Technical University of Crete

School of Electronic and Computer Engineering

Thesis

" Cache Management Policies for Streaming Proxies : the cases of fixed and variable video sizes and collaboration among proxies "

Athanasios Kyriazis

Chania, 2015

Ευχαριστίες

Με την ολοκλήρωση της παρούσας διπλωματικής εργασίας θα ήθελα να ευχαριστήσω αρχικά τον επιβλέποντα της διπλωματικής μου εργασίας κ. Μιχάλη Πατεράκη που μου έδωσε την δυνατότητα να ασχοληθώ με κάτι που μου αρέσει και με ενδιαφέρει, καθώς επίσης και για τις πολύτιμες συμβουλές και το ενδιαφέρον του. Τον ευχαριστώ για όλα όσα μου έμαθε κατά τη διάρκεια της συνεργασίας μας και το θεωρώ τιμή μου που συνεργάστηκα μαζί του. Ακόμη θα ήθελα να ευχαριστήσω τα μέλη της εξεταστικής μου επιτροπής κ. Πολυχρόνη Κουτσάκη και κ. Άγγελο Μπλέτσα που ασχολήθηκαν να διαβάσουν και να εξετάσουν την διπλωματική μου εργασία.

Επίσης η ολοκλήρωση της διπλωματικής μου εργασίας σηματοδοτεί την ολοκλήρωση των σπουδών μου. Για αυτό το λόγο θα ήθελα να ευχαριστήσω τους γονείς μου Αλέκο και Έφη Κυριαζή, καθώς επίσης και την γιαγιά μου Πελαγία Κυριαζή που με στηρίζουν οικονομικά και ψυχολογικά όλα αυτά τα χρόνια ώστε να καταφέρω να περατώσω με επιτυχία τις σπουδές μου. Ακόμη θα ήθελα να ευχαριστήσω την αδερφή μου (επίσης Πελαγία Κυριαζή) για την βοήθειά της στην συγκεκριμένη εργασία αλλά και για την στήριξη και την αγάπη της όλα αυτά τα χρόνια.

Τέλος θα ήθελα να ευχαριστήσω την κοπέλα μου Τίφανη Δανιήλ για τη βοήθεια και τη στήριξή της και όλους τους φίλους μου (Χρόνης, Βουτσάς, Νίκος, Μανώλης, Στέλιος, Αλέξανδρος, Στέλιος, Δημήτρης, Βασίλης, Γιάννης) για όλα αυτά τα υπέροχα χρόνια που περάσαμε κατά τη διάρκεια της φοίτησης μου στο πολυτεχνείο.

Αθανάσιος Κυριαζής

Abstract

The easy access on the Internet, the exponential growth of the Internet traffic and the ever enlarging sizes of files which are transferred over it, often lead to network congestion and significant delays in many applications that require timely data transfer. The most important part of data transfer over the Internet corresponds to media streaming applications which require a large portion of the available bandwidth, while at the same time there is the need of achieving low delays, because the delays in these applications are very annoying to the users and some of them may not be willing to experience them. In order to improve the services provided to the users of these applications, the techniques of video caching and of collaborative caching are used. Video caching embodies proxy servers on the network in order to keep frequently accessed data at locations close to the clients. Collaborative caching uses group of proxies which can collaborate with each other, and achieve better performance than with independent standalone proxies. This Thesis proposes new efficient algorithms for video caching which manage the cache of the proxy servers for video transfer in real time and compares them to existing algorithms. Also, a new effective collaborative scenario is proposed and evaluated.

In the first two parts of the Thesis the performance of three existing efficient algorithms in the literature, the Number of Common Clients - Size (NCCS), the Least Recently and Least Frequently Used (LRLFU) and the Least Recently Used (LRU) is evaluated and compared for the cases of fixed and variable video sizes. In each of the first two parts of the Thesis a new cache management algorithm is also designed and proposed, this new algorithm consists of a combination of existing efficient algorithms. In the first part of the Thesis the proposed new algorithm (referred to as the LRLFU_Aggressive) does not perform much better than the existing algorithms, but in the second part the proposed new algorithm (referred to as the LRLFU_Modified) significantly improves the performance of the system in comparison to the existing algorithms. In the cache replacement algorithms we examine a value defined for each video, namely the caching value, which is based on the popularity of the videos, in order to retain in the cache the most popular videos. For each requested video which does not exist into the cache, other videos have to be removed (the ones with the lower caching values) in order for the new video to be cached. The performance metrics we use to evaluate and compare the algorithms in our system are the Video Hit Ratio and the Fraction of the Delayed Starts.

In the third part of the Thesis a collaborative scheme of two client communities each associated with one proxy server, is defined and studied. While most of the studies in the area of collaborative proxies use the same cache management algorithm in each proxy and assume that each client community communicates via the corresponding proxy with the same content server, in our collaborative system we have used a different cache management algorithm in each of the two proxies and we assumed that there are two content servers each one associated with one of the client communities. Furthermore, we assumed that each of the client communities communicates via the corresponding proxy only with its corresponding content server. Also, we assumed that there is an overlap between the contents of the two content servers. Each of the two different cache management algorithms used in the collaborative scheme caches different videos in the corresponding proxy, and this helps the overall system performance. From the results of our simulation study we derive important conclusions for the performance of the examined collaborative scheme.

Contents

List of Figures

List of Tables

Chapter 1 : Introduction

1.1 Introduction to Video Caching in Streaming Proxies

The internet was designed as a free place for the whole world to communicate and interact on a broad scale. However, the easy access on it, the ever enlarging sizes of files which are transferred and the exponential growth of Internet traffic which shows no sign of slowing down, often lead to network congestion and significant delays in many applications that require timely data transferring. Especially nowadays, the most important part of data transfer has been occupied by media streaming applications, like on-line video lectures, streaming live tv, on-line movies and many others. The access to streaming media requires a high and stable transmission rate to satisfy the end user. But, because of their large size, streaming media requires a large portion of the available bandwidth which is one of the most important problems facing the internet today. Because of the stringent requirements of streaming media on network bandwidth, delay and loss rate, the delivery of multimedia contents with high quality and low cost over the internet is problematic even today, when the internet has experienced strong growth. An important role in the development of these problems, plays the rapid increase of users who have access to internet. As the number of users increases, it becomes more and more difficult for the system to serve their needs, which are to view the whole video without any interrupts or delays. These factors mentioned above, create the most significant problems in streaming media transfer and for that reason they are analyzed in section 1.1.1.

Many techniques like collaborative caching or streaming proxies have been used to control the media traffic and improve the quality of data transfer. Without the use of proxies on a congested network, users are likely to experience problems such as jagged video, patchy audio and unsynchronized video and audio as packets are dropped or arrive late. By using the proxy, it is possible to increase the quality of service, reduce pauses and buffering during playback and save network bandwidth. Even better results can be achieved using a hierarchy of proxies which can cooperate with each other. This is done by the technique of collaborative caching. So, it is worth mentioning how video caching and streaming proxies work in section 1.1.2, describe the technique of collaborative caching in section 1.1.3 and give a quick description of protocols and architectures which are used to streaming media transfer in section 1.1.4. The rest of this Chapter is organized as follows. The relevant work in the field of video caching and how the work in this Thesis is affected by it, is described in section 1.2. In section 1.3 the contribution of the Thesis is presented and discussed and finally in section 1.4 the outline of the work in this Thesis is given.

1.1.1 Main problems and features of streaming media

The streaming media content is sent in compressed form over the internet and is displayed by the viewer in real time. With streaming media a web user does not have to wait to download an entire file in order to play it. Instead, the media is sent in a continuous stream of data and is played as it arrives. This characteristic of streaming media has the advantage that users do not have to wait to download the entire file in case they want to watch it immediately. But, in this way many problems are created, such as the bandwidth limitations or the difficulty to be transferred very large files. These problems are created because of the specific features of streaming media and are discussed below.

The main feature of streaming media which cannot be limited is their large size. Of course that size varies depending on the quality of media and the compression that has been used. For example, a one hour file with quality of 240p and H.264 compression could be about 200MB, an SD (Standard Definition) file with MPEG-2 compression 1.5 GB and a FullHD 1080p file with MPEG-4 compression could reach 7 or 8 GB. Thus, it is clear that there are fundamental limitations in storage, especially in systems that provide high quality streaming media.

The second important feature of streaming media is the huge bandwidth requirements. These requirements are directly influenced by the file size. It is reasonable that, the larger the file to be transferred, the greater the bandwidth required in order to transmit the file without problems to the user. To understand the required bandwidth in a system, a simple example is given next. Let’s consider a system with 100 users who are watching simultaneously one video with SD quality. The bit rate required to avoid delays and pauses is about 3.5 Mbps * 100 = 350 Mbps [17] unless multicasting is used effectively. Of course there is no system that provides only one video file and it is very often in popular steaming media systems that there are much more than 100 users watching at the same time the same video. If we take into account the fact that most of the videos today are High Quality, it is obvious that the total bandwidth required is very high.

Finally, another feature of streaming media not considered in this Thesis, is that the viewers have freedom of movements within the video. According to surveys, most of the users do not watch the entire video. Instead, they watch a part of it or they use forward and backward movements. The immediate result of such user actions is to abruptly cut the data flow and complicate the way the system manages the available memory.

It is obvious that, for the reliable and timely data transfer to the users some performance metric goals, should be met. These performance metrics, constitute the performance evaluation criteria of our system that we will see in Chapters 2, 3 and 4. These metrics are the Video Hit Ratio (VHR) and the Fraction of Delayed Starts (DS) which are briefly described below.

With the term of Video Hit Ratio we mean the number of times the proxy server found an object requested by the user in the cache, divided by the total number of times it looked for some object in the cache. It is obvious that, the higher the VHR, the more effective the cache is at improving the system performance. The other performance metric, referred to as the Fraction of Delayed Starts corresponds to the fraction of times a user needs to wait, until it starts watching a video. The Fraction of Delayed Starts is the result of the “distance” between the video server and the user (client). We need to reduce the Fraction of Delayed Starts as much as possible in order to satisfy the users and improve the overall system performance.

To improve the quality of service provided to the user and to reduce the problems mentioned above, the technique of Video Caching is used which we analyze in the next section.

1.1.2 Basic Principles and Architecture of Video Caching

In recent years because of the problems that have been presented and discussed in section 1.1.1, the real time streaming media transfer received considerable attention by researchers in the field. Many techniques have been proposed to reduce these problems, one of which is the video caching. The main idea of video caching is to keep frequently accessed data at locations close to the clients. In that way, retrieving data from these caching locations, reduces transmission time across the internet and the workload imposed on the server. For streaming media objects, caching becomes especially attractive due to their features which are long duration, predictable sequential nature of accesses and high network resource requirements. The architecture of video caching consists of three basic parts which are represented in Figure 1.1 and their functions are described below.

The first part is the clients. With the term of client, it is meant the program that establishes connections for the purpose of sending requests. A very simple example of a client is the browser of a computer when the user clicks to watch a video.

The second part is the proxy server [19]. This is the main part of video caching. A proxy server is a dedicated computer or a software system running on a computer that acts as an intermediary between an endpoint device, such as a computer, and another server from which a user or client is requesting a service. When a proxy server receives a request for a video, it looks first in its local cache. If it finds the content, it returns it to the user without having to forward the request to the Internet. If the content is not in the cache, the proxy server, acting as a client on behalf of the user, requests the video from the server out on the Internet. When the content is returned, the proxy server relates it to the original request and forwards it on to the user.

Finally the third part is the Content Server in which all videos that exist in the system, or the videos that will be created in the future are stored. Its work is to accept connections in order to service requests by sending back responses.

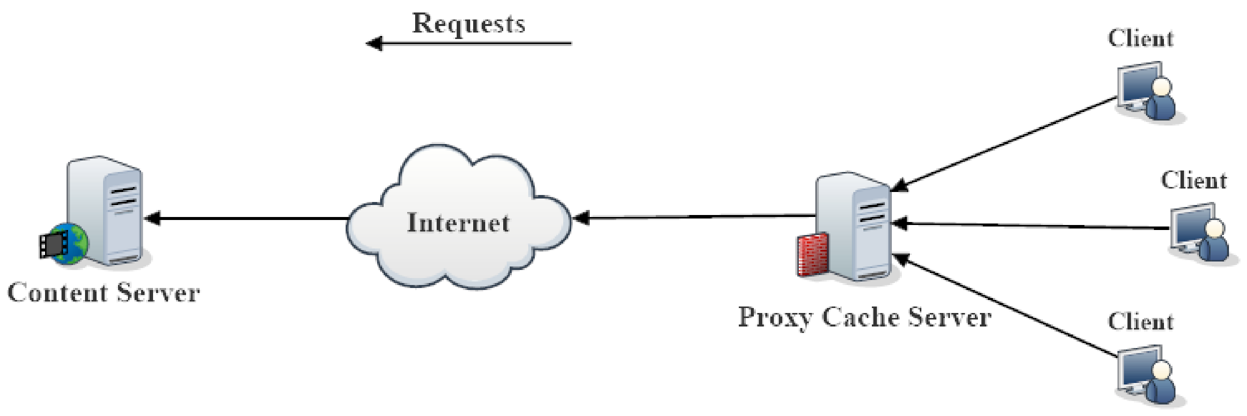

Figure 1.1 : Architecture of Video Caching

Although proxy caching is a really effective technique, with the ever increasing traffic on the internet there is the need of better architectures. So, a better technique developed, is to have a hierarchy or group of proxies that will be able to cooperate with each other. This technique named collaborative caching is described in section 1.1.3.

1.1.3 Basic Characteristics of Collaborative Caching

In general, proxies grouped together can achieve better performance than independent standalone proxies. Specifically, at group of proxies can collaborate with each other to increase the aggregate cache space, balance loads and improve system scalability. Many collaborative scenarios have been developed over the years, each of which with its own characteristics. But, the main idea of all collaborative scenarios is that when a cache miss occurs (that is the requested video is not in the cache) the proxy does not immediately request the video file from the content server. Instead, the proxy first searches for the file in another proxy into the collaborative group. If the file exists in another proxy, then the proxy brings the file and immediately serves the clients with low delay. Only if none of the proxy caches has the video, then the proxy contacts the Content server and requests for the file. Collaboration among caching systems requires the ability to identify other caches and to learn about their current content. This feature is implemented differently in each scenario, depending on the system structure that has been used. Ideally, a cache would always know the current state of all the other caches, in order to immediately know where to look about the requested content. However, it is impractical to keep a consistent and up-to-date view of all proxy cache contents. The architecture of a quite simple collaborative scenario is given in Figure 1.2.

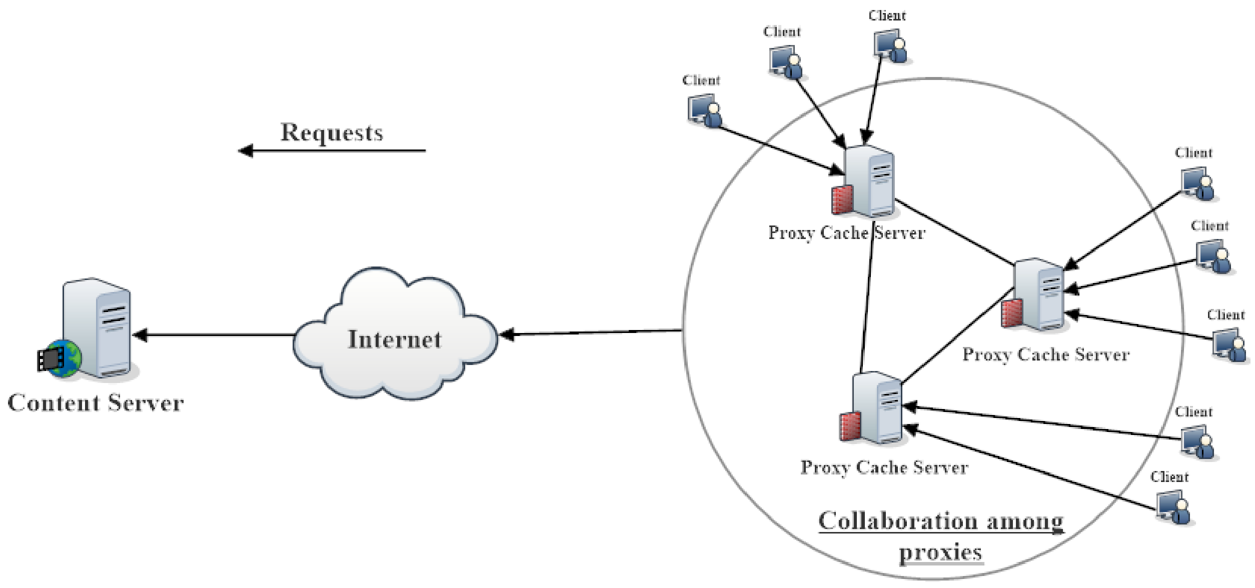

Figure 1.2 : Structure of a Simple Collaborative Scenario

1.1.4 Basic Protocols for On-line Video Streaming [18]

The popularity of Internet multimedia applications has grown dramatically in the past several years, spurred by the penetration of the Web and the increasing capacity of backbone and local-access networks. Although HTTP can be used to transfer audio and video content, most multimedia transmissions are simply initiated on the Web. So, the Integrated Services working group of IETF developed a model of internet services which is called integrated services and includes best-effort service and real-time service. This model consists of RTP, RTCP and RTSP protocols which help to improve the quality of services provided to the clients by the system. To understand how these protocols work we give a quick description of each of them.

The RTP protocol provides the basic functionality for transferring real-time data over packet networks. Since RTP does not include mechanisms for reliable delivery or flow control, transport of RTP packets must rely on underlying protocols such as UDP and TCP. RTP typically runs over UDP, though TCP is sometimes used for reliable transport or to stream across firewalls that discard UDP packets.

The RTCP protocol monitors the delivery of RTP packets. The protocol provides feedback on the quality of data distribution, establishes an identification for each participant, scales the control packet transmission with the number of participants and provides minimal session control information. RTCP packets consist of source descriptors, sender reports, receiver reports, BYE packets and APP packets, where a BYE packet is sent to report that one more sources have left the session and the APP packets are intended to experimental use. Receiver reports include the fraction of RTP packets lost, the sequence number of the last RTP packet received, and the packet interarrival jitter. Senders can use this information to modify their transmission rates or to switch to a different encoder. The sender reports include the stream's SSRC, the sequence number of the last RTP packet sent, the wallclock time of the last transmission, and the number of RTP packets and bytes sent. The client can use the RTP timestamp and wallclock time information for media synchronization.

