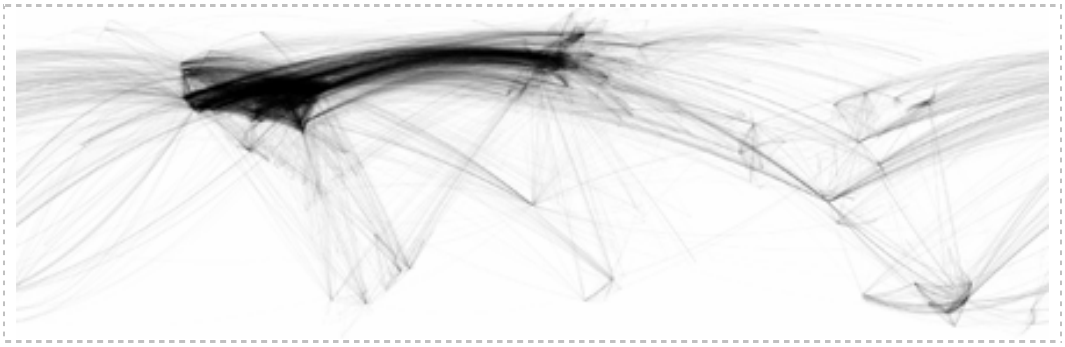
INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa



Content Distribution Networks

Integration with IPTV and Study Regarding Network Coding

Rui Dinis Mangueira Bento

EXTENDED ABSTRACT

Dissertation for the Obtainment of the Masters Degree in

Engenharia Informática e de Computadores

(Computer Systems and Engineering)

Comittee

President:	Prof. Luís Eduardo Teixeira Rodrigues
Coordinator:	Prof. Rodrigo Seromenho Miragaia Rodrigues
Thesis Committee Member:	Prof.ª Teresa Maria Sá Ferreira Vazão

Setembro, 2007

INDEX

INDEX ..... **II**

1. INTRODUCTION **1**

2. CONTEXT AND RELATED WORK **1**

3. STUDY REGARDING THE APPLICATION OF A CENTRALIZED CDN **3**

3.1 DESCRIPTION OF MOBBIT INSIGHT 3
3.2 DESCRIPTION OF CISCO ACNS 4
3.3 INTEGRATION PLANNING 5

4. STUDY REGARDING THE APPLICATION OF NETWORK CODING TO PEER-TO-PEER CDNS **6**

4.1 BITTORRENT 6
4.2 NETWORK CODING 6
4.3 APPLICATION MODELS FOR THE SIMULATORS 7
4.4 SIMULATION RESULTS AND ANALYSIS 8

5. CONCLUSIONS **10**

1. INTRODUCTION

Over the last couple of decades, the Internet has established itself as the main global vehicle for information sharing. Today, about 18% of the world population, over 1.1 billion people, actively use the Internet, a figure that is doubling every year. However, although the time to load contents off of key Internet sites has improved constantly throughout the last several years, the latency to access data on the World Wide Web remains generally high due to its skyrocketing utilization and growth of bandwidth intensive contents, affecting its users' perception of the quality of the service provided. On the other hand, content is often still served from only one server, which determines a single point of failure is not scalable and may present serious problems pertaining reliability and performance.

The approach suggested by Content Distribution Networks (CDNs) to address this problem is to move the content as close as possible to the end-users. By serving contents from local replicas, the perceived access latency is typically lowered, while the transfer rates are increased. Additionally, the loads on the original servers is decreased, yielding reduced infra-structure requirements. Generally speaking, the global network traffic is also reduced. The purpose of the CDNs is therefore to distribute specific forms of content, from static web pages to real-time multimedia streams, which usually must be up-do-date, in a quick manner and with high availability.

Although the term "CDN" usually refers to World Wide Web content distribution networks, it can also be interpreted in a broader sense, encompassing the notion of peer-to-peer distribution networks. These networks leverage on its users' bandwidth and storage capabilities to replicate the distributed contents as much as possible, increasing availability, reliability and transfer rates. It is estimated that about 80% of the overall traffic over a backbone IP connection is due to peer-to-peer applications. Of these 80%, roughly half is due to BitTorrent, a popular file-sharing application that, unlike most other alternatives, focuses on increasing the efficiency of content distribution instead of only finding the content. Recently, approaches such as Network Coding have been suggested to improve the efficiency of current file-sharing systems such as BitTorrent. Network Coding is a mechanism designed to maximize the amount of data that can be pushed through a network between two nodes. The core notion is to allow mixing of data at intermediate network nodes (coding). A receiver then deduces the original content from the coded blocks received.

