Multimedia Systems Manuscript No.
(will be inserted by hand later)

Design and Evaluation of a Multimedia Storage Server for Mixed Traffic

Igor D.D. Curcio[1⋆], Antonio Puliafito[2⋆⋆], Salvatore Riccobene[1⋆⋆⋆] and Lorenzo Vita[3†]

[1] Dipartimento di Matematica, Università di Catania, Viale A. Doria 6, 95125, Catania - Italy,
[2] Istituto di Informatica, Università di Catania, Viale A. Doria 6, 95125, Catania - Italy,
[3] Facoltà di Ingegneria, Università di Messina, Salita Sperone, 98166, Messina - Italy

Received: xx/xx/xxxx / Accepted: yy/yy/yyyy

Abstract. The relative simplicity of access to digital communications nowadays and the simultaneous increase in the available bandwidth are leading to the definition of new telematic services, mainly oriented towards multimedia applications and interactivity with the user. In the near future, a decisive role will be played in this scenario by the providers of interactive multimedia services of the On-Demand type, which will guarantee the end user a high degree of flexibility, speed and efficiency. In this paper some of the technical aspects regarding these service providers are dealt with, paying particular attention to the problems of storing information and managing service requests. More specifically, the paper presents and evaluates a new storage technique based on the use of Disk Array technology, which can manage both typical multimedia connections and traditional requests. The proposed architecture is based on the joint use of the Partial Dynamic Declustering and the Information Dispersal Algorithm, which are employed for the allocation and retrieval of the data stored on the Disk Array. We also define efficient strategies for request management in such a way as to meet the time constraints imposed by multimedia sessions and guarantee good response times for the rest of the traffic. The system proposed is then analyzed using a simulation approach.

Key words: Multimedia Servers - Disk Array - Video on Demand - Request Scheduling - Performance Evaluation.

1 Introduction

In the last few years multimedia technology has revolutionized the way in which computers and users communicate. Systems have become increasingly interactive and allow the user to choose the type of service according to her specific requirements. *Video on Demand* (VOD) services, for example, give the user greater flexibility in choosing services, thus revolutionizing the paradigm according to which the user is a passive participant, "forced" to receive the video streams broadcast by the various service providers, at pre-established times. According to the new paradigm, a user accessing these new services can choose which movie to watch from a varied daily menu. She can also decide when she wants to see the *movie* she has chosen and act as if she had a VCR at his disposal. Although on the one hand these new services provide flexibility, speed and a high level of efficiency, on the other they require new support technology and considerable effort needs to be made to improve the *Quality of Service (QoS)* provided by existing systems: the *Service Provider* has to have a large amount of memory available to store hundreds or thousands of movies in digital format, and also large processing systems which can serve thousands of users at the same time.

Media like audio and video have different time characteristics from conventional data; they are isochronous by nature and are classified as *Continuous Media (CM)* as they are composed of sequences of continuously transmitted audio samples and video frames which have very strict time constraints. CM data is different from other data on account of the following properties:

- *Real-Time Characteristics*
  Search, processing and presentation of CM is delay-sensitive. The data has to be read before a certain deadline so as not to degrade the quality of service. The data flow is also almost constant in time in the recording and reading phases. Meeting a request with these characteristics means that the reading of subsequent audio samples and video frames has to meet subsequent deadlines. To guarantee this, it is necessary to use buffers large enough not to cause delays in the consumption of data;
- *Large File Size and High Transfer-Rate*
  Audio and video data require a large amount of mass memory and playback requires a high transfer rate. A multimedia system therefore has to store, search for and process large amounts of data efficiently, at a high speed. Data compression systems thus play a

⋆ e-mail: curcio@acm.org
⋆⋆ e-mail: ap@iit.unict.it
⋆⋆⋆ e-mail: sriccobene@dipmat.unict.it
† e-mail: vita@mat520.unime.it

very important role in this context: in fact these techniques reduce the storage space and the transmission bandwidth required;

– *Multiple Data Streams*
A multimedia system has to support different media at the same time. The reading and transmission of a movie in a VOD system, for example, requires audio and video processing and synchronization. This information, in fact, is not always stored in a single file.

VOD systems are therefore *real-time* systems, so presenting data "on time" is an important quality factor which cannot be neglected. In VOD systems respecting time constraints is a dominant factor in data correctness. It is, in fact, preferable for frames to reach their destination partially incorrect rather than late due to retransmission.

A large amount of literature has recently been published about managing CM flows, analyzing both storage and retrieval aspects. Anderson et al. (1992) have developed a Continuous Media File System based on the concept of "session", for which a minimum data rate is guaranteed. Vin et al. (1993) studied the storage problem of HDTV video. They used a constrained block allocation to memorize the data on the mass memory. Berson et al. (1995) have presented a new technique to storage a movie on a disk array, called Staggered Striping. They used the flexibility of the Disk Array organization to handle media with different data rate requirement. MPEG stream reorganization in order to reduce frame losses during transmission is examined by Chen et al. (1995b). An Admission Control algorithm is presented by Vin et al. (1995). The authors based the decision about a new client admission on the past history, in the way such that all the already admitted clients are guaranteed about the service requirements. Mourad (1996) shows a technique for movie allocation on Disk Array using disk striping policy. He also analyzes the assignment of the movies to the mass memory, evaluating the replication factor. Steinmetz (1995) presents an interesting survey on disk scheduling algorithms, focusing on EDF, SCAN-EDF and GSS. Sahu et al. (1997) examine two policies for retrieval of VBR video. They analyze both the constant time lenght and the constant data lenght strategies.

As any Internet user knows, Web pages nowadays contain an increasing amount of different media, such as video, text, audio and images. We feel that correct sizing of a system for multimedia data must take into account a careful examination of non-multimedia flows, even though they do not usually have urgent time constraints. Even a system dedicated to providing VOD services may need to manage non-multimedia data. An example would be a *background copy* operation, i.e. the transfer of a movie between two servers: it is usually performed *off-line*, but this does not mean it does not interfere with normal server functioning. In addition, as background copy is an extremely onerous operation, an analysis of how it affects regular access to multimedia data may give general information about the coexistence in a server of multimedia and non-multimedia requests.

Table 1. Storage requirements for digital multimedia data

Media Type	Specifications	Data Rate (per second)
Voice-quality audio	1 channel, 8 bit samples at 8 KHz	64 Kbit
MPEG-encoded audio	Equivalent to CD-quality	384 Kbit
CD-quality audio	2 channels, 16 bit samples at 44.1 KHz	1,4 Mbit
MPEG2-encoded video	640x480 pixel/frame, 24 bit/pixel	0.46 Mbyte
NTSC-quality video	640x480 pixel/frame, 24 bit/pixel	27 Mbyte
HDTV-quality video	1280x720 pixel/frame, 24 bit/pixel	81 Mbyte

Particular consideration must also be given to the large amount of storage space needed to manage multimedia data (see Table 1, Gemmel et al. (1995)), which requires advanced compression techniques. Various standards have been proposed in the last few years (*MPEG, H.261, INDEO*, see Chen et al. (1995a), Le Gall (1991), Nass (1995)). MPEG-1 is mainly used for video clips, games and, in general, for applications not requiring the use of a large screen. MPEG-2, on the other hand, offers better resolution, at the same time using more limited compression. It is thus suitable for high-quality movies, even though the amount of memory occupied and the throughput required are much greater than MPEG-1.

A 90-minute movie in MPEG-1 format, for instance, which requires a transmission rate of 1.5 Mbit/s (for audio and video), needs about 1 Gbyte of memory. Likewise, a movie of the same length in MPEG-2 format (*HDTV-quality video*), which requires a transmission rate of 16 Mbit/s or more [Furht et al. (1995)], needs about 11 GBytes of memory.

MPEG-4 is currently being standardized and will offer a low bandwidth for interactive applications. H.261 and INDEO are mainly used for videoconferencing.

In this paper we deal in detail with *Multimedia Storage Servers* in a VOD service provider in which, given the large amount of data to be managed, tens or hundreds of disks are available on-line to the user. We will examine a disk array architecture which uses two techniques for data storage: *Partial Dynamic Declustering (PDD)* for non-multimedia data, and the *Information Dispersal Algorithm (IDA)*, together with PDD, for multimedia traffic [Puliafito et al. (1996)]. We will also examine disk scheduling algorithms. Using various optimization techniques in managing queues and serving requests, simulation tests will show a 75% improvement over the results obtained without applying these techniques. The rest of the paper is organised as follows.

Section 2 introduces VOD systems, outlining a possible complete scheme. Section 3 presents the disk array architecture proposed. Section 4 deals with the problem of disk scheduling, and Section 5 presents the simulation model, while Section 6 gives the results obtained referring to the performance of the system. The 7th and final

section presents our conclusions and points out the aims of future research.

2 Video-On-Demand Systems

In traditional TV systems, service providers transmit movies at pre-established times, so the user is forced to choose what she wants to see from a very limited set of options, and has to fit her viewing times in with the rigid program established by the service provider. The only form of interactivity this kind of system allows the user is to choose between the various movies broadcast at any one time.

VOD services, on the other hand, propose to overcome these limits both by putting a wider range of movies at the user's disposal and by making typical VCR services available, thus considerably increasing interactivity with the user.

