Methodologies for Designing Video Servers

Dikran Meliksetian, *Member, IEEE*, Frank Feng-Kuo Yu, and C. Y. Roger Chen

***Abstract*—This paper proposes an approach for integrating CPU/disk/network scheduling and memory management for supporting a variety of VCR operations and dynamic service changes efficiently. Under this approach we can optimize individual resources, and support a maximal number of clients on a given system. We present a framework of service modeling to characterize the requested video services and identify the scheduling parameters for supporting these services. A number of techniques and methodologies are developed for analyzing the behaviors of disk accesses, network operations, and CPU activities under the loads of both single and multiple clients. We also describe an admission control strategy that utilizes information about all the system resources to determine if a set of video services is acceptable.**

***Index Terms*—Admission control, resource management, scheduling, video server.**

I. Introduction

MULTIMEDIA applications have received a significant amount of attention from both academia and industry. One important multimedia application, broadly used in entertainment, business, and education, is video-on-demand. A number of issues make the design of the video server, in a video-on-demand application, difficult. First, a video server needs to simultaneously provide video services to multiple clients and guarantee the quality of service for each client. Second, a video server needs to manage system resources, including CPU/disk/memory, and schedule network activity so as to utilize maximally the resources, while not overloading the system. Third, a video server needs to be able to support a variety of VCR operations such as playback, fast-forward, slow-forward, pause, resume, indexing, and scrolling. Finally, a user watching a video may change from one service to another service—for example, from playback to fast-forward or from playback to slow-forward. A video server should support these dynamic service changes while efficiently using system resources.

In order to describe precisely the activities between a client and a server, we divide them into three levels: session, transaction, and service. A session is a connection between a client and a server, and its lifetime is defined as the duration from the login to the logout of a video client. Within a session a user can execute two types of transactions, query or video. A query transaction asks for meta-data of the video server. A video transaction is a sequence of video-related activities and is encapsulated by the *open* and *close* of a video file. The major difference between a query transaction and a video transaction is that the latter involves time-critical, video-related activities, whereas the former does not. Once a video transaction is established, a user can initiate a video service, terminate a video service, or switch from one video service to another where a video service is a playback, fast-forward, or slow-forward operation on the chosen video.

Previous work in video server design can be categorized into three areas: storage subsystems, communication subsystems, and operating system support. Gemmell *et al.* [1] present a survey of the design issues of digital multimedia storage servers. This tutorial paper partially reflects research efforts on multimedia applications and systems presented in various publications such as [2]–[7] that concentrate mainly on storage subsystems and assume that network subsystems deliver continuous media to the client sites according to its real-time specifications. Multimedia applications present challenges for network architectures and protocols, because traffic characteristics and performance requirements of these services differ from those of data-oriented, nonreal-time applications [8]–[12]. A major distinguishing feature of our work is that we are primarily interested in the workload of video services, while previous works emphasize either general behavior or other applications. Multimedia applications require operating system support—for example, CPU scheduling, timer manipulation, thread and memory management, interprocess communication, and synchronization—in order to guarantee their performance requirements. Efforts have been put into the development of multimedia operating systems or the extensions to traditional operating systems such as UNIX [13]–[15]. Many papers published in the field of operating system support for multimedia applications benefit from previous research work on real-time system design. One important issue shared by these two areas is scheduling theory [16]–[19].

One major contribution of our work is the integration into a single scheduling framework, under which we can systematically manage system resources and optimize individual resources. We achieve this by separating the retrieving task and the sending task in a video service and selecting the optimal scheduling parameters for each. The retrieving task and the sending task are scheduled with temporal and precedence constraints so that both timing requirements and data synchronization are guaranteed. Our admission control strategy considers all the system resources to determine whether a set of video services is acceptable.

Another contribution of this work is a model for evaluating the performance of a disk subsystem in supporting

Manuscript received May 21, 1999; revised January 11, 2000. The associate editor coordinating the review of this paper and approving it for publication was Dr. Ann Hac.

D. Meliksetian is with IBM, Southbury, CT 06488 USA (e-mail: meliksd1@us.ibm.com).

F. F.-K. Yu is with AT&T, Middletown, NJ 07748 USA (e-mail: franky@puma.mt.att.com).

C. Y. R. Chen is with Syracuse University, Syracuse, NY 13210 USA (e-mail: crchen@syr.edu).

Publisher Item Identifier S 1520-9210(00)02828-5.

multiple video clients. Our model covers a wide range of issues, including the data size of individual requests, the order/number/distribution of requests, the data layout, and the characteristics of a disk such as seek time, rotational latency, transfer rate, disk cache, disk scheduling algorithm, and the size of the command queue. In addition, we consider the impact of resource contention on I/O performance when both disk and network operations are active concurrently. In contrast, many previous works dealt only with a subset of these factors. A major distinguishing feature of our work is that we investigate the performance of network subsystem in a server under the workload of video services.