This dissertation has CDNs as its central subject of discussion. It comprises a chapter regarding how CDNs operate and which are the main challenges to be addressed. Additionally, it comprises two case studies: the practical application of a centralized CDN to a working system and the study of the effects of applying Network Coding as a strategy to optimize the dissemination of contents over peer-to-peer CDNs such as BitTorrent.

The first study consists on the application of a CDN, developed Cisco Systems, to a centralized commercial Corporate TV system, managed by Mobbit Systems. The used of the CDN avoids a situation where the same content has to be transferred through the same link more than once, inefficiently using the available bandwidth resources.

After deploying this system, it was realized that the design of the CDN was substantially simplified by the existence of a relatively static and predefined distribution tree. Hence, in the second part of the thesis, some of the problems that arise in more dynamic deployments were studied, in particular, the impact of Network Coding as a form of increasing the efficiency content distribution.

2. CONTEXT AND RELATED WORK

Although details underlying the operation of CDNs may vary according to different implementations, it is possible to identify a number of common challenges that must be addressed, namely: where to place the replicas, when to replicate contents, how to guarantee that all replicas are consistent and how to forward client requests.

Replica placemen deals with the number of replicas each object has and where these should be placed in the context of a network. These issues refer to two distinct problems: the replica-server problem and the replica-object problem. The first deals with the selection of network locales where – server-replicas (servers that contain replicas of the objects being distributed) should be placed, while the latter pertains the choice of the server-replicas where the replicas of each object should be placed. Intuitively, the replica-servers should be placed as near to the end-users as possible. There are several theoretical approaches used to model this problem but, due to its computational complexity, heuristic approaches are used. On the other hand, there are several mechanisms to choose which replica-servers should host the replicas of each object. These can be classified according to the direction of the choice: from the replica-servers to the original servers (pull-based caching) or the other way around (push replication).

One of the main goals of CDNs is to reply to requests with contents as up-to-date as possible. Maintaining consistency among replicas of a certain object can introduce a significant overhead to the system, especially when the consistent policy is strong (clients are intolerant to non-fresh contents). Therefore, it is important to determine which consistency models, consistency policies and content distribution methods according to the consistency requirements. A consistency model defines the consistency properties of the objects being distributed, such as its time to live or the number of operations that can be executed over it. A consistency model is usually adopted by a consistency policy, which defines how, when and which distribution mechanisms should be applied. The latter specify the protocols through which the replica-servers exchange object updates.

The request forwarding system deals with the selection of the most adequate replica-server to serve a certain request from an end-user. The system forwards the users' requests to the closest available replica-servers that are likely to be hosting the requested content. Therefore, is necessary to address a number of issues: determining the distance between requesting users and replica-servers, determining the availability of a certain replica-server and determining how likely a replica-server is to host a replica of a certain object. In order to determine the distance between users and servers, usually methods such as the number of hops (number of nodes through each a data packet goes through from its origin to its destination) or the round-trip time are used. None of these methods is exact. The availability of a replica-server depends on the amount of requests it is dealing with in a certain point in time. Servers can be proactive in periodically stating its availability or might be probed by clients. How likely a replica-server is to host a replica of a content depends on a number of policies underlying the CDNs operations, namely the type of contents being served, its update rate, the storage resources of the servers, amongst others. Several methods can be used to forward a request to a server: HTTP redirection, DNS indirection, anycasting and peer-to-peer forwarding.

3. STUDY REGARDING THE APPLICATION OF A CENTRALIZED CDN

The purpose of this section is to describe and justify the steps taken to integrate Mobbit InSight, a Corporate TV system developed by Mobbit Systems, with Cisco ACNS, a private and centralized CDN developed by Cisco Systems. To properly address the integration of the two systems, it is important to understand how they operate. Therefore, a brief description of both systems will be provided, stressing the relevant aspects for the integration.