Finally the RTSP protocol coordinates the delivery of multimedia files, acting as a “VCR remote control”. RTSP typically runs over TCP, though UDP can also be used. RTSP is stateful and has several different methods. The OPTIONS method inquires about server capabilities and the DESCRIBE method inquires about the properties of a particular file. Client and server state machines are created with the SETUP method, which also negotiates transport parameters. The client sends a SETUP message for each stream in the file. Streaming of the file is initiated with the PLAY method, which can include a Range header to control the playback point. The TEARDOWN method terminates the session, releasing the resources at the server and client sites. The protocol also includes a number of recommended or optional methods.

1.2 Related Work

In order to solve the problems mentioned in previous sections and improve techniques that support high quality steaming media, considerable research has been done. These studies focus their interest on the use of proxy servers for storing the most popular videos. Each work use different methods in order to achieve the desirable result.

The different methods mostly concern on how the system is structured (system model) as well as on the way the system manages the memory. However, important role on the implementation of each system play the different distributions that have been used to implement the different sizes of videos, the user request arrival processes and the popularity of each video.

1.2.1 System Model

In studies presented in the literature, different system models have been implemented. There is the simple system model that was described in section 1.1.2 and systems like that, or variations of that, have been used in [7] and [16]. Also, there is the collaborative system model which has grown rapidly in recent years and a simple form of it was described in section 1.1.3. Some studies that uses collaborative caching models are [1], [3] and [8]. Finally, some studies use a video segmentation scheme with two or three caches. Different segments of the same video can be stored in these caches. In our Thesis, ideas from the first two system models above have been adopted, while the segmentation schemes have not been considered. However, since video segmentation is an important technique for video caching, we briefly discuss it here for completeness.

In [1] and [2] many different system models have been proposed in order to evaluate the proposed therein cache management policy (LRLFU). Specifically, in [2] a simple scenario has been considered with a far-distant origin server, proxy servers located close to the client populations, and client devices. Also, two video segmentation schemes have been used. In the first scheme, a media object consists of multiple equal-size blocks, where the block is the smallest unit of transfer. Blocks of a media file received by the proxy server are then grouped into variable-size, distance-sensitive segments. The segment size increases exponentially from the beginning segment of the video, forming a pyramid. In second video segmentation scheme considered, the first segment consists of a small number of blocks Bp, forming the prefix of the video and later segments consist of a larger fixed number of blocks. Bp is chosen in a way to guarantee that enough blocks will be in the cache to mask the latency in the server-proxy path in order to guarantee continuous streaming of the video once it is started. Every proxy in these schemes, caches different parts of the video files (Cache A is used for prefix and Cache B for the remaining segments of the video) and use different cache replacement and admission policies. In [1], the same segmentation schemes as in [2] are used, but a collaborative scenario has also been added. In this collaborative scenario a hierarchical tree topology is considered, consisting of proxies equipped with a small size cache, located very close to the client communities and used to serve one client community each (referred to as type A proxies) and proxies equipped with a much larger size cache than that of type A proxies, located further away from the client communities and used to serve more than one client community each. If a client requests a video from a type A proxy and this proxy does not have the prefix of this video, it asks the others type A proxies for it. If it finds it there, it caches it and then sends it to the client. Instead, in the non-collaborative scenario, if the prefix of a requested video is not found in the corresponding type A proxy, then the video is requested from the corresponding type B proxy and the server and there is a delayed start for this video request.

In [5] a simple system structure with a single VPS (Content Server), a single proxy cache and multiple clients is examined. Also, a stochastic model-based algorithm incorporating a multi-level Hidden Markov Model (HMM) for video segmentation and indexing has been used. Furthermore, it has been assumed that three video quality layers, namely V_{org} (for the basic video quality layer) and, V_{mid} and V_{base} (for the two lower video quality layers), are generated for each video segment. The studies in [3] and [4] are variants of [5] with more complicated system models. In [4] the system consists of one or more origin servers that host video content and a number of proxy caches. Each proxy cache is responsible for clients in a certain geographical area in the sense that their requests to video content first reach that proxy cache. In [3] a collaborative scenario with a number of cooperating geographically distributed proxy caches at the edge of network is proposed. In this study, the proposed framework supports multiple levels of client privileges in terms of the visual quality and content abstraction of the delivered video.

In [7] a simple system with geographically dispersed origin servers is proposed, while in [8] a collaborative caching system that allows a program to choose different caching methods for its data is studied. Finally in [10] an Internet streaming media delivery architecture, consisting of caches deployed at the edge of the Internet, streaming media objects that are replicated either entirely or partially on these caches and clients, whose requests are satisfied (possibly jointly) by servers and caches is considered.

1.2.2 Memory Management

Over the years, many different methods for cache management have been proposed and used on systems implementing streaming media caching. The first methods implemented, were the LRU (Least Recently Used) and the LFU (Least Frequently Used) which both have been used in many streaming media caching systems [1]-[16], either as the main cache management policies or for the purpose of comparisons. Although these methods are quite simple, they have disadvantages especially in terms of the efficiency of cache management. Some later works tried to improve these methods, an example is the LRLFU method that has been introduced in [1] and [2] and constitutes a combination of LRU and LFU. Another method that has been proposed in the literature and was shown to outperform LRU and LFU is the NCCS [3]-[5]. Many other techniques based mostly on LRU and LFU have been proposed in the literature, like ARC [15], RBC [6], LRU-MRU [8], GDSF [12], as well as others, but none of them have been considered in our Thesis.

LRU is the most widely used cache replacement policy in many different system areas. The data objects in an LRU cache are sorted by the last time used and LRU replaces the least recently used object when cache is full. However, LRU has an important disadvantage referred to as the "cold cache pollution" [13]. For instance, in LRU whenever a new object is inserted in the cache, it is put on the top of the cache stack. If the object is not popular (e.g. it is used only once), it will take some time before it is moved down to the bottom of the stack (marked as “least recently used”) and dropped from the cache. Nevertheless, it is a method easy to implement and performs well in specific situations. The caching value of a video which is used to implement LRU is given by the expression $\frac{1}{T-T'}$, where T is the current time and T' is the last time that this video has been requested. A number of recency-based policies are more or less extensions or variants of the LRU policy. With no exaggeration, all researches in [1]-[16] have used LRU in some way (e.g. as a cache management policy, as a comparison case or as a basis for inspiration to create a new more efficient cache management policy).

LFU is a frequency based policy that tries to replace those videos that have rarely been accessed and keep in the cache the popular videos that are already accessed many times. The LFU policy replaces the least frequently used video when an existing video in the cache needs to be evicted for a newly arrived video. However, the LFU policy has a serious drawback that is referred to as the "hot cache pollution" [13], which is, that videos that have accumulated a large number of requests in the beginning of system operation, may not be accessed again. These stale videos occupy and waste the cache space. Thus, there is no free space for new videos to be cached. As a result, cache hit ratio may be severely decreased. The video caching value that has been used to implement LFU is given only by one variable that is denoted by RF in many studies, RF is equal to the number of requests the specific video has received until now. LFU has been used in many works in the literature, either as basis for inspiration to created new efficient cache management policies or as a comparison case, [1]-[6], [8], [9], [13].

LRLFU (Least Recently and Least Frequently Used) constitutes a combination between the LRU and the LFU policies. It is capable of capturing the changing popularities of the various videos by attaching a caching value to each video according to how recently and how frequently the video has been requested and decides to cache the most 'valuable' videos. The results of the performance study in [1] have shown that the LRLFU cache management policy when applied to a simple topology of proxies and compared with previous work in this area (which uses the LRU-i policy that constitutes a variant of the LRU) improves the byte-hit ratio, reduces the fraction of user requests with delayed starts and requires less CPU overhead. The caching value in the LRLFU policy is given by the expression $\frac{\mathrm{RF}}{\mathrm{T}-\mathrm{T}'}$, where RF is the number of requests for a specific video until now, T is the current time and T' is the last time that this video has been requested. In that way, LRLFU alleviates the problems of both the LRU and LFU policies and has been used in many works as the cache management policy or as comparison case, [1], [2], [16].

The NCCS (Number of Common Clients over Size) cache management policy has been considered in [3]-[5] and is compared with both the LRU and LFU policies. The results have shown that it outperforms both the LRU and LFU policies on the scenarios that have been considered. These scenarios assume that there is a time window (which is set to 3 seconds). Also, the client requests are clustered and a batch group of client requests arrives within this window. Every batch contains on average 150 client requests. Furthermore, it is assumed that there are three versions of video visual quality V_{org}, V_{mid} and V_{base} for each video with the corresponding size (V_{org} is the biggest video file which is of high-quality while V_{mid} and V_{base} are smaller video files with lower quality) and the proxy cache generates V_{mid} and V_{base} from V_{org} in real time after fetching V_{org} from the VPS and stores all of them locally. The caching value of the NCCS policy is given by the expression $\frac{\mathrm{NCC}}{\mathrm{FS}}$, where NCC is the number of common client requests requesting the particular video and FS is the requested video size. In the NCCS cache replacement policy, the video file with the minimum caching value is first identified and the versions V_{org}, V_{mid} and V_{base} of the identified video file are removed in that order, until there is enough space to load the requested video file in the proxy cache.

1.2.3 Various Probability Distributions

A main part of each simulation based system consists of the distributions used to emulate the various stochastic functions of the system. These distributions are often used to implement the client video preferences or otherwise the popularity of each video, the way that the user requests are arriving and the sizes of each video file. Of course, depending on the work considered in the literature, for some of these system functions no probability distribution is used. For example in [3], [6], [7] and [16] the file size is assumed fixed. In addition, in some works, probability distributions are used to model the data bandwidth, the viewing time limit [3] e.t.c. These probability distributions are usually used for mobile clients and they are not considered in our work.

The vast majority of the studies [1]-[10] have implemented the arrivals of requests with the Poisson process with parameter λ (mean arrival rate), which is selected differently in every case. For example, authors in [1], [2] assume that client requests for videos arrive according a Poisson process with mean interarrival time equal to 60 secs, that is 1 req/min on the average. In [3], [4] a Poisson arrival Process with maximum delay of 4s and variance of 2s is assumed, in [7] the mean request arrival rate is 30 reqs/hour and in [9] is equal to 0.2 reqs/sec.

For the implementation of the popularity of videos some studies use the uniform distribution [3], [4]. But most studies use a Zip-f Distribution [1], [2], [6], [7], [9], [10], [16] because it has been shown that Zipf models much better the video popularity. That happens because the more recently accessed videos are accessed more frequently than the older videos, so they should be more popular and the distribution should be skewed. That cannot be achieved with a uniform distribution. Also a significant advantage of the Zipf distribution is that there is the ability to adjust it on each system simply by changing its skew factor. If the value of skew factor is set equal to 1, the Zipf distribution becomes the uniform.

Finally, the varying video file sizes can be modeled based on a uniform or a normal distribution. As mentioned before, in some systems it is assumed that the video file sizes are fixed. The most common distribution used for varying video file sizes is the uniform [1], [2], [4]. In some studies, more than one scenarios have been examined, where the first scenario considers with fixed file sizes and in the second scenario a probability distribution for the video file sizes is used [4]. Also the normal distribution has been used [10], because the normal distribution with a high variance tends to be similar to the uniform probability distribution. So, by changing the value of the variance of the normal distribution we can observe the change in system performance.

1.3 Goals and Contribution of the Thesis

As we have already mentioned, our main goal is to reduce the Fraction of Delayed Starts and increase the Video Hit Ratio. To achieve these goals we have created a new system model which is based on ideas from [1]-[5].

In the first part of our work which consists of the simplest scenario we have used both the LRLFU and NCCS cache management policies to create a new cache management policy. Also we have not used video segmentation but we have taken the idea from [3]-[5] to use a time window and to process the user requests that have arrived within each time window at the end of the window, a characteristic not present in [1], [2]. In addition, we have used a Zip-f like distribution to model the video popularity instead of the uniform distribution used in [3]-[5]. Initially we assume that all videos have a fixed size, like in [3], and that user requests arrive according to a Poisson process.

Subsequently, in the second part of our work we introduce a popularity distribution for the varying video file sizes. We have used a normal distribution to implement the different video file sizes instead of [1], [2], [4] in which a uniform distribution has been used. Finally, we also designed a new cache management policy, a little different from our previous one, which achieves better performance.

In the last part of our work a new system based on collaborative scenarios from [1] and [3] is considered. The system model considered is quite simple, but we propose a new technique that has not been used in works [1]-[5]. We only use two collaborative proxies and assume that each of them implements a different cache management policy. Specifically, one of the proxies implements the best cache management policy and the other one implements the worst cache management policy according to the results from the second part of our work. Finally these two proxies take their content from two different content servers. These two content servers do not store the same videos, we assume that there is only an overlap of their contents.

In order to make fair comparisons, we have implemented the cache management policies from [1] and [3] as well as the LRU policy, and we compare the results based on our performance metrics.

1.4 Thesis Outline

The remainder of this work is organized as follows. In Chapter 2 the first part of the work is presented, which is based on the assumption that all videos are of the same size. Initially, in section 2.1 we introduce the basic ideas of video caching in proxy server and the way we manage the storage. In section 2.2 we explain in more details the architecture of our system and the way in which data are stored. In section 2.3 the performance metrics of our system are presented, the simulation model is described as well as results obtained by the simulated scenarios are presented. Finally, in section 2.4 we present the conclusions that have been obtained by the performance study in this Chapter.

In Chapter 3 the second part of our work is presented. In contrast to Chapter 2, we assume here that the videos which are stored in the Content Server have variable sizes. In section 3.1 an introduction to videos with variable sizes is given and the reasons we decided to introduce this feature are explained. In section 3.2 we describe the changes that have been made, in the algorithms and on how the cache storage is managed, so that the system can support videos with variable sizes. In section 3.3 the performance metrics are presented, the simulation model is described and the corresponding results are given. Finally, in section 3.4 we discuss our conclusions.

In Chapter 4 the third and final part of our work is presented, which consists of the implementation of a collaborative caching scenario. In section 4.1 we introduce the characteristics of this scenario. In section 4.2 we describe the changes have be made in the algorithms and on how the cache storage is managed, so that the system can support the collaborative scenario. In section 4.3 the performance metrics of this system are presented, the simulation model is described and the corresponding results are given. Finally, in section 4.4 we discuss our conclusions.

Finally, in Chapter 5 the conclusions of the work in this Thesis are presented. In section 5.1 we discuss the overview of our work, in section 5.2 we provide the main conclusions of our work and we discuss our research contribution, while in section 5.3 we present some ideas for future work.

Chapter 2 : Design and Performance Evaluation of Fixed-Video Size Cache Management Policies

2.1 Introduction

In this Chapter we introduce various algorithms for storing videos in proxy caches. In our simulation study we implement two algorithms that have already been used (LRU, LRLFU), a slightly modified version of the NCCS algorithm, and we propose an efficient new cache management algorithm. All these algorithms are capable of capturing the changes on video popularities by attaching a caching value to each video according to the different cache management policies. Each policy decides to cache the most "valuable" video and replace the most "useless" video according to its caching value. In our study in this Chapter we assume that all videos have the same size, the system consists of one proxy server with a single cache and users watch the videos serially from the beginning to the end. To compare the variable cache management policies we use two performance metrics which are : (1) the Video Hit Ratio and (2) the Fraction of Delayed Starts. In the next sections we describe the network topology, the cache management and replacement algorithms, the performance metrics, the simulation model and we present the results of our simulation study.

2.2 System Model

2.2.1 Network Topology

The system architecture that we consider is shown in Figure 1.1. Our system consists of a far-distance content server, a single proxy server and client devices. All the video files are stored in the content server, the proxy server is located close to the clients and caches a percentage of the content server files, which are the most popular based on their caching values. Clients send requests which are directed to the proxy server. If the proxy has the video file requested into its cache, it transmits it to the client and the video file remains into the cache. If the proxy has not the video file requested, it sends a request to the content server asking for the file. The content server sends the file to the proxy and the proxy serves the client. The new file is stored into the cache by removing another video file. It is also assumed that the bandwidth between the proxy and the clients is sufficient to support video streaming with negligible latency. So, if a requested file exists in the cache there is a request hit and no delayed start is experienced by the client. In case the requested video file does not exist in the cache there is a request miss and the proxy needs to communicate with content server to bring it. Because the latency between the content server and the proxy is quite significant, the client experiences a delayed start.

2.2.2 Cache Management/Replacement Policies

As we already mentioned in section 2.1, we have adopted three different cache management algorithms and we have implemented them in our system. These are, the LRLFU algorithm from [1], [2], the LRU algorithm and a modified version of the NCCS algorithm from [3], [4]. We have also implemented a new cache management algorithm which is more aggressive than LRLFU and we refer to this algorithm by the name LRLFU_Aggressive. Bellow we discuss the cache management policy which have been implemented and the way we have implemented the four different cache replacement algorithms.

When a client request for a specific video arrives to the proxy server, the proxy server acts as follows. First it looks for the video in its cache. If the requested video is stored in the cache, then its caching value is updated and the video is sent to the client. Instead, if the requested video is not stored in the cache, then the cache replacement policy is activated. The cache looks in its content for the less popular video, that is the video with the minimum caching value, and the proxy requests the new video from the content server. The video with the minimum caching value is removed from the cache and the new video is cached and is sent to the client. Since all videos have the same size, when one video is removed from the cache the new video that has been requested can it stored in its place. The new video obtains a caching value according to the rule of the cache replacement algorithm in use. Also, in our study we assume that time is divided into time windows, of equal length. In cache management policies that are time dependent (LRU, LRLFU, LRLFU_Aggressive) we update the caching values of all the videos in the cache at the end of each window.

In the LRU cache replacement policy the caching value depends on how recently a specific video has been requested. The video requested more recently has the higher caching value and is placed on the top of LRU stack, while the oldest video requested has the minimum caching value and is placed at the bottom of LRU stack. That means the oldest video, is the video victim and it will be removed first from the cache when a new video needs to be stored in the cache. The caching value of a video file in the LRU algorithm is given by the expression $\frac{1}{\mathrm{T}-\mathrm{T}'}$, where T is the current time and T′ is the last time that this video has been requested. The difference T-T′ in the denominator of the caching value is the time elapsed from the last request of a specific video. To update the caching value of a video file stored in the cache, we have to update the value of T, that is the current time, at the end of each window.

