Escola Politècnica Superior
de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Multiple Description Coding with Incentives for P2PTV Systems

TITLE: Multiple Description Coding with Incentives for P2PTV Systems

MASTER DEGREE: Master in Science in Telecommunication Engineering & Management

AUTHOR: Guillermo Enero González

DIRECTOR: Antoni Oller Arcas

CO-DIRECTOR: Alberto José González Cela

DATE: November 2nd 2009

Títol: Multiple Description Coding with Incentives for P2PTV Systems

Autor: Guillermo Enero González

Director: Antoni Oller Arcas

Subdirector: Alberto José González Cela

Data: 2 de Novembre de 2009

Overview

Degut a la amplia adopció d'accés de banda ampla a nivell residencial, diversos nous serveis d'Internet han sorgit. I un d'aquest serveis és la televisió basada en IP, que bàsicament consisteix en la distribució de continguts de televisió digital als usuaris a través de xarxes IP. Entre les diferents maneres de desplegar infraestructures de televisió IP, l'ús de sistemes de P2P TV és dels més populars. Però tot i que diverses aplicacions basades en aquest tipus de sistemes han estat desplegades amb un cert èxit encara hi ha molt treball a fer per tal de poder oferir la mateixa experiència d'usuari i qualitat que els sistemes tradicionals d'emissió de TV ofereixen.

L'objectiu d'aquest projecte és el de proporcionar un estudi d'algunes de les noves tècniques que podrien millorar el funcionament dels actuals sistemes de P2P TV. En concret, les dos tècniques que s'avaluaran són Multiple Description Coding (MDC) i estratègies basades en l'ús d'incentius per la redistribució. La intenció es poder determinar si l'ús d'aquestes tècniques pot ser una bona opció de cara a desenvolupar nous i millors sistemes de P2P TV o si pel contrari les millores que introdueixen no queden justificats tenint en compte el rendiment global. Finalment, no només s'ha d'extreure una conclusió al respecte si no que també s'han de poder plantejar les línies de treball futures en aquest àmbit.

Title: Multiple Description Coding with Incentives for P2PTV Systems

Author: Guillermo Enero González

Director: Antoni Oller Arcas

Co-director: Alberto José González Cela

Date: November 2nd 2009

Overview

With the widespread adoption of broadband residential access, several new internet services have emerged. And one of these services is Internet Protocol based TV, which mainly consists on digital television content delivered to the users over IP networks. Amongst the different ways of deploying an IPTV infrastructure, the use of P2P TV streaming systems is one of the most popular. But although several applications based on this type of systems have been successfully deployed, there is still plenty of work to do in order to achieve the same user experience and quality that traditional TV broadcasting offers.

The aim of this project is to provide a study of some of the new approaches that may be able improve the performance of the existing P2P TV streaming systems. More specifically, the two techniques that are going to be evaluated are Multiple Description Coding (MDC) and the incentive-based strategies for redistribution. And the goal is to be able to determine if the use of these techniques seems to be the right approach towards newer and better P2P TV systems or if, on the other hand, the improvements introduced are not worthy considering the overall performance. Finally, by discussing the results and all the work done in the project, not only the proper conclusion must be extracted, but also an idea of the future lines of work.

TABLE OF CONTENTS

FIGURES AND TABLES

INTRODUCTION

With the widespread adoption of broadband residential access, several new internet services have emerged. Internet Protocol based TV, which mainly consists on digital television content delivered to the users over IP networks, is nowadays one of the top services on the Internet. The fact that 10% of the main China ISP's traffic is from PPLive (which is a form of TV over Internet) is an example of it. What is known as IPTV is generally offered by service provider networks that are able to ensure a certain Quality of Service (QoS) with their own infrastructure. It is the alternative to what is known as Internet Television, which is the delivery of TV content over the public Internet infrastructure via streaming or some sort of multicast (like P2P TV).

When it comes to real-time (or soft real-time) multicast video streaming two different scenarios can be considered. One scenario would be the TV viewer migration from regular TV to IP-based TV to watch the same contents (sports, news, shows, etc.). This process may be considered as a natural technological migration, where IP-based technologies introduce added value to traditional TV by providing better flexibility and interactivity. The other scenario is the result of a new trend denominated UGC (User Generated Content) where end users share their own content. For that process of technological migration from traditional TV broadcasting to an IP-based broadcasting platform the new systems must be able to provide the same user experience. Several P2P streaming systems have been successfully deployed for that matter but they do not offer the same quality, stability and user experience (almost instant channel-switching) that actual TV broadcasting ensures. Hence newer approaches are necessary in order to develop competitive P2P streaming applications.

The goal of this thesis is to study the main limitations of P2P streaming systems and propose a new approach to solve them. Considering the fact that there are several issues to study, out of which some of them may be considerably difficult to solve, the scope of this thesis will not be to solve all of them. Instead new techniques will be tested and analyzed to determine if they can address the current problems of P2P streaming systems. The considered techniques will be mainly those from the latest research regarding to P2P streaming architectures and they will be tested through simulations in an appropriate environment. Finally the results obtained from the simulations will be explained to show the potential of the new approach and quantify it according to a set of well-known metrics.

The rest of the thesis is organized as follows: the next chapter sets out the current P2P streaming context and explains both the generic architecture of P2P streaming systems and their limitations. Chapter 2 provides the state of the art for the new approaches and techniques for P2P streaming. The third chapter explains the simulation test-bed with the main metrics, considerations, etc. Chapter 4 describes the systems to be evaluated and the fifth chapter lists the results for the performed simulations and analyzes them. Chapter number 6 is the final one and contains the conclusions extracted at the end of this project.

CHAPTER 1. P2P STREAMING CONTEXT

P2P streaming systems can be defined as the P2P based applications that allow the redistribution and broadcasting of media contents (typically TV channels) in soft real time over the Internet using a distributed overlay. These systems have several advantages over the traditional centralized ones. They are more robust and fault tolerant due its decentralized architectures. Also they are inherently scalable as the growth in the number of nodes can be handled without the need of re-designing the system or adding more equipment to support it. It is also important to point out the economic factor. Broadcasting via P2PTV-systems is much cheaper for content providers and broadcasters than the traditional broadcasting solutions.

However there are also drawbacks to be considered such as the heterogeneity of network and terminal capabilities, the churn that characterizes distributed systems and the problem of free riding when considering the contribution of each peer. These technological challenges are the ones that must be studied and solved in order to design P2P streaming systems that are competitive.

It is significant that several of these applications have been successfully deployed and are capable of serving thousands of simultaneous users. The one that can be considered as the pioneer in this field, CoolStreaming, reported more than 4000 simultaneous users in 2003 [11]. More recently a new wave of P2P streaming systems have reported a huge success, claiming that tens of thousands of simultaneous users watch their channels at rates that range between 300Kbps and 1Mbps [8]. These systems include PPLive, PPStream, UUSee, SopCast, TVAnts and many more. PPLive is one of the top examples having about 110 million users, offering more than 600 channels, and supporting approximately 2 million concurrent users. The use of these applications is clearly spreading as one can deduce considering some facts like the million of concurrent users that UUSee had during the broadcast of the Olympic Games, or the Obama inauguration watched in CNN via Octoshape by 300k peers.

1.1. Generic P2P streaming architecture

The different P2P streaming systems though typically being closed and proprietary have some similarities when it comes to their design and architecture. The main idea is that there is a source and a group of clients (peers) that watch the corresponding media originated by the source (the video stream). The transmitted media is divided into chunks of video that are distributed and shared amongst the peers that are consuming that media stream. Each client receives chunks from other clients, from the source or from both. Also it is important to take into consideration the fact that the system is going to be heterogeneous in terms of peer characteristics (different upload bandwidths, different client behaviors, etc.).

Streaming applications must be able to handle throughputs that ensure high quality of video and audio for a large number of peers which behavior is very dynamic in terms of joining and departure. It is also important to remember that data has to meet deadlines to ensure smooth playback of the media content in real-time.

1.1.1. Tree-based vs. Mesh-based

The architecture of these systems is typically either tree or mesh based. In the tree solution the nodes are organized into one or multiple trees and the content is injected through those trees to the nodes, so each node receives data from a parent node, which may be the source or a peer. If the rotation of peers is low, such systems require little overhead, since packets are forwarded without the need for extra messages. The problem, though, is that for highly dynamic environments (fast turnover of peers) the managing of the tree becomes more difficult and costly (the reorganization of the tree requires an important amount of messages that generate a lot of overhead). Also as a side effect, nodes must buffer data for at least the time required to repair the tree, in order to avoid packet loss. An example of a tree-based multicast overlay is ESM [13].

The other option, mesh-based architectures, is the one that most of the actual P2PTV deployed systems are using. This approach consists on a self-organized mesh of peers where peers that have an overlay link are said to be partners. The peers establish these relationships according to certain parameters such as available content or bandwidth and then they exchange data knowing what their other partners have. Every node needs information about which chunks are owned by its peers and then an explicit request (pull) of the chunks has to be performed. The major difference from tree-based overlays is that in mesh-based systems, there is no predefined route in which data flows. This architecture involves overhead, mainly due to the exchange of buffer maps between nodes and also to the pull process. On the other hand, thanks to the fact that each node relies on multiple partners to retrieve content, mesh based systems offer good resilience to node failures. Examples of mesh-based overlays for media streaming are Chainsaw [14] and CoolStreaming/ DONet [11], as well as some of the earlier mentioned popular IPTV systems, such as PPLive [15] and PPStream [16].

A classification of several overlay multicast systems according to [1] show which ones are mesh-pull based and which systems are tree-push based (Table 1.1).

Table 1.1 Classification of overlay multicast systems

<table>
<tr><th>System</th><th>Peer discovery</th><th colspan="2">Topology</th><th>Push/Pull</th></tr>
<tr><td>ESM</td><td>Underlying mesh</td><td rowspan="3">Single</td><td rowspan="3">Tree</td><td rowspan="4">Push</td></tr>
<tr><td>Overcast</td><td>Source</td></tr>
<tr><td>NICE</td><td>Bootstrap</td></tr>
<tr><td>SplitStream</td><td>Pastry</td><td>Multiple</td></tr>
</table>

CoopNet	Source			
Chunkyspread	SwapLinks			
mTreebone	Source	Tree + Mesh		Both
Bullet	RanSub			
MeshCast	Bootstrap	Unidirectional	Mesh	Pull
PRIME	Bootstrap			
Coolstreaming	Epidemic	Bidirectional		
Chainsaw	Bootstrap			

According to [9] mesh-based approach consistently exhibits a superior performance over the tree-based approach and according to [1] mesh-base systems outperform tree-based systems in terms of scalability regarding to streaming rate increasing or overlay size increase. For further study on the comparison of tree and mesh architectures see [9].

1.1.2. Overview of mesh-pull P2P streaming systems

According to [1] mesh-based systems inject a much higher number of duplicate packets into the network, but they perform better under a variety of conditions. In particular, mesh-based systems give consistently higher application goodput (average rate of data that was received before the deadline and that had not been received before) when the number of overlay nodes, or the streaming rates increase. They also perform better under churn and large flash crowds. Their performance suffers when latencies among peers are high, however. Overall, mesh-based systems appear to be a better choice than multi-tree based systems for peer-to-peer streaming at a large scale. Currently this type of systems is the most spread type of P2P streaming solutions. The generic architecture for these systems [10] is the one depicted in Fig. 1.1.

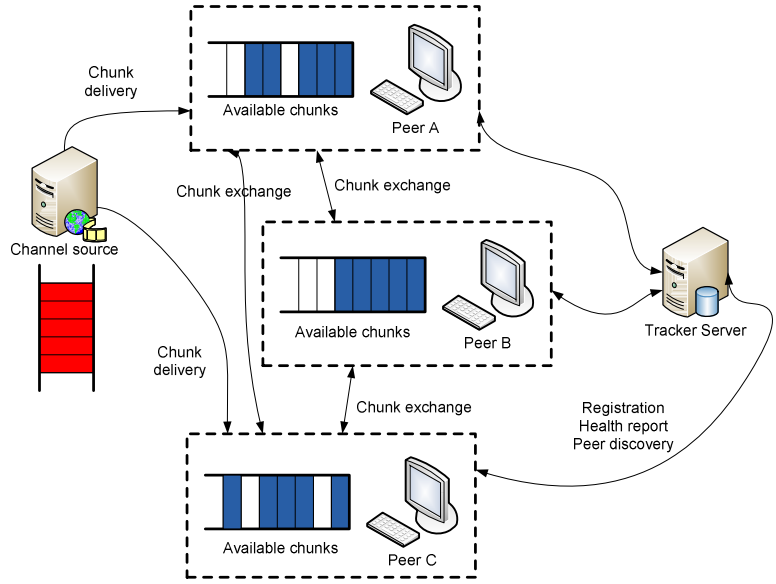

Fig. 1.1 Generic mesh-pull P2P streaming architecture

The source node serves the content (channel) dividing it into small video chunks. These chunks can be acquired from the source or from other peers because each peer cooperates with other peers to share their available video chunks. Finally, chunks are reassembled into the original video stream and sent to the player for its playback. The other element of the architecture is the Tracker server which provides the required information for peer registration and peer discovery and is also the responsible for maintaining the actualized status of the peers.

The main protocols used in these systems are: the protocols related to peer's registration, channel and peer discovery, and the protocols for the media chunk location, request and distribution.

The first step when a client joins the network is to register and obtain a list of available channels (interacting with the bootstrap or Tracker server). Once the user selects the desired channel it requests a list of peers that are consuming the same channel and communicates with those peers (additionally it can obtain new peers by asking the ones provided by the Tracker). It is important to understand that the P2P overlay can have a large number of peers and each new client only needs to know some of them (it would not be scalable to maintain a list of all the peers in each node). Typically each node has a set of partners and it exchanges information only with a reduced number of them. The list of peers with whom the node is exchanging information has to be updated to eliminate partners that are not contributing or that have a poor contribution.

Each peer buffers an amount of chunks (equivalent to a certain video time) within a sliding window. The state of its buffer at any given time is represented by a buffer map (BM) that indicate availability and peers exchange this information periodically with their partners. Knowing what chunks the other peers have available and considering the playback deadline the node's scheduler decides where to request each chunk [11]. There are different implementation for these exchange and scheduling mechanisms. Finally the received video is reassembled and sent to the media player after waiting until there is enough displayable video (consecutive chunks from the start point).

1.2. Main P2P streaming problems and limitations

P2P streaming systems present additional challenges to those of first generation P2P systems (focused on file-sharing). In these systems users not only expect to retrieve the content but also a certain degree of quality for the media. Considering that the quality of the streamed media depends on a combination of factors, related both to the characteristics of the sources (link capacity, availability, offered rate) and to the characteristics of the network (available bandwidth, packet loss rate), the goal is to focus on designing good peer selection strategies to perform high quality streaming sessions.

According to [1] and [2] the important metrics to consider when designing a P2P streaming architecture are:

- Start-up delay: the time that passes from the selection of the channel to the moment when the video starts its playback.
- End to end delay: delay between the content originator and the receiver, also known as playback delay.
- Goodput: average rate of data that was received before the deadline and that had not been received before
- Late data: average rate of data that was received after the deadline
- Duplicate data: average rate of data received before the deadline , but that had been received before
- Throughput: average rate at which all application data is received sum of goodput, late data and duplicated data
- Continuity index: is used to measure the effect of churn. (Goodput/streaming rate) defined in [14].

1.2.1. Bandwidth heterogeneity

One of the main issues that affects to these metrics is the impact of bandwidth heterogeneity in peer-to-peer systems. Streaming overlays must be able to cope with different bandwidth capabilities, both uplink and downlink (typically asymmetric), of its users (typically residential users will have a lower bandwidth than institutional peers). In [1] results of the impact of bandwidth heterogeneity for mesh-based and tree-based systems are provided. These results show that the throughput does not vary according to the amount of nodes with lower bandwidth but the useful data (goodput) does. Goodput decreases as the number of bandwidth constrained nodes increases. Both in tree-based and in mesh-based systems bandwidth-constrained peers become overwhelmed and perform a poor relay of data to the other peers. Mesh-based systems mitigate the impact of bandwidth-constrained nodes on high bandwidth nodes. But mesh-based systems penalize low bandwidth nodes since they receive very little of the stream.

1.2.2. Start-up delay

As theoretically demonstrated in [17], appropriate buffering can significantly improve video streaming quality. However, too much buffering may make the delay performance unacceptable for a streaming service. Start-up delay is the interval from when a channel is selected by the user until actual playback starts on the screen. While short start-up delay is desirable, certain amount of start-up delay is necessary for continuous playback. In [10] measurements for the average start-up delay have been performed. For popular channels 10-20 seconds are the average while non-so-popular channels may have start-up delays of up to 2min. Hence, the current state of the art of mesh-pull P2P streaming technology does not provide users with the same channel-surfing experience as traditional television. One of the possible solutions for this problem is the use of MDC mechanisms. With layered coding techniques the scheduling could be performed in such a way that a low start-up is achieved in spite of a lower initial quality that would improve gradually as the streaming advances.

1.2.3. Churn

In peer-to-peer systems the behavior of each peer in terms of entering or leaving the overlay is neither coordinated nor controlled. The joins and departures of the nodes occur at arbitrary times and these dynamics of peer participation are known as churn and are an inherent property of P2P systems. Churn significantly affects both the design and evaluation of P2P systems. It is important to understand that peers leaving the system during a given interval of time can adversely affect the performance of the system, as some nodes may find themselves disconnected or experience temporary service interruption. According to the results shown in [1] the system performance under conditions of high churn is better for mesh-based systems as they always have better continuity index.

1.2.4. Free-riding

P2P systems rely on voluntary resource contributions by individual peers. Empirical studies have shown free-riding (consuming resources without contributing) to be prevalent in P2P file-sharing networks [18] and [19]. The impact of non-cooperation in this type of systems according to [7] is that the level of cooperation affects the quality parameter Q of the system. Higher cooperation means better quality of service (QoS) as it can be seen in Fig. 1.2 (where K is the number of concurrent sessions). If the level of cooperation is low, the quality of the media is also low even when the network is not congested.

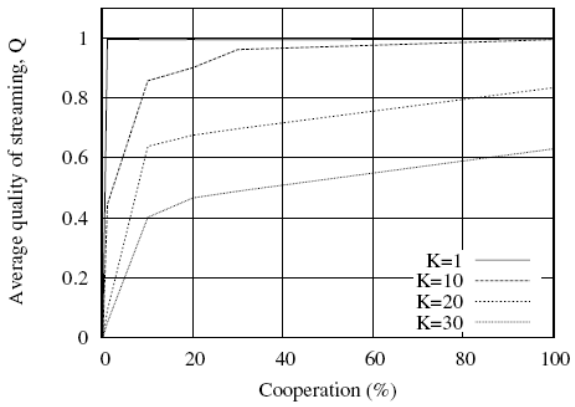

Fig. 1.2 Quality metric vs. cooperation percentage results from [7]

Due to the severe need of video chunk availability before the playback deadline, fair resource sharing hasn't been carefully addressed in the current mesh-pull systems, as no incentive mechanism has been deployed to encourage sharing between peers. Therefore an appropriate incentive mechanism should be developed in order to boost cooperation and improve the overall system performance.

CHAPTER 2. ALTERNATIVE APPROACHES FOR P2P STREAMING SYSTEMS

Considering the problems and limitations that current P2P streaming systems have (as explained in the previous chapter) it is clear that new approaches for the design of these systems are necessary. Several research has been done regarding to this matter, and in this chapter the state of the art for the main new strategies for P2P streaming will be detailed. Later, some of these strategies for new P2P streaming systems are going to be studied within the test-bed explained in the next section.

2.1. State of the art

In this section the state of the art of three techniques is going to be detailed. These techniques are Multiple Description Coding (MDC), Incentives for redistribution and Sub-Stream Trading.

