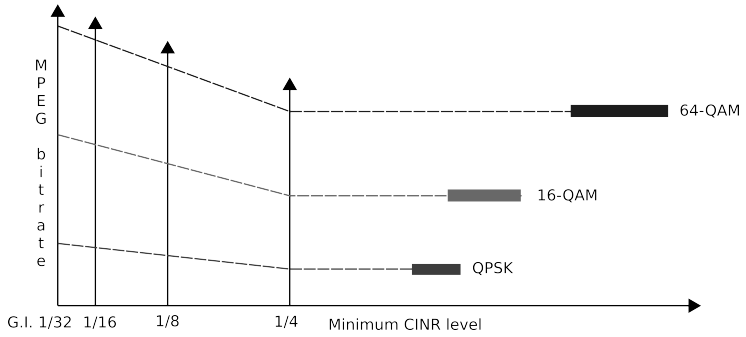

Figure 19. Trade offs of DVB-T physical layer configurations.

Moreover, the exact useful net data rate R_U (Mbits/s) bitrate of the modulation is defined in [47] as:

$$R_U = R_S \cdot b \cdot CR_I \cdot CR_{RS} \cdot (T_U / T_S) \qquad (17)$$

where R_S is the DVB-T symbol rate (6,75 Msymbols/s), b is the number of bits per carrier (2 for QPSK, 4 for 16-QAM and 6 for 64_QAM), CR_I is the inner FEC code rate (either 1/2, 2/3, 3/4, 5/6 or 7/8) and T_U/T_S is the ratio between the useful symbol duration and the symbol duration (4/5, 8/9, 16/17 or 32/33) depending on the GI. In any case, regardless of the configuration of the physical layer, the useful data rate is always constant.

III.2 MPEG Transport time model

It is clear that the delays of the intermediary blocks in DVB networks (e.g. the encoders, multiplexers, transmitters or receivers) may not be constant for every MPEG-TS packet (nor for every ES). However, MPEG transport systems are based on a constant delay timing model: the video or audio samples entering the encoder leave the decoder after a time delay that is constant for every sample. This model guarantees that the play-out rate of the decoded samples is exactly the same as the sampling rate at the video source. Therefore, it is necessary to provide a mechanism to inform the different system blocks about the temporal information of every packet in the stream. In MPEG systems, this temporal information is defined by means of the Transport - System Target Decoder (T-STD), which is an ideal decoder used to determine the temporal information of every byte in the transport stream. The T-STD model is presented in Figure 20:

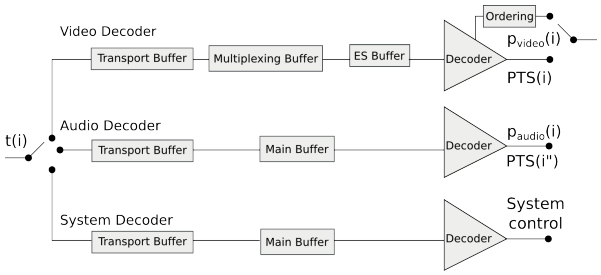

Figure 20. Transport System Target Decoder (T-STD) model of MPEG Transport Systems.

The T-STD represents an ideal implementation of an MPEG audio and video decoder. The T-STD helps ensuring that an MPEG-TS is compliant with the MPEG timing model and therefore, the different ESs of a service can be decoded without problems in a real decoder. In the T-STD, the incoming MPEG-TS packets are de-multiplexed and they enter the corresponding decoder (Audio, Video or System) Transport buffer. Note that the Audio and System decoders have a Transport Buffer and a Main buffer, while the Main buffer of the Video decoder is divided into the Multiplexing buffer and the ES buffer. Also, the video decoder has an ordering function. This is because video frames may not be presented in the

same order as they are decoded, since the encoder may use the information in both earlier and later frames to encode certain frames.

The timing model states that byte with index *i* inside the stream must be present at the input of the ideal decoder at time *t(i)*, which is referred to as the target decoding time of byte *i*. Similarly, each frame has a Presentation Time Stamp (PTS), which indicates the time at which the frame must leave the corresponding ideal decoder.

The time reference for all timing information is the MPEG System Clock, which has a frequency of 27MHz. The MPEG-TS must contain samples of its 27MHz System Clock in the MPEG-TS. These samples are referred to as Program Clock Rate (PCR). Each PCR is an Integer number representing the number of ticks of a counter synchronized with the MPEG System Clock. The PCR is included in the header of some MPEG-TS packets. This way, the value *PCR(i'')* represents the target decoding time *t(i'')* of the last byte conforming the PCR value in the corresponding MPEG-TS packet header field. Thanks to these embedded timestamps, other components in the system (p.eg. the decoder or the multiplexer) can calculate the target decoding time of each byte by interpolation:

$$t(i) = \frac{PCR(i'')}{27Mhz} - \frac{i - i''}{transport\_rate(i)} \tag{18}$$

where $i'' > i$ and *transport_rate(i)* is the instantaneous transport rate of the MPEG-TS, for byte *i*, calculated as:

$$transport\_rate(i) = \frac{(i' - i'') \cdot 27MHz}{PCR(i') - PCR(i'')} \tag{19}$$

Figure 21 depicts the time model defined by the S-STD target decoding time. The time between two consecutive PCRs can be calculated as the difference between the count of cycles of the System Clock in the two PCRs (*PCR(i')- PCR(i'')*), divided by the System Clock frequency (27MHz). In the figure, this time is noted as Δt_{PCR}. Moreover, the same time can be calculated as the count of bits between the two PCRs, *i'* and *i''*, divided by the transport stream rate (Eq. 19). Note that the count of bytes is always an Integer number of

MPEG-TS packets, noted as *n*. Since the MPEG-TS packet size is 188 bytes, *(i'-i'')=n*8*188*. This calculation is noted as Δt_B in the figure. Ideally, these two calculations should be equal to the time elapsed between the occurrence of the two PCR samples, Δt, as shown in the figure. Therefore, the network components (e.g. the multiplexer or the decoder) can calculate the timing of the bytes in the transport stream.

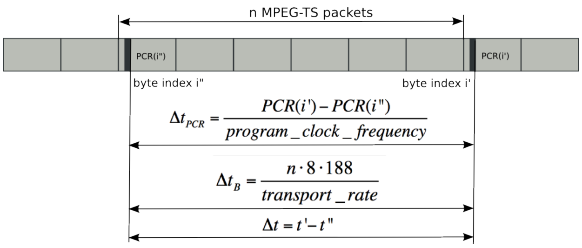

Figure 21. Time model based on the MPEG-TS S-STD.

In practice, any deviation in the arrival time of the PCR samples can introduce errors in the timing model. Furthermore, the multiplexer needs to modify the rate of the transport stream without introducing significant errors, which is why it is necessary to establish limits for the deviations in the timing model. The standard recommendation DVB Measurement Guidelines (DVB-MG) [47] defines quality measures for the precision of the timing information in MPEG-TSs. Among the different measurements defined, the most relevant are:

- **PCR Accuracy:** it represents the time precision of the PCR samples with respect to their byte position inside the MPEG-TS. Relating to Figure 21, the PCR accuracy is measured as the difference between Δt_B and Δt_{PCR}. The MPEG-TS specifications establish a limit of 500ns for the PCR accuracy. The PCR accuracy is affected in the multiplexing, remultiplexing and demultiplexing processes and thus, it is used to assess the quality of all sorts of multiplexing devices.
- **PCR Overall Jitter:** it represents high frequency components in the spectrum of the System Clock reconstructed from the PCR samples. The PCR Overall Jitter is obtained by comparing the frequency of the reconstructed System Clock with an

accurate reference. Moreover, the PCR Overall Jitter includes all timing errors introduced in the transmission chain at the input of a system component and it affects the System Clock acquisition in MPEG-TS devices.

The T-STD provides an ideal timing model, but in real implementations, a simple interpolation may not be enough to recover the timing information of the bytes in the MPEG-TS. In these cases, in order to obtain the System Clock of the encoder, intermediary devices can use a *Phase Locked Loop* (PLL), as shown in Figure 22:

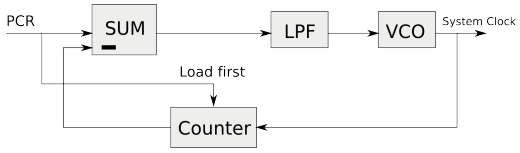

Figure 22. MPEG-TS System Clock acquisition with a Phase Locked Loop.

Thanks to the PLL, the PCR samples in the MPEG-TS are compared to the count of a local clock with a nominal frequency of 27MHz and an initial count value equal to the first PCR sample received. The local reference is kept in phase with the PCR count by correcting its frequency with every PCR sample received. As depicted in the figure, the instantaneous difference between both counters is passed through a low pass filter (LPF). The purpose of the LPF is to minimize the effect of fast changes in the PCR count, which could be due to jitter introduced by errors in the transport network. At the receiver, these fast changes can cause the decoder to loose the phase of the clock and produce artifacts in the decoded video samples and must be avoided. In fact, the 500ns limit in the MPEG specifications only applies to the high frequency components, since the PLLs can overcome slow changes in the frequency (frequency drift).

Up to this point, we have presented the time model of MPEG Systems. The next section will present how a Constant Bit Rate (CBR) multiplexer uses this model to maintain the timing information in the PCR during the multiplexing process. Additionally, the timing information is used to determine the insertion of filling packets, used to achieve the CBR requirement, providing the basis for Opportunistic Data Insertion.

III.3 Opportunistic Data Insertion in a DVB Multiplex

III.3.1 Constant Bit Rate and Variable Bit Rate MPEG encoding

A video source consists of a series of digital image samples called frames. The encoder reduces the size of the samples by taking advantage of both spatial (intra-frame) and temporal (inter-frame) redundancy. Spatial redundancy is the presence of repeated nearby information in the image. The best example is the presence of solid color areas. This way, the information of some image pixels can be encoded according to the information of adjacent pixels. On the other hand, temporal redundancy is the presence of repeated information in nearby frames. Hence, the information in some frames can be encoded according to the information in other frames. Depending on the pattern of the output bitrate, MPEG video encoders can be classified into two categories:

- Constant Bit Rate (CBR) encoders: A CBR encoder produces an ES with a constant transport rate, regardless of the redundancy in the image sequence. This means that, if the bitrate is set sufficiently high, the encoder will be able to achieve good picture quality even for complex sequences with little temporal or spatial redundancy. However, if the sequence is simple and has a lot of redundant information, the encoder will waste bitrate.
- Variable Bit Rate (VBR) encoders: VBR encoders use only the bitrate they need to encode a sequence, up to a fixed upper limit. Compared to CBR encoders, VBR encoders are able to exploit all redundancies of simple sequences and therefore they turn out to be more efficient. Moreover, since they adjust the bitrate to the complexity of the scenes, the picture quality turns out to be more constant than for CBR encoding. Furthermore, if the encoded picture quality is constant, the encoding method is referred to as constant quality encoding.

With this, coming back to Figure 18, the multiplexer will need to generate a CBR MPEG-TS bitrate from different contributions. It is likely that the addition of all the different bitrates do not match exactly the bitrate required by the physical layer configuration of the transmitter, which opens a window for opportunistic insertion. Next section discusses

briefly the details of the implementation of a CBR multiplexer with Opportunistic Data Insertion.

III.3.2 Constant bit Rate multiplexer with ODI

Figure 23 shows a CBR multiplexer with Opportunistic Data Insertion. In the figure, the grey shadowed area represents the actual bitrate used by the encoders and other (datacast) services. The black shadowed area represents the filling bitrate, not used by the encoder and therefore available for ODI. Additionally, the white areas represent the constant bitrate used for metadata (PSI/SI).

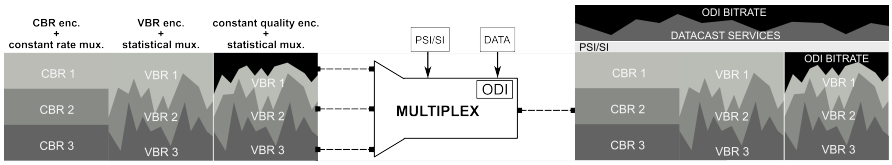

Figure 23. Constant Bit Rate multiplexer with Opportunistic Data Insertion.

The figure presents the different families of multiplexer implementations: CBR encoding with constant rate multiplexing, VBR encoding with statistical multiplexing and constant quality encoding with statistical multiplexing.

Regarding the CBR encoding with constant rate multiplexing, the implementation is rather straightforward and the multiplexer only needs to know the ratios between the input transport rates and the output transport rate to calculate the number of packets of each encoder that must be copied to the output buffer. Note that the timing information at the output MPEG-TS must be correct. Therefore, the multiplexer needs to rewrite the PCR samples so that the equations in Figure 21 hold at the output.

On the other hand, statistical multiplexing is a technology to mix together several VBR encoded streams, fulfilling the timing and bitrate requirements at the output. There are different implementations of statistical multiplexing and the technology has evolved significantly over time. In first generation statistical multiplexing, the pool of encoders are interconnected through a communication bus and negotiate the binary rate used by every

encoder, in order to deliver to the multiplexer a CBR stream made of several VBR streams. This configuration is used in the multiplexer noted as C57 in the measurements in section III.3.3. Another way to implement statistical multiplexing is to allow encoders to inform the multiplexer about the expected encoding bitrate of the forthcoming frames, using a communication protocol hereby referred to as statistical multiplexing protocol. The multiplexer weights the requirements of the different encoders and notifies each encoder of the available bitrate in the next time interval. At this point, the multiplexer can establish priorities between the different services, so that, in the event that the aggregated required bitrate exceeds the CBR limit, some services have better chances to get the requested bitrate. Later, the encoders apply the encoding rate indicated by the multiplexer. This allows each multiplexer to apply VBR encoding and the quality is only degraded if several encoders in the pool have complex sequences with few redundancies at the same time. This configuration is used in the multiplexer noted as C28. Normally, the management platform of the multiplexer at the head-end of the broadcaster lets operators to enter the minimum and maximum bitrates for statistical multiplexing in bits per second. Moreover, the priority of every stream is indicated as a percentage of the minimum allowed quality for a given stream, so that the highest priority stream gets the requested quality in case of contention. In these setups, the VBR encoders degrade the quality of the scenes in order to comply with the required bitrate.

Another interesting setup is to allow the pool of encoders to work with constant quality and let the bitrate to vary more freely, according to the complexity of the encoded scenes. In Figure 23 this is noted as constant quality encoder with statistical multiplexing. In this scenario, ODI insertion is more relevant because it allows encoders to work with an optimal bitrate, adjusted to the complexity of the stream, but at the same time, thanks to ODI, it uses the whole CBR capacity available in the broadcast network.

III.3.3 Measurement results

This section includes an estimation of the actual bitrate available for ODI insertion in commercial DVB networks. This estimation consists of measuring the bitrate of NULL packets in each multiplex. As mentioned, multiplexers that do not use ODI fill the

modulation MPEG-TS with NULL packets. Therefore, the NULL bitrate is the capacity available for a hypothetical deployment of ODI in these multiplexers.

The next results have been derived from measurement traces of 5,000 samples for each multiplexer. The different multiplexers under evaluation correspond to the three main national broadcast networks in Spain - the public broadcast network Televisión Española and the two main private networks, Atresmedia and MediaSet), plus the regional broadcaster in Televisió Autonòmica Valenciana. For each measurement sample, the number of null packets is measured every 4,060,800 bits of transport stream (21,600 MPEG-TS packets). This bit count corresponds to exactly 204ms of time at the useful net data rate modulation (Eq. 17) of the channels (19,91Mbps - 64-QAM, inner FEC=2/3, GI =1/4). The transport bitrate is measured from the PCR samples in the MPEG-TS according to Eq. 19. Later, the NULL bitrate at instant *i* is calculated from the ratio of NULL packets to the total number of packets as:

$$NULL\_bitrate_i = \frac{(null\_bits / all\_bits)}{tranport\_rate_i} \quad (20)$$

Table II summarizes the characteristics of each of the multiplexes under study together with the statistics of the NULL bitrate. The table shows the number of High Definition TV services (HDTV), Standard Definition TV services (SDTV) and digital Radio services included in each multiplex. Additionally, the table shows the main statistics of the Null rate found in each multiplex, the mean bitrate, the standard deviation (Std) and the minimum value found in the different measurements (Min).

TABLE II
CHARACTERISTICS OF THE DIFFERENT DVB-T MUXES UNDER STUDY

		Service List			NULL rate (kbps)		
		HDTV	SDTV	Radio	Mean	Std	Min
TVE	C58	0	4	2	1,586.6	898. 6	523.4
A3M	C46	2	3	0	1,052.2	250.5	0.0
	C43	1	4	5	928.0	159.7	693.0
	C67	0	6	0	576.0	266.1	44.2
MST	C40	1	3	0	1,688.4	181.3	1,511.4
	C68	0	5	3	893.0	249.5	678.3
RTVV	C28	1	3	2	457.0	65.6	213.8
	C57	0	5	2	632.2	94.1	258.0

The table shows that, in every multiplex, there are few hundred kbps available for ODI insertion. In some cases, the NULL rate exceeds 1Mbps. In general, the filling bitrate appears to be rather high. The standard deviation of the rate is, in most cases, much smaller than the mean rate and, except for one service, the minimum bitrate is always higher than zero.

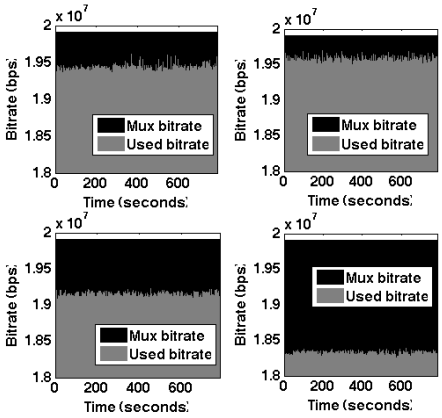

Figure 24. NULL bitrate available in four commercial DVB multiplexers.

Figure 24 displays the measurement trace for the multiplexers of two different network operators. The top row shows the multiplexer null rate and the used bitrate of the two

multiplex channels managed by the first operator. The multiplex to the top-left corresponds to channel C57, which is allocated to SD television and radio services, as indicated in the table. On the other hand, the figure to the top-right corresponds to channel C28, carrying the only HD service offered by the network. Similarly, the second row represents the multiplexers of another network operator. The bottom-left figure displays the rates in channel C68 and the figure to the bottom-right corresponds to channel C40. The figure shows that both operators have some filling capacity in their multiplexes. Also, the figure shows that in time scales in the order of tenths of minutes, the filling bitrate is rather steady.

In order to get a clearer picture of the statistics, Figure 25 and Figure 26 show the histogram of the NULL packets counted in every measurement time interval (204ms at modulation transport rate). The multiplexers under study have been grouped in two different figures according to their statistics. This classification is made for the sake of clarity, to use the same X and Y axis in all the plots in each figure. Figure 25 includes the statistics of C57 (top-left), C28 (top-right), C46 (bottom-left) and C67 (bottom-right).

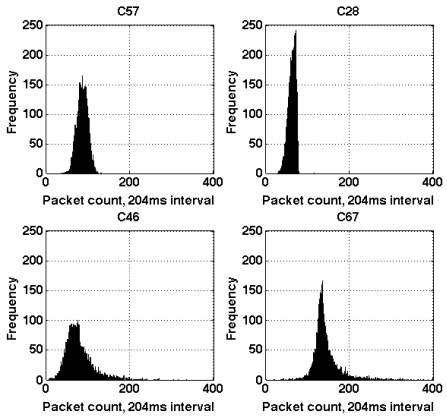

Figure 25. Histograms of the Null packet count in muxes with short tail distributions.

Clearly, the filling capacity in each multiplex has a different probability distribution. Each of the multiplexers transports a different combination of services and different network operators may use different multiplexing hardware. The results corroborate this, since the

figures show that the histograms for each multiplex are quite different. Figure 25 shows that the packet count of the four muxes therein has a clear body part and low frequencies in the tails of the distribution. On the other hand, Figure 26 shows the histogram for multiplexer C68, C40, C58 and C69. It can be noted that the tails in the distributions of these four channels have much higher frequencies than the multiplexers in Figure 25, especially C58. The four histograms resemble exponential distributions.

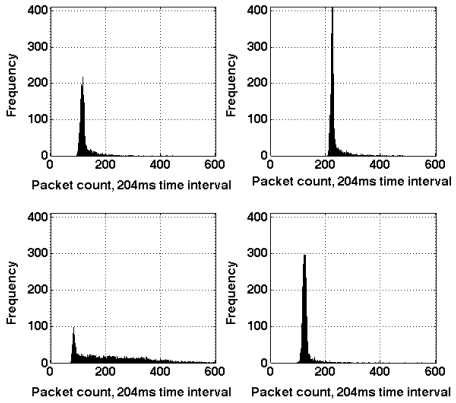

Figure 26. Histogram of the NULL packets in muxes with long tail distributions.

With this, it does not seem appropriate to try to draw a statistical model for the available bitrate in any multiplex. Instead, it is necessary to specify the multiplex under study. Then, as with previous system parameters (file size and access probability), it is possible to use the empirical data of the measurements directly on the simulations, or use parametric statistical models with similar properties as the measurements, in order to produce more general results.

In this sense, we have conducted a longer-term measurement on the bitrate available for ODI in channels C28 and C57, belonging to the regional network operator in Valencia, who participated in the study. The longer-term measurement of each channel consist of 61 measurements traces of approximately 5,000 measurement samples each. The procedure to obtain each sample is the same as with the previous studies. The tuner switches between

channels for every measurement trace, so that the total measurement time is approximately 36 hours in both measurements.

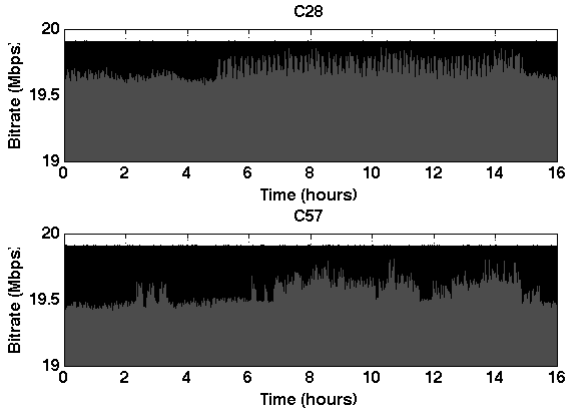

Figure 27. Longer-term NULL bitrate measurement in channels C28 y C53.

Note that, in order to see clearly the instantaneous NULL bitrate, the range of the Y axis is only 1Mbps, which is approximately 5% of the multiplex capacity. It can be noted that, in both measurement traces, the used capacity exhibits clear peaks above the mean value. However, as can be noticed in the histograms, it is rather seldom that these peaks occur. Hence, in general the average NULL bitrate is rather steady with time.

Now, let us analyze the histogram of the measurement traces. Figure 28 presents the normalized histogram of the bitrate of C57 together with the histogram of a normal distribution with mean $\mu=644.8kbps$ and standard deviation $\sigma=106.0kbps$, adjusted with the MLE method to fit the measurement data. The distribution passes the Paerson´s Chi-square goodness-of-fit test, meaning that the probability distribution of the measurement data is consistent with the normal distribution.

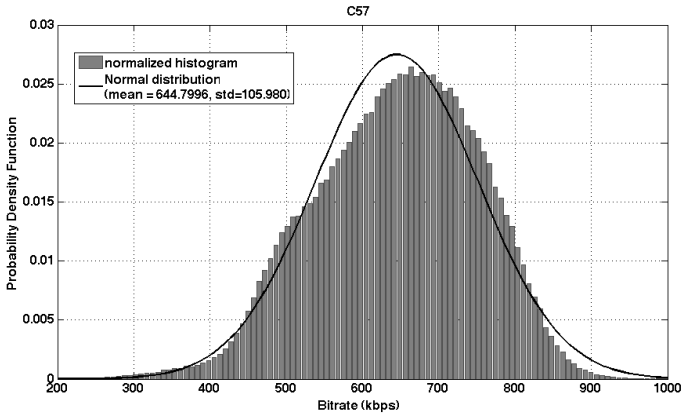

Figure 28. Histograms of filling capacity in channel C57 and its normal probability density function approximation.

It is worth noting that, out of the different muxes in Figure 25, C57 is the one better adjusted by a normal distribution. Clearly, the shape of the histogram resembles the PDF of a normal distribution. However, it is not always so straightforward to find a parametric statistical model that fits the null bitrate in a given multiplex. This is clearly depicted in Figure 29, where the normal distribution does not seem like a valid alternative to model the NULL bitrate in multiplex C28. For this reason, it is necessary to regard other alternatives. Among the different statistical distributions, the T-location distribution is useful to model statistical processes that have heavier tails than the normal distribution. The probability density function of the T-location distribution for the bitrate *b* is defined as:

$$pdf(b \mid \upsilon,\mu,\sigma) = \frac{\Gamma\left(\frac{\upsilon+1}{2}\right)}{\Gamma(\upsilon/2)\sqrt{\pi\upsilon\sigma^2}}\left(1+\frac{1}{\upsilon}\frac{(b-\mu)^2}{\sigma^2}\right)^{-\frac{\upsilon+1}{2}} \qquad (21)$$

where Γ is the Gamma function, υ the shape (also degree) parameter, μ is the location parameter and σ is the shape parameter.

As mentioned above, Figure 29 presents the normalized histogram of the NULL bitrate distribution in C28, together with the T-location distribution obtained with the MLE

method. The parameters of the distribution are υ=53.5kbps, μ = 467.8kbps and σ =56.74kbps. As shown in the figure, the T-location based distribution has similar kurtosis (that is, they are peaked in the same way) as the measurement data. However, the probability density function of the measurement data is skewed or asymmetric, while the T-location based distribution is symmetric. On the other hand, the Gamma distribution and the Generalized Extreme value distributions are asymmetric, but do not have the same level of kurtosis as the measurement data. In any case, the T-location based distribution is the only distribution that passes the Paerson´s goodness of fit test.

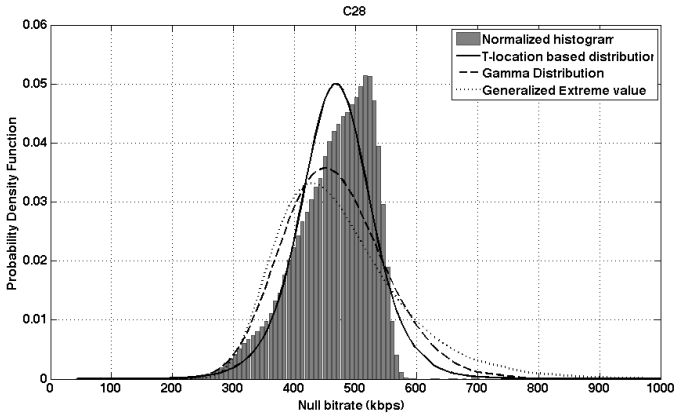

Figure 29. Histogram of filling capacity in channel C28 and its probability density function approximations.