3.1 Description of Mobbit InSight

Mobbit Systems' InSight aims at offering a Corporate TV service to a number of clients. In brief, the purpose of a Corporate TV system is to broadcast stored multimedia contents, arranged in lineups, through a network of screens (terminals) with the desired geographic dispersion. All of the management tasks, such as content handling, its assortment in lineups and terminal and client administration, are done centrally. Therefore, little processing is done at the terminal level, with each broadcasting the correct contents, following the lineup restrictions.

This architecture of the system comprises four distinct types of components: the backoffice, the database, the gateway and the terminals. The backoffice is where the management tasks are done. The database stores all the system configuration data persistently. The gateway ensures communication between the backoffice and the systems' terminals. The terminals receive and display the contents associated with each lineup on one or more linked screens.

In order to define how the broadcast management is ensured, it is fundamental to introduce three important concepts: channel, lineup and playlist. A channel consists of an overlay network, established between the central server and a number of clients. Tied in with the channel are the contents to be displayed, as well as the lineups that determine the sequences and schedules in which these should be displayed. Therefore, all of the terminals belonging to a certain channel display the same contents, following the same sequence of contents. Although this kind of behaviour might indicate that contents are multicast, InSight uses unicast exclusively, sending the content to each terminal separately. A lineup, borrowing the term from traditional television, is a sequence of contents bearing several kinds of constraints such as time schedule, loop or order restrictions. A playlist is the compilation of all the information that a terminal requires to correctly perform its function. There is exactly one playlist per channel, which is generated by the gateway using information from the clients, contents and lineups.

All the required information for the system to perform correctly is recorded in files according to the XML standard. There are several different types of files necessary to ensure the correct broadcast and display of contents, of which we will solely focus on the playlist. Each channel has a file named **playlist_X.xml** linked to it, in which X stands for the alphanumeric ID of the channel. This file incorporates all of the information that the terminal needs to function properly, such as which contents to display and where should those contents be fetched from (usually the path to the files stored in the central server).

3.2 Description of Cisco ACNS

Cisco ACNS is a CDN designed by Cisco Systems that operates by placing the contents to be distributed as close to the end-users as possible. All content distribution management is centralized, as well as the definition of distribution channels and replica positioning.

The Cisco ACNS network comprises two relevant components to this project: the Content Distribution Manager (CDM) and the Content Engines (CE). The main role of the CDM is to centrally manage all the aspects of the ACNS network, such as the devices that are in the network, the contents being distributed, how, where and when these contents are fetched, the resources allocated to the distribution process, amongst many others. ACNS channels are independent sub-networks used to distribute contents, which are assigned a number of ACNS devices as well as policies and constraints for the dissemination of information. The CE serves requests for content issued by the users of the network. The allocation of CEs to channels and which contents each CE should store is defined at the CDM. The Cisco ACNS system also comprises Content Routers (CR), which we do not describe sunce they are not relevant to this project.

Three important concepts must also be defined: root content engine (RCE), edge content engine (ECE) and manifest file. An RCE is a subtype of a CE that is assigned to obtain the contents being broadcast through a specific channel on their original servers. There exactly one RCE per channel. An ECE is a CE that is placed at the nearest location from an end-user and is supposed to serve its request. A channel can, therefore, be composed by several CEs, organized in a tree structure that can comprise two or more levels: the RCE is the root of the tree, the ECEs are the leaves and other CEs compose the interior nodes. Finally, a manifest file is an XML file that consists of a declaration (explicit or through crawling rules) of objects to be disseminated over a channel. Thus, it is possible to associate a manifest file to each channel.

The Cisco ACNS system comprises three fundamental forms of obtaining contents: demand-pull, pre-positioning and live content. As the contents being distributed in the Mobbit InSight system are static, the live content approach is inadequate. Furthermore, since contents being distributed are previously known and defined at the CDM, it is possible to use a pre-positioning approach.

Focusing on the pre-positioning approach, the contents that will populate the different CEs can be defined explicitly or using crawling tasks either using a Manifest File or using the graphic interface of the system. Despite being easy to specify the contents being distributed using its graphical interface, it requires human intervention whenever a change has to be made, rendering this approach unusable to the project. On the other hand, it is possible do dynamically generate manifest files for each ACNS channel based on the playlists created for each InSight channel.