2.1.1. Multiple Description Coding

Multiple Description Coding (MDC) is a coding technique that consists on the splitting of a video stream into several independent sub-streams that are displayable and that can be merged at the receiver side to reconstruct the original stream. This technique provides a more robust transmission due to the fact that the losses experienced by each sub-stream are independent (as long as they are received through disjoint paths) and that the reconstructed stream only needs a certain amount of information to display a video with acceptable quality. It also allows a certain degree of video scalability considering that the receiver end can demand a number of sub-streams according to its terminal capabilities.

An alternative to MDC is layered coding which is another coding technique for video transmission using multiple layers. This technique divides de video in several dependent layers where the base layer must always be received in order to decode the content and the subsequent layers provide enhancement of the previous ones improving gradually the video quality. According to [20] layered coding, though being more scalable and enhancing rate-control, is more sensitive to transmission losses. The decision between MDC and layered coding depends on the transmission scheme and results show that Layered Coding outperforms MDC for scenarios where rate-distortion optimized scheduling for the packet transmission is used. On the other hand, in scenarios where the packet schedule does not take into account the importance of individual packets and their dependencies MDC performs better than Layered Coding.

There are different choices for implementing MDC. The main ones are temporal, spatial and hybrid and there is always a pre-processing of the raw

video stream before the actual coding. The temporal approach consists in time interleaving where for a number of N descriptions description one will have frames 1, 1+N, 1+2N, etc. and so on for the other descriptions. This approach, according to [21], is better for videos with low movement. The spatial approach processes each frame and distributes its pixels (according to a certain algorithm) amongst the different descriptions, which is clearly better for high movement videos as shown in the results from [21]. The hybrid approach, proposed in [22], combines the two previously mentioned implementations to reduce the computational cost while maintaining an optimal performance of the MDC processing.

2.1.2. Incentive mechanisms for P2P streaming

As shown in page 8 the impact of non-cooperative environments is important in terms of received audiovisual quality and performance and it is necessary to have a mechanism that encourages cooperation between peers to solve this problem. Incentive mechanisms have been successfully deployed in file-sharing applications and they are a good approach for media streaming also. But for streaming applications there are new challenges different from those in file-sharing.

The main proposals for incentives over P2P streaming systems are to use rank-based incentive mechanisms for a global ranking of the peers and supplier-side scheduling strategies for chunk transmission based on tit-for-tat-like algorithms.

2.1.2.1. Scheduling-based incentives

In [6] a system using MDC and incentives for redistribution in P2P streaming is proposed. The system considers balanced MDC (where all descriptions have the same weight) and where each of these descriptions is divided into chunks (called DCs) and exchanged using the traditional chunk-based scheduling approaches. Some improvements though, have been introduced in order to adapt the regular chunk-based scheduling to the use of MDC and incentives. The main concept is that a peer measures its download rate from its neighbors and reciprocates them by providing a larger fraction of its upload rate.

The supplier-side scheduling proposed in [6] each supplier maintains a different request queue for each receiver. For a particular receiver the queue is first-in-first-out. The supplier transmits one DC to one receiver at one time and to determine which receiver should be served it takes into account the receiver's contribution to the supplier. At one time the supplier randomly selects a receiver and the probability of being served for that receiver is based on the download rate of the supplier from that receiver.

The receiver-side scheduler (according to [6]) has a two-dimensional buffer composed of N rows (where N is the number of descriptions) and M columns (where M is the size of the sliding window). Fig. 2.1 shows a representation of this buffer. DCs marked with X represent buffered chunks, -- represent

requested chunks, O corresponds to unavailable DCs and blanks are for available but not yet requested chunks.

D1	X	O		X				
D2		X	O	--		O		
D3	X	--	--					
D4	X	X	X		X	O	O	

Fig. 2.1 Receiver buffer

At the beginning of each round of requests the peer looks for the first DC time that has no buffered or requested DC. If there is availability for that DC time a DC will be requested randomly from the available ones. This process continues until each DC time has at least one buffered or requested DC. Then the heuristic continues with the DC times with just one buffered or requested DC and so on.

2.1.2.2. Rank-based incentives

A rank-based peer selection mechanism has been proposed in [7] in order to provide incentives through service differentiation. The proposal consists on a score-based incentive mechanism that is designed to encourage cooperation through indirect reciprocity between peers. The scheme for this mechanism is the one shown in Fig. 2.2. The idea is that the score earned contributing to other peers is mapped to a percentile rank that allows a fair peer selection. This process results in better quality of service.

Fig. 2.2 Score-based incentive mechanism scheme

The contribution of user i is denoted by X_i and it earns the score S_i that is mapped into the rank R_i. Peer selection depends on the rank of the requestors and candidate suppliers. Newcomers and free-riders are treated identically to avoid whitewashing (when free-riders take profit of systems that help newcomers, by simulating that they are newcomers).

Considering B_{out} as the amount of uploaded bytes and B_{in} as the amount of downloaded bytes, there are different possibilities f or the scoring function. One option is to consider the score as the contribution or uploaded bytes ($S = B_{out}$). Another option is to take the difference between uploaded and downloaded bytes to consider not only how much a peer contributes but also how much it consumes ($S = B_{out} – B_{in}$). And the same can be done taking $B = B_{out}/B_{in}$. Also ageing factors can be applied to encourage continuous contributions.

One issue when designing these score-based systems is whether to use a reward-penalty scheme or not. These schemes penalize users for refusing to serve a request just as they are rewarded when they contribute. But defections are difficult to detect [23] so a reward-only scheme can be a better approach.

As it has been said before, the ability of a user for selecting suppliers is determined by its score. Requests from user with score S_i are going to be responded only by nodes with $S \leq S_i$. Peers with $S \leq S_i$ can respond but are not bound to do it.

The scores are mapped into a percentile rank (based on the global distribution of the scores), but without a centralized entity, peers can locally estimate their rank based on a sample of user scores.

2.1.3. Substream trading

One of the main issues when designing a mesh-based P2P streaming system is the definition of the basic content unit for notifying, requesting and exchanging information. The widely adopted solution is to divide the stream into chunks worth of a few seconds of video time. This approach is the most used because it provides flexibility and robustness. However the chunk-based strategy has a drawback that is the playback delay and overhead trade-off because to reduce the playback lag frequent data availability messages need to be sent but this increases the signaling overhead.

One alternative approach is the Substream trading mechanism proposed in [8]. For a video stream, time dividing (like the one performed for the chunk-based approach) is done and each Substream is responsible for a set of these chunks as depicted in Fig. 2.3. In this example there would be 4 substreams and Substream 1 would be responsible for chunks 1, 5, 9 and so on.

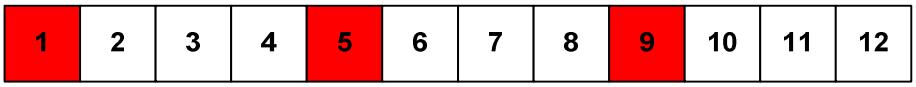

Fig. 2.3 Substream scheme

Then, when a supplier is assigned to send a particular substream to a receiver, it forwards any received chunk from that substream to the receiver without explicit notification. The number of desirable partners is approximately the number of substreams, and the information that the peers exchange is substream availability instead of chunk availability.

The partner selection mechanism proposed in [8] is the following. When two peers P and Q meet and they both have spare bandwidth (this is calculated based on the number of substreams, their rate, and the maximum upload rate of the peer) they decide if they want to form a partnership by exchanging

substream availability information. If they cannot trade immediately because one of them does not have substreams that the other peer is interested in they have to decide whether to form the partnership or not (this is because the partnership can be useful in the future when new substreams are available). In that situation they will form the partnership if they both agree and they will agree with probability:

$$P_x = (S - S_x/S)^{\gamma} \quad \textbf{(2.1)}$$

Where S is the total number of substreams, Sx is the number of substreams for peer x and γ is a control parameter.

The partnership maintenance mechanism proposed in [8] considers that when peers enter into a non-trading relationship there is a threshold W_n and when they cannot re-start their full-trading relationship before W_n they break the partnership. During the time before W_n where one peer is not trading with the other, the peer that has substreams for his partner will deliver them “for free”. To monitor the trading procedure peers use a double sliding window to control de downloading rates (one small window W_s and a large window W_l).

CHAPTER 3. VALIDATION TEST-BED

There are three main options to validate the techniques explained in the previous chapter: the analytical approach, the experimental approach and the simulation approach. The first one consists on the examination of a mathematical model of the system to provide an analytical solution. Its main drawback is that it is only suited for simple models [24] and many of the complexities of the real P2P systems are not taken into account [25]. There is also the possibility of implementing the designed system and deploying the software solution in an appropriate test-bed (with a large number of nodes) to perform a set of experiments to validate it. This solution has a time cost that exceeds the planning of this thesis. For the testing of the designed system the adopted approach has been to use a simulator. Simulators address the impracticalities of the first two alternatives. They also make it easier to implement the system and allow the evaluation of the main parameters under a controlled environment.

To perform the corresponding simulations a specific simulation software is needed. There are two options to choose from: the first one is to develop a simulator from scratch and the second one is to use an existing simulator that allows configuration or that can be modified for the purpose of this thesis. Here, the second option has been selected and a study of the different available simulators for P2P applications has been carried out.

Currently there are several simulators specifically designed for P2P applications testing. But it is important to take into account some criteria in order to select the one that is better suited for your purposes. Scalability, available documentation, learning curve and system limitations are some of the most critical characteristics that should be evaluated. According to the survey presented in [26] and the description of the systems from [27],[28] and [29] the selected simulator has been P2PTVSim [30]. P2PTVSim is a peer-to-peer TV event-driven simulator initially developed at the Politecnico Di Torino, that ended up evolving into a more general simulator to which several partners have contributed. It simulates the flow of chunks through a network of interconnected peers and provides a extensive set of statistics. It is a highly configurable simulator and the source code (in C++ programming language) is available and very well documented, making it easier to add modules and functionalities to the simulator.

3.1 P2PTV Simulator

The P2PTVSim is an event-driven simulator written in C++ and it allows the configuration of many different scenarios by simply modifying a set of well-defined parameters of the configuration file (config.txt). Once the configuration file is set according to the system that is going to be simulated the simulator can be started with its executable file (simulator.exe). When it finishes the statistics files are written to disc and stored in the output folder.

3.1.1 P2PTVSim configuration file

The configuration file (config.txt) is where the parameters of the system to be simulated are set. For an example of different configurations see ANNEX 1. CONFIGURATION FILES. This file allows the user to specify all the characteristics of the simulated scenario. The following sections provide an overview of the main parameters and for the rest of them see ANNEX 1. CONFIGURATION FILES.

3.1.1.1 Overlay types

One the most important parameters is the type of overlay that is going to be used. The parameter in the configuration file is "OverlayType" and the possible values are shown in Table 3.1.

Table 3.1 Overlay type parameters

Overlay type	Description
FULL	Fully connected degree N-1 overlay (each peer connected to every other peer).
GNR	Subclass of MBWO. Each peer is connected to a randomly chosen fixed-size subset of other peers.
GNP	Subclass of MBWO. Each peer is connected to a random set of neighbors and the degree is determined by a poisson process.
GNKREG	Subclass of MBWO. Random K-regular graph.
TREE	A tree structured directed graph.
MBWO	Abstract overlay of multi-bandwidth peers.
AS_AWARE	An AS aware overlay.
PREBUILT	Overlay that has been output from a previous simulator run.

3.1.1.2 Peer types

Another key parameter is the type of peer for the overlay. The parameter that has to be configured in config.txt is "PeerType" and the allowed values are shown in Table 3.2. Each type of peer has a different scheduling strategy.

Table 3.2 Peer type parameters

Peer Type	Description
LATEST_BA	Selects the most recent useful chunk and a random target neighbor, weighted by the target's upload BW.
LATEST_RANDOM	Selects the most recent chunk a random neighbor.
CONFIG_PEER	A re-configurable whose of chunk-scheduling strategy can be selected amongst the different available options.
AO_PEER	This peer dynamically adjusts its neighborhood to

	maintain a specified fraction of its outlinks unused over a time window.
AS_AW_PEER	An AS-network aware peer.

The config_peer type is a special type because it allows more combination for the scheduling approach. It can be dynamically configured over a range of chunk-scheduling strategies. The strategy is set by calling the set_strategy method that parses a string containing a sequence of tokens setting various attributes. There are two basic strategies that are the “best” strategy and the “weighted” strategy and they define how an entity of a specific type is selected from a subset of such entities using a filter function and then a weight function to weight the different elements.

A “best” strategy chooses the candidate with the highest weight, while a “weighted” strategy chooses one probabilistically according to the weight. The main parameters for the configuration of these strategies are the “filter” and the “weight”. If “filter” is NULL then all entities are evaluated, and if “weight” is NULL then all entities have value 1 and a random choice will be made for those passing the filter.

The chunk-scheduling strategies use algorithms to select chunks and peers. The different algorithms can be classified as “chunk first” or “peer first”. The “chunk first” approach selects the chunk and then an appropriate neighbor while the “peer first” approach selects a neighbor and then an appropriate chunk is chosen.

The filters used for to define the properties of the desired scheduling strategy are shown in Table 3.3.

Table 3.3 Configuration parameters for the config_peer

Type	**Values**	**Description**
Chunk filter	All chunks	All chunks are selected.
Chunk filter	UsefulToSelected	Only chunks that are needed by the selected peer.
Chunk filter	UsefulToSomeone	Avoid chunks that all the peers have.
Chunk weight	UnitWtChunk	Generic weight.
Chunk weight	Recentness	Weight based on the recentness of the chunk.
Peer filter	AllPeers	All peers are selected.
Peer filter	NeedsAny	Avoid peers that don’t need any of the available chunks.
Peer filter	NeedsSelected	Only peers that need the selected chunk.
Peer weight	UnitWtPeer	Generic weight.
Peer weight	UploadBW	Weight based on the upload bandwidth.
Peer weight	DownloadBW	Weight based on the download bandwidth.

3.1.1.3 Host types

The host is the element of the P2PTVSim that acts as the access point for the peer to get to the network. It has the bandwidth (both uplink and downlink) configuration as well as the latencies associated. This class is the responsible for the chunk delivery. Within the configuration file the "HostType" parameter can be configured to be one of the following (Table 3.4):

Table 3.4 Host type parameters

Type	**Description**
OLD_SIM	Chunk delivery time is computed based on BW and latency and events are scheduled for that time to notify the sender and the receiver.
TWEAKED_OLD_SIM	Like OLD_SIM except that when a high BW host is feeding a low BW one, the source completes its send at a time determined by its own upload BW and gets called back at that time, allowing to schedule other chunks.
PRIORITY_SHARING	Allows multiple chunks from lower-speed sources to be downloaded by the host at the same time but in a prioritized manner.

3.1.1.4 Additional configuration parameters

There are other parameters besides overlay, peer and host types that are important. In the next table (Table 3.5) you can see them with their definition.

Table 3.5 Additional configuration parameters

Parameter	**Description**
OverlaySeed	Random seed for generating the initial overlay network
DiffusionSeed	Random seed used for chunk diffusion
NumPeers	Number of peers in the overlay (not including source)
Degree	Size of the neighborhood
NumChunks	Number of chunks that will be diffused trhough the network before the statistics are collected
PeerBandwidths	Defines a set of different badwidth classes for peers
SourceUploadBW	Upload BW of the source peer
PlayoutDelay	Max. num. of seconds from when a chunk is available at the source until it arrives at the peer. Chunks that take longer to arrive are recorded as received but not redistributed.
ChunkDelayCounters	Sets how many counters (for chunk delay) are stored
ChunkSize	Sets the size of the chunk in MB
VideoRate	Sets the video rate in Mbps

The ChunkDelayCounters parameter is used when full-chunk arrival is not performed. It allows to store a certain amount of chunk delays. Each peer maintains a histogram of the delays of the chunks it receives from when they became available at the source. This parameter sets how many such counters, i.e., histogram bins, are stored at each peer. The bins span the range 0 to *PlayoutDelay*, with the last bin also recording any late arrivals. For example, if the *PlayoutDelay* is 5 seconds and *ChunkDelay_Counters* is 10, the bins will be for delays of 0-0.5, 0.5-1, 1-1.5, etc up until 4.5-infinity [7].

3.1.2 P2PTVSim outputs

P2PTVSim prints the gathered statistics into a set of text files (with extension .out) after performing the whole simulation. The list of files is shown in Table 3.6. Additional outputs can be extracted from the simulator by modifying the source code.

Table 3.6 Output files of the simulator

File name	Description
ConfigDump.out	The complete set of properties definend by config.txt
Degree.out	One line/peer: Peer num|degree|uplink BW|downlink BW
Links.out	One line/link (bidirectional): Source peer num|dest peer num
Upload_rate_peer.out	One line/peer: Peer num|uplink BW|num of BW class
Absolute_time_arrival.out	Matrix of (NUM_CHUNKSxNUM_PEERS) Each element is the absolute time a given chunk i arrived at peer j.
Chunk_delay.out	Same as the previous file but in terms of delay from when it was available at the source to when it arrived at the peer.
Out_of_sequence.out	One line/peer: num of out-of sequence chunks
Total_Chunk_Losses_per_peer.out	One line/peer: num of chunks failed to receive
Throughput_peer.out	One line/link: Source peer num|Dest. Peer num.|Chunks sent
Upload_total_peer.out	One line/peer: Peer num|num. of uploaded chunks
DelayHistograms.out	Gives the final state of the chunk delay counters (see ChunDelayCounters parameter described in section 5.1.1.4)

3.1.3 How P2PTVSim works

In this section the main algorithms used in P2PTVSim are going to be described as well as the behavior of the event-driven engine that performs the simulations. For more insight into the simulator's implementation the source code can be checked at [7]. Although it does not provide any API the code is well commented and self-explanatory.

The main thread of the simulator performs the operations shown in the following diagram (Fig. 3.1). First of all, it loads the configuration parameters both from the configuration file and from the optional arguments provided by the command line. Then it sets up the overlay, host and peer types according to the configuration parameters. When all the configuration is done, the simulator initializes the overlay. The overlay initialization consists on the creation of all the peers (including the source) and the establishment of their partnerships. When the overlay is created all callback function (for statistics gathering purposes) are set as well as the churning peers (if the configuration file includes churning). At this point the system is prepared and the chunk diffusion begins. And when the chunk diffusion is finished the statistics are dumped into the corresponding files.

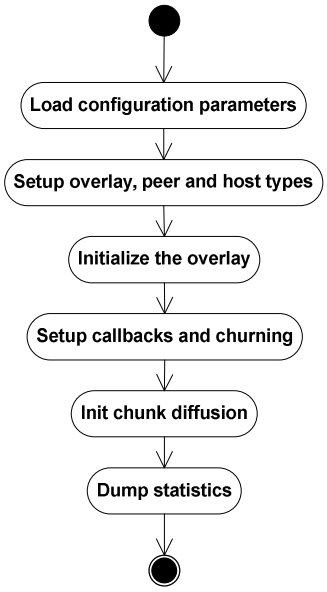

Fig. 3.1 P2PTVSim main thread execution

The simulator, as it has been stated before, is an event-driven simulator. It has an event queue where all the events that take place during the simulation are stored and eventually processed and handled. Each event has a type and a time, amongst other parameters. The following table (Table 3.7) shows the four types of events.

Table 3.7 Events defined in the simulator

Event type	Description
EVENT_CHUNK_SENT	Notify a peer that the chunk it was

	sending has been received and now it can schedule a new upload.
EVENT_CHUNK_DELIVERED	Notify a peer that the chunk has been received an inserted into its window.
EVENT_CHUNK_INITIATION	Give the next chunk to the source node to transmit it.
EVENT_CHUNK_TIMEOUT	When a chunk can't be delivered.

3.1.3.1 Overlay initialization

The overlay factory creates the corresponding overlay according to the "OverlayType" configuration parameter provided. Here the overlay type considered for the simulations is GNROverlay which is a multiple-bandwidth mesh overlay. The process of creating the overlay is shown in Fig. 3.2.