The remainder of this paper is organized as follows. Section II presents a model for specifying a video service, and investigates scheduling issues for both a single video service and multiple video services. Section III presents the general issue of resource management for video services. Sections IV and V describes the procedures of parameter selection and resource estimation for retrieving tasks and sending tasks, respectively. Section VI proposes an admission control strategy that considers system scalability and resource contention to determine if a set of video service is acceptable. Section VII describes the implementation status and concludes the paper.

II. Modeling and Scheduling Video Services

This section presents a model for specifying a video service and investigates the scheduling issues for both a single video service and multiple video services. The objective of modeling a video service is to formalize an approach to deriving a set of scheduling parameters for a video service based on which the resource requirements can be estimated and the required resources can be scheduled. The modeling task is divided into multiple phases as follows.

In the first phase, a video service is treated as a user request and is characterized by two components, a VCR operation and a video file, where the VCR operation can be normal playback, fast-forward or slow-forward. In the second phase, a video service characterized by an operation-file pair is converted into a display policy that specifies both the data and timing requirements for providing the video service. Display policies of different video services with regard to the same video file must be carefully specified so that the desired visual quality of each video service can be satisfied and service changes during a video transaction can be smoothly supported. Note that different display policies associated with a given video service would have different bandwidth and resource requirements. The display policy is controlled through selective frame dropping and variation of the frame display rate. Selective frame dropping has been proposed as a means for bandwidth control in networked video applications and has been implemented in a number of commercial products. The argument behind it is that visual quality is minimally impacted by selectively dropping frames. As described in the next paragraph, we use selective frame dropping within the context of the fast-forward service, where the visual quality requirements are orders of magnitude lower than regular service.

Because the basic video service provided by a video server is the playback of a video, the display policy assigned to a playback video service is essentially the entire data set, i.e., all the I, P, and B frames, displayed at the recording rate of a video file, e.g., 30 frames per second (fps). Our video server reserves the system resources for the playback video service for the lifetime of a video transaction. The reserved system resources are also used to provide slow-forward and fast- forward video service. Slow-forward video service can easily be supported, because it requires less of the systems resources than the playback video service. The fast-forward video service is supported by specifying a display policy that requires no more system resources than the playback video service. To do so, both the data and timing requirements of a fast-forward display policy must be precisely specified. Notice that the conventional schemes that support fast-forward either by displaying frames at a rate higher than normal playback, for example, all frames at 90 fps or by skipping a limited amount of frames, for example, only I and P frames at 30 fps for the same fast-forward service, might be unacceptable, because both would require a higher bandwidth than the playback video service. Feasible display policies for the fast-forward video service might include skipping all the B frames and sending all the I and P frames at 18 fps, or skipping all the P and B frames and sending all the I frames at 12 fps for a video captured at 30 fps. The idea behind our approach is to provide the common case, i.e., the playback video service, with good visual quality and minimal startup latency, and make the other cases such as the fast-forward video service acceptable.

The display policy is an abstraction for specifying the data and timing requirements of a video service. To schedule all the system resources, a display policy has to be mapped into a set of scheduling parameters as the third phase of service modeling. A video service is accomplished by scheduling two periodic tasks, i.e., a retrieving task and a sending task, with temporal and precedence constraints so that both timing requirements and data synchronization are guaranteed. The retrieving task is responsible for reading the video data from a file on disks to memory, and the sending task is responsible for sending the video data from memory to the network. The motivation for separating the sending task and the retrieving task is to explore the opportunity for overlapping the disk/network activity and optimizing the utilization of individual resources. We use the common dual-buffer design that utilizes separate buffers for a sending task and a retrieving task so that parallelism can be achieved. The purpose of scheduling these two periodic tasks with precedence constraints is to maintain the producer-consumer relationships between them and guarantee data synchronization.

The distinguishing feature in our scheduling model is that it allows us to incorporate retrieving/sending tasks for a variety of video services into a single scheduling framework with the opportunity to optimize individual resources. We achieve flexibility by allowing a retrieving task and a sending task for a single video service to perform at different paces. By doing so, we can select scheduling parameters for tasks that minimize their respective resources. Under this scheme the data retrieved by a retrieving-task instance may support one or more sending-task instances, and special attention to precedence constraint must

be taken into consideration. Definitions of the sending task, the retrieving task, and the precedence constraint are as follows.

A. Sending Task

A sending task of video service VS, denoted by $ST(VS)$, which sends data from memory to the network periodically, is characterized by: $t_{send,1}$, the start time; P_{send}, the sending period; N_{send}, the total number of sending-task instances; and NTU_i, the network transfer unit in period i, where $i = 1, \cdots, N_{send}$.

The parameter NTU specifies the data that should be sent within a sending period. From the point of view of memory access, an NTU is logically contiguous in memory and can be defined by its memory address and its length. For a sending-task instance of video service VS, denoted by $ST_i(VS)$, $i = 1, \cdots, N_{send}$, which sends NTU_i from the memory to the network, we define its ready time as $t_{send,i} = t_{send,1} + (i-1) \times P_{send}$, its completion time as $c_{send,i}$, and its deadline as $d_{send,i} = t_{send,1} + i \times P_{send} = t_{send,i} + P_{send}$. Task instance $ST_i(VS)$ meets its deadline if its completion time is earlier than its deadline, i.e., $c_{send,i} \leq d_{send,i}$.