As a summary of this section, the measurements conducted on different commercial DVB-T networks prove that the bitrate available for ODI is in the order of few hundreds of kbps, ranging from 457kbps to 1.64Mbps, depending on the multiplex. Apparently, the long-term average filling bitrate is rather steady in each multiplex. As for the short-term statistics, the results show that the bitrate in each multiplex has very different statistical properties. For this reason, in following simulations, it is necessary to specify the multiplex under evaluation. In this sense, we have derived simple parametric models from the statistics of the bitrate in two different multiplexes, in channels C28 and C57, managed by the same network operator. It appears that the NULL bitrate measured in C57 follows a normal distribution, while the NULL bitrate in channel C28 is well approximated by a T-location

based probability distribution. Both distributions pass the Paerson´s goodness-of-fit test. These models are used in following studies to simulate the bitrate available for ODI insertion in a DVB multiplex.

III.4 Second Generation terrestrial DVB Networks

As explained in previous sections, all services in a DVB multiplex share the same physical layer configuration. This can be a drawback if the services in the multiplex have different physical layer requirements. For instance, mobile television requires higher CINR levels than standard television, because the performance of antennas in mobile terminals is much worse than that of rooftop antennas. Additionally, mobile reception demands higher resilience against multipath transmission and the Doppler effect. Thus, as the operators offer different types of television services (p.eg. 3DTV, HDTV, mobile TV), service specific robustness becomes an important requirement for the network.

In first-generation networks, hierarchical multiplexing provides a basic mean for service specific robustness. Hierarchical multiplexing consists of dividing the total capacity of the transport network into two different MPEG transport streams, the High Priority (HP) MPEG-TS and the Low Priority (LP) MPEG-TS. The mapping of bits to data carriers is done such that the modulation applied to the LP stream is a robust QPSK, while the modulation applied to the HP stream is a less robust QPSK or 16-QAM.

However, hierarchical modulation is not flexible enough to provide service specific robustness to more than two types of services. For this reason, second-generation broadcast technologies incorporated more sophisticated mechanisms to provide service independent robustness to each service. The following sections describe these mechanisms for two broadcast standards: DVB-H and DVB-T2.

III.4.1 DVB-H networks

The DVB-H standard appeared to enable the delivery of mobile television services over broadcast networks. Strictly speaking, DVB-H is not considered a second-generation

broadcasting standard. However, it was the first broadcasting standard that incorporated mechanisms to provide service specific robustness.

The DVB-H physical and transport layers maintain backwards compatibility with the DVB-T standard. Thus, DVB-H networks broadcast a CBR MPEG-TS. However, video services are not transported directly in the transport stream. Instead, DVB-H defined an IP protocol stack [50] based on broadcast IP protocols. This way, the standard defines two different types of services: streaming services, using the Real-Time transport protocol (RTP) [51] and CDS services, using the FLUTE [1] protocol.

IP datagrams need some encapsulation protocol in order to be transmitted on top of the MPEG transport streams. In this sense, DVB-H IP datacast services use the Multiprotocol Encapsulation (MPE) [52] protocol to form an IP tunnel over the MPEG-TS broadcast stream. This way, service specific robustness is provided at the IP encapsulation level. On one hand, it is necessary to deal with the additional packet losses caused by the worst reception conditions in mobile communications. Mobile reception can be improved through additional Forward Error Correction (FEC) applied over IP datagrams, referred to as Multiprotocol Encapsulation FEC (MPE-FEC) [52]. On the other hand, battery consumption is constrained in mobile terminals. Consequently, [52] 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. As an example, Figure 30 shows the MPEG-TS bitrate used by a service with MPE-FEC and time slicing. The length of the MPEG-TS capture is 5.6 seconds. Each burst is approximately 1.5MB and the MPE-FEC encoding rate is 2/3, providing a useful burst payload of approximately 1MB. The multiplex bit rate is 29,273 kbps and the burst have an instantaneous bitrate of 20,154 kpbs, yielding an average burst duration of 63ms and an average burst cycle time of 867ms.

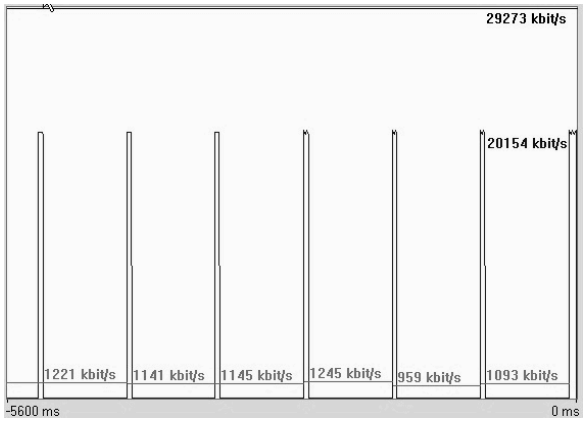

Figure 30. MPEG-TS bitrate of MPE-FEC service with time slicing.

Therefore, 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. The configuration of MPE-FEC and time slicing is crucial for the QoE of mobile TV. MPE-FEC trades off error resilience and effective capacity. Clearly, the more MPE-FEC parity is added to every burst, the more robust the service is against errors. However, the same IP burst payload occupies more MPEG-TS packets in the broadcast stream, reducing the effective bitrate in the multiplex.

Similarly, time slicing trades off battery life and zapping time, defined as the time needed to switch from one mobile television service to another. Regarding battery consumption, the battery used is proportional to the ratio between the burst duration and the burst cycle time. In the example, an ideal decapsulator implementation needs only to switch on the DVB receiver 7.2% of the time (63/867ms). On the other hand, the zapping time is related to time slicing and the configuration of the encoder. When a terminal tunes to a DVB-H service, first it needs to wait for a burst during an average time equal to half the burst period. Later, the encoder will need to fill the buffer for a time known as the initial buffer delay, required to guarantee seamless playback even if there are variations in the bit rate. The decoder might have to wait for a longer time in case it did not receive any intra-prediction frame during that time.

III.4.2 DVB-T2 Networks

The DVB-T2 standard [49] is a revision of the DVB-T norm. The primary objective of DVB-T2 is to provide broadcasters with a more advanced, more efficient alternative to previous broadcast standards. DVB-T2 is not meant to be an upgrade of DVB-T networks, but rather a complementary system that allows a more efficient usage of broadcast spectrum. While being able to share infrastructure with DVB-T [53], DVB-T2 provides improved coverage, higher capacity and better flexibility than its predecessor.

Regarding service specific robustness, DVB-T2 implements a quite advanced mechanism, known as Multiple Physical-Layer Pipe (M-PLP) mode. In this configuration mode, DVB-T2 physical layer frames are regarded as a grid of cells in the time-frequency domain, each cell representing a data carrier in an OFDM symbol. A PLP is composed of any arbitrary group of cells, making a virtual physical channel on top of the DVB-T2 frequency channel. Figure 31 shows a simplified schematic representing the PLP concept.

The figure shows the frame preamble (P1), used for synchronization at the beginning of every T2 frame, and the L1 signaling, with information about the PLPs in the frame. After the preamble and the signaling, the frame contains the different data PLPs, three in the example above (PLP1, PLP2 and PLP3). PLP1 is signaled as a common frame, indicating receivers that they must demodulate this frame in addition to the PLP containing the service selected by the user. Also, if the PLPs do not use the overall capacity of the frame, dummy cells are appended at the end of the frame. Dummy cells contain no information and they are used to fill a frame whenever PLPs do not use all the cells available.

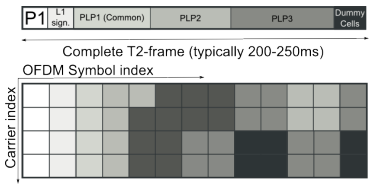

Figure 31. DVB-T2 Physical Layer Pipes diagram.

Back to Figure 30, time slicing is clearly a form of Time Domain Multiplexing (TDM). However, the M-PLP mode provides a very flexible framework enabling combinations of TDM and Frequency Domain Multiplexing (FDM) or OFDMA (Orthogonal Frequency Domain Multiple Access). Thus, it is possible to adapt the characteristics of the virtual channel provided by each PLP to the demands of the service.

III.5 Opportunistic Data Insertion in a DVB tunnel

The previous section highlighted the mechanisms incorporated in state-of-the-art broadcasting systems to provide service specific robustness. On one hand, DVB-H provides MPE-FEC and time slicing, which enables TDM of different services. On the other hand, DVB-T2 implements the M-PLP mode, allowing different combinations of TDM, FDM or OFDMA in DVB-T2 frames. Television services are multiplexed on top of these mechanisms.

As with DVB-T services, each television service may apply either CBR encoding or VBR encoding. If the overall sum of the bitrate of all services is less than the channel capacity, there is guaranteed ODI capacity available in the multiplex (MPEG-TS packets in DVB-H and dummy cells in DVB-T2).

In this thesis, we propose the use of ODI to simplify the multiplexing process of dedicated tunnels (PLPs or MPE-FEC frames) and achieve 100% channel capacity utilization. This is accomplished by sending a background CDS service together with every video streaming service [54]. In this sense, there are different ways to multiplex each video service with its companion CDS service. For instance, it is possible to use opportunistic data insertion at MPEG-TS level, as described in section III.3.

If the service in question incorporates an IP protocol stack, there is a second alternative: to perform the insertion at the IP layer level, inserting IP packets belonging to the background CDS into the same (MPE or PLP) tunnels that carry the streaming service. This implementation is compatible with both DVB-H and DVB-T2 system specifications and is addressed in the following subsections.

III.5.1 IP datagram encapsulation with ODI

In this section we are going to highlight some implementation details of two different ways to insert a background CDS together with a streaming service at the IP layer. In the first scenario, the DVB physical layer provides a guaranteed bitrate capacity to the IP layer. The insertion process consists of a queuing process with priorities, where the streaming service packets conform the high priority queue and the CDS service packets conform the low priority queue, as shown in Figure 32.

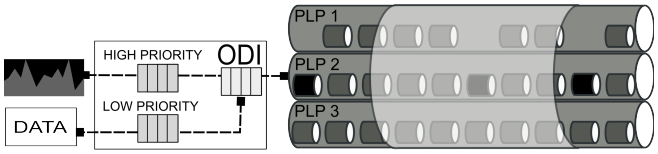

Figure 32. Opportunistic Data insertion in an IP tunnel.

In the figure, the total capacity of the DVB channel is divided in three different PLPs, which are regarded as guaranteed bandwidth IP tunnels. The services in PLP 1 and PLP 3 do not use the guaranteed capacity at all times. In DVB-T2, this results in the presence of dummy cells in the DVB-T2 frame. In PLP 2 we have a similar service, except that it uses ODI to insert packets from a background CDS service. Hence, ODI manages a queue that inserts packets into the PLP at a constant bitrate, equal to the guaranteed bitrate of the tunnel. The packets of the streaming priority service have priority over the packets of the background CDS, so that no packets from the streaming service are lost and the timing requirements are still met. On the other hand, the CDS has no bandwidth or timing requirements and the packets will always be useful payload.

This scenario is very similar to the scenario presented in the previous section (see III.3.2), except that the ODI process is performed on a service basis, before the multiplex, and that the insertion is done at the IP layer level. In any case, the implementation is quite similar. First, it is necessary to detect the available residual bitrate. In case that the streaming services use the RTP protocol, it is possible to monitor the transport stream bitrate from PCR timestamps at the RTP level. [51] defines a mechanism to encapsulate

MPEG-TS into RTP packets where the PCR count is mapped into the timestamp field of the RTP packet header. Additionally, [51] specifies that the number of MPEG-TS packets in every IP packet should be constant (typically, 7 MPEG-TS packets per IP packet). Therefore, the instant bitrate can be determined by measuring the packet inter-arrival time in the video streaming service queue. Later, the insertion process can be carried out locally, with the available bandwidth detection, or remotely, as in ODI insertion in a DVB multiplexer.

For the second scenario, let us assume that the multiplexing scheme applied at the physical layer uses some form of TDM so that the packets belonging to service arrive in bursts. Also, as with DVB-H, the burst periods are larger than the video sampling rate, so that every burst contains a set of frames, equivalent to a certain playback time. This will always be beneficial for battery constraint devices, because it is possible to shut down the receiver at idle times and still play out the frames in a burst while expecting the next one.

Ideally, every burst should contain the data belonging to the playback of a period of time equivalent to the burst cycle, with at least one intra prediction frame per burst. This way, the player application can start playback immediately after receiving a burst. Unfortunately, due the intrinsic properties of video traffic, these ideal bursts do not have a constant size. Instead, their size changes from one burst to the next, giving two possible alternatives for multiplexing:

- Deterministic multiplexing: one alternative is to allocate a fixed amount of capacity in the multiplex to each service, sufficient to allocate the maximum expected burst size of the service. If there is no data available in a given slot, the overall efficiency will be degraded.
- Statistical multiplexing: consists of changing the configuration parameters (MPE-FEC/time slicing in DVB-H, PLP configuration in DVB-T2) dynamically from burst to burst, adjusting these parameters to the burst size statistics of all services, while at the same time keep control over QoE metrics like the video quality, the zapping time or the battery consumption.

In the second case, the efficiency gain brought by statistical multiplexing depends on the statistics of the bursts and there is no guarantee that the capacity of the channel is completely used during the whole time. It is also worth noting that in second-generation networks, the implementation of statistical multiplexing is more complicated than in first generation networks as it is necessary to take into consideration service robustness. On the other hand, background content download services do not have any timing constraints. Therefore, it is possible to provide a background content download service together with every video service so that the resulting burst duration, that is the time needed to transmit the burst size, is kept constant or within some boundaries.

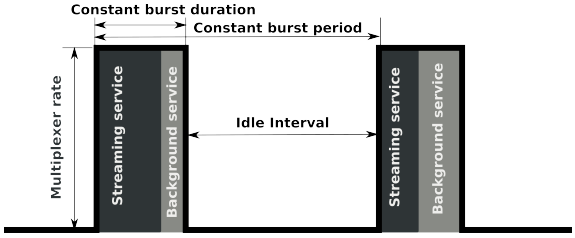

Figure 33. Video streaming service plus background CDS service and TDM multiplexing.

Clearly, this simplifies the multiplexing process, making it easier to use the whole channel capacity, while meeting the QoS requirements of the services the whole time. The proposal is compatible with both deterministic multiplexing and statistical multiplexing. This way, there is no efficiency lost in deterministic multiplexing working with video services, since the reserved capacity is always be utilized either by the primary streaming service or by the background content download service. In statistical multiplexing, the data packets can be used to adjust the statistics of the burst size to optimum values for the specific multiplexing algorithm.

The following subsection shows some results to help illustrating these concepts.

III.5.2 Results of ODI in guaranteed bitrate tunnel

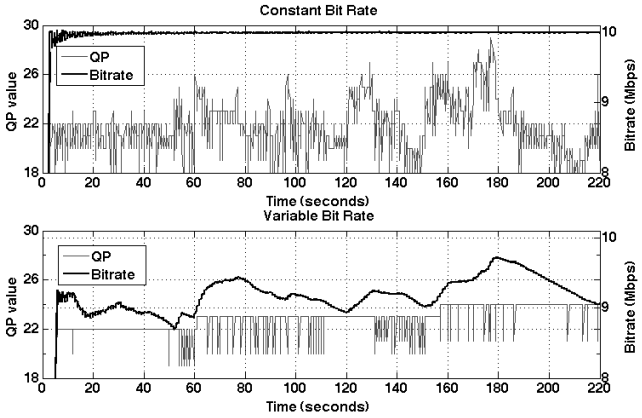

Figure 34. Comparison of QP and bitrate of CBR encoding and VBR encoding.

Figure 34 provides two traces of the bitrate and the quantization parameter (QP) of the same video sequence encoded with Constant Bit Rate and Variable Bit Rate, using the open source H.264 video codec application x264 [55]. The video samples have a resolution of 1280x720 pixels and the sampling rate is 30fps. The bitrate of the encoder is calculated as:

$$b_{vbr}(i) = \frac{\sum_{j=0}^{i} f_{size}(j)}{t(i)} \qquad (22)$$

In Eq. 22, f_{size} is the frame size and t is the time of frame i. The CBR encoder achieves the constant bitrate by enforcing the configured bitrate every fixed amount of encoded video bytes, known as the Video Buffer Verifier (VBV) size. This method ensures that a T-STD decoder with a Transport Buffer (Figure 20) of VBV size does not overflow or underflow at any time if the encoded stream is written at the encoding bitrate.

Figure 34 shows the bitrate and the Quantization Parameter (QP) of every frame for the same video sequence encoded with CBR and VBR. The QP is related to intra frame encoding. A high value of QP indicates that the information of the frame is highly compressed, being somewhat related to subjective quality. The CBR encoder changes the

QP of every frame to enforce the constant bitrate. In the example, the CBR bitrate is 10Mbps and the VBV size is 2Mbits. On the other hand, the VBR encoding policy does not have such strict bitrate constraint. It is configured to provide a long-term average bitrate of 8Mbps, but it is allowed to change the bitrate up to a 25% (6-10Mbps) according to complexity of the scene. Since the bitrate requirement is more relaxed, the encoder uses a more constant QP across frames.

Looking at the QP traces, it is clear that the excess of bitrate used by CBR encoding does not actually provide any improvement in encoded video quality. The QP trace of the VBR is lower than the QP trace of the CBR video for most frames. Moreover, the scene increases its complexity at some time instants (e.g. around frame index 8000 and frame index 18000, causing an increase on the average QP and VBR bitrate.

Now assuming that both encoders use tunnels of 10Mbps, the VBR encoder leaves some capacity for a background CDS service and no penalty on encoded video quality. As depicted in Figure 32, the ODI manages a queue with priorities. The output queue will serve at a rate equal to the capacity of the tunnel, B (10Mbps). Therefore, the capacity available for the background CDS service at every frame index i is given by:

$$b_{CDS}(i) = B - \frac{\sum_{j=0}^{i} f_{psize}(j)}{t(i)} \qquad (23)$$

where $f_{psize}(j)$ is the size of the frame accounting for the packetization overhead.

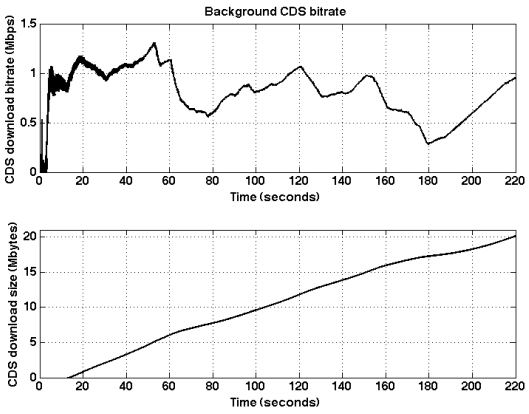

Figure 35. Background CDS download bitrate and achieved download size.

Figure 35 shows the bitrate $b_{CDS}(i)$ available for ODI when the video under study is packetized for RTP transmission and transmitted through a tunnel of 10Mbps capacity. The bitrate is shown together with the amount of data downloaded during the duration of the video. The figure shows how the CDS bitrate decreases when the scene is more complex. The background CDS service achieves a total download size of 6.1 Mbytes after 60 seconds, 11.8 Mbytes after 120 seconds and 20 Mbytes after 220 seconds of video. Clearly, these values are rather high and the service can be used for different applications.

III.5.3 Results of ODI with time slicing encapsulation

The following results regard the case when the video is sent in bursts as in Figure 33. As explained, video streaming services have strict timing requirements. The most efficient mapping of video packets to TDM bursts is to insert the data corresponding to a playout time of length equal to the burst period into each burst. This way, the receiver can start playing the data received in one burst while expecting the next one. Hence, the minimum payload data of every burst consist of a GOP. In the CBR and VBR encoding examples, GOPs are made of 12 frames. Figure 36 shows the statistics of the GOP sizes found in the two encoded sequences. As depicted in the figure, the payload of these ideal bursts does not have a constant size. Regarding the CBR video, the GOP size is almost constant and the

histogram shows a large frequency peak in a bean centered at approximately 500kbytes. Contrarily, the histogram spread of the GOP size of the VBR video is much larger, with a minimum size around 200kbytes and a maximum size of approximately 900kbytes. Moreover, there are no frequency peaks in the VBR GOP size histogram.

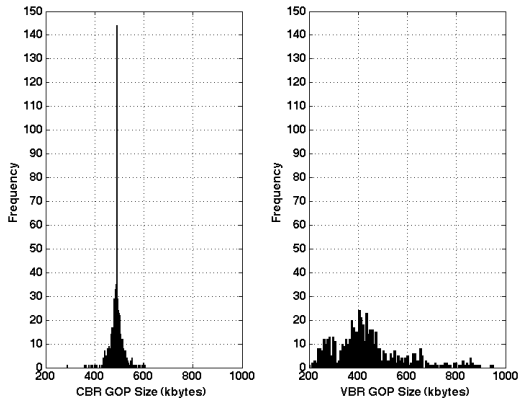

Figure 36. Comparison of the histogram of the size of GOPs with VBR and CBR encoding.

Now, let us compare two different time slicing encapsulation techniques: statistical encapsulation and deterministic encapsulation with ODI insertion. In order to perform such comparison, the multiplexing rate, *mux_rate*, is set to:

$$mux\_rate = \frac{c_{ts} \cdot \max(gop\_size)}{gop\_rate} \qquad (24)$$

In the equation, the *gop_rate* is defined as the frame rate divided by the number of frames in a GOP (30/12 in the example). In order to fulfill the timing requirements of the video, bursts should arrive exactly at the GOP rate. In the example, this results in a burst period of (12/30 = 400ms). On the other hand, the instantaneous bitrate of the multiplexer must be at least equal to the maximum instantaneous bitrate required by the bursts, which is exactly the maximum GOP size divided by the GOP rate. Otherwise, the multiplexer will not be able to send the largest burst completely and at the same time fulfill the timing requirements. Normally, the multiplexing bitrate is significantly higher than this minimum

value, in order to achieve short burst durations and create idle intervals between bursts. As explained, these idle intervals allow to multiplex different bursts of different services in the time domain and to shutdown the receivers to save battery. In equation (21), the ratio between the actual multiplexing rate and the minimum required multiplexing bitrate is noted as c_{ts}.

As shown in Figure 33, the burst duration is the time necessary to transfer a burst at the multiplexer rate. Hence, the burst duration is equal to the burst size divided by the multiplexer rate. Note that Eq. 24 implies that the maximum burst duration is $1/c_{ts}$ of the burst period. At every burst period, the multiplex utilization is defined as the portion of time that the service uses the multiplex capacity. Alternatively, the battery reduction is the portion of time that an ideal receiver could shut down the demodulator. Each time slicing technique deals with the changing burst size in a different manner. In statistical encapsulation, the payload of every burst is adjusted to the GOP size and the burst duration will vary from one GOP to the next. In deterministic encapsulation with ODI, the payload of every burst is adjusted to the maximum GOP size, adding CDS packets to fill the bursts so that, at the end, burst size and duration are kept constant. In order to keep the timing requirements of the video, the burst period is equal to the frame rate divided by the GOP size. In the example, this results in a burst period of 400ms.

The following example illustrates the differences between statistical encapsulation and deterministic encapsulation with ODI. Setting c_{ts}=2 yields the following results:

TABLE III
CHARACTERISTICS OF STATISTICAL TIME SLICING AND DETERMINISTIC WITH ODI TIME SLICING

	Statistical		**Deterministic+ODI**	
	CBR	**VBR**	**CBR**	**VBR**
Mux bitrate (kbps)	24854.4	39011.5	24854.4	39011.5
max burst duration (s)	0.2	0.2	0.2	
min burst duration (s)	0.0941	0.0413		
mean burst duration (s)	0.1610	0.0928		
Battery reduction	59.75%	76.82%	50%	
Mux utilization	40.25%	23.18%	50%	

As indicated in Eq. (24), the multiplexer bitrate is set according to the maximum GOP size, resulting in 24.85 Mbps for the CBR video and 39.01 Mbps for the VBR video. Since c_{ts} equals 2, the maximum burst duration is half the GOP period (400ms/2 = 200ms). In statistical multiplexing, the burst duration is adjusted according to the size of every burst. This way, the capacity of the TDM tunnel is adjusted to the bitrate of the service, and the battery reduction and multiplexer utilization vary from burst to burst. The table provides the average of the session. On the other hand, deterministic encapsulation with ODI adjusts the size of every burst to the maximum GOP size, by inserting packets from the background service whenever necessary. Therefore, the burst duration is always equal to the maximum burst duration (200 ms). Consequently, the tunnel capacity is half the multiplexer capacity (12.43 Mbps in the case of CBR video and 19.51 Mbps in the case of VBR video), resulting in a battery reduction and multiplexer utilization of exactly 50% ($1/c_{ts}$).

Furthermore the bitrate of the CDS service added by ODI insertion is approximately a 25% of the CBR encoder bitrate and a 10.23% of the multiplex bitrate. Also, as indicated in Table III, the battery reduction decreases from 59.75% down to 50%. On the other hand, the CDS is able to push approximately 60Mbytes of data only after 220 seconds.

The table highlights the benefits and drawbacks of deterministic encapsulation with ODI. On one hand, it simplifies the multiplexing process, because it provides constant burst sizes. However, by adding ODI packets, the overall multiplex utilization and the battery consumption increase. This effect is more noticeable with VBR.

Figure 37 shows the performance of the CDS service with CBR encoding. In the figure, the burst payload and the bitrate of the background CDS service is rather constant -around 2.4Mbps- once the CBR encoder has achieved its bitrate. This can be explained by looking at the histogram of CBR burst sizes. By allocating a burst size equal to the max burst size, the encapsulator is always using a capacity of 12.427Mbps, inserting CDS packets into each burst, except for the burst transporting the larger GOP. Therefore, the remaining bitrate for the CDS service tends to a constant bitrate of 2.427Mbps (approximately the tunnel capacity, 12.427Mbps minus the CBR bitrate, 10Mbps).

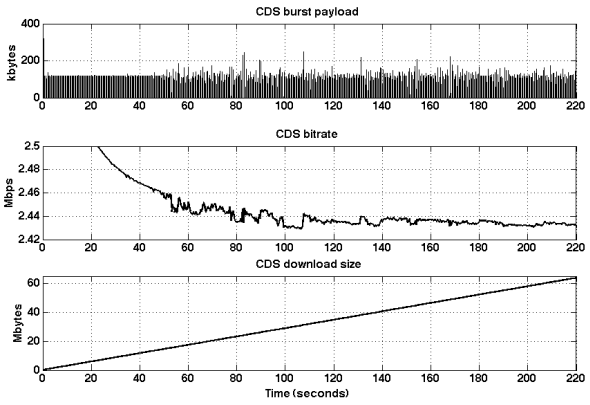

Figure 37. Background CDS burst capacity, bitrate and total download size with CBR encoding and time slicing.

Figure 38 shows the performance of deterministic encapsulation with ODI with the VBR encoded sequence. Note that the scale is different for the sake of clarity. In this case, the burst payload and the bitrate of the background CDS are more variable. The average bitrate of the tunnel is 19.5Mbps and the mean VBR bitrate in the encoded sequence is 8.86Mbps. This yields to an average bitrate for the CDS service of 10.32Mbps. Therefore, the bitrate

of the background CDS service with deterministic encapsulation with ODI is 116.5% of the bitrate of the streaming service. This means that the multiplexer utilization goes from the 23.18% of statistical multiplexing to the 50% of deterministic multiplexing with ODI. Furthermore, the battery reduction achieved by statistical multiplexing is 76.82%, against the 50% of statistical multiplexing with ODI. On the other hand, the CDS total download size after 220 seconds is 274Mbytes, which is quite a significant amount of data.