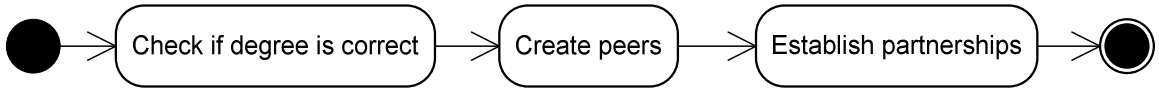

Fig. 3.2 Overlay creation

Peers are created using the bandwidth classes defined in the configuration file so that the number of peers of each class is the correct one. Partnerships in the GNR overlay are established using the "make_random_neighborhood" function that links a peer to a set of N other random peers (where N is the configured degree).

3.1.3.2 Churning configuration

P2PTVSim allows the simulation of churn (peer rotation). It can be used only if the configured overlay is of the type GNR and the peers are AOPeers (Adaptative Overlay Peers). For that purpose the configuration file has a parameter that corresponds to the percentage of churning nodes that are going to be set for the simulation. If the required parameters are set the peers that are marked as churners will make a departure from the overlay at a specific time and then after another period of time they will connect as new peers. The time that those peers are going to be connected is obtained using the minimum and maximum connection time (established in the configuration file) and a seed number and passing them to the uniform function.

3.1.3.3 Chunk diffusion

After every element in the system is configured the simulation starts. The trigger is the "init_chunk_diffusion" function and after that the Event Handler starts to process the events stored in the event queue. The first event, of the type EVENT_CHUNK_INITIATION, is generated by "init_chunk_diffusion". This

event triggers a loop in the source node that works according to the state diagram depicted in Fig. 3.3.

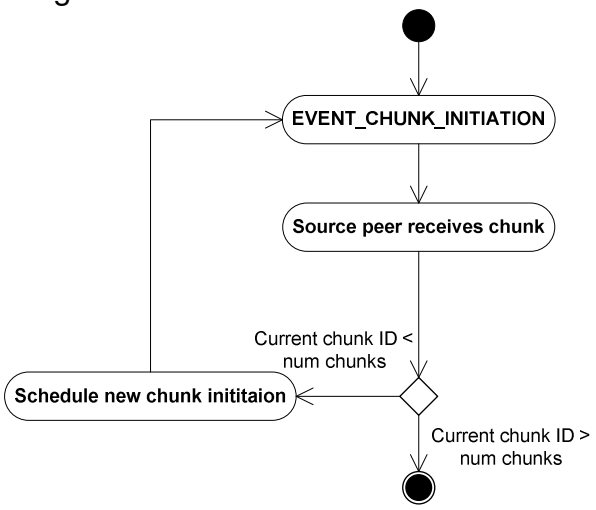

Fig. 3.3 Chunk initiation loop diagram

Upon the reception of a chunk, any peer (including the source node) inserts the chunk in its window, records the arrival in the corresponding statistics file, and if the variable p_upload_busy is false (that is, the peer is not currently uploading any chunk to another peer) it schedules a new chunk upload. The next figure (Fig. 3.4) shows the state diagram of the chunk scheduling process performed in the simulator by any peer.

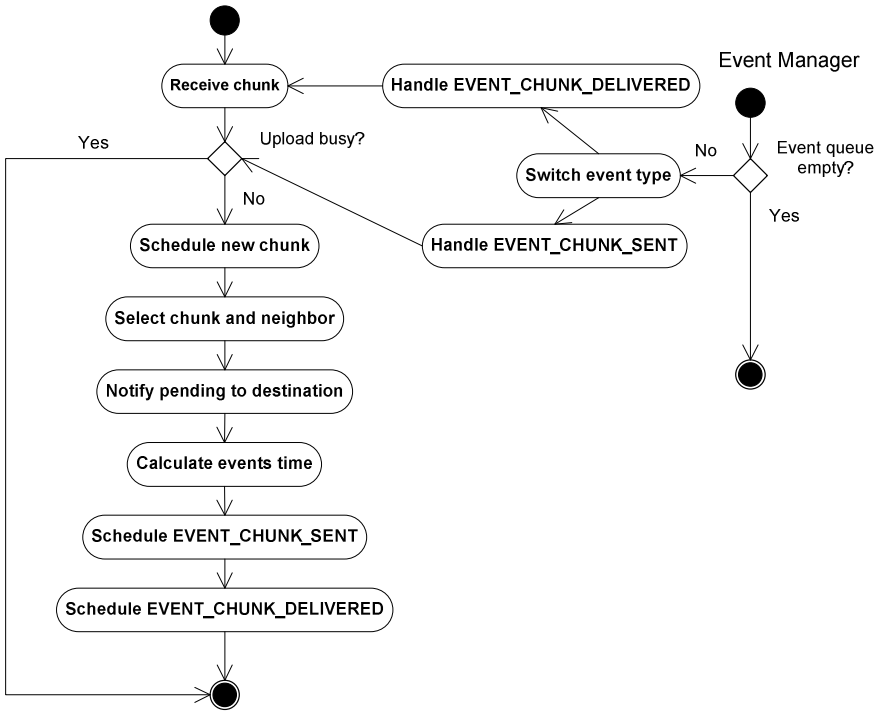

Fig. 3.4 Chunk scheduling diagram

The schedule of a new chunk can be triggered either by the reception of a chunk or by the end of the transmission of the previous chunk. Once the scheduling begins (provided that the peer is not already busy sending a chunk) the first step is to select the neighbor that is going to be served and then send one the possible requested chunks. This can be performed in different ways depending on the type of peer (see Table 3.2 and Table 3.3).

Once the chunk and the peer that is going to receive it are selected the simulator continues by notifying the destination peer that the selected chunk is on its way. Then the events (CHUNK_SENT and CHUNK_DELIVERED) are scheduled for a certain time that is calculated according to the bandwidths, delays and latencies associated to the peers. These events will eventually trigger further chunk scheduling.

3.2 P2PTVSim modifications

The code of the P2PTVSim has been modified in order to simulate the techniques that are being discussed in this thesis (that is MDC and incentives for redistribution). Four different simulator versions have been compiled for the scenarios that are going to be evaluated.

Table 3.8 Simulator modified versions

Version	Characteristics
2.0	Original version with peer-first scheduling
2.1	Original version modified to work with MDC and peer-first scheduling
2.2	Original version with incentives for redistribution and peer-first scheduling
2.3	Original version working with MDC and incentives for redistribution and peer-first scheduling

Further work would be to modify the code in order to configure the use of these different techniques with a set of new parameters in the configuration file but for the simulations performed in this thesis the four different versions are going to be used independently.

3.2.1 General modifications

There is one change that has been applied to all four versions of the simulator. It consists on the modification of the scheduler for the AOPeer to make it a peer-first scheduler (see peer types in page 16). The use of AOPeer for the simulations of this thesis is motivated by the fact that it is the only type of peer to implement churn. But the AOPeer implementation provided by the original simulator uses a chunk-first strategy (again, see peer types in page 16) and the scenario studied here works with peer-first scheduling strategies.

3.2.2 Incentives modifications

The simulator has been changed to perform the supplier-side scheduling in a weighed fashion where peers that contribute more are rewarded and have more weight, thus being more likely to be served first. For that purpose several modifications have been made in the source code of P2PTVSim. The following classes have been modified:

- AOPeer.cpp
- Stats.cpp
- Stats.h
- Peer.cpp
- Peer.h

The main change comes in the scheduling function. When a peer has to select one neighbor to be served from all the possible requests it has to do the selection according to a weight. The weight of a neighbor is related to the amount of information that the peer has received from that neighbor, and the stats files (header and source) have been rewritten to capture this information and make it available to the scheduler. The corresponding methods have been also added to the peer classes (that encapsulate the access to the stats). Finally in the AOPeer scheduler the function that chooses the weighted neighbor has been modified also to use the corresponding weight.

3.2.3 MDC modifications

The original P2PTVSim simulates the flow of chunks for a single video trace through a P2P TV Streaming overlay. Using MDC means that instead of a single video stream, the source splits the video in several substreams (see ANNEX 2. MULTIPLE DESCRIPTION CODING) and distributes them. To perform this type of P2P video streaming in P2PTVSim several files have been rewritten also, and these are:

- Overlay.cpp
- Overlay.h
- Videostream.cpp
- Videostream.h
- Window.cpp
- Window.h
- AOPeer.cpp
- Stats.cpp
- Stats.h

The window and the videostream classes have been redefined to work with multiple streams instead of just one. Now the window is a matrix of chunks and the videostream has the corresponding methods to deal with the distribution of these substreams.

Also the stats have been changed in order to record the chunk arrivals according to the new scheme. The files that record the chunk arrival time and delay (see Table 3.6) now record it for every chunk adding the descriptor that the chunk belongs to.

And again, the scheduler of the AOPeer has been rewritten to deal with the multiple streams and redistribute them accordingly. The scheme for receiver-side scheduling is the one described in page 10.

CHAPTER 4. EVALUATED SYSTEMS DESCRIPTION

With the appropriate test bed set and ready the different systems and configurations can be evaluated but first one must specify which scenarios to deal with. In this project the systems that are going to be evaluated in order to compare the different techniques and extract conclusions are the four systems detailed in this section.

4.1 Base system

The P2PTV system that is going to be considered as the reference is defined as base system. It is a mesh-pull P2P streaming system (as described in 1.1.2) based on the CoolStreaming/DONet design [11]. The main algorithms for this system are detailed below.

4.1.1 Neighborhood selection and maintenance

When a new peer enters the overlay a random set of existing peers is assigned as its neighborhood. This process is typically performed by connecting to a centralized tracker or to a bootstrap node, but in the simulator this is simply done by selecting the random set from the list that the simulator itself holds and manages.

That neighborhood is maintained by the peer by performing periodic adjusting of the list of partners. These adjustments are done according to the number of unused links. Two levels are defined to know when to adjust and how, low-tide level and high-tide level. So every time the neighborhood update is started the first thing that the peer does is to count all the unused links since last scan. If the number is lower than the low-tide level the neighborhood is increased by adding a suitable subset of the partners' neighborhoods. If the number is higher than the high-tide level the neighborhood is reduced.

4.1.2 Chunk scheduling

There are two scheduling algorithms regarding the exchange of chunks, the supplier-side scheduling and the receiver-side scheduling. The supplier-side scheduling works as follows: the peer that is going to send one of its chunks to one of its neighbors has to select amongst all the partners that are requesting chunks from him. This selection in the base system is performed randomly. Then when the neighbor is selected the first chunks that this neighbor requested (the 'best' chunk according to a certain criteria) is sent.

The chunk scheduling strategy at receiver-side is based on chunk recentness. Basically the chunk is selected by a weighted function according to its recentness so the chunks that are newer are requested with higher probability.

4.2 Base system with incentives

This system is an evolution of the previous one. Here the neighborhood selection and maintenance algorithms are the same and the overall system has the same design. Even the chunk scheduling strategy for the receivers is the same. The only change is in the supplier-side scheduling.

Incentives for redistribution are introduced in this system in order to encourage cooperation for a better overall performance of the system. This means that when a peer has to select which of its requesting neighbors is going to be served first, instead of a random selection it uses a weighed selection. This weighed selection is such that neighbors contributing more to this peer are more likely to be selected and served than those who don't contribute as much. This strategy prevents free-riders from consuming resources from other peers without contributing themselves to the system leading to a poorer performance of the system.

4.3 MDC system

The Multiple Description Coding (MDC) based system introduces several changes regarding to its predecessors. Here the main difference is that instead of a single stream being transmitted through the overlay, by exchanging its chunks, there are several substreams that set up the whole transmitted stream. Each individual substream is displayable by itself and it can also be combined with the rest of substreams which results in a better video quality. As it can be seen in Fig. 4.1 the buffer map, that represents the chunks each peer has, is made up of a set of substreams (6 in this example) and the chunks the substream is divided into. The group of chunks of different descriptions (substreams) that belong to the same time are called a DC time (Descriptor Chunk time). Again, the neighborhood selection and maintenance algorithms are the same as the ones described in 4.1.1.

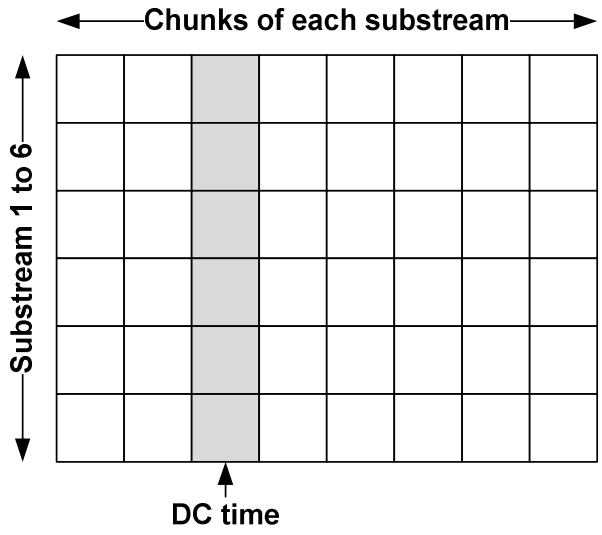

Fig. 4.1 MDC buffer map

Supplier-side chunk scheduling is the same as the one that the base system uses. The receiver-side scheduling, though, is different. In this scheme the algorithm for chunk request scheduling starts by looking for the first DC time without a chunk buffered or requested. If there is such a DC time a random chunk from that DC time is requested. The algorithm continues by looking for the first DC time with just one chunk buffered or requested and so on (with the limit being the number of descriptors).

4.4 MDC system with incentives

The last system to be evaluated is a combination of the algorithms explained previously. Here the neighborhood selection and maintenance algorithms are also like the ones detailed in 4.1.1 and the receiver-side chunk scheduling strategies are the same as the strategies used in the MDC system. The difference with that previous system is that here the supplier-side scheduling algorithm is incentive based like in the system defined in 4.2.

CHAPTER 5. ANALYSIS OF ALTERNATIVE APPROACHES FOR P2P STREAMING

The systems to be evaluated have been explained as well as the validation test-bed. Now the configuration of that test-bed and the set of tests designed to evaluate the different P2P TV systems are going to be detailed along with the obtained results.

5.1 Test set description

Here the main parameters for the configuration of the P2PTVSim are listed with the rest of the configuration and environment characteristics. Also the list of tests performed are explained.

5.1.1 Main configuration parameters

There are some configuration parameters that are different for each simulation or that may depend on the scenario but there are others that are general to all the simulations performed. In this section these general parameters are listed and discussed.

Table 5.1 Main simulations configuration parameters

Parameter	**Value**
Chunk size	0.1Mbit
Video rate	1Mbps
Playout delay	5 seconds
Overlay type	GNR (see Table 3.1)
Peer type	AOPeer (see Table 3.2)
Num. peers	1000
Num. chunks	2500
Degree	10
Source upload BW	5Mbps

As it is shown in Table 5.1 the video rate is 1Mbps and the chunks size is 1Mbit. The video used as reference has a resolution of 768x576 and has been coded to that rate using a MPEG container and a MPEG-2 video codec. With these values and the specified chunk size the total running time for the simulations has been set to 1000 chunks which is the equivalent to 14 minutes of this specific video.

The playout delay is set to 5 seconds which is the buffer size (chunks older than that are discarded). This playout delay is an important parameter to take into account when considering the losses and delay of the simulations. The type of peers chosen for the simulations is AOPeer because it is the only type of peer provided by the P2PTVSim that allows churning configuration as well as

multiple bandwidth profiles. And the type of overlay of P2PTVSim that works with these peers is the GNR one, which is a multiple-bandwidth overlay. The degree is the average number of neighbors that a node will have, and in this case 10 has been the selected value. This is also an important parameter that has impact in the simulations and that has to be considered when discussing the conclusions of the results. The overlay that is going to be built and simulated in each test is made up of 1000 peers.

As stated previously, the peers can be configured to have different bandwidths by defining a set of profiles. The profiles established for this set simulations (4 different profiles) are show in Table 5.2.

Table 5.2 Peer profiles for the simulations

Peer type	Upload BW	Download BW	Percentage
Class A	5Mbps	Inf.	10%
Class B	1Mbps	Inf.	40%
Class C	500Kbps	Inf.	40%
Class D	0Kbps	Inf.	10%

All four profiles have an infinite download bandwidth. This is a typical assumption used in many of the studies related to this type of systems (like in [5] or [6]) that state that the download bandwidth constraint is not a parameter that affects strongly the result of the simulations. Also it is important to mention that the source peer is a class A peer and thus it has an upload bandwidth of 5MBps.

5.1.2 Software and hardware environment

The software used for the simulations is the simulator P2PTVSim with the modifications introduced within the context of this project (see CHAPTER 3) and it runs on Windows Vista OS using a HP Pavilion dv4 laptop (this has no impact on the results of the simulations other than the time required to complete them).

5.1.3 Tests set

The set of tests defined to obtain the required results are the result of the combination of the elements in Table 5.3:

Table 5.3 Parameters to define the set of simulations

Scenarios	Parameters	Variables
Base	Continuity Index	Losses (%)
Base with incentives	Delay	Churn (%)
MDC	Average number of descriptors	
MDC with incentives		

For each of the four scenarios the parameters that are going to be measured are the continuity index and the delay. The continuity index is defined in the single description scenarios (Base and Base with incentives) as follows:

$$\frac{Num.\,of\ received\ chunks}{Total\ number\ of\ chunks} = Continuity\ Index\ (CI) \qquad \textbf{(5.1)}$$

And for the Multiple Description Coding (MDC) scenarios the Continuity Index is obtained according to:

$$\frac{Num.\,of\ DC\ times\ with\ one\ or\ more\ chunk}{Total\ number\ of\ chunks} = Continuity\ Index\ (CI) \qquad \textbf{(5.2)}$$

The delay for a chunk is the difference between the time at which the chunk arrives to the node and the time at which the chunk was available at the source. Then for each peer the average delay (average of all the chunk delays) is calculated.

For the MDC systems an additional parameter is considered to add measure of received quality and complement the continuity index parameter. This is the average number of descriptors that measures the average of the number of different chunks for DC time for all the stream.

These three parameters are measured for their corresponding scenarios in different conditions by changing some variables. The variables used here are the percentage of losses and the percentage of churn. The percentage of losses includes the losses caused by all the possible factors (transmission errors, late packets, etc.). So each parameter measured in each system is obtained for several values of these two variables. Specifically the losses percentage values used are 0%, 1%, 5%, 10%, 15%, 20% and 40% and the values of churning percentage are 0%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% and 100%. It is also important to point out that each simulation (the same combination of parameters, scenario, and variables) is performed 5 times and averaged so the total amount of simulations performed is 360.

5.2 Analysis of test results

Once all the simulations have been performed the results can be collected and represented in several graphs to extract conclusions about them. In this section these results are shown and discussed.

5.2.1 Impact of churn and losses

First, the behavior of the base system for different values of the two variables, losses and churn, is explained. This is done in order to understand the results of the different systems to evaluate by comparing them to this reference system.

As it shows the next figure (Fig. 5.1) an increase of the percentage of losses in the system results in a decrease of the continuity index. The impact, though, may seem lower than what it could be expected, considering that a 20% is a really high percentage of losses and thus the CI should be lower than the CI in this figure. The reason for this behavior is that the degree and the playout delay (both parameters explained in section 5.1.1) are big enough to have a positive impact in the CI. A bigger playout delay results in less chunks being lost because of being late. A bigger degree also reduces losses by increasing the number of possible sources to retrieve the chunks. So the graph of Fig. 5.1 shows a behavior that in a system with lower degree and playout delay would be the same between the 0% and 25 or 20% losses range (here the 40% losses point is introduced to show that behavior but it is a really high amount of losses).