B. Retrieving Task

A retrieving task of video service VS, denoted by $RT(VS)$, which retrieves data from a file on disks to the memory, is characterized by: N_{read}, the total number of retrieving-task instances; $t_{read,j}$, the ready time of jth retrieving task instance, where $j = 1, \cdots, N_{read}$, and $\{t_{read,2}, \cdots, t_{read,N_{read}}\} \subseteq \{t_{send,1}, \cdots, t_{send,N_{send}}\}$; DAU_j, the data access unit in period j, where $j = 1, \cdots, N_{read}$; $BUF_{read,j}$, the address of buffer for DAU_j, where $j = 1, \cdots, N_{read}$.

The parameter DAU specifies the data that should be retrieved within a retrieving period. From the point of view of file access, a *DAU* is logically contiguous in a video file and can be defined by its file offset and its length. The parameter *BUF* specifies a general strategy of memory management. Because the retrieving task $RT(VS)$ depends on the sending task $ST(VS)$, the ready times of the retrieving-task instances form a subset of the ready times of the sending-task instances, except for the ready time for the first retrieving-task instance. This arrangement implies that a DAU is an integer multiple of NTU's, which simplifies the synchronization between a retrieving task and a sending task. For a retrieving-task instance of video service VS, denoted by $RT_j(VS)$, $j = 1, \cdots, N_{read}$, which retrieves DAU_j from a file on disks to the memory, we define its completion time as $c_{read,j}$, and its deadline as $d_{read,j}$, where $d_{read,j} = t_{send,i}$ and $t_{send,i}$ is the ready time of the first sending-task instance that uses the data retrieved by task instance $RT_j(VS)$. Task instance $RT_j(VS)$ meets its deadline if its completion time is earlier than its deadline, i.e., $c_{read,j} \leq d_{read,j}$.

C. Precedence Constraint

In order to maintain the producer-consumer relationships and guarantee data synchronization between a retrieving task and a sending task, the following conditions must be satisfied. The relation "$x \prec y$" in the following conditions means that task instance x finishes before task instance y.

1) The first condition specifies the constraint for the start time of a sending task with regard to data prefetching. That is, $t_{send,1} = D + t_{read,1}$, where D is the initial delay of the sending task and $D \geq c_{read,1}$.
2) When the disk subsystem of the video server presents transient overload, the retrieving-task instance that misses its deadline is allowed to run to completion, but no corresponding sending-task instances are allowed to proceed. That is, $\forall i, \forall j$, $RT_j(VS) \prec ST_i(VS)$ if $d_{read,j} \leq t_{send,i}$.
3) When the network subsystem of the video server presents transient overload, the sending-task instance that misses its deadline is allowed to run to completion, but no successive retrieving tasks are allowed to proceed so as to avoid the violation of buffer usage. That is, $\forall i, \forall j$, $ST_i(VS) \prec RT_j(VS)$ if $d_{send,i} \leq t_{read,j}$.

In addition, the following two conditions must be satisfied.

1) All the retrieving-task instances must be performed in sequence. That is, $\forall j$, $RT_j(VS) \prec RT_{j+1}(VS)$.
2) All the sending-task instances must be performed in sequence. That is, $\forall i$, $ST_i(VS) \prec ST_{i+1}(VS)$.

The reasons for adopting the run-to-completion policy are as follows. First, both retrieving-task instances and sending-task instances are I/O activities supported by operating systems, device drivers, I/O buses, network adapters, and disk devices. The mechanism, or the application programming interface, for aborting I/O activity may not be supported by the system. On the other hand, the cost of aborting an I/O activity may be very expensive so that the termination operation itself may further violate the timing constraint. Second, Section VI presents a strategy for employing admission control to determine whether a new service can be supported. Consequently, transient overload of system components that causes timing violation is rare under the protection of admission control. Data synchronization with the run-to-completion policy serves only as our strategy for handling exceptions. Finally, one aspect of scheduling video services at a video server is its soft real-time requirement. Unlike hard real-time applications that require strict guarantees concerning deadlines, a video client can tolerate the data delivery delay introduced by either the server or the networks with the help of data buffering. Such delays are essentially absorbed by the client-side buffer and will not create a dramatic impact on visual quality.

The scheduling model for a single video service leads us to a timer/event-driven I/O scheduling for multiple video services. Because of the properties of device hardware and software, I/O activities such as retrieving tasks and sending tasks are nonpreemptive. The scheduling of multiple retrieving tasks, specifically DAU requests, depends on the disk-scheduling algorithms implemented by the underlying disk controllers. Notice that, in the case of a video server with multiple disks, DAU requests can be distributed among different disks where each disk may have different characteristics and its own scheduling algorithm. Multiple sending tasks are scheduled on a first-come-first-serve basis.