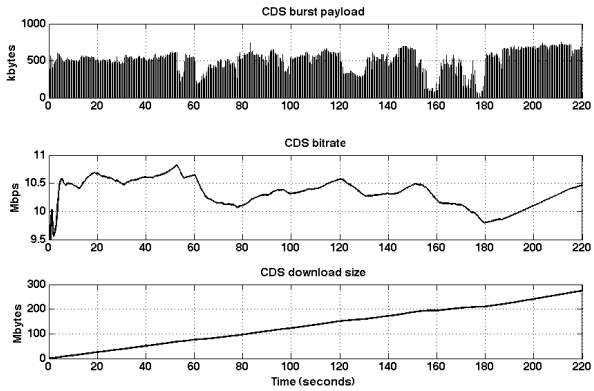

Figure 38. Background CDS burst capacity, bitrate and total download size with VBR encoding and time slicing.

In both cases, the bitrate of the background CDS service appears to be rather large. Depending on the application, it could be interesting to trade off bitrate with some other parameter. One way to achieve lower bitrates is to use larger bursts, made of several GOPs. However, larger bursts mean longer burst cycles and in extent longer initial buffer delays and zapping times. Hence, it is possible to trade off background CDS bitrate and buffering delay and zapping time.

Figure 39 shows the background bitrate achieved with different burst sizes. As the burst size increases, the difference between the maximum burst size and the burst size become smaller and consequently, the bitrate reserved for the tunnel tends to the CBR encoding bitrate. In turn, the capacity available for the background CDS decreases. The figure, highlights the big decrease in the background CDS bitrate achieved when the burst payload

consist of two burst. However, for burst sizes of 3 or more GOPs, the improvement is not so significant.

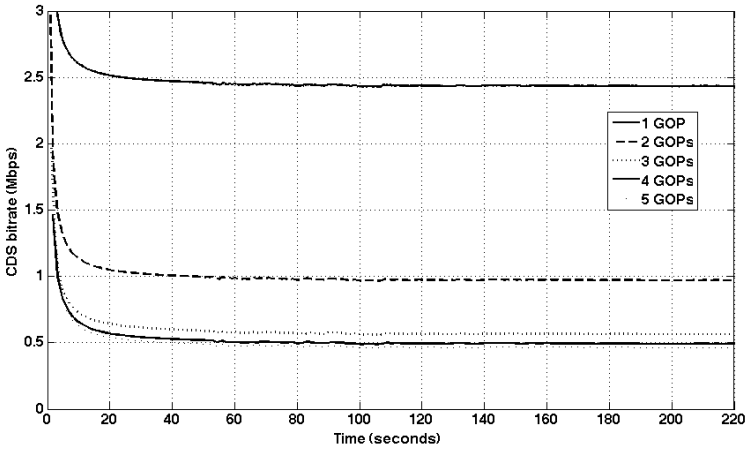

Figure 39. Background CDS bitrate with deterministic encapsulation with ODI for different burst sizes and CBR encoding.

The multiplex utilization and battery consumption also decrease with the Background CDS bitrate. To illustrate this, Figure 40 represents the CDS bitrate and the battery consumption degradation against the burst size. The battery consumption degradation is defined as the difference in battery consumption between statistical multiplexing and deterministic multiplexing with ODI insertion. In Figure 40, the mux rate is kept constant. On the other hand, each burst transports an increasing number of GOPs and the burst period changes to adapt to the playback duration of every burst. As indicated, with time slicing, the zapping time is half the burst period. Hence, the zapping times in Figure 40 correspond to burst periods of 0.4 (1 GOP) to 16 seconds (40 GOPs). At this point, it is worth noting that the right-side values of the figure may not be useful for mobile television services. The graph shows that, as the burst period and the zapping time increase, the battery reduction decreases at the same path as the CDS bitrate.

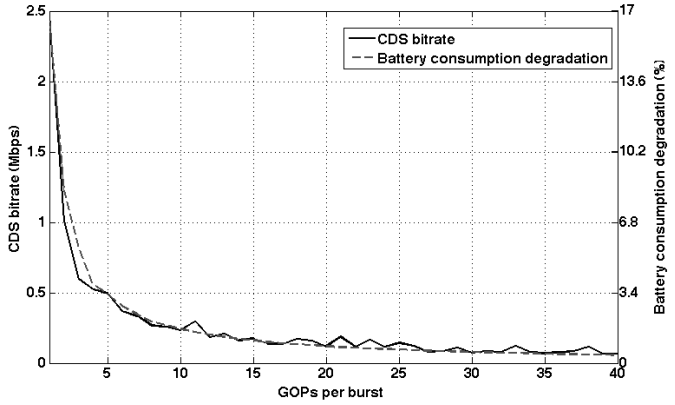

Figure 40. Background CDS bitrate and battery consumption degradation against zapping time with ODI for different burst sizes and CBR encoding.

With VBR encoding, the effect is similar. Figure 41 shows the CDS bitrate achieved with different burst sizes, from 1 GOP to 33 GOPs, with the VBR video sequence. As with CBR, the differences between the burst sizes decrease with larger bursts, in general, but not in all cases, because with VBR it depends on the statistics of the burst sizes and a larger burst does not always mean a lower CDS bitrate. This can be noted in the figure, since a burst size of 25 GOPS has a larger bitrate than a burst size of 17 GOPs. However, the tendency is the same as with CBR and larger burst generally mean lower CDS bitrate. Moreover, larger burst periods mean fewer bursts and thus, fewer opportunities for data insertion. Another clear effect is that using larger bursts has a smoothing effect in the CDS bitrate, which becomes more and more constant.

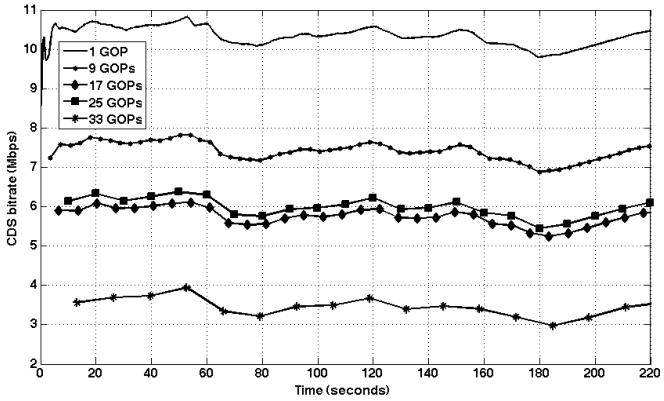

Figure 41. Background CDS bitrate with deterministic encapsulation with ODI for different burst sizes and VBR encoding.

Figure 42 shows the CDS bitrate and the battery consumption degradation for different burst sizes. The multiplexer bitrate is kept constant and burst periods and sizes are changed to encapsulate a different number of GOPs (from one to 16) in each burst.

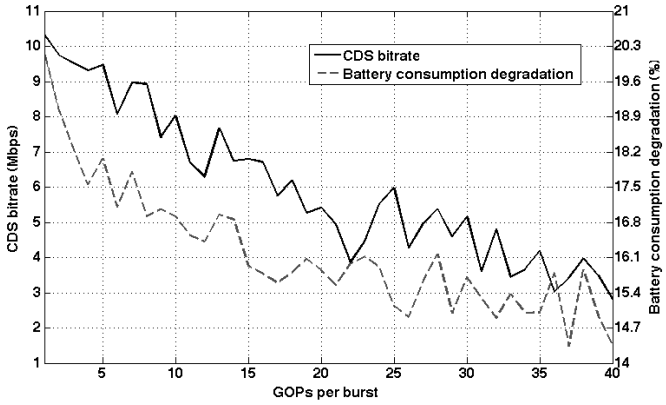

Figure 42. Background CDS bitrate and battery consumption degradation against zapping time with ODI for different burst sizes and VBR encoding.

The figure shows a decreasing tendency of the bitrate and the battery consumption degradation. Unlike with the CBR encoding, the battery utilization and the bitrate do not

follow the exact same path. This is because the battery utilization degradation is a measure of the difference between the battery reduction with statistic encapsulation – only the streaming service - and deterministic encapsulation with ODI – the streaming service plus the CDS service. The battery reduction is related to the idle duration while the bitrate is related to the burst duration. With CBR encoding, the bitrate used is constant and so it is the relationship between the burst duration and the idle duration, regardless of the number of GOPs included in every burst. However, this does not hold with VBR and it is not possible to state that the bitrate and the battery reduction vary in the same way.

In summary, in time slicing, the ideal burst payload consists of a GOP. Both CBR and VBR encoding generate GOPs of different sizes. The differences in CBR GOPs are rather small, while the differences in VBR GOPs are considerably larger. Deterministic encapsulation with ODI achieves a constant burst size by adding packets from a background CDS service to every burst. This insertion results in a higher multiplex utilization and a higher battery consumption. With the small differences found in the burst sizes of the CBR encoded sample, the maximum bitrate of the background CDS is around 25% of the bitrate of the video streaming service. On the other hand, the large differences in the GOP sizes in the VBR video sequence result in a background CDS bitrate over 116% of the average video streaming bitrate. In both services, the total amount of download data after 220 seconds is quite large, over 60Mbytes in the case of CBR encoding and over 200Mbytes with VBR encoding. At the same time, ODI insertion simplifies the multiplexing process. On the other hand, the resulting CDS capacity may be too large depending on the application. In these cases, it is possible to increase trade-off CDS bitrate by increasing the burst size and the burst period, thus trading off background CDS bitrate with initial decoding delay and in turn zapping time.

III.6 Conclusions

Terrestrial broadcast networks use a constant bitrate transport stream to multiplex together several television services. Due to the timing requirements of digital video and the constant bitrate constraint of broadcast networks, multiplexers need to insert filling packets into the broadcast transport stream, in order to achieve the transport stream rate at all times.

Opportunistic Data Insertion (ODI) uses this filling capacity to broadcast additional services.

We have evaluated the filling bitrate in different multiplexers belonging to one national public network, two private national networks and one regional public network. The results show that commercial DVB network operators use filling bitrates in the order of several hundreds of kbps. In some cases the filling capacity exceeds 1Mbps. The results show that the use of ODI for the delivery of download services and local storage is an interesting study case. For instance, the mean bitrate available for ODI -averaged in all the multiplexes under study- is 976.7kbps. At this bitrate, a content download service will be able to push around 9.8Gbytes of data a day. We have measured the filling capacity of two multiplexers during 36 hours in order to obtain models for the available bitrate. These models are used in the following chapter in order to perform a thorough evaluation of background CDS networks over DVB networks.

On the other hand, second-generation DVB networks use dedicated physical layer tunnels for every service. The example, consisting of encoded scenes of a commercial TV program, showed that there is a lot of capacity remaining in a fixed capacity tunnel transporting a VBR encoded sequence. In this context, statistical multiplexing implies adapting the capacity of every tunnel to the exact bitrate of the services, dynamically, and at the same time, guaranteeing that the tunnel provides an optimum QoS for the service. Clearly, this is quite challenging. Alternatively, we have presented a multiplexing technique referred to as deterministic multiplexing with ODI. With this technique, each streaming service is associated to a background service that uses the excess of capacity in the capacity reserved for the primary streaming service. We have presented an example of the performance of this technique with a commercial video sequence of approximately 20 minutes long. In the example, the background content download service managed to download over 250 Mbytes of data during the transfer of the video. Therefore, this kind of background services can have several applications for both fixed and mobile terminals.

The results also show the performance of the technique in combination with time slicing, a technology designed to save battery in mobile devices. With time slicing, several frames are grouped into bursts. Then, each burst is sent at a higher rate than necessary, in order to conform idle intervals between bursts. With time slicing, the ideal burst payload consists of a GOP, since this achieves minimal initial buffering delays and zapping times. In this sense, both CBR and VBR encoding generate GOPs of different sizes. The differences in CBR GOPs are rather small, while the differences in VBR GOPs are considerably larger. Deterministic encapsulation with ODI achieves a constant burst size by adding packets from a background service to every burst. This insertion provides an additional service and simplifies the multiplexing process, at the expense of a higher multiplex utilization and a higher battery consumption.

With the small differences found in the burst sizes of the CBR encoded sample, the maximum background CDS bitrate is around 25% of the bitrate of the CBR video. On the other hand, the large differences in the GOP sizes in the VBR video sequence produce bitrates over 116% of the VBR average bitrate. In both services, the total amount of download data after 220 seconds is quite large, over 60Mbytes in the case of CBR encoding and over 200Mbytes with VBR encoding. The resulting background CDS capacity may be too large depending on the application. In these cases, it is possible to trade-off bitrate capacity and zapping time by increasing the burst size (in number of GOPs).

Chapter IV

Background Content Download Services in DVB Networks

This section presents an evaluation of background Content Download Services (CDSs) in the scenarios under study, according to the measurements and models presented in previous sections. Each scenario is defined by different characteristics of the service architecture:

- **Content.** As indicated in Chapter II, this thesis regards two different kinds of content catalogues: the IMDB (television content) and YouTube (short video clips, mainly User Generated Content). Each model is used to characterize the Content Repository in the context of different background CDS applications: The IMDB catalogue represents long duration programs, like movies and episodes of series, while the Youtube catalogue represents short duration programs, like music videos or commercials.
- **Network.** Chapter Chapter III described the available bitrate for ODI in first-generation DVB networks and second-generation DVB networks. Each one of the two different networks provides a different model for the ODI insertion in the architecture.
- **Terminal.** There are two different kinds of terminals regarded in this study, television sets and mobile terminals. The main differences in the context of this study are reception and screen resolution.

DVB radio frequency network planning targets error free transmission for fixed (rooftop) reception and 5% MPE-FEC frame error rate for good mobile reception. Therefore, it is assumed that the physical layer is able to correct the errors introduced in the communication channel when the target terminal is a television set. Oppositely, for mobile terminals, we regard losses of 5% (good reception) and 50% (bad reception) in a TU6 [36] channel as explained in the Channel model section in Chapter II.

Regarding screen resolution, both kinds of terminals have a wide range of formats. The Standard Definition (SD) DVB format inherits the pixel resolution of the former Phase Alternating Line (PAL) Analogue television format (720x576). Higher resolution formats include, High Definition TV (HDTV, 1280x720), full HDTV (1920X1080) or 4K Ultra High Definition TV (4K UHDTV, 3840 × 2160). On the other hand, mobile phones use a wide variety of resolution formats, from the Quarter Video Graphics Array (QVGA, 320x240) or VGA (640x480) to the aforementioned HDTV format. In the context of this thesis, the resolution will determine the characteristic encoding bitrate of the videos[1]. The encoding rates used in the Youtube site [18] are used as a reference to establish the relationship between resolution and encoding bitrate.

[1] Youtube Encoding approximate bitrates: HDTV 3.3Mbps (2-2.9Mbps video, 2x192kbps audio); full HDTV 5Mbps (3-4.3Mbps video, 2x192kbps audio); QVGA 0.4Mbps (0.25 video + 2x64kbps audio); 360p 0.7Mbps (0.5Mbps video + 2x96kbps audio).
(source: http://en.wikipedia.org/wiki/YouTube)

Table IV summarizes the parameters of each of the scenarios under study:

TABLE IV
PARAMETERS OF THE SIMULATION SCENARIOS

	Scenario 1: Background CDS for television sets	**Scenario 2: Background CDS for mobile terminals**
Content source	IMDB, Youtube	Youtube
Network	DVB Multiplex	DVB tunnel
Terminal	Television set	Mobile terminal
Encoding bitrates	2,3.3,5,7 Mbps	0.4,0.7,2,3.3
Channel model	Ideal channel	TU6 (50 Km/h Doppler); 5% or 50% losses

With this, the following section shows the performance of the background CDS server in the two scenarios under study.

IV.1 Background Content Download Services for television sets

Section III.3.2 showed the bitrate available for ODI in different commercial DVB-T multiplexers. The results presented the long-term statistics of the number of MPEG-TS packets available for ODI. In this section we are going to present the properties of the filling bitrate in longer time intervals.

IV.1.1 Long term bitrate of ODI in a DVB multiplex

Assume that a file j of size s_j bytes is transferred over a background channel that uses Opportunistic Data Insertion in a DVB multiplex. In section III.3.2, the filling capacity is measured at regular measurement intervals of t_{meas} seconds. From the measurements, it is possible to derive a statistical model for the number of filling MPEG-TS packets -p_{ODI}- found at every measurement interval (204ms in the models in section III.3.2). With this model for the available bitrate, the expected bitrate during the download of a file j over the background channel, referred to as E[b_j] is approximated by the following expression:

$$\mathrm{E}[b_j](bps) \approx \frac{s_j \cdot 8}{n_m \cdot t_{meas}}, \quad n_m / p_{mpegts} \cdot \sum_{k=1}^{n_m} p_{ODI}(k) \geq s_j \tag{25}$$

In the equation, the ODI inserter needs n_m intervals of t_{meas} seconds (204ms) to transfer the entire file. In every interval k, the ODI manages to insert a random number of MPEG-TS packets p_{ODI} (p_{mpegts} is the payload size of MPEG-TS, equal to 184 bytes). Hence, the transfer is completed somewhere between the interval n_m and the interval n_m+1, but since we do not have statistics about the available rate on time scales smaller than t_{meas}, we cannot estimate the bitrate with better accuracy. Let us express the equation as:

$$\mathrm{E}[b_j](bps) \approx \frac{p_{mpegts} \cdot 8}{t_{meas}} \cdot \left(\frac{1}{n_m} \cdot \sum_{k=1}^{n_m} p_{ODI}(k) \right) \tag{26}$$

The summation indicates that the expected bitrate is an average of the underlying available ODI rate, characterized by the random process p_{ODI}. According to the central limit theorem, the distribution of the average tends to a normal distribution, regardless of the statistical distribution from which the average is computed:

$$\sqrt{n_m} \left(\frac{\sum_{k=1}^{n_m} p_{ODI}(k)}{n_m} - \mu \right) \xrightarrow{D} N(0, \sigma^2) \tag{27}$$

where $\xrightarrow{D} N()$ means *convergence to a normal distribution*, while μ and σ are respectively the mean and the standard deviation of p_{ODI}. Thanks to the properties of the normal distribution, the equation can be rewritten as:

$$b_m = \frac{\sum_{k=1}^{n_m} p_{ODI}(k)}{n_m} \xrightarrow{D} \mathrm{N}\left(\mu, \left(\frac{1}{\sqrt{n_m}} \sigma \right)^2 \right) \tag{28}$$

This property allows us to model the average bitrate experienced during the download of a file of a given size. If n_m is approximated as $n_m = ceil(s_i / \mu \cdot p_{mpegts})$, then, the expected bitrate E[b] can be approximated by:

$$\mathrm{E}[b_j](bps) \approx \frac{p_{mpegts} \cdot 8}{t_{meas}} \cdot (b_m), \quad b_m \xrightarrow{D} \mathrm{N}\left(\mu, \left(\frac{1}{\sqrt{n_m}}\sigma\right)^2\right) \tag{29}$$

This approximation can be very convenient in the simulation of CDS services, because it is not necessary to evaluate the available bitrate every few MPEG-TS packets. Instead, the average bitrate over the download is estimated only once per file, depending on the file size s_i (which will provide the value of n_m).

In order to evaluate the accuracy of this approximation, we have conducted a series of simulations using the statistical models derived from the measurement traces of NULL packets in multiplexers C28 and C57. The file size is generated using the file size models for IMDB content and Youtube content (II.1.1) and the different encoding bitrates (2, 3.3, 5, 7Mbps) characteristic of SDTV, HDTV, full HDTV and UHDTV television formats. For each encoding rate, the file size model generates 5000 different file sizes from each distribution.

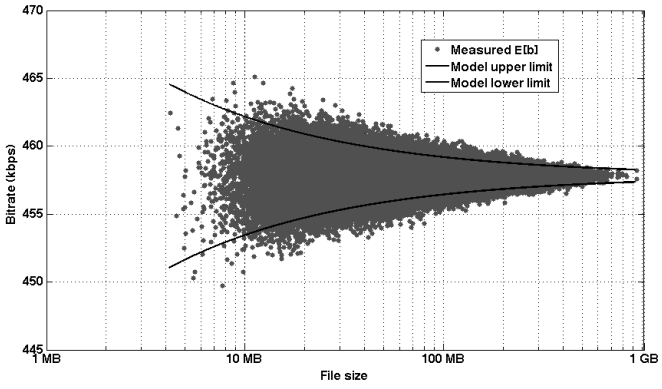

Figure 43. Average bitrate during the transmission of Youtube files with ODI over C28.

Regarding channel 28, Figure 43 presents the results of the simulations conducted with the model of the channel and Youtube files, together with the upper and lower limits of the 95% confidence intervals of the normal distribution approximation provided in Eq. (29). Note that for the normal distribution of b_m, the 95% confidence intervals are provided by:

$$\mathrm{CI}(95\%) = \left[\mu - 2\sigma / \sqrt{n_m}, \mu + 2\sigma / \sqrt{n_m}\right] \tag{30}$$

The figure shows the average bitrate E[b_j] during the transfer of file sizes. The confidence intervals of the model fit the expected bitrate estimated in the different simulations, proving that the model efficiently provides an accurate average bitrate depending on the file size. The figure shows how the standard deviation of the simulation results decreases with the file size, as expected considering the central limit theorem. Hence, when evaluating the expected average bitrate during the transmission of a file, Eq. 29 produces equivalent results as launching simulations with the model derived from the measurements. Moreover, Figure 44 shows the same study, but using the IMDB file size model instead of the Youtube model.

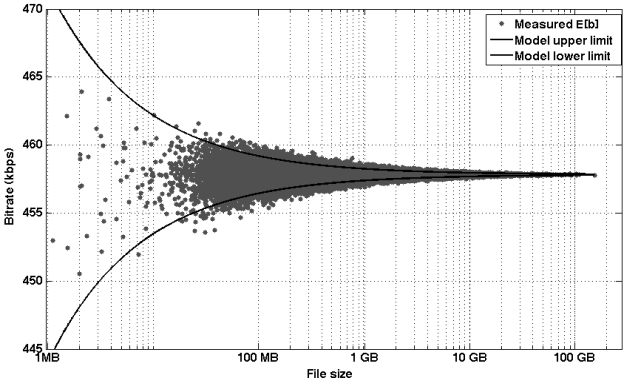

Figure 44. Average bitrate during the transmission of IMDB files with ODI over C28.

In this case, it can be noted that the file sizes produced by the IMDB model vary in a wider range than the file sizes produced by the Youtube model. Still, the bitrate model appears to

be rather accurate and we can conclude that the model in Eq. 29 provides a good estimation of the expected bitrate during the transfer of a file with ODI insertion in C28.

Similarly, Figure 45 shows the bitrate measurements and the bitrate simulated with the model for the ODI bitrate obtained for the multiplex in C57. In this case, the files have been simulated using the Youtube file size model of section (II.1.1) and the same encoding rates as the previous study (2Mbps - SDTV, 3.3Mbps - HDTV, 5Mbps - full HDTV and 7Mbps - UHDTV).

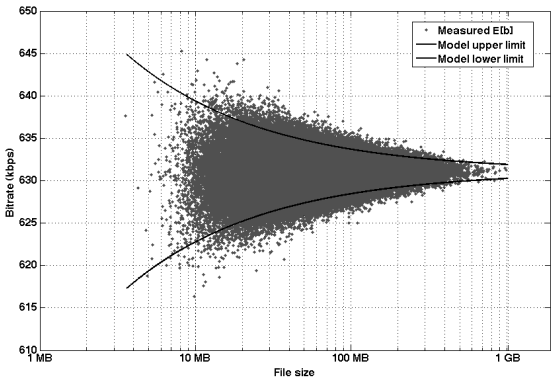

Figure 45. Average bitrate during the transmission of Youtube files with ODI over C57.

It can be noted that the expected bitrate is higher in C57 than in channel C28. Again, the models and the simulations provide equivalent results and the standard deviation of the bitrate decreases for larger file sizes. Additionally, Figure 46 shows the same result but using the IMDB file size model over C57. The conclusions are the same as in previous cases.

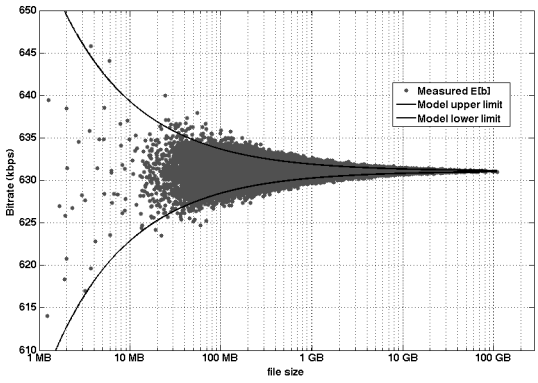

Figure 46. Average bitrate during the transmission of IMDB files with ODI over C57.

IV.1.2 Carousel times of Background CDSs in DVB multiplexes

The carousel time, t_C, is related to the average of the download time of each of the files in the file carousel. Thus, the central limit theorem can be used again to obtain the relationship between the mean and the variance of the download time and the mean and the variance of the carousel time. This way, t_D^j is the download time of file with index j and size s_j in a carousel of N files, over a channel with an average bitrate E[b_j] during the transfer of the file. The carousel time converges to a normal distribution as:

$$t_C = \sum_{j=1}^{N} t_D^j \xrightarrow{D} \mathrm{N}\left(N\mu_D, \left(\sqrt{N}\sigma_D\right)^2\right), \quad t_D^j = \frac{s_j}{b_j} \tag{31}$$

where μ_D and σ_D are the mean and standard deviation of the download time. Figure 47 shows the mean carousel time in C28 against the number of files in the carousel for YouTube files encoded at the different encoding rates under study. The methodology is the same as in the previous studies, except that now, each carousel is evaluated 500 times. Note that the Y axis is in logarithmic scale and that the different curves actually show the linear dependence of the carousel time with the number of files in the carousel.

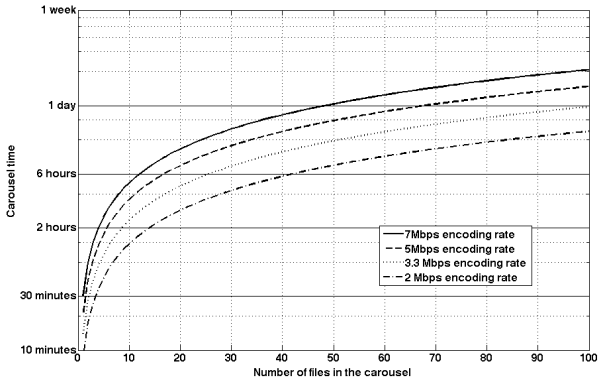

Figure 47. Mean Carousel time for different carousel sizes for Youtube content over C28.

Figure 47 depicts the impact of both the number of files and the encoding rate in the carousel time. Clearly, the resulting carousel times show that the available bitrate cannot be used to provide a live pull content download service, because the times are too large to provide a satisfactory user experience for this kind of services. However, the values are reasonable for a background push content download services.

For instance, a background CDS client could download 100 Youtube videos in HDTV format after 1.5 days, provided that the client had enough storage capacity and there were no losses in the channel. Although the amount of time may seem too large, the download process is transparent to the user, who only experiences that new additional content from the network provider is available almost on a daily basis. Moreover, since the average duration of the videos is approximately 3 minutes, the client will have almost 5 hours of additional content.