According to the type of interaction, VOD services can be classified as in Little and Venkatesh (1994):

- *Broadcast (NO-VOD)* in which the user has a passive role and cannot modify the service given by the service provider (e.g. current TV systems);
- *Pay-Per-View (PPV)* in which the user has a specific contract with the service provider to see certain categories of programs;
- *Quasi Video-On-Demand (Q-VOD)* in which users are grouped according to subjects of common interest. In this case users can change groups interactively;
- *Near Video-On-Demand (N-VOD)* in which functions like forward and reverse are simulated at discrete time intervals, e.g. every 5 minutes;
- *True Video-On-Demand (T-VOD)* in which the user has complete control of the program she receives at home. She can use typical VCR functions such as forward and reverse play, pause and random positioning.

Below we will refer to the last of these categories, which represents the higher degree of interaction, among VOD systems, between the service provider and the user. For the sake of simplicity we will use the term VOD instead of T-VOD. *Interactive television* (ITV) represents an enhancement of VOD systems as it offers the user a much greater level of interactivity. It is a *two way* system which allows to request informations, ask for services or buy things. However, ITV does not match a precise definition because the exact form that will ultimately emerge is yet to be detemined [Pavlik (1996)].

A large number of users can access a VOD system at the same time - several thousands on a metropolitan scale - and so it has to guarantee that the QoS provided is always kept within acceptable limits. The described architecture proposes a logical organization of the storage hierarchy. Real implementations are not necessarily restricted to a 2-step hierarchy of central server/local server, but each level of the hierarchy can be further expanded.

According to a hierarchical architecture (see Fig. 1), for instance the one considered in Furht et al. (1995), the *Central Multimedia Server (CMMS)* has an archive containing all the movies to be transmitted. Through a high-speed network it sends copies of movies to several *Local Multimedia Servers (LMMS)* which then distribute them according to user requests.

In this architecture a hierarchical approach to mass storage is the most suitable [Little and Venkatesh (1994)]. Given the enormous amount of data needed to manage hundreds of videos in digital format, the CMMS needs to store them in compressed form on low-cost, high-storage-capacity magnetic tapes.

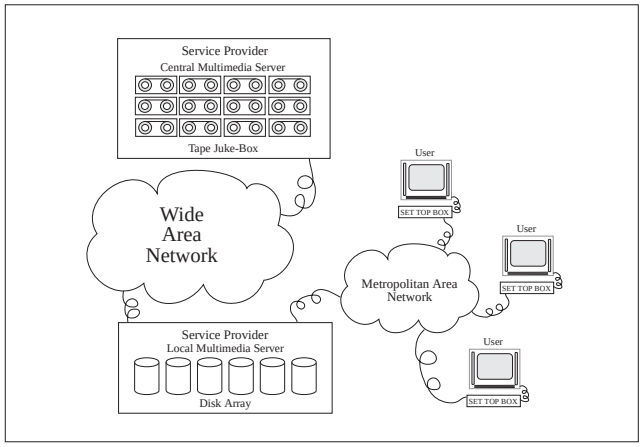


Fig. 1. Architecture of a typical Video-On-Demand System

The transfer of movies between a CMMS and an LMMS (movie distribution policy) can occur off-line and therefore can be performed in nonreal-time [see Netravali and Lippman (1995)] and at times of day when the load on the system is relatively light. Fig. 2 shows a possible model for the daily workload on a VOD database [see Little and Venkatesh (1994)]. The model is only taken as a reference model: in general, a varied series of models can be created according to the geographical area of the users, the kind of users, the time of year, etc. As can be seen, in this model the period with the greatest workload is around 8-9 p.m., whereas the amount of traffic is relatively low during the night and early morning. During this period it will thus be possible to transfer copies of movies to secondary servers without excessively penalizing active multimedia sessions. The choice of which videos to transfer can be made on the basis of their popularity rating and can thus be predicted with a reasonable degree of success. When a user requests a less popular movie, the system has to be able to transfer it in a reasonably short time, even at times when the traffic is heavy.

An LMMS is a system with a large number of disks containing the movies most often requested by users. There are hundreds of disks, each of which contains several Gbytes of storage space. To enhance the performance and transfer rate a disk array strategy can be followed, thus allowing parallel access. Such a choice also considerably increases the reliability and availability of the VOD system as a whole.

The user has a unit called a *Set-Top Box (STB)* [Furht et al. (1995)] which decodes the data the LMMS sends over a *Metropolitan Area Network (MAN)* with a

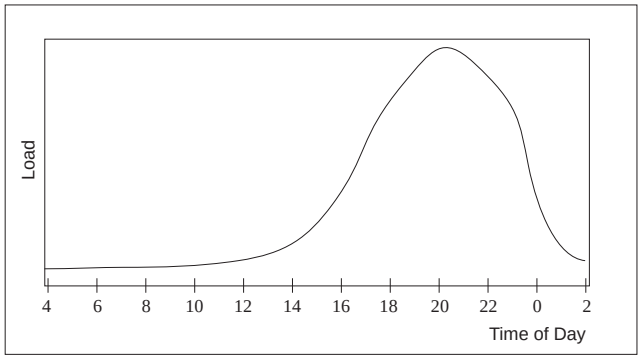


Fig. 2. A model for the daily workload in a VOD system

broad bandwidth (2 Gbit/s or more). An STB is a simplified multimedia computer with a memory, CPU, MPEG decoder. It is connected to other peripheral units and it is activated by a special remote control with which the user can communicate with the Service Provider and choose the movie and the time she prefers to see it. It also has the classical VCR functions (play, rewind, fast forward, pause, stop).

3 The Multimedia Disk Array System Architecture

The architecture of the system considered is based on the work proposed by Puliafito et al. (1996). It assumes that the system has to serve two kinds of request:

- *multimedia (MM)* requests (typically VOD traffic)
- *non multimedia (NON MM)* or normal requests (also called *aperiodic* or *asynchronous*).

The point of view analyzed here is that of the LMMS. We will therefore not deal with the details of the CMMS. In addition, the following assumptions will be made about the system:

- No account will be taken of aspects relating to the network; in particular, we will neglect *end-to-end delay* in transmission of data from the LMMS to the user. This delay depends on the bandwidth, the traffic on the network, and the QoS parameters. Various attempts have been made recently to estimate these parameters [see Nahrstedt and Steinmetz (1995)];
- Audio and video data will be considered to be stored together and not in separate files; some coding techniques already do this;
- Server access is almost exclusively read access: we assume that the movie distribution policy works efficiently, so that most of the requests are served using the data already stored on the local Disk Array.

When the size of a single disk is not sufficient to contain all the data to be stored, it is preferable to use an interconnected set of disks or *Disk Array* [see for example Ganger et al. (1994)]. Splitting I/O requests between several parallel disks, this organization greatly reduces queueing delay and increases disk bandwidth. Disk arrays also offer considerable advantages in terms of *expandibility, availability, reliability and performability* [Catania et al. (1995)]. These advantages essentially depend on two factors:

- data distribution
- data replication.

As far as the first factor is concerned, various data distribution techniques have been proposed in literature. The most frequent are Clustering, Interleaving and Declustering.

In the *Clustering* [Catania et al. (1995)] technique, the various disks are managed separately, from both the logical and the physical point of view. Each unit, in fact, has a controller and a request queue. Each disk has its own file system, which is physically separate from that of the other units. The parallelism that can be exploited in this kind of organization is the possibility of serving requests addressed to different units (separate files) simultaneously.

In *Interleaving* [Kim (1986)], the various units share the same file system. The disks are all the same and synchronous. There is a single controller and only one request queue. Each file is split into macroblocks of a preestablished size, called *stripe units*. In turn, stripe units are split into *blocks* which are distributed in a round-robin fashion over the various units in the system. In this way, I/O requests are parallelly addressed to all the disks, thus automatically distributing the workload. In practice, the whole system of disks is logically to be seen as a single virtual disk with larger sectors and a high level of throughput. In access to large amounts of data, this permits short service times.

In addition to the first two, there is also an asynchronous solution with combines the advantages of both. In *Declustering* [Chen et al. (1994)], the disks are physically managed independently (i.e. there is a controller and a request queue for each disk), but they are connected from the logical point of view. The file system, in fact, is distributed as in Interleaving. In this way requests for large amounts of data exploit the declustering of files over several units, while smaller requests involving only a few disks do not constitute an overload for the system.

To implement high-capacity subsystems, i.e. ones with a large number of disks, it is convenient to split the system into groups. Then file declustering is applied within each group (*Partial Static Declustering*, PSD) [Catania et al. (1995)]. The reason for such a choice is that the performance of the I/O subsystem tends to degrade if excessively large stripe units are used [see also Chen et al. (1994)]. Splitting the system up into groups, on the other hand, makes it possible to use smaller stripe units. The optimal size will depend on the type of workload applied.

Data replication is considered in relation to the reliability of the system. Thanks to the introduction of a certain degree of redundancy during the data writing phase, in fact, it is possible to reconstruct the contents of a disk on the occurrence of a fault. The most widespread tech-

niques are called *RAID (Redundant Arrays of Inexpensive Disks)* [Chen et al. (1994)]. Redundancy is achieved either by total duplication of information (*RAID 1*) or with other coding techniques in which an additional disk is used to store parity information (*RAID 4*). This disk can also be logically distributed over several physical mass storage units (*RAID 5*).

In our system we will use the *Partial Dynamic Declustering* technique to store non-multimedia data and a coding technique based on the *Information Dispersal Algorithm* to store data relating to multimedia traffic. In the following subsections we will give a detailed description of these techniques.

3.1 Partial Dynamic Declustering

The Partial Dynamic Declustering (PDD), described by Catania et al. (1995) is an evolution in the organization of the file system used in PSD. The main feature of PDD is the possibility of dynamically forming groups to store a file. In PSD each group of disks has its own file system, logically separate from that of the other groups. In PDD, on the other hand, this separation is not possible.

