Journal of Parallel and Distributed Computing **57**, 91–120 (1999)
Article ID jpdc.1998.1521, available online at http://www.idealibrary.com on IDEAL®

Dynamic Scheduling of Multimedia Documents in a Single Server Multiple Clients Environment*

Ishfaq Ahmad, William Y. M. Lai, and Bo Li

Department of Computer Science, The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong

Received February 1, 1998; revised November 12, 1998; accepted November 18, 1998

In a typical single server and multiple client distributed multimedia system, clients may send sporadic requests to the server for certain multimedia documents. The requests must be served with a fast response time and with the required quality of service guarantee. This requires the server to determine the transmission schedule of each multimedia stream while ensuring necessary inter- and intrastream synchronizations. There are two major drawbacks in the existing scheduling algorithms. First, it is assumed that all channels are available at the beginning of the scheduling, but in reality, requests arrive when others are in service; second, the cost of the scheduling itself is usually ignored. In general a feasible scheduling algorithm should have the following features: (1) the schedule must be generated in real time, (2) it should have small scheduling cost, and (3) it must be capable of handling multiple requests from multiple clients. In this paper, we propose two dynamic scheduling algorithms whose worst time complexity is $O(n \log nm + nm)$, where n is the total number of data units in a retrieved multimedia document and m denotes the number of available channels. The salient feature of the proposed algorithms is their inherent dynamic nature which can adjust the scheduling times for each individual request according to the slack time between consecutive requests. If the slack time between two requests is large, the scheduler can run longer in an attempt to find a better solution. This reduces the response time while maintaining a good quality of presentation. Through both simulation and analysis, we evaluate our algorithms and demonstrate their applicability in a realistic environment.

1. INTRODUCTION

Multimedia is an emerging technology that integrates various types of data, such as video, audio, image, graphics, and textual data, into a wide range of applications. With recent advances in computer and network technologies, a variety of

* This research was supported in part by the Hong Kong Research Grants Council under Contract HKUST 734/96E.

0743-7315/99 $30.00

distributed multimedia applications are becoming more feasible. In a distributed multimedia system, users can access diverse applications, such as computer-supported cooperative works [11], video conferencing systems [1, 5], internet surfing (for example, the World Wide Web), and on-demand multimedia services [8] like video-on-demand.

Multimedia data can be classified as either *live* data or *preorchestrated* and *stored* data, depending on its generation time. Live data is generated in real-time (for example, video phones and video conferencing) while preorchestrated multimedia data refers to stored data for which the playout script has already been specified at the time of authoring and storage. Examples are: video on-demand, information clearing-houses, computer-based training (CBT) training, and instruction systems. Generally, a *multimedia document* can be a composite of preorchestrated and stored multimedia data.

For presenting a multimedia document, the presentation system must observe and obey the temporal relations among various multimedia objects at the time of playout. The coordination of real-time presentation of multimedia documents and the maintenance of the timing-order among component media are known as temporal synchronization. The temporal synchronization can be natural or synthetically created. Natural relationships are ones which are implied, such as the simultaneous playback of video and audio. Synthetic relationships are explicitly formulated.

In a distributed environment, a multimedia system may consist of multiple sources/servers or/and clients connected via a network. Unlike the majority of today's point-to-point network applications, emerging multimedia applications such as LAN TV, collaborative computing, and desktop conferencing depend on the ability to send the information from one host to many clients. Since a multimedia document may be a mixture of anisochronous data (for example, text and images) and isochronous data (for example, audio, voice, video, and animation), the management and distribution of heterogeneous data with vastly different storage, communication, and presentation requirements thus pose tremendous engineering challenges [19]. Transportation of isochronous data requires a service supporting real-time delivery and fine-grained synchronization. On the other hand, transportation of anisochronous data poses less stringent end-to-end delivery requirements, yet it requires superior reliability in transmission.

Ideally, if an entire system (including a multimedia server, a network system, and a presentation system) is able to operate in a synchronized fashion, then the presentation of a multimedia document at the client site can be done in a perfect manner. Unfortunately, limitations exist in reality. It may be due to the cost of equipment, exponential increment of users, limited channel bandwidth, and limited processing power. In addition, packets/cells in a real packet/cell switched network (such as IP and ATM networks) suffer statistical queuing delays introduced by the network. Thus, the presentation system at the client site needs to perform proper resynchronization actions to compensate for various delays and missing packets/cells. In addition, the presentation system can also suffer from many constraints such as the low speed of the CPU, I/O transfer rates, and limited buffer sizes.

Existing work has largely focused on the problem of designing disk scheduling and admission-control algorithms that guarantee real-time continuous access to

storage devices. Less attention has been given to integrate real-time continuous media streams with non-real-time, sporadic traffic. Although there has been considerable work in real-time scheduling, the application of the principles of scheduling tasks for execution by the processor and computing the feasibility of such a schedule has not been fully explored in the context of multimedia systems.

In a typical single server and multiple client configurations, clients may send requests to the server for certain multimedia documents. The requests may come in a sporadic fashion and must be served by the server immediately by determining the transmission schedule of each multimedia stream in order to ensure that the requested multimedia documents arrive at the client's site in time. In a sense, this is a dynamic scheduling problem with timing constraints in a soft real-time environment. This requires determining an order of transmission with proper intra-stream and inter-stream synchronization. Server-based scheduling has been studied previously (see [3, 9]); however, investigation was limited to a static and single client scheduling environment [3]. To cope with a real-time environment, the multimedia server needs to generate schedules in real-time. In addition, it must be fast and efficient to meet the timing requirements. In this paper, we propose two dynamic scheduling algorithms.

The remainder of this paper is organized as follows. In the next section, we provide an overview of the multimedia document model used in our study. In Section 3, we discuss the multimedia scheduling problem, and present a brief overview of existing multimedia scheduling algorithms. In Section 4, we describe the proposed algorithms. In Section 5, we present the multimedia system simulator. In Section 6, we provide the details of our experimental methodology, experimental results, and analysis. We conclude this paper in Section 7 by providing some final remarks.

2. MULTIMEDIA SYNCHRONIZATION AND COMMUNICATION MODEL

To enforce presentation and synchronization of multimedia information, a model specifying the timing requirements of various media are essential. A number of conceptual models have been developed to represent such objects [6, 18, 30]. These models can be classified into five categories: graphical models [27], Petri net based models [26], object-oriented models [20], language based models [24], and temporal abstraction models [2]. Some models are primarily aimed at synchronization aspect of multimedia data, while the others are more concerned with the browsing aspect of the objects. The models based on graphs and Petri nets have the additional advantage of illustrating synchronization semantics and, hence, are useful for visual orchestration of multimedia presentations.

The Petri net has been shown to be an effective modeling technique [7, 26, 18], which can, in general, define specific temporal relationships among objects. In distributed multimedia systems, Petri net based models have been used to develop the conceptual models and browsing semantics of multimedia objects [7, 26, 18]. The basic idea in these models is to represent various components of multimedia

objects as places and describe their inter-relations as transitions. These models have been shown to be effective for specifying multimedia synchronization requirements.

2.1. *Object Composition Petri Net* (*OCPN*)

A Petri net based model can be used to specify *object* level synchronization requirements. By taking advantage of this unique characteristic, a model known as the *object composition Petri net* (OCPN) was proposed [18]. The salient feature of this model is the ability to explicitly capture all the necessary temporal relations. In general, an OCPN refers to a Petri net graph (an acyclic directed graph). Each *place* in the OCPN represents the playout of a multimedia object while each *transition* represents a synchronization point.

The attributes of each object such as type and size are assumed to be stored within the object itself. Since an OCPN represents temporal relationships among objects, it is easy to determine playout deadlines for all objects by analyzing their precedence-relations and playout durations.

2.2. *Extended Object Composition Petri Net*

The OCPN model guarantees synchronization at object level, not between small data-units such as a frame. In the absence of proper synchronization between small data-units, objects can suffer from "out-of-synch" problem which can become quite visible. In a distributed environment, multimedia objects are usually too coarse to be used as data units for communication and synchronization. To control the transmission and playout of objects, it is necessary to define some finer-grained data units. Such data units can be obtained by dividing each object into a sequence of finer-grained subobjects of a shorter presentation duration [30]. Such a data unit is called the object synchronization interval unit (SIU). The transmission of an object implies the transmission of a stream of SIUs.

For the purpose of inter-stream synchronization, temporally related objects may be divided into SIUs of equal duration [29]. Each SIU is marked with the synchronization interval number to which it belongs in the form of a header information. These sequence numbers are used at the client site to determine the level of asynchrony between multiple streams and to resynchronize them if needed.