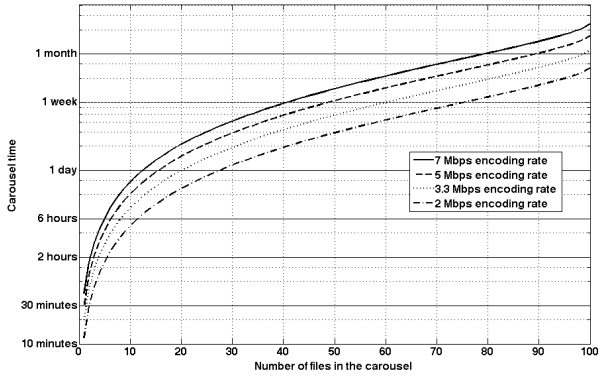

Figure 48. Mean carousel time for different carousel sizes for IMDB content over C28.

Figure 48 shows the mean carousel time achieved for IMDB content using the available bitrate in C28. This kind of content has significantly larger mean duration (around 60 minutes) and larger standard variation. It can be noted that the carousel times are significantly larger than those provided for YouTube content (note the different Y scale in both figures). However, the carousel times are reasonable for a moderate number of files in the carousel. Hence, the background service is also useful for IMDB content, except that the number of files in the carousel may be lower than the number of files for Youtube content. For instance, the mean carousel time for 20 HDTV content items is 1 day and 5 hours. After this time, a background CDS client may have downloaded all the videos in the carousel.

On the other hand, Figure 49 shows the average carousel time of a background CDS service in C57 with Youtube content. It can be noted that, since the available bitrate is higher, the carousel times are lower in this channel. Relating to the example above, the carousel time for 100 HDTV Youtube videos is 1 day, instead of approximately 1,5 days in C28. In this sense, the improvement in carousel time is proportional to the increase in the available bitrate in the channel.

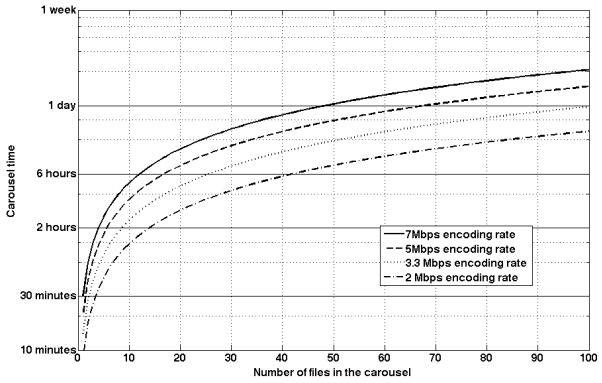

Figure 49. Mean carousel time for different carousel sizes for Youtube content over C57.

Similarly, Figure 50 shows the carousel times of the background CDS over C57 with IMDB content. Again, the carousel times are lower than the download times obtained for C28, especially for short carousels. For instance, in this case, the carousel time for 5 HDTV videos is only 2 hours, instead of 1 day in C28. Furthermore, a carousel of 20 file sizes provides a mean carousel time of 17 hours instead of 1 day.

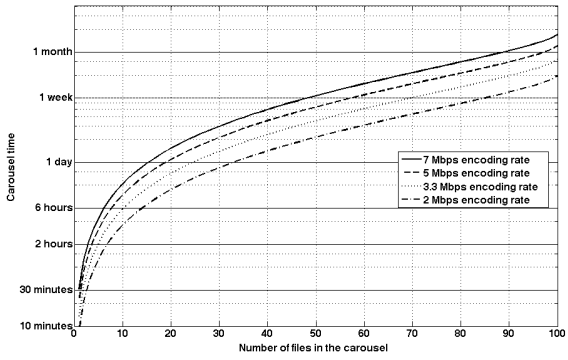

Figure 50. Mean carousel time for different carousel sizes for IMDB content over C57.

IV.1.3 Access times of Background CDSs in DVB multiplexes

Finally, we are going to analyze the *access time* (t_A) of the different study cases. The access time is defined as the time that the client application needs to wait from the instant of time that it decides that a file should be stored in local storage until that particular file is completely downloaded to local storage. With this in mind the access time t_A depends on the *waiting time,* t_W and the *download time*, t_D. 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.

This concept is represented in Figure 51. In the figure, two different client applications want to download the second file in the carousel, F2. The first client accesses the carousel during the transfer of file F2. In our model, client applications need to receive the beginning of the transfer of a file in order to start downloading it. Hence, client 1 needs to wait until the next cycle until it can start downloading file 2. This is because it is assumed that the client of the content download protocol needs to receive metadata information about the file before it can start the download. For instance, with the FLUTE protocol, clients need to receive the FDT section describing a file before they can start downloading it. Hereby, it is assumed that this metadata information is sent right before the transfer of the file. Likewise, client 2 accesses the carousel during the transfer of file 3 and therefore, it also needs to wait until the next cycle to start downloading the file. It is worth mentioning that packet losses are not considered at this point.

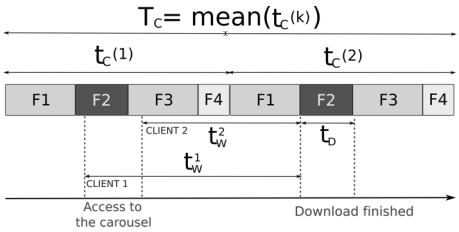

Figure 51. Access time in file carousels.

Note that the use of ODI implies that the carousel time –i.e. the time needed to broadcast all files in the carousel – is not constant for a given set of files. Above, the average carousel time is noted as T_C. Assuming that the client application can join the carousel at any time with the same probability within the carousel cycle, the average of the waiting time is $T_C/2$ for any file in the carousel. Moreover, the average of the download time of file j is defined as T_D^j, yielding the following expression for the average access time to a file j:

$$T_A^j = \mathrm{E}[t_A^j] = \mathrm{E}[t_W^j] + \mathrm{E}[t_D^j] = T_C / 2 + T_D^j \tag{32}$$

At this point, it is assumed that all the different files have the same access probability. With this assumption, the mean access time across the N files in the carousel is:

$$T_A = \mathrm{E}[T_A^j] = T_C / 2 + \mathrm{E}[T_D^j] = T_C / 2 + \sum_{i=1}^{N} p_i \cdot T_D^i, \;\; p_i = 1/N \;\; \forall i \tag{33}$$

Figure 52 presents the mean access time found for HDTV content (3.3Mbps encoding rate) in the different study cases regarded in this section. Every point of the graph is obtained evaluating the download time of 500 instances of file carousels obtained with the respective file size model.

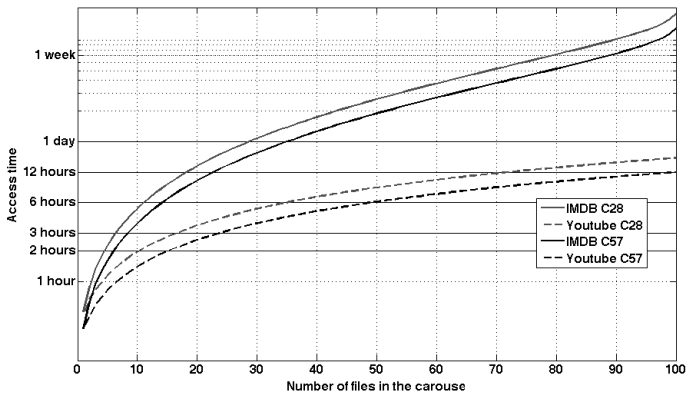

Figure 52. Mean access times for IMDB and Youtube content encoded in HDTV over the background CDS in C28 and C57.

Obviously, the different mean access times resemble the shape of the curves found for the carousel times, due to the linear relationship between them, expressed in Eq. 33. Figure 52 highlights the main conclusions of this section. First, the times found are too high to provide an on-demand CDS with acceptable QoE, especially when the number of files in the carousel is high. However, they seem reasonable for background services where users do not experience directly the delay. In any case, the next chapter presents object multiplexing, a technique aimed at reducing the access times to files by accounting for their relative popularity. The differences in the access time for the two different channels under study are proportional to the difference in the long-term bitrate of both channels. Since the mean duration of IMDB content is longer than the mean duration of YouTube content, the average access time to IMDB content increases at a much faster rate than the average access time to YouTube content.

IV.2 Background Content Download Services for mobile terminals

In this section, background services for mobile terminals, where losses are taken into consideration, are evaluated. Therefore, the effect of packet losses is introduced in the calculation of the download time and the access time.

IV.2.1 Access times of background CDSs over DVB tunnels

The section II.2 presented the models used to characterize the performance of broadcast CDS in the presence of losses. In carousel transmissions, receivers need several carousel cycles to download the files. Therefore, the access time is mainly dependent on the carousel times and the number of cycles needed to download a file. In order to illustrate this, the following figure represents the time needed to access a file in the presence of errors:

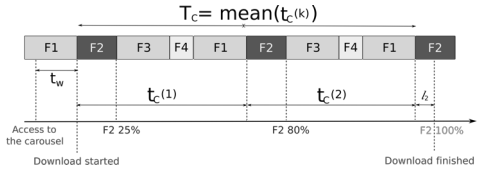

Figure 53. Access time in channels with errors.

In the last section, we explained how, if there are no losses, the access time to a file in the carousel is the summation of the waiting time and the download time. The waiting time is a statistical variable dependent of the instant of time when the client enters the carousel and the carousel time. Assuming that the access time is uniformly distributed in time, the expected value of the waiting time is equal to $T_C/2$, where T_C is the expected time of the carousel time. Similarly, the download time of file *j* is a statistical variable that depends on the file size and the bitrate of the background CDS service.

Oppositely, if there are losses in the reception of the carousel, the download time becomes 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) plus a fraction of the transmission (l_j) to download the file. For instance, in Figure 53, in order to download file F2, the client needs 2 entire cycles ($c_2=2$) plus a fraction of the last cycle (l_2).

The average number of entire cycles needed to download file *j* is defined as $\mathrm{E}\left[c_j\right]=\bar{c}_j$ and, as explained in section II.2, it depends on the size of the file and the losses on the channel. Note that at this point, AL-FEC is not considered. If the receiver starts the download at cycle $t_C^j(k)$, after a time equal to $\sum_{l=k}^{k+\bar{c}_j} t_C^j(l)$, the receiver begins the downloading of the last portion of the file. In order to fetch this last portion, the receiver waits 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 l_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:

$$\mathrm{E}\left[t_D^j(k)\right]=\sum_{i=k}^{k+\bar{c}_j} t_C^j(i)+\mathrm{E}\left[\frac{s_j \cdot l_j}{b}\right] \qquad (34)$$

Now, as in the previous case, since the bitrate is not constant, the carousel time at every carousel cycle is not necessarily the same. However, if the mean carousel time T_C is the

average of the carousel time measured in a number of cycles K, the average of the download time in the same time interval is:

$$\mathrm{E}\left[t_D^j\right] = \frac{1}{K}\sum_{k=1}^{K}\left(\sum_{i=k}^{k+\bar{c}_j} t_C^j(i) + \mathrm{E}\left[\frac{s_j \cdot l_j}{b}\right]\right) = \bar{c}_j \cdot \frac{1}{K}\sum_{k=1}^{K} t_C(i) + \mathrm{E}\left[\frac{s_j \cdot l_j}{b}\right] = \bar{c}_j \cdot T_C + \mathrm{E}\left[\frac{s_j \cdot l_j}{b}\right] \quad (35)$$

Note that, since both summations add consecutive carousel cycles, the terms of the summations can be rearranged to obtain the relationship between the average download time of a file and the carousel time. With this expression, the average access time to file j is:

$$\mathrm{E}\left[t_A^j\right] = \mathrm{E}\left[t_W\right] + \mathrm{E}\left[t_D^j\right] = (1/2 + \bar{c}_j) \cdot T_C + \mathrm{E}\left[\frac{s_j \cdot l_j}{b}\right] \quad (36)$$

IV.2.2 Evaluation results

As in the previous scenario, we are going to use the file size model for Youtube content. This time, in order to model the available bitrate in a DVB tunnel, we are going to use the trace of an encoded VBR video, corresponding to footages from a football match[2]. Figure 54 shows the encoding bitrate of the video, together with some frames representative of the contents of the video.

The video corresponds to a 3DTV stereoscopic video in Side-by-Side format, encoded according to the DVB specifications for 3D content [56]. The video is encoded with VBR, at an average bitrate of 8Mbps and a bitrate tolerance of 25%. The beginning of the video presents a time-lapse sequence of the audience taking their sits in the stadium. This sequence is quite complex and the encoder has problems to keep the bitrate below 10Mbps. Later, the video shows the line-ups of both teams. With the players in the foreground and the audience in the background, these frames also have a lot of information. Later, in most

[2] The footage was produced by the company Mediapro. The authors have the right to use the footage for research purposes in the framework of the Project ImmersiveTV (TSI-020302-2010-61).

of the game, frames mostly show the field, allowing the encoder to reduce the bitrate to approximately 7Mbps. Eventually, some close-ups of the players or interesting actions cause some peaks in the bitrate.

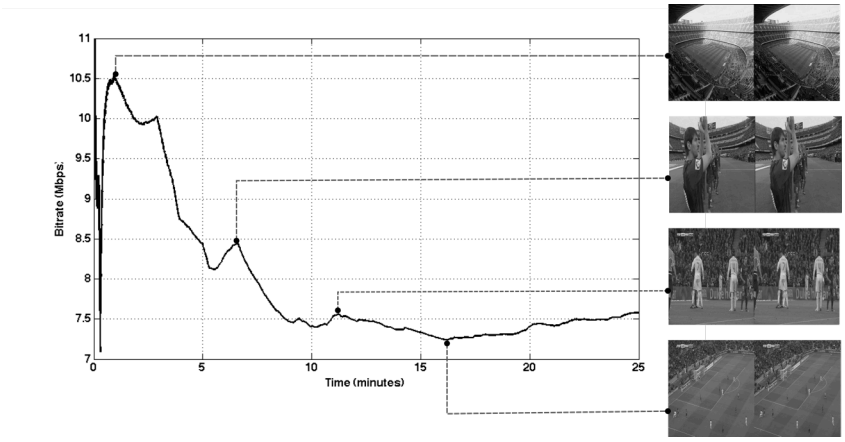

Figure 54. Bitrate trace of the video used for the measurements, together with few representative frames.

In order to generate the following results, a background channel is conformed out the excess of capacity in a 10Mbps tunnel in which the previous video is encapsulated. The background channel capacity is used to broadcast a CDS with videos targeting mobile terminals. Note that the target receiver is not the same for the primary streaming service and the background CDS –the streaming services targets stereoscopic 3DTV sets and the background Content Download Service targets mobile terminals. This kind of heterogeneous scenario can be found in different use cases, as addressed in DVB Home Networks [57] (DVB-HN). In DVB-HN, the services are retransmitted in the local network by a DVB gateway. The purpose of this gateway is to provide access to the services to devices in the local IP network that are not physically connected to the DVB network. In the following example, it is assumed that mobile terminals are connected to the gateway through a wireless connection.

For every simulation, the mean carousel time is evaluated generating 500 carousel instances. The duration of the videos is obtained from the model for YouTube content. The

encoding rate applied to the videos corresponds to the resolutions QVGA (0.4 Mbps), VGA (0.7 Mbps), SD (2 Mbps) and HDTV (3.3 Mbps) as indicated in Table IV. For the losses in the mobile channel, we have considered the two different reception scenarios evaluated in II.2, characterized by packet error rates of 5% and 50%.

For every file, the mean number of carousel cycles is obtained from the model for the carousel cycles presented in section II.2.2. On the other hand, for every simulation, we select a random instant of time in the duration of the video for the start of the carousel.

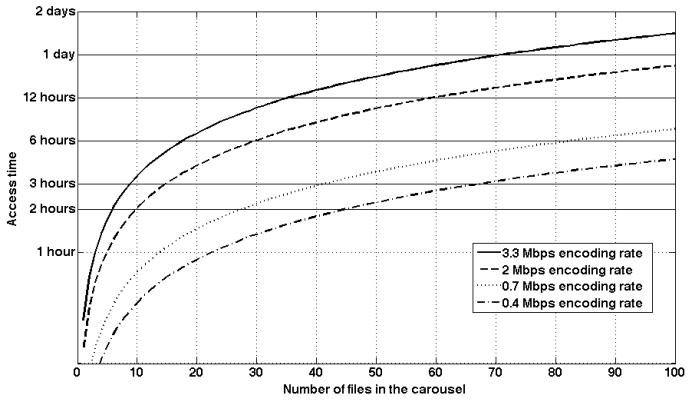

Figure 55. Access time for a background CDS with Youtube content encoded at different rates over a communication channel with 5% channel losses.

Figure 55 shows the access time of the background CDS with a packet error rate of 5% in the communication channel. As in the previous study case, the access time depends to a great extent on the encoding rate and the number of files in the carousel. Since the background CDS targets mobile devices, it is not feasible to assume that the receivers are connected to the background channel for a long time. For instance, in the examples in the previous section, it is assumed that the television sets can be connected to the background channel for approximately one day. This is not realistic for a mobile terminal that is not connected to the power supply. However, it is feasible to assume that the mobile terminal is connected to the wireless network during the duration of the television event,

approximately two hours for a football match. In the following analysis, this limit is used as the upper bound for the access time.

The access time found in the example shows that the background capacity can be used to provide a CDS to mobile terminals, especially for low resolution videos (VGA and QVGA). The access time for carousels of small size (4-5 items) is less than an hour for all the encoding rates, except for 3.3Mbps (HDTV). For instance, if the resolution of the additional content is set to VGA, it is possible to send several carousel sessions with 4 or 5 files during the duration of the match, ensuring that the interested clients are able to recover the different files. Moreover, it is possible to broadcast a carousel of more than 20 files during the whole football match.

On the other hand, Figure 56 presents the access time for a background CDS over a channel with 50% packet error rate. Note that this is a quite high packet error rate, corresponding to rather bad reception conditions. As expected, the access times are much larger than in the previous case. The access time is greater than 2 hours for carousels of less than 10 files for all resolutions except QVGA.

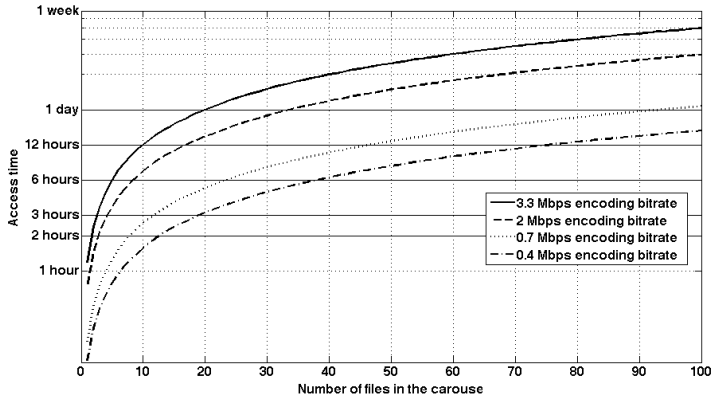

Figure 56. Access time for a background CDS with Youtube content encoded at different rates over a communication channel with 50% channel losses.

In the presence of channel losses, AL-FEC encoding can help decreasing the access time to files. The following chapter covers the improvement of the access time and the download time brought by AL-FEC encoding in the two reception conditions analyzed in this section.

IV.3 Conclusions

In this section we have evaluated two different background CDS services, one targeting television sets and one targeting mobile terminals. The service for television sets uses the background capacity available in a DVB multiplex. The simulations apply a model for the available bitrate that has been obtained from the measurements presented in section III.3.3. We have compared the average bitrate during the download of a file obtained with the model to the average bitrate obtained with the measurement traces. The results in both cases are analogous, proving the validity of the model. Later, we use the model to evaluate the carousel time and the access time for IMDB content and YouTube content. The results highlight the usefulness of the service, although the access times increase drastically with the number of files added to the carousel. In this sense, the following chapter presents object multiplexing as a technology that improves the scalability of the service with the number of files.

On the other hand, the service targeting mobile devices uses the remaining capacity available in a fixed capacity tunnel used to deliver a live television service. We have introduced packet losses in the simulations, according to the channel models presented in II.2. The results also highlight the utility of the service, but in this case, the packet losses increase the access times in a drastic manner. However, a service operator can apply AL-FEC encoding in order to reduce the number of cycles needed to download a file. The following chapter presents the improvements in the download time brought by AL-FEC encoding.

Chapter V

Object Multiplexing and AL-FEC

The goal of this chapter is to evaluate the improvement in the performance of background Content Download Services (CDSs) brought by two different technologies: object multiplexing and Application Layer Forward Error Correction (AL-FEC). The basis of Object Multiplexing consists of what is known as *weighted carousels*. In a weighted carousel, more popular files are sent more frequently, in order to reduce their download time (at the expense of increasing the download time of less popular files). Similarly, AL-FEC encoding consists of adding redundancy in the transmission of every file so that the number of cycles needed to download each file is reduced, at the expense of increasing the carousel time. The chapter is structured as follows; the next section presents a theoretical analysis of the download time and the access time in background unidirectional push CDSs with object multiplexing and AL-FEC. Later, section V.2 presents an evaluation of the access time in carousels with Object multiplexing. Finally, section V.3 presents an evaluation of the access time in carousels with AL-FEC.

V.1 Analysis of object multiplexing with AL-FEC

The first step in the analysis is to model the delivery of content download services, including the effect of AL-FEC, object multiplexing and opportunistic insertion. Later, this model is used to estimate the optimal configuration of object multiplexing. For the sake of

clarity, first we obtain a model incorporating object multiplexing, opportunistic insertion and losses and later, we incorporate AL-FEC encoding.

Thus, the unidirectional background push CDS delivers N files of sizes $S=\{s_1,s_2,...,s_N\}$ that have a certain access probability in the service area. The access probability is represented by $P=\{p_1,p_2,...,p_N\}$ where p_j is defined as the number of downloads of file j divided by the overall number of file downloads. The objective of object multiplexing is to schedule file transmissions so as 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 $\mathrm{E}[t_A^j]$. This way, taking into account the access probability to files in the service area, the overall access time is calculated as:

$$\mathrm{E}[t_A]=\sum_{j=1}^{N}\mathrm{E}[t_A^j]\cdot p_j=\sum_{j=1}^{N}\left(\mathrm{E}[t_W^j]+\mathrm{E}[t_D^j]\right)\cdot p_j \qquad (37)$$

As in the previous chapter, t_A^j depends on the waiting time t_W^j and the download time t_D^j. The first objective is to find an expression of Eq. 37, accounting for the effect of object multiplexing and opportunistic insertion. Figure 57 shows a new example of a transmission of three files, incorporating object multiplexing and losses.

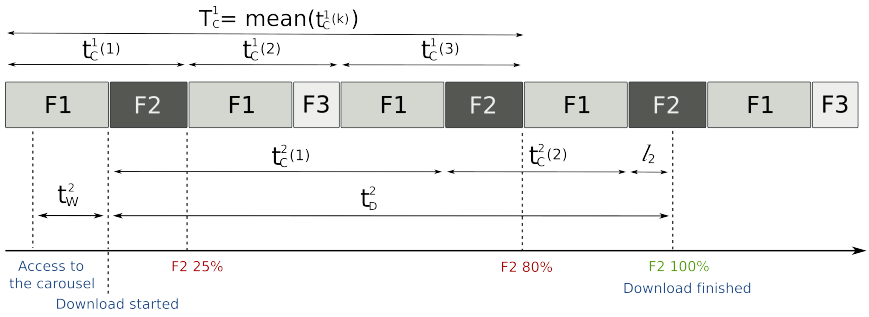

Figure 57. Time model of the access time with object multiplexing.

Figure 57 shows the purpose of object multiplexing: different files have different *sub-carousel cycle time*s, $t_C^j(k)$, which is defined as the maximum time containing the k^{th} transmission of file *j*. The idea is that most popular files have shorter sub-carousel cycle times so that in turn, they have shorter access times. Note that $t_C^j(k)$ is not necessarily the same for every sub-cycle index *k*, since object multiplexing and/or opportunistic insertion can change the sub-carousel cycle times. Hence, each $t_C^j(k)$ can be regarded as a random variable (see IV.1.2 for a detail explanation). Moreover, as depicted in Figure 57, both t_W^j and t_D^j depend on $t_C^j(k)$. Therefore, if T_C^j is defined as the *long-term carousel cycle* time of file *j* (also depicted in Figure 57), it is possible to obtain expressions of $\mathrm{E}[t_W^j]$ and $\mathrm{E}[t_D^j]$ depending on T_C^j. Later these expressions can be combined in Eq. 37, in order to obtain an expression for the access time depending on the different T_C^j. The set of values of T_C^j that minimize this expression are the optimal long-term carousel cycle times that the object multiplexing scheduler needs to apply in order to minimize the overall access time. Hence, Eq. 37 can be used to define the optimization problem that needs to be solved in order to achieve optimal object multiplexing.

Let us start with t_W^j. According to Figure 57, a client application joins the channel at an instant of time *t*, within sub-cycle *k*. Therefore, t_W^j is a statistical variable that depends on the time when the client application access the channel and the sub-carousel cycle time $t_C^j(k)$. Thus, in order to obtain $\mathrm{E}[t_W^j]$, it is necessary to regard all possible values of *t* and every possible *k*.

This way, considering the instant of time *t*, we assume that the client application can join the carousel at any time with the same probability within $t_C^j(k)$. Then, it is clear that the expected value for $t_W^j(k,t)$ is:

$$t_W^j(k) = \mathrm{E}[t_W^j(k,t)] = t_C^j(k)/2 \qquad (38)$$

Now, if we consider the sub-carousel index *k*, each carousel sub-cycle contains a different sequence of files (for instance in Figure 57, for F1, the sequence is either F1-F2 or F1-F3). Consequently, assuming that the instant when the client accesses the channel is uniformly distributed in time, the client does not join the channel at any sub-cycle with the same probability, just because every sub-cycle has a different duration and therefore, they do not have the same probability of occurrence. If ρ_k is the probability that the client joins the channel at sub-cycle *k*, the expected value of t_W^j, calculated in *K* different sub-cycles is:

$$\mathrm{E}\left[t_W^j\right] = \sum_{k=1}^{K} \rho_k t_W^j(k) = \sum_{k=1}^{K} \rho_k \frac{t_C^j(k)}{2} = \frac{T_C^j}{2} \tag{39}$$

Note that Eq. 39 implicitly defines T_C^j as the expected value of $t_C^j(k)$ in *K* cycles.