In the LRLFU replacement policy the caching value depends on how recently and how frequently a specific video has been requested. The caching value of a video file in the LRLFU algorithm is given by the expression $\frac{1}{\mathrm{T}-\mathrm{T}'} * RF$, where $\frac{1}{\mathrm{T}-\mathrm{T}'}$ is the same fraction as in LRU. Here, the caching value of LRU is multiplied with the RF, which is a counter that counts the number of times a video has been requested from the time it has been stored in the cache until the current time. A video not stored in the cache that has been requested within the last time window is not necessarily the most valuable video as it happens in the LRU. Instead, if this video is requested for the first time it might be less valuable than other videos that have been requested some time windows before, but they have been requested frequently and they have a high RF value. The counter RF is updated each time that a cached video is requested and when a video is dropped off the cache its RF value is deleted. The next time that a request for that video will come, the value of RF will start from one. Finally, as in LRU, we update the value of current time T at the end of each time window.

In the NCCS replacement policy the caching value of a video file depends on how many times the video is requested within the current time window divided by the file size. In our study in this Chapter where all the video files are assumed to have the same size, the NCCS policy does not depend on the file size because the denominator of the video caching value will be the same for all the videos. So, while in [3], [4], the caching value is given by the expression $\frac{\mathrm{NCC}}{\mathrm{FS}}$, where NCC is the number of common client requests for the particular video file and FS is the file size, in our simulation in this Chapter the NCCS caching value consists only of the NCC. In other words, we only count how many times a request for a specific video arrives in each time window. In this case, the NCCS replacement policy is not very efficient. For this reason only in this Chapter we propose a modified NCCS replacement policy. This cache replacement policy, referred to as the NCCS_Modified, chooses from the candidate videos to be removed (i.e., among which have the least NCC value), the video which has been requested first in the past. The caching value in the NCCS_Modified policy is given by the product $\mathrm{NCC}^* \mathrm{T}'$, where T' is the last time the video has been requested.

The cache replacement policy that we propose in this Chapter is the LRLFU_Aggressive policy. The LRLFU_Aggressive is a combination of LRLFU and NCCS. The caching value we propose in this replacement policy is given by the expression $\frac{\mathrm{RF+NCC}}{\mathrm{T-T'}}$. The idea is to give a boost to videos which have been requested the most times in a specific window without losing the characteristics of LRLFU. However, the results that have been obtained from our simulations, as we will see in the next section, are not as good as we expected.

2.3 Performance Evaluation

2.3.1 Performance Metrics

In order to achieve efficient use of the proxy cache, the system has to store the most valuable videos and the cache should have the ability to quickly adapt its content to popularity changes. So, the system’s goal is to achieve to reduce the amount of data transferred from the content server to proxy. An important system performance metric is the Video-Hit Ratio (VHR) which is defined as the total number of times a video is found in the proxy cache when a request for that video arrives, divided by the total number of videos that have been requested during the duration of the simulation. The VHR metric gives us an indication of network traffic because the bigger the VHR is, the less videos need to be transferred from the content server and consequently the less network bandwidth is required.

A second important performance metric is the Fraction of Delayed Starts which is defined as the percentage of the total video requests that have been served with delay, divided by the total number of requests that have been served by the system during the duration of the simulation. There is a delayed start for a requested video if the video is not stored in the cache when requested and the proxy needs to contact content server to bring it. We want to reduce the Fraction of Delayed Starts as much as possible, because if this metric take big values it means that the cache management policy tends to store the wrong videos into the cache. The delay in the beginning of a video playback is very annoying to the client and in the end some clients may not be willing to experience it.

Although we introduce both the VHR and the Fraction of Delayed Start metrics, it is worth mentioning that in our case of full video caching these are complementary, that is, they always have a sum equal to one. This happens because in the policies examined in the Thesis we store whole videos in the cache instead of video segments.

2.3.2 Simulation Model

In the next section the simulation results of the four different policies introduced in section 2.2.2 are presented.

We assume that time is divided into equal length windows, where each window length is equal to 30 sec. The client requests for the videos, arrive according to a Poisson process, therefore the inter-arrival times are independent exponentially distributed random variables with mean $\frac{1}{\lambda}$, where λ is the Poisson arrival rate. The default value for $\frac{1}{\lambda}$ is equal to 5 sec, that is, on average we have 0.2 requests per second. Consequently, in each time window we have on average 30*0.2 = 6 requests arriving. We run our simulation for 30.000 windows which corresponds to about 10 days of real system operation. Also, in 30.000 windows we have on average 30.000 * 6 = 180.000 requests arriving.

As mentioned in section 1.2 the most commonly used and effective distribution to simulate the popularity of videos in video caching systems is the Zipf distribution. In our simulation study, the popularity of each of the V videos is assumed to follow a Zipf-like distribution Zipf (s,V), where s corresponds to the degree of skew and V to the total number of videos in the content server. Every video x, $x \in \{1,\ldots,V\}$ has a probability given by $p_x = \frac{c}{x^{1-s}}$, to be requested by the clients, where $c = \frac{1}{\sum_{x=1}^{V} \frac{1}{x^{1-s}}}$ is a normalization constant. For s=0 the distribution is highly skewed, while for s=1 the distribution is uniform with no skew. In our simulation the default value used for s is equal to 0.2.

In our system the content server size as well as the cache size are expressed in terms of the number of videos. We assume that there are V=2000 distinct videos of equal sizes that are stored in the content server. Also, the total capacity of the proxy cache is taken to be equal to 200 videos, which corresponds to 10% of the content server capacity.

In Table 2.1 below, all the parameters of the simulation in this Chapter together with their default values are given.

Parameters	*Definition (Default values)*
w	Window duration (30 sec)
$\frac{1}{\lambda}$	Mean request inter-arrival time (5 sec)
V	Number of distinct video titles (2.000)
s	Parameter s of Zipf distribution (0.2)
C	Total cache capacity (200 Videos)
Runs	Number of time windows in a simulation run (30.000)

Table 2.1 : System Parameters together with their default values

2.3.3 Simulation Results

In this section we present and discuss the performance results for the four cache management policies. In Figure 2.1 our first performance metric, the VHR, is presented versus time, while in Figure 2.2 the Fraction of Delayed Starts versus time is also presented. As we can see from the simulation results the LRLFU policy achieves the higher VHR and consequently the least Fraction of Delayed Starts. Our policy, the LRLFU_Aggressive, comes second with a small difference from the LRLFU. The LRU is quite worse than the LRLFU and the LRLFU_Aggressive and finally the NCCS_Modified comes last with very low VHR and quite high Fraction of Delayed Starts. As it can been seen from the results our idea for a more Aggressive LRLFU policy does not help to increase the VHR, while the NCCS policy performs very poorly.

Specifically, the LRLFU policy finds in the cache 47.9% of the total requested videos, while the LRLFU_Aggressive, that we proposed, finds in the cache 46.1% of the total requested videos. That means the LRLFU outperforms the LRLFU_Aggressive in terms of the VHR by 1.8%. This result shows us that the NCC factor that we have introduced in the numerator of the caching value fraction of the LRLFU_Aggressive policy does not help the cache to adapt better to changes of video popularity.

One can agree that this happens because in our system only 6 requests arrive on average within each time window, which means that the NCC counter takes small values. For that reason, we run another simulation the results of which are shown in Figure 2.3. In that simulation we set $\frac{1}{\lambda}$ = 0.25 sec which means that on average 4 requests arrive per second, consequently 4*30 = 120 requests per time window. We run this simulation only for 1.500 windows which corresponds to 180.000 requests in total. From the results of this scenario the NCCS_Modified policy again achieves very low VHR results quite similar to those of our default scenario. Also, in this scenario we observe a slightly reduced performance gap (1%) between LRLFU and LRLFU_Aggressive policies. Therefore, we conclude that the NCCS_Modified policy cannot capture the popularities of the videos and this is the main reason why the NCCS_Modified finds in the cache only 35.4% of the requested videos. It is worth mentioning here that in the studies in which the NCCS cache replacement policy has been proposed ([3]-[5]), a uniform distribution was used for the popularity of videos, instead of the Zipf distribution.

Finally, the differences in the VHR between the LRU and the two LRLFU methods are expected, as similar results have been obtained in [1], [2], and this is due to the fact that the RF factor which counts the frequency of the requests for a particular video helps the cache to better adapt to the popularities of the videos.

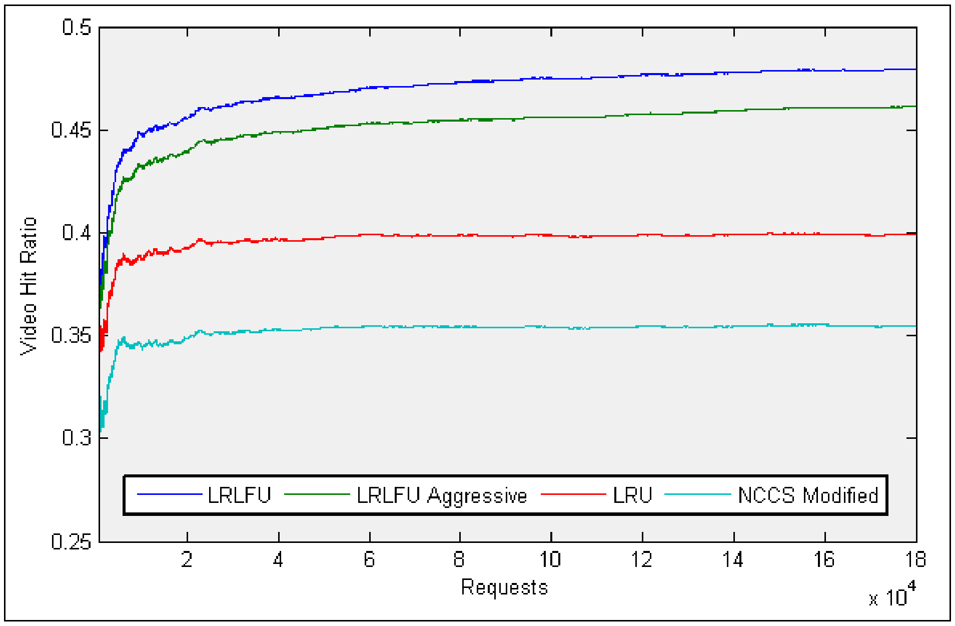

Figure 2.1 : Video-Hit Ratio versus time (expressed as number of requests served)

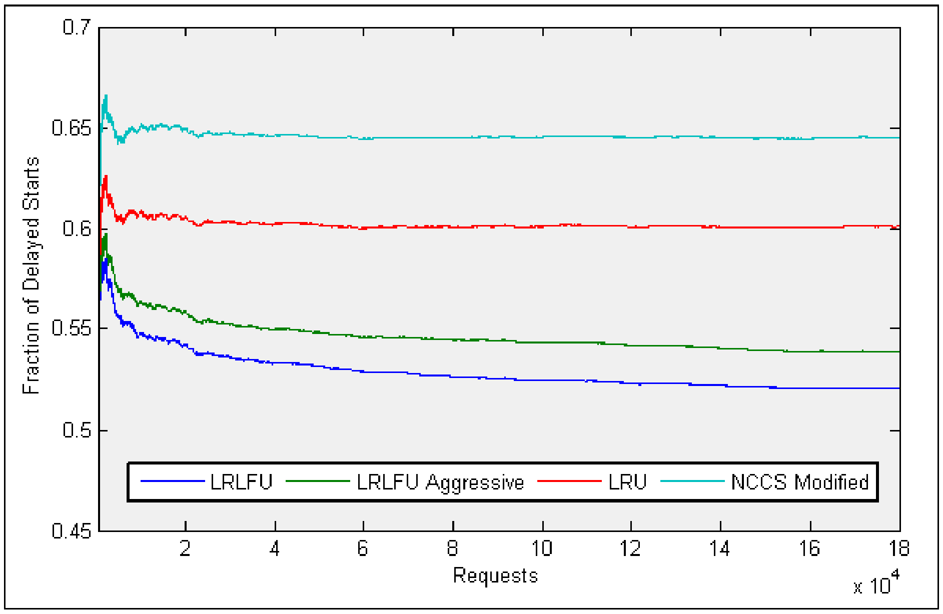

Figure 2.2 : Fraction of Delayed Starts versus time (expressed as number of requests served)

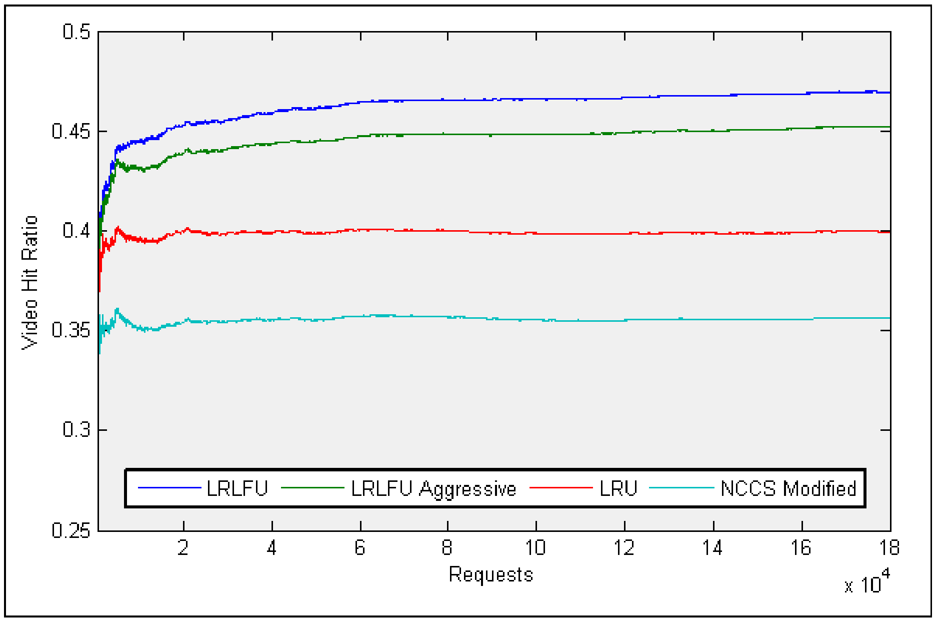

Figure 2.3 : Video-Hit Ratio versus time (expressed as number of requests served) for λ=4, Runs=1500

In the next three sections we examine the impact of different cache sizes, of different number of videos stored in the content server and the impact of different values of the skew parameter s of the Zipf popularity distribution.

2.3.3.1 The impact of different cache sizes

In this section, the effectiveness of the four cache replacement policies examined in this Thesis is presented for different values of the total cache size. The results for the VHR are shown in Figure 2.4 and for the Fraction of Delayed Starts in Figure 2.5. In this scenario we vary the cache size from 100 videos to 500 videos, while we maintain constant the total size of content server. This means that we vary the percentage of the videos stored into the cache from 5%-25% of the total videos in content server. From the results it is seen that the effectiveness of the four cache replacement policies follows the same trend as in section 2.3.3. When we store more videos in the cache, we expect that the VHR is improved and that the Fraction of Delayed Starts is reduced for all the cache management policies. From the results we see that by increasing the cache size the difference between the LRLFU_Aggressive and the LRLFU policies is reduced, but this reduction is very low. Specifically, when the cache stores 100 videos (5% of content server) the LRLFU is better than the LRLFU_Aggressive by 2.1%, while when the cache stores 500 videos (25% of content server) this difference is 1.1%. Instead, the difference between the NCCS_Modified and the LRU policies is increased, with the increase of the cache size. This difference is 3.6% for the smallest cache (with 100 videos) and becomes 4.8% for the biggest cache (with 500 videos). The LRLFU policy is the best method, and the NCCS_Modified the worst.

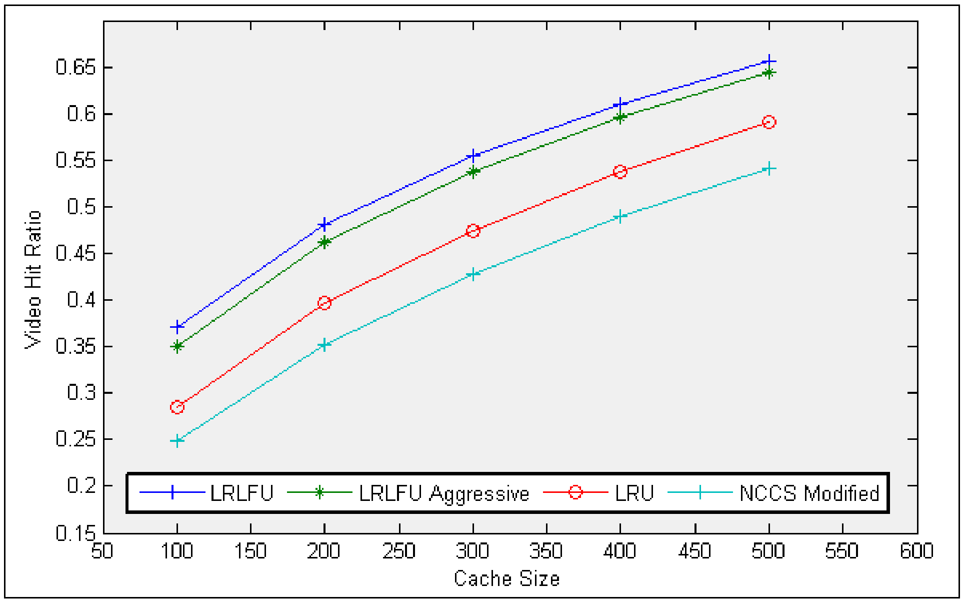

Figure 2.4 : Video-Hit Ratio versus cache size

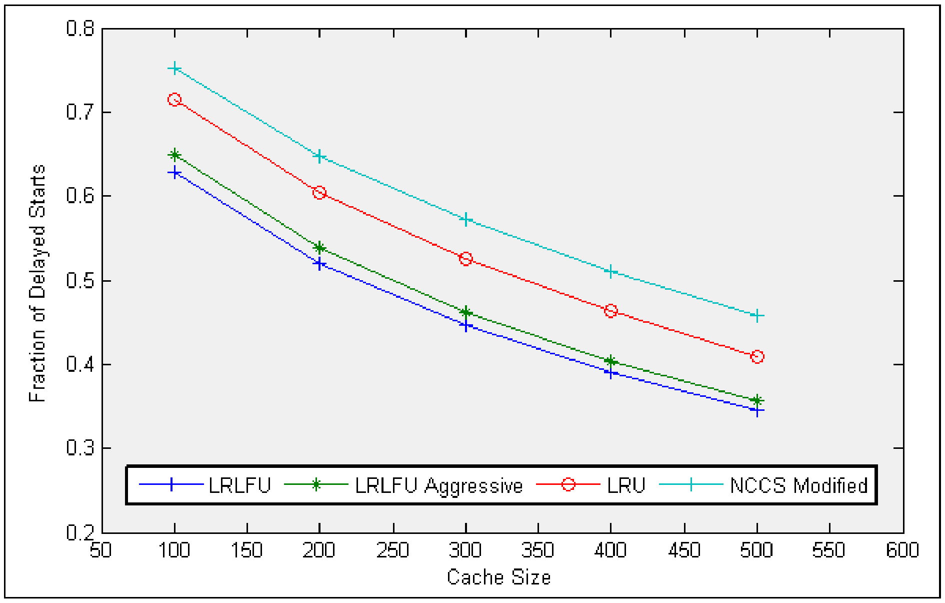

Figure 2.5 : Fraction of Delayed Starts versus cache size

2.3.3.2 The impact of different number of videos in the content server

In this section, we vary the total number of videos in content server from 1000 to 3000, maintaining the cache capacity equal to 10% of content server's capacity. The results for the VHR are shown in Figure 2.6, while those for the Fraction of Delayed Starts in Figure 2.7. When we increase the total number of videos in the content server, the VHR is also increased, while the Fraction of Delayed Starts decreases. These changes on the curves of VHR and of the Fraction of Delayed Starts are very slow. This is expected for two reasons. On the one hand, by increasing the total number of videos, the cache replacement policies have to choose the most popular videos among more videos. So there would be more requests to less popular videos, which most probably are not stored in the cache. The result of this fact is to slightly reduce the VHR and increase the Fraction of the Delayed Starts. On the other hand, by increasing the total number of videos, the cache size is increased, so as to continue storing 10% of the total videos. Therefore, there is more space in the cache for the requests for less popular videos we mentioned before. The result is that the increase of cache size prevails to the requests for less popular videos and there is a slow increase in VHR and a slow decrease in the Fraction of Delayed Starts. This is why we observe slow changes, instead of the sharp changes of the curves in section 2.3.3.1.