When a file is created, the system decides not only which disks to use but also how many (see Fig. 3). This allows the system to decide case by case on the optimal size of the stripe unit to be used, thus adapting the features of the I/O system to those of the workload, which means an improvement in performance.

Another very important feature of PDD is flexibility in the formation of groups within the set of disks. In PSD, the maximum number of groups is N/k, where N indicates the total number of disks and k the size of the group. In PDD we will have a number of possible groups equal to:

$$\sum_{i=1}^{N} \binom{N}{i}.$$

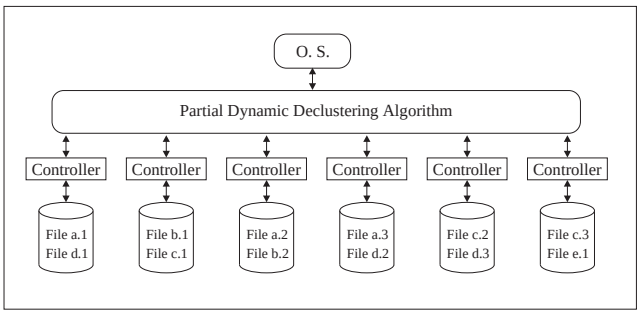


Fig. 3. Partial Dynamic Declustering

If the number of disks making up the groups is set to k, this value is reduced to

$$\binom{N}{k}$$

which is always greater than in PSD. Of course, whereas in PSD the groups are always disjoint, in PDD they can partially overlap, i.e. have disks in common.

The large number of groups and overlapping facilitate distribution of the various files over the disks, thus increasing at the same time the probability that a generic workload will be uniformly distributed over all the disks. This is of fundamental importance in enhancing performance. The management of groups and the sharing out of files between them is dealt with by an algorithm which has proved to be particularly efficient.

3.2 The IDA Technique

This coding [Rabin (1989)] was originally envisaged to increase the degree of fault tolerance of a generic system.

The data to be stored is subdivided into *macroblocks* of a fixed size. A macroblock is the smallest unit of data processed by the IDA in each operation. Its size is determined according to the features of the system and the data it has to process. Each macroblock is then subdivided into M *blocks* of a size D.

If R is the value of desired *redundancy*, it represents the number of blocks that have to be added, thus bringing the total number of output blocks to $N = M + R$. In other terms, starting from M initial blocks, the IDA coding produces N new blocks of size D (see Fig. 4). Each of the N output blocks is obtained as a linear combination of the M input blocks. To perform this operation it is necessary to use an $N \times M$ matrix. Choice of the matrix (i.e. of the coding) will be such that any subset of M elements chosen from the N output elements will make it possible to reconstruct the M input blocks.

In our specific case, N indicates the number of disks in a group formed using the PDD technique, while M indicates the minimum number of disks in the group needed to reconstruct the information as provided for by the IDA. So if a file is stored using N stripe units, in the reading phase it will be sufficient to read only M stripe units to reconstruct it.

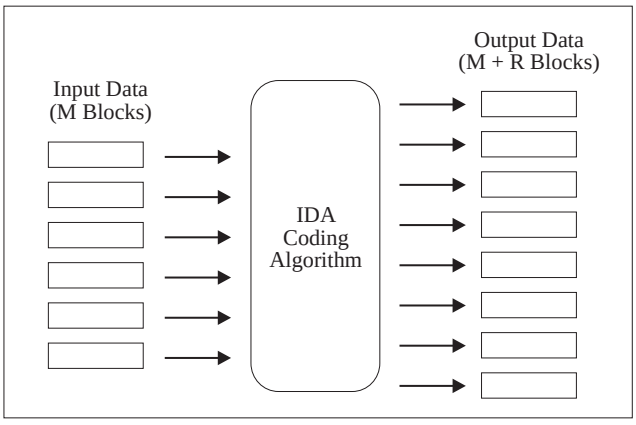


Fig. 4. The IDA coding technique

The IDA technique obviously introduces a certain additional workload, as it is necessary to encode and decode data. However, as stated by Rabin (1989), these phases are *computationally efficient*. In any case, the overhead on the system can be reduced by performing these operations directly in hardware.

IDA coding in a disk array offers an important advantage: it increases the *MTBF (Mean Time Between Failures)* of the system. According to the degree of reliability required, it is possible to choose the redundancy value, R, in such a way as to obtain the desired MTBF. This means that the system can undergo up to R faults without jeopardising integrity in the reconstruction of the original data. In a disk array organized according to PDD tecnique it is also possible to decide the degree of redundancy file by file.

It should, however, be pointed out that using IDA in disk array systems may degrade performance. The structure of IDA coding, in fact, requires a minimum number of stripe unit blocks needed to reconstruct data (M blocks) to be retrieved in the reading phase. This means that even if the reading request only refers to a single block, the system has to read at least M. Likewise, in the writing phase the whole stripe unit of N blocks has to be rewritten.

However, these features are not always an overhead for the system. They only penalize small I/O operations, especially writing operations.

A workload involving the reading of large amounts of data (one or more stripe units), on the other hand, is not penalized. The possibility of choosing the units from which to read the M blocks needed for decoding makes effective run-time balancing of the workload possible, thus enhancing performance.

Finally, we should point out that these are the features to be found in the workload of a typical VOD server. The multimedia requests involved, in fact, almost exclusively refer to reading and require the transfer of large amounts of data. Writing phases (the transfer of movies from one server to another) are quite rare and generally occur off-line, in low-workload periods. A detailed analysis of movie distribution policies can be found in Little and Venkatesh (1994) and Netravali and Lippman (1995).

In any case, the logical organization we have devised makes management of these transfer phases possible even during normal functioning; the priority is lower than that of the serving of multimedia sessions, but service times are not too long.

In the following section we will describe some request management policies envisaged for this type of organization.

4 Disk Scheduling Policies

In the previous section we dealt with the architecture of the disk array system. Here we will tackle the problem of disk scheduling, outlining special optimization techniques that allow efficient exploitation of the advantages offered by PDD and IDA. Traditional scheduling algorithms such as *FCFS (First Come First Served), SSTF (Shortest Seek Time First), SCAN, C-SCAN, FSCAN, N-Step SCAN,* and LIFO [Wiederhold (1987)] are not suitable for multimedia systems as they do not contemplate the concept of a deadline for a request, which is very important in real-time systems. C-SCAN, for instance, would serve multimedia requests in the order in which they are placed on the disk (not taking into account any time constraint on data presentation), whereas it would serve non-multimedia requests in scan order and they might have to wait a long time to be served after a certain number of multimedia requests.

Wheareas the main aim of traditional disk scheduling algorithms is to reduce the cost of the seek time so as to guarantee maximum throughput, that of multimedia scheduling algorithms is to meet all the deadlines for the requests. If there are also non-multimedia requests in a multimedia system, the scheduling algorithm has to be able to find the right balance between multimedia and non-multimedia requests, avoiding excessive delays for the latter, trying to guarantee a short service time, and also preventing the serving of aperiodic requests from disturbing that of multimedia requests.

Recently some interesting algorithms have been presented in literature to manage real-time I/O requests. In the paper by N. Reddy and Wyllie (1994) the Scan-EDF algorithm is proposed. This policy normally handles requests in EDF order but, for requests with the same deadline, the Scan optimization is adopted. The efficiency of this policy depends on the probability that the same deadline applies to a huge number of requests. However, if it is not the case, the algorithm does not introduce any optimization, because it reduces to the simple EDF policy. The behavior of this algorithm really depends on the adopted criteria for assigning deadlines to I/O requests.

RT-Window, proposed by Chen and Thapar (1996), is an enhancement of the Scan-EDF algorithm. Its performance does not depend on the workload, i.e. on the presence of requests carrying the same deadline, and its behavior can be tuned by adjusting some parameters, indicated as threshold and window size. The authors do not make explicit reference on how to deal with multimedia and non-multimedia requests. However, we believe that for a correct sizing of the system, multimedia and non-multimedia flows must be considered at the same time.

In the following section we propose a new disk scheduling policy which is able to manage multimedia and non-multimedia traffic and allows performance improvements by run-time selecting the disks to be used. By properly combining IDA and PDD, our scheduling policy reduces the idle time of each disk by inserting an adequate amount of non-multimedia requests among two consecutive multimedia requests.

4.1 Scheduling Multimedia Requests

Multimedia requests are associated with sessions opened by users who want to watch a video. They are periodic requests and in systems like VOD it can reasonably be assumed that video viewing is continuous. It is thus possible to schedule multimedia requests automatically and continuously. The elements that come into play when the data is transferred from the server to the client, and

which introduce delays and thus a degradation in the QoS, are not only the server itself but also the communication system and the client. The latter is equipped with a storage resource (*buffer*) to store the data from the server. We will indicate a generic data transfer between server and client *(multimedia request)* in a given communication session as a *round*. The amount of data to be transferred in each round strictly depends on the size of the buffer at the user's disposal. In each round it is therefore possible to identify a *deadline* by which the whole block of data has to reach its destination. The server has to estimate the instant at which to start serving a request in a round. As compared with the theoretical value, however, it is advisable to bring this slightly forward so as to absorb any delays accumulated during the service phase. It must not, however, be brought forward too much in order to avoid overruns in the user buffer. The value depends on various factors such as the load on the system, the size of the multimedia requests, and the maximum number of violations allowed. As far as the client is concerned, she is interested in the frames arriving at the right time, so as not to suffer a degradation in the QoS.