On the other hand, regarding t_D^j, as shown in Figure 57, due to the channel losses, clients need several carousel sub-cycles to download a file. Therefore, t_D^j is also a statistical variable that depends on the number of cycles needed to download the file and, just as with t_W^j, it is necessary to obtain an expression regarding all possible values. This way, the expression of the average download time when the file starts the download at cycle *k,* $t_D^j(k)$ is:

$$\mathrm{E}[t_D^j(k)] = \sum_{i=k}^{k+\bar{c}_j} t_C^j(i) + \mathrm{E}\left[\frac{s_j \cdot l_j}{b}\right] \tag{40}$$

Note that this equation is the same as Eq. 34 in section IV.2.1, but it is included here too for the sake of clarity. Hereby the same notation is also used: $\bar{c}_j$ is the average number of cycles and l_j is also the percentage of the file that is downloaded in the last cycle. In order to obtain the expression for the download time when the download starts at any cycle t_D^j, we make an average on *K* different cycles as with t_W^j:

$$\mathrm{E}[t_D^j] = \sum_{k=1}^{K} \rho_k \cdot \mathrm{E}[t_D^j(k)] = \sum_{k=1}^{K} \rho_k \cdot \left(\sum_{i=k}^{k+\overline{c}_j} t_C^j(i) + \mathrm{E}\left[\frac{s_j \cdot l_j}{b} \right] \right) \tag{41}$$

Both summations (over *k* and *i*) add cycles of the same file *j*. Hence, 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:

$$\mathrm{E}[t_D^j] = \overline{c}_j \sum_{k=1}^{K} \rho_k \cdot t_C^j(k) + \mathrm{E}\left[\frac{s_j \cdot l_j}{b} \right] = \overline{c}_j T_C^j + \mathrm{E}\left[\frac{s_j \cdot l_j}{b} \right] \tag{42}$$

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

$$\mathrm{E}[t_A^j] = \mathrm{E}[t_W^j] + \mathrm{E}[t_D^j] = T_C^j \cdot \left(\overline{c}_j + \frac{1}{2} \right) + \mathrm{E}\left[\frac{s_j \cdot l_j}{b} \right] \tag{43}$$

Once we have obtained the average of the access time depending on the long-term carousel cycle, we are going to introduce the effect of AL-FEC encoding. 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. AL-FEC encoding reduces the average number of cycles $\overline{c}_j$ at the expense of increasing the amount data transferred when the file is transmitted. Recall that the ratio between the amount of data after AL-FEC encoding and the file size is defined by the FEC ratio, FR, (FR>1). Thus, AL-FEC increases the size of the data of files in the carousel to $s_j^{FEC} = s_j \cdot FR$. Moreover, if AL-FEC is applied, the long-term carousel cycle of each file is noted as $T_{C,FEC}^j$.

Now, recall that object multiplexing sends some files more often, in order to adjust the long-term carousel cycle of each file. The optimum configuration for the long-term carousel cycles provides a minimum of the overall access time, as defined in Eq. 37. Therefore, in order to determine the optimization problem, it is necessary to substitute $\mathrm{E}\left[t_A^j\right]$ in Eq. 37 for the expression obtained in Eq. 43 with AL-FEC encoding:

$$\mathrm{E}\left[t_A\right] = \sum_{j=1}^{N}\left(T_{C,FEC}^{j}\cdot\left(\overline{c}_j + \frac{1}{2}\right) + \mathrm{E}\left[\frac{s_j^{FEC}\cdot l_j}{b}\right]\right)\cdot p_j \tag{44}$$

Clearly not all $T_{C,FEC}^{j}$ are the same: files that are sent more often have shorter cycles and, in turn, shorter access times. This way, since files have different long-term carousel cycle times, files are not transmitted at the same rate in the long run. Therefore, files have different *long-term bitrates* and files with shorter long-term carousel cycles can be seen as files with higher long-term bitrates. The long-term bitrate assigned to file *j* (b_j) is:

$$b_j = \frac{s_j^{FEC}}{T_{C,FEC}^{j}} \tag{45}$$

Taking this into account, Eq. 44 can be rewritten as:

$$\mathrm{E}[T_A] = \sum_{j=1}^{N}\frac{s_j^{FEC}}{b_j}\cdot\left(\overline{c}_j + \frac{1}{2}\right)\cdot p_j + \sum_{j=1}^{N}\mathrm{E}\left[\frac{s_j^{FEC}\cdot l_j}{b}\right]\cdot p_j \tag{46}$$

Note that, due to opportunistic insertion, the server works exactly at the available bitrate, *b*, establishing a constraint in the set of long-term bitrates:

$$\sum_{j=1}^{N} b_j = b \tag{47}$$

The only relationship between the different long-term bitrates is the boundary condition in Eq. 47, therefore they are independent variables, but subject to that condition. Thus, the following auxiliary function is used to solve the optimization problem:

$$\mathrm{f}\left(b_1, b_2, ..., b_N\right) = \sum_{j=1}^{N}\frac{s_j^{FEC}\cdot\left(\overline{c}_j + 1/2\right)\cdot p_j}{b_j} + \varphi\left(b - \sum_{j=1}^{N} b_j\right) \tag{48}$$

In order to find the minimum of Eq. 48, its derivate is equaled to zero:

$$\frac{\delta \mathrm{f}(b_j)}{\delta b_j} = -\frac{s_j^{FEC} \cdot (\overline{c}_j + 1/2) \cdot p_j}{b_j^2} + \varphi = 0 \tag{49}$$

Providing the set of optimal bitrates:

$$b_j = \frac{b\sqrt{s_j^{FEC} \cdot (\overline{c}_j + 1/2) \cdot p_j}}{\sum_{i=1}^{N} \sqrt{s_i^{FEC} \cdot (\overline{c}_i + 1/2) \cdot p_i}} \tag{50}$$

Hence, by applying these optimal bitrates, the multiplexing process takes into account the file sizes after AL-FEC encoding, the expected number of cycles and the access probability in order to calculate the set of optimal rates. Recall that Chapter II presented models for the file sizes, the access probability and the average number of cycles (dependent on the AL-FEC rate, the file size and the channel losses). In the next section, object multiplexing is applied to the use cases presented in section IV.1 to evaluate the improvements in the access time introduced with this technique.

V.2 Object Multiplexing

V.2.1 Lower bounds of object multiplexing with ODI

In this section, we evaluate the average access times achieved with object multiplexing, applying the optimal long-term bitrates obtained in the previous section. These results can be regarded as the lower bounds for the access time, that is, the lowest access times that can be achieved with object multiplexing. The objective is to assess the potential gain of object multiplexing. With this motivation, we compare the lower bounds against the access times achieved with no object multiplexing, in the use cases defined in this thesis.

First we are going to evaluate the access time without channel losses ($\overline{c_j} = 1 \ \forall \ \mathrm{j}$) and no AL-FEC encoding (FR=1). Thus, in order to evaluate the lower bounds, we need to characterize the available bitrate, the file size and the probability of access. The available bitrate is obtained from the results in section IV.1 *Background services for television sets*. Recall that this section presented the bitrate available for opportunistic insertion in two

different commercial multiplexers, referred to as C28 and C57. On the other hand, we use the IMDB and YouTube content models for the file size and the probability of access.

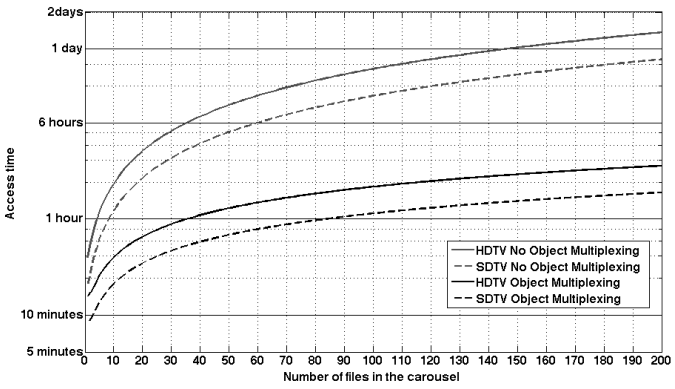

Figure 58. Access time for YouTube content encoded at 3.3 Mbps and 2 Mbps with and without object multiplexing in channel C28.

Let us start with the evaluation of object multiplexing over channel C28, delivering YouTube content. Figure 58 represents the access time achieved for YouTube files, applying encoding rates of 2Mbps and 3.3Mbps (corresponding respectively to the SDTV and HDTV formats). The popularity of the content is also set according to the model for YouTube content in section II.1.3, a ZIPF with exponential cut-off ($\alpha = 0.84$, $\chi = 10^{-3}$). For each carousel size, the access time is the average over 500 simulations.

The main conclusion of the result is that object multiplexing can reduce the access time drastically. The access times achieved with object multiplexing are significantly lower, especially when the number of files in the carousel is large. Although the figure shows the lower bounds for the access time, obtained from the theoretical analysis in the previous section, the differences show the great potential of object multiplexing, especially for large carousels.

In order to evaluate the relationship between content popularity and object multiplexing, Figure 59 shows the object multiplexing gain for the different popularity distributions

presented in section II.1.3. The object multiplexing gain is defined as the ratio between the access time without and the access time with object multiplexing.

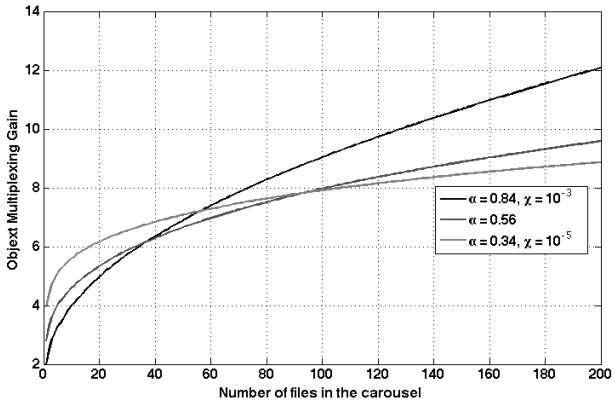

Figure 59. Object multiplexing gain for IMDB content in C28 with different probability distributions.

The popularity distributions under evaluation are the ZIPF distribution with exponential cutoff ($\alpha=0.84$ *and* $\chi=10^{-3}$) of YouTube content [18], the ZIPF distribution of web content ($\alpha=0.56$) [22] and the ZIPF distribution with exponential cutoff ($\alpha=0.35$ *and* $\chi=10^{-5}$) of IMDB content in II.1.3. Comparing the YouTube popularity distribution and the IMDB popularity distribution, it can be noted that the relationship between the object multiplexing gain and the α coefficient is different for large carousels and small carousels. For instance, the object multiplexing gain for 20 file carousels is 5 for α = 0.84 and 6.15 for α = 0.34. However, for 100 file carousels the object multiplexing gain is 9 for α = 0.84 and 8 for α = 0.34. This can be extended to the comparison of any two distributions: the distributions where the popularity decreases more slowly (smaller α) exhibit higher object multiplexing gains for small carousels than distributions with greater α. However, the tendency is the opposite for larger carousels.

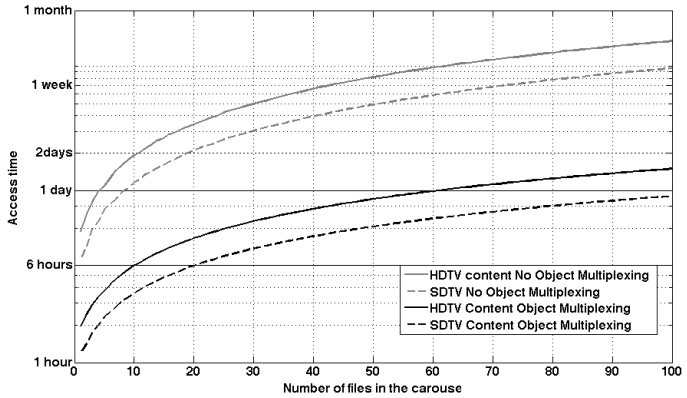

Figure 60. Access time for IMDB content encoded at 3.3 Mbps and 2 Mbps with and without object multiplexing in C28.

On the other hand, Figure 60 shows the access time for IMDB content over channel C28, with the same encoding bitrates as the previous example (2 Mbps, SDTV and 3.3Mbps, HDTV). The popularity for the content follows the ZIPF distribution with exponential cutoff of IMDB content ($\alpha=0.34;\ \lambda=10^{-3}$). The results are almost equivalent to the previous study case. Similarly, the traces of the object multiplexing gain for IMDB content and different popularity distributions are depicted in Figure 61.

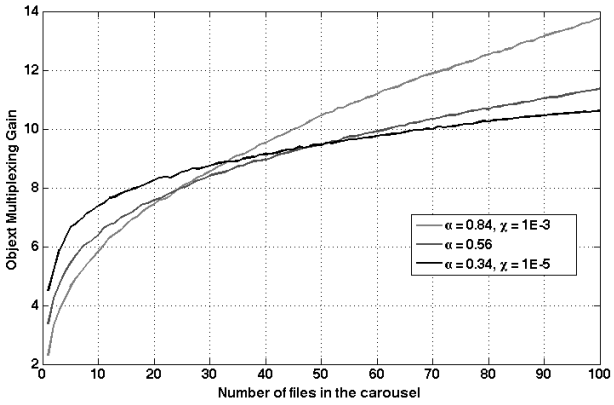

Figure 61. Object multiplexing gain for IMDB content in C28 with different popularity distributions.

Compared to the YouTube case, it can be noted that, for the same probability distribution, the object multiplexing gain is slightly higher for IMDB content. This can be explained by the fact that the content lengths of IMDB content items are more variable than the content lengths of YouTube content (section II.1.1). This variability favors the object multiplexing gain.

Figure 62 shows the access time accomplished with object multiplexing on the bitrate available for ODI in C57. It can be noted that the reduction in the access time is equivalent in the two multiplexers under study. In this sense, the figure is included only for information purposes, to show the access times achieved with object multiplexing at a different mean bitrate.

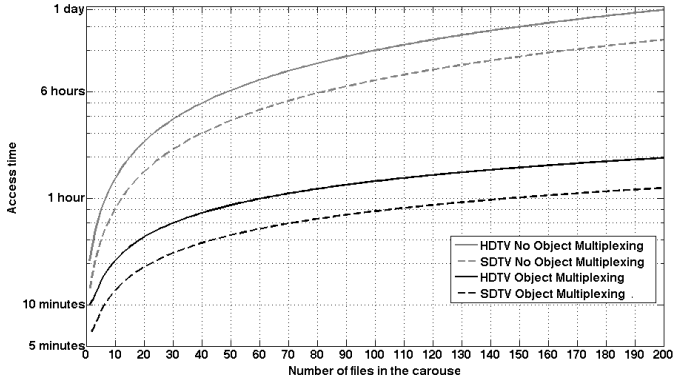

Figure 62. Access time for Youtube content encoded at 2 Mbps and 3.3 Mbps with and without object multiplexing in C57.

In summary, the potential benefits of object multiplexing in background Content Download Services (CDS) are remarkable. The simulations showed that the potential gain (reduction in the access time) in the different cases under study is really high. For instance, the access time with object multiplexing can be reduced up to 9 times for carousels of 100 YouTube files, using the file size and the popularity distribution of YouTube content, whereas the access time can be reduced by a factor higher than 8 for 20 IMDB files, using the models for IMDB content.

Nevertheless, these results should be regarded as theoretical upper bounds. There are two main factors that can affect the actual multiplexing gain: First, the quality of the estimation of the long-term bitrate and second, the accuracy of the scheduler in achieving the long-term bitrates.

Regarding the quality of the estimation, it is important to highlight that the optimal bitrates depend on the access probability and the expected number of cycles. For this reason, the service operator needs to estimate the access probabilities and the channel losses in order to apply object multiplexing.

Specifically, the access probability can be estimated from audience measurements or access statistics on the primary services. Network operators invest a lot of money in audience metrics, which are taken into account when shaping the program grid of their television channels. On the other hand, frequency network planning tools (used to decide the location of transmitters and repeaters) need an estimation of the channel losses found in the service area. The accuracy of network planning tools has a direct impact on the network infrastructure costs. In summary, the parameters necessary to compute the optimal bitrates can be estimated from operational data of broadcasters. Therefore, it is feasible to assume that an accurate estimation of the long-term bitrate is available. In any case, the quality of the estimation has an impact on the object multiplexing gain in an actual implementation. Quantifying this impact is out of the scope of this thesis.

On the other hand, the scheduler needs to create a carousel that provides the estimated long-term average bitrates, by multiplexing the different files in the time domain. The algorithms or heuristics used by the scheduler may not be able to achieve the estimated long-term bit rates, because bandwidth is not infinitively divisible. This reduces the actual object multiplexing gain, compared to the theoretical limits, even if the estimations of the popularity and the average number of cycles are very accurate. The next sections present algorithms to implement object multiplexing.

V.2.2 The Modified Virtual Clock Algorithm

The optimal bitrates obtained in section V.1 provide an optimum share of the available bandwidth between the different content items. The last section showed that sending files at exactly those long-term bitrates could reduce the access time by factors of 8 and 9, for relevant carousel sizes of IMDB and YouTube content.

The fundamental problem behind object multiplexing is to schedule the transmission of objects of different sizes 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 [4], hereby adapted to account for the channel losses.

Algorithm I contains lines of pseudo-code describing how the MVC algorithm works. The MVC algorithm has two different phases. In the initialization phase the algorithm assigns to each file *j* a *delay*, which is a value directly proportional to the long-term carousel cycle $T_{C,FEC}^{j}$. Note that the delay in the equation is calculated applying the analytical expression for the minimum long-term carousel cycles.

In the multiplexing phase, the algorithm sorts the files according their delay value (*QueueObjectsByOrderIncreasingTag*), so that the files with shorter carousel cycles appear first in the sorted queue. Thus, in the multiplexing phase, the algorithm tries to adjust the cycle period of each data element according to their delay, placing data elements in a multiplexing queue that is ordered by increasing delay values. This way, the algorithm tries to adjust the long-term bitrates of files to the optimal values calculated in the previous section. It is worth noting that hereby we have introduced some modifications to the algorithm in [4]. First, the delays account for losses and AL-FEC encoding, by applying the values of the optimal bitrates obtained in this thesis. Additionally, the *while(ActiveQueueEmpty)* loop (line 9) is introduced so that the algorithm is work conserving, in the sense that there is always files in the active queue.

ALGORITHM I: MODIFIED VIRTUAL CLOCK (MVC) ALGORITHM

Phase 1: Initialization

1: **for** j=1 to N

2: object(j).delay= $\sum_{i=1}^{n} \sqrt{s_i^{FEC} \cdot (\overline{c}_i + 1/2) \cdot p_i} \; / \sqrt{s_j^{FEC} \cdot (\overline{c}_j + 1/2) \cdot p_j}$

3: object(j).tag = object(j).delay

4: object(j).count = 0

5: object(j).enabled = TRUE

6: QueueObjectsByOrderIncreasingTag()

7: **end**

Phase 2: Multiplexing

8: **while** (not_exit)

9: **while** (ActiveQueueEmpty)

10: **for** j=1 to N

11: object(j).count = object(j).count + 1

12: **if** (object(j).count >= object(j).delay)

13: object(j).count = object(j).count-object(j).delay

14: object(j).enabled = TRUE

15: **end**

16: **end**

17: l = FindObjectWithLeastTagInActiveQueue()

10: SendObject(l)

11: object(l).tag = object(l).tag + object(l).delay

12: object(l).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**

In order to show the performance of the MVC algorithm, we include a comparison between the optimal long-term bitrates obtained analytically and the bitrates achieved with the MVC algorithm.

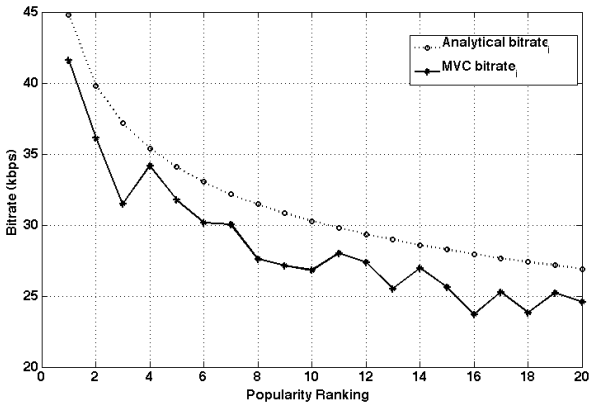

Figure 63. Long-term bitrates achieved with the MVC algorithm (YouTube content) in C57.

Figure 63 shows the mean long-term bitrate achieved with the MVC algorithm for every file in 20 file carousels, sorted by their popularity ranking. For this example, we use a ZIPF distribution with $\alpha=1$. Figure 63 shows the average computed over 500 different simulations, each with different file sizes generated with the YouTube file size model. The simulations account for MPEG-TS encapsulation and ODI, using the channel model for C57. It can be noted that the object multiplexing technique provides slightly lower values than the analytical values, although the difference is, in general, small. However, this assumption does not hold if we regard the bitrates of a specific carousel, instead of the average on 500 different iterations.

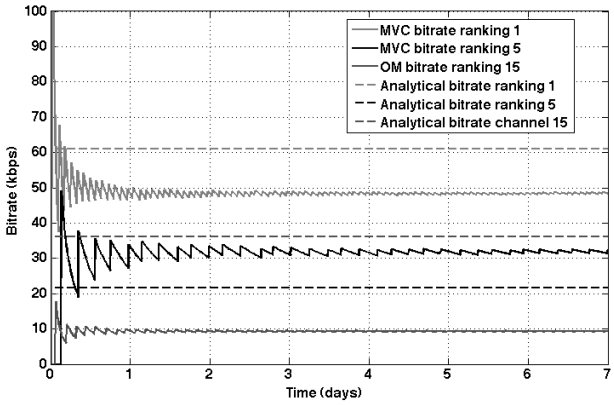

Figure 64. Bitrates achieved with MVC for files with different rankings (1, 5 and 15) using ODI in C57.

Figure 64 shows the bitrate achieved for three content items with different rankings, together with the analytical long-term bitrates that the MVC needs to accomplish. First it is important to note the convergence of the bitrates of the MVC algorithm with time. The three traces tend to steady values. In this sense, the MVC algorithm manages to adjust the relative bitrates of files to different values in time. The problem is that these values do not match with the ideal bitrates obtained analytically. This can be noted by comparing the different MVC bitrates with their respective ideal bitrates. Therefore, by evaluating the long-term bitrate for any particular carousel, it can be concluded that the MVC algorithm fails to adjust the long-term bitrates to the desired values.

There are two reasons for these deviations. The first of them is a fundamental problem of resource management: Resources like bandwidth or time are not infinitively divisible. For this reason, it is not possible to fulfill the optimal bitrates exactly at all times. This is the reason why the MVC bitrate trace has a “saw teeth” shape. The other reason is that the MVC as defined in [4] does not manage appropriately carousels composed of files of different sizes. As noted in the MVC algorithm, the object count is incremented in 1 unit every time a file is transmitted, regardless of the size of the files. For this reason, the algorithm does not actually account for the size of files appropriately.

V.2.3 The Modified Weighted Fair Queuing Algorithm

The Modified Weighted Fair Queuing (MWFQ) algorithm is a modification of the Weighted Fair Queuing (WFQ) algorithm that tries to apply the principles of Fluid Fair Queuing (FFQ) to file scheduling. [58] provides a good overview of the principles of FFQ as well as describing the VC and WFQ algorithms among other packet switching algorithms. An FFQ system is characterized by the set of bitrates $[b_1, b_2, \ldots b_N]$ of N different flows. Then, at any time t, the service rate for a non-empty queue i is exactly $b_i \cdot C \Big/ \sum_{j=1}^{N} b_j$, where C is the channel capacity. Since bandwidth it is not infinitively divisible, real life algorithms can only approximate the FFQ system performance. For instance, the WFQ algorithm selects items from the active queues according to their service times in the corresponding FFQ system at the instants of time when the service needs to schedule an output packet.

This way, at every time t a WFQ algorithm selects the item with the lowest FFQ service time, that is the item with the shortest deadline for its ideal scheduling time. Applied to file scheduling, the FFQ service time for the k^{th} transmission of file j is:

$$t_{FFQ}^{j}(k) = k \cdot s_j^{FEC} / b_j = \frac{k \cdot s_j^{FEC} \sum_{i=1}^{N} \sqrt{s_j^{FEC} \cdot (\bar{c}_j + 1/2) \cdot p_j}}{b \sqrt{s_j^{FEC} \cdot (\bar{c}_j + 1/2) \cdot p_j}} = k \cdot t_{FFQ}^{j} \qquad (51)$$

The WFQ principle applied to file scheduling yields the following algorithm:

ALGORITHM II: MODIFIED WEIGHTED FAIR QUEUING (MWFQ)

```
Phase 1: Initialization
1:    for i=1 to N
2:        object(i).step= t^i_FFQ
3:        object(i).tag = object(i).step
4:        object(i).count = 0
5:        object(i).enabled = TRUE
6:        QueueObjectsByOrderIncreasingTag()
7:    end
Phase 2: Multiplexing
8:    while (not_exit)
9:        while (ActiveQueueEmpty)
10:           for j=1 to N
11:               object(j).count = object(j).count + 1
12:               if (object(j).count >= object(j).step)
13:                   object(j).count = object(j).count-object(j).step
14:                   object(j).enabled = TRUE
15:               end
16:       end
17:       k = FindObjectWithLeastTagInActiveQueue()
10:       SendObject(k)
11:       object(k).tag = object(k).tag + object(k).step
12:       object(k).enabled = FALSE
13:       for l=1 to N
14:           object(l).count = object(l).count + 1
15:           if (object(l).count >= object(l).step)
16:               object(l).count = object(l).count-object(l).step
17:               object(l).enabled = TRUE
18:           end
19:       end
20:   end
```

The main difference with respect to the MVC algorithm is that, in the multiplexing phase, the algorithm tries to adjust the transmission time of every file to the corresponding time in the FFQ system, instead of trying to adjust the long-term bitrate of each file.

At this point, let us compare the performance of the algorithm against the analytical bitrates. Figure 65 represents the average long-term bitrate achieved by the MWFQ algorithms, using 500 simulations with carousels of 50 files and applying the YouTube file model with $\alpha=1$ for the ZIPF popularity distribution. The carousels use the available bitrate in channel C57.

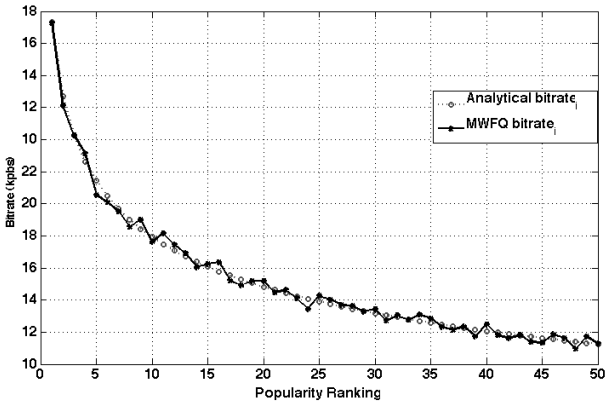

Figure 65. Long-term bitrates achieved with the MVC algorithm with ODI in C57.

Also, as in the previous case, let us use one of the iterations to compare the long-term bitrates achieved with the MWFQ algorithm against the ideal bitrates. Figure 66 shows the MWFQ long-term bitrate of files with popularity rankings 1, 5 and 15 out of the 50 files in the carousel (instead of the 20 files used to evaluate the MVC algorithm). Note that the long-term bitrates of every file are lower than those of Figure 64, simply because there are more files in the carousel. Actually, we have selected two different carousel sizes because the MVC algorithm did not manage to adjust the bitrate for large carousels, due to its problems to deal with different file sizes. On the other hand, the bitrates achieved by the MWFQ algorithm are the same as the analytical ideal bitrates.

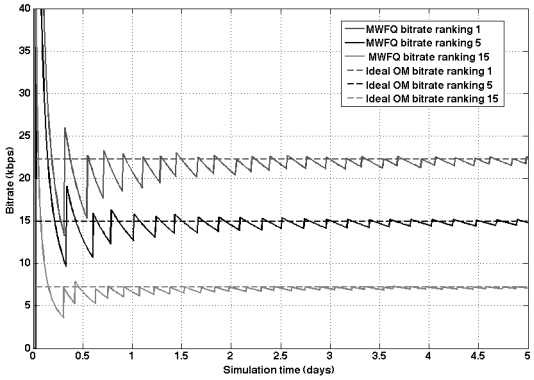

Figure 66. Bitrates achieved with MWFQ for files with different rankings (1, 5 and 15) using ODI in C57.