ACNS comprises three fundamental forms of content obtainment: WCCP, direct proxy routing and distributed file system. The first refers to transparently forwarding the users' requests to the ECE using RCs. The second implies that the clients should explicitly use a CE as a proxy to obtain the contents through the ACNS network. The latter involves obtaining contents from a distributed file system giving the illusion that these are being obtained locally. As the first system implies an overhead in devices added to the network and the second one requires the obtainment of contents via-HTTP (which may arise access control issues), the third option has been adopted.

3.3 Integration Planning

After an analysis of both systems and their points of integration, one can formulate a solution that will both meet the projected goals, incurring in the lightest technical and economical impact possible since this was a commercial system already being used in several private and public institutions. Therefore:

1. To the InSight network, composed by a central server and a number of terminals, will be added the following Cisco ACNS components:
    a. A CDM, which has the purpose of managing the ACNS network.
    b. An RCE, which will obtain the contents being distributed through the ACNS network and make them available to the lower level CEs.
    c. A number of ECEs, which will be used to directly serve requests issued by the terminals. Typically, one ECE is supposed so serve a local network of terminals.
    d. Eventually, one or more middle-level CE. These devices should be understood as middle nodes in a tree network and their presence is not mandatory. They are usually used in more complex networks, either in size or geographic dispersion.
2. One Cisco ACNS channel should be set up for each Mobbit InSight channel. In both contexts, the term *channel* has a similar meaning: a virtual network, formed by one or more sources and one or more receptors, destined to distribute contents, exclusive from other channels. Therefore, in the case of the ACNS system, a channel will be formed by one RCE, one or more ECEs and, if necessary, middle-level CEs. Each Mobbit InSight terminal should obtain the contents it needs in the nearest ECE.
3. The central InSight server must make contents being distributed available to the RCE.
4. The CDM should define, for each ACNS channel, which contents should be distributed. Thus, for each channel, there should be a manifest file containing all necessary files that the InSight system will need to function properly (contents and XML configuration files).
5. For each Mobbit InSight channel, whenever playlist is updated, a new manifest file must be generated. Upon receiving a new manifest file, the CDM orders the CEs accordingly. The proposed solution is to keep a process running on the central server, periodically checking each playlist for changes and generating a new manifest file when these occur. As the CDM checks the manifest files periodically for updates, playlist updates are eventually propagated to the ACNS system.
6. The Mobbit InSight network topology, meaning the locales of the backoffice and terminals, will be static. Henceforth, the Cisco ACNS overlay network can also be static, making it possible to know beforehand which ECE is responsible for the requests of each terminal. Terminals can obtain the contents from the ECE in a non-transparent form, using the CIFS protocol.
7. As it happened before the integration, each Mobbit InSight terminal should periodically check its playlist (now stored at the closest ECE and not at the central server). When a new content is added to the playlist, the terminal must fetch it from the closest ECE.
8. The upstream flow of data – such as statistical information – is independent from the Cisco ACNS network. There is no point in passing these contents through the ACNS networks since they are always different and destined only to the central server.

4. STUDY REGARDING THE APPLICATION OF NETWORK CODING TO PEER-TO-PEER CDNS

As mentioned, in order to study some issues arised by dynamic deployments of CDNs, this section addresses the application of Network Coding strategies to peer-to-peer CDNs, particularly BitTorrent, in order to determine how it may increase the efficiency of content distribution. To better analyse the advantages of Network Coding to these systems, we shall compare the performances of two BitTorrent simulators: one that simulates the traditional system and another that simulates the system modified to incorporate Network Coding. In order to understand this analysis properly, a brief introduction to BitTorrent and to the concept of Network Coding will precede.

4.1 BitTorrent

BitTorrent is currently the most widely used peer-to-peer CDN. A great part of its popularity is due to the fact that if focuses its efforts on efficiently distributing content, unlike most of its competitors which focus solely the process of finding the content, yielding inefficient transfer rates.

The BitTorrent algorithm works by defining one independent overlay network (swarm) for the distribution of each separate content. For each swarm, the content being distributed is divided into a large number of fragments (pieces), so that each peer can obtain those pieces concurrently from a number of peers, maximizing transfer rates. Additionally, any peer can start serving the content being distributed from the moment it receives its first piece. Each peer only interacts with a subset of all the nodes in the swarm: its peer set.

