Chapter 1

Scheduling multimedia documents in a distributed system[1]

Ishfaq Ahmad, Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong, iahmad@cs.ust.hk

William Y. M. Lai, Bo Li and Xin Deng, Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong

Abstract

In a client-server based distributed multimedia system, requests for multimedia documents arrive sporadically. These requests must be served by delivering the requested multimedia documents with a fast response time and ensure a certain quality of service. This requires the server to determine the transmission schedule of each multimedia stream while ensuring necessary inter - and intra-stream synchronizations. The multimedia server has to generate schedules in real-time for multiple requests. Therefore, the scheduling algorithm must incur small execution cost. In this paper, we propose a number of dynamic heuristic algorithms. The main feature of our algorithms is that they include an embedded dynamic nature which can adjust their scheduling times for each request. If the slack time between two requests is large, the scheduler runs longer in an attempt to find a better solution. This reduces the response time while maintaining a good quality of presentation. Using simulation and analytical results, we evaluate our algorithms and demonstrate their potential to be applicable in real distributed system.

Keywords: distributed systems, databases, knowledge bases, multimedia

1 Introduction

In a distributed multimedia system, with a single server / multiple clients configuration, 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 such 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. Server -based scheduling has been studied previously (see [2], [6]); however, investigation was bound to a static and single client scheduling environment [2] what happened to [6]. 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. We propose two dynamic scheduling algorithms in this paper to fulfill this purpose.

The remainder of this paper is organized in the following manner. In the next section, we provide an overview of the multimedia document model used in our study. In Section 3, we expound the multimedia scheduling problem, include a brief overview of existing multimedia scheduling algorithms. In Section 4, we describe the proposed algorithms. In Section 5, we provide some details of our 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 is required. To this end, various techniques have been extensively studied by researchers [4], [8], [14] to develop conceptual models for representing multimedia objects. These models can be classified into four categories: graphical models [12], Petri net based models [11], object-oriented

[1] This research was supported by the Hong Kong Research Grants Council under contract number HKUST 734/96E.

models [9], and temporal abstraction models [1]. Some models are primarily aimed at synchronization aspects of multimedia data, while the others are more concerned with the objects browsing aspects.

The Petri net has been shown to be a useful modeling technique [5], [11], [8], which can help developers to define specific temporal relationships among objects. A Petri net based model is used to specify object level synchronization requirements. Making use of this specific character, a model known as the object composition Petri net (OCPN) was proposed [8]. 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.

For the purpose of inter -stream synchronization, temporally related objects may be divided into SIUs of equal duration [13]. 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 re-synchronize 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. Although the duration of a SIU for each of the two streams is the same, the size of a 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 bounds, and delay jitter for objects. From the QoS parameters, playout deadlines for individual SIUs can be derived from the object deadlines. Let us assume:

Δ_i is the synchronization interval of object O_i
d^i is the playout deadline of object O_i
d_j^i is the deadline of j-th SIU of object O_i
then, the playout deadlines is $d_j^i = d^i + (j - 1)\ \Delta_i$

For uncompressed data, the size of a SIU can be determined from the overall size of the object, its duration, and the length of the synchronization interval. For compressed data, we assume that information on the sizes of the SIUs is also stored when data is compressed.

Mathematically , a formal specification of XOCPN can be found in [14]. The XOCPN is divided into two categories: the transmitter and the receiver model. When retrieving a multimedia document both models are needed to be generated. The transmitter -XOCPN is used by the multimedia server . According to the transmitter -XOCPN model, the server knows when it should schedule a SIU into a channel. 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 because of the nature of the Petri net firing rule. Inter -stream pacing points provide the capability for much finer grained synchronization.

3 Scheduling

The problem of scheduling the transmission of multimedia data in a channel-deficient system at the server site can be formulated as a deterministic, non-preemptive parallel machine scheduling problem. It has been identified as a non-deterministic problem, equivalent to the knapsack NP-Hard problem [3]. We assume the network adopts static resource reservation and provides multiple virtual channels for each client with guaranteed QoS, specifically bandwidth and bound on the 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 pre-scheduling time is referred to as the control time . Since the delays are random, some SIUs may arrive earlier than their deadlines and will need to be buffered.

The problem of scheduling multimedia documents can be formulated as a deterministic, 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.

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 within the required bounded delay. SIUs with the same playout deadline may have different end-to-end delays even when separate channels are available for them. Consequently, deriving the transmitter -XOCPN should take delays in the network 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 $[\delta_i, d_i]$, 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 in the document
c_i is the i-th channel's effective bandwidth
δ_i is the bound on end-to-end transit delay of SIU_i stored with data itself
β_i is the transmission start time of SIU_i with respect to d_i
δ_i^t is the transmission delay of SIU_i then $\delta_i^t = s_i / c_i$

To meet its playout deadline at the client site, the server should transmit SIU_i no later than $(d_i - \delta_i^t - \delta_i)$ which means we can set $\beta_i = d_i - \delta_i^t - \delta_i$. At the server side, SIU_i occupies the outgoing channel for $[\beta_i, \beta_i + \delta_i^t)$ 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.