We also computed the Zipf cumulative probability mass function for the different numbers of total videos. We note that if the cache replacement policies were ideal and they stored only the most popular videos, the VHR for 1000 total videos in content server would be 52.4%, while for 3000 total videos would be 55%. From our results, we see that for 1000 total videos the VHR of LRLFU is 46.7% and the VHR of LRLFU_Aggressive is 45% and for 3000 total videos the corresponding results for LRLFU is 49.1%, while for LRLFU_Aggressive is 47.4%. Consequently, it is seen that these cache replacement policies achieve quite good VHR relative to the ideal VHR values. Instead, the NCCS_Modified policy achieves again very low results with the VHR to be equal to 33.5% for 1000 total videos and 36.9% for 3000 total videos. The LRU policy achieves performance results in between to those of the other methods as in the previous sections.

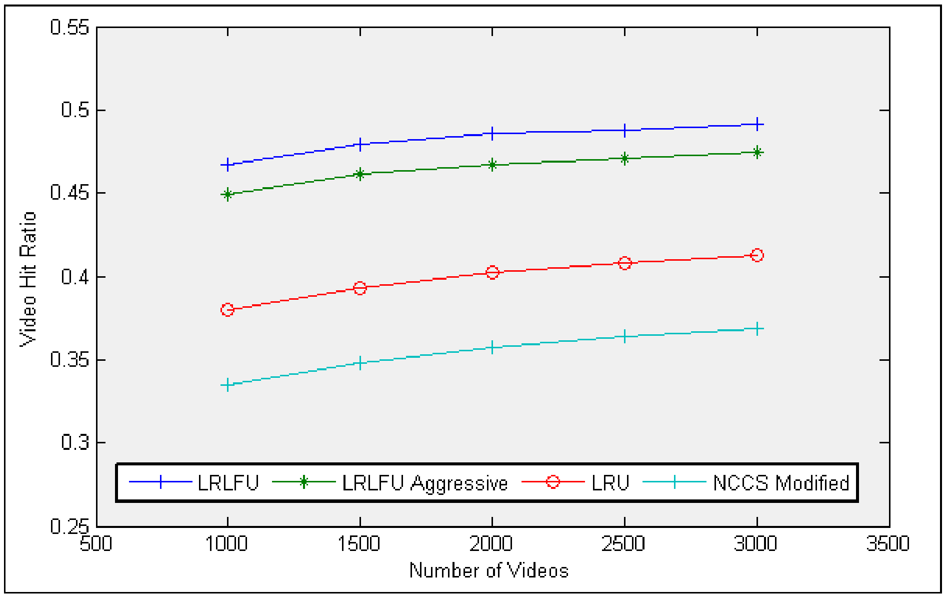

Figure 2.6 : Video-Hit Ratio versus number of total videos

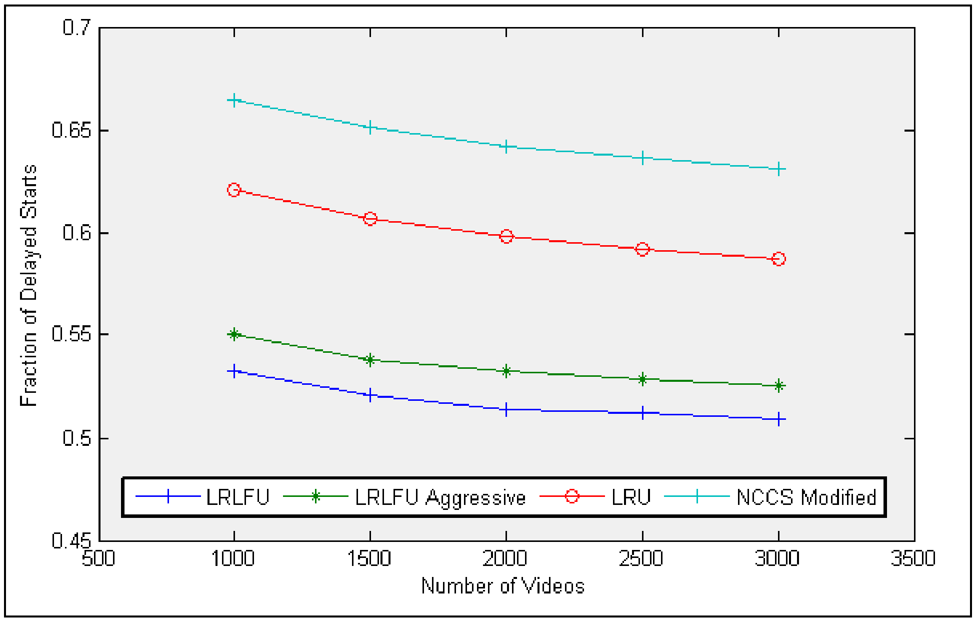

Figure 2.7 : Fraction of Delayed Starts versus number of total videos

2.3.3.3 The impact of different values of skew parameter in popularity distribution

The results in Figures 2.8 and 2.9 show the impact of the degree of the skew parameter in the Zipf video popularity distribution on the VHR and the Fraction of Delayed Starts. As already mentioned, small values of the parameter s correspond to a more skewed popularity distribution (more clients are interested in fewer videos), while higher values of the parameter s correspond to a more uniform popularity distribution (client's preferences are dispersed among a plethora of videos). So, when the parameter s approaches zero, it is easier to distinguish the most popular videos because their probabilities of appearance are higher. When the parameter s approaches one, the probabilities of appearance of each video are close to each other. This scenario is quite important because it can give us an indication for the "behavior" of the examined policies in a system where the clients have almost the same preferences for the various videos and in a completely different system where the clients have quite different preferences.

From the presented results we observe that as the parameter s increases the VHR decreases while the Fraction of Delayed Starts increases. For values of s smaller than 0.5 which corresponds to a skewed popularity distribution, we observe that the trend of the four cache replacement policies is the same as in previous simulations. The LRLFU policy achieves the best results in term of VHR and of the Fraction of Delayed Starts, the LRLFU_Aggressive is a little worse while the NCCS_Modified achieves quite low cache efficiency. However, as the value of s increases the four policies achieve almost the same results. For s equal to 0.8 which corresponds to a popularity distribution very close to uniform, the LRLFU policy achieves a VHR equal to 10.8%, the LRLFU_Aggressive equal to 10.6% while the other two methods achieve VHR equal to 10.5%. These results show us that the efficiency of all the cache replacement policies examined is very low for big values of s.

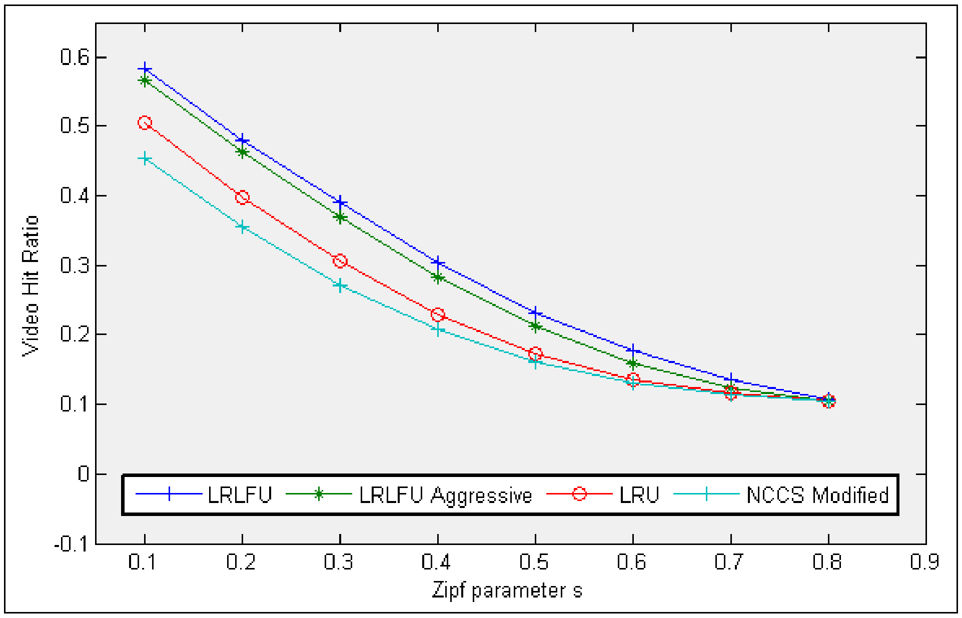

Figure 2.8 : The impact of skew in Zipf video popularity on VHR

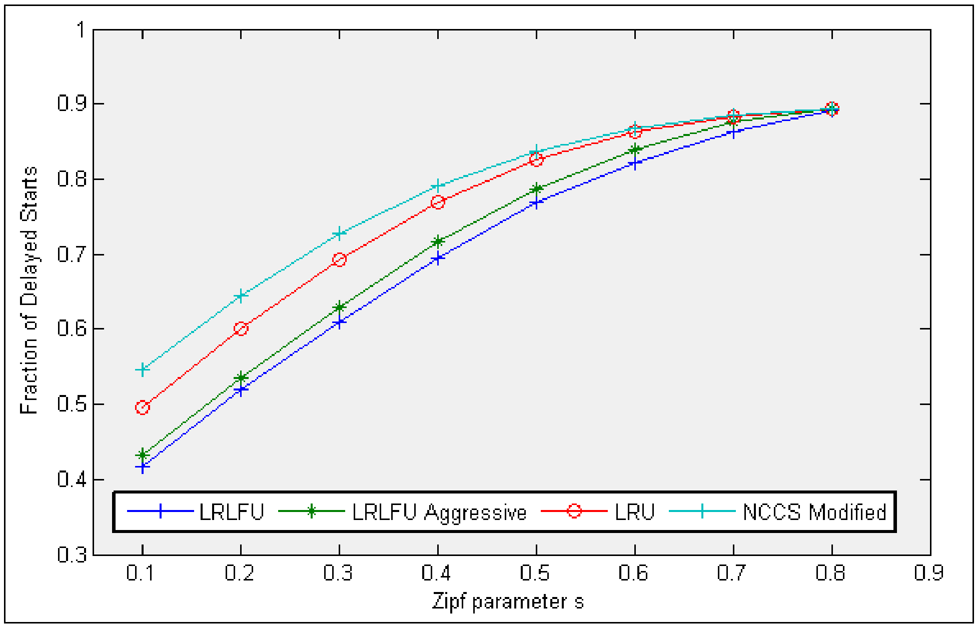

Figure 2.9 : The impact of skew in Zipf video popularity on Fraction of Delayed Starts

2.4 Conclusions

From the simulation results presented in this Chapter we conclude that the LRLFU_Aggressive policy we proposed does not achieve the expected results. Instead, the LRLFU policy outperforms the LRLFU_Aggressive in all the scenarios we examined. Also, the NCCS_Modified policy does not perform well in our system model in which we assumed a Zipf distribution for the video popularities and fixed file sizes. The only case in which all the four policies achieve almost the same performance results is when we assumed a Zipf distribution with s equal to 0.8, which corresponds to a distribution very close to uniform. Finally, the LRU policy although as we explained in section 1.2.2 is an easy method to implement, achieves not so good performance.

Chapter 3 : Design and Performance Evaluation of Variable Video Size Cache Management Policies

3.1 Introduction

In this Chapter we examine various algorithms for storing videos in proxy caches. In contrast to Chapter 2, this Chapter we assume variable video sizes. With this change we can implement the NCCS cache replacement policy as it has been proposed in [3] and [4], and we propose a new algorithm which performs better in the case of variable video sizes. Also, we implement two cache management policies from Chapter 2 (namely the LRLFU and LRU). We decided to not implement the LRLFU_Aggressive policy in this Chapter. We took this decision because we expected that LRLFU_Aggressive will outperform LRLFU. However, the simulation results of Chapter 2 have shown the opposite. Furthermore, we slightly change the cache management policies in order to support variable video sizes and we continue using the same network topology as in Chapter 2. The performance metrics are the same as in Chapter 2, that is, the Video Hit Ratio and the Fraction of Delayed Starts, while in one simulation experiment we introduce a different performance metric, the Byte Hit Ratio. We also introduce some new characteristics in our simulation model in order to achieve to implement the variable video sizes. In the next sections we discuss the changes which have been adopted in this Chapter and we present and discuss the simulation results.

3.2 System Model

3.2.1 Network Topology

In this Chapter we have used the same network topology as in Chapter 2 with a far-distance content server, a single proxy server and client devices. For the details refer to section 2.2.1.

3.2.2 Cache Management/Replacement Policies

As we already mentioned in section 3.1, we have implemented four different algorithms in our simulation study with the aim to compare them. In this section we discuss the cache management policies of these algorithms and the changes which have been made in order to support variable video sizes. We also discuss the new algorithms that we have implemented for cache replacement, which are the NCCS and a modification of the LRLFU which is more suitable for systems with variable video sizes. We will refer to the latter algorithm as LRLFU_Modified. The algorithms LRU and LRLFU which have also been implemented in this Chapter are the same as in Chapter 2. For more details on the latter algorithms refer to section 2.2.2.

Since in this Chapter we assume that the videos have variable sizes and that content server stores 2000 videos, the cache size cannot be expressed in terms of the number of videos stored. However, we assume that the cache can store 10% of total volume of the videos which are stored in the content server. Consequently, during the duration of the simulation the cache content expressed in terms of the number of videos stored varies according to the size of videos which are stored by each algorithm. For example, in our simulation the LRU and the LRLFU policies whose caching value does not depend on the file size store about 200 videos, the LRLFU_Modified policy whose caching value depends on the file size stores about 240 videos, while the NCCS policy whose caching value heavily depends on the file size stores about 420 videos. In this Chapter in which variable video sizes are supported, when a client request arrives to proxy server the proxy server acts as follows. First it looks for the video in its cache and if the requested video is stored in the cache, the caching value is updated and the proxy serves the client. Otherwise, the requested video has to be transferred from the content server and stored into the cache. If the cache is not full and there is enough space, the video is stored. Otherwise, the cache replacement policy is invoked and the four algorithms act as follows. First they look for the less popular video with the minimum caching value which is the video victim and it has to be removed from the cache. When the video victim is removed from the cache, there is not always the necessary space in the cache to store the requested video because the video victim may be smaller than the requested video. For that reason we have implemented a recursive function which removes the number of videos required until there is enough space for the requested video to be stored. Sometimes the video victim is quite big and the requested video is much smaller. The result is that after such a replacement takes place, the subsequent requested videos, depending on their sizes, could be stored without invoking the replacement algorithms. The new videos are assigned a caching value according to the replacement algorithm in use. The remaining characteristics of the cache management policies are the same as in Chapter 2 (section 2.2.2).

In this Chapter the NCCS replacement policy has been implemented as it has been introduced in the literature in [3], [4] in our system model, which assumes a Zipf popularity distribution for the videos. The caching value that we have used is given by the expression $\frac{\mathrm{NCC}}{\mathrm{FS}}$, where NCC is the number of common clients requesting that particular video file and FS is the video size. So, here we can see the performance of NCCS policy in a different system setup than the one examined in the literature. From the simulation results presented in the next sections we can see that the NCCS policy achieves again poor performance.

The new cache replacement policy we propose in this Chapter consists a combination of the NCCS and the LRLFU policies and is referred to as the LRLFU_Modified policy. The idea is to make the LRLFU algorithm more adaptable to variable video sizes. For that reason the caching value we propose for the LRLFU_Modified policy is given by the expression $\frac{\mathrm{RF}}{\mathrm{T-T'}} * \frac{1}{\mathrm{FS}}$, where the expression to the left of the multiplication is the caching value of LRLFU policy and FS denotes the video file size. In that way we want to achieve to store the most valuable videos which have also the right size, so as to efficiently use the cache storage. At the same time, if a large size video is quite valuable it will remain into the cache, while a small video which is not valuable will be removed. From the simulation results presented in the next sections we see that this algorithm achieves better performance than that of the LRLFU algorithm.

3.3 Performance Evaluation

3.3.1 Performance Metrics

In this Chapter we have used the same performance metrics as in Chapter 2, namely the Video Hit Ratio and the Fraction of Delayed Starts. For the details of these two performance metrics refer to section 2.3.1.