III. Resource Management for Video Services

In order for a video server to support the maximum number of video services under the limited resources available on a given system, three major design issues are

1) efficiently supporting individual video services;
2) correctly estimating system scalability;
3) employing admission control algorithms to determine whether a new service can be supported without affecting services already being supported.

For any given system, the fewer resources a single video service requires, the more video services a video server can support simultaneously. Issues of scalability are examined along two dimensions, i.e., as the number of video services and the number of resources involved increase. For a particular resource, the scalability as the number of video services increases evaluates the resource requirements as a function of the number of video services. The characteristics of resources and video files, as well as their relationships, are taken into consideration when scalability is assessed. We also evaluate the impact of increasing the number of system resources (for example, adding more disks) and integrating different system resources (for example, considering the disk and network subsystem as a whole) on the number of video services. Both positive and negative effects of using multiple resources are quantitatively measured. The negative effect of using multiple resources is due to the contention of system resources that occurs when multiple retrieving and sending tasks are performed simultaneously. A video server can accept a new video service only if its resource requirements, together with those of existing video services, are within the capability of the system. Our admission control strategy is based on the capability of disk/network subsystems, the impact of resource contention on I/O performance, and the characteristics of requested video services. The major criteria of admission control are the estimated aggregate resource utilization demanded by a set of video services and the utilization bounds of the individual system resources for given miss ratios.

IV. Parameter Selection and Resource Estimation for Retrieving Tasks

We first consider parameter selection and resource estimation for executing a single retrieving task from a single disk. We then study the scalability issue of executing multiple retrieving tasks from a single disk. The major goals of investigating the execution of a single retrieving task on a single disk are 1) to identify the appropriate range for scheduling parameters of all retrieving tasks from that particular disk and 2) to derive a method for estimating the resource requirements of a given retrieving task whose scheduling parameters are in that range. A video server has a wide range of alternatives when scheduling a given retrieving task. Specifically, what we want to identify is an appropriate range of DAU sizes—subject to a set of performance and system constraints imposed by MPEG compression, timer resolution, buffer requirement, and NTU/DAU relationship—for all the video files stored on a particular disk. Because the variety of video contents and the quality of compression schemes produce variations of data size, an appropriate range rather than a single optimal value for DAU sizes is preferred. Once an appropriate range of DAU sizes is identified, a given video is decomposed into DAU's accordingly and a DAU index is created. At runtime a retrieving task accesses DAU's based on the DAU index within which the temporal requirement is embedded. The major criterion for evaluating a specific set of scheduling parameters assigned to a given retrieving task is the disk utilization, which is defined as the fraction of total disk access time spent in the entire duration or the fraction of a single DAU access time spent in executing a retrieving- task instance. The smaller the disk utilization per retrieving task, the more retrieving tasks the server will support. Two important issues that can affect the reading of data from a disk are the disk cache and the data layout. Although the read-ahead performed by the disk drive may improve the performance of executing a single retrieving task, it has a negative impact on the overall I/O performance under the workload of multiple video streams. In the latter scenario the read-ahead of multiple retrieving tasks can interfere with each other and create difficulty in estimating system scalability and guaranteeing timing requirements. Thus, we turn off the read-ahead feature of the disk cache. Because a noncontiguous DAU may cause an unpredictable positioning delay within the access of a DAU, we require that a DAU be contiguous on a disk, we store the video files such that the DAU's are contiguous. The reading time for a contiguous DAU may depend on its position on a disk because of the variation of the transfer rate, particularly between the inner tracks/zones and the outer tracks/zones. To accommodate this problem, we consider both the mean and the standard deviation of the execution time for reading a contiguous DAU. Thus, the performance of accessing a DAU is statistically predicted. Our methodology for identifying an appropriate range of DAU sizes is to choose an initial range of data sizes based on the typical workload of an MPEG video/audio stream, and to conduct a set of experiments and analyze the performance of measurements. The measurement results suggest a set of guidelines and performance information for decomposing a video into DAU's and scheduling retrieving tasks. These guidelines include, for example, an appropriate range for DAU sizes and retrieving periods with lower disk utilization, and the corresponding buffer requirement per task. In addition, the functions of the mean and standard deviation of DAU read time, i.e., $f_{read}^{mean}(dau\_size)$ and $f_{read}^{std}(dau\_size)$ are also concluded from the measurement results.

When serving multiple retrieving tasks, the overall performance is affected by the data size of individual requests, the order and number of requests, the distribution of requests on a disk, and the characteristics of a disk such as seek time, rotational latency, transfer rate, disk scheduling algorithm, and the size of command queue. Our initial step toward deriving the execution time and disk utilization is to estimate the overhead of the seek time as a function of the number of requests. Modern SCSI disks support multiple read requests via command queuing, which allows the video server to issue multiple read requests and allows the disk controller to determine the execution order based on a disk-scheduling algorithm. For the workload of multiple video services, the length of the command queue is typically not a bottleneck. Consequently, we assume that the command queue can hold all the requests from multiple retrieving

tasks and the disk controller can perform request reordering on all these requests. The estimation of the seek time per request is achieved by using a simulation model. For a given number of requests, the variation of the seek time represents the effect of the distribution of requests on a disk. Such variation becomes less as the number of requests increases due to the seek optimization by the native disk scheduling algorithms. The simulation results produce the mean and standard deviation of seek time per request as a function of the number of requests, i.e., $f_{seek}^{mean}(n)$ and $f_{seek}^{std}(n)$.