A multimedia request therefore features an *estimated time* (the time estimated for execution), an *execution time* (the time actually required for execution), a *release time* (the time when the request arrives), a *start time* (the time execution starts), a *deadline* (the time by which execution of the request has to be completed) and a *completion time* (the actual time at which it is completed) (see Fig. 5).

The start time is calculated using the expression:

$$T_{start} = Deadline - T_{estimated} - T_{FixedAnticipation}$$

where by *fixed anticipation* we mean the length of time needed between the start time and the deadline for the request to be served by its deadline. The actual execution time may be longer or shorter than the estimated time. The completion time may also be after the deadline, in which case there is a violation which is calculated using the following expression:

$$Violation = T_{completion} - Deadline.$$

If $Violation$ is a positive quantity, there is an actual violation of the deadline and a situation of buffer starvation: the buffer has to wait for data to consume. This means the transmission of the movie will be interrupted (*glitch*).

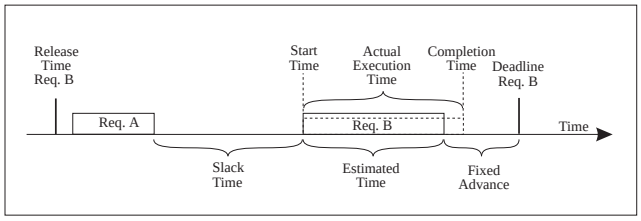


Fig. 5. Typical Time Instants for Multimedia Requests.

In a VOD system deadlines are *soft*, i.e. violations do not jeopardise the integrity of the system [Burns (1991)]. A deadline violation in fact corresponds to a variation in the speed at which the video is transmitted and this means that on reception there is degradation in the quality of audio and video. These oscillations may not be perceived by the user if they are very small, so instead of requiring each periodic request to respect its deadline very closely, we can try to limit the frequency of violations occurring and minimize the distance from the deadline that each violation causes.

The scheduling policy used in our architecture is that of a traditional real-time system: the *Earliest Deadline First (EDF)* algorithm [see N. Reddy and Wyllie (1994), and Steinmetz (1995)]. From the requests waiting to be served, the algorithm selects the one with the earliest deadline and serves it. If various requests have the same deadline they are served on a FIFO policy.

There are two possible implementations of the EDF algorithm. The first, *EDS (Earliest Deadline as Soon as possible)*, executes the request as soon as possible, i.e. it tries to bring the start time as close as possible to the release time. The second, *EDL (Earliest Deadline as Late as possible)*, delays execution of requests as long as possible, i.e. it tries to bring the start time close to the deadline.

In our architecture we chose EDL, which is optimal if the request service time is known a priori. In our case it is possible to use algorithms to predict read requests because, as mentioned previously, a user is highly likely to watch a movie uninterruptedly. With knowledge of the video allocation policy, in fact, it is possible to make optimal predictions concerning the service time for each request. We assume that once a multimedia request starts being executed it cannot be suspended until it has been completely served. Due to this property, the algorithm is also said to be *non-preemptive*. An algorithm of this kind is simple to implement and only involves a slight overhead on the system.

On the basis of what was said about the IDA in Section 3.2, a single multimedia request is considered to have been served when M out of N disks respond to their part of the subrequest. Typical parameters on which to base selection of the M disks to which the multimedia read requests are to be sent are the length of the disk queues, the number of blocks processed per time unit, the number of blocks waiting to be read, the length of disk idle time per time unit, etc.

4.2 Scheduling Non-Multimedia Requests

In a VOD system with mixed traffic the server may reach peak loads for non-multimedia requests. In this case the scheduler has to ensure that the deadlines for multimedia requests are met even in the presence of non-multimedia overloads and also try to guarantee an adequate response time to aperiodic requests [Ramakrishnan et al. (1995)].

We will assume here that non-multimedia requests are treated as low-priority requests and therefore do not have a guaranteed response time, even though the attempt is made to serve them as soon as possible. They will also be scheduled according to a *FIFO* policy (as

we shall see in the next section, this condition can be relaxed).

The EDL policy combines well with serving non-multimedia requests, as the service of multimedia requests is delayed as long as possible. While the disk is idle it is therefore possible to serve non-multimedia requests. We define this time (*slack time*) as the interval between completion of the last request (n) and the start time of the next one ($n+1$) (see Fig. 5) and calculate it as follows:

$$T_{slack_n} = T_{start_{n+1}} - T_{completion_n}.$$

According to Chetto and Chetto (1989), the EDL algorithm maximizes the slack time in each time interval $[t_1, t_2]$, where t_1 is the release time of the request and t_2 is its deadline.

Immediately after serving a multimedia request, the algorithm can use the remaining time to give priority to non-multimedia requests. For each aperiodic request queued it calculates the available slack time and, if it is greater than or equal to the time needed to serve the request, it serves it. This test is repeated until the slack time runs out or a request waiting to be served requires longer than the time available. In this case it cannot be served and will have to wait until the next slack time. It should be emphasized that forcing non-multimedia request service beyond the available slack time would be very likely to violate the subsequent deadline, even though it would considerably reduce the service times for aperiodic requests and would avoid leaving short residual slack times unused. This will be analyzed in greater detail in the following section.

If at a given instant in a slack time the queue of non-multimedia requests is empty, the disk is idle and the slack time becomes *idle time*. The disk goes back to being busy when a new aperiodic request arrives or at the start time of a multimedia request. This algorithm is of the *work-conserving* kind, which means that the disk will never be idle if there are multimedia or non-multimedia requests waiting to be served. If an aperiodic request arrives while a multimedia request is being served, it will have to wait for the next slack time available, even if the disk was previously idle. If, on the other hand, an aperiodic request arrives at an instant during a slack time and the disk is idle, it is served immediately (*Immediate Server Scheduling*) [Lin and Tarng (1991)].

If there are only sporadic non-multimedia requests, the EDL+FIFO policy guarantees a short service time and still respects all the multimedia request deadlines. If, on the other hand, the volume of non-multimedia traffic is large, aperiodic requests may be excessively delayed. This happens when:

- the size of the non-multimedia request is very large;
- the slack time available is short (e.g. when there is a large amount of multimedia traffic).

To avoid these drawbacks and guarantee a minimal advance in the serving of aperiodic requests, all non-multimedia requests are subdivided into subrequests with a maximum size of one track. This is of great advantage when it is not possible, due to the brevity of the available slack time, to serve an aperiodic request because it is too large: split into several subrequests it can be served in more than one slack time. Figure 6 shows the case of a non-multimedia request which requires 5 time units to be executed. As it cannot be served completely in either the first or the second slack time, it is split into several smaller subrequests and served in subsequent slack times.

This technique implements a sort of preemptive algorithm in a non-preemptive context. We therefore define the *service time* for a non-multimedia request as the time interval between the arrival of the request and total completion of it. By total completion we mean the moment at which all the N disks in the group have responded (see the section on Partial Dynamic Declustering) to their part of the request (or subrequests).

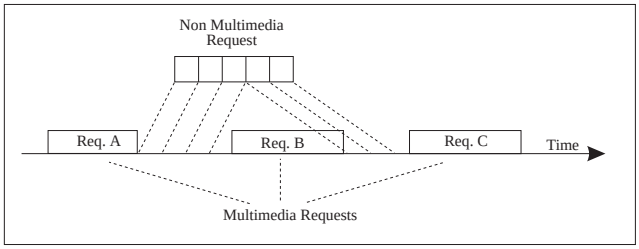


Fig. 6. Splitting a non-multimedia request into several smaller subrequests.

4.3 Advanced Management Policies

In the previous section we considered the EDL algorithm for multimedia requests and FIFO for non-multimedia requests. In this section we will see how system performance can improve considerably if these policies are associated with advanced management policies for multimedia requests and variations on the way in which non-multimedia requests are scheduled. The benefits of these techniques will be examined quantitatively in the section devoted to performance evaluation. Here we will confine the discussion to a description of these strategies.

4.3.1 Random Optimization Technique (ROT)

We will use the *ROT* technique as our point of reference. With ROT we will indicate the fact that there is no optimization in system management. When the scheduler inserts a new multimedia request, the choice of M out of N disks is totally *random* and the data is then reconstructed using the IDA technique. Although this solution is easy to implement and computationally fast, it has the disadvantage of not taking into account the workload on each single disk, so it could choose the disks which are most heavily loaded at that time, thus cancelling out the advantages of the IDA technique. Non-multimedia requests would inevitably be affected by this overloading and their average service time would certainly be longer.

4.3.2 Workload-based Optimization Technique (WOT)

With the aim of choosing only less heavily loaded disks to serve a new multimedia request, we could consider them according to the length of their queues. In practice, however, it has been seen that this strategy only manages to measure the workload in part, as the number of queued requests is not a good index of a disk's workload: there may be several small requests queued or just a few "heavy" ones, and this does not make it possible to measure the actual workload adequately.

A more appropriate measure is given by the *total number of blocks waiting to be read* on a given disk. This value gives a more accurate quantitative estimate of the requests waiting to be served. The *WOT* policy chooses to insert a new multimedia request only on the less heavily loaded disks (i.e. those with the lowest total number[1] of blocks waiting to be read). The benefit of this optimization policy is felt by both multimedia and non-multimedia traffic, the latter having the indirect advantage of a shortening of the average service time.

However, a problem in this policy is due to the fact that the serving of non-multimedia requests may be blocked if the request at the head of the queue is too large to be served in the available slack time. This would lead to a waste because the slack time would not be used at all and would become idle time. Indeed, the situation could repeat itself indefinitely, thus causing an immoderate growth of the non-multimedia queues and their service times. This happens when the deadlines for multimedia requests are very close to each other and the slack time is very short.