The object type dictates the size and the playout duration of a SIU. For instance, a video object may be divided into multiple SIUs, with each SIU holding information for a complete video frame. Using this choice for the size of a SIU, a video clip can be divided into a sequence of SIUs with a duration of 1/30 of a second each (Fig. 1 shows an example OCPN and the equivalent XOCPN). Although the duration of a SIU for each of the two streams is the same, the size of the SIU (in bytes) is different for the two streams. Each SIU inherits the media type and the quality of service (QoS) requirement of its objects. The QoS requirements may include bandwidth, end-to-end delay, and jitters for the object. From the networking parameters a bound on end-to-end *transit delay* that includes both propagation and quering time can be derived for each SIU. Accordingly, playout deadlines for individual SIUs can also be derived from the object deadlines. Let us assume:

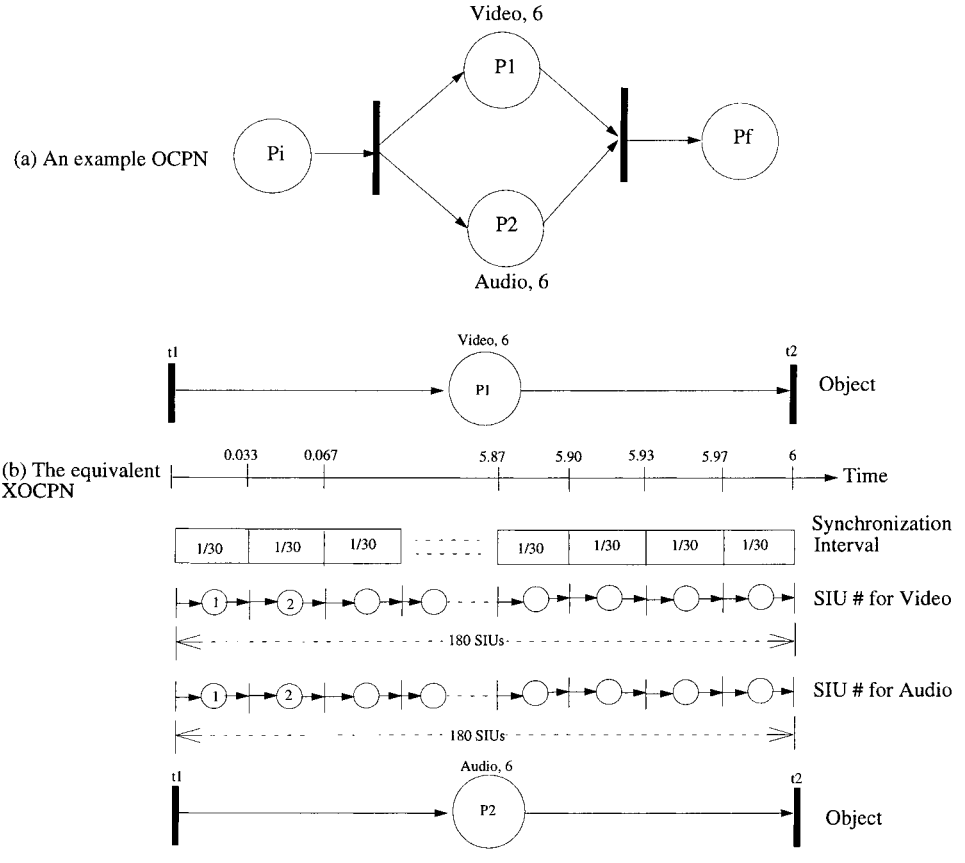


FIG. 1. An example OCPN and its equivalent XOCPN.

Δ_i is the synchronization interval of object O_i, i.e., the length of the SIU

D_i is the playout deadline of object O_i

d_i^j is the deadline of jth SIU of object O_i

then, the *playout* deadline is

$$d_i^j = D_i + (j-1)\,\Delta_i.$$

Mathematically, a formal specification of XOCPN can be found in [30]. The XOCPN is divided into two categories: the *transmitter* and the *receiver* model. At the time of retrieval of a multimedia document, both models need to be generated. The transmitter-XOCPN is used by the multimedia server. According to the transmitter-XOCPN model, the server determines when it should schedule a SIU for transmission. On the other hand, the receiver-XOCPN model is used by the presentation system to synchronize objects while playing. This XOCPN is generated simultaneously with the transmitter-XOCPN when the multimedia document is accessed by the user.

The XOCPN incorporates two new synchronization points, *inter-object synchronization points* (ISPs) and *inter-stream pacing points* (IPPs). Inter-object synchronization points correspond to transition points in the OCPN which in turn correspond to

the transition between two different component objects and impose strict timing constraints among the IPPs according to the Petri net firing rule. Inter-stream pacing points provide the capability for much finer grained synchronization. At each IPP, some information is collected regarding the current state of synchrony among related streams. This information can be derived from the currently playing SIU numbers of each stream. The collected information is then compared among the streams for skew. When some asynchrony (or skew between slow and fast streams) becomes greater than a given threshold value during the playout of the streams, the slower stream can be *pulled up* by skipping the playout of some intermediate SIUs to align it with the other streams.

3. SCHEDULING

The problem of scheduling the transmission of multimedia data in a channel-deficient system can be formulated as a deterministic, non-preemptive parallel machine scheduling problem. It has been identified as a nondeterministic problem, equivalent to the knapsack NP-hard problem [4]. To simplify the problem, we assume the network system performs uni-casting. We also assume the network adopts static resource reservation and provides multiple virtual channels for each client with guaranteed QoS, specifically bandwidth and delay. Based on the XOCPN model, synchronization can be achieved if all multimedia objects in the document are delivered prior to their playout deadlines specified by the specification model.

To ensure the availability of each SIU at the destination before the playout deadline in the presence of random network delays, the SIUs need to be scheduled ahead of their respective deadlines by a factor greater than the maximum expected delay. This prescheduling time is referred to as the *control time*. Since the delays are random, some SIUs may arrive earlier than their deadlines and need to be buffered. The amount of buffering required depends on the control time and the presentation rate required, which in turn is dependent on the network delay distributions. In case of multiple streams, inter-stream synchronization along with intra-stream is also required. The receiver can then buffer the incoming objects until their playout deadlines occur.

The problem of scheduling multimedia documents can be formulated as a deterministic, but on-line, non-preemptive, uniform parallel-processor scheduling problem, where SIUs act like jobs and available channels act like processors. In short, the scheduling of n SIUs on m channels is equivalent to assigning n tasks to m uniform parallel processors.

3.1. Basic Principles of Scheduling

We first define a correctness criterion for the execution of a multimedia transaction and then identify the respective feasible schedules. As with conventional transactions, the semantics of a multimedia transaction determine the correctness of its execution. Unlike conventional transactions, however, the time constraints defined

within multimedia transactions are of prime importance; therefore, we introduce the following semantic correctness criterion.

DEFINITION 3.1 (Semantic correctness criterion). The execution of a multimedia transaction T is correct if the time constraints specified on the transactions in T are preserved.

This semantic correctness criterion is theoretically applicable to the executions of multimedia transactions. However, in a practical, delay-prone system, this criterion cannot be applied directly by the scheduler to enforce the execution of multimedia transactions. Given the pervasive nature of the delay, a strict application of this rule would result in the aborting of a vast majority of multimedia transactions. A more realistic scheduling criterion is therefore needed.

DEFINITION 3.2 (Synchronous schedules). A schedule S of a multimedia transaction is synchronous if all intra-and inter-synchronization points defined among the transactions of the multimedia transaction are preserved.

Thus, synchronous schedules guarantee that no temporal deviation will occur within the presentation at synchronization points by preserving all the synchronization constraints defined in the transaction.

A synchronous schedule, however, can cause large delay between the delivery operations of different transactions during some intervals. Thus, synchronous schedules should not be considered to be completely correct. We introduce the concept of acceptable schedules by incorporating the effect of delays into the definition of synchronous schedules.

DEFINITION 3.3 (Acceptable schedules). A schedule S is acceptable if and only if S is synchronous and all delays occurring between synchronization points are within the permissible range.

When a user requests the retrieval of multimedia documents, the multimedia server retrieves the structure of the document and requests the network to transfer the multimedia objects constituting the document. The deadline of each SIU can only be met when the network provides a set of channels with adequate bandwidth and a bounded delay. SIUs with the same playout deadline may have different end-to-end delays when separate channels are available for them. Consequently, deriving the transmitter-XOCPN should take network delay into account. Also, these SIUs may not be transmitted concurrently because of the limited number of channels.

The duration of channel occupation of SIU_i, which is represented as the tuple $[\beta_i, d_i]$, can be calculated as follows. Suppose

s_i is the size of SIU_i

d_i is the playout deadline of SIU_i determined by the temporal specification

c_k is the kth channel's effective bandwidth

δ_k is the bound on end-to-end transit delay associated with channel k

β_i is the required transmission start time of SIU_i with respect to d_i

T_i^k is the transmission time of SIU_i over channel k, $T_i^k = s_i/c_k$.

To meet its playout deadline at the client site, the server should transmit SIU_i no later than $(d_i - T_i^k - \delta_k)$ which means we can set $\beta_i = d_i - T_i^k - \delta_k$. At the server side, SIU_i occupies the outgoing channel for $[\beta_i, \beta_i + T_i^k)$ time units. By calculating the channel occupancy time for all SIUs belonging to the same object, we can find the channel occupancy duration of each object.

If the network is unable to provide a set of channels with proper QoS parameters required, retrieval of the requested multimedia document cannot be accomplished in a synchronous manner, and the client's request may be denied. Alternatively, some amount of multimedia data can be pretransmitted to the client site and can be played out later. This scheme incurs some initial delay in order to buffer objects at the client site prior to their synchronous playout. The initial delay should be long enough to provide inter-stream and intra-stream synchronization after the client starts the presentation.

We are given a set of n SIUs $S = \{SIU_1, SIU_2, ..., SIU_n\}$, and a set of m channels $C = \{C_1, C_2, ..., C_m\}$. Suppose SIU_i is scheduled for transmission on channel C_k at time α_i according to some scheduling policy. In other words, α_i is the actual transmission time of SIU_i. Then the arrival time, A_i, of SIU_i at the client site becomes

$$A_i = \alpha_j + \frac{s_i}{c_k} + \delta_k.$$

The *tardiness* of SIU_i with respect to its playout deadline can be defined as

$$T_i = \max\{0, A_i - d_i\}.$$

If $T_i > 0$, SIU_i misses its playout deadline which results in intra-stream as well as inter-stream asynchrony. This is illustrated in Fig. 2.

To ensure synchronization and to avoid missing playout deadlines of SIUs, we may need to delay the start of the presentation until the tardy SIUs become available at the client site. Conceptually, we can delay the start by its maximum tardiness as $T_{\max} = \max_{1 \leqslant i \leqslant n} \{T_i\}$.

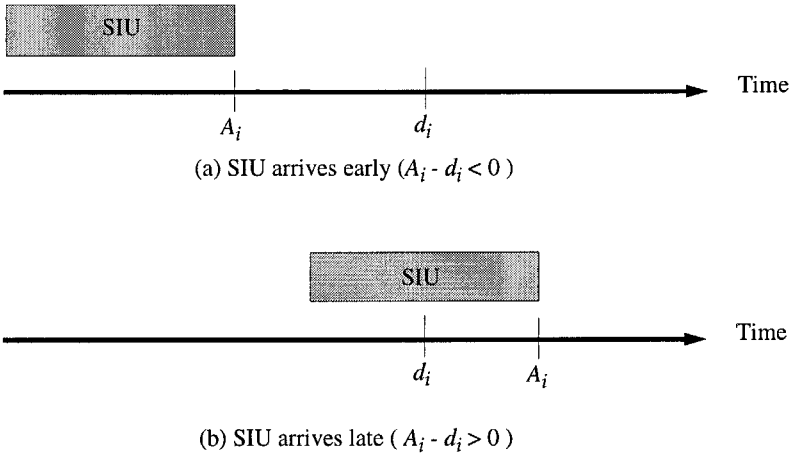


FIG. 2. Effect of SIUs arriving time.