The final issue we consider is the relationships between the miss ratios and the disk utilization. The experimental measurement of such relationships, i.e., $f_{read}^{utl}(miss\_ratio)$, is accomplished by evaluating the performance of executing multiple retrieving tasks; an example is shown in Fig. 1. The miss ratio is the metric for evaluating the quality of a disk for supporting a set of retrieving tasks. Given a specific value of the miss ratio that is acceptable by video clients and end users, the corresponding disk utilization is the maximum utilization possible for a given set of retrieving tasks. The estimated aggregate disk utilization and the disk utilization bound, i.e., the corresponding disk utilization for a given miss ratio, provides the basis of our admission control strategy with regard to disk subsystems.

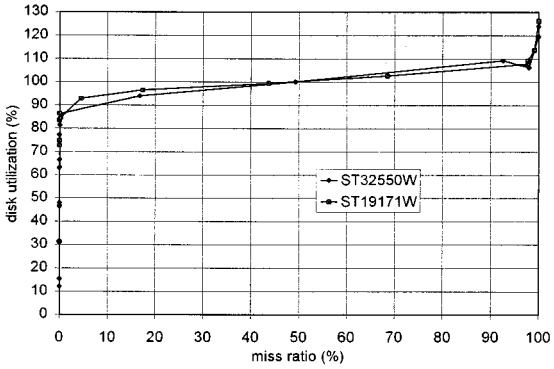

Fig. 1. Miss ratio versus disk utilization.

V. Parameter Selection and Resource Estimation for Sending Tasks

This section first describes our methodology for identifying an appropriate range for scheduling parameters of all sending tasks. We use this information to guide NTU decomposition and estimate the CPU utilization dedicated to a sending task without the concern of scalability. We then present system scalability as the number of sending tasks increases. Finally, we discuss the relationships between the miss ratios and the CPU utilization when a set of sending tasks is supported by a network subsystem.

Our first issue is to identify an appropriate range of NTU sizes, the corresponding range of sending periods, and the packet size for all video files, where an NTU is the sending unit within a sending period and a packet is the basic unit for sending an NTU. Similar to the issues of parameter selection for retrieving tasks, parameter selection for sending tasks is also subject to performance and system constraints. One network-specific issue is to reduce the burstiness of network traffic via parameter selection. Usually, a larger NTU is more likely to produce bursty traffic that may cause the congestion of the network and degrade the reliability of data received by a video client. The major criterion for evaluating a specific set of scheduling parameters assigned to a given sending task is the CPU utilization dedicated to the sending task that is defined as the fraction of total send time spent in the entire duration or the fraction of a single NTU send time spent in executing a sending-task instance. The smaller the CPU utilization per sending task is, the more sending tasks a server will support. The CPU utilization also indicates the trend of the network utilization that is the ratio of achievable throughput to network bandwidth. Our methodology is first to choose an initial range for NTU sizes and the packet sizes based on the typical workload of an MPEG video/audio stream and the system constraints, and second to conduct a set of measurements on the experimental environment. The measurement results suggest a set of guidelines and performance information for decomposing a video into NTU's and scheduling sending tasks. These guidelines include, for example, the percentages for the send time due to packet sizes and NTU sizes, an appropriate range for NTU sizes, an appropriate range for packet sizes, and the corresponding CPU utilization, as well as the mean NTU send time as a function of NTU size and packet size, i.e., $f_{send}^{mean}(ntu\_size, pkt\_size)$ are also concluded from the measurement results. Further details of our approach are given in [20].

When serving multiple sending tasks, the overall performance may be subject to the degradation due to the congestion of I/O path between the main memory and the network interface card. In order to quantify the overhead of executing multiple sending tasks, we measure the performance of scalability as the number of sending tasks increases. The measurement results conclude the maximum utilization overhead as a function of the total bandwidth, i.e., $g(bw)$; an example is shown in Fig. 2

The last issue we investigate is the relationships between the miss ratios and the CPU utilization. The experimental measurement of such relationships, i.e., $f_{send}^{utl}(miss\_ratio)$, is accomplished by evaluating the performance of executing multiple sending tasks. The estimated aggregate CPU utilization and the CPU utilization bound provide the basis of our admission control strategy with regard to network subsystems.

VI. Admission Control

We first introduce the notions of the *resource profile* to represent the information of disk/network resources of a video server and the *service profile* to represent the information of a set of video services. The resource profile consists of three parts: the first and second part describes the characteristics of disk and network subsystems, respectively, and the third part describes the characteristics of resource contention. Consider that a video server has M disk devices $dk_1, dk_2, \cdots, dk_M$, and N network devices $nt_1, nt_2, \cdots, nt_N$. A disk device and a network device are characterized using the attributes shown in Table I.