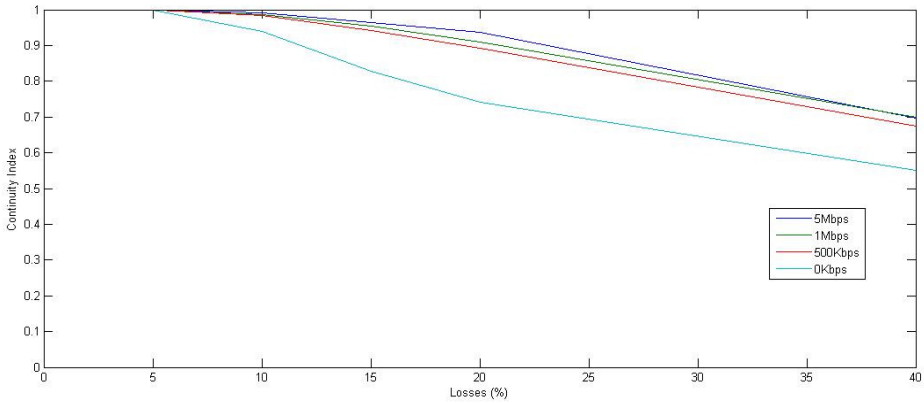

Fig. 5.1 CI vs. Losses - Base system

The impact of churn in the CI of the system is almost inexistent as it can be seen in Fig. 5.2. This, again, is due to the effect of the size of the degree. A bigger degree means that peers are more tolerant to churn because the departure of a peer is less likely to affect the probability of receiving the required chunks (the number of options to retrieve the chunk is higher).

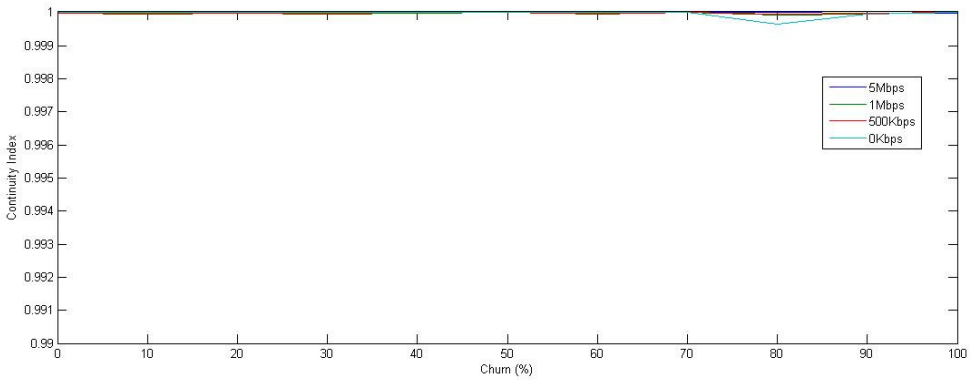

Fig. 5.2 CI vs. Churn - Base system

From the results shown in Fig. 5.3 the same conclusion as the one in the previous graph can be extracted: the impact of churn is almost inexistent. Here the delay is not heavily increased by the increase of churn. Even when all the peers are churners (which means that at some point they are going to leave the system and that new peers can also enter) the delay stays more or less the same.

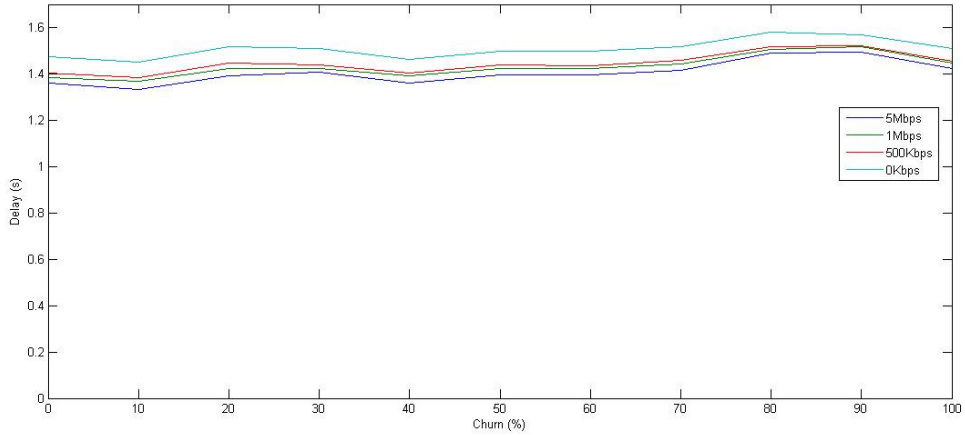

Fig. 5.3 Delay vs. Churn - Base system

The effect of losses, on the contrary, can be noticed when looking at the evolution of churn for different loss percentages. The figure below (Fig. 5.4) shows how the initial delay is about 1.5 seconds for all the four types of peers while the delay increases to 3 (peers of classes A,B and C) and almost 3.4 for class D peers (free-riders) when there is a 20% of losses.

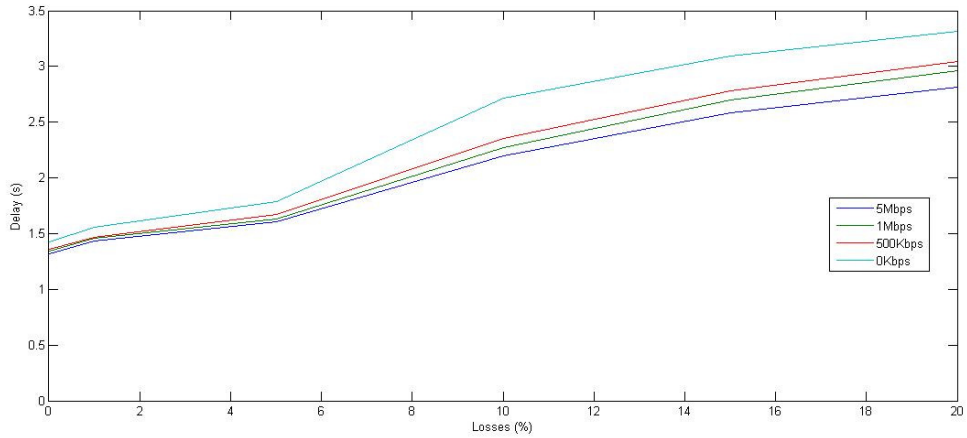

Fig. 5.4 Delay vs. Losses - Base system

The next figure (Fig. 5.5) shows the delay dispersion for three loss percentage values (0%, 5% and 10%). Here the values of delay for each peer are displayed so we can see the variance and also that the overall behavior corresponds to the results of Fig. 5.4.

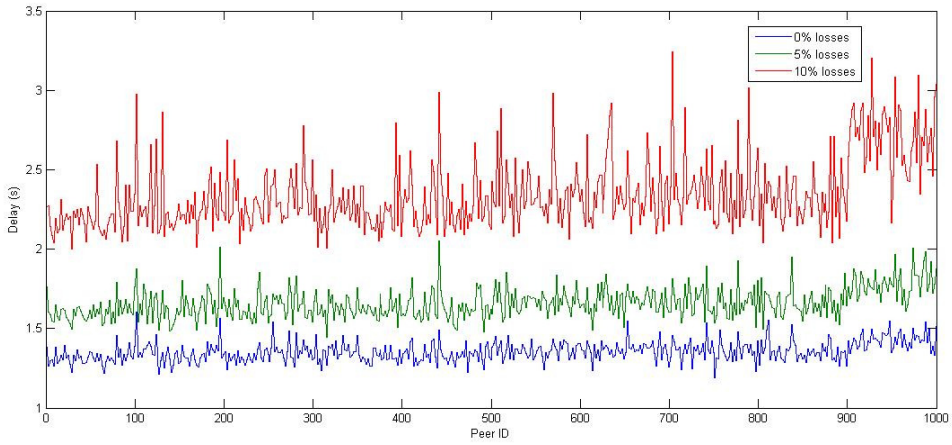

Fig. 5.5 Delay dispersion for 0%, 5% and 10% losses - Base system

5.2.2 Incentives for redistribution

The graphs from the previous section showed that depending on the type of peer (and more specifically on the upload bandwidth) the results are better or worse, with peers that contribute more having better measures than the others. This concept of rewarding the contribution is known as incentives for redistribution and it can be forced to improve the overall performance of the system. The difference between peer types in the base system is the result of the periodic adjusting of the neighborhood where peers that contribute more have a more stable neighborhood and therefore less losses and delay.

In Fig. 5.6 the CI for different degrees of churn is shown and it is clear that the CI of the free-riders (peers of class D, with 0Kbps of upload BW) has decreased drastically and that the CI of the rest of the peers has only improved a little compared to the base system. The use of incentives here may not be justified because the rest of the peers did not have the need to improve (the CI on the base system was almost 1).

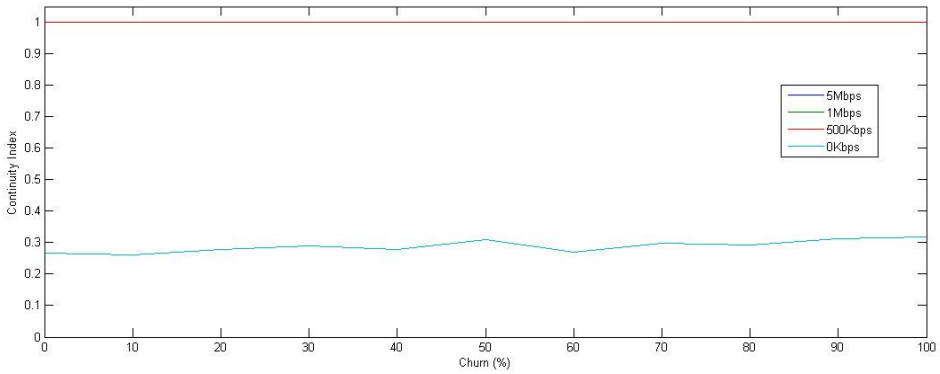

Fig. 5.6 CI vs. Churn - Base system with incentives

But studying the impact of losses and how the use of incentives improves the CI of peers of classes A, B and C (Fig. 5.7) it is clear that there is a significant

improvement. Free-riders have a very low CI because the highly aggressive incentives system punishes the peers that do not contribute to the system. On the other hand the rest of the peers have a better CI against losses. Peers of class A almost keep their near-to-one CI while peers of class B and C have a good CI almost until the 20% of losses and from that point they decrease to 0.8 and 0.65 respectively.

The concept is simple: if the peers that are not contributing start exchanging chunks and uploading the overall performance of the system gets better and if they do not cooperate the same overall performance is tried to achieve by punishing those peer (they are served rarely) and thus increasing the CI of the rest of the peers. Again, the fact that the CI is that good even after the 20% losses barrier is related to the configuration of the degree and playout delay for these simulations.

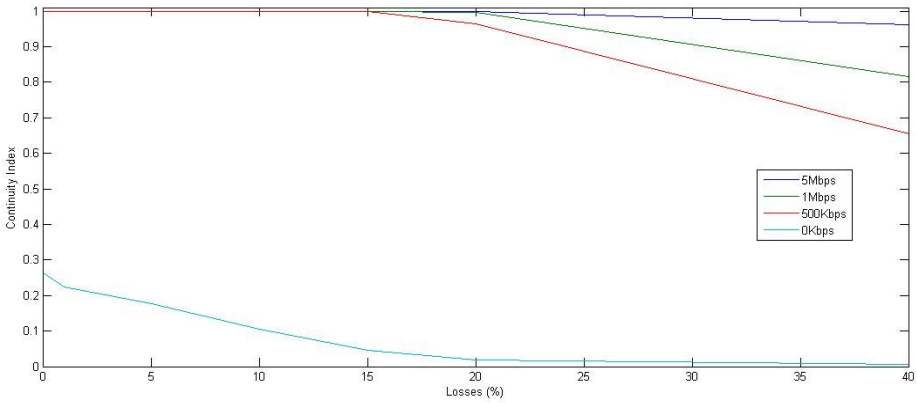

Fig. 5.7 CI vs. Losses - Base system with incentives

When looking at the delay measured (Fig. 5.8) when the base system with incentives is under the effect of churn it can be seen that the increase of churn is not affecting drastically the results. The delay for free-riders though has increased in general due to the use of incentives (from 1,4 seconds to more than 2,5 seconds). This decrease for the free-riders has the consequence of the rest of the peers having a lower delay (from 1,4 approximately to a delay between 0,5 and 0,7 seconds) which is another of the expected outcomes of the use of incentives.

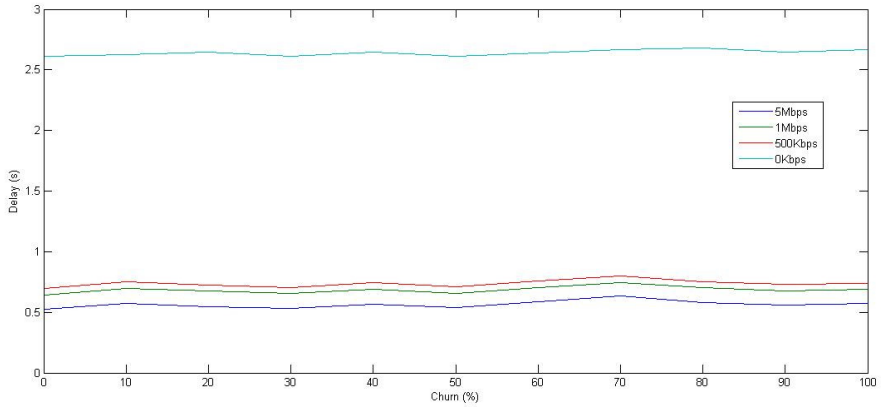

Fig. 5.8 Delay vs. Churn - Base system with incentives

The effect of losses in the delay is depicted in Fig. 5.9 and it shows how the increase of the losses percentage results in an increase of the delay. In this scenario the values of the delay are different than the ones in the base system (Fig. 5.4). Here the use of incentives, again, punishes the peers that do not cooperate and as a consequence their overall delay increases (from 1,5 to more than 2,5 seconds at the lower stages of losses). On the other hand this incentive-based strategy benefits the rest of the peers that lower their delays to about 0,6 seconds (when the losses are between 0% and 10%). The decrease of the delay for the free-riders from the 10% of losses to the 40% is not a real decrease but the effect of the way that measures and representations have been done for this graph. What is happening is that from the 10% losses level the free-riders have more losses and as the delay displayed in the graph is only the average of the received packets the value obtained is lower than it should be. If the delay of the lost packets should be counted here the behavior of the free-riders line would be similar to the behavior of the rest of the peers.

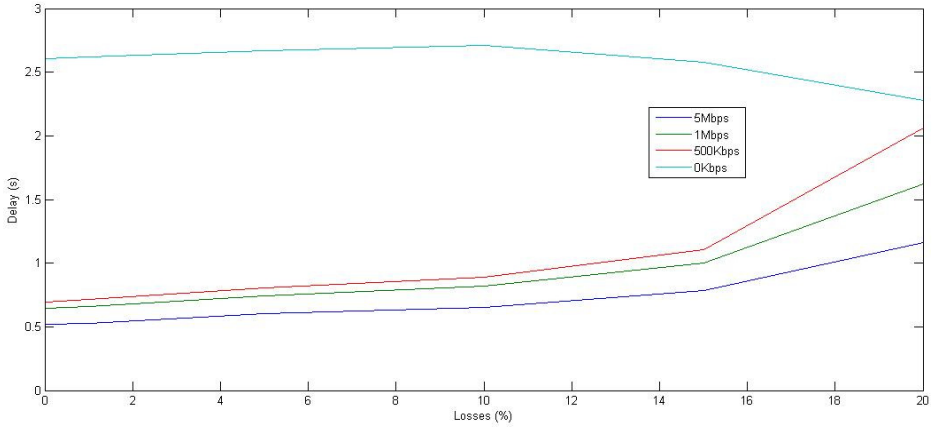

Fig. 5.9 Delay vs. Losses - Base system with incentives

Again, in order to have an idea of the variation of the delay for different scenarios, the delay dispersion (values of delay for each peer) are shown in Fig. 5.10. The IDs of the peers are ordered according to the classes so the first 100 peers are class A peers, peers from 100 to 500 are class B peers, peers from 500 to 900 are class C peers and the last 100 are free-riders (class D).

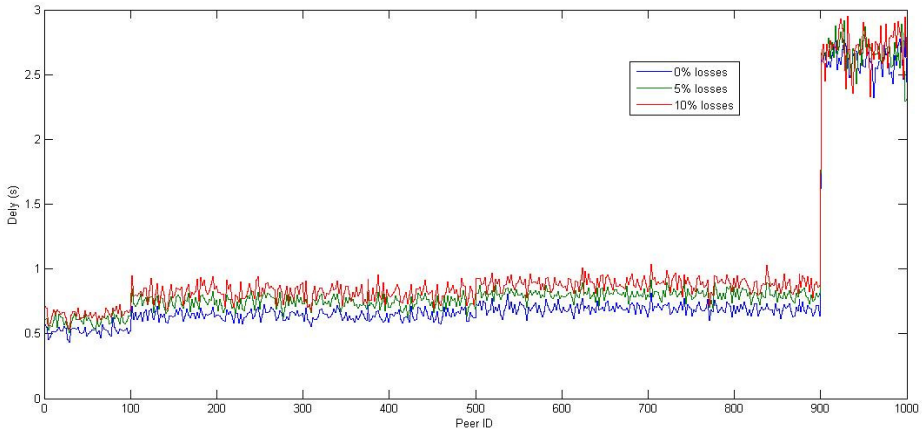

Fig. 5.10 Delay dispersion for 0%, 5% and 10% losses - Base system with incentives

5.2.3 MDC performance

The impact of churn and losses has been discussed as well as the improvements introduced by the use of incentives. Now the differences between the performance of the single layered systems vs. the Multiple Description system are going to be studied.

First as it can be seen from Fig. 5.11 the impact of churn on the CI is not significant while the losses (Fig. 5.12) have some effect (especially when passing the 20% range). One of the conclusions that can be obtained from the comparison between the CI in this system and in the previous ones is that it has increased even for the simulations where the amount of losses was the 40%.

The lowest CI in the MDC system is the one at 40% of losses and its value is around the 0,9 which is a really solid continuity index. The single layered systems (Fig. 5.1 and Fig. 5.7) had a worst CI from the 20% loss barrier of about 0,7 in the base system and between 0,7 and 0,8 for the incentive-based system. Here at the point of 20% of losses de CI is almost 1 it only drops to 0,9 from that point. This is the result of using a multiple layered coding scheme where to keep the continuity index only one chunk of each DC time is needed.

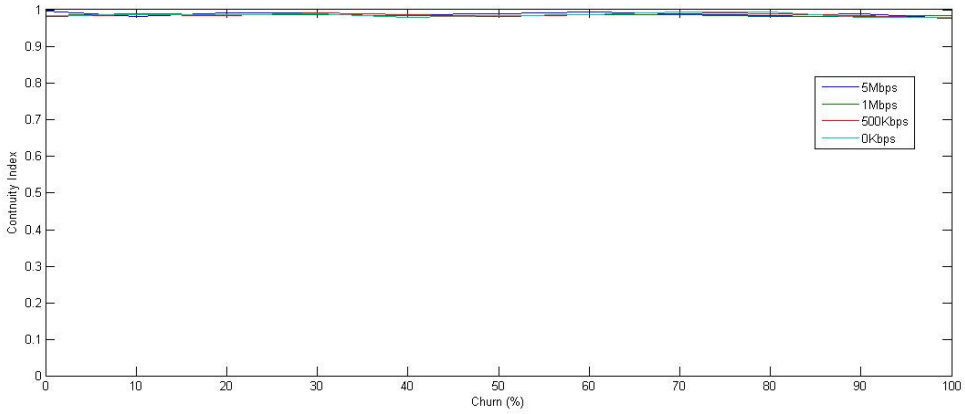

Fig. 5.11 CI vs. Churn - MDC system

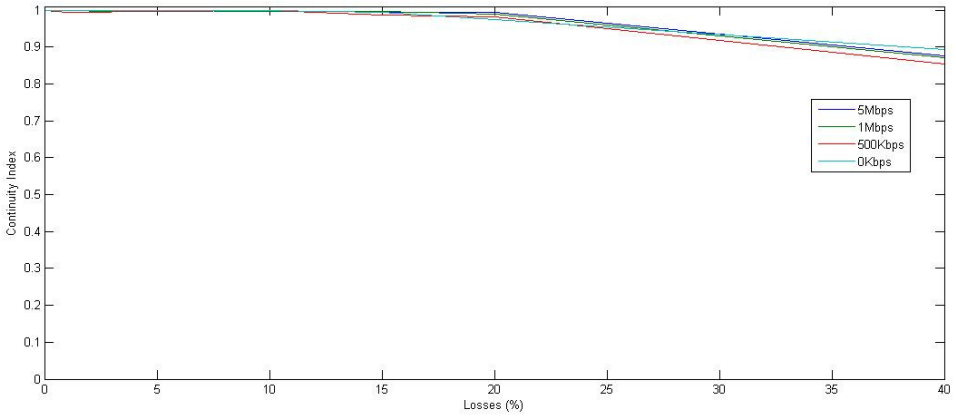

Fig. 5.12 CI vs. Losses - MDC system

But to have a sense of the real perceived quality of the user an additional measure is needed. This measure is the average number of received descriptions per DC time and it is depicted in Fig. 5.13. As it can be seen there is a difference between the different peer classes, though in the CI graph they seem to be receiving the same. But the CI only indicates if the displayed video has continuity or not. This measure on the other hand allows to put a quantity value on how good is the quality. Of course this is not the typical video quality measure (it is not like PSNR or other similar calculations) but the average number of received descriptors indicates the level of quality because the more descriptions the peer gets the better the merged video will be (better PSNR) is indirectly it is showing the quality level.