However, sometimes the user may prefer a faster response over a little degradation in the quality of the presentation. This leads to a trade-off between speed and quality.

If we define the induced playout deadlines to be $(d_i + T_{\max})$, where $1 \leqslant i \leqslant n$, then the induced playout deadlines are the earliest playout deadlines that can be met while achieving a synchronous presentation according to a given schedule. Compared to the original playout deadlines d_i, $T_{\max}$ in the induced playout deadlines is the initial delay in the presentation process.

3.2. Related Work

Scheduling in a distributed multimedia system is somewhat equivalent to dynamic scheduling with timing constraints in a soft real-time environment. Dynamic scheduling calls for tasks to be scheduled as they arrive. Soft real-time scheduling involves tasks which have deadlines that are not absolute but can only be missed a certain percentage of the time. There is one difference between scheduling in a traditional soft real-time environment and multimedia systems. In the latter, we can characterize the tolerance to missed deadlines and other errors. For example, in most presentations if an image is displayed 0.5 s late there is no noticeable change in the presentation, as long as it is eventually displayed. In a video presentation if 10 frames are displayed 0.5 s late, their relationship to other frames would make the presentation nonsensical. Clearly, the tolerance to miss the playout deadlines depends on the media and the presentation.

Multimedia scheduling and some of the similar and related scheduling works are known to be NP-hard problems, and, therefore, heuristic methods are widely selected as possible solutions. Various approaches include artificial intelligent and dynamic programming [17, 28]. Some of these incorporate the feedback approach [14] and some use the priority or weighting approach [13]. In the feedback approach, the scheduling algorithm performs differently according to the feedback from the user.

Earliest deadline first (EDF) is an algorithm which is believed to be suitable for scheduling continuous media [17, 32]. EDF is an optimal, dynamic algorithm [25]. Five efficient heuristic algorithms have been proposed in [3] to obtain approximate solutions to the scheduling problem. All algorithms try to preschedule the transmission of multimedia objects so that they arrive at the destination before their play-out deadlines. The objects are scheduled one by one over the earliest available channels. The main problem is that these algorithms are evaluated in a static fashion and single client environments, and thus, their behavior in a dynamic environment is unknown.

A modified earliest deadline first (MEDF) algorithm has been proposed [17]. In this algorithm, a first-come first-serve (FCFS) algorithm is used for non-real-time traffic and virtual deadline-first algorithm is used for real-time traffic. A EDF algorithm is finally applied to the real-time scheduled data elements if they miss their virtual deadlines. The real time data are assigned to have the highest priority to be transmitted, but this is not the case for the non-real-time streams. As a result, synchronization may not be achieved between isochronous and anisochronous streams whenever it is needed.

Rate-monotonic (RM) is the other algorithm which is believed to be suitable for scheduling continuous media [32]. Yau and Lam [32] proposed the use of adaptive rate-controlled (ARC) schedulers with negotiable reserved rates and conditional QoS guarantees. It allows applications to specify a reserved rate (between 0 and 1) and a time interval known as a period (in microseconds). The ARC scheduling is based upon a uniform class of dynamically computed priority values, one for each process. The rate-monotonic scheduling considers only periodic tasks. However, the sporadic server-scheduling algorithm handles aperiodic tasks as well as periodic tasks [17].

Yau and Lam [32] also proposed a firewall property in their rate-controlled algorithm (RC). The algorithm is conceptually similar to the virtual clock (VC) approach [28, 33]. Each process makes progress at its reserved rate, independent of the behavior of other processes. It is achieved by a form of rate control mechanism. A preemptive fixed rate-monotonic priority algorithm is used to schedule isochronous tasks which are the most time-sensitive tasks [22]. The algorithm allows better control of jitter at a finer granularity. Huang and Du [15] showed that the rate-monotonic approach can also solve the multidimensional bin-packing problem.

4. THE PROPOSED ALGORITHMS FOR SCHEDULING

One of the drawbacks of the existing algorithms is that they assume a static status of the system at each scheduling phase. In other words, all channels are always available at the beginning of each scheduling phase. In a real situation, the status of channels may change at any time and some of the channels may be only available after a certain time. The second problem is that these algorithms do not consider the cost of scheduling, that is, the running time of the scheduling algorithm itself. To cope with a real-time environment, the multimedia server needs to generate schedules in real-time as clients access multimedia documents. It must be fast and efficient to meet the timing requirements. As a solution, we propose two server-based scheduling algorithms.

In this section, we first describe two previously proposed static scheduling algorithms called the *forward scheduling* (FS) algorithm and the *backward scheduling* (BS) algorithm [29]. We next propose some variations in these algorithms such that they can be used in a dynamic environment. Finally we propose two new dynamic algorithms. The first algorithm, called a *self-stabilizing scheduling* (SSS) algorithm, schedules multimedia document units to the channels using a slow heuristic algorithm. If the algorithm receives a second request while it is serving the first request, it schedules the rest of the units, beginning with the first request using a fast brute-force algorithm. The second is an overlapped algorithm, namely the *overlapped forward scheduling* (OFS) algorithm, which uses a *pipelining* approach to schedule all multimedia units on-the-fly whenever a new request has arrived. For comparison, we also present a simple approach called the *round-robin scheduling* (RRS) algorithm. This algorithm has a low complexity and, thus, can serve as a benchmark to assess the effectiveness of more sophisticated algorithms.

Each algorithm consists of two steps: sorting the SIUs in the order of their playout deadlines and assigning the ordered SIUs to the channels. We will discuss these algorithms in detail in the following sections. First we define some additional notations.

Let L_j be the schedule on channel j, π_i^j be the channel duration if SIU_i is scheduled to be transmitted over channel j, and let Tx_i^j be the transmission start time of SIU_i over channel j.

4.1. The Forward Scheduling Algorithm

The FS algorithm [30] uses the earliest due date (EDD) rule by sorting SIUs in a nondecreasing order of their playout deadlines. If two SIUs have the same playout deadline, the longest processing time (LPT) rule is applied; that is, a SIU with a larger size precedes the one with a smaller size. The FS algorithm constructs a schedule forward in time, starting with the SIUs needed at the beginning of the presentation and attempts to schedule each SIU on the channel that allows transmitting it the earliest.

FS Algorithm.

1. Sort SIUs in S using EDD rule with LPT rule for the tie breaking
2. Initialize $L_j = \{\ \}$ and $\alpha_j = \text{channel}_j\text{'s earliest available time}$
3. Starting from the head of the list, schedule each SIU in S on the channel that minimizes its arrival time at the client location, specifically.
   $L_j = L_j \cup \{\text{SIU}_i\}$ if $j = \min\{\alpha_k + s_i/c_k + \delta_k\}$ for $1 \leqslant k \leqslant m$
   Set $Tx_i^j = \alpha_j$ and $\pi_i^j = s_i/c_j$
   The arrival time of SIU_i is then $A_i = \alpha_j + s_i/c_j + \delta_i$
   Update α_j to $\alpha_j = \alpha_j + s_i/c_j$
4. $T_{\max}$ is then $\max\{0, A_i - d_i\}$ for $1 \leqslant i \leqslant n$.

4.2. The Backward Scheduling Algorithm

The *backward scheduling* (BS) algorithm [30] uses the EDD rule by sorting SIUs in a nondecreasing order of their playout deadlines; however, it uses the shortest processing time (SPT) rule to sort SIUs with the same playout deadline. In the SPT rule, the SIU with a smaller size precedes the one with a larger size. The BS algorithm tries to schedule a SIU with the largest playout deadline among unscheduled SIUs on the channel that gives transmission start time of a SIU closest to its playout deadline.

BS Algorithm.

1. Sort SIUs in S using EDD rule with SPT rule for the tie breaking
2. Initialize $L_j = \{\ \}$ and
   $\alpha_j = \max\{d_1, d_2, ..., d_n\} + \text{channel}_j\text{'s earliest available time}$

3. Starting from the end of the list, schedule each SIU in S on the channel that minimizes its start time, that is,
   $L_j = \{\text{SIU}_i\} \cup L_j$ if $j = \max\{\min\{d_i, \alpha_k\} - s_i/c_k - \delta_k\}$ for $1 \leqslant k \leqslant m$
   $A_i = \min\{d_i, \alpha_j\}$
   Set $\pi_i^j = s_i/c_j$
   Set $Tx_i^j = A_i - \pi_i^j - \delta_i$
   Update α_j to $\alpha_j = A_i - s_i/c_j$
4. $T_{\max}$ is then $\max\{\min\{\alpha_1, \alpha_2, ..., \alpha_m\} + A_i - d_i\}$
5. For all SIUs in the m schedule L_j's,
   update the transmission start time as $Tx_i^j = Tx_i^j + T_{\max}$

4.3. The Round-Robin Scheduling Algorithm

The *round-robin scheduling* (RRS) algorithm uses a simple heuristic approach. It schedules the multimedia objects to channels one by one in a round-robin fashion. That is, if the current selected channel is j, the next coming SIU will be assigned to channel number $j+1$.

RRS Algorithm.

1. Sort SIUs in S using EDD rule
2. Initialize $L_j = \{\ \}$ and $\alpha_j = \text{channel}_j$'s earliest available time
3. Point to the first channel j
4. Starting from the head of list, schedule the head SIU in the list to the pointing channel
   $L_j = L_j \cup \{\text{SIU}_i\}$
   Update α_j to $\alpha_j = \alpha_j + (s_i/c_j)$
5. Advance the channel pointer to the $j+1$ channel
6. Repeat last two steps until the end of the list

Both the FS and BS algorithms described above are greedy algorithms and satisfy the necessary condition for optimality. For both the algorithms, the complexity for sorting n SIUs is $O(n \log n)$ and the complexity for scheduling SIUs on m channels is $O(nm)$. Thus, both algorithms have the same time complexity of $O(n \log n + nm)$. On the other hand, the RRS algorithm is only a simple scheduling algorithm. The complexity for scheduling SIUs is $O(n \log n + n)$.