In order to use the available slack time as much as possible and obtain as short as possible a service time for aperiodic requests, the strategy for serving non-multimedia requests can be modified. We will adopt the technique called *First Fit*[2], whereby the first request with a service time that is less than or equal to the available slack time is served. In this way the queue of non-multimedia requests is scanned according to a FIFO policy, but any requests which are too large for the available slack time are "skipped" (see Fig. 7).

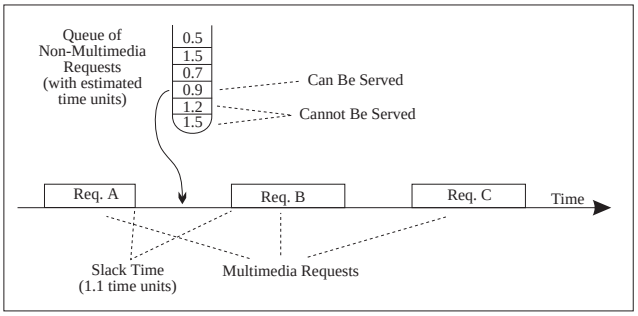


Fig. 7. The First Fit Technique.

[1] By total number we mean the sum of blocks relating to multimedia requests alone.

[2] Other similar techniques, e.g. *Smallest Fit* and *Best Fit*, can be used but, as will be shown in Section 6, their performance is not as good as that of the *First Fit* technique.

In this way the queue is not blocked and the probability of the slack time transforming into idle time is reduced. This strategy is satisfactory because it allows the average service time for aperiodic requests to be kept very low (see section on performance evaluation).

4.3.3 Mid-Point Optimization Technique (MOT)

With the previous methods our aim was to minimize the workload on the disks and the average response time for non-multimedia requests. Up to now we have neglected the fact that the deadlines should be met with a low percentage of violations and as low as possible an anticipation constant. In this way the probability of buffer overrun would be reduced. *MOT* is a technique devised for this very reason and at the same time aims to guarantee a low service time for aperiodic requests. Fig. 8 outlines the problem involved.

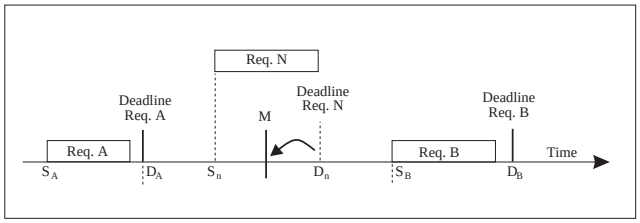


Fig. 8. MOT queuing policy

In choosing the subset of M disks to send the multimedia subrequests to, it is necessary to check whether the insertion of a new request will cause a deadline violation. In the figure the new request has been inserted between the two multimedia requests A and B[3]. It is necessary to ascertain:

1. whether the start time for the new request N is before the deadline for request A, because if it is, the deadline for the new one will certainly be violated (as it would start being served late);
2. whether the deadline for the new request N is after the start time of request B, because in this case request B's deadline would certainly be violated.

The decision about where to place the new multimedia request is based on analysis of the deadline of the previous request (D_A) and the start time of the subsequent one (S_B). Calling M the mean value between D_A and S_B, and calling M_n the mean value between the start time of the new request S_n and the relative deadline D_n, if M_n is to the right of M it will be made to coincide with M (moving the deadline, D_n, of the new request) so as to make the three requests equidistant. If, on the other hand, M_n falls between D_A and M, it cannot be modified. This equidistance makes it possible to reduce the mutual effects between multimedia requests, thus allowing the fixed anticipation they are given to be

[3] The case in which the new multimedia request is the one with a greater deadline than those already queued (in the figure such a case would be inserted on the right of B) can be analyzed in the same way.

reduced. This also means an improvement in the serving of non-multimedia traffic, if the First Fit technique is used.

On the basis of these considerations, the disks chosen will be those in which the distance between D_A and M_n is greatest. Below we give the algorithm for the *Mid-Point* procedure referring to this optimization policy.

```
Procedure Mid-Point(NewRequest, N)
begin
   Let N be the set of disks in a group according to the PDD
      technique;
   Let M ⊂ N be the subset of disks according to the IDA
      technique;
   Let m be the fixed size of the subset M of disks;
   for i := 1 to | N | do
      begin
         Let A be the previous request and B the request after
            NewRequest;
         M := (D_A + S_B) / 2;
         M_n := (D_n + S_n) / 2;
         if (M_n > M) then
            begin
               Moves the NewRequest deadline back by M_n − M;
               Weight_i := M − D_A
            end
         else
            Weight_i := M_n − D_A
      end;
   M = ∅;
   for i := 1 to m do
      M := M ∪ { n ∈ N \ M | Weight_n = max_{j∈N\M}
            Weight_j };
   return M;
end;
```

As far as non-multimedia traffic is concerned, request serving is allowed to overlap partially with multimedia requests. This overlapping, however, must not be greater than the maximum violation allowed by the system, which in our case is equal to the frame viewing time (0.033 sec.).

4.3.4 A Summary of Optimization Techniques

The WOT and ROT optimization techniques can be analyzed from a different point of view. According to the type of request and the moment at which the optimization occurs, in fact, the various policies can be classified as:

- *Optimization for MM*
  If the optimization concerns multimedia requests. This phase occurs during placement of the requests in the disk queues, so before they are served. This category includes the strategy whereby a disks workload is measured according to the number of blocks waiting to be read (A), and the Mid-Point strategy which aims at maintaining a certain equidistance between the various queued requests (B).
- *Optimization for NMM*
  If the optimization refers to non-multimedia requests. This phase occurs during the serving of aperiodic requests. The First Fit technique (C) belongs to this category.

Table 2. Optimization techniques

	ROT	**WOT**	**MOT**
(A)	NO	YES	NO
(B)	NO	NO	YES
(C)	NO	YES	YES

Table 2 summarizes the combination of strategies used in the various optimization techniques.

In general in our work we can distinguish between two classes of optimization techniques: one which, regardless of the type of multimedia traffic, aims to minimize the workload on the disks, and one which, regardless of the workload on the disks, aims to organize the queues in a more disciplined way. The advantage of the former is that it favours balancing of the workload, and also does not cause system saturation. The advantage of the latter is that it reduces the dependence between adjacent multimedia requests and thus the probability of deadline violations. In a real system it may be important to consider a combination of the two classes and configure the system according to the features of the workload. The function mixing the two optimization techniques will therefore take the following form:

$$Weight_i = \alpha W_i + (1 - \alpha) M_i$$

where W_i is the weight given by the workload optimization technique, M_i is the weight given by the Mid-Point technique, and α is the mixing parameter which will depend on the particular workload on the system.

5 Simulation Model

Before giving the results we obtained it is necessary to illustrate the simulation model used and the basic simulation parameters.

5.1 Videos and Arrival Rates

The system continuously serves users who ask to see a video during a *multimedia session*. It is assumed that a video lasts from a minimum of 5 minutes (i.e. the length of a video-clip or the time it takes a user to decide she no longer wants to watch a video) to a maximum of 2 hours (the user watches the whole movie). In the tests performed no account is taken of typical VCR operations such as pausing, rewinding and fast-forwarding. In addition, our system supports the following *admission control* policy: each request is automatically accepted by the server up to a fixed number of multimedia sessions (according to the desired QoS). As regards request interarrival times, we assume that the generations of multimedia sessions (MM) and non-multimedia requests (NMM) are Poisson processes with an average interarrival time ($1/\lambda_{MM}$ and $1/\lambda_{NMM}$). The former is proportional to

the number of users in the system, and $\lambda_{MM} = 0.000333$ (session/sec. per user) is assumed for a single user. The latter varies between 1 and $4req/s$. In addition, the size of the video (in seconds) and that of non-multimedia requests (in blocks) have a random uniform distribution, the former between 5 minutes and 2 hours, the latter between 1 and 500 blocks.

The videos are stored in the *MPEG-1* format which requires a *bit rate* $\leq$ 1.856 Mbit/s. As the variable data flow (we recall that MPEG is a *Variable Bit Rate (VBR)* data compression algorithm), the number of blocks needed to store each second of movie will also be variable. As was done by Krunz and Hughes (1995), we will assume that the distribution of the size of the frames of an MPEG video is *lognormal*, whose density function is:

$$f(z) = \begin{cases} \frac{1}{z\sqrt{2\pi\sigma^2}} e^{[\frac{-(\ln z-\mu)^2}{2\sigma^2}]} & if z > 0 \\ 0 & otherwise, \end{cases}$$

whereas we chose the following pattern for a *GOP (Group of Pictures)*:

IBBBPBBBPBBB.

For the frame rate we assume that of the *NTSC* American TV standard (30 frames/sec.). In addition, all the user buffers have a 1 MByte capacity (enough to store approximately 4.3 seconds of video) and the *significant violation* value is one frame per second (0.033 sec.). Up to 200 users will be assumed in the following.

5.2 Disk Array Modelling

The disk layout adopted in our model is a combination of *Contiguous Placement* and *Scattered Placement* [see for example Gemmel and Chistodoulakis (1992) and Gemmel et al. (1995)]. In contiguous placement data is stored consecutively but the maximum size of contiguous data is equal to one disk track. In this way data is grouped in *variable-sized clusters*. In scattered placement the various clusters of data belonging to the same movie are distributed over the whole disk, thus allowing efficient interleaving of various videos on the disk. In other words, contiguous allocation is used to store single clusters varying in size from one block to a whole track. This choice is accounted for by the fact that the videos are compressed using VBR algorithms, so the space required for storage varies from one video sequence to another and to access a certain fixed quantity of video it is necessary to access a fixed number of variable-size clusters. The size of each cluster is therefore determined file by file.