There are essentially two strategies that are responsible for the performance of BitTorrent: the algorithms for piece selection (rarest first algorithm) and maximization of network bandwidth as a consequence of the maximization of the download rate of each peer (choke algorithm). The main goal of the rarest first algorithm is to avoid a problem commonly referred to as the *last piece problem*: when the distribution of pieces among peers is asymmetric, some blocks may be more common and others more rare; thus, if the users who possess the rare blocks decide to leave the swarm, the others are not able to finish the download. This algorithm aims at attenuating the last piece problem by selecting, as the next piece to be obtained by each peer, the rarest amongst its peer set. The choke algorithm, on the other hand, aims at maximizing the download rate of each peer. Since BitTorrent lacks a central coordination, each node is responsible to maximize its transfer rates and it does so by seeking to obtain pieces from nodes that provide faster download rates and chooses to serve pieces to nodes with each it maximizes its upload. This strategy also encourages reciprocity, contributing to reduce the amount of free-riding (downloading content without serving it).

4.2 Network Coding

Network Coding is a method of attaining maximum information flow in a network. When it first emerged, around the year 2000, this field of information theory and coding theory and attracted a

great deal of attention as it promised to significantly improve the efficiency of content transferring over a computer network. Briefly, Network Coding is a method that allows intermediate nodes in a network not only to forward the information flows they receive, but also to process them. This process consists of linearly combining a number of packets, with random coefficients, forwarding them to other nodes in the network. Therefore, a receiver ends up being delivered several packets that result of the combination of multiple other packets, as well as the coefficients used. By arranging the received packets in a linear equation system, and when enough packets are received, a node is able to recover the original packets by inverting the equation system.

This approach contrasts with the traditional packet routing methods. According to these, each packet is indifferently routed throughout the network so that a receiver must collect all of the original packets to rebuild the content. It has been mathematically proved, using the max-flow min-cut theorem of the graph theory, that traditional routing cannot achieve the max value, but Network Coding can.

The process of content distributing using a Network Coding approach can be summed through the following steps, without loss of generality:

1. One server possesses one file, B, which wishes to distribute, composed by n pieces.
2. In order to send a piece E_1 to a client A, the server combines all of B's pieces: it chooses a random vector of n coefficients, C, then multiplies each element i of B by the element i of C (with $1 \leq i \leq n$) and adds the results of the multiplications.
3. The server then sends the piece E_1, along with the vector C, to A.
4. If A already has a piece E_2, generated from original pieces of the content along with the coding vector C', suppose A wants to send a piece of information E_3 to a client B.
5. Given that A has only two pieces in its possession, it generates E_3 by multiplying E_1 with a random coefficient c''_1 and E_2 with a random coefficient c''_2.
6. A sends E_3 to B, along with its coding vector C''. Notice that $C'' = c''_1 \cdot c + c''_2 \cdot c'$, resulting in $C'' = \{c''_1 c_1 + c''_2 c'_1, c''_1 c_2 + c''_2 c'_2, \ldots\}$.
7. Either A or B, in case they possess n linearly independent pieces, can reconstruct the original content in a process that is similar to solving a linear equation system.

Notice that the efficiency of a Network Coding bases system depends on the field size in which the coding vectors are generated: the bigger the field, the smaller the chances of two independent nodes generating two linearly dependent piece. On the other hand, a Network Coding system also faces the overhead of sending a coding vector for each generated piece. Therefore, a dimension for the coding has to be picked in other to achieve a fair trade-off between the probability of linearly dependent pieces being generated and the size of the coding vector being attached to each piece. It is estimated that a field of size 2^8 introduces an overhead of only 3% in the total amount of data traded, yielding a probability of generation of linearly dependent blocks that is significantly lower than the probability of the occurrence of other disturbances over the network, such as packet loss.

4.3 Application Models for the Simulators

In the course of this study, two simulators have been used: one emulates the traditional BitTorrent system and the other emulates a BitTorrent system modified to use Network Coding.