4.4. The Self-Stabilizing Scheduling Algorithm

Techniques for scheduling OCPN onto the channels of a multiusers architecture needs to be able to trade-off three factors: scheduling and synchronization costs; deadline missing rate; and response time. The current scheduling techniques typically consider only one or two of these three factors at a time.

The self-stabilizing scheduling (SSS) algorithm exploits the advantages of both the FS and RRS algorithms. The SSS algorithm handles multiple requests and is thus self-stabilizing. This means that if the scheduler needs to serve one request and

there is no new request, it will try to find an optimal solution. The larger the inter-arrival time of the request, the more time the scheduling algorithm gains. However, if the next request arrives while the scheduler is trying to find an optimal solution for the previous request, it will change its strategy and retreat to the RRS approach; therefore, this is an on-line optimization technique. The algorithm is summarized in the following procedures:

SSS ALGORITHM.

1. Sort SIUs in S using EDD rule
2. Initialize all channel's earliest available time
3. Start scheduling using FS
4. If a new request come, the scheduler is triggered by the event
5. Stop scheduling immediately
6. Start scheduling the *rest* of non-scheduled SIUs using RRS
7. Serve the next request from starting from Step 1.

4.5. *The Overlapped Forward Algorithm*

The FS algorithm produces an optimal solution in a timely manner. One of its major drawbacks is its large response time under a real-time environment. We propose a new method called the *overlapped forward scheduling* (OFS) algorithm, which retains the qualities of the FS algorithm but decreases the response time. This is achieved by merging multiple requests. Specifically, if a new request arrives while the scheduler is serving the first request, the SIUs of the new request are merged with the remaining SIUs of the first request. The scheduler, therefore, maintains a pool of SIUs which could be a combination of previous and new requests, and it continues to schedule SIUs as long as the pool is not empty.

OFS ALGORITHM.

1. Sort SIUs in S using EDD rule with the LPT rule for the tie breaking
2. Initialize L_j and $\alpha_j = \text{channel}_j$'s earliest available time for all j
3. Starting from the head of the list, schedule each SIU in S on the channel that minimizes its arrival time at the client location. In other words,
   $L_j = L_j \cup \{\text{SIU}_i\}$ if $j = \min\{\alpha_k + s_i/c_k + \delta_k\}$ for $1 \leqslant k \leqslant m$
   Set $Tx_i^j = \alpha_j$ and $\pi_i^j = (s_i/c_j)$
   The arrival time of SIU_i is then $A_i = \alpha_j + s_i/c_j + \delta_j$
   Update α_j to $\alpha_j = \alpha_j + s_i/c_j$
4. If new request arrives before the last request is finished, the scheduler is triggered
5. Read the corresponding OCPN files
6. Merge the rest (non-scheduled SIUs) with the new coming SIUs
7. Go to the first step and continue the scheduling until finished
8. $T_{\max}$ is then $\max\{0, A_i - d_i\}$ for $1 \leqslant i \leqslant n$.

5. THE SIMULATOR

We have developed a simulator for a single server multiple clients distributed multimedia system. The simulator captures many fine details of a multimedia system, including all of the scheduling algorithms described above, and can simulate a network with various characteristics and functionalities such as allocation of resources for optimum utilization, transportation of each multimedia stream within the specified QoS parameters, and simple inter-switch signalling for low connection setup latency. The communication channels between a client and the server are set up through a number of virtual circuits.

We assume that the network adopts static resource reservation and provides multiple channels with guaranteed QoS, specifically bandwidth and bound on the delay. An example of such a guaranteed network is an ATM-based network which can support multiple virtual channels with various QoS.

Presentation of preorchestrated or live multimedia information requires synchronous play-out of time-dependent multimedia data, according to some specified quality and temporal relations (which are assumed to have been captured at the time of creation of the multimedia information [12, 18]). The precreated multimedia document structures are based upon XOCPN. Anisochronous data (for example, text and images) need to be available to the play-out devices at the destination prior to

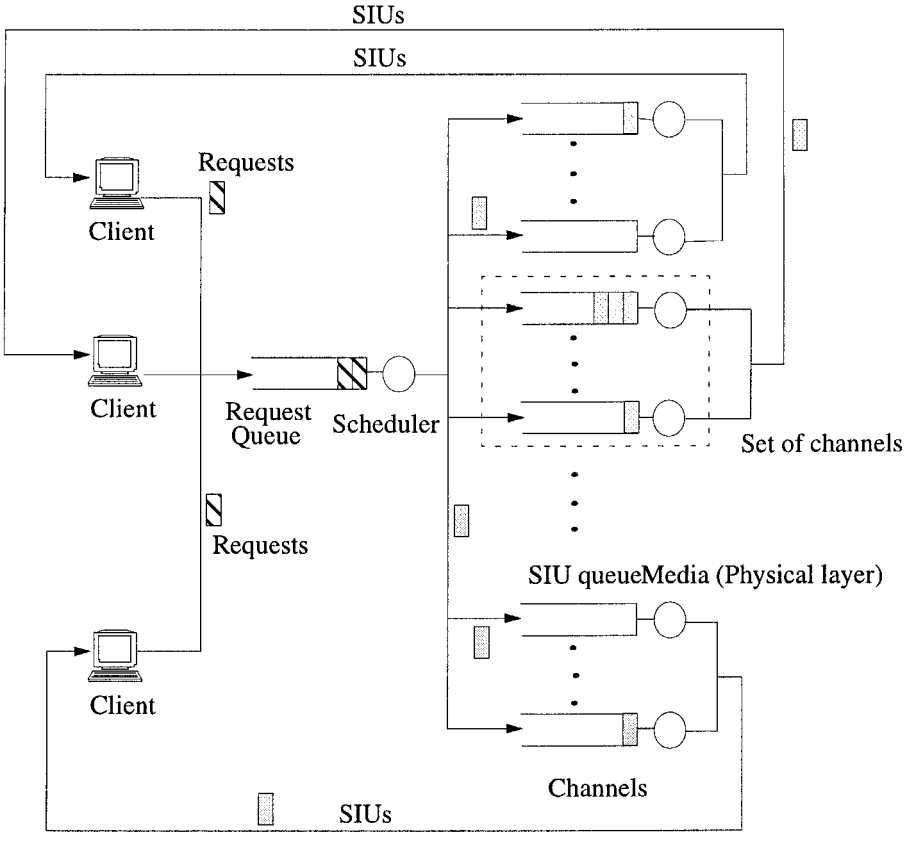


FIG. 3. Scheduler simulation model.

their play-out deadlines. In contrast, isochronous objects (for example, video and audio) can be decomposed into smaller presentation units (SIUs) which are meaningful only to the play-out device. Therefore, anisochronous data occupy exactly one SIU while isochronous data occupy one or more SIUs depending on the play-out duration of that object. Figure 3 illustrates the overall simulator architecture.

Each client gets its own set of virtual channels for transmission. The multimedia scheduler serves one or more clients at a time. Each client makes requests to the server for a number of multimedia documents. The request rate is assumed to be less than the service rate.

The simulator is a discrete-event simulator and consists of three major parts: *requesters*, *scheduler*, and *virtual channels*. The requesters simulate the client activities such as generating requests and receiving multimedia objects. The scheduler simulates the multimedia server which accepts and processes in-coming requests, prepares multimedia objects, and schedules the objects to the available channels. The channels simulate the physical transmission media which transmits packets in the network. The system inter-arrival time of requests is exponentially distributed. The simulator uses a variety of seeds extracted from [10]. The simulator is designed to run long enough to give stable results and to eliminate the transient impact.

6. EXPERIMENTS AND RESULTS

In our experiments we used multimedia documents that include video-clips, audio-clips, and images.

6.1. Workload Generation

Multimedia documents are based on the Petri net model, and are constructed in the following manner. First, we generate an OCPN model randomly, and then we transform it to the transmitter-XOCPN model. The XOCPN includes connection-setup and connection-release operations. The duration of each multimedia document (OCPN) is the total playout duration among the longest path in the model. Each object is associated with a playout deadline. When an object has reached the requester later than its specified deadline, it is counted as a miss.

For each thread in the XOCPN, the *connection-setup* place is added as an output place of the initial transition t_{init} to establish the connection between the transmitter (the server) and the receiver (the client) sites. Subsequently, the *SIU-transmit* places are added according to the sequence of SIUs within a channel. The complete duration of each SIU-transmit place is specified by $d_1 + d_2$, where d_1 represents an artificial delay that is introduced in order to account for the time gap between two successive SIUs in transmission and d_2 represents the transmission delay for the SIU-transmit place because a SIU occupies (only for a specific amount of time) the output port of the transmission system.

Without loss of generality, we assume that audio and video have the same playout duration of 1/30 of a second. The actual number of objects in each interval is randomly generated from a uniform distribution ranging from one to a specific

upper bound, which is defined in terms of the maximum number of concurrent objects. For selecting different types of objects, we perform multiple experiments with the associated distribution of media types. The distribution of media types is described by the tuple (P_v, P_a, P_i, P_t), where P_v, P_a, P_i, and P_t indicate the probabilities for video, audio, image, and textual data, respectively. We have chosen three different sets of probabilities as shown in Table 1.

The sizes of SIUs for each object are generated as discrete values based on object-related distributions. We assume that video and image objects are stored in a compressed form. Consequently, the sizes of SIUs for these types of objects are generated from a uniform distribution ranging from 35 kbits to 285 kbits. An audio object is assumed to be stored without any postprocessing. Hence, a fixed size of 2184 bits per SIU, which is approximately equivalent to 64 kb/s, is used. For textual data, the objects are generated from a uniform distribution ranging from 7 kbits to 15 kbits. Figure 4 shows an example of an OCPN generated in our simulation.