Resource contention among resources is characterized using an $(M+N)\times(M+N)$ pair-wise contention matrix, where the

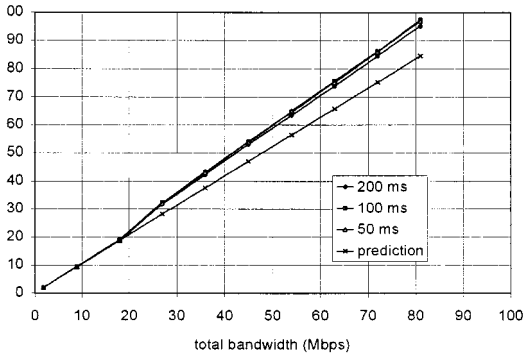

Fig. 2. Network scalability.

element $CM_{i,j}$ for $1 \le i, j \le M+N$ represents the overhead of resource n due to the contention with resource m.

The service profile presents the characteristics of a set of video services. Assume that a video server is serving a service set consisting of K video services VS_1, VS_2, $\cdots$, and VS_K. Each video service is characterized using the attributes listed in Table I.

The resource profile models the system resources based on the off-line measurement/simulation/analysis as described in the last two sections, and the service profile models the video services demanded at runtime. To present our admission control strategy, we begin with the *occurrence vector* that indicates the subset of disk/network resources involved by a service set. The occurrence vector for a given service set SS, OV_{SS}, is a $1 \times (M+N)$ vector of 0's and 1's, where the ith element, $OV_{SS}[i]$, $i = 1, 2, \cdots, M+N$, is 1 if and only if there is at least one video service in service set SS that requires resource i. In an occurrence vector, the first M elements are used to indicate the disk devices and the last N elements are used to indicate the network devices. For a given service set, its occurrence vector serves two purposes. First, the occurrence vector for a given service set indicates which resources should be examined by the admission control strategy. Second, the product of the occurrence vector for a given service set and the pair-wise contention matrix, called *contention vector*, represents the utilization overheads of resources due to the inference of other resources used by the given service set. For a given service set SS, let CV_{SS} be the $1 \times (M+N)$ contention vector, then $CV_{SS} = OV_{SS} \times CM$.

Based on the above assumption, we define the formulas for estimating the resource utilization and propose our admission control strategy as follows. For a given service set SS with K video services, an upper bound on the utilization of disk device dk_m, $m = 1, 2, \cdots, M$, is

$$U_{dk_m}(SS) = CV_{SS}[m] \times \sum_{k|dk_k = dk_m} \frac{1}{P_{read,k}} \times \left({}_m f_{read}^{mean}({}_{\mathrm{DAU}}S_k^{\max}) + {}_m\alpha \times {}_m f_{read}^{std} \times ({}_{\mathrm{DAU}}S_k^{\max}) + {}_m f_{seek}^{mean}(K_m) + {}_m\beta \times {}_m f_{seek}^{std}(K_m) \right)$$

where K_m is the number of video services that involve disk device dk_m. Notice that in the above formula per-device attributes are associated with left subscript m. There are several sources

TABLE I
ATTRIBUTES OF DEVICES AND SERVICES

Disk Device Attributes	
$f_{read}^{mean}(dau\_size)$, $f_{read}^{std}(dau\_size)$	Mean and standard deviation of DAU read time
α	Precision of read time
$f_{seek}^{mean}(n)$, $f_{seek}^{std}(n)$	Mean and standard deviation of seek time
β	Precision of seek time
$f_{read}^{util}(miss\_ratio)$	Miss-vs-utilization function
B_{read}^{miss}	Bound of miss ratio
Network Device Attributes	
$f_{send}^{mean}(ntu\_size, pkt\_size)$	Mean of NTU send time
$g(bw)$	Bandwidth-vs-utilization function
$f_{send}^{util}(miss\_ratio)$	Miss-vs-utilization function
B_{send}^{miss}	Bound of miss ratio
Video Service Attributes	
${}_{DAU}S_k^{max}$	Maximum DAU size
$P_{read,k}$	Average retrieving period
dk_k	Disk device, where $dk_k \in \{dk_1, dk_2, ..., dk_M\}$
${}_{NTU}S_k^{max}$	Maximum NTU size
$P_{send,k}$	Sending period
nt_k	Network device, where
	$nt_k \in \{nt_1, nt_2, ..., nt_N\}$