In order to decrease the frequency of discontinuity in the playback phase, the optimal size for a block has to be chosen in such a way as to minimize variance in the response time and maximize throughput. Small blocks, in fact, mean that the workload is balanced between the disks but there is an increase in the seek and latency times, whereas large blocks improve throughput but deteriorate the balancing of the workload and the variance of the response time. An optimal choice of block size will greatly depend on the number of disks in the system, the maximum number of users who are to access the server at the same time, and their data rate requirements [Shenoy (1995)]. In our model we assumed that the system is a traditional real-time one and that the size of each block is 4.5 KB. We also assumed that access to movies is uniformly distributed. For simulations we used an array of *Seagate ST-43400N 3.5"* disks. Table 3 summarizes the parameters of this disk drive.

Table 3. Parameters of the disk used

Bytes per sector	512
Sectors per track	99
Tracks per cylinder	21
Cylinder per disk	2627
Disk capacity	2.6 GB
Revolution time	11.1 ms.
Single cylinder seek time	1.7 ms.
Average seek time	11.0 ms.
Max stroke seek time	22.5 ms.
Sustained transfer rate	4.6 MB/s.

Table 4. Disk Array Parameters

No. of disks in the system	32
Total system capacity	83.2 GB
Redundancy introduced	33%
No. of movies of 90/120' storable	62/47
Group size	8
No. of disks for IDA reconstruction	6
Block size	4.5 KB
Max. size NON MM requests	500 Blocks

The seek time for each disk is calculated using the following formula:

$$SeekTime(x) = \begin{cases} 0 & \text{if } x = 0 \\ a\sqrt{x-1} + b(x-1) + c & \text{if } x > 0 \end{cases}$$

where x is the seek distance in cylinders and a, b and c are chosen to meet the single-cylinder-seek-time, average-seek-time and max-stroke-seek-time requirements, as described by Lee and Katz (1993). The parameters of the disk array are summarized in Table 4.

6 Performance Results

To evaluate the performance of the system, an event-driven simulator was developed in C language. Simulations were run on an *HP K200* multiprocessor under *HP-UX* operating system, and measurements were taken in the steady state. As already mentioned in Section 4.1, below we will use the expression "multimedia request" to indicate the user buffer filling operation that occurs in each round in a multimedia session. This operation therefore involves transfer of a whole block of frames.

As regards the QoS the system offers users, the approaches generally followed are [Gemmel et al. (1995)]:

- *Deterministic* in which all the system deadlines are guaranteed to be met;
- *Statistical* in which the system deadlines are guaranteed with a certain probability (e.g. 90%).

We adopted a *Quasi-Deterministic* approach. In fact in our model the simulation parameters were set in such a way that the system guaranteed that at least 99% of the multimedia requests did not undergo a delay longer than 0.033 seconds (one frame). This means that the system can lose at most one frame per multimedia request, so the total number of frames delayed is much less than 1% of the total number of frames, i.e. very low. Below we will give the values obtained in the simulations.

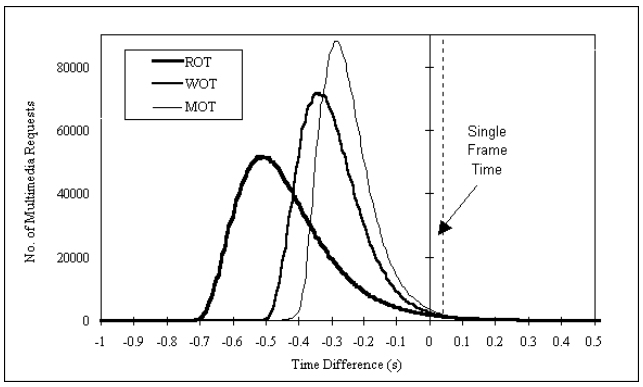


Fig. 9. Completion time distribution for multimedia requests

6.1 Multimedia Requests

For multimedia requests the results are summed up in Figs. 9 and 10. Fig. 9 shows the distribution of the difference between the completion time for multimedia requests and the deadline, comparing the various policies with a high multimedia workload (200 users).

ROT does not fully exploit the advantage of the IDA technique and, to guarantee the required performance, needs a large anticipation on the deadline: with ROT this value is 710 ms (see also Fig. 10).

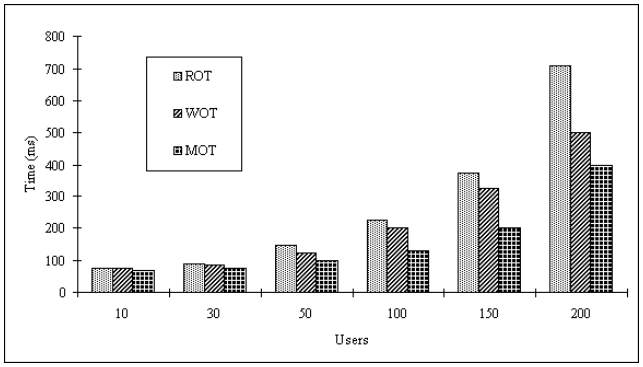


Fig. 10. Anticipation of Deadline

With WOT better results are obtained. In this case, to meet 99% of the deadlines it is sufficient to have an anticipation constant of 500 ms. Recalling that WOT bases choice of the disks on the number of multimedia blocks waiting to be served (see Section 4.3.2), this parameter is decisive if the IDA technique is to be exploited in the disk array.

MOT is the technique which gives the best performance. As can be seen from Fig. 9, the curve is closer to the zero and thus limits any buffer overrun situations. The anticipation constant in this case is 400 ms. With this policy 98.6% of the requests are served within the fixed deadline and only 0.6% are served with a delay ≤ 0.033 sec. Violations of over a frame (0.033 sec.) are equal to 0.8%, i.e. 34695 in 24 hours of system functioning, during which time 4779 multimedia sessions were served. This leads us to conclude that the total number of significant violations per session is on average less than 8, equal to 7×10^{-3}% of the total number of frames.

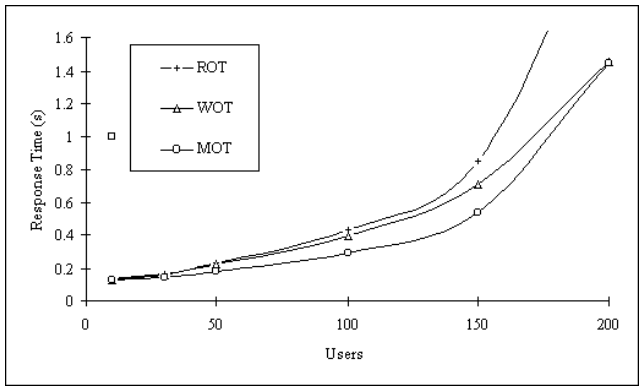


Fig. 11. Response time for non multimedia requests (2 req./s.)

6.2 Non-Multimedia Requests

To assess the response time for non-multimedia requests we compared the various optimization policies again. The results are given in Figs. 11, 12 and 13. In all cases the average size of a non-multimedia request is about 1.1 MB. Fig. 11 is a graph of the system response time for varying numbers of multimedia users. With a non-multimedia traffic of 2 requests per second, assuming MOT and a few tens of users (low workload), a 5% gain in the response time is obtained (0.128 sec.) as compared with ROT (0.134 sec.).

When the number of multimedia users increases, the difference in performance between the various optimization policies becomes more evident. With a load of 200 users ROT guarantees an average service time of 2.449 seconds; WOT gives good results for both multimedia requests and non-multimedia requests. In fact, with a multimedia load of 200 users, it guarantees a response time of 1.453 seconds and a 69% gain as compared with ROT. We recall that WOT uses the First Fit technique to serve non-multimedia requests. It is important to point out that worse results are obtained using the Smallest Fit and Best Fit alternatives. With the same workload, Smallest Fit gives a response time of 1.530 seconds (only a 60% gain) and Best Fit, which is even worse, gives a response time of 1.693 seconds (45% gain).

MOT is equivalent to WOT with both a low and a high multimedia load, but with intermediate loads it gives better performance. This can be seen by observing the curve which is clearly lower than all the others. Fig.

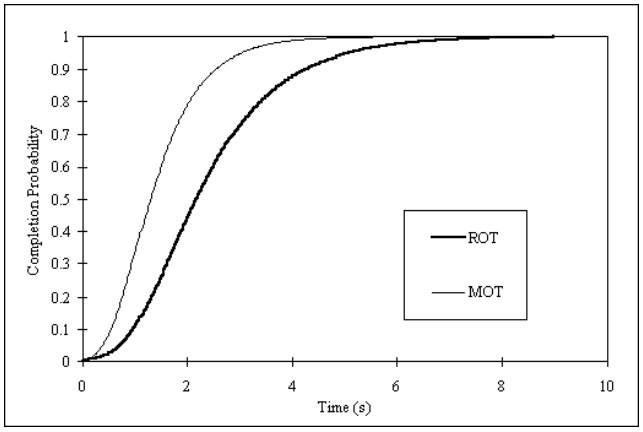


Fig. 12. CDF for non multimedia requests (2 req./s.)

12 shows the CDFs for the ROT and MOT policies with a multimedia load of 200 users.

From this graph the degree of optimization introduced into the serving of non-multimedia requests by MOT can be seen. 80% of these requests are in fact served in less than 2 seconds.