We also investigate video-intensive multimedia documents for video-on-demand services, where a customer looks through a remote database of video films. This usually involves a "set-top box" which interacts with the customers remote control and decodes the MPEG (motion picture engineering group) coded video signal at 1.5 (T1) or 2 Mbit/s (E1). For this purpose, we use proportions of video, audio, and text objects given in Table 2, which according to [8] are the most appropriate combinations for such applications.

6.2. *Comparison Methodology*

Comparison among the algorithms is a "cyclic problem" since we cannot compare the performance of each algorithm with respect to others without knowing their relative performance. The load of the scheduler cannot be more than unity, implying that the request rate can never exceed the service rate. In other words,

$$\rho \leqslant 1, \qquad \mu \geqslant \lambda, \qquad \frac{\lambda}{\mu} = \rho,$$

where ρ is the load, μ is the service rate, and λ is the request arrival rate

Obviously, without fixing the values of any two variables, it is not possible to measure the third one for comparison. Therefore, we first fix the request rate to be an input parameter. The next step is to decide whether we should fix the load or

TABLE 1

Distribution of Multimedia Objects

Set number\Probabilities	P_v	P_a	P_i	P_t
Set 1	0.25	0.25	0.25	0.25
Set 2	0.4	0.4	0.1	0.1
Set 3	0.1	0.1	0.4	0.4

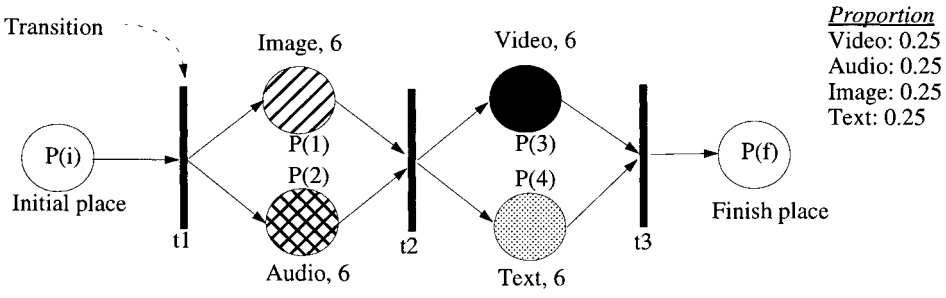


FIG. 4. An example of a randomly generated OCPN model.

the service rate. If we fix the load, the performance comparison of the algorithms is not meaningful since their scheduling times are different.

To handle this problem, we first measure the average scheduling times of each algorithm using a fixed sequence of requests; that is, we execute each scheduling algorithm without the simulator for all of our test data and measure the actual execution times of all algorithms. Table 3 summarizes the results in terms of the scheduling time per SIU.

In comparing the performance of these algorithms in terms of their scheduling times, the RRS algorithm is the fastest and the BS algorithm is the slowest. The FS algorithm is two times slower than the RRS algorithm. The performance of the SSS algorithm lies in between that of the FS and RRS algorithms. Note that, in our experiments, we did not use the average benchmark value of the processing time of SSS. Since SSS is a combination of FS and the RRS, we use the combination of their values, respectively. If p is the number of the SIUs being scheduled by the RRS, then the time for scheduling n SIUs using SSS is given by

$$(n-p)\times\mu_{FA}+p\times\mu_{RR}.$$

These timings were then used in the simulator to simulate the scheduling times of the algorithms. Thus, in a sense, our simulation is a trace-driven simulation. We measured the performance of the scheduling algorithms with more than 2200 samples. The first set of the experiments is for mixed media services, such as computer based training (CBT) with different combinations of objects from of all streams. The second set of the experiments is for a typical video on demand service, with only video and audio objects. The third set of experiments is for a typical slide presentation with objects from the text and image streams.

TABLE 2

Distribution of Video-on-Demand Multimedia Objects

Set number\Probabilities	P_v	P_a	P_i	P_t
Set 4	0.72	0.18	0.0	0.1

TABLE 3

Scheduling Algorithms Benchmarks (Second per SIU)

Algorithm\Machine	Ultra Sparc 1/170	Sun Sparc 5	Sun Sparc IPC
FS	0.000016	0.000046	0.000166
BS	0.000021	0.000062	0.000226
RRS	0.000008	0.000025	0.000086
SSS	0.000012	0.000037	0.000129
OFS	0.000016	0.000046	0.000166

6.3. Documents with Mixed Media

For the first set of simulations, we used multimedia documents that consist of two or four intervals, each with a duration of two to six time units. In each interval, the maximum number of concurrent objects was restricted to two or four. We also chose three sets of distributions for media types (see Table 4).

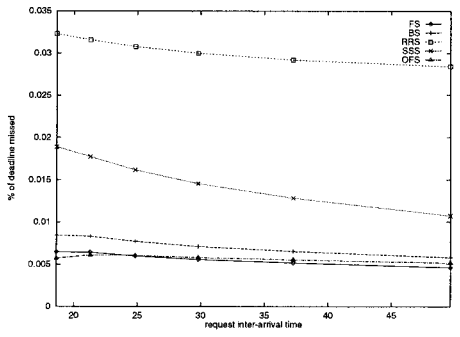
To set up a simple environment for comparison, we investigate the effects of using different numbers of virtual circuits. The sum of the virtual circuits' bandwidths is fixed. We select 640 Kbps as the total bandwidth per client in our studies because it allows a clear observation of results. For the transit delay of each channel, we selected a uniform distributions with a range from 0 to 0.05 time units. For this set of experiments, a total of 240 multimedia document instances were generated by varying the structure of the multimedia documents. The video clips are assumed to be MPEG encoded streams. Table 4 summarizes the design of the first set of simulation experiments.

Figures 5 to 7 depict the results of the mixed media using different algorithms. For the results shown in Fig. 5, we varied the request inter-arrival rate while keeping the shifting time fixed as 14 time units. As indicated by these results, the FS, BS, and the OFS algorithms yield better solutions in terms of handling the deadline missing rate. In terms of response time, OFS outperforms both FS and BS. On the

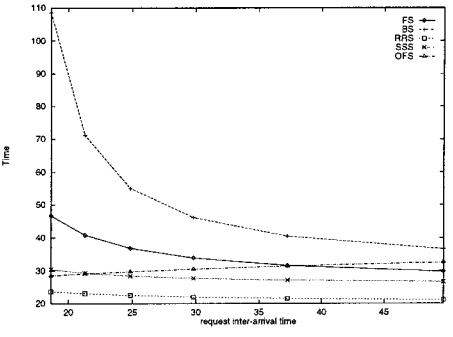
TABLE 4

Experimental Design for Simulation

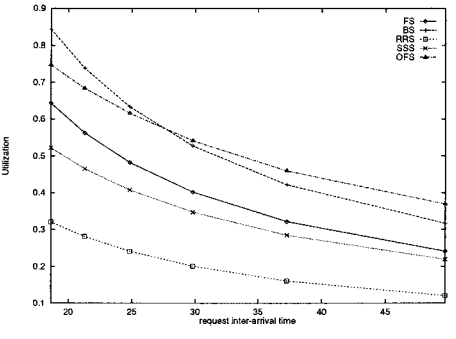
	Values used	No. of varies
Number of intervals	2, 4	2
Duration of intervals	2, 6	2
Maximum number of concurrent objects	2, 4	2
Distribution of media types	(0.25, 0.25, 0.25, 0.25)	3
(P_v, P_a, P_i, P_t)	(0.4, 0.4, 0.1, 0.1)	
	(0.1, 0.1, 0.4, 0.4)	
Distribution of the transit delay	U[0, 0.5]	1
Number of combinations	N/A	24
Problem instances/combination	N/A	10
Total number of instances	N/A	240
Number of channels	10, 8, 5, 2, 1	5
Number of clients	5	1



(a) Deadline missing rate .



(b) Client response time.



(c) Scheduler utilization.

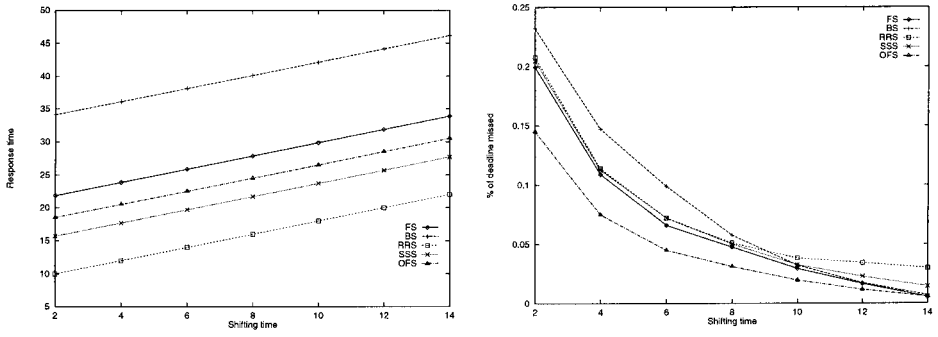
FIG. 5. Comparison of algorithms (mixed media).

other hand, SSS performs even better than OFS in terms of response time, since it exploits the advantages of both FS and RRS. As the inter-arrival time of requests increases, the deadline missing rate for all algorithms decreases gradually.

The response time (see Fig. 5b) of all algorithms, except OFS, decreases as the request inter-arrival time increases. This is because when a new request is received before the last one is finished, OFS merges the requests together. In other words, the new request does not suffer from the queuing delay. The more frequent the requests come, the faster the average response time. The queuing delay, therefore, is a factor which affects the response time, that is, the response time is high if the queuing time is high. Since there is less overall merging in the case of longer request inter-arrival time, there is more queuing delay and, as a result, the average response times increases slightly.

The BS algorithm is the most complex algorithm, and thus creates a higher load for the scheduler (see Fig. 5c). In contrast, RRS creates the least amount of load. The load of SSS lies in between the load of FS and RRS. In general, the load of the scheduler is dependent only on the request arrival rate. The load of the scheduler using OFS is higher than that of FS because of the merging of the requests.