Now as the graph (Fig. 5.13) shows the class A peers receive more description on average than the others and the class B and C are more or less on the same range while free-riders receive less descriptors. The behavior when there are losses is that the average number of descriptors decreases and it affects all the peers in the same fashion.

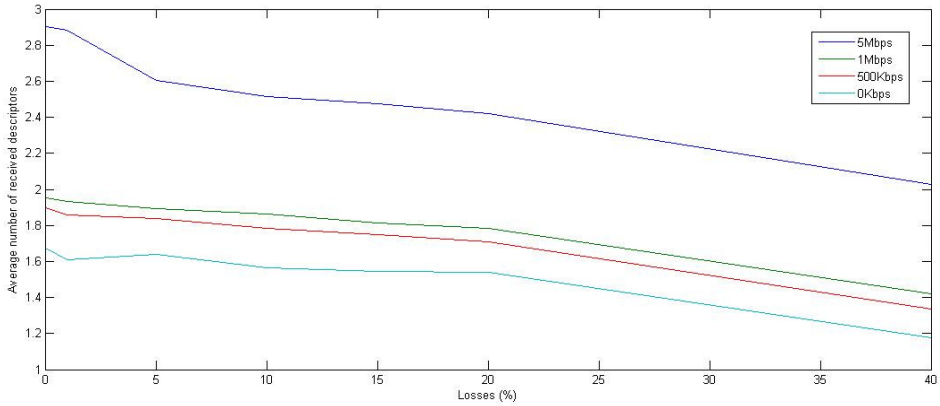

Fig. 5.13 Avg. number of received descriptors vs. Losses - MDC system

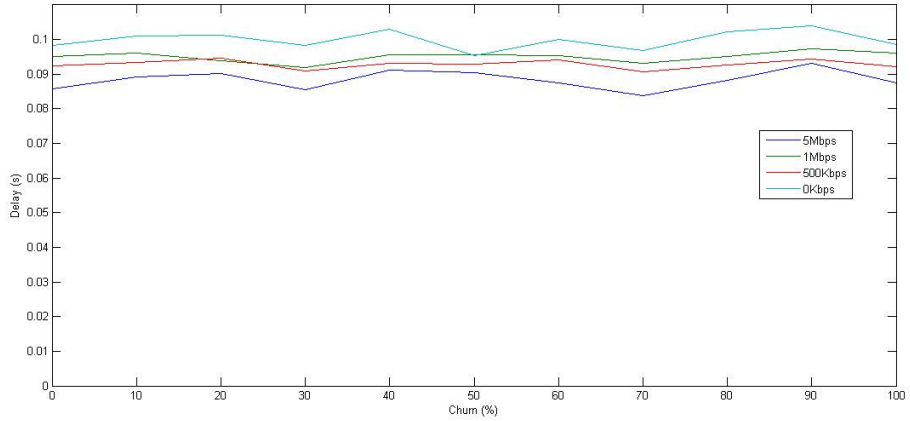

Fig. 5.14 Delay vs. Churn - MDC system

The delay when the MDC system is being used decreases significantly compared to the delay of the single layered system (in the range of 0,5 to 3,5 seconds). Here it is about 0,08 seconds (see Fig. 5.14 above) which is a very

low delay and it is due to the fact that the chunks are smaller and therefore their associated delay decreases accordingly. The size of the chunks in relation to the chunks of the single layered system decreases according to the number of descriptors used. The delay associated to the processing time required when using MDC would increase the delay so it would be higher than the one showed in these graphs but it would still be lower than the delay of the single layered systems.

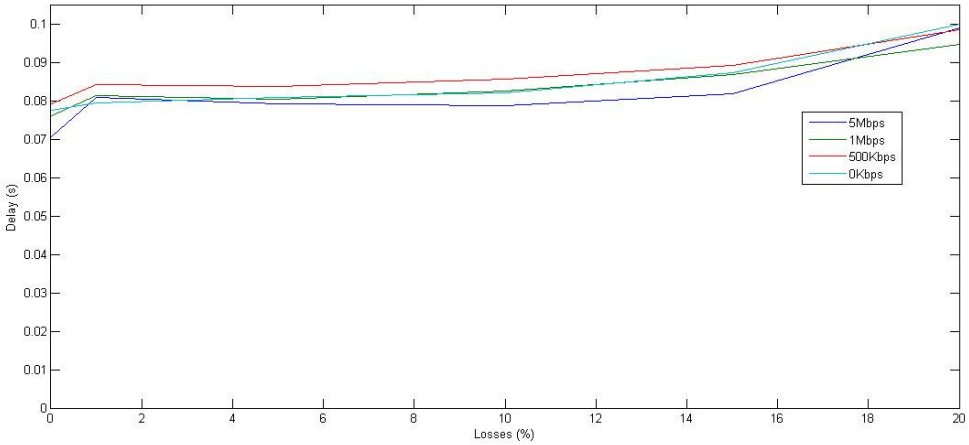

Fig. 5.15 Delay vs. Losses - MDC system

This low delay is perfectly suited for the use of MDC as a technique for fast start-up scheduling. In systems where the user wants to have a low start-up delay when changing from one channel to another it would be possible to use this technique to send a lower quality version of the stream (with a lower delay) and allow a zapping styled browsing of the channels.

Another important point to discuss is the fact that there are drawbacks to the use of MDC systems. These drawbacks are mainly the high resource consumption on the computers and the signaling overhead generated because of the important increase of the number of chunks and thus the size of the buffer maps. Here the measure of these parameters has not been included for practical reasons but it is a must to in order to have an idea on whether it is worth or not to use MDC system vs. Single layered systems.

5.2.4 Combining MDC with incentives for redistribution

From the previous results, the conclusion that both the MDC techniques and the use of incentives introduce an improvement in the P2P TV systems can be extracted. And combining them seems to offer an interesting approach to evaluate. This last system is the result of a MDC system that uses incentives for redistribution. In this section the performance of this last scenario is going to be discussed.

The first result is the continuity index (CI) measured for different churning percentages. The graph (Fig. 5.16) shows how the CI is practically one for peers of classes A, B and C while peers of class D (free-riders) have a lower CI

(about 0,8). The low effect of churn is again present here but the difference from the result obtained in the base system with incentives (Fig. 5.6) is that due to the use of MDC the CI for the free-riders only decreases to 0,8 instead of reaching the 0,3 level. So in this system even the free-riders being punished because of their lack of cooperation can achieve a minimum quality with a pretty good CI.

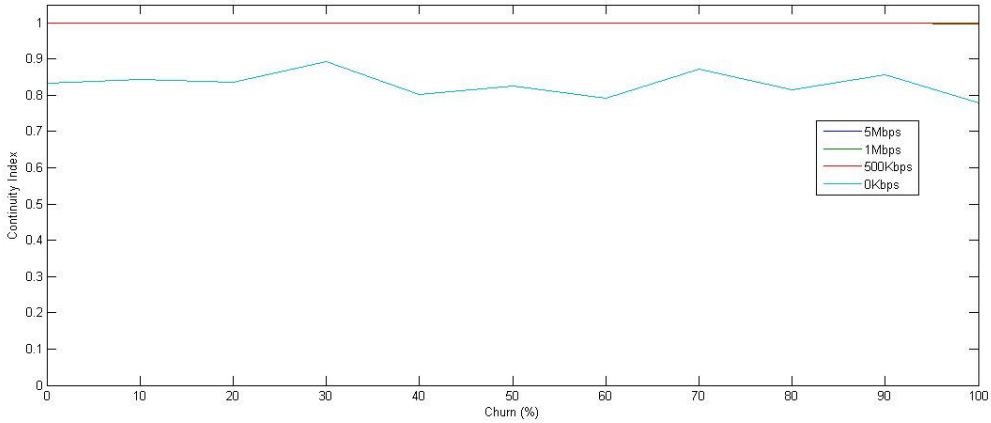

Fig. 5.16 CI vs. Churn - MDC system with incentives

When losses are taken into consideration the results show that the CI for free-riders drops drastically (the combination of losses and the aggressive incentive-based mechanism is the responsible for this result). The rest of the peers maintain a very good CI level (as it can be seen in Fig. 5.17), dropping only to about 0,95 in the worst case (40%). Again, this shows how robust is the MDC technique in front o f losses by providing a really stable CI. The increase of CI for the free-riders from the point of 15% losses has the same explanation as the one given in 5.2.2 for Fig. 5.9.

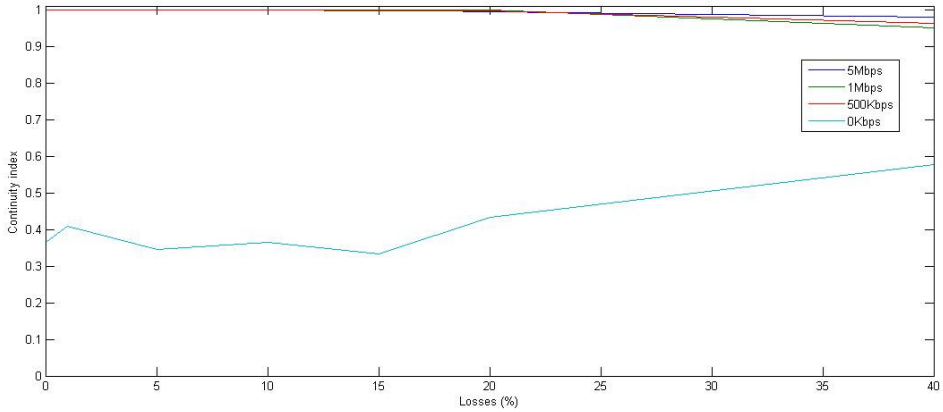

Fig. 5.17 CI vs. Losses - MDC system with incentives

Like the it has been discussed in the previous section, the CI by itself cannot be used to describe the quality of the received video at the peers. It only shows the continuity and so another measure is needed. The next graph (Fig. 5.18) shows the result for the MDC system with incentives. As the figure shows, the average number of descriptors for the free-riders has also dropped (not only they have

now a bad CI but also less quality) to less than 1 descriptor. That means that most of the times the free-riders have no chunk for a DC time and for some they do have one chunk.

Also the improvement for the rest of the peers is more obvious here. The CI from the MDC system to this MDC system with incentives has only improved from 0,9 in the worst case (40%) to almost 1, which is not a bad improvement, but this measure only does not justify the use of incentives. On the other hand the results of Fig. 5.18 show that where the real improvement is taken place is in the average number of descriptors. Peers of class A have an average of almost 3,25 descriptors at 0% losses and almost 2,25 at 40% which is better than the 2,9 to 2 average descriptors range in the MDC system. The peers from classes B and C have improved from a range of 1,9 to 1,4 descriptors to a range of approximately 2,6 to 1,6. As it can be seen the overall quality improves though at the highest level of losses the improvement is not so big. It is important to remember that the average number of descriptors only provides an idea of the degree of quality that the peers are receiving but to evaluate if the improvement from 2,9 to 3,25 descriptors in average is a good improvement, a PSNR measure should be performed and discussed.

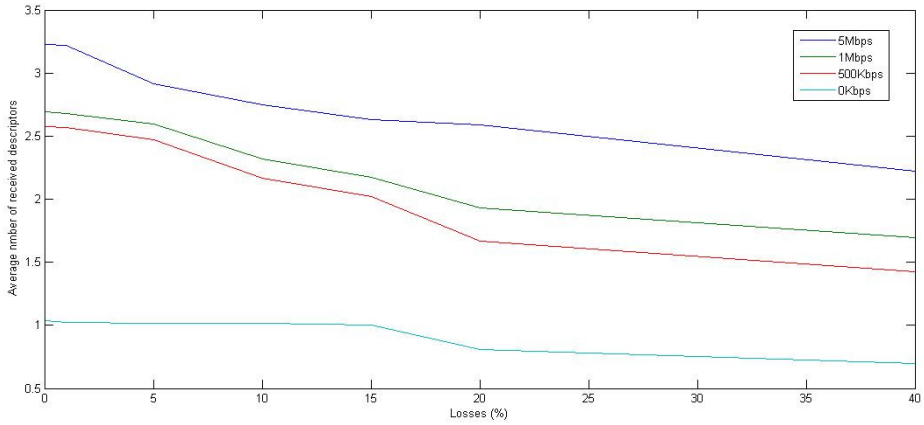

Fig. 5.18 Avg. number of received descriptors vs. Losses - MDC system with incentives

The delay in the MDC system with incentives is similar to the delay measured in the MDC system and therefore the same conclusions can be extracted. The only difference is that with the use of incentives the delay for the free-riders increases but in this case it does not result in a decrease of the delay for the rest of the peers. In Fig. 5.19 and Fig. 5.20 the behavior of the delays for different churn and losses percentages are depicted.

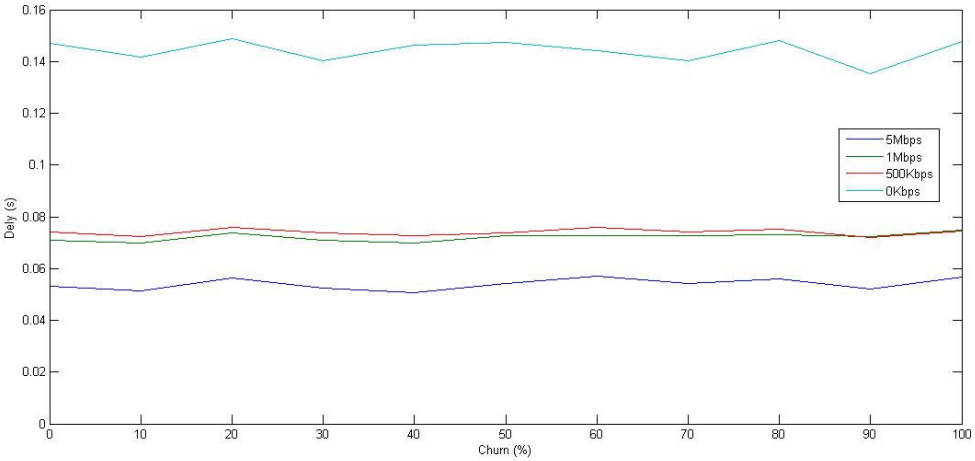

Fig. 5.19 Delay vs. Churn - MDC system with incentives

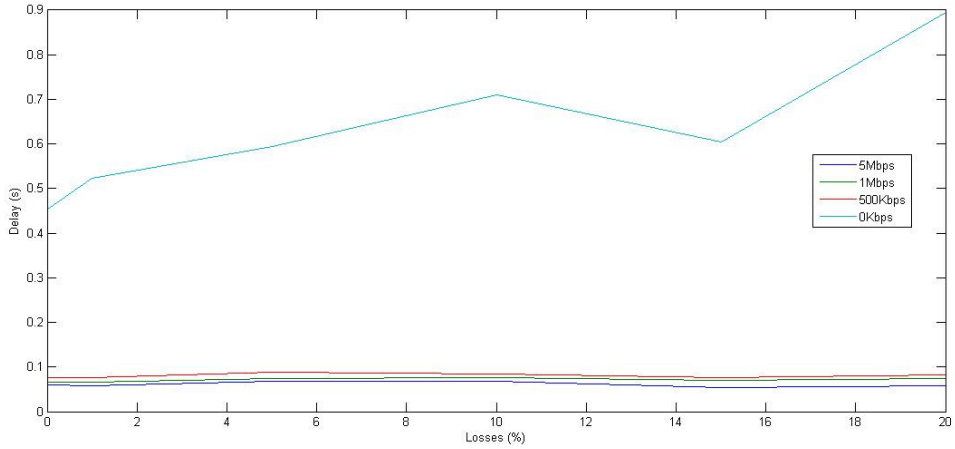

Fig. 5.20 Delay vs. Losses - MDC system with incentives

CHAPTER 6. CONCLUSIONS AND FUTURE WORK

Nowadays the use of distributed applications has is well spread and as a result of the first P2P TV systems (like CoolStreaming [11]) several of these P2P TV streaming applications have been successfully deployed. The problem is that these applications are not able to provide the same user experience as the traditional TV broadcasting networks.

From the different proposals to improve these systems that have been published during the last years several stand out from the rest and in that group techniques like Multiple Description Coding or Substream Trading are included. Here in this project the initial aim was to study some of these new approaches by using a simulation environment. The idea was to evaluate them and decide if they are the most suitable improvements to P2P TV streaming systems. And finally the techniques that have been studied by using a modified P2P TV simulator have been MDC and incentives for redistribution.

Several measures and conclusions have been extracted and the results have been discussed in detail in 5.2. In this section the main conclusions of the whole project are going to be explained as well as the future lines of work in this area.

6.1 Environmental, social and economical impact

From an environmental, economical and social perspective, instead of from the technological point of view, the results of this project have also some impact. And this is what is going to be discussed briefly in this section.

These type of applications, meaning P2P TV streaming applications, provide an alternative way of distributing media content with less resources needed. This may allow the creation of new content providers that find in these systems a way of distributing themselves the content at a low cost and being able to compete with existing and well-established traditional content providers. It also relates to the User Generated Content (UGC) trend where the users could generate and distribute their content taking advantage of these platforms. And all of this could even end up creating a whole new scenario with a wider offer in media contents.

The economical impact is found on the fact that the use of P2P TV streaming applications may provide a cheaper alternative to the traditional content distribution networks. And related to this, the environmental impact could be understood in terms of required equipment to deploy these systems, because that is what finally determines the energy consumption and the amount of equipment manufactured. That in turn is the only part of the process of development and deployment of these systems where an actual impact on the environment is taking place.

6.2 Future work

As a result of the work done in this project several new lines of study have come up and there are still some issues left to be finished to complete some the conclusions that could have been extracted. Here in this section these issues for future work are detailed and discussed.

6.2.1 Completing the results for the MDC with incentives system

As it has been mentioned while discussing the performance of the MDC based systems, there are some measurements that should have been performed in order to have a complete view of the advantages and disadvantages of these techniques.

More specifically the measurement of the overhead is one of the main issues left to study in this area. It is important to discuss if the use of MDC is worth according to the improvements it introduces and the impact it has on the overall system performance due to signaling overhead. One approach would be to measure the overhead for different number of descriptors. Then the results could be relates to the increase of overhead due to the number of descriptors and compared to the behavior of the quality (increase or decrease) for the different scenarios. Also regarding to these systems there is the fact that the resources consumed in the computers when using a MDC based system are really high as it has been studied in [21].