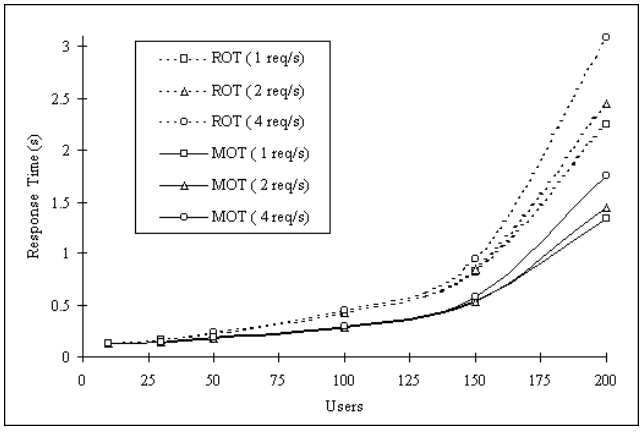


Fig. 13. Response time for non multimedia requests assuming 1/2/4 req./s.

Fig. 13 is a graph of the response times for aperiodic requests with varying amounts of non-multimedia traffic and multimedia users. For the sake of simplicity we have only compared ROT and MOT (the results of WOT fall between these two extremes). The tests were run with a non-multimedia traffic of 1/2/4 requests per second. Whereas with a low multimedia workload MOT maintains a 5% gain as compared with ROT, with a high number of users the improvements introduced by MOT become greater, above all when the amount of non-multimedia traffic increases. If, in fact, with a non-multimedia traffic of 1 req./s the system guarantees an average response time of 1.345 seconds and a 68% improvement over ROT, with a large amount of non-multimedia traffic (4 req./s) the system gives a slightly higher response time (1.764 seconds) and an overall gain of 75% over ROT which saturates the system.

This suggests that the background copy operation can be done at any time (if it is required). With a low multimedia and non-multimedia workload, for example, the reading operation of a 90-minute movie can be achieved in 2 minutes, while with a heavy multimedia and non-multimedia workload the same movie can be read in 27 minutes. If the transfer refers to a less popular movie requested by a certain user, the latter will certainly be able to start viewing it as soon as the transfer starts, and so the latency time to load the video will no longer be 27 minutes but only a few seconds.

From the tests performed it was also seen that the disk queues never saturate with the WOT and MOT policies and that disk usage is equal to 81% with a non-multimedia traffic of 4 req./s and 200 multimedia users, thus showing that the system is working under a heavy workload.

In the tests carried out with the simulator we also assessed the confidence intervals achieved. Despite the high confidence values, the intervals were very short, thus making it useless to include them in the various diagrams. However, we thought it useful to give an example, showing the interval referring to distribution of the completion time, assuming a multimedia workload of 100 users. The interval shown in the diagram has a confidence of 99.9%. Given the closeness between the lower and upper limits, Fig. 14 shows an enlargement referring to the most critical area of the diagram, i.e. the time interval around the deadline (-0.04, + 0.1).

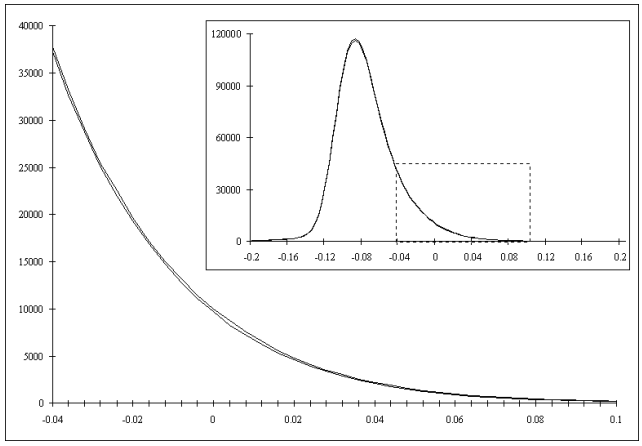


Fig. 14. Confidence interval of 99%

6.3 Scalability of the System

In a further series of tests, we tried to see whether the system continues to perform well even when the number of disks in the array and the number of multimedia sessions increase proportionally. For this purpose, we performed tests with the configuration shown in Table 5, leaving the remaining parameters unaltered (See Table 4).

For this configuration we multiplied the number of disks by four and proportionally increased the maximum number of multimedia users. The results obtained using the ROT and MOT techniques are summarized in Table 6 and Figs. 15 and 16.

Table 5. Scaleup parameters

No. of disks in the system	128
Total system capacity	332.8 GB
No. of movies of 90/120' storable	252/190

Table 6. Scaleup results

	32 Disks 200 Users	128 Disks 800 Users	Ratio
NMM Resp. Time (ROT)	2.449 sec.	2.488 see.	0.98
Antic. Constant (ROT)	0.710 sec.	0.830 sec.	0.86
NMM Resp Time (MOT)	1.449 sec.	1.532 sec.	0.95
Antic. Constant (MOT)	0.400 sec.	0.450 sec.	0.89
Disk Utilization	77%	73%	0.95

All the results in Table 6 refer to high multimedia workloads (200 users with 32 disks and 800 users with 128 disks) and a non-multimedia traffic of 2 - 8 req./sec. As can be seen, the system scales almost linearly to 128 disks and 800 multimedia users. Fig. 16 shows the response time trend with a disk array of 128 storage units, stressing the system up to saturation. We note in particular that the MOT policy still gives acceptable response times with 900 multimedia users. Fig. 15, on the other hand, shows the distribution of the completion time for multimedia requests in a system with 128 disks, 900 multimedia users and 8 req./sec. of non-multimedia workload. As the figure shows, although the ROT policy has a greater fixed advance (1.0s vs. 0.72s) it has a higher number of deadline violations than MOT both in the interval [0,0.033] (single frame time) and beyond 0.033 seconds from the theoretical deadline.

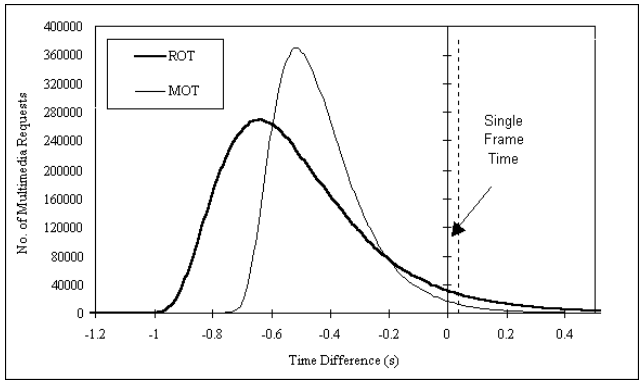


Fig. 15. Completion time distribution for multimedia requests (900 users)

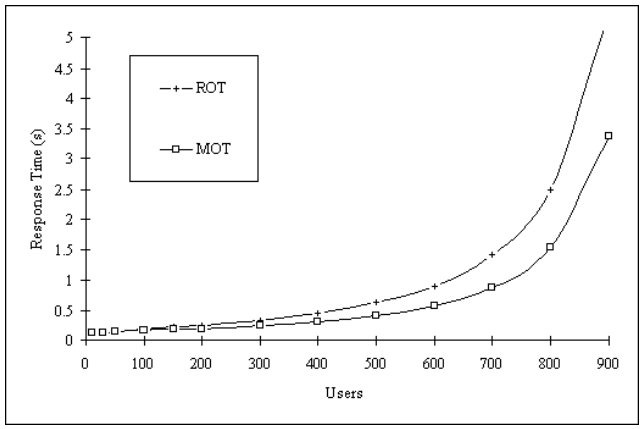


Fig. 16. Response time for non multimedia requests (128 disks)

7 Conclusions

The paper presented a multimedia server capable of managing both CM and normal traffic. This makes the proposed architecture highly flexible and suitable for the current trend towards integrating data of different kinds in a single context.

The system analyzed is based on a traditional disk array in combination with two particular storage-data management techniques - Partial Dynamic Declustering and the IDA. The simultaneous use of these techniques provides an optimal solution to the problem of multimedia servers in terms of performance, reliability and flexibility. As can be seen from the results presented, the designed architecture makes it possible to adopt efficient request queue management policies in order to improve both multimedia and non-multimedia quality of service. We propose to introduce a fixed anticipation to the start time of the request in order to meet its timing constraints. In particular, we present a new scheduling policy, named Mid point Optimization Technique (MOT), which allows to strongly reduce the anticipation of multimedia requests, without increasing the number of violations. This scheduling policy also reduces the response time for non-multimedia traffic. Finally, the simulation tests showed that the proposed solution is easily scalable.

Possible evolutions will be addressed to implementing typical VCR operations such as pause, fast-forward and rewind. It would also be appropriate to develop special admission control and batching algorithms to merge several sessions into one. In this context new scheduling algorithms can be developed, evaluating the system response in the presence of faults as well.