By varying the shifting time (that is, the initial delay), different algorithms perform differently as shown in Fig. 6. It is obvious that the fixed arrival rate gives a direct proportional relationship between response time and shifting time in our test bed environment. The most interesting aspect of these experiments is the



(a) Response time vs shifting time.

(b) Deadline missing vs shifting time.

FIG. 6. Deadline missing probability and response time vs shifting time.

deadline missing rate distribution. Clearly, the deadline missing rate decreases as the shifting time increases. The BS algorithm performs the worst with a small shifting time. This is mainly due to the high processing overhead imposed by BS (more SIUs cannot meet their deadlines on time after the start of the presentation). However, BS performs better after a certain critical point. The best performance is still obtained by OFS.

Figure 7 shows the combined impact of different numbers of channels and request inter-arrival times used by each client. As mentioned previously, each client is connected to the multimedia server with a fixed total bandwidth. When the total number of channels increases, the deadline missing rate decreases. This is mainly because with more channels more SIUs can be delivered concurrently on time without the need to wait in a queue. Another observation is that the difference in the deadline missing rate of various algorithms becomes less when there are more channels available. As the number of channels increases, the corresponding bandwidth of each channel decreases, and, consequently, the transmission time of SIUs becomes longer. In general, each algorithm does not perform better or worse in terms of deadline missing rate starting after a certain number of channels (e.g., five).

6.4. Video-on-Demand

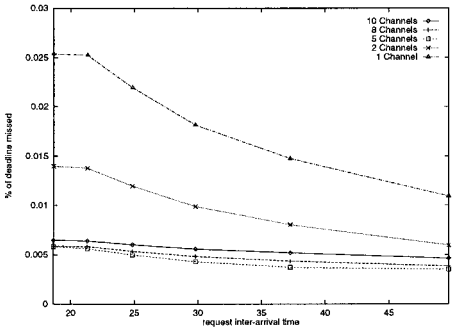
In the second set of experiments, we concentrate on video-on-demand type of services. We changed the probability distribution of media types as shown in Table 5. The proportion of media types is based on VOD service.

Considering the impact of the arrival rate (see Fig. 8), we find that SSS still performs very well for the video-on-demand multimedia service.

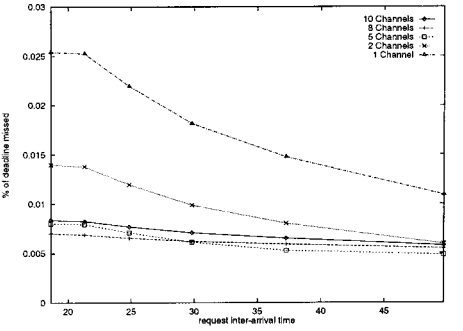
From Fig. 8b we can observe that most scheduling algorithms cannot support the video-intensive multimedia system. The load at the scheduler can become greater than one, which can eventually lead to an infinitely long queue.

The reason is that video-intensive multimedia documents contain more SIUs, as well as a large size of each SIU. The scheduler needs more time in each scheduling phase. The SIUs need to wait in queues for longer periods of time.

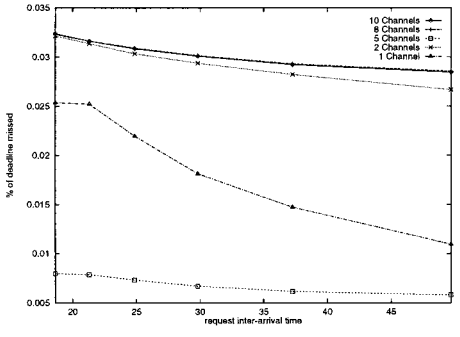
The relative performance of all algorithms in terms of the deadline missing rate are shown in Fig. 9. We can observe the increase in the deadline missing rate with



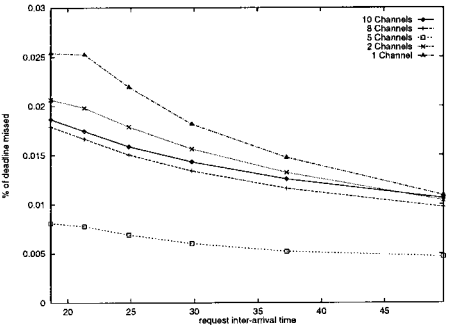
(a) The FS algorithm.



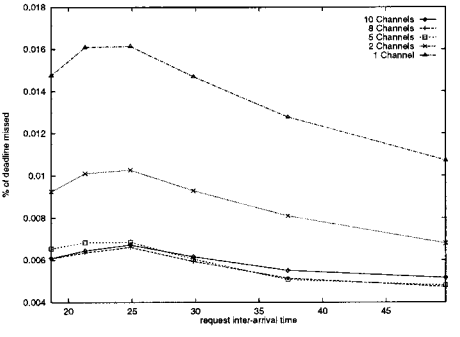
(b) The BS algorithm.



(c) The RRS algorithm.



(d) The SSS algorithm.



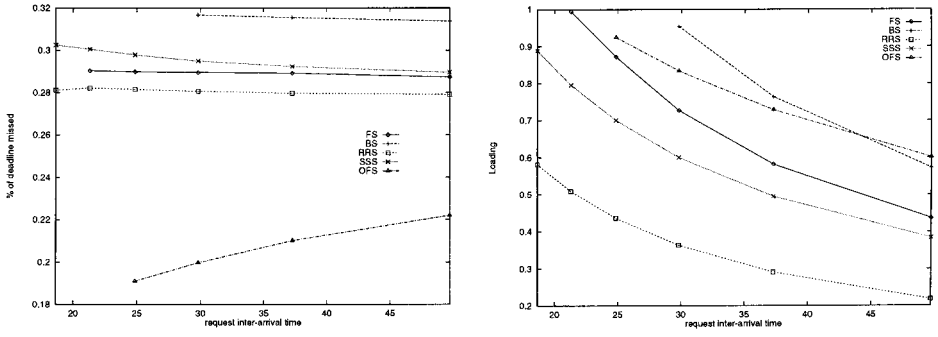
(e) The OFS algorithm.

FIG. 7. The deadline missing vs arrival rate for each algorithm.

TABLE 5

Parameters for VOD Experiments

	Values used	No. of varies
Duration of intervals	[900, 1800], [2700, 3600]	2
Distribution of media types (P_v, P_a, P_i, P_t)	(0.72, 0.18, 0, 0.1)	1



(a) The deadline missing probability

(b) The scheduler load at a very high request rate.

FIG. 8. Deadline missing probability and response time at a very high request rate (VOD).

the increase in the request inter-arrival time generated by OFS. This is in contrast to the previous cases because each SIU occupies a channel for a longer period of time during the transmission. In other words, T_i^k is large. The probability of concurrent transmission among the user increases when the inter-arrival time decreases. Therefore, we can obtain a reduction in the deadline missing rate through efficient pipelining effects. However, the deadline missing rate does not drop rapidly using FS, BS, RRS, and SSS.

Figure 9 also shows the effects of the number of channels in a VOD system. The overall observation is that the performance improves as the number of channels increases, which is similar to the previous set of experiments. With more channels, chances of transmission of SIU increase without blocking others SIUs. For these experiments, RRS generates a stable deadline missing rate rather than the poor performance in the last experiment. This is again due to the fact that the video-intensive multimedia documents contain large sizes of SIUs. Video stream and audio stream are both periodic and, hence, the variation in size is small. On the other hand, the "get-and-pack" strategy in a round robin fashion generates reasonable results.

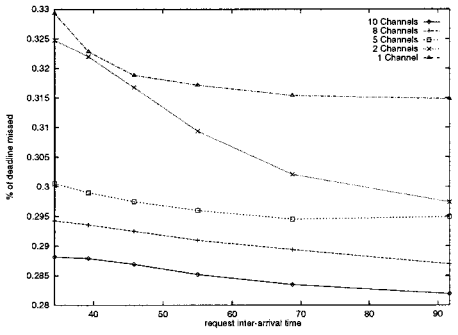
6.5. Multimedia without Video

In the third set of experiments video objects were omitted. The main objective of these experiments is to study how the performance varies among the five different algorithms for a slide-like presentation over the network. In this set of experiments the distribution of media types was redesigned as shown in Table 6.

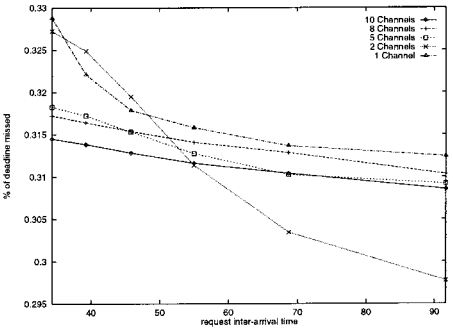
In contrast to the video-on-demand documents, a slide-like presentation document contains a relatively small set of SIUs and the size of each SIU is small. From the overall performance observed, all the algorithms perform well.

6.6. Algorithm Analysis

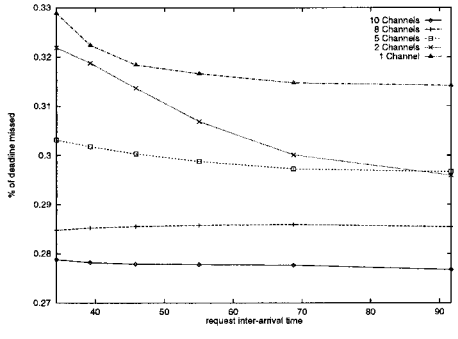
In this section we analyze the FS, OFS, and SSS algorithms statistically using both mathematical and experimental techniques. We start by examining the deadline missing rates of these algorithms which were obtained experimentally by



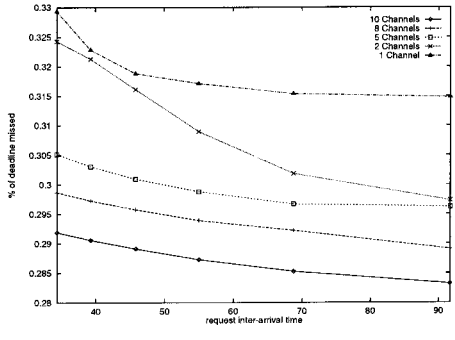
(a) The FS algorithm



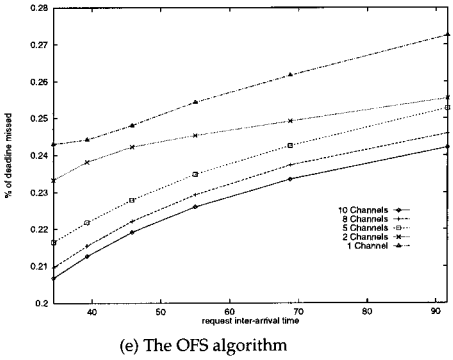
(b) The BS algorithm



(c) The RRS algorithm



(d) The SSS algorithm



(e) The OFS algorithm

FIG. 9. The deadline missing (VOD) vs arrival rate.