Another parameter that could be interesting to measure is the PSNR. By calculating the average PSNR for the whole time that a peer is connected a more accurate idea of the received quality could be obtained. This would be of special interest when evaluating the MDC based systems to complement the measures taken of the average number of descriptors.

Alternatively, another possible line of study that could lead into a set of more realistic measurements is to implement this simulator over OMNet [32] and more specifically using the models and interfaces defined in the OverSim P2P Simulator. The reason for the proposal of these new approach is that using these tools not only the simulations can be performed but there are some additional advantages. One of them is that it allows any simulator to interact with a real implementation of the same system (provided the interfaces and protocols are properly defined) deployed in the PlanetLab test-bed ([33]). That can be used to compare more realistic measurements (from the PlanetLab deployed nodes) with the ones taken in the simulator and therefore adjusting the simulator implementation and configurations to obtain more accurate results. One of the results that may be interesting for complementing the work of this project and that could be obtained using these tools is the effect that the processing in the peers has in the average delay.

6.2.2 Study of fast start-up scheduling

When studying the use of Multiple Description Coding in this project one idea came up that could be interesting to test. That idea is to use the fact that there are several layers with lower quality, and that the delay for the chunks in MDC is lower, to design a scheduler with a very low start-up time.

One possible application for that idea is the use of this fast start-up scheduling to offer to the user a zapping-based experience. The initial approach could be to show thumbnailed versions of the available channels and the when the user chooses one of them the regular scheduling starts. To implement this technique a different MDC receiver-side scheduler must be designed.

6.2.3 Study of sub-stream trading

As it has been explained in 2.1.3 besides the MDC and incentive mechanisms there are other possible new techniques that could improve the P2P TV systems, like the one proposed in [8]. By modifying again the simulator used in this project to add a new configuration parameter that allows the use of substream trading this technique could be evaluated also. As the aim of this technique is to reduce the signaling overhead the results could be compared to the ones that have been mentioned in 6.2.1.

6.3 Final conclusions

At the end of this project, after having discussed and examined all the work done and the extracted results, several conclusions can be obtained. First, regarding to the projection of the P2P TV systems: though several of these systems have already been deployed with some success, it is clear that the user experience is not like what traditional TV is offering (it is way lower). So with that in mind the target users are nowhere near the amount of users that these systems could have with a better performance.

Second, regarding to the use of MDC and incentives for the improvement of the performance of P2P TV systems: the results show that these techniques may be the appropriate new approach for P2P streaming but to determine if that is correct a deeper analysis should be performed. This analysis could be done by starting from the results and test-bed of this project and extending them according to what has been discussed in 6.2.1. By evaluating the obtained results it could be determined whether or not there is a possibility of improving this distributed systems to a level where they can offer a good enough user experience. Which is finally, the main question that should be answered to know the future of P2P TV streaming.

CHAPTER 7. BIBLIOGRAPHY

[1] Seibert, J., Zage, D., Fahmy, S. and Nita-Rotaru, C., "Experimental Comparison of Peer-to-Peer Streaming Overlays: An Application Perspective", *Technical Report CSDTR-07-020*, Department of Computer Science, Purdue University, (2007).

[2] Marfia, G., Pau, G., Di Rico, P. and Gerla, M., "P2P Streaming Systems: A Survey and Experiments", UCLA and Universita degli Studi di Pissa, (2007).

[3] Agarwal, S., Pal Singh, J., Mavlankar, A., Baccichet, P., and Girot, B., "Performance of P2P Live Video Streaming Systems on a Controlled Test-bed", *Proc. of 4th Intl. Conf. on Testbeds and Research Infrastructures for the Development of Networks & Communities*, TRIDENTCOM, Innsbruck, Austria, (2008).

[4] Silveston, T. and Formaux, O., "Towards an Incentive Mechanism for Peer-to-Peer Multimedia Live Streaming Systems", UPMC University, Paris, (2008).

[5] Liu, Z., Shen, Y., S.Panwar, S., W.Ross, K. and Wang, Y., "Using Layered Video to Provide Incentives in P2P Live Streaming", Polytechnic University, Brooklyn, 2007.

[6] Liu, Z., Shen, Y., S.Panwar, S., W.Ross, K. and Wang, Y., "P2P Video Live Streaming with MDC: Providing Incentives for Redistribution", Polytechnic University, Brooklyn (2007).

[7] Habib, A. and Chuang, J., "Incentive Mechanism for Peer-to-Peer Media Streaming", UCLA, California (2004).

[8] Liu, Z., Shen, Y., S.Panwar, S., W.Ross, K. and Wang, Y., "Substream Trading: Towards an Open P2P Live Streaming System", *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, (2008).

[9] Magharei, N., Rejaie, R. and Guo, Y., "Mesh or Multiple Tree: A Comparative Study of Live P2P Streaming Approaches", *Proceedings of IEEE INFOCOM*, (2007).

[10] Hei, X., Liang, C., Liang, J., Liu, Y. and W.Ross, K., "A Measurement Study of a Large-Scale P2P IPTV System", IPTV Workshop, International World Wide Web Conference, (2006).

[11] Zhang, X., Liu, J., Li, B. and Peter Yum, T.S., "CoolStreaming/DONet: A Data-Driven Overlay Network for Efficient Live Media Streaming", *Proceedings of IEEE INFOCOM*, (2005).

[12] Stutzbach, D. and Rejaie, R., “Towards a Better Understanding of Churn in Peer-to-Peer Networks”, *Tech Report CIS-TR-04-06*, Dpt of CS, University of Oregon, (2004).

[13] Chu, Y., Rao, S. G. and Zhang, H., “A case for end system multicast”, *in Proceedings of ACM SIGMETRICS*, (2000).

[14] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. Mohr, “Chainsaw: Eliminating trees from overlay multicast”, *in IPTPS*, (2005).

[15] http://www.pplive.com

[16] http://www.ppstream.com

[17] R. Kumar, Y. Liu, and K. W. Ross, “Stochastic fluid theory for P2P streaming systems”, *in Proceedings of INFOCOM*, (2007).

[18] E. Adar and B. Huberman, “Free riding on gnutella”, vol. 5, no. 10, (2000).

[19] S. Saroiu, K. P. Gummadi, and S. D. Gribble, “Measuring and analyzing the characteristics of napster and gnutella hosts”, *Multimedia System*, vol. 9, no. 2, pp. 170–184, (2003).

[20] J. Chakareski, S. Han, B. Girod, “Layered Coding vs. Multiple Descriptions for Video Streaming over Multiple Paths”, *Proc. ACM Multimedia*, 422–431 (2003).

[21] http://www.i2cat.net/es/projecte/trilogy-0

[22] Meng-ting, L., Chang-kuan, L., Jason, Y. and Homer H, C., “Multiple Description Coding with spatial-temporal hybrid interpolation for video streaming in peer-to-peer networks”, National Taiwan University.

[23] Feldman, M., Lai, K., Stoica, I. and Chuang, J., “Robust Incentive Techniques for Peer-to-Peer networks”. *ACM Conference on Electronic Commerce (EC'04)*, (2004).

[24] A. M. Law and D. M. Kelton, “Simulation Modeling and Analysis”. McGraw-Hill Higher Education, 1999.

[25] S. Joseph, “An Extendible Open Source P2P Simulator,” *P2P Journal*, pp. 1–15, (2003).

[26] S. Naicken, A. Basu, B. Livingston, and S. Rodhetbhai, “A Survey of Peer-to-Peer Network Simulators,” *Proceedings of The Seventh Annual Postgraduate Symposium*, Liverpool, UK, (2006).

[27] I. Baumgart, B. Heep, and S. Krause, “OverSim: A flexible overlay network simulation framework,” *in Proceedings of 10th IEEE Global*

Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007, Anchorage, USA, (2007).

[28] A.P Couto da Silva, E.Leonardi, M.Mellia, M.Meo, "A Bandwidth-Aware Scheduling Strategy for P2P-TV Systems", *8th International Conference on Peer-to-Peer Computing 2008 (P2P'08)*, (2008).

[29] Richard Lobb, Ana Paula Couto da Silva, Emilio Leonardi, Marco Mellia, Michela Meo, "Adaptive Overlay Topology for Mesh-Based P2P-TV Systems" *Proceedings of ACM NOSSDAV* (2009).

[30] http://www.napa-wine.eu/cgi-bin/twiki/view/Public/P2PTVSim

[31] http://www.oversim.org/

[32] http://www.omnetpp.org/

[33] http://www.planet-lab.org/

ANNEX

TITLE: Multiple Description Coding with Incentives for P2PTV Systems

MASTER DEGREE: Master in Science in Telecommunication Engineering & Management

AUTHOR: Guillermo Enero González

DIRECTOR: Antoni Oller Arcas

CO-DIRECTOR: Alberto José González Cela

DATE: November 2nd 2009

ANNEX 1. CONFIGURATION FILES

In this annex the configuration files used for the simulations performed in this project are provided in order to give a reference to any person that wants to use the P2PTVSim.

```
// Base system sample configuration file
// With 0% of losses and 0% of churning peers

// Video source characteristics
ChunkSize = 0.1     // Mb
VideoRate = 1.0     // Mbps

// The PlayDelay sets the time for which each peer keeps
// chunks in its window
// Chunks older than this are discarded (and hence not
// diffused to neighbours)
PlayoutDelay = 5         // Seconds

// The maximum degree of TREE overlays
MaxTreeDegree = 5

// Chunks are lost with this probability.
ProbChunkError = 0.0

// Time-jitter to add to all events before enqueing
TimeJitter = 0.000001  // secs

// Number of chunk-delay counters to keep in each PeerStats
// record
ChunkDelayCounters = 1000

// Various output and input file names
// ===================================
LinksFile = "Links"
LinksAtEndFile = "LinksAtEnd"
DegreeFile = "Degree"
DegreeAtEndFile = "DegreeAtEnd"
OutOfSequenceFile = "Out_of_sequence"
ThroughputFile = "Throughouput_peer"
TotalChunkLossesPerPeerFile = "Total_chunk_losses_per_peer"
UploadRatePeerFile = "Upload_rate_peer"
UploadTotalPeerFile = "Upload_total_peer"
DegreeOverTimeFile = "DegreeOverTime"
DelayHistogramsFile = "DelayHistograms"
ChunkRequestsOverhead = "Chunk_overhead"
BMExchangeOverhead = "BM_overhead"
ChurningTimes = "Churn_times"
ConfigDumpFile = "ConfigDump"
ChunkArrivalsFile = "Absolute_time_arrival"
```

```
ChunkDelaysFile = "Chunk_delay"
FilenameSuffix = ".out"
FilenamePrefix = "output/"

// Network specification
// =====================
NetworkType = NONE
HostType = OLD_SIM

// Overlay configuration
// =====================

OverlayType = GNR
PeerType = AO_PEER       OverlaySeed = 123456
NumPeers = 1000
Degree = 10
NumChunks = 2500

// The next line needs some explanation. It is required
only if the selected OverlayType is a
// subclass of MBWOverlay. It defines a 2D array
// of doubles, using standard C-style initialiser syntax
(but without the terminating ';').
// The array specifies a set of different peer-bandwidth
classes for the MBWOverlay.
// Each class is specified by:
//    (fraction_of_peers_of_this_class,   upload_bandwidth,
download_bandwidth)
// A value "INF" denotes infinity. Bandwidths are in Mbps.

PeerBandwidths = {{0.1, 5, INF},{0.4, 1, INF},{0.4, 0.5,
INF},{0.1, 0, INF}}

// The bandwidth values are uniformly distributed with MEAN
given by class mean
// and the following (fractional) variance. Zero and
infinite bandwidths are unaffected by this.
BandwidthFractVariation =  0.10

SourceUploadBW = 5  // If not given, a randomised class-1
value is used.

StartStats = 0

// AOPeer configuration values -- irrelevant to everyone
not using AOPeer (and can all be deleted)
// ============================
ScanPeriod = 50
ReplenishMode = BA_REPLENISH
StatusReportInterval = 50
LowTidePercent = 10
```

```
HighTidePercent = 30
MinInLinks = 2
MinOutLinks = 1
ChurnPercent = 0
MinConnectTime = 100
MaxConnectTime = 1000
ChunkSchedDesirability="SQRT_OUTDEG"

// ================================================== //

// Base system with incentives sample configuration file
// With 0% of losses and 0% of churning peers

// Video source characteristics
ChunkSize = 0.1     // Mb
VideoRate = 1.0     // Mbps

// The PlayDelay sets the time for which each peer keeps
// chunks in its window
// Chunks older than this are discarded (and hence not
// diffused to neighbours)
PlayoutDelay = 5         // Seconds

// The maximum degree of TREE overlays
MaxTreeDegree = 5

// Chunks are lost with this probability.
ProbChunkError = 0.0

// Time-jitter to add to all events before enqueing
TimeJitter = 0.000001  // secs

// Number of chunk-delay counters to keep in each PeerStats
// record
ChunkDelayCounters = 1000

// Various output and input file names
// ===================================
LinksFile = "Links"
LinksAtEndFile = "LinksAtEnd"
DegreeFile = "Degree"
DegreeAtEndFile = "DegreeAtEnd"
OutOfSequenceFile = "Out_of_sequence"
ThroughputFile = "Throughouput_peer"
TotalChunkLossesPerPeerFile = "Total_chunk_losses_per_peer"
UploadRatePeerFile = "Upload_rate_peer"
UploadTotalPeerFile = "Upload_total_peer"
DegreeOverTimeFile = "DegreeOverTime"
DelayHistogramsFile = "DelayHistograms"
ChunkRequestsOverhead = "Chunk_overhead"
BMExchangeOverhead = "BM_overhead"
```

```
ChurningTimes = "Churn_times"
ConfigDumpFile = "ConfigDump"
ChunkArrivalsFile = "Absolute_time_arrival"
ChunkDelaysFile = "Chunk_delay"
FilenameSuffix = ".out"
FilenamePrefix = "output/"

// Network specification
// =====================
NetworkType = NONE
HostType = OLD_SIM

// Overlay configuration
// =====================

OverlayType = GNR
PeerType = AO_PEER      OverlaySeed = 123456
NumPeers = 1000
Degree = 10
NumChunks = 2500

// The next line needs some explanation. It is required
only if the selected OverlayType is a
// subclass of MBWOverlay. It defines a 2D array
// of doubles, using standard C-style initialiser syntax
(but without the terminating ';').
// The array specifies a set of different peer-bandwidth
classes for the MBWOverlay.
// Each class is specified by:
//    (fraction_of_peers_of_this_class,   upload_bandwidth,
download_bandwidth)
// A value "INF" denotes infinity. Bandwidths are in Mbps.

PeerBandwidths = {{0.1, 5, INF},{0.4, 1, INF},{0.4, 0.5,
INF},{0.1, 0, INF}}

// The bandwidth values are uniformly distributed with MEAN
given by class mean
// and the following (fractional) variance. Zero and
infinite bandwidths are unaffected by this.
BandwidthFractVariation =  0.10

SourceUploadBW = 5  // If not given, a randomised class-1
value is used.

StartStats = 0

// AOPeer configuration values -- irrelevant to everyone
not using AOPeer (and can all be deleted)
// ============================
ScanPeriod = 50
```

```
ReplenishMode = BA_REPLENISH
StatusReportInterval = 50
LowTidePercent = 10
HighTidePercent = 30
MinInLinks = 2
MinOutLinks = 1
ChurnPercent = 0
MinConnectTime = 100
MaxConnectTime = 1000
ChunkSchedDesirability="RECV_CHUNKS"

// ================================================== //

// MDC system sample configuration file
// With 0% of losses and 0% of churning peers

// Video source characteristics
ChunkSize = 0.025   // Mb
VideoRate = 1.0     // Mbps

NumDescs = 4

// The PlayDelay sets the time for which each peer keeps
// chunks in its window
// Chunks older than this are discarded (and hence not
// diffused to neighbours)
PlayoutDelay = 5         // Seconds

// The maximum degree of TREE overlays
MaxTreeDegree = 5

// Chunks are lost with this probability.
ProbChunkError = 0.0

// Time-jitter to add to all events before enqueing
TimeJitter = 0.000001  // secs

// Number of chunk-delay counters to keep in each PeerStats
// record
ChunkDelayCounters = 1000

// Various output and input file names
// ===================================
LinksFile = "Links"
LinksAtEndFile = "LinksAtEnd"
DegreeFile = "Degree"
DegreeAtEndFile = "DegreeAtEnd"
OutOfSequenceFile = "Out_of_sequence"
ThroughputFile = "Throughouput_peer"
TotalChunkLossesPerPeerFile = "Total_chunk_losses_per_peer"
UploadRatePeerFile = "Upload_rate_peer"
```

```
UploadTotalPeerFile = "Upload_total_peer"
DegreeOverTimeFile = "DegreeOverTime"
DelayHistogramsFile = "DelayHistograms"
ChunkRequestsOverhead = "Chunk_overhead"
BMExchangeOverhead = "BM_overhead"
ChurningTimes = "Churn_times"
ConfigDumpFile = "ConfigDump"
ChunkArrivalsFile = "Absolute_time_arrival"
ChunkDelaysFile = "Chunk_delay"
FilenameSuffix = ".out"
FilenamePrefix = "output/"

// Network specification
// =====================
NetworkType = NONE
HostType = OLD_SIM

// Overlay configuration
// =====================

OverlayType = GNR
PeerType = AO_PEER      OverlaySeed = 123456
NumPeers = 1000
Degree = 10
NumChunks = 2500

// The next line needs some explanation. It is required
only if the selected OverlayType is a
// subclass of MBWOverlay. It defines a 2D array
// of doubles, using standard C-style initialiser syntax
(but without the terminating ';').
// The array specifies a set of different peer-bandwidth
classes for the MBWOverlay.
// Each class is specified by:
//    (fraction_of_peers_of_this_class,   upload_bandwidth,
download_bandwidth)
// A value "INF" denotes infinity. Bandwidths are in Mbps.

PeerBandwidths = {{0.1, 5, INF},{0.4, 1, INF},{0.4, 0.5,
INF},{0.1, 0, INF}}

// The bandwidth values are uniformly distributed with MEAN
given by class mean
// and the following (fractional) variance. Zero and
infinite bandwidths are unaffected by this.
BandwidthFractVariation =  0.10

SourceUploadBW = 5  // If not given, a randomised class-1
value is used.

StartStats = 0
```

```
// AOPeer configuration values -- irrelevant to everyone
not using AOPeer (and can all be deleted)
// ============================
ScanPeriod = 50
ReplenishMode = BA_REPLENISH
StatusReportInterval = 50
LowTidePercent = 10
HighTidePercent = 30
MinInLinks = 2
MinOutLinks = 1
ChurnPercent = 0
MinConnectTime = 100
MaxConnectTime = 1000
ChunkSchedDesirability="SQRT_OUTDEG"

// ================================================== //

// MDC system with incentives sample configuration file
// With 0% of losses and 0% of churning peers

// Video source characteristics
ChunkSize = 0.025   // Mb
VideoRate = 1.0     // Mbps

NumDescs = 4

// The PlayDelay sets the time for which each peer keeps
// chunks in its window
// Chunks older than this are discarded (and hence not
// diffused to neighbours)
PlayoutDelay = 5         // Seconds

// The maximum degree of TREE overlays
MaxTreeDegree = 5

// Chunks are lost with this probability.
ProbChunkError = 0.0

// Time-jitter to add to all events before enqueing
TimeJitter = 0.000001  // secs

// Number of chunk-delay counters to keep in each PeerStats
// record
ChunkDelayCounters = 1000

// Various output and input file names
// ===================================
LinksFile = "Links"
LinksAtEndFile = "LinksAtEnd"
DegreeFile = "Degree"
```

```
DegreeAtEndFile = "DegreeAtEnd"
OutOfSequenceFile = "Out_of_sequence"
ThroughputFile = "Throughouput_peer"
TotalChunkLossesPerPeerFile = "Total_chunk_losses_per_peer"
UploadRatePeerFile = "Upload_rate_peer"
UploadTotalPeerFile = "Upload_total_peer"
DegreeOverTimeFile = "DegreeOverTime"
DelayHistogramsFile = "DelayHistograms"
ChunkRequestsOverhead = "Chunk_overhead"
BMExchangeOverhead = "BM_overhead"
ChurningTimes = "Churn_times"
ConfigDumpFile = "ConfigDump"
ChunkArrivalsFile = "Absolute_time_arrival"
ChunkDelaysFile = "Chunk_delay"
FilenameSuffix = ".out"
FilenamePrefix = "output/"

// Network specification
// =====================
NetworkType = NONE
HostType = OLD_SIM

// Overlay configuration
// =====================

OverlayType = GNR
PeerType = AO_PEER      OverlaySeed = 123456
NumPeers = 1000
Degree = 10
NumChunks = 2500

// The next line needs some explanation. It is required
only if the selected OverlayType is a
// subclass of MBWOverlay. It defines a 2D array
// of doubles, using standard C-style initialiser syntax
(but without the terminating ';').
// The array specifies a set of different peer-bandwidth
classes for the MBWOverlay.
// Each class is specified by:
//    (fraction_of_peers_of_this_class,   upload_bandwidth,
download_bandwidth)
// A value "INF" denotes infinity. Bandwidths are in Mbps.

PeerBandwidths = {{0.1, 5, INF},{0.4, 1, INF},{0.4, 0.5,
INF},{0.1, 0, INF}}

// The bandwidth values are uniformly distributed with MEAN
given by class mean
// and the following (fractional) variance. Zero and
infinite bandwidths are unaffected by this.
BandwidthFractVariation =  0.10
```

```
SourceUploadBW = 5  // If not given, a randomised class-1
value is used.

StartStats = 0

// AOPeer configuration values -- irrelevant to everyone
not using AOPeer (and can all be deleted)
// ============================
ScanPeriod = 50
ReplenishMode = BA_REPLENISH
StatusReportInterval = 50
LowTidePercent = 10
HighTidePercent = 30
MinInLinks = 2
MinOutLinks = 1
ChurnPercent = 0
MinConnectTime = 100
MaxConnectTime = 1000
ChunkSchedDesirability="RECV_CHUNKS"

// ================================================== //
```

ANNEX 2. MULTIPLE DESCRIPTION CODING

Within the context of the Trilogy project developed at the i2Cat foundation ([21]). a study of the different MDC splitting and merging techniques was done. In this annex the results obtained are provided.