of variability when measuring disk utilization: the variation of DAU sizes (due to the characteristics of a video file and its DAU decomposition), the variation of read times (due to the location of data on a disk and the intrinsic transfer rate of a disk), and the variation of seek times (due to the number and distribution of DAU requests). Because we want to derive an upper bound on the disk utilization, we use maximum DAU sizes rather than average DAU sizes. Since an appropriate range for DAU sizes is suggested, the variation of DAU sizes for a retrieving task is small. Consequently, the usage of a maximum DAU size actually produces a tight upper bound. On the other hand, the variations of transfer rate and seek time are significant. For example, for a Seagate ST32550W [21] the ratio of the maximum transfer rate to the minimum transfer rate is $(72.0\ \mathrm{Mbps}/49.4\ \mathrm{Mbps}) = 1.46$, while the maximum seek time is $(19.0\ \mathrm{ms}/0.6\ \mathrm{ms}) = 31.7$ times larger than the minimum seek time. To consider the variations of read times and seek times, we resort to a statistical approach and introduce the parameter α as the precision level for estimating the read time, $\alpha > 1$, and the parameter β as the precision level for estimating the seek time per request, $\beta > 1$.

In other words, the value of parameter α estimates the variation of the read time, and the value of parameter β estimates the variation of the seek time per request. Based on Chebyshev's theorem [22], the proportion of values from a data set that will fall within α (or β) standard deviations of the mean will be at least $1-(1/\alpha^2)$ (or $1-(1/\beta^2)$).

An upper bound on the CPU utilization to serve network device nt_n, $n=1, 2, \cdots, N$, is

$$U_{nt_n}(SS) = CV_{SS}[M+n] \times g\left(\sum_{k|nt_k=nt_n} \frac{{}_{NTU}S_k^{\max}}{P_{send,k}}\right) \times \sum_{k|nt_k=nt_n} \frac{{}_nf_{send}^{mean}({}_{NTU}S_k^{\max}, ps)}{P_{send,k}}.$$

Notice that in the above formula per-device attributes are associated with left subscript n. Because we want to derive an upper bound on the CPU utilization, the maximum NTU sizes, instead of the average sizes, are used. Since an appropriate range of NTU sizes is suggested, the variation of NTU sizes for a sending task is small. For example, [20] shows that the maximum NTU size is only 12.5%, on average, more than the average NTU size for a set of five video files. As a result, using the maximum NTU size actually produces a tight upper bound.

A given service set SS is accepted if and only if all the participating resources are within their respective utilization bounds. That is,

$$U_{dk_m}(SS) \le {}_mf_{read}^{utl}(B_{read}^{miss}); \quad \forall dk_m \in \bigcup_k dk_k,$$

and

$$U_{nt_n}(SS) \le {}_nf_{send}^{utl}(B_{send}^{miss}); \quad \forall nt_n \in \bigcup_k nt_k.$$

This admission control policy guarantees with a level of confidence determined by the parameters α and β that no service exceeds its prespecified miss ratio.

VII. Conclusions

This paper addresses some issues for designing a video server with the capability of supporting a variety of VCR operations. We have implemented a video server using a Pentium II 266 MHz running Windows NT. Different versions of video clients are available on Windows 95/98/NT. The underlying network between the server and the clients is a 100 Mbps Fast Ethernet. The results of simulation and measurements indicate that up to 40 MPEG-1 clients can be supported simultaneously. We also developed server-side utilities for preprocessing MPEG videos, configuring SCSI disks, and customizing the data layout on SCSI disks. Three specific objectives, i.e., performance, functionality, and reliability, were taken into consideration during the life cycle of design, implementation, and validation of a video-on-demand system. The concern for performance leads us to fully explore such issues as resource estimation and admission control. It also forces us to reconsider important issues such as the policy and mechanism for dynamic mapping of the on-disk file structure onto the in-memory data structure. In order to support a wide range of VCR operations, we identify a set of primitives between client and server as the foundation and develop corresponding mechanisms to handle these primitives and operations efficiently. We achieve system reliability in several ways. A video client can tolerate severe data losses and recover by synchronizing the delivered video/audio data with the display schedule. On the server side, the exception handling strategy for managing session/transaction/service and synchronizing retrieving/sending tasks of video services enhances server robustness. In addition, system reliability is accomplished by conducting a comprehensive test plan to validate and verify all the functionality, service changes, and boundary conditions of all the operations.

References

[1] D. J. Gemmell, H. M. Vin, D. D. Kandlur, P. V. Rangan, and L. A. Rowe, "Multimedia storage servers: A tutorial," *Computer*, pp. 40–49, May 1995.

[2] D. J. Gemmell and S. Christodoulakis, "Principles of delay sensitive multimedia data storage and retrieval," *ACM Trans. Inform. Syst.*, vol. 10, no. 1, pp. 51–90, Jan. 1992.

[3] P. V. Rangan and H. M. Vin, "Efficient storage techniques for digital continuous multimedia," *IEEE Trans. Knowl. Data Eng.*, vol. 5, no. 4, pp. 564–573, Aug. 1993.

[4] L. A. Rowe and B. C. Smith, "A continuous media player," in *Proc. 3rd Int. Workshop on Network and Operating System Support for Digital Audio and Video*, San Diego, CA, Nov. 1992.