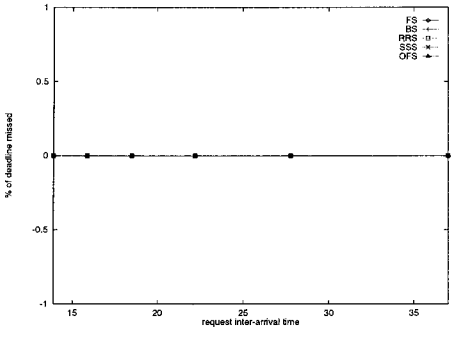
TABLE 6

Parameters for Slide-like Presentation Experiments

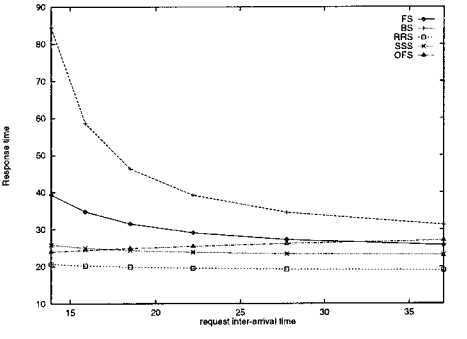
	Values used	No. of varies
Duration of media types (P_v, P_a, P_i, P_t)	(0, 0.33, 0.33, 0.34)	1

adjusting the arrival time of the next request. The arrival time, x, of the second request can be late, early, or at the end of the first request's scheduling as shown in Fig. 11. The length of each block represents the required scheduling time. In this experiment, we designed a new series of samples. Each sample contains two multimedia documents with $(P_v, P_a, P_i, P_t) = (0.25, 0.25, 0.25, 0.25)$ as the probability distribution of the media objects. Figure 12 shows the obtained results. The deadline missing rate of FS remains constant for all of the arrival times. The SSS algorithm performs with an increase in the arrival time while the deadline missing rate of OFS drops at the start and grows at the end.

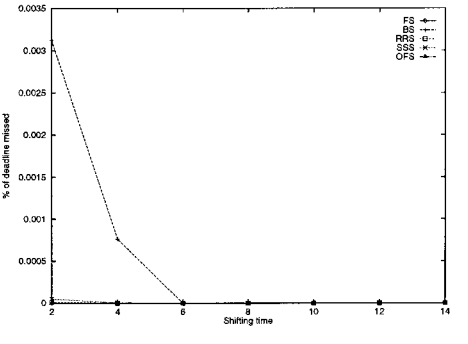
The constant rate generated from the FS algorithm is mainly due to the nonpreemptive feature of the scheduler. No matter when the second request arrives, the scheduler can only be available to handle the second request at the end of the first scheduling phase. This leads to a constant assignment of the SIUs into the channels with no change in the performance. When the second request arrives before the scheduler finishes the first one, more SIUs from the first request would be scheduled using RRS if SSS is employed. This causes an increase in the deadline missing rate when the second request arrives much earlier. In the case of OFS, the requests are merged once the second request has arrived. Before the merging of the requests, some of the SIUs from the first request may not be scheduled in advance since their deadlines occur later. The OFS algorithm has the advantage of delaying the SIUs while trying to minimize the deadline missing rate. The delayed time slots are



(a) The deadline missingrate .



(b) The response time.



(c) The deadline missing probability versus shifting time

FIG. 10. Results for multimedia documents without video.

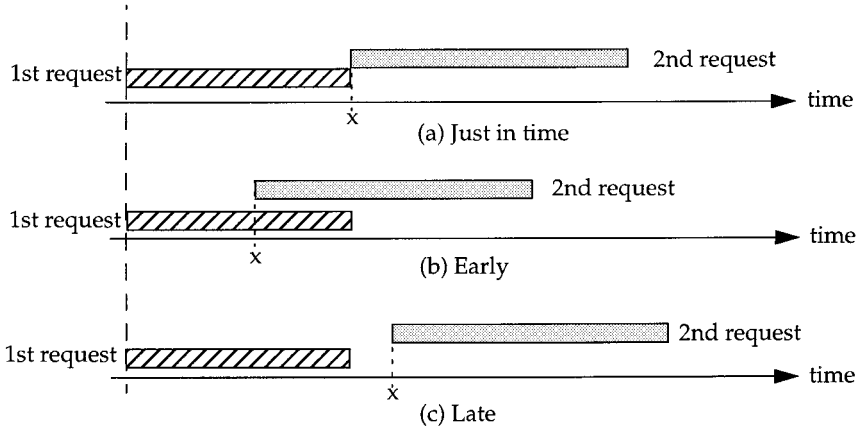


FIG. 11. Relationship between two requests.

available for the SIUs from the second request. Consequently, this technique minimizes the overall deadline missing rate. When all possible time slots are fully utilized, joined requests will result in a higher deadline missing rate because neither the first request nor the second request can be satisfied (see Fig. 12).

We now elaborate on the total scheduling time of using the FS, SSS, and OFS algorithms. We again consider the case of scheduling two requests with different arrival times relative to the first request. Suppose we have n SIUs in the first requested multimedia document and m SIUs in the second request. Let μ_{FA} and μ_{RR} be the average scheduling times of one SIU using FS and RRS, respectively. The total processing time of the first request using FS is

$$\mu_{FA} \times n. \tag{1}$$

The total processing time of the first request using SSS is

$$\mu_{FA} \times (n-p) + \mu_{RR} \times p. \tag{2}$$

Similarly, the total scheduling time of the first request using OFS is

$$\mu_{FA} \times (n-p), \tag{3}$$

where p is the number of SIUs which are yet to be scheduled.

The scheduling time of OFS and FS are the same because the objective of the former is to take the advantages of merging requests using the latter. The scheduler still needs the same amount of time to decide which SIU should go to which channel. However, the scheduler does not gain any advantage over its processing speed. The relationship between μ_{FA} and μ_{RR} should be held as

$$\mu_{FA} > \mu_{RR}. \tag{4}$$

In other words, $\mu_{RR}/\mu_{FA} < 1$.

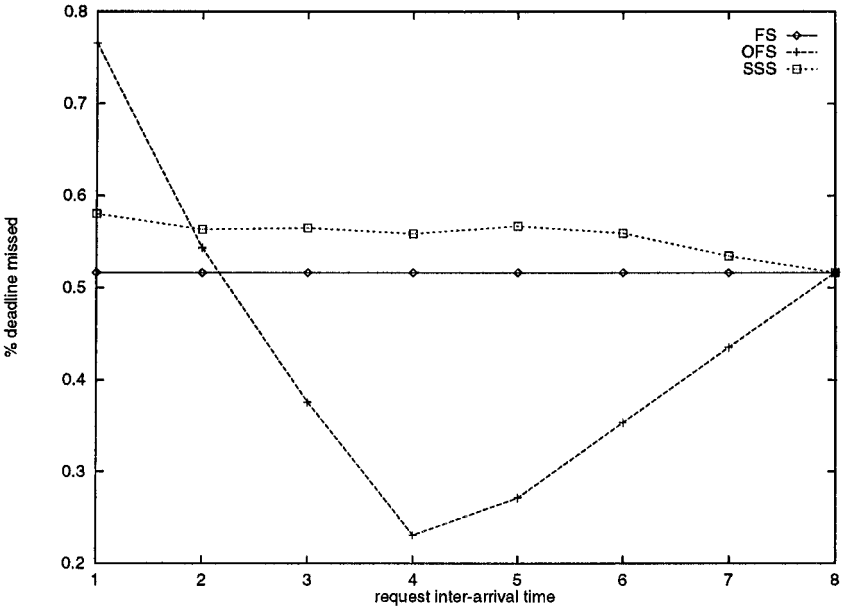


FIG. 12. Deadline missing rate in static mode.

Dividing Eq. (2) by Eq. (1), we get

$$\frac{\mu_{FA}\times(n-p)+\mu_{RR}\times p}{\mu_{FA}\times n}=\frac{(n-p)}{n}+\left(\frac{\mu_{RR}}{\mu_{FA}}\right)\times\frac{p}{n}$$

$$=1-\left(1-\frac{\mu_{RR}}{\mu_{FA}}\right)\times\frac{p}{n}$$

$$=1-\left(1-\frac{\mu_{RR}}{\mu_{FA}}\right)\times(\%\text{ of unscheduled SIUs}),\qquad(5)$$

where $(1-\mu_{RR}/\mu_{FA})>0$.

TABLE 7

Total Scheduling Time

Algorithm	Total scheduling time
FS	$(\mu_{FA}\times n)+(\mu_{FA}\times m)$
SSS	$(\mu_{FA}\times(n-p)+\mu_{RR}\times p)+(\mu_{FA}\times m)$
OFS	$\mu_{FA}\times(n-p)+\mu_{FA}\times(m+p)$

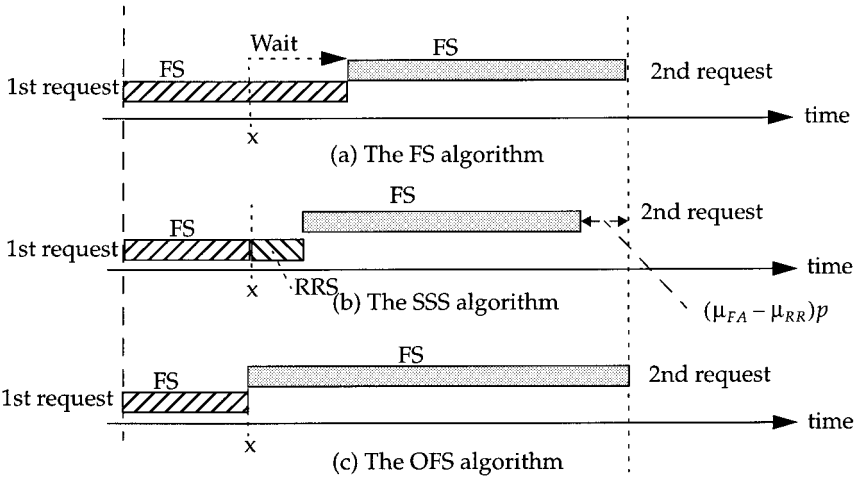


FIG. 13. An illustration of merging of requests by the OFS algorithm.