First, the spatial MDC technique is evaluated for three different sample video streams that have high, medium and low degree of movement respectively. The graphs show the results of PSNR for different levels of losses as well as the behavior for different number of descriptors as it can be seen in Fig. A2.2, Fig. A2.3 and Fig. A2.4. The spatial approach processes each frame and distributes its pixels (according to a certain algorithm) amongst the different descriptions. Fig. A2.1 shows how the spatial technique distributes the pixels for 4 descriptors.

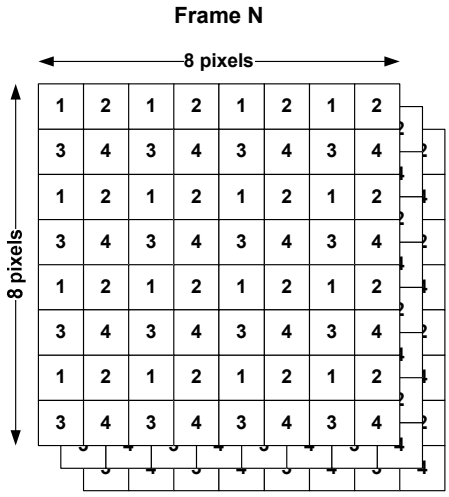

Fig. A2.1 MDC spatial technique

Fig. A2.2, Fig. A2.3 and Fig. A2.4 show that the number of descriptors does not affect drastically the resulting quality. The only difference between the three video traces is that the more movement the video has the lower the PSNR goes at 15%, though the variation is not very high.

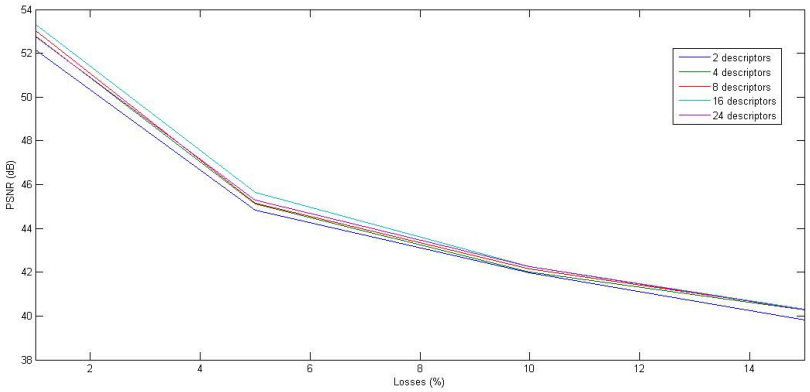

Fig. A2.2 High motion video - spatial technique

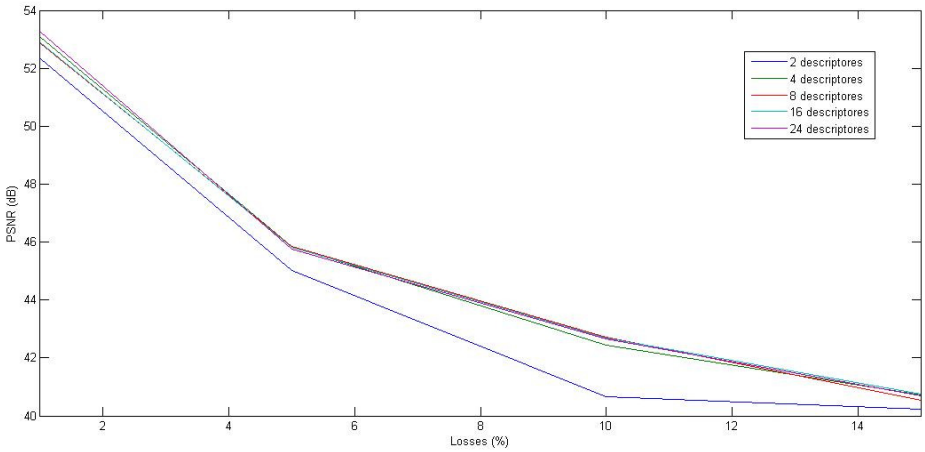

Fig. A2.3 Medium motion video - spatial technique

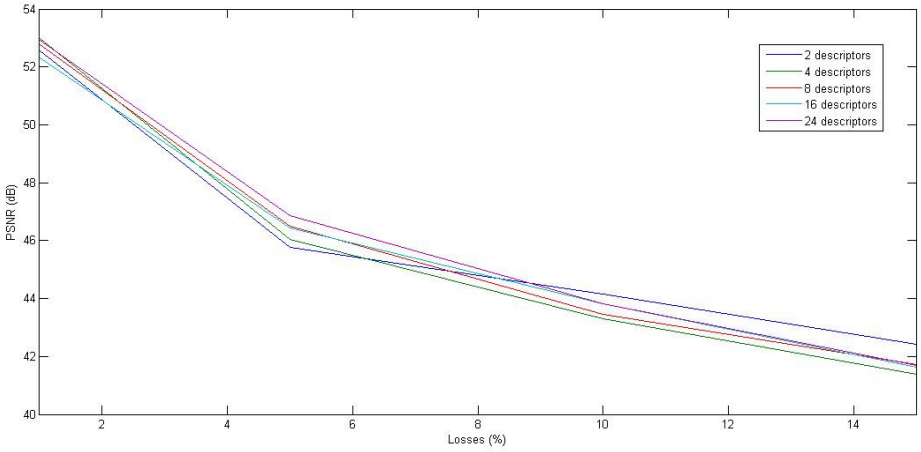

Fig. A2.4 Low motion video - spatial technique

For the temporal technique there is more variation between the results of the different simulations depending on the descriptors used. Specially for video traces with higher movement. This is due to the fact that with more movement the temporal approach has poorer results (the variation between frames is higher so the loss of a descriptors has a higher impact on the resulting PSNR).

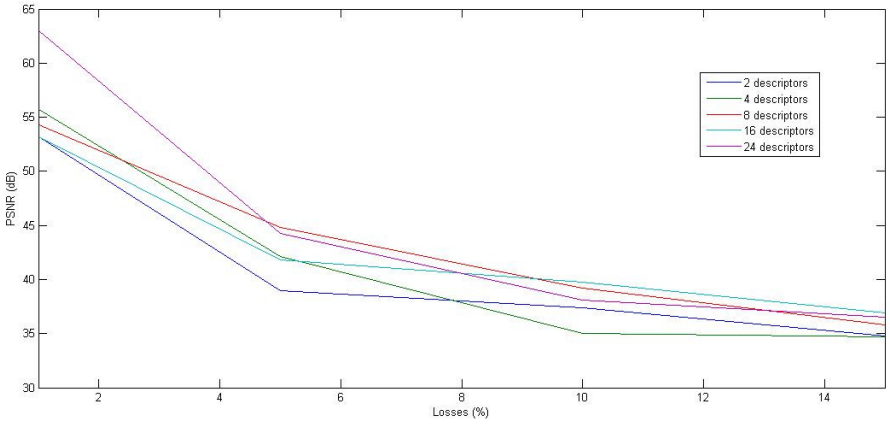

Fig. A2.5 High motion video - temporal technique

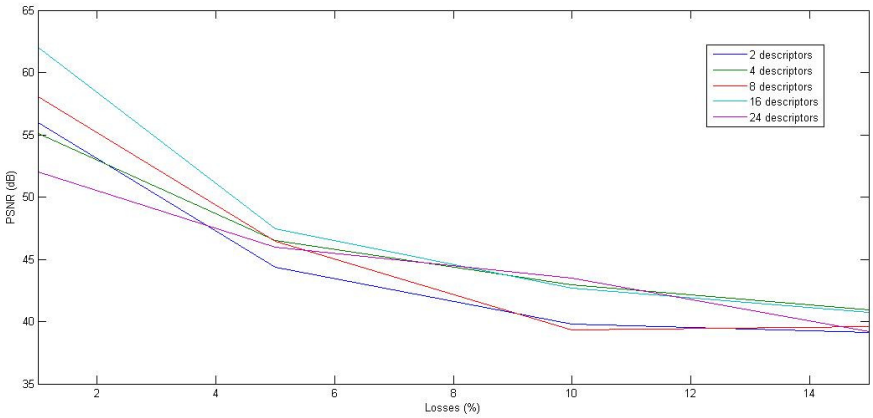

Fig. A2.6 Medium motion video - temporal technique

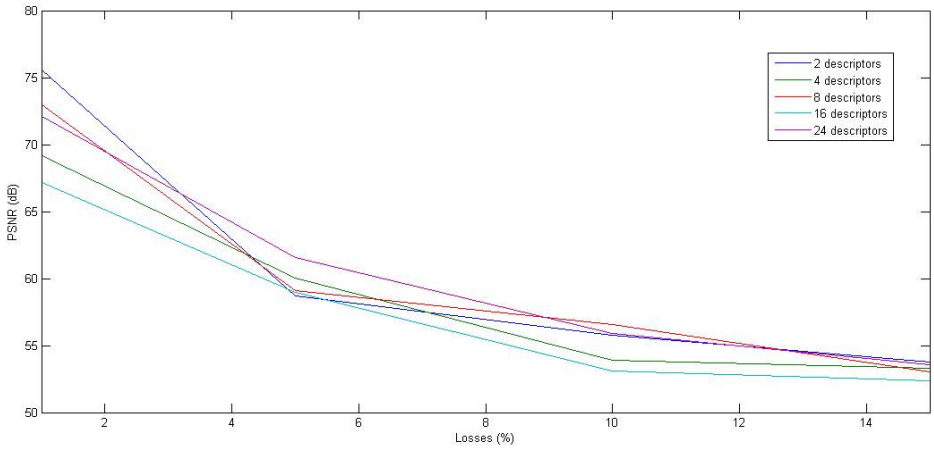

Fig. A2.7 Low motion video - temporal technique

The temporal approach consists in time interleaving where for a number of N descriptions description one will have frames 1, 1+N, 1+2N, etc. and so on for the other descriptions like it is shown in Fig. A2.8.

Fig. A2.8 MDC spatial technique

As it can be seen from the previous figures the temporal approach provides better results than the spatial technique when the video has low motion. On the other hand the spatial technique performs better than the temporal one when there is a lot of movement in the video stream.

The results presented here have been simulated using MATLAB for the splitting and merging processes and NS-2 simulator for the simulation of the transmission of chunks and the corresponding losses. The scenario is such that

each descriptor is sent over a different link (as Fig. A2.9 shows) so the results cannot be directly compared to those obtained in this project. They are useful, though, to provide more insight in the possibilities of the Multiple Description Coding technique.

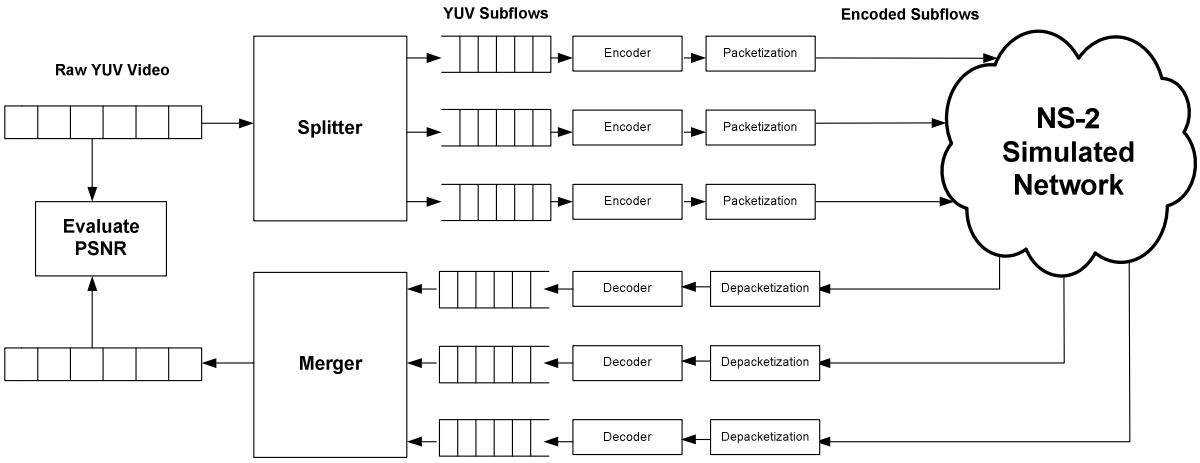

Fig. A2.9 MDC simulations scheme

ANNEX 3. PAPER ON MDC WITH INCENTIVES

As a result of the work done in this thesis, a publication has been written and will be presented to some conferences that discuss these topics. More specifically the call for papers (CFP) that have been considered are: IEEE CCNC 2010 IEEE CCNC 2010 and IEEE SECON 2010.

Using Multiple Description Coding with Incentives for Redistribution in P2PTV Streaming Systems

***Abstract*—With the widespread adoption of broadband residential access, several forms of IPTV have emerged as the new popular Internet applications. But amongst the different possible architectures for the deployment of IPTV applications, P2PTV streaming systems stand out. More specifically, mesh-pull based P2P systems are the ones that have been deployed with more success. Despite these systems address scalability in an efficient manner they still present several limitations that difficult them to offer the same user experience that traditional TV broadcasting offers. These limitations are mainly the free-riding (non-cooperation) effect, the long start-up delays and the impact of churn and bandwidth heterogeneity in the stability of the system. In this paper we study the performance of Multiple Description Coding (MDC) combined with the use of incentives for redistribution in order to solve some of these problems. Several simulations are performed, both for a reference mesh-pull P2PTV system and for the proposed alternative systems. The comparison of the results shows that the use of MDC and the use of incentive-based scheduling strategies improve the overall performance of the systems in terms of Continuity Index (CI) and chunk distribution delay.**

***Keywords*— P2PTV, MDC, Incentives, Continuity Index (CI), Delay, Simulations.**

I. INTRODUCTION

P2PTV streaming systems have become a popular service on the Internet, with several successful deployments ([1], [2]), and the most spread form of what is known as Internet TV, which is the delivery of digital TV content over the public Internet infrastructure via streaming or application level multicast.

The use of these systems is interesting because it can introduce added value to traditional TV by providing flexibility, in terms of the contents offer (video-on-demand as well as real-time contents), and interactive services. But in order to become a truly successful application it needs to be able to provide the same user experience TV broadcasting offers. To achieve this goal the main limitations of P2PTV systems have to be solved. These limitations we are referring to are mainly the free-riding (non-cooperation) effect, the long start-up delays and the impact of churn and bandwidth heterogeneity in the stability of the system.

In this paper we propose a MDC based system, which uses incentives for redistribution, in order to address the impact of losses in the Continuity Index (CI) and the delay as well as the performance problems due to the effect of free-riding. The results show how the use of Multiple Description Coding increases the Continuity Index significantly and how the use of incentives compensates for the impact of free-riders.

The paper is organized as follows. In section II we provide an overview of the relevant related work in order to understand the references considered. In section III we talk about the P2PTV challenges and limitations to give some insight into the motivation and objective of this work. In section IV we describe the proposed solution by explaining the algorithms and schemes used for the design of the new P2PTV system. In section V the results of the performed simulations are shown and discussed and finally in section VI the corresponding conclusions are presented as well as the future work that can be derived from this paper.

II. Related Work

The amount of research done in the field of P2PTV streaming systems is considerable. Several approaches have been studied for the design of these systems and both simulations and real-environment measures have been performed for the different scenarios considered. As mentioned before, this paper is focused on the use of MDC combined with incentives for redistribution and the goal is to perform the corresponding simulations in order to validate these techniques and discuss their performance.

Multiple Description Coding (MDC) is a technique designed to enhance error resilience and increase transmission robustness without increasing the delay. It consists on the processing of the video before encoding at the source and after decoding at each receiver. The result of this processing at the source is a set of several independent descriptions that hold different information of the video (information can be distributed in different ways amongst the descriptions: temporal, spatial, etc.) and these description may have the same importance (balanced scheme) or different importance (unbalanced scheme). The more descriptions received, the higher the quality of decoded video. The results obtained in [3] and [4] show that using MDC the delivered quality is acceptable even at high loss rates.

For the evaluation of the techniques studied in this paper we chose to perform a set of simulations instead of carrying out a costly deployment and testing the application in a controlled test-bed. For this purpose, we need a simulator that suits well our requirements and, though the development of a simulator from scratch could be an option, here we have decided on the use of an already developed and tested simulator. The choice has been P2PTVSim [5] which is an event-driven simulator specifically designed for P2PTV systems, that simulates the flow of chunks through a P2P streaming overlay. This simulator has been validated in [6] and [7]. Other simulators have been considered, but according to the comparison provided in [26] and the characteristics of the simulators P2PTVSim has been the selected one because it adjusts to the requirements of this paper.

III. P2PTV Challenges

The design of P2PTV streaming systems has several additional challenges to those seen in P2P file sharing applications. These applications must ensure an acceptable degree of quality for the received media while being able to manage a large number of nodes that present a highly dynamic behavior. Also, it is important to point out that delivered data has to meet deadlines (related with the playback of the media) in order to ensure proper playback continuity in real-time.

All this requirements derive in the need to optimize the following metrics: start-up delay (i.e. the delay from when the user selects a content and when the content starts playing), end-to-end delay (i.e. the delay between the content from the source node and the content received at the clients) and the playback continuity, usually measured as the Continuity Index (i.e. the percentage of received data). But this has to be achieved in a best-effort environment like the Internet where nodes have different bandwidth constraints.