Furthermore, the system model with the variable video sizes in this Chapter gives us the ability to examine another performance metric which is the Byte Hit Ratio (BHR). The Byte Hit Ratio is similar to Video Hit Ratio, with the difference that in the BHR we count the total number of Bytes found in the cache in contrast to the total number of videos that is counted in the VHR. So, the BHR is defined as the total number of bytes found in the proxy cache when a request for a video arrives, divided by the total number of bytes for all the videos that have been requested during the duration of the simulation. We have chosen to examine this performance metric because we want to see the changes, if any, on the performance of the four cache replacement algorithms when we count bytes of videos found in the cache instead of number of videos. The BHR performance metric has been also examined in the literature, [1]-[4]. However in our system model in which we replace entire videos and no video segmentation is used, the VHR performance metric is more appropriate than BHR. It happens because the VHR counts entire videos while the BHR counts bytes, which correspond to segments of videos. For that reason we have chosen to run only one simulation experiment for BHR, simply to observe the changes.

3.3.2 Simulation Model

Our simulation model in this Chapter is similar to the simulation model in Chapter 2, but we have added some characteristics in order to support variable video sizes. In this section we mention the characteristics which remain the same and we discuss the characteristics that have been added.

As in Chapter 2, we assume here that the time is divided into time windows and the client requests arrive according a Poisson process with the same parameters. Also, we run our simulation experiments for the same number of time windows with those in Chapter 2. Finally, we again assume that there are 2000 distinct videos in the content server and that the popularity of these videos is given by a Zipf-like distribution with the same parameters as in Chapter 2.

In our simulation model in this Chapter we have added a normal distribution to model the variable video sizes. As mentioned in section 1.2.3 the most common distribution used in the literature for varying video sizes is the uniform. However, we chose the normal distribution because we assume that most of the videos have sizes near the mean of the video size distribution and only a few videos have considerably larger or smaller sizes. Furthermore, with the normal distribution we can examine many scenarios by simply changing the standard deviation of the video size distribution (which is denoted by S_d in this Chapter). We further assume that the mean of the normal distribution is equal to 500 MB, while the default value of standard deviation is 200 MB. A histogram of the distribution we have used with the above parameters is shown in Figure 3.1. Notice that we have filtered the values of video sizes to not take values bigger than 1000 MB and smaller than 10 MB. We decided to filter the video sizes as above to support cases of higher S_d values than 200 MB.

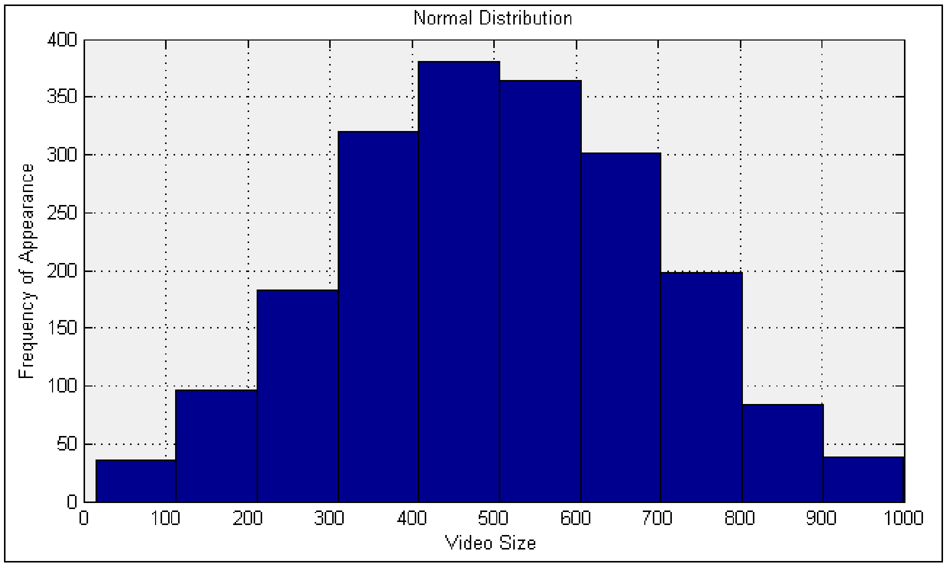

Figure 3.1 : Normal Distribution for Variable video sizes with mean = 500 MB, S_d = 200 MB

As we can see from Figure 3.1 the majority of the videos has video sizes from 400 MB to 600 MB while only a few videos are smaller than 200 MB or larger than 800 MB. As the standard deviation is increased the video sizes tend to be uniformly distributed, while as we decrease the standard deviation there are more videos with sizes near the mean size and less videos with bigger or smaller sizes.

Furthermore, it is worth noticing that since the content server stores 2000 video files this corresponds to a total capacity equal to 2000*500 = 1.000.000 MB or 1 TB on average. Consequently, the total capacity of the cache, which is assumed to be equal to 10% of the content server, is equal to 100.000 MB or 100 GB on average.

In Table 3.1 below, all the parameters of the simulation model for this Chapter together with their default values are given.

Parameters	***Definition (Default values)***
w	Window duration (30 sec)
$\frac{1}{\lambda}$	Mean request inter-arrival time (5 sec)
V	Number of distinct video titles (2.000 ➔ about 1 TB)
s	Parameter s of Zipf distribution (0.2)
C	Total cache capacity (about 100 GB)
m	Mean of the Normal Distribution for the video sizes (500 MB)
S_d	Standard Deviation of the Normal Distribution for the video sizes (200 MB)
Runs	Number of time windows in a simulation run (30.000)

Table 3.1 : System Parameters together with their default values

3.3.3 Simulation Results

In this section we present and discuss the performance results for the four cache management policies. In Figure 3.2 the VHR is presented versus time, while in Figure 3.3 the Fraction of Delayed Starts versus time is also presented. As we can see from the simulation results, the LRLFU_Modified algorithm that we propose in this Chapter achieves the higher VHR and consequently the lowest value for the Fraction of Delayed Starts. The LRLFU policy comes second with small difference from our proposed policy. The LRU and the NCCS algorithms achieve results comparable to the corresponding results in Chapter 2. The LRU policy is quite worse than the LRLFU policy and the LRLFU_Modified policy while the NCCS policy comes last with very low VHR and quite high Fraction of Delayed Starts. As it can been seen from the presented results, the NCCS policy performs very poorly even though it takes into account in its caching value the sizes of the various video files. Furthermore, our idea to introduce the video size in the caching value of the LRLFU_Modified help this policy to outperform the LRLFU in terms of the VHR and the Fraction of Delayed Starts.

Specifically, the proposed LRLFU_Modified policy finds in the cache 50.7% of the total requested videos, while the LRLFU policy finds 48.7%. That means the LRLFU_Modified policy outperforms the LRLFU both in terms of the VHR and the Fraction of Delayed Starts, by 2%. This result indicates that the LRLFU_Modified policy stores in the cache the videos which are the most popular and at the same time have the best suitable size. This explains why the LRLFU_Modified policy better adapts both in terms of popularity and video size changes.

Also the results show that both the LRU (which has a VHR equal to 40.3%) and the LRLFU policies achieve almost the same results with the corresponding ones in Chapter 2 in terms of the VHR and the Fraction of Delayed Starts. The latter is expected because in these two policies the variable video size is not taken into account in their caching values.

Finally, the NCCS policy finds in the cache only 29.1% of the total requested videos. Our expectation was that the NCCS policy would perform better in the system model of this Chapter relative to that of Chapter 2 because its video caching value takes into account the variable video size. However, as it can been seen from the results the NCCS policy cannot capture at all the popularities of the various videos when these popularities follow a Zipf distribution.

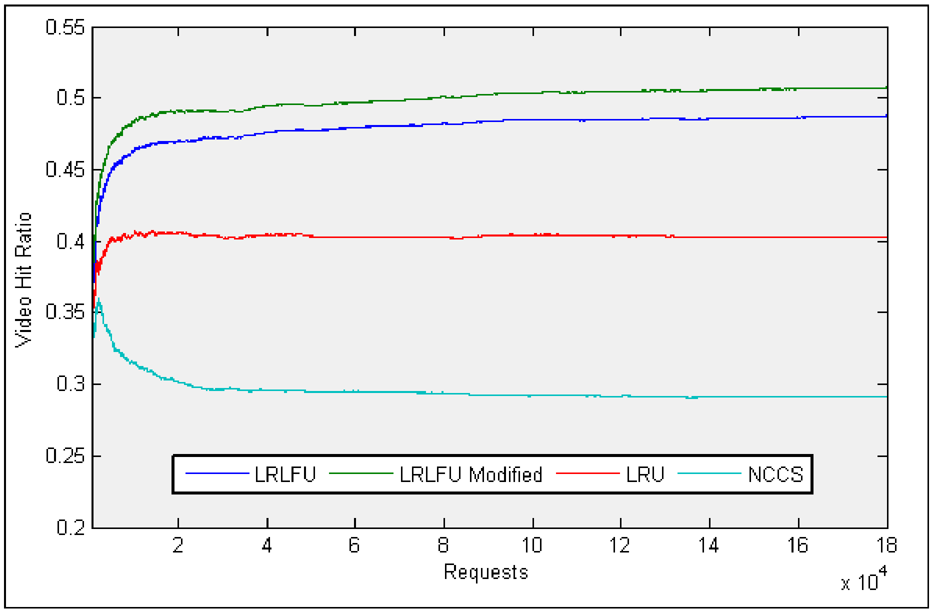

Figure 3.2 : Video-Hit Ratio versus time (expressed as number of requests served)

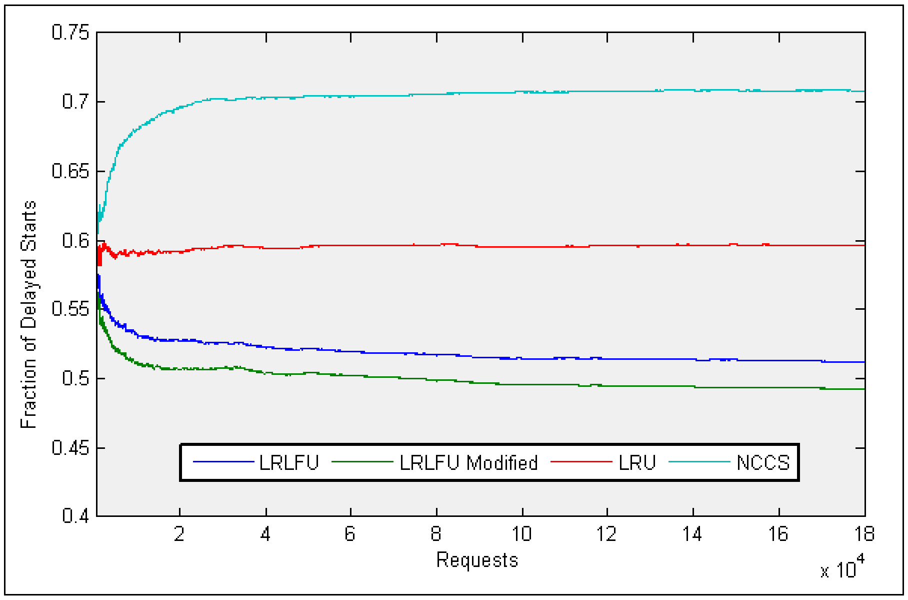

Figure 3.3 : Fraction of Delayed Starts versus time (expressed as number of requests served)

In the next four sections we examine the impact of different cache sizes, different number of videos stored in the content server, different values of the standard deviation of the Normal distribution of file sizes and of different values of the skew parameter s of the Zipf video popularity distribution. Finally, in section 3.3.3.5 we present the results of a simulation scenario with the new performance metric, the BHR.

3.3.3.1 The impact of different cache sizes

In this section, the effectiveness of the four cache replacement policies examined in this Chapter is presented for different values of the total cache size. The results for the VHR are shown in Figure 3.4 and those for the Fraction of Delayed Starts in Figure 3.5. As in the corresponding scenario of Chapter 2, we vary the percentage of the bytes of videos stored in the cache from 5% to 25% of the total bytes of videos stored in content server, while we maintain constant the total size of the content server. In this Chapter the cache size and the total size of the content server are measured in bytes and each cache replacement algorithm stores different number of videos in the cache according to its caching value function.

From the presented results, it can be observed that the effectiveness of the four cache replacement policies follows the same trend as in section 3.3.3. It is expected that as the cache storage is increased the VHR is improved and the Fraction of Delayed Starts is reduced for all the cache management policies. We also observe that by increasing the cache size, the difference between the LRLFU_Modified and the LRLFU policies remains almost constant with the LRLFU_Modified being better by about 1.4%. Instead, the difference between the NCCS and the LRU policies is increased, with the increase of the cache size. This difference is 12.6% for the smallest cache (which corresponds to 5% of content server) and becomes 17.1% for the biggest cache (which corresponds to 25% of content server). The LRLFU_Modified policy remains the best and the NCCS the worst with big differences between each other.

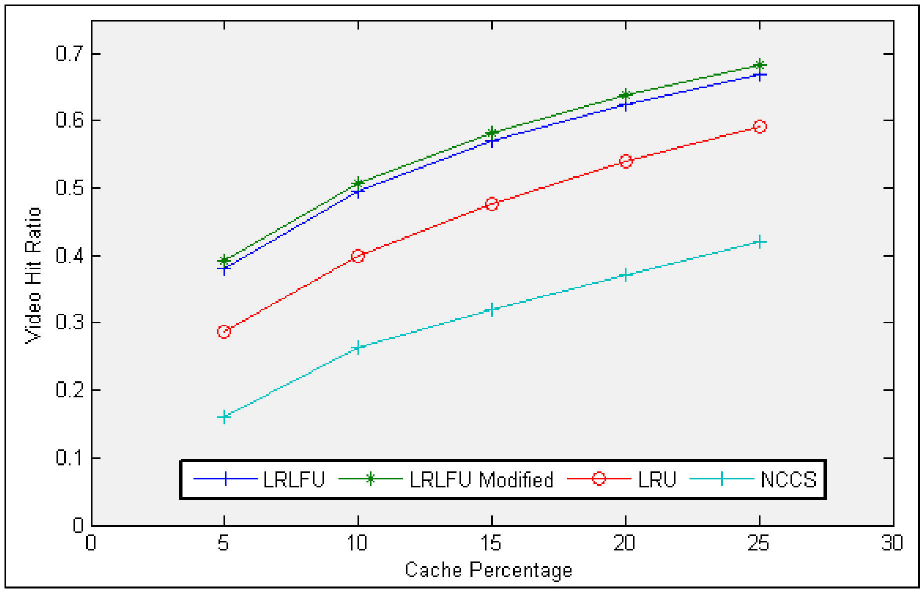

Figure 3.4 : Video-Hit Ratio versus cache size

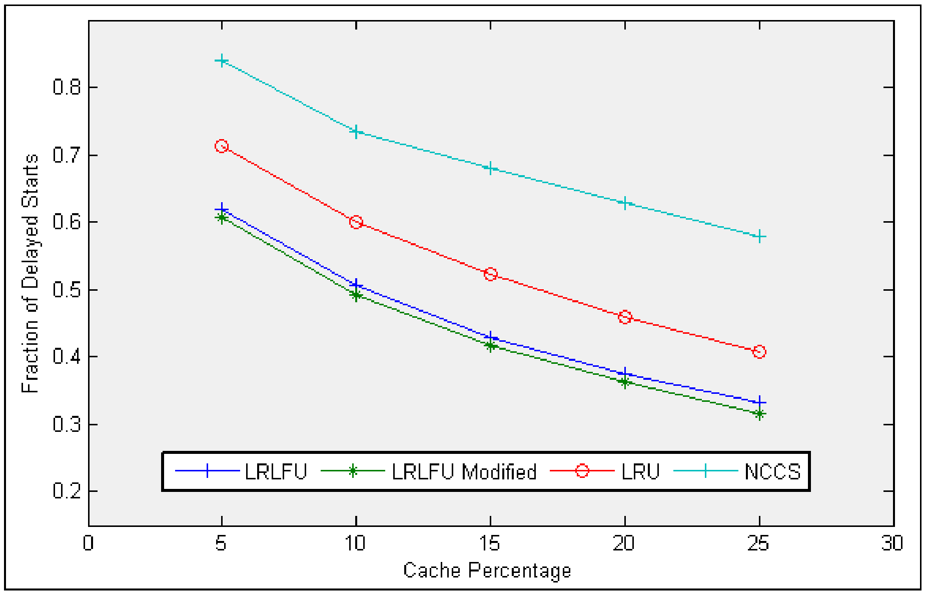

Figure 3.5 : Fraction of Delayed Starts versus cache size

3.3.3.2 The impact of different number of videos in the content server

In this section, we vary the total number of videos in content server from 1000 to 3000, while maintaining the cache capacity equal to 10% of content server's capacity. In Figures 3.6 and 3.7 the results for the VHR and for the Fraction of the Delayed Starts are given. We observe that by increasing the total number of videos in the content server the results of the VHR and the Fraction of Delayed Starts remain almost the same for each of the four cache management policies. Specifically, from the results of the three first cache management policies, namely the LRLFU, the LRLFU_Modified and the LRU, we see that the VHR is slightly increased and the Fraction of Delayed Starts is slightly decreased. From the results of the NCCS policy we observe some ups and downs which also are very slow.

More specifically, the NCCS with 1000 videos stored in the content server finds in the cache 24.3% of the requested videos while with 3000 videos in the content server this percentage becomes 24.9%, an increase of about 0.6%. Instead, in the LRU policy which achieves the highest increase of the VHR as the total number of videos in the content server increases, the VHR from 1000 to 3000 videos is increased by 3.4%. The LRLFU policies have almost the same increase in terms of VHR from 1000 to 3000 videos stored in the content server, about 2%.

These results are expected, as it was mentioned in the discussion in section 2.3.3.2, because there are two factors which affect the cache management policies when we increase the total number of videos. On one hand, when the total number of videos is increased, the cache size is also increased accordingly which is positive, on the other hand the cache management policies have to search for the popular videos among many more videos.

Finally, we observe that the performance hierarchy of the four cache management policies does not change, with the LRLFU_Modified policy to be the best and the NCCS policy the worst.