The simulator used to mimic the behaviour of BitTorrent has been developed by Ashwin Bharambe, Cormac Herley and Venkat Padmanabhan for Microsoft Research and freely distributed over the Internet. Its goal is to simulate the data plane of BitTorrent as a sequence of discreet events. The activity of each node is thoroughly modelled, as well as the main mechanisms of the system (the rarest first and choke algorithms). The simulator allows the definition of different simulation scenarios, in which is possible to determine sever different aspects such as the size of the file being distributed, its division in pieces, the number of nodes in the swarm (seeds and lechers), the join and exit rates, the download and upload bandwidths and the simulation time. Regarding the network model, each node is defined as having two independent connections: one for upload and another for download. The bandwidths of the two connections are independent, allowing asymmetric connections to be modelled. The simulator uses this information to adequately determine the delays in which the transfer of data incur: this delay accounts for the upload bandwidth of the node sending the piece, the download bandwidth of the node receiving the piece and the number of connections each node shares. As outputs, besides the behaviour of each node and a chronological list of all the events that took place, the simulator has been modified to provide other important information for this analysis, namely average bandwidth usage for download and average download times. Some details pertaining BitTorrent are not compass of this simulator, as they do not have a big impact in the analysis of the performance of the system. Amongst the pretermitted details are, for example, network delays on data transfers, the modelling of TCP packages or the fact that BitTorrent subdivides pieces into blocks to take advantage of a pipelining mechanism to increase transfer rates.

So to reproduce the behaviour of a BitTorrent system modified to use Network Coding, the original BitTorrent simulator was altered to incorporate this feature specifically for this analysis. The main issue in adapting the simulator was providing all of the nodes with the capability of generating innovative pieces of information by linearly combining previously held pieces. Therefore, a leecher no longer requests a specific original piece since, broadly speaking, any generated piece can be useful in reconstructing the original content. As a side effect, the rarest first algorithm was suppressed because as the notions of rare and common pieces ceased to exist. On the other hand, the notions of innovative and non-innovative nodes were created: for a certain leecher A, the node B is innovative if it is able to generate a piece that is linearly independent from all the remaining pieces that A already has. The new piece-scheduling algorithm thus decides to request a piece from a remote node only if the remote node is innovative. The choke algorithm, on the contrary, was kept since its usefulness is independent from the chosen routing algorithm.

4.4 Simulation Results and Analysis

The goal of the simulations being conducted is to determine the gain that comes from introducing Linear Network Coding to BitTorrrent, as well as to identify the situations in which this gain is more substantial. Hence, the performance of both systems was analyzed by performing simulations according to the variation of two particular parameters: the bandwidth of the leechers populating the swarm and the proportion of seeds in the swarm. As to the output, in order do determine the performance of both systems comparatively, two figures were taken into account: the average bandwidth used for download purposes and the average download time.

In order to comparatively analyze both systems, all the parameters remained fixes throughout the different simulations, except for the leechers' bandwidths. The fixed parameters were defined having in account several observations made on popular swarms: the simulation referred to a two-hour period of a 100MB file being distributed (split in 400 pieces), in a swarm composed by 100 fixed seeds, with 1 leecher joining per second. Each leecher had a probability of 75% of leaving the swarm after finishing the download and, it case it remained, it would leave after serving 88% of the content. The leechers' bandwidths were modelled to resemble commonly used Internet connection speeds, from GSM Circuit Switched Data (CSD) to Fiber To The Buildings (FTTB), as well as many others such as Dial-Up, ISDN, Cable or ADSL. The two systems improved with the increasing bandwidth capabilities, both in terms of download bandwidth and download time. It was also possible to observe that the Network Coding system performs better than BitTorrent in nearly all cases and the difference of performances also increases as the bandwidths grow (fig. 2). The better performance of the Network Coding system over BitTorrent is mostly due to its piece scheduling system: the fact that each user must obtain exactly each of the pieces of the original content might cripple the performance of the system as it might happen that not all pieces be equally available (a consequence of the last piece problem) or, at least, it might not me possible to obtain all pieces at good transfer rates. As leechers only have a subset of the original pieces of the content, they're usefulness as content servers as limited. Additionally, as some pieces become rarer, they might only be available in few nodes and the only ones that surely possess them are seeds. Therefore, one can deduct that the BitTorrent system depends more on seeds than Network Coding since, on the latter system, all leechers are able to generate innovative pieces. Thus, as the bandwidth of the seeds decreases when compared to that of the leechers, it becomes a bottleneck and cripples the overall performance of the system.