It is important to understand the effect of the main P2PTV systems' limitations to design a solution accordingly. In [1], results of the impact of bandwidth heterogeneity for mesh-based and tree-based systems are provided. These results show that the throughput does not vary according to the amount of nodes with lower bandwidth but the useful data (goodput) does. Goodput decreases as the number of bandwidth constrained nodes increases. Both in tree-based and in mesh-based systems bandwidth-constrained peers become overwhelmed and perform a poor relay of data to the other peers. Mesh-based systems mitigate the impact of bandwidth-constrained nodes on high bandwidth nodes. But mesh-based systems penalize low bandwidth nodes since they receive very little of the stream.

As theoretically demonstrated in [17], appropriate buffering can significantly improve video streaming quality. However, too much buffering may make the delay performance unacceptable for a streaming service. Start-up delay is the interval from when a channel is selected by the user until actual playback starts on the screen. And while short start-up delay is desirable, certain amount of start-up delay is necessary for continuous playback.

In P2P systems the behavior of each peer in terms of entering or leaving the overlay is neither coordinated nor controlled. The joins and departures of the nodes occur at arbitrary times and these dynamics of peer participation are known as churn and are an inherent property of P2P systems. Churn significantly affects both the design and evaluation of P2P systems. It is important to understand that peers leaving the system during a given interval of time can adversely affect the performance of the system, as some nodes may find themselves disconnected or experience temporary service interruption. According to the results shown in [1] the system performance under conditions of high churn is better for mesh-based systems as they always have better continuity index.

P2P systems rely on voluntary resource contributions by individual peers. The impact of non-cooperation in this type of systems according to [7] is that the level of cooperation affects the overall quality of the system. Higher cooperation means better quality of service (QoS). If the level of cooperation is low, the quality of the media is also low even when the network is not congested.

IV. Proposed Solution

The proposed solution for addressing the issues discussed in the previous section is presented here. Our system is a mesh-pull P2P streaming system that uses Multiple Description Coding with a balanced scheme (all the descriptions have the same importance). It also introduces the use of incentives based on the contribution of the partners in the supplier-side scheduler.

In mesh-pull systems, the video is divided into chunks that are introduced into the overlay by the source and then shared between the different nodes. Each peer exchanges information on the availability of the chunks and then they share chunks in order to retrieve the video stream within the playback deadline constraints. When the system uses Multiple Description Coding, video is pre-processed before encoding and several descriptions are generated and then encoded and distributed in chunks like regular mesh-pull systems do with a single stream. In this case peers exchange the chunks obtained from each substream (description) and the more chunks they receive from different description the better the received quality will be. Then in order to encourage cooperation the incentive-based mechanism is used, so that peers contributing more to their partners are more likely to receive more descriptions and therefore more quality.

A. Chunk request scheduler

Peers in the overlay request chunks from their partners according to their availability, playback deadlines and other constraints discussed in this section. Each node has a sliding window of chunks that is represented through a buffer map to indicate which chunks are already buffered, which ones have been requested and also the chunks that have not yet been requested. Partners exchange their buffer maps periodically and they perform rounds of chunk requests to get the missing chunks in the sliding window considering the availability information provided by their neighbors. The schedule of these requests is critical to achieve an optimal result and retrieve the maximum number of descriptions and ensure the best Continuity Index as well as the best quality.

In Fig. 10 an example of an MDC buffer map is shown along with the representation of the chunk requests scheduling order for 17 rounds. Chunks that have already been received are marked with a B and chunk that have been requested with R. The numbers indicate the order of the requests. The scheduler considered here looks, at the beginning of each round, for the first chunk index that does not have a buffered or requested chunk from any description. When it finds the first chunk index satisfying this condition it selects a random description (from the available ones according to the availability information provided by its neighbors) and makes the request. The it continues with the next chunk index that has none buffered or requested chunk until there are no more chunk indexes fulfilling this condition. The scheduling algorithm continues by doing the same routine over chunk indexes with just one buffered or requested chunk, then with just two, and so on.

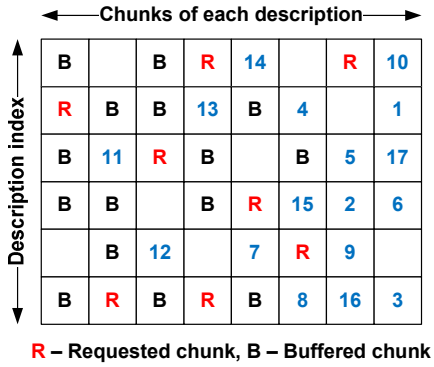

Fig. 10 Example of buffer map for the MDC system

The goal of this scheduling algorithm is to get the maximum number of descriptions for each chunk index time but always trying to avoid having high variation between the received number of descriptions from one time to another.

B. Supplier-side incentive-based scheduler

As stated earlier the design considers an incentive-based strategy for the supplier-side that takes into account the contribution of their partners and serves them accordingly. The effect of this supplier strategy is that peers that contribute more are more likely to receive a larger number of descriptions and thus better quality. As a supplier, each peer has a queue of requests from its partners and at each round it has to decide which of these requests is going to be served first. Instead of selecting a random one or handle them in a first-in-first-out manner, a weighted selection is performed. The weight in this case is assigned by computing the percentage of chunks that have been provided by a specific neighbor from the total. Then the

selection of the neighbor to be served is done using the weights for each one.

V. RESULTS

In the simulations performed for this paper four different scenarios have been considered. The first scenario is the reference system which is a mesh-pull P2P streaming system with single layered video and no incentives. The simulations for this scenario allow us to compare the results for our proposed system with results obtained in the exact same conditions. Then a second scenario, which is like the reference one but with the use of incentives for redistribution. The third scenario is an MDC based mesh-pull P2P streaming system and the last scenario is a variation of the MDC, adding incentives for redistribution.

The main parameters for the simulations, that are common for the four different scenarios are the following: a total of 1000 peers, with a mean degree of 10 partners and a 5 second buffer. Four different types of peers: class A, peers with 5Mbps of upload bandwidth (10% of the total); class B, peers with 1Mbps of upload bandwidth (40% of the total); class C, peers with 500Kbps of upload bandwidth (40% of the total); and class D, peers that act as free-riders, with 0Kbps of upload bandwidth (10% of the total). No download bandwidth constraint is assumed according to [5] and [6]. The simulations last for the distribution of 2500 chunks.

For each scenario several simulations have been performed for different loss values (0%, 1%, 5%, 10%, 15%, 20% and 40%) and each of these simulations has been done 5 times and their results have been averaged.

A. Reference system results

When considering the effect of losses in our reference system we can see that the Continuity Index (CI), measured as the number of received chunks for the total number of chunks, decreases as the losses increase. It is important to point out that the degree of 10 partners and the 5 second buffer have an impact in the results and with different (more restrictive) values for these parameters the results would have been worst in terms of CI. In Fig. 11 the results for the simulations performed for different values of losses are shown.

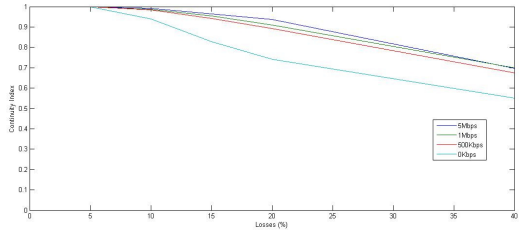

Fig. 11 Reference system – CI vs. losses

The delay, as it can be seen in Fig. 12, increases as the percentage of losses increases and it is in the range of 1,5 to 3,5 seconds which is coherent considering that this delay is the end-to-end delay measured only for the useful received chunks (within the 5 seconds sliding window). Each line has the average values of delay for the peers of the different classes at each loss percentage.

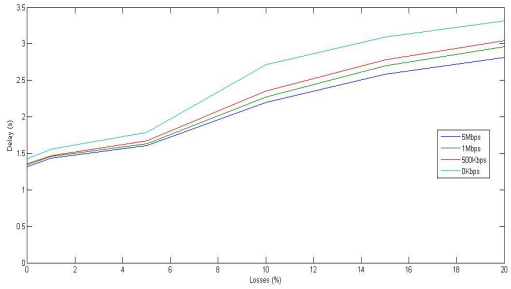

Fig. 12 Reference system – Delay vs. losses

The results of the end-to-end delay are also shown in Fig. 13 but his time depicting the delay for each peer, for three different loss rates (0%, 5% and 10%) instead of showing the average. This result allows us to see the variance of the delay values.

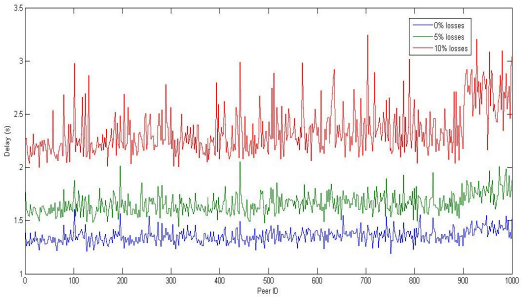

Fig. 13 Reference system – Delay for 0%, 5% and 10% losses

B. Reference system with incentives

When the use of incentives is introduced as it has been explained in previous sections the overall performance of the system is expected to improve. In Fig. 14 we can see how the CI of the cooperating peers has increased significantly while the free-riders are aggressively punished having their CI importantly reduced.

Even at high loss rates like 20%, the CI keeps above 0,9 for the cooperating peers and at the 40% loss rate class A peers still maintain the CI over 0,9 while class B and C peers drop to 0,8 and 0,65 approximately. Then evaluating the delay results when using incentives (Fig. 15) we found that there is also a significant improvement for the cooperating peers, that have a lower delay (being reduced by 1 second approximately). The free-riders delay behavior as depicted in Fig. 15, does not effectively drop from the 10% loss rate, but it does

drop in the picture because the delay is measured only for the useful received chunks and as it can be seen in Fig. 14 free-riders do not have that many chunks from that point. So the average decreases as the late received chunks are not included.

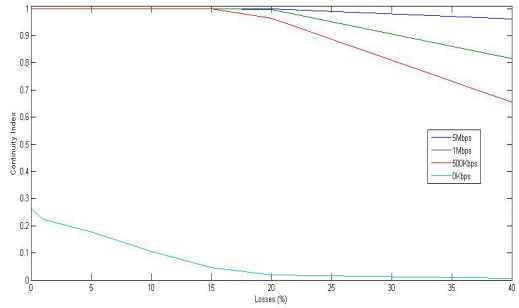

Fig. 14 Incentive-based system – CI vs. Losses

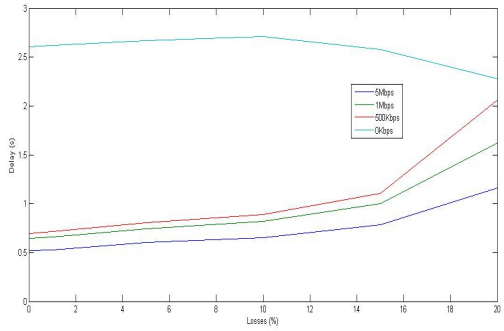

Fig. 15 Incentive-based system – Delay vs. Losses

The results show that cooperating peers benefit from the incentive-based mechanism. That is because the bandwidth that these peers were wasting serving free-riders is now available and used for these cooperating peers, increasing their overall performance.

C. *MDC system*

In order to have a more robust behavior against losses the Multiple Description Coding scheme is introduced and the proposed system has the same characteristics as the reference one but using multiple layers and the chunk request scheduling strategy explained in section IV (no incentives for redistribution are used). Here the results for the simulations performed with the MDC system are presented. First, in Fig. 16, we can see the CI achieved using the MDC system. As it can be seen the continuity increases significantly and it only drops to 0,9 approximately when the loss rate is 40%.

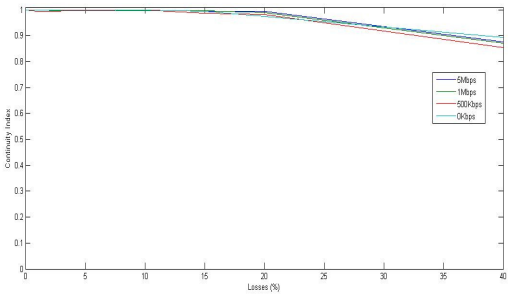

Fig. 16 MDC system – CI vs. Losses

This result shows how the use of MDC can improve one of the main metrics that we want to optimize in a P2P streaming system: the continuity playback or CI. But it is important to point out that this measure only indicates the level of continuity (being almost the same for the different peer classes) but does not indicate the received quality. This quality depends on the number of descriptions as well as on the characteristics of the MDC techniques used ([14]), and to show the difference between the four peer classes the average number of received descriptions is depicted in Fig. 10. Here we can see that class A peers receive in average a higher number of descriptions than the rest of the classes.

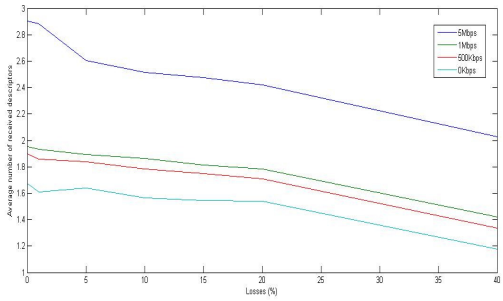

Fig. 17 MDC system – Avg. Number of descriptions vs. Losses

Also as it can be seen in Fig. 18 the delay decreases drastically. This is due to the fact that the delay is measured for individual chunks, that are smaller in the MDC system (their size in relation to the size of the single layered system chunks can be considered proportional to the number of descriptions used). For that reason the delay here has to be understood in a different context than the delay for the single layered system. Here it is interesting to take into account these low delay values for MDC end-to-end chunk delivery because in systems where there are low-delay requirements MDC could be used.

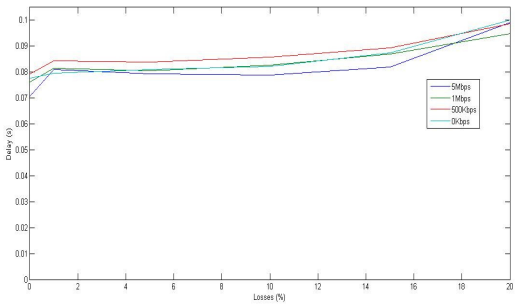

Fig. 18 MDC system – Delay vs. Losses

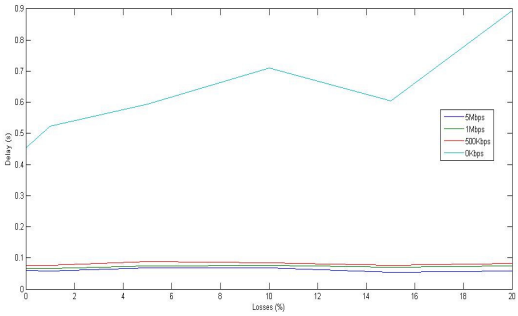

Fig. 21 MDC system with incentives – Delay vs. Losses

D. *MDC system with incentives*

The last step is to combine the use of MDC with incentives for redistribution and the system proposed in this section is the one explained in section IV. Here to the improvements introduced by MDC in terms of CI increase and low end-to-end delay we can add the overall performance boost introduced by the use of incentives. As it is shown in Fig. 19 the CI can be maintained at almost the maximum level even at high loss rates like 40%. This combination allows a high level of continuity playback. The quality, in terms of average number of descriptions is also increase for the cooperating peers as it can be seen in Fig. 20 and the only metric that is not significantly enhanced in this system is the delay (Fig. 21) that is approximately the same delay that the MDC system showed.

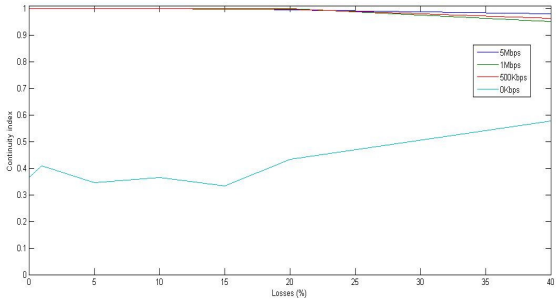

Fig. 19 MDC system with incentives – CI vs. Losses

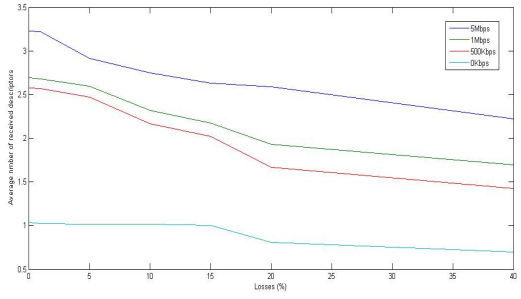

Fig. 20 MDC system with incentives – Avg. Number of description vs. Losses

VI. Conclusions and Future Work

The aim of this work was to study the performance of MDC systems with incentive-based mechanisms for redistribution. The results show that the proposed solution clearly improves the considered metrics and the overall behavior of the system.

In order to provide a more comprehensive validation of this system other measurements and simulations should be performed to complement the results of this work. More specifically the measure of the overhead introduced by the use of MDC should be considered in order to see if it is worth to deploy this technique. Also PSNR quality measures could be performed by selecting a specific MDC technique and running the corresponding simulations. To complete these results, the effect of churn, concretely the impact of flash-crowds, could be evaluated in order to see if the proposed solution performs satisfactorily under these conditions.

With the obtained results, another line of work that could be followed is the use of MDC for systems or applications with low-delay requirements and study the corresponding approaches to it.

References

[1] X. Zhang, J. Liu, B. Li and P. Yum, "DONet: A data-driven overlay network for efficient live media streaming", in Proc. of IEEE INFOCOM, 2005.

[2] X. Hei, C. Liang, Y. Liu, and K. W. Ross, "Insights into PPLive: A measurement study of a large-scale P2P IPTV system," in Workshop on Internet Protocol TV (IPTV) services over World Wide Web, Edinburgh, Scotland, May 2006.

[3] Chakareski, J., Han, S. and Girod, B., "Layered coding vs. multiple descriptions for video streaming over multiple paths". Springer Multimedia Syst. i10.

[4] Soldani C., Leduc G., Verdicchio F. and Munteanu A., "Multiple Description Coding versus Transport Layer FEC for Resilient Video Transmission" in Digital Telecommunications, ICDT '06. International Conference, Cote d'Azur, Aug. 2006

[5] http://www.napa-wine.eu/cgi-bin/twiki/view/Public/P2PTVSim

[6] A.P Couto da Silva, E Leonardi, M. Mellia, M. Meo, "A Bandwidth-Aware Scheduling Strategy for P2P-TV Systems"
8th International Conference on Peer-to-Peer Computing 2008 (P2P'08), Aachen.

[7] R. Lobb, A. P. Couto da Silva, E. Leonardi, M. Mellia, M. Meo,
"Adaptive Overlay Topology for Mesh-Based P2P-TV Systems"
Proceedings of ACM NOSSDAV 2009.

[8] S. Naicken, A. Basu, B. Livingston, and S. Rodhetbhai, "A Survey of Peer-to-Peer Network Simulators," Proceedings of The Seventh Annual Postgraduate Symposium, Liverpool, UK, (2006).

[9] Seibert, J., Zage, D., Fahmy, S. and Nita-Rotaru, C., "Experimental Comparison of Peer-to-Peer Streaming Overlays: An Application Perspective", Technical Report CSDTR-07-020, Department of Computer Science, Purdue University, 2007.

[10] R. Kumar, Y. Liu, and K. W. Ross, "Stochastic fluid theory for P2P streaming systems", in Proceedings of INFOCOM, 2007.

[11] Habib, A. and Chuang, J., "Incentive Mechanism for Peer-to-Peer Media Streaming", UCLA, California 2004.

[12] Liu, Z., Shen, Y., S.Panwar, S., W.Ross, K. and Wang, Y., "Using Layered Video to Provide Incentives in P2P Live Streaming", Polytechnic University, Brooklyn, 2007.

[13] Liu, Z., Shen, Y., S.Panwar, S., W.Ross, K. and Wang, Y., "P2P Video Live Streaming with MDC: Providing Incentives for Redistribution", Polytechnic University, Brooklyn 2007.

[14] Tillier C., Olivier C., Béatrice P. and Christine G., "Comparison of four Video Multiple Description Coding Schemes", EUSIPCO, Poznan, 2007.