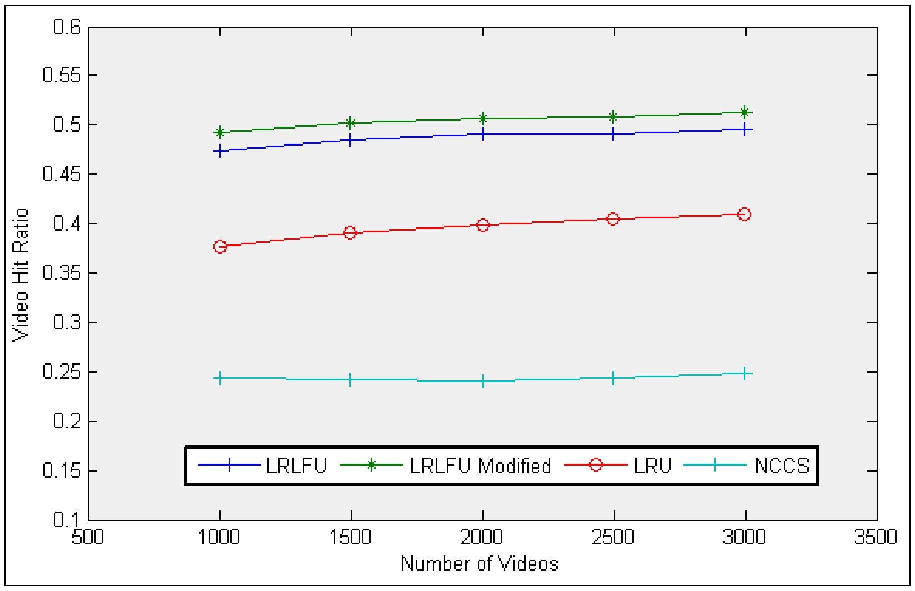

Figure 3.6 : Video-Hit Ratio versus number of videos

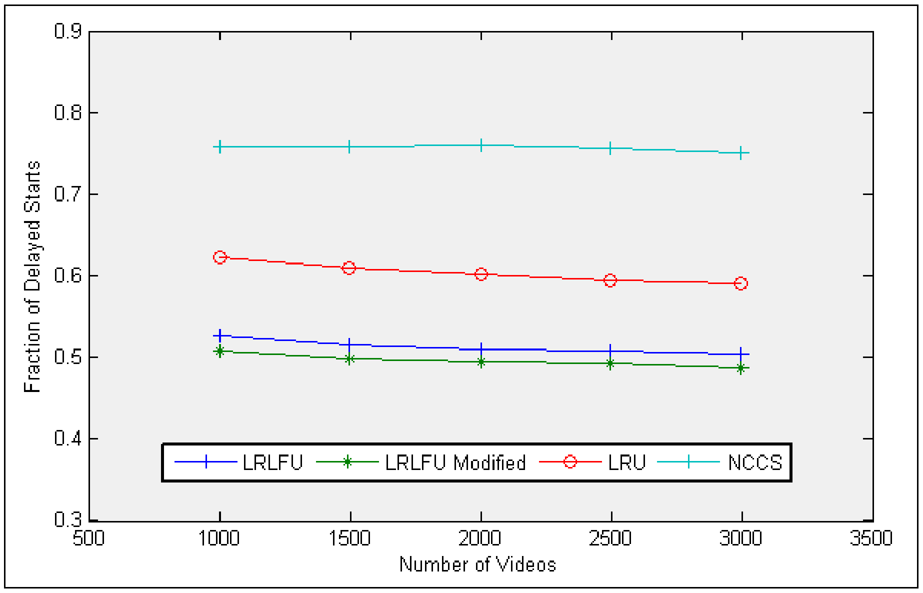

Figure 3.7 : Fraction of Delayed Starts versus number of videos

3.3.3.3 The impact of different values of the Standard Deviation of the Normal Distribution for the Video Sizes

In this section we vary the Standard deviation (S_d) parameter of the Normal Distribution, used for the simulation of the variable video sizes, from S_d equal to 100 to S_d equal to 800. The results in Figures 3.8 and 3.9 show the impact of the different values of S_d on the VHR and on the Fraction of Delayed Starts. As already mentioned, as S_d increases the video sizes tend to be more uniformly distributed while as we decrease S_d there are more videos with sizes near to the mean and less videos with bigger or smaller sizes. In our simulations when $S_d = 100$ the majority of the videos have sizes between 400 MB and 600 MB (the mean is equal to 500 MB) and there are few videos with size smaller than 400 MB or higher than 600 MB. Instead, when S_d approaches 800 MB all the video sizes between 10 MB and 1000 MB (remember that we filter out values smaller than 10 MB and higher than 1000 MB) have almost the same probability to appear. The simulation scenario in this section is an important one because we can examine the cache management policies in two different systems at the same time, one with normally distributed video sizes and the other with uniformly distributed video sizes.

From the presented results, we first observe that the LRLFU and the LRU policies achieve almost the same performance results in terms of the VHR and the Fraction of Delayed Starts for all the examined different values of S_d. This is expected because the video caching values of these two policies do not depend on the video sizes. Instead, the other two cache management policies examined, namely the LRLFU_Modified policy and the NCCS policy contain the video size in their video caching value functions. By increasing the parameter S_d, the VHR for these two policies is also increased and the Fraction of Delayed Starts is decreased. Specifically, for S_d =100 the LRLFU_Modified policy achieves a VHR equal to 48% while for $S_d = 800$ it achieves a VHR equal to 53.5%. The corresponding results for the NCCS policy are 19.5% and 32.8%, respectively. Therefore, we conclude that both the LRLFU_Modified and the NCCS policies achieve better performance in a system model with uniformly distributed video sizes than in a system with normally distributed video sizes with relatively low values of S_d.

Finally, it is worth mentioning that for $S_d = 800$ the LRLFU_Modified increases the difference from the LRLFU policy, compared with the results shown in the previous sections, and outperforms it by 4%. On the other hand the NCCS policy approaches the LRU policy.

Specifically, for $S_d = 100$ the difference between these two policies is 20.5% and for $S_d = 800$ this difference becomes 7.3%.

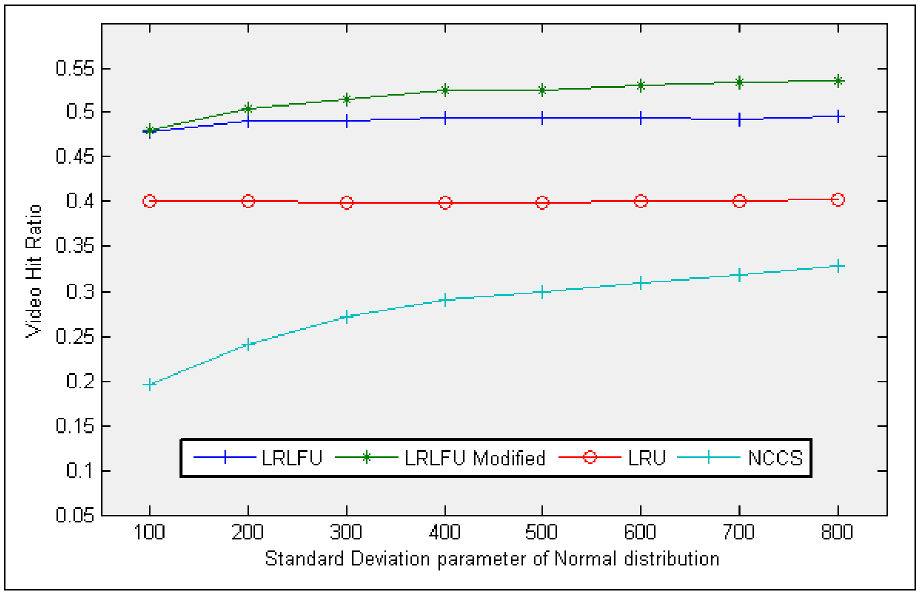

Figure 3.8 : The impact of the Standard Deviation of the Normal Distribution on VHR

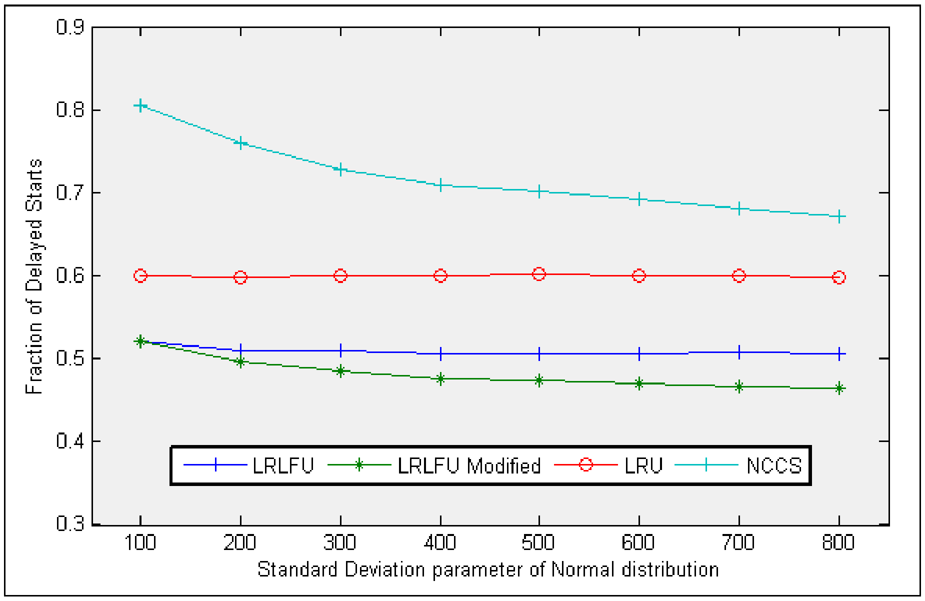

Figure 3.9 : The impact of the Standard Deviation of the Normal Distribution on the Fraction of Delayed Starts

3.3.3.4 The impact of different values of skew parameter in popularity distribution

The results in Figures 3.10 and 3.11 show the impact of the value of skew in the Zipf video popularity distribution on the VHR and on the Fraction of Delayed Starts. We have discussed in section 2.3.3.3 how the different values of the skew parameter affect the video popularity distribution.

From the results presented in this section we observe that as the parameter s increases each of the four cache management policies is affected in a different way. Firstly, the three of the four cache management policies, namely the two LRLFU policies and the LRU policy follow the same trend. For these policies, as the parameter s increases, the VHR is quickly decreased while the Fraction of the Delayed Starts is also quickly increased. However, for values of the parameter s between 0.5 and 0.8 in which the video popularity distribution tends to be uniform the VHR for the LRLFU_Modified and the LRU policies decreases slower than the VHR of the LRLFU policy. Specifically, for s = 0.1 the LRLFU policy outperforms the LRU policy by 8.7%, for s = 0.5 this difference becomes 6.7%, while for s = 0.8 these two policies almost achieve the same results (the difference is only 1%). Furthermore, the LRLFU_Modified policy outperforms the LRLFU policy only by 0.9% for s = 0.1 while for s = 0.8 the difference becomes 5%. Instead, the difference between the LRLFU_Modified policy and the LRU policy is pretty much maintained, it is reduced only by 3.7%, from 9.7% for s = 0.1 to 6% for s = 0.8.

The "behavior" of the NCCS policy is quite different than the other three policies. For values of the parameter s from 0.1 to 0.4 we observe a very slow decrease of the VHR and a very slow increase of the Fraction of Delayed Starts while for values of s from 0.4 to 0.8 the NCCS policy maintains the same performance. From the results we see that the NCCS policy for s = 0.5 outperforms the LRU policy by 3.5%, for s = 0.6 it outperforms the LRLFU policy by 2.3% and for s equal to 0.7 or 0.8 it outperforms all the other cache management policies. Specifically, the NCCS policy in comparison to LRLFU_Modified policy for s = 0.1 is worse by 34.6% while for s = 0.8 the NCCS policy outperforms the LRLFU_Modified by 4.2%. These results of NCCS are expected, because as we already mentioned the NCCS adapts better to uniformly distributed video popularities.

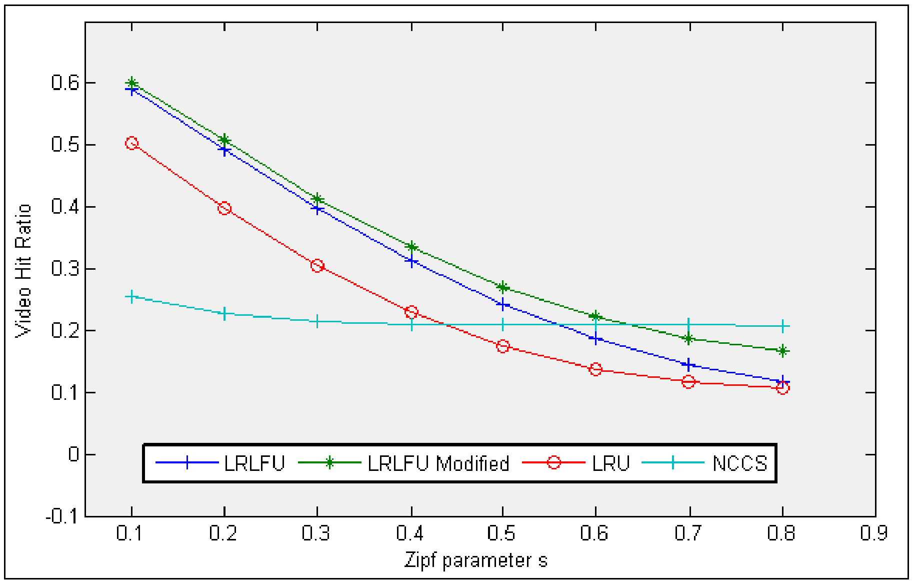

Figure 3.10 : The impact of the skew in Zipf video popularity distribution on VHR

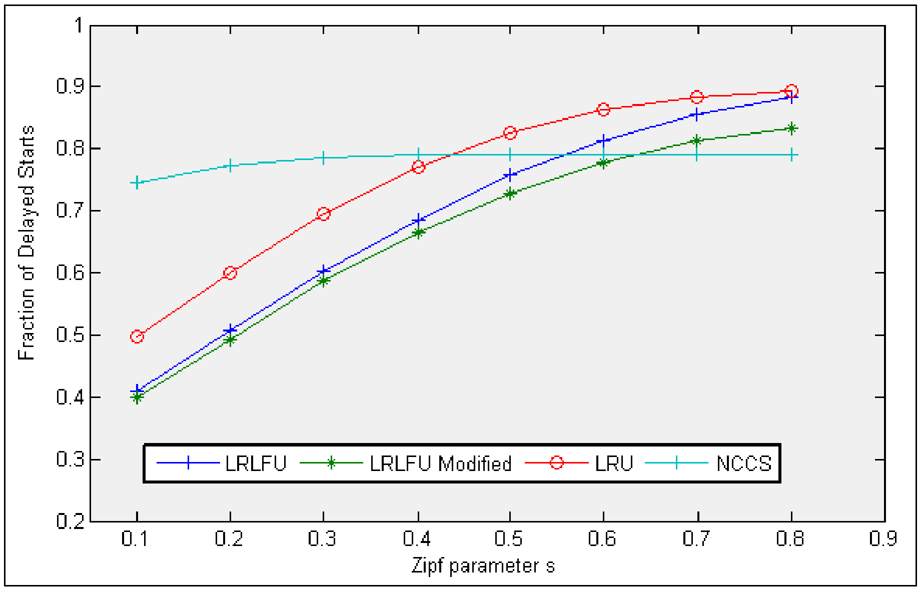

Figure 3.11 : The impact of the skew in Zipf video popularity distribution on the Fraction of the Delayed Starts

3.3.3.5 The examination of Byte-Hit Ratio versus time

In this section we examine the performance of the four cache management policies in terms of the Byte Hit Ratio (BHR) performance metric introduced in section 3.3.1. In Figure 3.12 the results of the BHR versus time are presented. We can notice several differences in comparison to the corresponding results of the VHR shown in Figure 3.2. First we observe that the performance of the NCCS policy is quite worse in terms of the BHR. Specifically, the VHR of the NCCS is 29.1% while the BHR of the NCCS is 12.3% which is a significant difference. This result is expected if we consider that the NCCS policy stores the videos with the smallest sizes. For the same reason we observe a reduction in the LRLFU_Modified policy in terms of BHR in comparison to VHR. This small reduction compared to the one in the NCCS case, is due to the fact that the caching value of the LRLFU_Modified policy is affected by more factors than the video size (i.e., the request frequency and the time elapsed since the last request of the videos). The reduction of the LRLFU_Modified policy is enough for the LRLFU policy to outperform it by 0.8%. Finally, the LRLFU and the LRU policies, the caching values of which do not depend on the video size, achieve the same BHR performance.

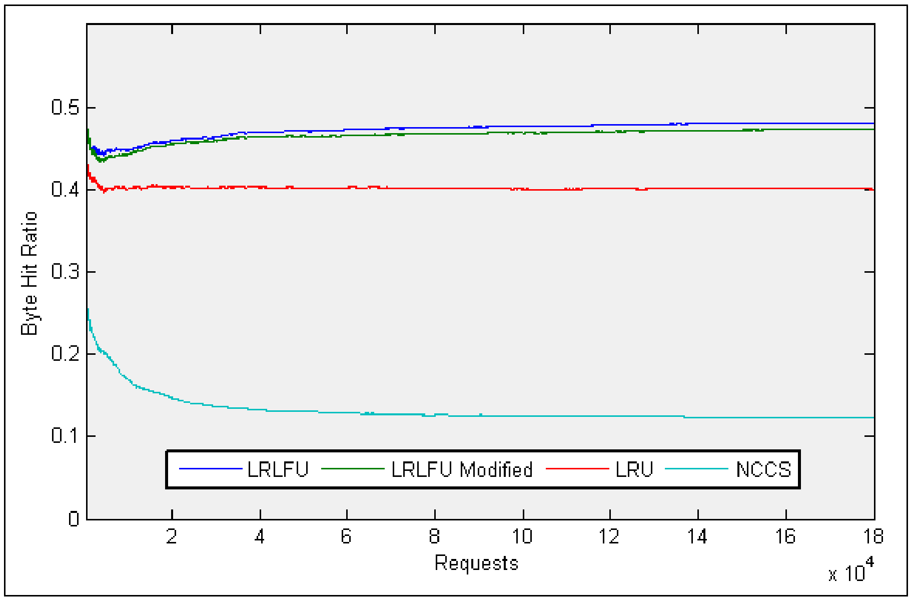

Figure 3.12 : Byte-Hit Ratio versus time (expressed as number of requests served)

3.4 Conclusions

From the results presented and discussed in this Chapter we conclude that our proposed LRLFU_Modified policy outperforms the LRLFU policy in terms of the VHR and the Fraction of Delayed Starts in all the examined scenarios. Also, the LRLFU_Modified policy improves its performance in comparison to the LRLFU policy in the scenarios examined in sections 3.3.3.3 and 3.3.3.4. Generally, when a system model with uniformly distributed video sizes or uniformly distributed video popularities is assumed, the factor of the video size in the video caching value of the LRLFU_Modified policy helps it to retain its performance at high levels, in comparison with the LRLFU and LRU algorithms. This can be shown from the comparison of the results in sections 3.3.3, 3.3.3.4 and 3.3.3.5. In section 3.3.3 in which our default scenario is examined, the LRLFU_Modified policy outperforms the LRLFU policy by 2%, while in section 3.3.3.4 this difference for $S_d = 800$ becomes 4% and in section 3.3.3.5 for $s = 0.8$ it becomes 5%.