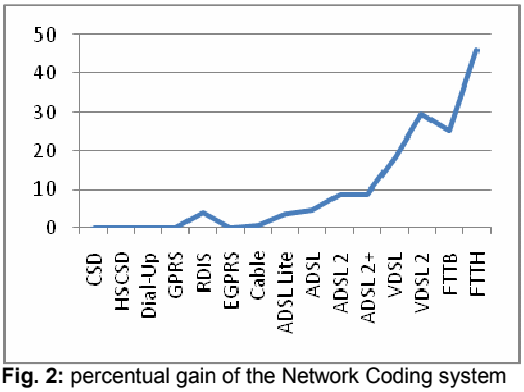


Fig. 2: percentual gain of the Network Coding system relatively to BitTorrent in terms of bandwidth usage.

The influence of the proportion of seeds in a swarm was analyzing by varying the seeding period of the nodes, while fixing the remaining parameters. The node population was modelled according to ADSL bandwidths, as it is the most commonly used connection. Compared to the previous simulation, content size was increased to 300MB to better test the role of the seeds, while the probability of a leecher to leave upon finishing its download was set to zero (meaning that all leechers, when finishing the download, would remain as seeds for a custom period of time). It was possible to conclude that both systems perform better as the seeding time increases, as the content availability grows larger. Overall, the Network Coding system yielded better results than BitTorrent, with its gain being significantly larger as the seeding time decreases (fig. 3). This result also confirms the previously exposed thesis that BitTorrent depends more on seeds, as in its quantity and bandwidth capabilities, than the Network Coding system.

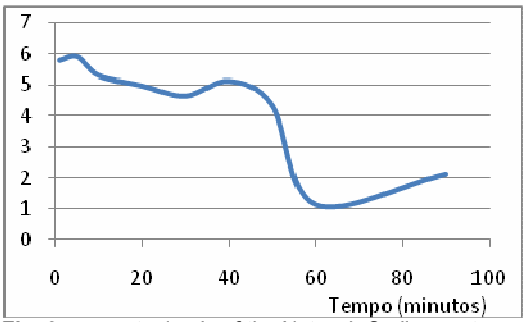


Fig. 3: percentual gain of the Network Coding system relatively to BitTorrent in terms of seeding time.

5. CONCLUSIONS

The first study carried through in the scope of this dissertation consisted on the integration of a Corporate TV system, developed by Mobbit Systems, with a commercial CDN, by Cisco Systems. This integration aimed at solving the problem of an inefficient usage of WAN bandwidth: in case two or more geographically close located required the same contents from the central server, these would be requested separately. Therefore, the situation of the same data being transferred several times through the same WAN link was possible, consisting of a waste of bandwidth resources. By applying a CDN solution to this system, contents could now be cached near the end-users and would cooperate in obtaining data from remote locales. The profit that resulted from this integration were clear, though not linear: they depended on the contents being distributed, on the number and locales of the Mobbit InSight terminals and on the content lineups drafted for each terminal. Although this project was essentially practical, the biggest amount of time was invested in analyzing the two systems and on the most cost-effective way to integrate them, not on actually programming the integration module.

The second study, on the other hand, referred to a study on how the usage of Network Coding methods would improve the performance of peer-to-peer CDNs, such as the popular BitTorrent. To support this analysis, two simulators were used: one to mimic the behaviour of BitTorrent and the other to emulate a BitTorrent system modified to use Network Coding on content transferring. Although, broadly speaking, the introduction of Network Coding turned out to be profitable, both in bandwidth usage and download time, this advantage is especially relevant in specific cases: when the average bandwidths of the leechers are high (especially when these are higher than the average seed bandwidths) and when the concentration of seeds in the swarm is scarce. Still, several other issues must be taken into account: the amount of data transferred in a Network Coding system is higher, since a coding vector must be sent along with each piece; there is a probability, higher than zero, that amongst all of the pieces received by a node two or more are linearly dependent; finally, although it does not pertain to data transfer, in a Network Coding system the content must be reconstructed from the coded pieces, in an operation that renders this system more computationally complex than BitTorrent, therefore more prone to errors, and possibly more processor intensive.