Note that this time, the bitrates do converge to the ideal values in the three cases highlighted. Finally, in order to compare both algorithms, Figure 67 highlights the average error in the long-term bitrate accuracy -i.e. the difference between the ideal bitrate of a file and the long-term average bitrate achieved with the object multiplexing algorithm. The bitrate accuracy is displayed against the difference between the mean file size and the actual file size.

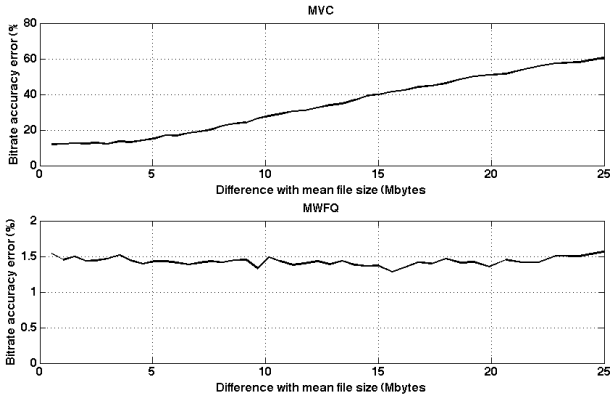

Figure 67. Comparison of the bitrate accuracy achieved by the MVC and the MWFQ algorithms.

The figure highlights that the MVC algorithm does not manage non-constant file sizes appropriately. For instance, the average error found in files that are 10 Mbytes larger or smaller than the mean file size is approximately 28% of the optimal weight. Note however that for some applications this might not be a problem, as long as the file size is relatively constant. On the other hand, the bitrate accuracy of the MWFQ algorithm does not depend on the differences in file sizes: The value is approximately 1.5% of the optimal carousel weight regardless of the differences between the file size and the mean file size.

As a summary of this section, if we consider that the access probabilities are not the same for every file in the carousel, the average access time can benefit from object multiplexing. We have presented the potential gain of object multiplexing for different access probabilities, considering both YouTube and IMDB content. The results showed the great potential benefits of object multiplexing. We have evaluated the performance of two packet scheduling algorithms adapted to work with file scheduling, the MVC algorithm, originally proposed in [4] and the MWFQ algorithm, proposed in this thesis. The results show that the MVC does not achieve the optimal long-term bitrates when the sizes of files are not sufficiently similar. Unfortunately, this is the case for the two content catalogues under study in this thesis work. However, the MWFQ provides long-term bitrates only about 1.5% different from the optimal values, thus adjusting the bitrate to the analytical optimal values.

V.3 AL-FEC

The last section showed the reduction in the average access time and the download time obtained with object multiplexing in broadcast carousels. This section focuses on the evaluation of the reduction achieved with AL-FEC, completing the theoretical analysis of the effect of AL-FEC presented in section V.1.

The trade-offs of AL-FEC have been discussed already: AL-FEC parity increases the size of the carousel, but this redundant information helps receivers to recover the files in fewer cycles. The model for the average number of cycles needed to download a file, depending on the channel losses, the amount of AL-FEC parity and the file size is presented in section

II.2.3. In this section, this model is used to evaluate the effect of AL-FEC in the carousel time and the access time of file carousels.

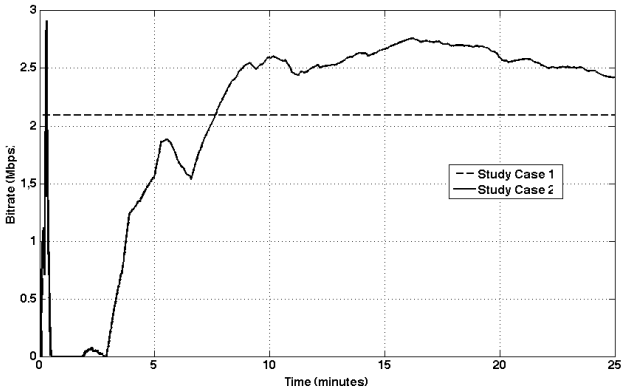

Figure 68. Traces of the bitrate used in the two study cases for AL-FEC.

The evaluation considers two different study cases. First, we regard a channel with a constant capacity of 2.1Mbps. This scenario helps providing an overview of the performance of AL-FEC encoding in unidirectional push CDS, as well as establishing the relationships between AL-FEC encoding and object multiplexing. Later, AL-FEC encoding is applied to the unidirectional background push CDS for mobile terminals presented in section IV.2, using the available bitrate for opportunistic insertion in a tunnel of 10Mbps. The bitrate capacity of the two study cases is depicted in Figure 68. As shown in the figure, the bitrate achieved with opportunistic insertion changes with time. However, both scenarios have the same average capacity (that is, the average of the background capacity is 2.1Mbps).

Another objective of the evaluation is to provide guidelines for planning background push content delivery services in packet erasure channels. A question to ask then is: What is the best AL-FEC encoding rate to apply in a given scenario?

A thorough answer to this question is outside the scope of this thesis, because the topic is sufficiently complex to be treated separately. In fact, we have contributed with different

proposals [32], [33] as well as with the undergoing thesis of the main author of the aforementioned research papers. Hereby, AL-FEC is regarded as a technology enabler that reduces in a drastic manner the number of carousel cycles needed to download a file. We assume that the service operator applies the same AL-FEC code (for instance LDPC) and encoding rate to all files in the carousel. Later, we evaluate the improvement in the average download time in two different extreme situations: very low packet losses (5%) and very high packet losses (50%). These study cases only regard homogeneous packet losses in the service area, that is, all users experience the same packet loss rate at all times. Obviously, the packet losses in wireless networks are heterogeneous and therefore, the study cases are not situations found in real life wireless networks, where the packet loss ratio experienced by each user inside the service area depends on the characteristics of their respective communication channel towards the server. In our methodology, we evaluate two extreme situations, assuming that it is possible to extrapolate the results to intermediate loss rates and, in extension, to the specific distribution of packet loss rates of a given scenario. With this approach, we provide a high level evaluation of AL-FEC encoding in unidirectional background push CDSs, lacking the accuracy of more sophisticated simulations, but sufficiently strict to draw general conclusions.

Therefore, in order to determine which AL-FEC parity configuration provides the best service performance, the first results in V.3.1 evaluate the performance of different configurations of AL-FEC encoding – adding no AL-FEC parity, 5%, 10%, 25% or 50% AL-FEC parities - with the two different packet loss ratios mentioned above. Later, the results evaluate what is the maximum number of files that can be transmitted in a carousel, provided that the service operator wants to guarantee a minimum bitrate for the service. Additionally, we present the download times for different carousel sizes, in order to relate with previous sections. In section V.3.2, These results are repeated with and without object multiplexing, to analyze the relationship between AL-FEC and object multiplexing. Finally, section V.2.3 evaluates the effect of using a background channel instead a constant bitrate channel.

V.3.1 Improvement of the download time with AL-FEC

Figure 69 represents the Complementary Cumulative Density Function (CCDF) of the goodput (the bitrate perceived at application level) during the download of a file from a carousel with 100 items in the two scenarios regarded in the simulations. The goodput is calculated by dividing the file size by the download time of every file in the carousel, taking into account the different iterations in the simulations. Thus, the goodput accounts for varying file sizes, carousel periods and retransmissions. The files are generated using the YouTube file size model, applying an encoding rate of 2Mbps.

The CCDF of the goodput can be useful for planning background CDS services, because it allows establishing a lower bound for the goodput experienced by a percentage of users during the download of a file. As an example, we look at the AL-FEC configurations providing the highest goodput in each scenario: 10% AL-FEC parity with 5% channel losses and 50% AL-FEC parity with 50% channel losses. This way, in the 5% packet loss rate scenario, the CCDF for 10% AL-FEC parity shows that the goodput is higher than 15kbps in 90% of the cases. Likewise, the goodput is higher than 2kbps in 90% of the cases, with 50% AL-FEC parity and 50% packet losses.

In addition to this, Figure 69 shows the ranges in which the goodput varies. The results show that the goodput varies in a wider range when AL-FEC parity is added to the carousel. This is due to the relationship between the channel losses, the AL-FEC parity and the inefficiency ratio (i.e. the ratio between the amount of data needed by an AL-FEC decoder to recover the contents of the file and the file size). Ideally, the sender should include at least an amount of AL-FEC such that, after one cycle, there are enough packets at the input of the receiver to decode the file.

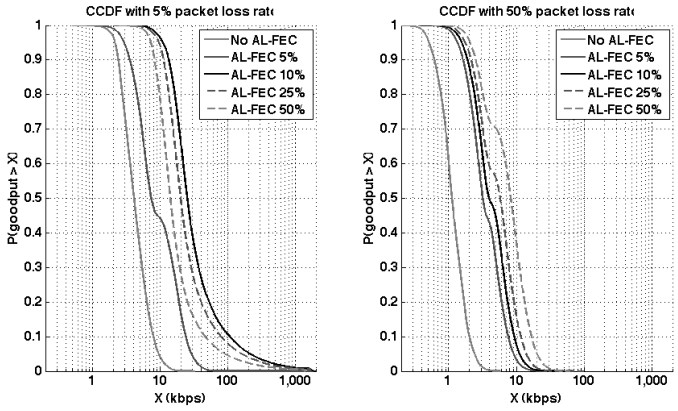

Figure 69. CCDF of the goodput in a 100 files carousel with no object multiplexing.

Furthermore, in [32] we show that the inefficiency ratio of LDPC AL-FEC codes is slightly higher than one when the percentage of AL-FEC parity added is near the channel packet loss rate. Consequently, the optimal AL-FEC parities for a single file with the channel loss rates under study should be around 10% for a 5% packet loss rate and over 50% with a 50% packet loss rate. Figure 69 confirms that this result holds in file carousels. In both scenarios, the lowest download time among the different configurations corresponds to the optimal AL-FEC encoding rates. Adding less parity provides shorter carousel times, but higher number of cycles. On the other hand, adding more AL-FEC parity provides longer carousel times, but lower number of cycles. Therefore, the other configurations provide lower goodputs. Moreover, adding any amount of AL-FEC to the carousel improves the goodput to a great extent. This is because, in both scenarios, AL-FEC reduces significantly the number of cycles needed to download the files.

Moreover note that the CCDFs of the AL-FEC configurations are wider than the CCDFs with no AL-FEC parity. This is due to the variation in the instantaneous channel losses: In some cases, it can happen that the receiver requires several cycles to recover the file, because the amount of AL-FEC is too short; In some other cases, the goodput is penalized because the amount of AL-FEC parity is too large. These factors change the shape of the CCDF, increasing the span of possible goodput values.

Figure 70 shows the CCDF of the goodput for different carousel sizes in the two channel losses under study, applying the AL-FEC encoding rate that provides the best goodput in each scenario. The figure highlights how the goodput decreases when the number of files added to the carousel is increased. This plot is useful to estimate the number of files that can be added to the carousel, provided a lower bound on the goodput. For instance, if a service operator wants a minimum goodput of 50kbps in 90% of the cases with a packet loss rate of 5%, the carousel should not include more than 25 files.

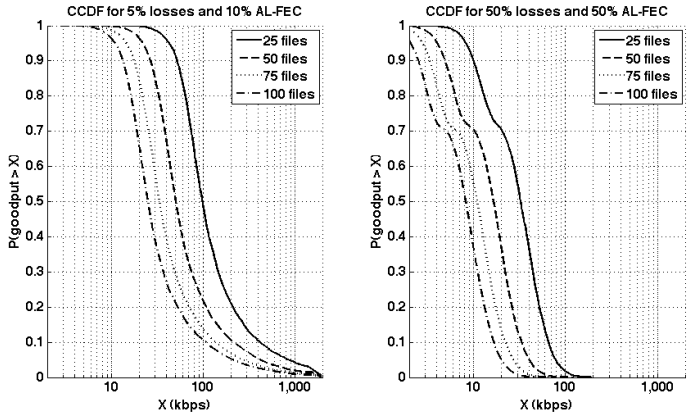

Figure 70. CCDF of the goodput for different carousel sizes with no object multiplexing

Figure 71 presents the mean download with a packet loss rate of 5%. All the configurations of AL-FEC improve the download time achieved with no AL-FEC parity. For instance, for carousels of 100 files, the average download time with no AL-FEC parity is almost 1 day. The addition of only 5% AL-FEC parity reduces the download time approximately to 12 hours. With 10% AL-FEC encoding, the download time is approximately 6 hours.

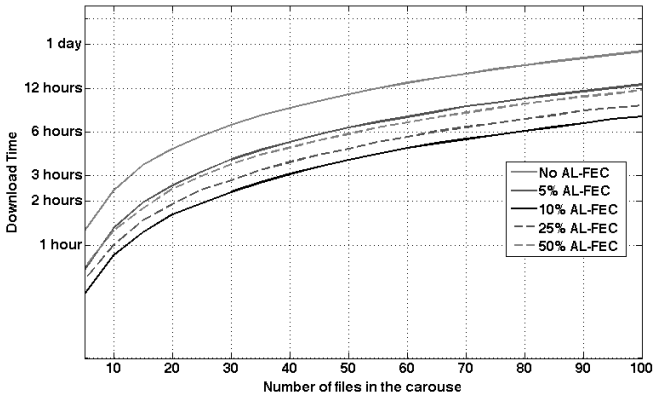

Figure 71. Mean download time on a channel with 5% packet losses without object multiplexing.

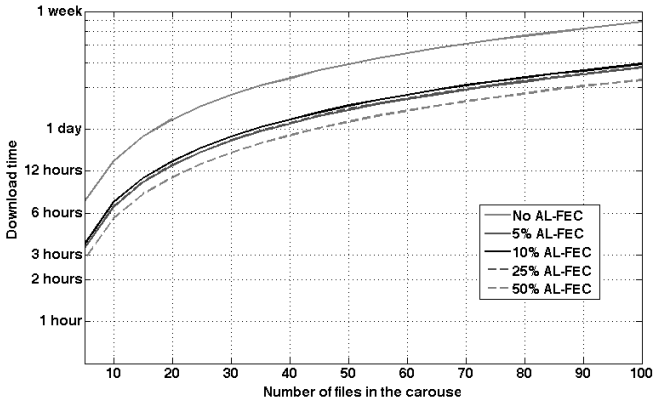

Figure 72. Mean download time on a channel with 50% packet losses without object multiplexing.

On the other hand, Figure 72 shows the mean download times achieved with the same AL-FEC encoding rates in a channel with 50% packet loss ratio. As expected, the download times are a lot higher than in the previous study case, but again, they are drastically reduced with AL-FEC encoding. The average download time for carousels with 50 files is around 3 days. Applying AL-FEC encoding with 5% parity reduces the average download time down to 36 hours. If the AL-FEC parity is increased to 50%, the average download time is less than 24hours.

In order to show clearly the improvement in the download time, Figure 73 shows the reduction in percentage of the download time without AL-FEC encoding, achieved by the different configurations of AL-FEC in the two cases under study. The results display the mean values and the 95% confidence interval.

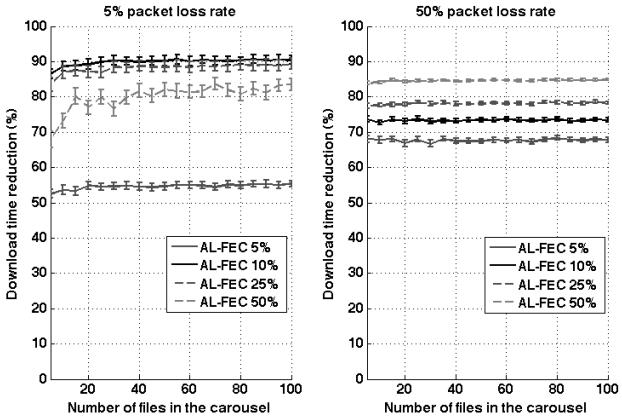

Figure 73. Reduction of the download time for different AL-FEC configurations.

Note that the reduction is more or less independent of the number of files in the carousel in all cases. The trace of 50% AL-FEC parity and 5% packet loss rate seems to be more dependent of the carousel size, but the variations are very similar to the confidence intervals. The relative reduction is really high in all cases. It can be noted that an AL-FEC parity of 25% provides a high reduction in both packet loss scenarios: 82% reduction with 5% channel losses and 78% with 50% channel losses. These values are close to the download time reduction achieved with the optimal configurations.

Now, let us take back the initial considerations opening this section: a) The service operator applies the same AL-FEC encoding rate to all files and b) users experience channel losses distributed in a range between 5% and 50%. Clearly, the actual rate providing the minimum mean download time depends on the specific distribution of channel losses experienced by users. Applying this optimum encoding rate would require a feedback channel from the clients back to the server and also, applying the AL-FEC encoding rate in a dynamic way,

as suggested in [33]. If this is not possible, a service operator willing to bound the access time to files should probably apply an amount of AL-FEC parity, sufficiently high to guarantee a low number of cycles with the minimum amount of channel losses expected in the service area, but sufficiently low to not penalize too much the access time of clients with a good connection. Moreover, looking at the results for 5% losses in Figure 73, it can be noted that the addition of more AL-FEC parity than 10% does not degrade the access time as much as having too less parity. Hence, the addition of more AL-FEC parity than the optimum for the minimum expected losses should provide a good trade-off for intermediate loss rates. In this sense, the addition of 25% AL-FEC parity provides high reductions in both scenarios under evaluation. In any case, in order to find the best configuration for a given scenario, it is necessary to conduct measurements with the specific channel loss rates found in the particular study case.

V.3.2 AL-FEC in combination with Object Multiplexing

The following results show the improvement in the download time achieved by AL-FEC encoding in carousels with object multiplexing. Recall that the main objective of object multiplexing is to reduce the carousel time of most popular files, at the expense of increasing the carousel time for the less popular files. In channels without losses, the reduction of the carousel time leads to a reduction of the waiting time. In the presence of packet losses, the download time is also affected by the carousel time and it is expected that object multiplexing can also reduce the download time.

Let us start by analyzing the effect of adding AL-FEC at different encoding rates to a carousel of 100 files with object multiplexing. As in the previous case study, the file sizes are generated with the YouTube model, with a mean encoding rate of 2Mbps. Figure 74 shows the CCDF of the goodput of file carousels using object multiplexing in a channel with 5% channel losses and different AL-FEC encoding rates (No AL-FEC, 5%, 10%, 25% and 50% of AL-FEC parity).

Regardless of the channel losses, the CCDF with object multiplexing is very similar to the CCDF without object multiplexing depicted in Figure 69, except that the goodput is slightly

higher with object multiplexing (that is, the CCDFs appear to be shifted upwards). This means that object multiplexing does not alter the download time to the same extent as experiencing channel losses.

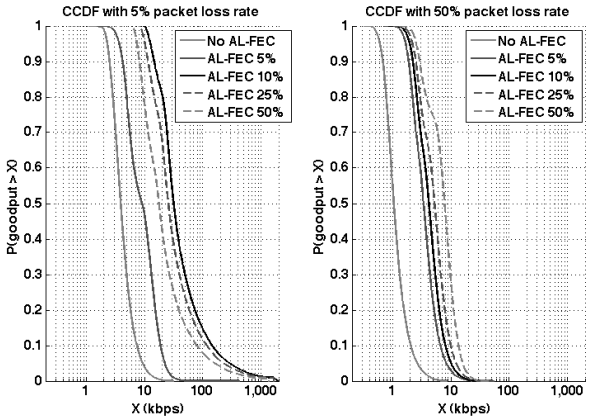

Figure 74. CCDF of the goodput with AL-FEC and object multiplexing.

Figure 75 shows the CCDF of the goodput for carousels of different file sizes with 5% and 50% channel losses, using 10% and 50% of AL-FEC parity in each case. The probability that the goodput is higher than a given value is higher with object multiplexing, due to the object multiplexing gain. This means that object multiplexing allows to either send more files in the carousel, provided the same lower bound on the goodput, or to send the same carousel at a higher goodput. For instance, regarding 5% packet losses with no object multiplexing, a carousel of 25 files provides a minimum goodput of 50kbps for 90% of the cases. With object multiplexing, the same carousel size provides a minimum goodput of 60kbps. On the other hand, instead of sending 50 file carousels at a higher goodput, it is possible to send carousels of up to 70 files with the same requirement on the minimum goodput.

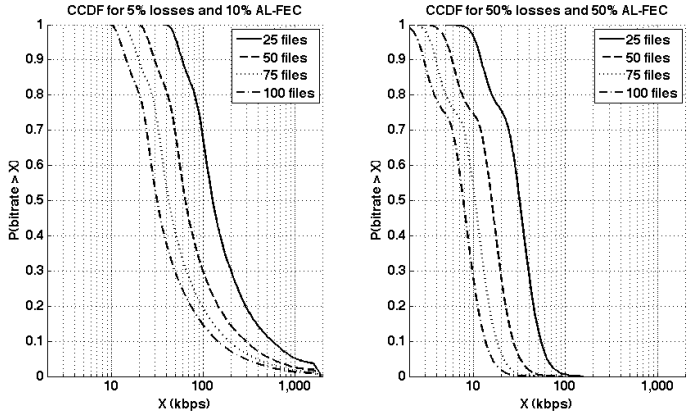

Figure 75. CCDF of the goodput for different carousel sizes with object multiplexing.

Figure 76 shows the mean download time of file carousels using AL-FEC and object multiplexing in a channel with 5% packet loss rate.

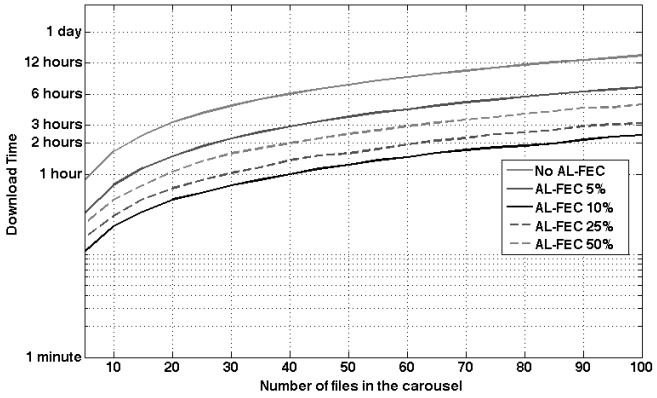

Figure 76. Mean download time on a channel with 5% packet losses with object multiplexing.

By comparing the results of Figure 76 and Figure 71, it can be noted that applying object multiplexing reduces the download time for all the configurations of AL-FEC analyzed. As in the previous case study, the lowest download times are provided with an AL-FEC parity of 10% and the worst results are obtained when no AL-FEC parity is added to the carousel.

Recall that the average download time for 100 file carousels with no object multiplexing is around 1 day. The average time is reduced down to 14 hours applying object multiplexing with no AL-FEC. With 5% AL-FEC parity, the download time is slightly higher than 6 hours and with 10% AL-FEC parity, the download time is around 2 hours.

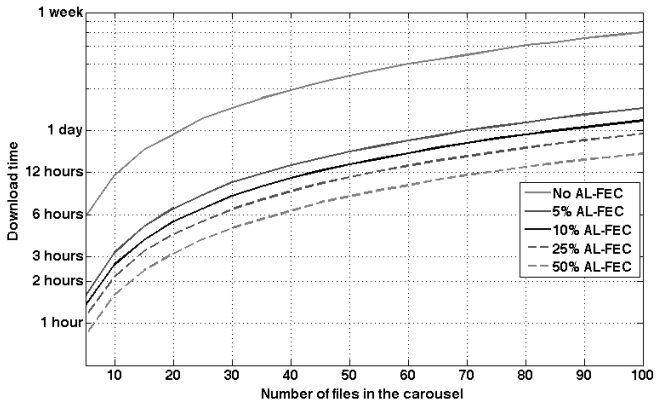

Figure 77. Mean download time on a channel with 50% packet losses with object multiplexing.

Figure 77 shows the mean download times for carousels with object multiplexing on channels with 50% channel losses and different configurations of the AL-FEC rate. In this case, the results also show lower mean download times than those obtained without object multiplexing (Figure 72) for all the configurations of AL-FEC parity under study. The minimum download time with this channel loss rate is accomplished with an AL-FEC parity of 50%. The rest of AL-FEC configurations provides similar download times, improving to a great extent the download time achieved with object multiplexing alone.

In order to evaluate the actual gain of AL-FEC in carousels with Object Multiplexing, Figure 78 presents the reduction of the download time achieved with the different configurations of AL-FEC under study.

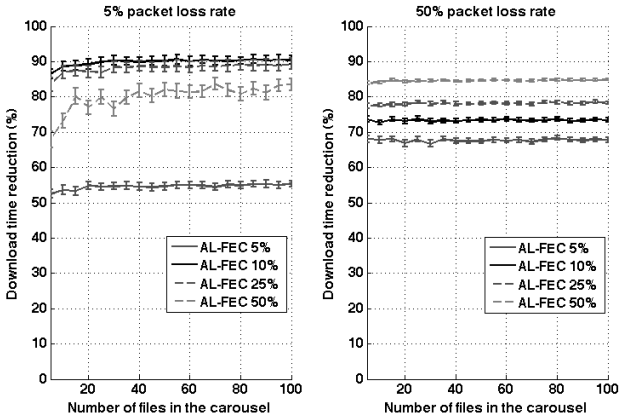

Figure 78. Reduction of the download time for different AL-FEC configurations with object multiplexing.

Comparing the results of AL-FEC with and without object multiplexing (Figure 73), the improvement of combining both techniques is noticeable with 5% packet losses. The improvement is about 5% higher for the different AL-FEC parities, except for the 25% AL-FEC parity, which exhibits an improvement of about 10%. However, it can be noted that reduction of the download time with 50% packet losses is the same with and without Object Multiplexing. Recall that the reduction of the download time is relative to the download time achieved with object multiplexing but no AL-FEC parity in the carousel. Therefore, the aforementioned 5% improvement is due to the combination of object multiplexing and AL-FEC parity, rather than just to the effect of AL-FEC encoding.

V.3.3 AL-FEC with Variable ODI bitrate

The previous study case showed the mean download time with object multiplexing and AL-FEC over a channel with constant capacity. In this study, we are going to analyze the statistics of the effective bitrate over a background channel (presented in section III.5.2). Later, the results are compared to the statistics of the effective bitrate over a constant capacity channel. In the simulations, the bitrate available for opportunistic insertion described in the introduction (Figure 68) is circularly shifted, so that the carousels start at a random point of the time line of the available bitrate. Then, the shifted bitrate trace is repeated a number of times in order to complete the simulation time. Therefore, the

available bitrate consists of a periodic trace with a period of approximately 25 minutes. Otherwise, the simulations use the same methods as previous studies.

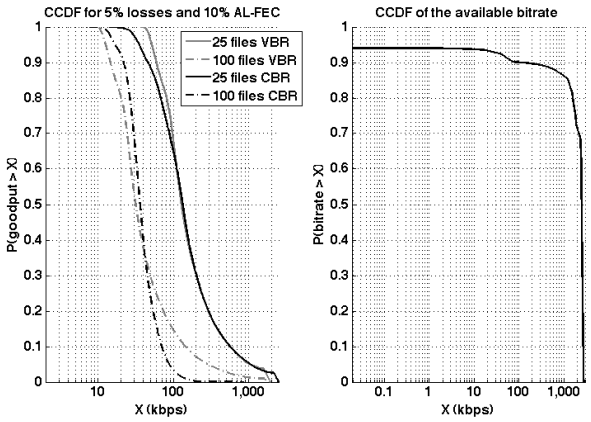

Figure 79. a) CCDF of the goodput and b) CCDF of the available bitrate for different carousel sizes.