Another important conclusion is that in our system model the NCCS policy achieves again very poor performance results. This happens because the NCCS policy cannot adapt well to Zipf video popularities with low values of the skew s. On the other hand, from the scenario in section 3.3.3.5 in which for $s = 0.8$ (which corresponds to an almost uniform video popularity distribution) we see that the NCCS policy outperforms all the other cache management policies.

Chapter 4 : Design and Performance Evaluation of a simple Collaborative Scenario

4.1 Introduction

In this Chapter we introduce a collaborative scenario with two proxy servers, where each of them is served by a different content server and serves different client communities. In our simulation study we implement different algorithms for each proxy cache. The first proxy cache is assumed to run the best cache replacement policy while the second proxy cache runs the worst cache replacement policy from the policies examined in Chapter 3. Therefore, the first proxy cache uses the LRLFU_Modified policy and the other proxy cache uses the NCCS policy. Furthermore we assume that there is an overlap between the video contents of the two content servers. We have also adapted the above two cache replacement policies to our collaborative scenario. We have used the same system model as in Chapter 3 with variable video sizes with the necessary changes in order to support the collaborative scenario and the same performance metrics as in Chapters 2 and 3. In the next sections we describe the system considered in this Chapter and we present its performance evaluation, while we also discuss the changes which have been adopted relative to Chapter 3 in order to support the collaborative scenario.

4.2 System Description

4.2.1 Network Topology

The system architecture that we consider for the collaborative scenario of this Chapter is shown in Figure 4.1. Our system consists of two far-distance content servers, two proxy cache servers and two different client communities, one for each proxy cache server. The two content servers in general store different video files with an overlap between their contents. Each proxy cache server is located close to its client community and sends requests for videos to its content server. As it is seen from the Figure 4.1 the LRLFU_Modified proxy cache server communicate only with Content Server 1 and the NCCS proxy cache server communicate only with Content Server 2, to request for the video files that they do not have in their cache. Each proxy cache server receives requests from different client communities and it caches a fraction of its content server files, which are the most popular based on their caching values. The procedure for each proxy cache when a request for a video arrives is described next. If the proxy has the video file requested into its cache, it transmits it to the client and the video file remains in the cache. In that case we have a request hit and no delay start is experienced by the client. If the proxy has not the video file requested, then it first looks for the file in the other proxy cache in the collaborative scheme which is denoted by the cycle in Figure 4.1. If a proxy finds the requested video file in the other proxy cache, then the proxy brings the file in its cache without delay and it serves the client. So, in this case we also assume that we have a request hit. The only case that there is a request miss and the proxy has to contact its content server to bring the video file requested, is when the requested video does not exist in either of the two proxy caches. In that case the client experiences a delayed start because the latency between the content server and the proxy is quite significant. When a new video file arrives, either from the other proxy in the collaborative scenario or from the content server, it is always stored in the cache by removing other video files, if necessary. Finally, we assume that the bandwidth between the proxy and the clients is sufficient to support video streaming with negligible latency.

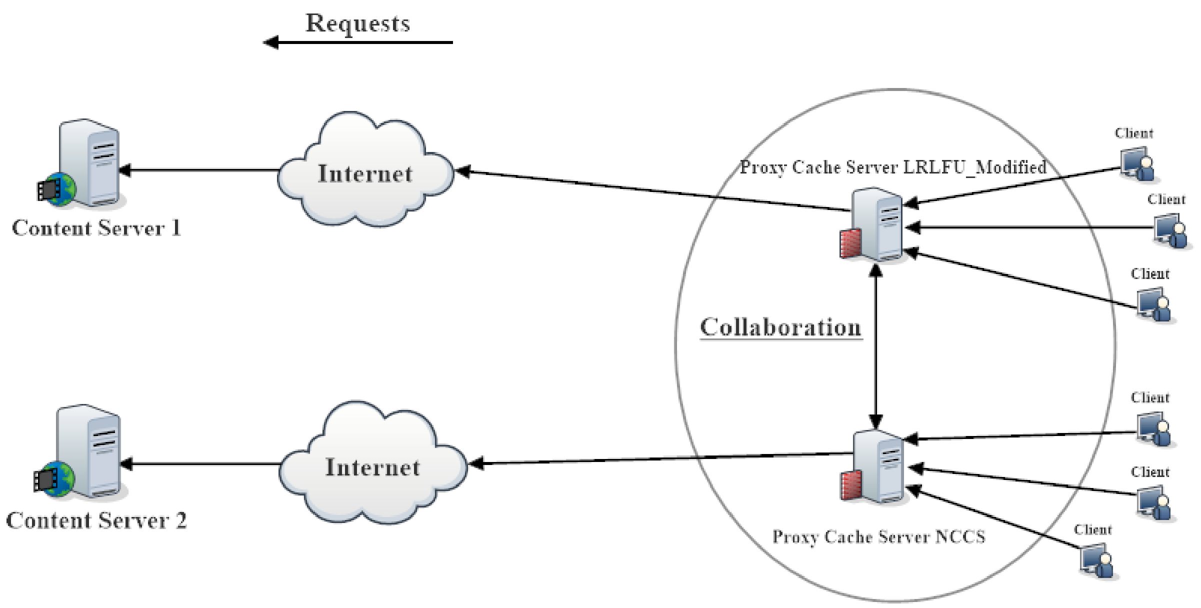

Figure 4.1 : Collaborative Scenario

4.2.2 Cache Management/Replacement Policies

As mentioned in section 4.1, we have implemented different cache replacement algorithm for each of the two proxy caches in order to achieve efficient use of memory of the entire system. In this section we discuss the cache management algorithms that have been used and the changes which have been implemented for them to support the collaborative scenario. For cache replacement, we have adopted the best and the worst cache replacement policies from the policies examined in Chapter 3. These are the NCCS and the LRLFU_Modified algorithms. For more details of these algorithms the reader is referred to section 3.2.2.

Most of the characteristics that we have assumed for the cache management policies in this Chapter are the same with the ones in Chapter 3. Consequently, assuming that each of the two caches can store 10% of the content of its content server, the LRLFU_Modified policy stores about 240 videos and the NCCS policy stores about 420 videos in total (for an explanation refer to the discussion in section 3.2.2). In this Chapter in which the collaborative scheme is assumed, when a client request arrives to a proxy server, the proxy server acts as follows. First it looks for the video in its cache and if the requested video is stored in the cache, the caching value is refreshed and the proxy serves the client. If the requested video is not stored in the cache, the proxy first looks for the video in the other proxy cache before it contacts its content server. If the proxy finds the requested video in the cache of the other proxy, it has to free the necessary space by removing other videos in order to bring the requested video in its cache. We implement this by running a recursive function until the necessary space is freed to store the new video. The new video gets a caching value according to the replacement algorithm used. In the case that the proxy server does not find the video in its cache or in the other's proxy cache, it has to contact to content server with a delay cost. The rest of this procedure is exactly the same with the one discussed in section 3.2.2 of Chapter 3.

It is worth mentioning that the idea to use two different cache management policies in our collaborative scenario may seem quite unusual. Our idea was based on the expectation that had we implemented the same policies in each of the two proxies, the result would have been that the two proxies would store almost the same videos in their caches. Therefore, there would not be much help from one proxy to the other. Instead, with the collaborative scheme introduced here with two very different cache management policies used by each of the two proxies, the collaboration between them would be significant. This is due to the fact that these two cache management policies store very different videos in their caches, thus it is quite probable to find a requested video in one of the two proxies and transmit it to the requesting client without delay cost.

4.3 Performance Evaluation

4.3.1 Performance Metrics

In this Chapter we have used the same performance metrics with the ones in Chapters 2 and 3, namely the Video Hit Ratio and the Fraction of Delayed Starts. For the details of these two performance metrics the reader is referred to section 2.3.1.

4.3.2 Simulation Model

The simulation model in this Chapter supports the same characteristics with the simulation model in Chapter 3 with the variable video sizes, however we have added some extra features which are necessary for the implementation of the collaborative scenario. In this section we mention the common characteristics with the ones in Chapter 3, and we discuss the extra features.

As in Chapters 2 and 3 we assume here that the time is divided into time windows of equal length and that the client requests arrive according a Poisson process. The difference in this Chapter is that we have two client communities and two proxy servers. We assume that we have two content servers with 2000 videos each. The popularity of these videos is given by a Zipf-like distribution as explained in Chapter 2. Finally the sizes of the various videos are random and are distributed according to the same normal distribution with the one in Chapter 3.

In our system model here we have added the characteristic of an overlap between the contents of the two content servers. The idea is to examine the efficiency of the cache replacement algorithms in a collaborative scenario where the two content servers do not have the same content. We have run scenarios for different overlaps between the contents of the two content servers. The overlap is defined as the number of common videos stored in the two content servers which are further assumed to be the most popular among all the videos stored in the two content servers. For example, our default value of the overlap equal to 100 videos corresponds to 5% in terms of the number of videos stored in each of the content servers and to a cumulative popularity equal to 44%. Although our default value of the overlap is equal to 100 videos, in our simulations we have run some scenarios for different values of overlap. In our results we express the overlap as the cumulative popularity of the common video files according to the Zipf popularity distribution. In Table 4.1 below we give the values of the overlap in terms of the number of common total videos that we have chosen to experiment with in our simulations, together with the corresponding cumulative probability from the Zipf video popularity distribution.

Overlap expressed in terms of the total number of common videos (corresponding percentage)	Overlap expressed in terms of cumulative probability of the Zipf video popularity distribution
20 videos (1%)	25%
50 videos (2.5%)	35%
100 videos (5%)	44%
200 videos (10%)	54%
300 videos (15%)	61%
400 videos (20%)	66%
500 videos (25%)	70%

Table 4.1 : Overlap expressed in terms of the number of common video files and in terms of the cumulative probability of the Zipf distribution

The values shown in Table 4.1 are expected and this difference is created because the increase in terms of common videos is linear while the increase in terms of the Zipf popularity follows the Zipf distribution with s equal to 0.2. It means that this increase is very fast especially in the beginning of the Zipf distribution which concern the most popular videos. Finally, all the parameters of the simulation of this Chapter and their default values are given bellow in Table 3.1.

Parameters	***Definition (Default values)***
w	Window duration (30 sec)
$\frac{1}{\lambda}$	Mean request inter-arrival time (5 sec)
V1	Number of distinct video titles in the Content Server 1 (2.000 → about 1 TB)
V2	Number of distinct video titles in the Content Server 2 (2.000 → about 1 TB)
C1	Total cache capacity of the LRLFU_Modified Proxy Cache Server (about 100 GB)
C2	Total cache capacity of the NCCS Proxy Cache Server (about 100 GB)
Overlap	Overlap between the contents of the two content servers V1 and V2 (100 videos → the 44% most popular videos)
s	Parameter s of the Zipf distribution (0.2)
m	Mean of the Normal Distribution for the video sizes (500 MB)
S_d	Standard Deviation of the Normal Distribution for the video sizes (200 MB)
Runs	Number of time windows in a simulation run (30.000)

Table 4.2 : System Parameters together with their default values

4.3.3 Simulation Scenarios

It is worth mentioning that we have run three different simulation scenarios for our collaborative scheme. Our basic scenario is the one described in section 4.2.1 in which each proxy runs a different cache replacement policy. The other two scenarios examined are similar to our basic scenario with the difference that each of the two proxy caches runs the same cache replacement policy. In the first case, the two proxy caches run the LRLFU_Modified cache replacement policy and in the second case they run the NCCS cache replacement policy. As expected, the results for the latter two cases were poor. When the two caches in our collaborative system use the same cache replacement policies they cannot help each other because they store the same videos in their caches most of the times. Even in the case where the two proxies run the NCCS policy, in which frequently the videos stored in the two caches are different, because they depend on the requests which arrive in the last few windows, we do not observe significant differences in the system performance. For the above reasons we chose not to present the results of these two collaborative scenarios. So, in the next sections we only present the results of our best collaborative scenario.

4.3.4 Simulation Results

4.3.4.1 The impact of different overlap percentage

In this section we present and discuss the performance results of our collaborative scenario when we vary the percentage of the overlap between the contents of the two content servers. The overlap is expressed here in terms of the cumulative probability of the Zipf video popularity distribution. We have examined separately the performance of the LRLFU_Modified proxy cache and the NCCS proxy cache in terms of the Video Hit Ratio and the Fraction of Delayed Starts. The corresponding results are shown in Figures 4.2 and 4.3, respectively.

As we can see from Figure 4.2 for overlap equal to 25%, the VHR of the LRLFU_Modified is higher than that of the NCCS by 9.7%. As the overlap between the contents of the two content servers increases, the VHR of the NCCS proxy cache increases at a higher rate compared to the

VHR of the LRLFU_Modified which also increases but at a very lower rate. The same trends are observed for the Fraction of Delayed Starts, where in the case of the LRLFU_Modified proxy we observe a slow reduction while in the case of the NCCS the Fraction of Delayed Starts is reduced at a faster rate with the increase of the overlap. As a result, the NCCS proxy outperforms the LRLFU_Modified proxy for values of overlap higher than 39.4%. Specifically for overlap equal to 35% the LRLFU_Modified proxy cache outperform the NCCS proxy cache by 2.1% while for overlap equal to 44% the NCCS proxy outperforms the LRLFU_Modified proxy by 2.3%. This performance difference, with the NCCS proxy being better, increases slowly with the increase of the overlap for values of the overlap higher than 44%. Finally, this performance difference becomes equal to 4.5% for an overlap equal to 70%.

If we take into account the fact that our simulations in Chapters 2 and 3 showed that the NCCS policy achieved very low performance, especially when compared to the LRLFU_Modified policy, the results shown in Figures 4.2 and 4.3 are rather unexpected. The only explanation we could give is that in our collaborative scheme the NCCS proxy is helped very much by the collaboration with the LRLFU_Modified proxy, while the LRLFU_Modified proxy is not helped at all by the collaboration with the NCCS proxy. In order to ascertain this conclusion we have run a simulation described in the next section in which we measure the local and the remote Video Hit Ratio for each proxy cache.

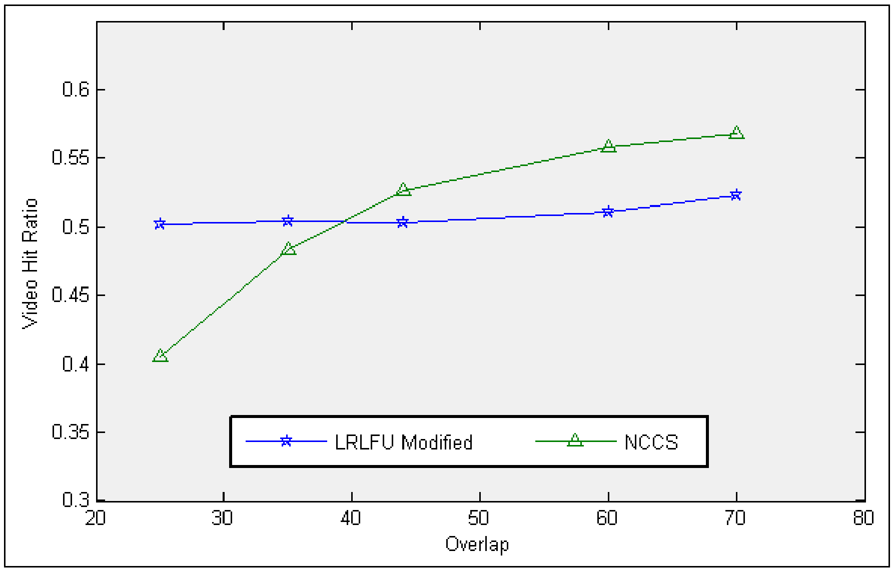

Figure 4.2 : The impact of different values of the overlap percentage on VHR

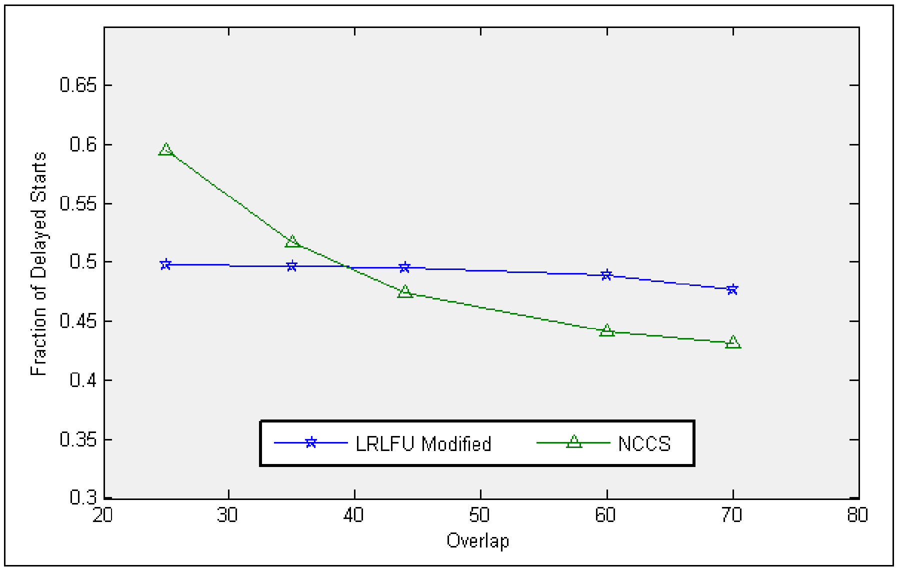

Figure 4.3 : The impact of different values of the overlap percentage on Fraction of Delayed Starts

4.3.4.2 The impact of different values of the overlap percentage on the Local and Remote Video-Hit Ratios of the two cache replacement policies

In this section we run a simulation scenario in which the local and the remote VHR are shown for each proxy cache in order to ascertain and explain the results in the previous section. To make clear what we mean with the terms of local and remote VHR we explain them below. When the proxy which runs the LRLFU_Modified policy finds a video requested by a client in its cache, we say that have a local hit. Instead, when this proxy does not find the requested video in its cache but it finds it in the other proxy cache which run the NCCS policy then we say that we have a remote hit. In a similar way we define the local and remote VHR of the NCCS proxy.