In other words, SSS runs faster than both FS and OFS in the first round by $(1 - \mu_{RR}/\mu_{FA}) \times (\%$ of unscheduled SIUs) *percentage* of a time unit. In the second round of the scheduling, all three algorithms perform identically. The total scheduling time for the second request using either FS or SSS is

$$\mu_{FA} \times m. \tag{6}$$

The total scheduling time for the second request using OFS is

$$\mu_{FA} \times (m + p). \tag{7}$$

Table 7 includes expressions for the total scheduling time of the two requests. The relationship can be also be pictorially explained using Fig. 13.

7. CONCLUSIONS

We have modified and evaluated two previously proposed static scheduling algorithms in a dynamic context and have proposed two new dynamic scheduling algorithms for server-based multimedia scheduling. We have studied the performances and made comparisons using a detailed multimedia system simulator. The two proposed algorithms outperformed the two previous counterparts in many aspects, such as the deadline missing rate, response time, and scheduler load. Both algorithms are suitable in different environments. In general, because of its simplicity and the faster response, SSS is the more suitable in an extremely high request rate environment if relaxation in QoS requirements is acceptable. In the case of high QoS user requirements, the OFS algorithm is a better choice.

REFERENCES

1. S. R. Ahuja and R. Ensor, Coordination and Control of Multimedia Conferencing, *IEEE Commun. Mag.* **30** (May 1992), 38–43.
2. D. Anderson, S. Tzou, R. Wahbe, R. Govinda, and H. Andres, Support for continuous media in the dash system, *in* "Proceedings of 10th International Conference on Distributed Computing Systems, May 1990," pp. 54–61.
3. S. Baqai, M. Farrukh Khan, M. Woo, S. Shinkai, A. A. Khokhar, and A. Chafoor, Quality-based evaluation of multimedia synchronization protocols for distributed multimedia information systems, *IEEE J. Selected Areas Commun.* **14**, 7 (Sep. 1996), 1388–1403.
4. J. Blazewicz, K. Ecker, G. Schmidt, and J. Weglarz, "Scheduling in Computer and Manufacturing Systems," Springer-Verlag, Berlin/Heidelberg, 1993.
5. W. Clark, Multipoint multimedia conferencing, *IEEE Commun. Mag.* **30** (May 1992), 44–50.
6. Y. F. Day, S. Dagtas, Mitsutoshi Iino, A. Khokhar, and A. Ghafoor, Spatio-temporal modeling of video data for on-line object-oriented query processing, *in* "Proceedings of the International Conference on Multimedia Computing and Systems, 1995," pp. 98–105.
7. Y. Deng and S. Chang, A framework for the modeling and prototyping of distributed information systems, *Internat. J. Software Engineering and Knowledge Engineering* **1** (1991), 203–226.
8. D. Deloddere, W. Verbiest, and H. Verhille, Interactive video on demand, *IEEE Commun. Mag.* **32** (May 1994), 82–89.
9. M. L. Escobar-Moiano, S. Gandeharizadeh, and D. Lerardi, An optimal resource scheduler for continuous display of structured video objects, *IEEE Trans. on Knowledge and Data Engineering* **8**, 3 (June 1996), 508–511.
10. G. S. Fishman, "Principles of Discrete Event Simulation," Wiley, 1978.
11. I. E. Greif, "Computer Supported Cooperative Work: A Book of Readings," Morgan Kaufmann, Sam Mateo, CA, 1988.
12. M. Haindl, A new multimedia synchronization model, *IEEE J. Select. Areas Commun.* **14** (Jan. 1996), 73–83.
13. N. G. Hall and M. E. Posner, Earliness-tardiness scheduling problems—I. Weighted deviation of completion times about a common due date, *Operations Research* **39**, 5 (Sept.-Oct. 1991), 836–46.
14. J. R. Haritas, M. Livny, and M. J. Carey, Earliest deadline scheduling for real-time database systems, *in* "Proceedings. Twelfth Real-time Systems Symposium," pp. 232–242.
15. J. Huang and D. Z. Du, "Resource Management for Continuous Multimedia Database Application," pp. 46–54, IEEE, New York, 1994.
16. R. Gopalakrishnan and G. M. Parulkar, Bring real-time scheduling theory and practice closer for multimedia computing, *ACM Performance Evaluation Rev.* **24**, 1 (May 1996), 1–12.
17. S. D. Liman and S. Ramaswamy, Earliness-tardiness scheduling problems with a common delivery window, *Operations Res. Lett.* **15**, 4 (May 1994), 195–203.
18. T. D. C. Little and A. Ghafoor, Synchronization and storage models for multimedia objects, *IEEE J. on Select. Areas in Comm.* **8** (Apr. 1990), 413–427.
19. T. D. C. Little and A. Ghafoor, Network considerations for distributed multimedia object composition and communication, *IEEE Network Mag.* (Nov. 1990), 32–49.
20. Y. Masunaga, An object-oriented multimedia database management system, *J. Inform. Process.* **14** (1991), 60–74.
21. S. S. Park and D. H. Cho, Performance improvements of scheduling algorithms for multimedia server, *IEICE Trans. INF & SYST.* **E790D**, 6 (June 1996), 706–711.
22. K. K. Ramakrishnam, 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* **v3** (1995), 53–65.

23. B. Sonah, M. R. Ito, and G. Neufeld, The design and performance of a multimedia server for high-speed networks, *IEEE Comput*. (1995), 15–22.
24. R. Steinmeta, Synchronization properties in multimedia systems, *IEEE J. Select. Areas in Commun.* **8** (April 1990), 401–412.
25. R. Steinmetz, Analyzing the multimedia operating system, *IEEE Multimedia* (Spring 1995), 68–84.
26. P. Stotts and R. Furuta, Petri-net-based hypertext: Document structure with browsing semantics, *ACM Trans. Office Automation Systems* **7** (Jan. 1989), 3–29.
27. F. Tompa, A data model for flexible hypertext database system, *Inform. Services and Use* **7** (Jan. 1989), 85–100.
28. Q. Wang, D. R. Broome, and A. R. Greig, Intelligent gain scheduling (IGS) using neural networks for robotic manipulators, *in* "Proceedings of the Workshop on Neural Network Applications and Tools, 1994," pp. 103–8.
29. M. Woo, A synchronization framework for networked multimedia services, Ph.D. thesis, Purdue University, Dec. 1995.
30. M. Woo, N. U. Qazi, and A. Ghafoor, A synchronization framework for communication of preorchestrated multimedia information, *IEEE Network* (Jan./Feb. 1994), 52–61.
31. G. G. Xie and S. S. Lam, Delay guarantee of virtual clock server, *IEEE/ACM Trans. On Networking* **3** (6) (Dec. 1995), 683–689.
32. K. Y. Yau and S. S. Lam, Adaptive rate-controlled scheduling for multimedia applications, *ACM Multimedia 96*, 129–140.
33. L. Zhang, Virtual clock: A new traffic control algorithm for packet switching networks, *ACM Transactions on Comput. Syst.* **9**, 2 (May 1991), 101–124.

ISHFAQ AHMAD received a B.Sc. degree in electrical engineering from the University of Engineering and Technology, Lahore, Pakistan, in 1985. He received his M.S. degree in computer engineering and his Ph.D. degree in computer science, both from Syracuse University in 1987 and 1992, respectively. Currently, he is an associate professor in the Department of Computer Science at the Hong Kong University of Science and Technology, where he is also the Director of Video Technology Center. His research interests are in the areas of parallel programming tools, scheduling and mapping algorithms for scalable architectures, video technology, and interactive multimedia systems. He has published over 80 papers in the above areas. He has served on the committees of various international conferences, has been a guest editor for two special issues of *Concurrency Practice and Experience* related to resource management, and is co-guest-editing a forthcoming special issue of the *Journal of Parallel and Distributed Computing* on the topic of software support for distributed computing. He also serves on the editorial board of Cluster Computing. He is a member of the IEEE Computer Society.

WILLIAM LAI received a B.Sc. degree in computer science from Hong Kong Polytechnic University in 1995 and an M.Phil. degree in computer science from Hong Kong University of Science and Technology, Hong Kong in 1997. His research interests include parallel and distributed systems, networking, and multimedia systems. He is at present a network engineer with LinkAGE Online Ltd.

BO LI received the B.S. (cum laude) and M.S. degrees in computer science from Tsinghua University, Beijing, in 1987 and 1989, respectively, and the Ph.D. degree in computer engineering from the University of Massachusetts at Amherst in 1993. Between 1994 and 1996, he worked on high performance routers and ATM switches in IBM Networking System Division, Research Triangle Park, North Carolina. He joined the faculty in the Computer Science Department of Hong Kong University of Science and Technology in January 1996. He is on the editorial board for *ACM Mobile Computing and Communications Review* (MC2R) and *Journal of Communications and Networks* (JCN). He is currently serving as a Co-TPC Chair for the First International Conference on Mobile Data Access to be

held in Hong Kong in December 1999. He has served on the Technical Program Committees (TPC) for a number of IEEE conferences and workshops such as Infocom, ICDCS, and ICC. His current research interests are: wireless mobile networking supporting multimedia, all optical networks using WDM, web traffic analysis, active networking, and performance evaluation. He can be reached at BLI@cs.ust.hk.

Printed in Belgium