Figure 79 a) shows the CCDF of the goodput with variable bitrate (VBR), against the goodput with constant bitrate (CBR) for two different carousel sizes (25 and 100 files). The simulations apply 5% packet losses and 10% AL-FEC parity. The figure shows how the variable bitrate changes the statistics of the goodput: the CCDF of the goodput with VBR changes with respect to the CCDF with CBR for both carousel sizes. These differences are due to the statistics of the variable bitrate, depicted in Figure 79 b). There are complex video scenes that require high video bitrates, leaving very little bandwidth available for the background services. On the other hand, the probability that the bitrate is higher than the average (2.1 Mbps) is around 71%. This is because, most of the time, the encoders are able to reduce the bitrate significantly, leaving a lot of bandwidth available for the background service.

For this reason, the CCDF of the goodput with VBR improves for short carousels (25 files) and low goodputs: the probability that the goodput is higher than a given value increases with VBR, compared to CBR. For instance, with 25 file carousels the probability that the goodput is higher than 50kbps is 0.96 with VBR and 0.88 with CBR. This shows that, for

short carousels, it is likely that there are no complex scenes during the transmission of the carousel, thus increasing the probability of having a high goodput. On the other hand, for long carousels (100 files), the CCDF of the low goodputs is worst with VBR than with CBR. Longer carousels take more time to be transmitted, increasing the probability that several video bitrate peaks occur during the download of a file. For instance, with 100 file carousels the probability that the bitrate is higher than 20kbps is 0.80 with VBR and 0.93 with CBR. On the other hand, the CCDF improves for high goodputs: the probability that the bitrate is higher than 100kbps is 0.15 with VBR and 0.03 with CBR.

Therefore, the use of opportunistic insertion changes the statistics of the goodput. The actual changes depend on the carousel duration and the video scenes.

V.4 Conclusions

This chapter has presented object multiplexing and AL-FEC: Two technologies to reduce the average access time to files in carousel transmissions. First, we have presented a theoretical analysis of the access time with object multiplexing and AL-FEC. This analysis provides a lower bound for the average access time. The evaluation of the lower bound showed that the potential gain (reduction in the access time) is really high, proving that the access time can be reduced almost tenfold for the different study cases.

The section has also presented two different algorithms to implement object multiplexing, the Modified Virtual Clock (MVC) algorithm and the Modified Weighted Fair Queuing algorithm (MWFQ). The MVC algorithm has been proposed in a previous work and hereby adapted to work with channel losses and AL-FEC. On the other hand, the MWFQ algorithm has been proposed in this thesis. The results show that the MVC cannot adjust the bitrate of each file adequately when the file sizes in the carousel vary in a wide range. On the other hand, the MWFQ achieves accurate long-term bitrates regardless of the file sizes in the carousel.

Regarding AL-FEC, the results have shown that it achieves a great reduction of the access time in the presence of channel losses. Applying AL-FEC is a must for any CDS service

dealing with packet losses. We have evaluated the access time in two different packet loss scenarios, with 5% and 50% packet channel losses. In both cases, AL-FEC reduces the access time to almost 90% of the access time without AL-FEC. Regarding the optimum AL-FEC encoding rate, the results show that there is an optimum encoding rate that provides the best access time, depending on the channel losses. However, the results also show that adding additional AL-FEC parity above the optimum value does not degrade the access time considerably. For this reason, it is recommended to apply additional AL-FEC parity, above the optimal value for the expected packet channel loss ratio.

The results also show the CCDF of the goodput with object multiplexing AL-FEC and channel losses. The CCDF of the goodput is useful to plan background CDS, because it allows establishing a minimum goodput for a percentage of the file downloads in the service area. The results compare the goodput achieved in a constant channel and in a background channel, showing that the effect of opportunistic insertion depends on the carousel size.

Chapter VI

Popularity, Storage Management, Personalization and QoE

The final section of this thesis regards the management of storage capacity at the client side, the service personalization and the QoE. These three aspects of the service are tightly related to the content popularity. First, recall that the server performs an estimation of the content popularity that is used by the file scheduler to adjust the long-term bitrate of every item in the carousel. Therefore, it is easier for client applications to fetch the most popular files from the carousel than to fetch the less popular items. Second, although all client applications receive the same broadcast carousel, the storage management policy of a particular client keeps in storage only the files that better fit the preferences of the user (personalization). Hence, the user experience is related not only to the estimation of the popularity made by the server, but also to the ability of the recommender to estimate the probability of access of each content item. In this sense, Section VI.1 provides a brief overview of the means available for service providers and client applications to estimate the popularity. Third, the performance of the storage management policy also depends on the relationship between the file sizes, the carousel scheduling, the carousel size and the available storage capacity.

These aspects of background content download services are analyzed in the following subsections. Section VI.2 regards storage management for stand-alone background push

CDSs, that is, the performance of the storage management policy is evaluated only regarding the quality metrics of a push content download service. On the other hand, VI.3 evaluates the performance of the background push CDS working as a local cache for a primary VoD service.

VI.1 Estimation of the popularity of television content

VI.1.1 Audience measurements

Originally, television was a highly time-dependent, hardly accessible media. There were no means to store programs to watch them at a later time, broadcasting was expensive and so were television terminals. Therefore, viewers had limited and fleeting opportunities to watch the content they like and huge masses gathered to watch the few television programs available. These facts motivated the development of strong relationships between television as a service and the influence it had on society, known as social aspects of TV [59]. Given the many ramifications of this area and the topic of the chapter, this section only focuses on social aspects accounted for in the estimation of the popularity of television content and its application to the management of television content delivery systems.

The most representative aspect is the relationship between the popularity of television content and the content scheduling. Recalling that there were no other means to access content, popular programs gathered many simultaneous spectators and broadcasters used this gathering to shape their program grids, placing popular shows in the day parts with more potential viewers and giving birth to the concept of prime time. Up to the date, popularity is the most important metric for television service operators, since the viewership of programs has a direct impact on their incomes.

The way that popularity is measured depends on the topology of the service. Traditional television broadcast platforms lack of a feedback channel from the user back to the television service provider. For this reason, it was necessary to develop audience estimation systems based on other mechanisms, like telephone surveys or by installing audience measurement equipment on a representative sample of the population. Nowadays, audience

measurement agencies still use these techniques to estimate the popularity of television content. The advertisement rates are calculated according to these measurements.

Contrarily, IP based TV content delivery technologies provide technical means to measure the audience directly. For instance, TV on Demand portals run on Content Management Systems (CMS) able to log the requests that are issued to each program. Media players can trigger events related to the play-out of the video and inform the server of when the video is paused or what is the percentage of video that has actually been consumed. Again, advertisement revenues are calculated based on the audience measurements. It is worth noting that the costs of streaming are proportional to the workload of the servers and in extension to the actual content consumption. Therefore, there is a direct relationship between investments and revenues that does not exist in broadcasting.

However, regardless of the content delivery technology, broadcasters use audience tracking for positioning content: television operators must ensure that the access to most popular contents is easier. Hence, most popular shows are scheduled on privileged time slots in linear TV program grids and are reserved the most visible areas of TV on Demand portals. In summary, popularity is an important aspect of their revenue streams and they invest heavily to obtain good estimations.

Regarding unidirectional background push CDSs, it is clear that the estimation of the popularity needed by the file scheduler is already used in the management of the other primary services (e.g. broadcasting and VoD). Furthermore, these services make the same usage of the estimation of the popularity: To favor the delivery of popular content.

VI.1.2 Audience segmentation and content personalization

Beside the audience ratings, audience measurement regards demographic aspects (e.g., gender, age, geographical proximity) and social aspects (e.g. level of education, cultural proximity) of the audience, because these are also important business metrics for broadcasters, since they are helpful to better focus content to specific target groups. This practice, known as audience segmentation, improves the effectiveness of advertisement

campaigns (including campaigns for related programs), increases user satisfaction and, in turn, helps at gaining audience in general terms.

In traditional broadcast TV services, audience segmentation can only be achieved by creating thematic channels or by configuring the program grid, for instance, to favor specific age groups (like showing cartoons in the morning). Then again, IP based TV services offer more means to better implement audience segmentation. Since users do not necessarily share the same interface with the service, it is possible to adapt it according to the information about the user that is available to the service operator. There are different alternatives to obtain demographic information about a user, for instance, explicitly when the user subscribes to the service. In this sense, social networks [59] represent an outstanding source of information for TV service providers. Therein, users do not only provide explicit demographic or social information, but in addition, there is a lot of implicit information, including information about social links between users. All this information can be used to narrow down the service segmentation to more homogeneous and smaller groups to which the content offer is adapted for.

Ultimately, when the segmentation strategy considers information about the users as individuals, service segmentation becomes service personalization. Personalization is nowadays a common feature in most popular video on demand web sites, which implement recommender systems [5] to highlight content of interest for the user (as well as advertisements of interesting products).

Similarly, the unidirectional background push CDS service presented in this thesis uses the information of a recommender system for personalization: The recommender determines the future probability of access of every content item. This information is used by the storage management policy in order to decide which files must be kept in cache. Next section provides a brief overview of recommender systems for background push CDSs.

VI.1.3 Recommender Systems for background push CDS

The role of the recommender in the architecture of the client application has been discussed at several sections since the beginning of the thesis. In the service architecture description

(Figure 2), the recommender receives information from the user interface to build a user profile. Later, the user profile is compared to the content description of a content item in order to obtain an estimation of the probability of access to that particular content item. As explained in Section II.3.1, the storage management uses this information to decide which files should be stored in cache. This decision is made every time that the storage management policy receives feedback from the recommender. This section provides an overview of how this estimation is calculated.

The architecture in Figure 2 implicitly describes a content filtering recommender [5]. Content filtering recommenders work with metadata descriptions, according to a specific data model. Common metadata systems like the PSI/SI [46] used in DVB services or the standard TVAnytime [60] provide data models that can be used to build very rich content descriptions. The metadata description can contain any kind of information, such as the genre of the content, its length or its language. DVB television systems include metadata descriptions, compliant with the PSI/SI specifications, which are used to build Electronic Program Guides (EPGs).

The basic principle of a content filtering recommender is to compare the content description of an item with the user profile: A metadata description of the preferences of the user. There are different ways to learn the preferences of the user in order to obtain the user profile. They can be classified into two basic categories, explicit methods and implicit methods. Basically, explicit methods ask users for their preferences, while implicit methods learn the user preferences from the historical usage of the service. This way, the user profile is updated every time that the user interacts with a content item, for instance, by adding the content description of the content item watched to the user profile.

In order to compare the user profile with the metadata, it is necessary to obtain an Information Retrieval model of the corresponding data. Basically, an Information Retrieval model is a mathematical model of the metadata, which provides the basis to perform the comparison. The most widely used Information Retrieval models in recommender systems

are the vector space model, used by heuristic recommenders, and the standard Boolean model, used by Bayesian recommenders.

The vector space model is based on semantic spaces. A semantic space is an algebraic representation of the content description as a vector in which each dimension is a different characteristic of the content item. This way, different properties of the metadata file are regarded as independent magnitudes (i.e. orthogonal dimensions) in which a content item can be represented. For instance, the genre or the language of a content are regarded as different dimensions. Thus, content descriptions are regarded as vectors representing the content in a particular semantic space. The similarity between two vectors is computed as the cosine function between the vector representing the content item and the vector representing the user profile.

As an example, let $d_j = (w_{1,j}, w_{2,j}, \ldots w_{t,j})$ be the vector in t dimensions representing the metadata description of content item j and $w_{n,k}$ be the weight of the description in dimension n. Similarly, let $d_u = (w_{1,u}, w_{2,u}, \ldots w_{t,u})$ be the vector representing the user profile of user u. The probability of access of the content j for the user u is computed as the cosine of the angle between the two vectors $\Phi_{j,u}$:

$$p_j = \cos(\phi_{j,u}) = \frac{d_j \cdot d_u}{|d_j||d_u|} \tag{52}$$

Mainly, the complexity of a heuristic recommender depends on how the vectors are computed from the metadata description and on how the user profile is built. Both aspects are tightly related to the data model used to build the content descriptions. For instance, some fields of the PSI/SI EPG specification, like the Short Program Description, support free-text input (a plain text value of any length) descriptions of the programs, like a brief synopsis of the content or information about the main cast. Obtaining a vector model from free-text entries requires some sort of natural language processing called Latent Semantic Analysis. On the other hand, the value of other EPG data fields, like the content descriptor, only accepts a coded value in a predefined range.

The content descriptor is present in the Event Information Table (EIT), which is the name given to the metadata that describes a program in the DVB SI specifications. The value of the Content Descriptor group is an array of 4-bit value pairs called *content nibbles*, which are two-level genre identifiers. The nibble of level 1 encodes the genre category of the content and can take a value from 0x1 (*movie drama*) to 0xF (*user defined*). The nibble of level 2 classifies the genre of the content within a content category. The encoded value of the content nibble 2 is related to the level of generality of the genre definition. Regardless of the category, an encoded value of 0x0 means *general*. For instance, if the content nibble 1 is 0x2 – *news/current affairs*, a content nibble 2 of 0x0 means *general news/current affairs*, a value of 0x2 means *news/weather report*, a value of 0x3 *documentary* and a level of 0x4 means *discussion/interview/debate*. These coded values can be mapped directly into a semantic vector model. As an example, Figure 80 illustrates a possible mapping of the value of the Content Description into a vector model.

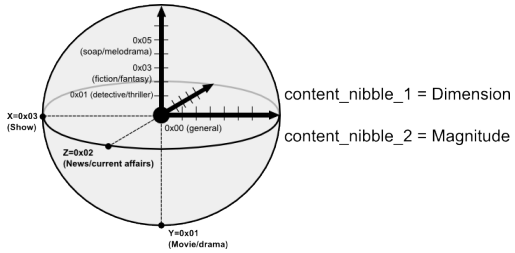

Figure 80. Mapping of Content Descriptor into a semantic vector model.

The content nibble values can be mapped into a semantic space of 15 dimensions, one per every admitted value of content nibble 1, as shown in Figure 80. In each dimension, the magnitude represents the level of generality of the genre, provided by the encoded value of the content nibble 2.

This way, the array of content nibble value pairs encoded in a content descriptor can be mapped into a vector representation of the metadata. The vector representing the user profile can be constructed from the historical usage of the service, for instance as the mean between the vectors of the programs watched, or from explicit feedback from the user.

Later, it is possible to compute the cosine of the angle between the two vectors to compute the probability of access to the content.

On the other hand, the standard Boolean model is based in Boolean logic and regards content descriptions as finite sets of index terms, that is, $d_j = \{k_{1,j}, k_{2,j}, \ldots k_{s,j}\}$, where $k_{o,j}$ is the keyword o in the content description j. This mathematical model provides the basis for Bayesian recommenders, or naïve Bayes classifiers. Bayesian recommenders use the Bayes principle to calculate the probability that the content item belongs to category C_i, $P(C_i|d_j)$, as:

$$P(C_i \mid d_j) \propto P(C_i) \prod_l P(k_{l,j} \mid C_i) \tag{53}$$

where $P(C_i)$ is the probability that a content item belongs to the category C_i, while $P(k_{l,j}|C_i)$ is the probability that the keyword $k_{l,j}$ occurs in the category i.

Note that Bayesian recommenders obtain a value proportional to the probability that a content item belongs to a category. Hence, in order to use a Bayesian recommender to obtain a probability of access, it is necessary to define a meaningful set of categories in the range of probability access [0,1]. For instance, a possible set of categories is:

$$C_1 = \{p_j \in [0,0.5[\}, \; C_2 = \{p_j \in [0.5,1]\} \tag{54}$$

where C_1 can be regarded as the category with low probability of access C_2 can be regarded as the category with a high probability of access. Thus, a Bayesian recommender provides the category with the highest $P(C_i|d_j)$ for a content item. Later, the storage management policy needs to assign a numerical value to the probability of access of each category, in order to apply the heuristic of the knapsack problem. For instance, in the example above, the storage management can assign a value of 0.25 to C_1 and 0.75 to C_2.

Despite their simplicity, Bayesian recommenders are widely used in recommender systems because they are simple to implement. They require little training data to estimate the probabilities, compared to other approaches. In this sense, the complexity of the

implementation of a Bayesian recommender mainly depends on how the probabilities in Eq. 54 are computed from the user interaction. Just as with heuristic recommenders, the application can use specific feedback, asking the user to place the content into a category after it is consumed. Otherwise, the application can infer the category by observing the user behavior (zapping patterns, time spent watching a particular show, etc.).

Regarding unidirectional background push CDSs, the most important requirements for the recommender are low complexity and lack of explicit feedback. It is important to keep the recommender as simple as possible in order to avoid any negative effect on the performance of the primary services. On the other hand, although the inference of preferences from user patterns is in general more complex than requesting explicit feedback, it does not seem appropriate to bother the user asking for feedback from a background service, so it is necessary to incorporate some implicit inference into the recommender. In this sense, it is interesting to look into recommender applications that can infer user patterns from the usage of the primary streaming services [60]. In this reference, the authors translate user actions into an Integer value, negative actions having a negative sign and positive actions having a positive sign. For instance, zapping away from a program is translated into -2 and watching the entire program is translated into +2. Then, the user profile is updated by adding up the numerical values corresponding to the user interaction.

VI.2 Storage management for background push CDSs

VI.2.1 Evaluation of the loading time

The loading time is the time needed by the storage management policy to fill the cache with the files that maximize the value of the cache, according to the heuristic of the knapsack problem presented in section II.3.1. The simulations use the model for the available bitrate in the DVB multiplex C28 presented in section III.3.3. The evaluation regards three different parameters: The storage utilization, the overall probability of access and the loading time. The storage utilization is the average of the percentage of storage capacity occupied by the files in cache. On the other hand, the overall probability of access is the

sum of the probability of access of the files stored in cache. Thirdly, the loading time is the time needed to fill the cache.

From a broader perspective, the storage utilization is proportional to the overall play-out time of the files in local storage. For instance, with a constant encoding rate of 5 Mbps, 1GB storage is approximately 30 minutes of play-out time. Similarly, the total probability is related to the contents that are not downloaded into a particular client and therefore to the level of personalization introduced in the service. On the other hand, the loading time is related to the time needed to push the files into local storage. An optimum end-to-end system configuration should regard the number of files in the carousel and the client storage size, in order to achieve an optimum storage utilization and level of personalization after a target loading time.

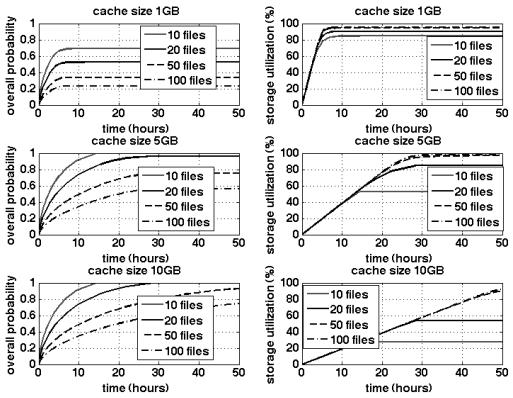

Figure 81. Probability of access to files in storage and used storage against time for IMDB content.

With this in mind, the results in Figure 81 show the performance of the storage management policy against time for three different storage capacities (1 GB, 5 GB and 10 GB). The carousel scheduler applies object multiplexing with the MWFQ algorithm presented in section V.2.3. The figure shows results for carousels with 10, 20, 50 and 100 files. The file sizes have been generated with the file size model for television content

presented in section II.1.1, applying an encoding rate of 5Mbps. We have generated 1000 different carousels for each configuration under study.

The straight lines in the storage utilization graphs show the average speed at which the background push service fills the storage capacity. It can be noted that this speed is rather constant regardless of the number of files in the carousel, for each of the cache sizes under study. At a given point, the storage capacity usage stops increasing, either because the storage capacity is full (it approaches 100%) or because all the files in the carousel are completely downloaded (the probability of access reaches 1). For instance, with a cache size of 5GB, the storage capacity downloads all the files of the 10-file carousel after an average time of 15 hours, using approximately 60% (3GB) of the storage capacity. With the same storage capacity, 20-file carousels reach a maximum average storage usage of 87% after approximately 28 hours. Similarly, 50-file carousels and 100-file carousels reach 100% average storage usage after the same average time of 28 hours. Hence, as expected, larger carousels require longer loading times, but achieve higher average storage usage percentages. Another interesting result is that, since the files are rather large, the 100% storage capacity usage is only reached for carousels with large number of files. This is particularly noticeable in the 1GB study case.

The overall probability of access in the left-hand graphs also increases monotonically with time, although the effect of the carousel size is not the same as with the average storage usage. In this case, larger carousels require longer loading times to achieve the maximum overall probability of access, but shorter carousels achieve higher average overall probabilities. Following the example above, with 5GB storage, 10-file carousels achieve an overall probability of access of 1 after approximately 15 hours. 20-file carousels achieve a probability of access of 0.95 after approximately 25 hours. 50-file carousels and 100-file carousels achieve a maximum probability of access around 0.75 and 0.55 after approximately 28 hours. This is because with shorter carousels, it is easier to download most of the files of the carousel to local storage, thus achieving a higher overall probability of access.

Let us analyze the results from the point of view of end-to-end service management, focusing on the relationship between the carousel size and the local storage capacity. Given the encoding rate of 5Mbps, the video play-out time corresponding to each case is approximately, 30 minutes for 1GB, 2.5 hours for 5GB and 5 hours for 10GB. This play-out time is achieved after a loading time of respectively 13 hours, 26 hours and 50 hours, provided that there are enough files in the carousel. Focusing on the cache size of 5GB, a background push CDS operator working with the background capacity left in C28 could provide around 2.5 hours of alternative content every 26 hours. Recall that this encoding bitrate is characteristic of HD content. The carousels should be sufficiently large (for instance 50 or 100 files) in order to leave some margin for personalization. Larger storage sizes would allow users to store larger number of files over time. These results clearly depict the potential of background services over terrestrial DVB networks.

Similarly, Figure 82 shows the average overall probability of access, the average storage usage and the loading time achieved for YouTube content. Again, the scheduler uses the MWFQ algorithm and the background capacity model emulates the background capacity found in multiplexer C28. The figure presents the results obtained after 1000 simulations with carousels of 20, 50, 100 and 200 files. In this study case, the cache sizes evaluated are 0.5GB, 1GB and 2GB.

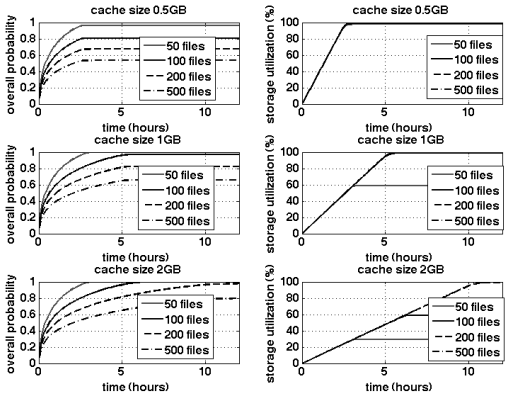

Figure 82. Probability of access to files in storage and used storage against time for YouTube content.

Comparing the results for 1GB storage in Figure 81 and Figure 82, it can be noted that the loading time is approximately the same (around 5 hours) for both kinds of content, provided that there are enough files in the carousel. Moreover, since the files are smaller, it is easier to achieve higher storage utilization. For instance, with 1GB of local storage, 100-file YouTube carousels achieve 99% storage utilization, while 100-file IMDB carousels achieve 95% storage utilization. Otherwise, the results in the two graphs are equivalent. As with IMDB content, the more files are added to the carousel the lower is the local probability of access and consequently, the greater is the margin for optimization.

As in the previous example, the results can be used to evaluate the potential of background push CDS dealing with YouTube content. Provided that the encoding rate used is 5Mbps, the cache sizes under evaluation are equivalent to approximately 15 minutes, 30 minutes and 1 hour of content play-out. Moreover, the cache loading times are approximately 3, 6 and 12 hours. With a local cache of 1GB, a content provider could push approximately 30 minutes of alternative Youtube content every 6 hours. The carousels should have more than 100 files, so that the storage management policy can introduce some personalization in the service.

VI.2.2 Object multiplexing and storage management

In this section we present the relationships between the storage management policy and the configuration of other system blocks: The file scheduler, the definition of value used by the knapsack algorithm to decide which files should be kept in local storage and the local estimation of the probability of access. Let us start with the evaluation of the effect of object multiplexing in the loading time. Figure 83 shows the performance of the storage replacement policy with two different carousel schedulers: object multiplexing with the MWFQ algorithm working with the optimal long-term bitrates and a sequential scheduler. The sequential scheduler sends the files in descending order of popularity ranking (most popular file first, less popular file last). It is worth noting that, although the sequential scheduler is simpler, it also requires an estimation of the popularity. The results show the overall probability of access and the storage usage achieved after 1000 simulations with 10-file and 100-file carousels, generated with the IMDB content model. The cache sizes under

analysis are 1GB, 5GB and 10GB. The encoding rate is 5Mbps as in the previous study case.

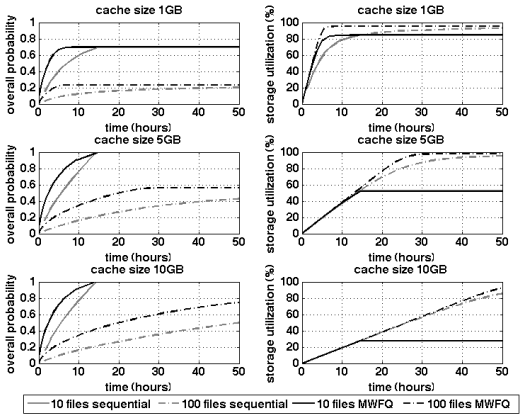

Figure 83. Loading time for carousels with and without object multiplexing.

Looking at the overall probability of access, it is clear that object multiplexing with the MWFQ algorithm improves the results obtained with sequential scheduling in the loading stage. To illustrate this, note that with a 10GB cache and 100-file carousels, after 10 hours, the mean overall popularity with sequential scheduling is 0.2 and the mean overall popularity with the MWFQ algorithm is 0.4. This improvement is due to the fact that the MWFQ algorithm fosters a more efficient use of the storage space. Consequently, clients achieve higher levels of popularity of access at an earlier time during the loading phase. However, after the loading time is completed, both scheduling policies reach the same overall probability of access.

The effect on the mean storage usage is similar, although less acute. During the loading phase, the MWFQ scheduler yields higher storage utilization percentages than the sequential scheduler. Although both reach the same level of occupancy after the loading time is completed, the loading time for the sequential scheduler is significantly larger.

On the other hand, section II.3.1 presented different broadcast cache storage management policies found in related literature, namely, the PIX policy, which considers the frequency

of access in the definition of value and the PIXS algorithm, which considers both the frequency of access and the size of files.

TABLE V
EVALUATION OF DIFFERENT CACHE REPLAMENT POLICIES

	P			**PIX**			**PIXS**		
Value	**P**			**P*T_c**			**P*T_c/s**		
Cache size (GB)	5	10	50	5	10	50	5	10	50
0.9 Load time (hours)	18.4	18.2	18.0	18.2	18.3	18.2	18.1	18.5	18.3
maximum probability	0.95	1.00	1.00	0.95	1.00	1.00	0.95	0.99	1.00
Storage usage (%)	87.2	57.8	11.8	86.6	57.7	11.7	87.0	58.2	11.8

We have repeated the simulations of the previous study case for the three policies. The parameters under evaluation are the 0.9 load time - which hereby is the time needed to achieve an overall probability of access equal to 0.9 - the maximum overall probability at the end of the simulation and the maximum mean storage usage. Table V provides a summary of the results obtained with 20-file carousels using IMDB content.