In Figure 4.4, the results of the local and remote VHR for the two proxy caches are shown. As we can see from these results, the LRLFU_Modified proxy finds locally 51.5% of the total number of requested videos for all the values of the overlap. The remote VHR of the LRLFU_Modified proxy is very close to zero. Specifically, for percentage of overlap between 25% and 44%, it is about 10^{-3} % while it reaches 1.7% for overlap equal to 70%. These results clearly demonstrate that the LRLFU_Modified proxy is not helped at all from the collaboration with the NCCS proxy. In contrast, the results clearly demonstrate that the NCCS proxy finds a very high percentage of the total number of requested videos in the remote proxy cache. Specifically, the NCCS proxy finds in its cache (locally) only 25.3% of the total number of requested videos for all the values of the overlap. The remote VHR of the NCCS increases with the increase of the overlap. One important conclusion is that for percentage of overlap higher than 25.3%, the remote VHR of the NCCS proxy becomes higher than the local VHR. For overlap equal to 25%, the remote VHR of the NCCS proxy is 16.7% (8.6% less than the local VHR) while for overlap equal to 70% the remote VHR becomes 32.1% (6.8% higher than the local VHR). Therefore, the collaborative scheme significantly helps the NCCS proxy while the LRLFU_Modified proxy achieves almost the same results with the corresponding in the non-collaborative schemes of Chapter 3.

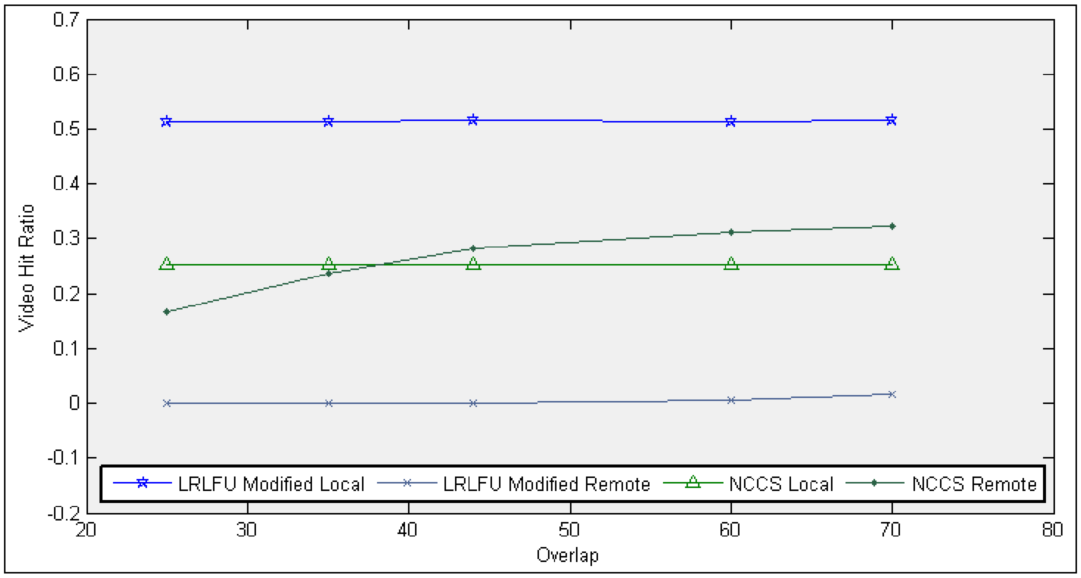

Figure 4.4 : The impact of different values of the overlap percentage on Local and Remote Video-Hit Ratio of the two proxies

4.3.4.3 The impact of different cache sizes

In this section we present and discuss the performance results of our collaborative scenario when we vary the total cache size, while the overlap between the contents of the two content servers is equal to 44% or 100 videos. The results for the VHR are shown in Figure 4.5 and those for the Fraction of Delayed Starts in Figure 4.6. As in the corresponding scenario of Chapter 3, we vary the percentage of the bytes of videos stored in the cache from 5% to 25% of the total bytes of videos stored in content server, while we maintain constant the total size of the content server. We chose to run this simulation scenario with overlap equal to 44% (the default value of the overlap) because for this value the performance difference of the two examined policies is small, with the NCCS being the best. This will help us to better observe the changes of the curves and the trends of the two policies.

From the presented results, it can be observed that the trend of the two policies is the same, since with increasing the cache storage the VHR is improved and the Fraction of Delayed Starts is reduced for both the LRLFU_Modified and the NCCS policies. The changes between the two curves are noticed on the increase rate of the VHR and on reduce rate of the Fraction of Delayed Starts. For the smallest cache size the LRLFU_Modified policy achieves worse performance results than the NCCS policy while for the largest cache size it happens the opposite. It means that the VHR of the LRLFU_Modified policy increases more quickly than the VHR of the NCCS policy and the Fraction of Delayed Starts reduces more quickly than the Fraction of Delayed Starts of the NCCS policy. More specifically, for cache size equal to 5% of content server size the difference on VHR between the two policies is 1.8% with the NCCS being the best while for cache size equal to 25% this difference becomes 2.6% with the LRLFU_Modified being best this time. As it seems rom the presented results for values of cache size smaller than 15% the NCCS outperforms the LRLFU_Modified while for bigger values of cache size than 15% the LRLFU_Modified outperforms the NCCS.

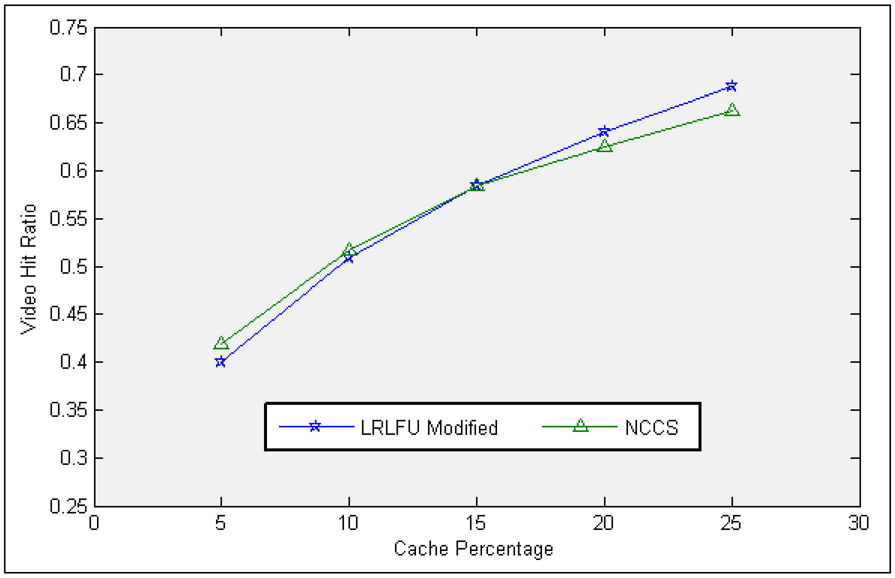

Figure 4.5 : Video-Hit Ratio versus cache size, Overlap = 44%

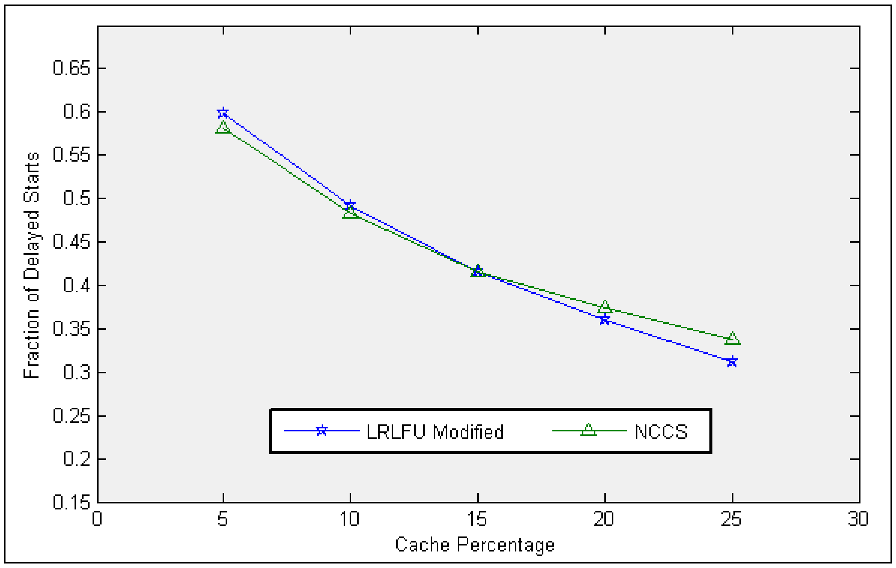

Figure 4.6 : Fraction of Delayed Starts versus cache size, Overlap = 44%

4.4 Conclusions

From the results presented and discussed in this Chapter we conclude that the proposed collaborative scheme improves the overall system performance. Especially for the network topology we examined and for a high percentage of overlap, both the NCCS and the LRLFU_Modified policies achieve very good performance and at the same time store different videos in the two caches thus better serving the client requests. We also observe that the NCCS is helped much more from the collaboration than the LRLFU_Modified, and its performance in the collaborative scheme of this Chapter is better by up to 27% compared to the one in Chapter 3 (with no collaboration). Furthermore, in the collaborative scheme with increasing the size of each cache the performance also increases and the two policies achieve very close performance results.

Chapter 5 : Conclusions

5.1 Overview of Work

In the first part of the Thesis (Chapter 2), we have studied four different algorithms for video caching designed to reduce the bandwidth requirements and shield the clients from delays. In our system model, we implemented three video caching algorithms from the literature and we combined the two of them to create a new algorithm, referred to as the LRLFU_Aggressive with the aim to improve the performance of the two previous algorithms. However from the experimental results we concluded that the new algorithm achieves worse performance than the one of the two algorithms used in the combination, namely the LRLFU.

In the second part of this Thesis (Chapter 3), we again combined the same two algorithms in a little different way in a slightly different system model with variable video sizes. We inserted the characteristic of the video size in the caching value of the LRLFU and we referred to the new algorithm as the LRLFU_Modified. The experimental results show that the new algorithm outperforms both the other two initial algorithms.

In the third part of the Thesis (Chapter 4), we examined the performance of a system with two collaborative proxies where each of them is implemented to run a different video caching algorithm. Specifically, in the system model of Chapter 4 we used the best and the worst algorithms of Chapter 3 because these two algorithms store very different videos in their caches and they could collaborate better. As it is shown from the experimental results in Chapter 4, the collaborative system model that we consider improves the performance of the entire system. Although the cache operating under the high performance algorithm (the LRLFU_Modified) does not gain in performance by the collaboration with the cache operating under NCCS algorithm, it retains its good performance, while the cache operating under the low performance algorithm (the NCCS) achieves better performance than the one using the LRLFU_Modified algorithm.

5.2 Main Conclusions and Contribution

The initial purpose of the Thesis was to combine some characteristics of efficient video caching algorithms presented in [1]-[5] so as to compare the performance of the two different cache replacement policies (the NCCS and the LRLFU) in various scenarios. We introduced a new system model which combines characteristics from two system models of the literature. We also introduced and evaluated in Chapters 2 and 3 various ideas for effective video caching in our system model and a collaborative scenario in Chapter 4. From our simulation results, interesting conclusions can be drawn, which are discussed next.

From the results in Chapters 2 and 3, we conclude that the NCCS algorithm performs poorly when it is assumed that the video popularities follow a Zipf distribution. The NCCS algorithm achieves very low performance in both cases of fixed and variable video sizes. Instead, when the video popularities follow a uniform distribution, the NCCS policy achieves equal or better performance than that of the LRLFU algorithm. In all other cases, the LRLFU policy achieves better performance than that of the NCCS policy.

In Chapter 2 we proposed an algorithm similar to the LRLFU with the difference that we took into account batch arrivals in the caching value of each video. From the results we concluded that this change does not improve the performance of the LRLFU algorithm. Instead, our new algorithm is outperformed by the LRLFU. For that reason we decided not to examine this algorithm in the subsequent Chapters of the Thesis.

In the second part of this Thesis (Chapter 3) we introduced in the system model the case of variable video sizes and we proposed a new algorithm referred to as the LRLFU_Modified. The caching value of the new algorithm is equal to the LRLFU caching value divided by the video size. Our simulation results have shown that this change in the caching value helps the LRLFU_Modified algorithm to outperform the LRLFU algorithm by 2% (for video sizes distributed according to a normal distribution) to 5% (for video sizes distributed according to a uniform distribution). An additional conclusion is that if our algorithm is chosen to manage the cache of a video segmentation system with the performance metric being the Byte Hit Ratio, the video segmentation scheme must be designed carefully in order to obtain better performance than that of the LRLFU algorithm in this case.

In the last part of the Thesis (Chapter 4), a collaborative scheme has been proposed and evaluated. From the corresponding performance results we conclude that in general when there is a collaborative scheme of proxies designed carefully so that each proxy runs a different cache replacement algorithm, the overall system performance is improved. Especially for the collaborative system architecture examined in Chapter 4 and for big percentage of overlap between the contents of the two content servers, both the NCCS and the LRLFU_Modified algorithms achieve very good performance and at the same time they store more videos in the two caches to better serve the client preferences. Our results show that the proxy which runs the NCCS policy is helped much more by the collaboration than the other proxy which runs the LRLFU_Modified algorithm. Finally, the proxy running the NCCS algorithm in the collaborative scheme achieves a performance which is better by up to 27% relative to the performance achieved by the same proxy with no collaboration.

5.3 Ideas for Future Work

An interesting idea for future work would be to expand the collaborative scheme to support more than two proxies and client communities. It would be a little complicated, because for the implementation of this idea the system architecture of the collaborative scheme needs to be designed carefully.

It would also be interesting to evaluate the performance of the examined algorithms in a system video caching model which supports video segmentation. In such system model there would be two or three types of cache, each type used to store different segments of videos (prefix, suffix e.t.c.) and potentially running different cache replacement algorithms.

Finally, another interesting feature which could be added to our system model is the case of multiple personalized versions of a video or a video segment at varying levels of content abstraction. A feature like this has been implemented in [3], in which it was assumed that there are three versions for each video and each version represents a distinct level of visual quality with a corresponding video file size.

References

[1] A. Satsiou, M. Paterakis, “Frequency-Based Cache Management Policies for Collaborative and Non-collaborative Topologies of Segment Based Video Caching Proxies,” in Springer Multimedia Systems, vol. 12, Sept. 2006, pp. 117-133.

[2] A. Satsiou, M. Paterakis, “Impact of Frequency-Based Cache Management Policies on the Performance of Segment Based Video Caching Proxies”, in Proc. of the 2004 IFIP Networking Conference, Athens, Greece, 2004, pp. 1120-1131.

[3] S. M. Bhandarkar, L. Ramaswamy, H. K. Devulapally, “Collaborative caching for efficient dissemination of personalized video streams in resource constrained environments”, in Springer Multimedia Systems, vol.20, Jan. 2013, pp. 1-23

[4] P. Parate, L. Ramaswamy, S.M. Bhandarkar, S. Chattopadhyay, H.K. Devulapally, “Efficient Dissemination of Personalized Video Content in Resource-Constrained Environments”, in Proc. of the 5th International Conference Collaborative Computing (CollaborateCom 2009), Washington, DC, 11–14 Nov. 2009, pp. 1-9.

[5] S. Chattopadhyay, S. M. Bhandarkar, Y. Wei, L. Ramaswamy, “Video Personalization and Caching for Resource Constrained Environments”, in Proc. of the International Workshop on Mobile Multimedia Processing, Tampa, Florida, Dec. 2008.

[6] J. M. Almeida, D. L. Eager, M. K. Vernon, “A Hybrid Caching Strategy for Streaming Media Files”, in Proc. of the Multimedia Computing and Networking, San Jose, CA, Jan 2001, pp. 200-212.

[7] E. Balafoutis, A. Panagakis, N. Laoutaris, I. Stavrakakis, “The Impact of Replacement Granularity on Video Caching”, in Proc. of the 2nd IFIP Networking Conference, Pisa, Italy, May 2002, pp. 214-225.

[8] X. Gu, C. Ding, “On the Theory and Potential of LRU-MRU Collaborative Cache Management”, in Proc. of the 10th International Symposium on Memory Management, San Jose, CA, Jun 2011, pp. 43-54.

[9] Z. Xiang, Q. Zhang, W. Zhu, Y. Zhong, “Cost-Based Replacement Policy for Multimedia Proxy across Wireless Internet”, in Proc. of the IEEE GlobeCom, San Antonio, TX, vol. 3, 25-29 Nov 2001, pp. 2009-2013

[10] S. Jin, A. Bestavros and A. Iyengar, “Accelerating Internet Streaming Media Delivery using Network-Aware Partial Caching”, in Proc. of the 22nd IEEE International Conference on Distributed Computing Systems, Vienna, Austria, 02-05 Jul 2002, pp. 153-160.

[11] L. Shi, Z. Wang, Y. Yao, L. Wei, “Streaming Media Caching Model Based on Knapsack Problem”, in Journal of Networks, vol. 6, Aug 2011, pp. 1379-1386.

[12] J. Dilley, M. Arlitt, “Improving Proxy Cache Performance-Analyzing Three Cache Replacement Policies”, Technical Report, HP Laboratories, Palo Alto, CA, Oct 1999.

[13] R. Ayani, Y. M. Teo, Y. Seen Ng, “Cache Pollution in Web Proxy Servers”, in International Parallel and Distributed Processing Symposium, IEEE Computer Society, Apr 2003, pp. 248 .

[14] J. S. Kushwah, J. K. Gupta, M. S. Yadav, H. Madan, “Modified LRU Algorithm to Implement Proxy Server with Caching Policies”, in International Journal of Computer Science Issues, Vol. 8, Issue 6, Nov 2011, pp. 352

[15] N. Megiddo, D. S. Modha, “Outperforming LRU with an Adaptive Replacement Cache Algorithm”, in IEEE Computer, Vol. 37, Issue 4, Apr 2004, pp. 58-65

[16] M. Neves, M. Rodrigues, E. Azevêdo, D. Sadok, A. Callado, J. Moreira, V. Souza, “Selecting the most suited cache strategy for specific streaming media workloads”, in Proc. of the International Symposium on Integrated Network Management, May 2013, pp. 792-795

[17] http://en.wikipedia.org/wiki/Bit_rate#Video

[18] http://www9.org/w9cdrom/349/349.html

[19] http://whatis.techtarget.com/definition/proxy-server