References

D.P. Anderson, Y. Osawa and R. Govindan, "A File System for Continuous Media", *ACM Transactions on Computer Systems*, Vol. 10, No. 4, November 1992, pp. 311-337

S. Berson, R. Muntz, S. Ghandeharizadeh and X. Ju, "Staggered Striping: A Flexible Technique to Display Continuous Media", *Multimedia Tools and Applications*, Vol. 1, No. 2, 1995, pp. 127-148

A. Burns, "Scheduling Hard Real-Time Systems: a Review", *Software Engineering Journal*, May 1991, pp. 116-128

V. Catania, A. Puliafito, S. Riccobene and L. Vita, "Design and Performance Analysis of a Disk Array System", *IEEE Transactions on Computers*, Vol. 44, No. 10, October 1995, pp. 1236-1247

P.M. Chen, E.K. Lee, G.A. Gibson, R.H. Katz and D.A. Patterson, "RAID: High-Performance, Reliable Secondary Storage", *ACM Computing Surveys*, Vol. 26, No. 2, June 1994, pp. 145-185

C. Chen, D.W. Lin and T. Russell Hsing, "Digital Visual Communications Over Telephone Networks", *Journal Of Visual Communication and Image Representation*, Vol. 6, No. 2, June 1995, pp. 97-108

H.J. Chen, T.D.C. Little and D. Venkatesh, "A Storage and Retrieval Technique for Scalable Delivery of MPEG-Encoded Video", *Journal of Parallel and Distributed Computing*, Vol. 30, 1995, pp. 180-189

S. Chen and M. Thapar, "I/O Channel and Real-Time Disk Scheduling for Video Servers", *Proceedings NOSSDAV, Network and Operating Systems Support for Digital Audio and Video*, April 23 - 26, 1996.

H. Chetto and M. Chetto, "Some Results of the Earliest Deadline Scheduling Algorithm", *IEEE Transactions on Software Engineering*, Vol. 15, No. 10, October 1989, pp. 1261-1269

B. Furht, D. Kalra, F.L. Kitson, A.A. Rodriguez and W.E. Wall, "Design Issues for Interactive Television Systems", *IEEE Computer*, Vol. 28, No. 5, May 1995, pp. 25-39

G.R. Ganger, B.L. Worthington, R.Y. Hou and Y.N. Patt, "Disk Arrays. High-Performance, High-Reliability Storage Subsystems", *IEEE Computer*, Vol. 27, No. 3, March 1994, pp. 30-36

J. Gemmell and S. Chistodoulakis, "Principles of Delay-Sensitive Multimedia Data Storage and Retrieval", *ACM Transactions on Information Systems*, Vol. 10, No. 1, January 1992, pp. 51-90

D.J. Gemmell, H.M. Vin, D.D. Kandlur, P. Venkat Rangan and L.A. Rowe, "Multimedia Storage Servers: A Tutorial", *IEEE Computer*, Vol. 28, No. 5, May 1995, pp. 40-49

M.Y. Kim, "Synchronized Disk Interleaving", *IEEE Transactions on Computers*, Vol. 25, No. 11, November 1986, pp. 978-988

M. Krunz and H. Hughes, "A Traffic Model for MPEG-Coded VBR Streams", *Proceedings ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 1995, pp. 47-55

D. Le Gall, "A Video Compression Standard for Multimedia Applications", *Communications of the ACM*, Vol. 34, No. 4, April 1991, pp. 46-58

E.K. Lee and R.H. Katz, "An Analytic Performance Model of Disk Arrays", *Proceedings ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 1993, pp. 98-109

T. Lin and W. Tarng, "Scheduling Periodic and Aperiodic Tasks in Hard Real-Time Computing Systems", *Proceedings ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 1991, pp. 31-38

T.D.C. Little and D. Venkatesh, "Prospects for Interactive Video-on-Demand", *IEEE Multimedia* , Vol. 1, No. 3, Fall 1994, pp. 14-24

A.N. Mourad, "Issues in the Design of a Storage Server for Video-on-Demand", *Multimedia Systems*, Vol. 4, No. 2, 1996, pp. 70-86

K. Nahrstedt and R. Steinmetz, "Resource Management in Networked Multimedia Systems", *IEEE Computer*, Vol. 28, No. 5, May 1995, pp. 52-63

A.L. Narasimha Reddy and J.C. Wyllie, "I/O Issues in a Multimedia System", *IEEE Computer*, Vol. 27, No. 3, March 1994, pp. 69-74

R. Nass, "As Video Codecs Mature, Two Take the Spotlight", *Electronic Design*, Vol. 43, No. 24, November 20 1995, pp. 65-74

A. Netravali and A. Lippman, "Digital Television: A Perspective", *Proceedings of the IEEE*, Vol. 83, No. 6, June 1995, pp. 834-842

J. V. Pavlik, *New Media Technology: Cultural and Commercial Perspectives*, Allyn and Bacon, Boston, 1996.

A. Puliafito, S. Riccobene, G. Iannizzotto and L. Vita, "Modeling of a Multimedia System for VOD Service", in K.K. Bagchi and G. Zobrist (Eds.), *Modeling and Simulation of Computer and Communication Networks Systems*, Gordon and Breach, 1996, pp. 187-201

M.O. Rabin, "Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance", *Journal of the ACM*, Vol. 36, No. 2, April 1989, pp. 335-348

K.K. Ramakrishnan, L. Vaitzblit, C. Gray, U. Vahalia, D. Ting, P. Tzelnic, S. Glaser and W. Duso, "Operating System Support for a Video-on-Demand File Service", *Multimedia Systems*, Vol. 3, No. 2, 1995, pp. 53-65

S. Sahu, Z. Zhang, J. Kurose and D. Towsley, "On the Efficient Retrieval of VBR Video in a Multimedia Server", *Proceedings IEEE International Conference on Multimedia Computing and Systems*, 1997

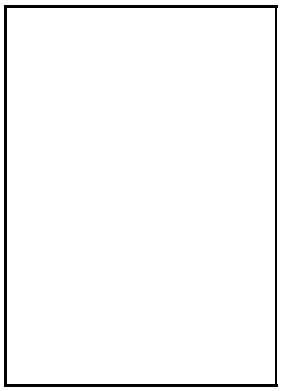
P.J. Shenoy, P. Goyal and H.M. Vin, "Issues in Multimedia Server Design", *ACM Computing Surveys*, Vol. 27, No. 4, December 1995, pp. 636-639

R. Steinmetz, "Multimedia File Systems Survey: Approaches for Continuous Media Disk Scheduling", *Computer Communications*, Vol. 18, No. 3, March 1995, pp. 133-144

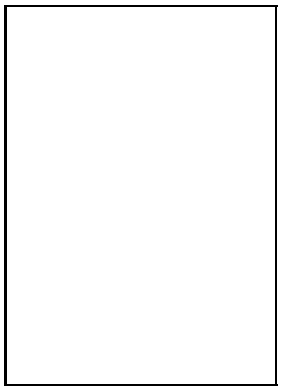
H.M. Vin, A. Goyal and P. Goyal, "Algorithms for Designing Multimedia Servers", *Computer Communications*, Vol. 18, No. 3, March 1995, pp. 192-203

H.M. Vin and P. Venkat Rangan, "Designing a Multiuser HDTV Storage Server", *IEEE Journal on Selected Areas in Communications*, Vol. 11, No. 1, January 1993, pp. 153-164

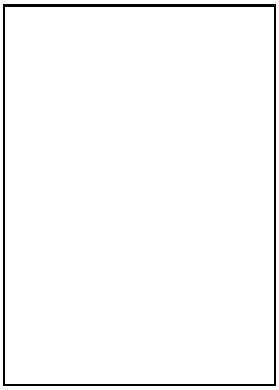
G. Wiederhold, *File Organization for Database Design*, McGraw-Hill, 1987



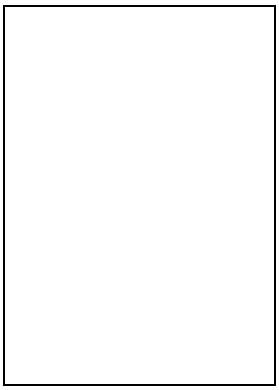
IGOR D.D. CURCIO was born in Milan on '68. From 1986 to 1992 has worked in Milan for several companies as a software engineer and computer science educator. He received the Laurea Degree in Computer Science from the University of Catania (Italy) in 1997. He is an ACM member since 1990, IEEE Computer Society member since 1991, and member of various IEEE CS Technical Committees & Councils (TCMC, TCRT, TCFT, TCSE). His interest area includes parallel I/O performance evaluation, interactive television, multimedia systems and fault-tolerant storage servers.



ANTONIO PULIAFITO received the electrical engineering degree in 1988 from the University of Catania and the Ph.D. degree in 1993 in computer engineering, from the University of Palermo. Since 1988 he has been engaged in research on parallel and distributed systems with the Institute of Computer Science and Telecommunications of Catania University, where he is currently an assistant professor of computer engineering. His interests include performance and reliability modeling of parallel and distributed systems, networking and multimedia. Dr. Puliafito is co-author (with R. Sahner and Kishor S. Trivedi) of the text entitled *Performance and Reliability Analysis of Computer Systems: An Example-Based Approach Using the SHARPE Software Package*, edited by Kluwer Academic Publishers.



Salvatore Riccobene received the electronic engineering degree in 1992 from the University of Catania. Since 1992 he has been engaged in research on multimedia systems with the Institute of Computer Science and Telecommunications of Catania University. In 1996 he completed his Ph.D. studies in Computer Science at the University of Palermo. Currently he is an assistant professor of computer science at the University of Catania. His research interests include performance evaluation, parallel processing, distributed and parallel systems, design of multimedia systems storage servers and network management.



Lorenzo Vita received the electrical engeenering degree from the University of Catania in 1979. Since 1980 he has been cooperaring in many research fields with the Institute of Computer Science and Telecommunication of Catania University, where he was Associate Professor. Actually he is Full Professor of Computer Science at the University of Messina, Italy. His research interests include parallel processing, distributed and parallel systems, fuzzy logic, multimedia systems and network management. He is an IEEE Computer Society member.

This article was processed by the author using the LaTeX style file *cljour2* from Springer-Verlag.