As shown in the table, all the different cache replacement policies exhibit very similiar performance. In general, it seems like the PIX and PIXS policies provide slightly better results than the P policy. However, the differences are very small in all cases. The 0.9 load time is around 18 hours and the differences between the different policies are in the order of tenths of minutes. Similarly, the maximum difference found in the maximum storage usage is around 1%. The reason for this is that the power law behavior of the popularity introduces great differences between the value of each file. On average, these differences are much larger than the differences in the file sizes. Additionally, the carousel cycle is inversely proportional to the popularity. Therefore, file sizes and carousel cycles are taken into consideration, the values still depend mainly on the popularity. Due to this, the three policies end up making very similar decisions and achieve similar performance.

It is important to bear in mind that the implementation of the PIX and PIXS policy is slightly more complex than the implementation of the P policy, because the clients need to know (or estimate) the carousel period of every file. The server can signal this information together with the metadata description of every file, but this implies that the server needs to know in advance when will be the next time that the item will be transmitted. Looking at Table V, the slight improvement in the results does not justify the added complexity associated to these policies.

Similarly, we have repeated the simulations with different estimations of the local popularity at the client. The objective is to assess how simple probability estimations affect the performance of the storage management. In this experiment, we repeat the configuration parameters above. Again, we assume that the actual preferences of the user are modeled by the same power law distribution as estimated by the server for the whole content area.

TABLE VI
EVALUATION OF DIFFERENT PROBABILITY ESTIMATION METHODS

	10-file, 1GB			**20-file, 5GB**			**50-file, 10GB**		
Method	I	5C	2C	I	5C	2C	I	5C	2C
Load time (hours)	3.6	3.7	3.7	18.3	18.3	18.4	49.3	49.4	50.7
Overall probability	0.70	0.70	0.68	0.96	0.95	0.97	0.75	0.75	0.75
Storage usage (%)	83.9	84.6	83.6	85.4	84.6	83.6	98.3	98.2	98.4

In order to emulate the effect of the recommender, we use three different methods to simulate virtual recommenders. The first one (I) provides the exact value of the local popularity of every file. This is the method used in previous studies. The other two methods emulate Bayesian recommenders, providing an estimation in a finite set of values. The first (virtual) Bayesian recommender (5C) provides a probability estimation in the set {0.1, 0.3, 0.5, 0.7, 0.9}, that is, it uses five different categories. The second Bayesian recommender, (2C) provides a value in the set {0.25,0.75} (two different categories).

Similar to the previous study case, the different methods used to estimate the probability provide very similar results. This is due to the fact that popular items are a lot more popular than less popular items. Since the popularity decreases very rapidly, the storage management policies end up taking the same decisions with the three popularity estimation methods of the popularity. It is particularly relevant that a Bayesian recommender with only two categories provides only slightly worse results than an ideal estimation of the popularity. This indicates that the storage management does not need an accurate estimation of the popularity from the recommender, provided that the probability follows a power-law distribution.

VI.3 Background push CDS as a VoD local cache

In this section we are going to evaluate the performance of the background push CDS working as a prefetching cache for a VoD service. The metric used for the evaluation is the cache hit ratio, which is the ratio of programs that are directly served from the local cache, instead of being served from the remote VoD server. Therefore, the cache hit ratio is related to the bandwidth savings in the VoD server. Moreover, since the download process occurs in the background, the users do not experience the cache loading time. Instead, the latency of access experienced by users is reduced when the files are present in the local cache. Hence, the cache hit ratio is also related to the QoE of the service. This section gathers results from two different studies, without losses (section VI.3.1) and with losses (section VI.3.2).

VI.3.1 Cache hit ratio without channel losses

In order to evaluate the cache hit ratio, for the first study in this section we generate carousels using the IMDB and Youtube file size models and applying an encoding rate of 5Mbps. The scheduler applies object multiplexing with the MWFQ algorithm. There are three different combinations of carousel size and local storage size under evaluation: 10-file carousels working with 1GB caches, 50-file carousels working with 5GB caches and 100-file carousels working with 10GB carousels. The total number of requests, λ is set to 5000 requests per day. The requests issued to every file j are generated using a Poisson distribution where the number of requests per second, λ_j is calculated as $\lambda \cdot p_j$. This model is

described in detail in section II.3.1. The cache hit ratio is calculated as the ratio between the requests to files already stored in cache to the total number of requests issued. Figure 84 shows the cache hit ratio against time for the different configurations under study.

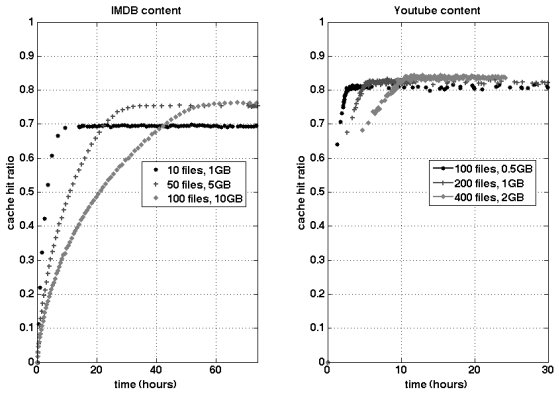

Figure 84. Cache Hit ratio against time for different carousel configurations and cache sizes.

Figure 84 highlights the most important features of background push CDS working as local caches. First, the main drawback of the proposal is the long loading times required to provide high cache hit ratios. For instance, for IMDB content, the loading times are about 15 hours (10 file carousels and 1GB storage), 30 hours (50 file carousels and 5GB storage) and 60 hours (100 file carousels and 10GB storage). Once the loading time is completed, the cache hit ratio is rather high (0.69, 0.75 and 0.76 in the three cases under study). It is clear that the proposal is only valid for VoD servers dealing with rather stationary catalogues that are not frequently updated and which probability can be easily estimated. Although these assumptions do not hold for User Generated Content, the loading times can easily fit the requirements of television program grids.

VI.3.2 Cache hit ratio with channel losses

In this study case, we are going to evaluate the effect of the loading time in two reception conditions, characterized by packet loss ratios of 5% and 50%. We apply AL-FEC encoding at two different encoding rates, adding 5% of AL-FEC parity and 50% AL-FEC

parity. The results are compared to the case when no parity is added to the carousel. In order to generate the carousels, we use the Youtube content duration model, and apply an encoding rate of 5Mbps. The background capacity is modeled with the empirical model obtained for C28. On the other hand, the carousel size is set to 400 files and the storage capacity is set to 2GB.

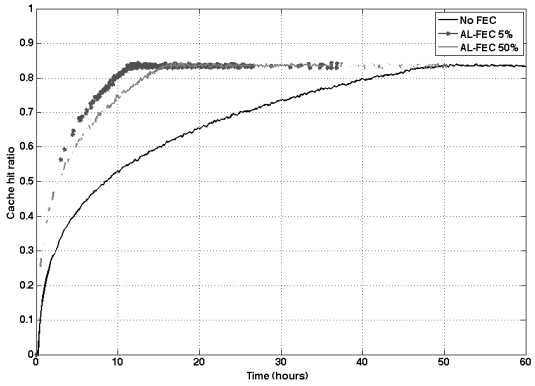

Figure 85. Cache hit ratio for 2GB caches for Youtube content with 400-file carousels with different AL-FEC parity and 5% channel losses.

Figure 85 shows the cache hit ratio against time with channel packet losses of 5%. The channel losses make it difficult for the clients to download the files and consequently, the loading time increases. Due to this, the loading time without AL-FEC parity increases to up to 50 hours. On the other hand, with AL-FEC parity, the increase in the loading time is not so drastic: the addition of 10% AL-FEC parity yields to a loading time of about 12 hours, while the addition of 50% AL-FEC parity results in a loading time of about 15 hours. However, once the loading time is completed, the three configurations exhibit the same level of cache hit ratios.

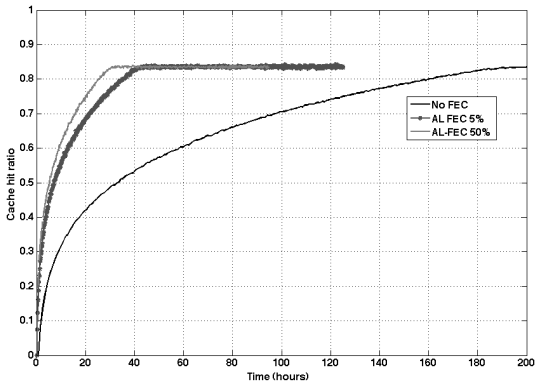

Figure 86. Cache hit ratio for 2GB caches for Youtube content with 400-file carousels with different AL-FEC parity with 50% channel losses.

Figure 86 shows the cache hit ratio with 50% channel losses. It can be noted that the same effect happens here, although as expected, the loading time increases to a greater extent (note the different scale in this figure). In this case, the loading time without AL-FEC parity is almost 200 hours. On the other hand, the addition of 10% AL-FEC parity produces a loading time of about 40 hours and the addition of 50% AL-FEC parity a loading time of approximately 30 hours.

From these results, it is clear that AL-FEC parity improves to a great extent the loading times of caches in channels with losses. Nevertheless, the loading times achieved are still longer than in channels without losses. In relation to previous results, the improvement achieved for a given packet loss rate depends on the amount of AL-FEC parity added to the carousel. The best results are provided by the same AL-FEC parities that minimize the download time of a file: 10% AL-FEC parity in channels with 5% packet losses and 50% in channels with 50% packet losses. In this sense, the differences between two different configurations are not as large as the differences with respect to carousels without AL-FEC parity. Therefore, it is always convenient to apply AL-FEC encoding to some extent, even though the exact packet loss rate is not known a priori.

In summary, the same conclusions than for the case without channel losses apply here: The proposal is only valid for rather stationary content catalogues. The content cannot be updated frequently or otherwise, the local cache does not reach significant values. In any case, the use of AL-FEC parity becomes necessary in channels with packet losses, in order to keep the loading time as short as possible.

VI.4 Conclusions

In this final chapter of the thesis, we have analyzed the features of background CDS related to storage management, personalization and QoE. First, we have introduced the system aspects related to the estimation of the popularity of television content. The estimation of the content popularity is a key aspect of object multiplexing. As explained, it is also a fundamental metric in the business model of television content providers and therefore, content providers use different techniques to accurately track content popularity. On the other hand, the storage management policies need an estimation of the probability of access of the different content items. This chapter describes how recommender systems can be adapted to provide such estimations.

Regarding the QoE, the results evaluate the cache hit ratio, i.e. the proportion of file requests that are served from a local cache loaded with content by a background push CDS. Files served from local cache exhibit zero latency and therefore, the cache hit ratio is crucial for the user experience.

The results show that such caches can serve a considerable amount of VoD requests using little storage capacity (few gigabytes) in the client. We have presented results for meaningful carousel sizes and local storage capacities, with and without channel losses and AL-FEC. In the different study cases, the cache hit ratio increases with time until it reaches a maximum value. In the results, this loading time is in the order of hours. This is not so critical for pre-produced television content, because it is normally available for delivery long time before it is programmed for transmission. Once the loading time is completed, the cache hit ratios achieved are rather high. For instance, for Youtube carousels of 400

files and 2GBytes of local storage capacity, the cache hit ratio is 0.82. On the other hand, with 100 IMDB files and 5 GBytes local storage, the cache hit ratio is 0.78.

Regarding object multiplexing, the results show that they improve the probability of access to files during the loading time, but it does not affect the maximum cache hit ratio. On the other hand, regarding the client application, the results show that the most relevant parameter to assess the value of files in cache is the future probability of access. Moreover, simple estimators based on Bayesian recommenders achieve equivalent results than using exact probability estimations in the storage management application. Apparently, taking into account the size of files or their broadcast frequency does not improve the cache hit ratio. However, knowing the scheduling beforehand could help at reducing the battery consumption in battery constraint devices, because the receiver could shutdown the background reception during the transmission of files that are not to be downloaded. This is important because battery consumption is one of the main drawbacks of background services for mobile terminals.

Channel losses do not affect the maximum cache hit ratio, but the loading time. In this sense, AL-FEC drastically reduces the cache loading times in the presence of packet losses, improving the usefulness of the service proposal. The results prove that background push CDS can be a valuable asset for content providers dealing with television content delivery for television sets and mobile terminals.

References

[1] CISCO: Cisco visual networking index: forecast and methodology, 2010-2014, http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html (2011).

[2] OOYALA: Ooyala video index report, Q1 2012, http://go.ooyala.com/rs/OOYALA/images/Ooyala-Global-Video-Index-Q1-2012.pdf (2012).

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

[4] 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, 2004.

[5] 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, 2005.

[6] S. Acharya, R. Alonso, M. Franklin and S. Zdonik, "Broadcast Disks: Data Management for Asymmetric Communication Environments," Mobile computing, The Kluwer International Series in Engineering and Computer Science, vol. 353, pp. 331-361, 1996.

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

[8] The Internet Movie Database (IMDB), http://www.imdb.com.

[9] J. Famaey, F. Iterbeke, T. Wauters, F. De Turck, "Towards a predictive cache replacement strategy for multimedia content," Journal of Network and Computer Applications, vol. 36, pp. 219-227, 2013.

[10] D. Meddour, A. Abdallah, T. Ahmed, R. Boutaba, "A cross layer architecture for multicast and unicast video transmission in mobile broadband networks," Journal of Network and Computer Applications, vol. 35, pp. 1377-1391, 2012.

[11] Y. He, Z. Xiong, Y. Zhang, Z. Tan, Z. Li, "Modeling and analysis of multi-channel P2P VoD systems," Journal of Network and Computer Applications, vol. 35, pp. 1568–1578, 2012.

[12] J. Broberg, R. Buyya, Z. Tari, "MetaCDN: Harnessing 'Storage Clouds' for high performance conten delivery," Journal of Network and Computer Applications, vol. 32, pp. 1012-1022, 2009.

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

[14] 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), Cincinatti, USA, Aug. 2001.

[15] P. Barford, M. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation," Performance Evaluation Review, vol. 26, pp. 151-160, 1998.

[16] M. Mitzenmacher, "Dynamic Models for File Sizes and Double Pareto Distributions," Internet Mathematics, vol. 3, pp. 305-333, 2004.

[17] W.J. Reed, M. Jorgensen "The Double Pareto-Lognormal Distribution – A New Parametric Model for Size Distributions," Communications in Statistics -Theory & Methods, vol. 33, no. 8, pp. 1733-1753, 2004.

[18] P. Gill, M. Arlitt, Z. Li, A. Mahanti, "Youtube characterization: A view from the edge," In Proc. of the 7th ACM SIGCOMM conference on Internet measurement, pp. 15-28, San Diego, USA, Oct. 2007.

[19] M. Zink, K. Suh, Y. Gu, J. Kurose, "Characteristics of Youtube network traffic at a campus network-Measurements, models and implications," Journal of Computer Networks vol. 53, pp 501-514, 2009.

[20] P. Ameigeiras, J. J. Ramos-Muñoz, J. Navarro-Ortiz, J.M. López-Soler, "Analysis and modelling of Youtube traffic," Transactions on Emerging Telecommunication Technologies, vol. 23, pp. 360-377, 2012.

[21] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," Science vol. 286, pp. 509–512, 1999.

[22] 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*, pp. 126-134, New York, USA, Mar. 1999.

[23] A.L. Chevernak, "Tertiary Storage: An Evaluation of New Applications," *University of California at Berkeley, Computer Science Division Technical Report UDB/CSD 94/847*, December, 1994.

[24] M. Cha, H. Kwak, P. Rodriguez, Y-Y. Ahn, S. Moon, "I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System," *In Proc. of the 7th ACM SIGCOMM conference on Internet measurement*, pp 1-14, San Diego, USA, Oct. 2007.

[25] S. Mossa, M. Barthélémy, H. E. Stanley, and L. A. N. Amaral, "Truncation of Power Law Behavior in Scale-Free Network Models due to Information Filtering," Physical Review Letters vol. 13, 2002.

[26] H. Yu, D Zheng, B. Zhao, W. Zheng, "Understanding user behavior in large-scale video-on-demand systems," ACM Operating Systems Review vol. 40, pp. 333-344, 2006.

[27] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "Analyzing the video popularity characteristics of large-scale user generated content systems," IEEE/ACM Transactions on Networking, vol. 17, pp. 1357-1370, 2009.

[28] S. Mossa, M. Barthélémy, H. E. Stanley, and L. A. N. Amaral, "Truncation of Power Law Behavior in Scale-Free Network Models due to Information Filtering," Physical Review Letters, vol. 13, 2002.

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

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

[31] J. Poikonen and j. Paavola, "Comparison of Finite-State Models for Simulating the DVB-H Link Layer Performance," in Proc. of the 2nd International Symposium on Wireless Communication Systems, pp. 153-157, Jun. 2005.

[32] 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, 2012.

[33] 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 Multimedia, vol. 14, no. 3, pp. 641-650, 2012.

[34] 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.

[35] ETSI EN 302 304 V1.1.1, "Transmission system for handheld terminals", Nov. 2004.

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

[37] 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.

[38] C. 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.

[39] D. Negrua, A. Mehaouaa, Y. Hadjadj-aoula, C. Berthelotb, "Dynamic bandwidth allocation for efficient support of concurrent digital TV and IP multicast services in DVB-T networks," Elsevier Computer Communications vol. 29, no. 6, pp. 741-756, Mar. 2006.

[40] M. Luby, M. Watson, T. Gasiba, T. Stockhammer, W. Xu, "Raptor codes for reliable download delivery in broadcast wireless systems", 3rd IEEE Consumer Communications and Networking Conference (CCNC), pp. 191-197, Las Vegas, USA, Jan. 2006.
[41] D. Barquero, A. Bria, "Forward Error Correction for File Delivery in DVB-H," in proc. of the Vehicular Technology Conference, pp. 2951-2955, Baltimore, USA, Oct. 2007.
[42] T. Lohmar, J. Huschke, "Radio resource optimization for MBMS file transmissions," in proc. of the 9th IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), pp. 1-7, Bilbao, Spain, May 2009.
[43] P. Lungaro, Z. Segall, J. Zander, "Predictive and Context-Aware Multimedia Content Delivery for Future Cellular Networks," in Proc. of the IEEE Vehicular Technology Conference (VTC), pp. 1-5, Taipei, Taiwan, May 2010.
[44] ETSI EN 300 744 V1.5.1, "Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television (DVB-T)," Jan. 2009.
[45] ISO/IEC 13818-1:1996, "– Information Technology – Generic coding of moving pictures and associated audio information: Systems," Jun. 1996.
[46] ETSI TR 101 290 v.1.1.1, "Measurement guidelines for DVB Systems," Jan. 2001.
[47] ETSI TR 101 190 v1.3.2, "Digital Video Broadcasting (DVB); Implementation guidelines for DVB Terrestrial services; Transmission aspects," May 2011.
[48] ETSI EN 300 468 v.1.13.1, "Specification for Service Information in DVB systems," Sep. 2012.
[49] ETSI EN 300 755 v1.3.1, "Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for a second generation digital terrestrial television broadcasting system (DVB-T2)," Apr. 2012.
[50] ETSI TS 102 472 v1.3.1, "Digital Video Broadcasting (DVB), IP Datacast over DVB-H: Content Delivery Protocols," Jun. 2009.
[51] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," IETF RFC 3550, Jul. 2003.
[52] ETSI EN 301 192 v1.4.2, "Specification for data broadcasting," Apr. 2008.
[53] F. Fraile, C.Nader, J.C. Guerri, N. Björsell, "On the reuse of DVB-T transmitter infrastructure for DVB-T2," in Proc. of the IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), pp: 1-6, Nuremberg, Germany, Jun. 2011.
[54] F. Fraile, I. de Fez, R. Belda, J.C. Guerri, "Evaluation of a background push services for personal multimedia devices," in Proc. of the IEEE International Congress on Consumer Electronics (ICCE), pp: 231-232. Las Vegas, USA, Jan. 2011.
[55] X264, a free software library and application for encoding video streams into the H.264/MPEG-4 AVC format, http://www.videolan.org/developers/x264.html
[56] ETSI TS 101 547 v1.1.1, "DVB Frame Compatible Plano Stereoscopic 3DTV (DVB-3DTV)," Jan. 2012.
[57] ETSI TS 102 905 v1.1.1, "Digital Video Broadcasting (DVB); Technical Specification for DVB Services in the Home Network Phase 1," May 2010.
[58] H. Zhang, "Service disciplines for Guaranteed Performance Service in packet switching networks," Proceedings of the IEEE, pp. 1374-1396, vol. 83, no. 10. Oct. 1995.
[59] F. Fraile, P. Arce, R. Belda, I. de Fez, J. C. Guerri, "Social Aware TV Content Delivery over intelligent networks," Social Media Retrieval, Computer Communications and Networks, Ed. Springer, pp. 373-391, 2013.
[60] ETSI TS 102 822-2 V1.4.1, "Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 2: Phase 1 - System description," Jul. 2007.
[61] M. Gude, S. Grünvogel, A. Pütz, "Predicting Future User Behavior in Interactive Live TV," in proc. of the 6th European Conference on Interactive Television (EuroITV), pp. 117-121, Salzburg, Austria. 2008 July 3-4, 2008.

Appendix A. List of publications

Following, there is a list of the publications that provide most of the content of this thesis dissertation. The following notation is used. B refers to book chapters, C to conference papers and J to journal papers.

Chapter II

[J.1] I. de Fez, F. Fraile, R. Belda, and J. C. Guerri, "Analysis and evaluation of adaptive LDPC AL-FEC codes for content download services," IEEE Trans. Multimedia, vol. 14, no. 3, pp. 641-650, Jun. 2012.

[J.2] 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, Springer Netherlands, vol. 60, no.3, pp. 669-688, Oct. 2012.

[C.1] I. de Fez, F. Fraile, R. Belda, and J. C. Guerri, "Evaluation of adaptive LDPC AL-FEC codes for content download services," presented at IEEE International Conference on Multimedia and Expo (ICME), Barcelona (Spain), Jul. 2011.

[J.3] F. Fraile and J.C. Guerri, "Simple models of the content duration and the popularity of television content," submitted to the Journal of Network and Computer Applications, Feb. 2013.

Chapter III

[J.4] F. Fraile and J.C. Guerri, "Evaluation of the bitrate available for opportunistic data insertion in DVB networks," submitted to the Journal of Wireless Communications, Jan. 2013.

Chapter IV

[C.2] F. Fraile, I. de Fez, R. Belda, and J. C. Guerri, "Evaluation of a background push content download service for personal multimedia devices," International Conference in Consumer Electronics, pp. 231-232, Las Vegas (USA), Jan. 2011.

Chapter V

[J.5] F.Fraile, I. de Fez, J.C. Guerri, "Evaluation of background push content download services to mobile devices over DVB networks," submitted to the IEEE Trans. Broadcasting, Dec. 2012.

[J.6] F. Fraile, J.C. Guerri and V. Pla, "Evaluation of Opportunistic Data Insertion in DVB multiplexers and its application to Content Download Service delivery," submitted to the Journal of Computer Communications, Jan. 2013.

Chapter VI

[J.7] A. Gil, F. Fraile, M. Ramos, I. de Fez, and J. C. Guerri, "Personalized Multimedia Touristic Services for Hybrid broadcast/broadband Mobile receivers," IEEE Transactions on Consumer Electronics, vol. 56(1), pp. 211-219, 2010.

[B.1] F. Fraile, P. Arce, R. Belda, I. de Fez, J. C. Guerri, and A. Pajares, "Social Aware TV Content Delivery over intelligent networks," Social Media Retrieval, Computer Communications and Networks, Ed. Springer, 2013, pp.373-391.

[C. 3] F. Fraile, I. de Fez, and J. C. Guerri, "Modela-TV: Service Personalization and Business Model Management for Mobile TV," in 7th European Interactive TV Conference (EuroITV), Leuven (Belgium), Jun. 2009.

[C. 4] A. Gil, F. Fraile, M. Ramos, I. de Fez, and J. C. Guerri, "Personalized Multimedia Touristic Services for hybrid broadcast/broadband mobile receivers," in Proc. International Conference in Consumer Electronics, Las Vegas (USA), Jan. 2010.

Additionally, the author has participated in the following publications related to the topic during the development of the thesis:

[C.5] F. Fraile, D. Kopparhead, H. Persson, and J. C. Guerri, "Interactive TV Arena: a TestBed for Hybrid Broadcast/Broadband Services," Broadband Multimedia Systems and Broadcasting (BMSB), Bilbao (Spain), May 2009.

[C.6] P. Arce, F. Fraile, M. Monfort, and J. C. Guerri, "Interactive TV-centric Health Care Services in Smart Homes," in EuroITV, Tampere (Finland), Jun. 2010.

[C.7] I. de Fez, F. Fraile, R. Belda and J. C. Guerri, "Implementación y evaluación de la codificación LDPC para la transmisión de ficheros en entornos unidireccionales," Jornadas de Ingeniería Telemática (JITEL 2010), Valladolid (Spain), Sep. 2010.

[C.8] F. Fraile, P. Guzmán, J. C. Guerri, T. R. Vargas, and N. Flórez, "Evaluation of an interactive TV service to reinforce dental hygiene education in children," in 9th European Interactive TV Conference (EuroITV), Lisboa (Portugal), Jun. 2011.

[C.9] F. Fraile, C.Nader, J.C. Guerri, N. Björsell, "On the reuse of DVB-T transmitter infrastructure for DVB-T2," Broadband Multimedia Systems and Broadcasting (BMSB), Nuremberg, Germany, Jun. 2011.

[C.10] T. R. Vargas, P. Arce, I. de Fez, V. Murcia, F. Fraile, R. Belda, P. Acelas, and J. C. Guerri, "Solutions to improve the video streaming service over heterogeneous networks," 1st Workshop on Future Internet: Efficiency in High-Speed Networks (W-FIERRO), Cartagena (Spain) , Jul. 2011.

[C.11] F. Fraile, P. Guzmán, T. R. Vargas, Y. N. Florez, and J. C. Guerri, "Evaluation of an interactive TV service to reinforce dental hygiene education in children," in 1er Congreso Internacional de Telecomunicaciones, Bucaramanga (Colombia), Sep. 2011.

[C.12] M. Ericsson, S.M. Hasibur Rahman and F. Fraile, "Efficient Interactive Multicast over DVB-T2 – Utilizing Dynamic SFNs and PARPS," Accepted for publication in Broadband Multimedia Systems and Broadcasting (BMSB), London (UK) Jul. 2013.

The author has participated in the following projects:

- Redes Híbridas para la provision de servicios turísticos "RRHH" TSI-020302-2010-165 / TSI-020302-2008-94.
- ImmersiveTV: una aproximación a los medios inmersivos "IMMERSIVETV" TSI-020302-2010-61.
- Interactive and personalized content overlay for online video "GRAFITV" TSI-090100-2011-173.
- Hogar digital y contendidos audiovisuals adaptados a los usuarios "HAUS" IPT-2011-1049-430000-AR.
- Centro comercial interactivo con interacción natural "COMINN" IPT-2012-0883-430000