[5] D. Anderson, Y. Osawa, and R. Govindan, "A file system for continuous media," *ACM Trans. Comput. Syst.*, vol. 10, no. 4, pp. 311–337, Nov. 1992.

[6] P. Lougher and D. Shepherd, "The design of a storage server for continuous media," *Comput. J.*, vol. 36, no. 1, pp. 32–42, Feb. 1993.

[7] A. L. N. Reddy and J. C. Wyllie, "I/O issues in a multimedia system," *Computer*, pp. 69–74, Mar. 1994.

[8] C. M. Aras, J. F. Kurose, D. S. Reeves, and H. Schulzrinne, "Real-time communication in packet-wwitched networks," *Proc. IEEE*, vol. 82, pp. 122–139, Jan. 1994.

[9] D. Ferrari, A. Banerjea, and H. Zang, "Network support for multimedia—A discussion of the tenet approach," *Comput. Networks ISDN Syst.*, vol. 26, pp. 1267–1280, 1994.

[10] G. Karlsson, "Asynchronous transfer of video," Swedish Institute of Computer Science, Sweden, Res. Rep. R95:14, 1995.

[11] R. Garg, "Characterization of video traffic," Dept. EECS, Univ. California, Berkeley, TR-95-007, 1995.

[12] S. Böcking, "Communication performance models," Int. Comput. Sci. Inst., Berkeley, CA, TR-95-13, 1995.

[13] T. Burkow, "Operating system support for distributed multimedia applications: A survey of current research," in *Memoranda Informatica 94-48*: Comput. Lab., Univ. Cambridge, U.K., and Faculty Comput. Sci., Univ. Twente, The Netherlands, 1994.

[14] R. Steinmetz, "Analyzing the multimedia operating system," *IEEE Multimedia*, pp. 68–84, Spring 1995.

[15] R. Govindan and D. P. Anderson, "Scheduling and IPC mechanisms for continuous media," in *Proc. 13th ACM Symp. Operating Systems Principles*, 1991.

[16] K. Ramamritham and J. A. Stankovic, "Scheduling algorithms and operating systems support for real-time systems," *Proc. IEEE*, vol. 82, pp. 55–67, Jan. 1994.

[17] J. A. Stankovic, M. Spuri, M. D. Natale, and G. C. Buttazzo, "Implications of classical scheduling results for real-time systems," *Computer*, pp. 16–25, June 1995.

[18] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, Jan. 1993.

[19] K. Jeffay, D. F. Stanat, and C. U. Martel, "On nonpreemptive scheduling of periodic and sporadic tasks," in *Proc. 12th IEEE Real-Time Systems Symp.*, San Antonio, TX, Dec. 1991, pp. 129–139.

[20] F.-K. Yu, "Methodologies for designing video servers," Ph.D. dissertation, Syracuse Univ., Syracuse, NY, 1999.

[21] Specification for ST32550W, Seagate Technology, Inc..

[22] C. W. Helstrom, *Probability and Stochastic Processes for Engineering, 2nd ed.*. New York: Macmillan, 1991.

Dikran S. Meliksetian (S'88–M'91) received the B.E. degree in electrical engineering from the American University of Beirut, Beirut, Lebanon, in 1975, and the M.S. and PhD. degrees in computer engineering from Syracuse University, Syracuse, NY, in 1990 and 1992, respectively.

He is currently with the Internet Technology group at the IBM Corporation, Southbury, CT. Previously, he has held a number of teaching and research positions. Between August 1992 and December 1994, he was an Assistant Professor of Electrical and Computer Engineering at the South Dakota School of Mines and Technology, Rapid City. Between January 1995 and August 1999, he was a Research Fellow at Syracuse University. He is a member of the IEEE Computer Society and the ACM.

Frank Feng-Kuo Yu received the B.S. degree in applied mathematics from the National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 1985, the M.S. degree in applied mathematics from the National Chung-Hsing University, Taiwan, in 1987, and the PhD. degree in computer and information science from Syracuse University, Syracuse, NY, in 1999.

He currently works at AT&T, Middletown NJ. His research interests include computer networks, operating systems, and multimedia communication/storage systems.

C. Y. Roger Chen received the M.S. and PhD. degrees in computer science from the University of Illinois at Urbana-Champaign in 1985 and 1987, respectively.

He has been with the Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY, since 1988 and is currently a Professor. Before joining Syracuse University, he was with Texas Instruments, Inc. Dallas, TX, as Member of Technical Staff. His research areas are in collaboration system, multimedia technologies, object-oriented databases, component based programming, and VLSI CAD. He has published more than 100 technical papers in journals and international conference proceedings and has completed the supervision of more than 20 PhD. students. He has been PI of projects sponsored by the Air Force Research Laboratory, National Science Foundation, and Intel, and Co-PI of projects sponsored by NYNEX, IBM, New York State, and DARPA.

Dr. Chen was program Co-Chair of International Conference on Parallel Processing (1992). He has received best paper awards in the ACM/IEEE Design Automation Conference and Midwest System Symposium, as well as a U.S. patent on the design of network switch architecture.