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_j at time α_i according to some scheduling policy. Then the arrival time, A_i, of SIU_i at the client site becomes

$$A_i = \alpha_j + s_i/c_j + \delta_j$$

It should be noted that the transit delay δ_j of channel Cj may not be the same as the required transit delay δ_i^t of SIUi. The tardiness of SIUi with respect to its playout deadline is 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. To ensure synchronization and to avoid missing playout deadlines of SIU_s, 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 $T_{max} = \max_{1 \le i \le n}\{T_i\}$.

However, sometimes a 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 $1 \le i \le n$, then the induced playout deadlines are the earliest playout deadlines that can be met while achieving a synchronous presentation according to the 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.

There exists a number of scheduling algorithms in literature. Earliest deadline first (EDF) is an algorithm which is believed to be suitable for scheduling continuous media. EDF is an optimal, dynamic algorithm [10]. Five efficient heuristic algorithms have been proposed in [2] to obtain approximate solutions to the scheduling problem. A modified earliest deadline first (MEDF) algorithm has been proposed [7]. 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.

4 The Proposed Algorithms for Scheduling

In this section, we first describe two previously proposed static scheduling algorithms called the forward scheduling (FS) algorithm, the backward scheduling (BS) algorithm[13]. We propose changes tailored for dynamic environment. We next propose two new dynamic algorithms. For comparison, we also propose 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. Some of the notations used in the subsequent discussion are provided in Table 1.

Table 1: Notation representation

Notation	Representation
α_i	Transmission time on channel j for SIUi
A_i	Arrival time of SIUi at the client site
C_i	Effective bandwidth rate associated with channel Ci
C	$=\{C_1, C_2, \ldots C_m\}$
d_i	Playout deadline of SIUi
δ_j	End-to-end transit delay of SIUi
L_j	Schedule on channel j
m	Number of channels
n	Number of SIUs to be scheduled
$\pi^j_{(i)}$	Channel occupation duration of SIUi
S_i	Size of SIUi
S_i/C_i	Transmission time or delay
S	$=\{SIU_1, SIU_2, \ldots, SIU_n\}$
$Tx^j_{(i)}$	Transmission start time of SIUi on channel Cu

The FS algorithm [14] uses the earliest due date (EDD) rule by sorting SIUs in a non- decreasing 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 to transmit it the earliest.

FS Algorithm :

1. Sort SIU_s in S using EDD rule with LPT rule for the tie breaking
2. Initialize $L_j=\{\}$ and α_j= channel$_j$'s earliest available time (i.e., Idle) 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,
   $L_j = L_j \cup \{SIU_i\}$ if $j = \text{argmin}\{\alpha_k + s(i)/c(k) + \delta_k\}$ for $1 \le k \le m$
   Set $Tx^j_{(i)} = \alpha_j$ and $\pi^j_{(i)} = s(i)/c(j)$

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 $1 \le i \le n$

The backward scheduling (BS) algorithm [14] uses the EDD rule by sorting SIUs in a non- decreasing 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 SIU_s in S using EDD rule with SPT rule for the tie breaking
2. Initialize $L_j=\{\}$ and
   $\alpha_j = \max\{d1,d2,\ldots,dn\}$ + channel$_j$'s earliest available time (i.e., Idle)
3. Starting from the end of the list, schedule each SIU in S on
   the channel that minimizes its start time, that is,
   $L_j = \{SIU_i\} \cup L_j$ if $j = \text{argmax}\{\min\{d_i, \alpha_k\} - s(i)/c(k) - \delta_k\}$ for $1 \le k \le m$
   $A_i = \min\{d_i, \alpha_j\}$
   Set $\pi^j_{(i)} = s(i)/c(j)$
   Set $Tx^j_{(i)} = A_i - \pi^j_{(i)} - \delta_k$
   Update α_j to $\alpha_j = A_i - s(i)/c(j)$
4. T_{max} is then $\max\{\min\{\alpha_1, \alpha_2, \ldots, \alpha_m\} + A_i - d_i\}$
5. For all SIU_s in the m schedule L_j's,
   update the transmission start time as $Tx^j_{(i)} = Tx^j_{(i)} + T_{max}$

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 SIU_s in S using EDD rule
2. Initialize $L_j=\{\}$ and α_j = channel$_j$'s earliest available time (i.e. Idle)
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
   $L_j = L_j \cup \{SIU_i\}$
   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(nlogn + nm). On the other hand, the RRS algorithm is only a simple scheduling algorithm. The complexity for scheduling SIUs is $O(n \log n + n)$.

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 to 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 SIU_s 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 SIU_s using RRS
7. Serve the next request from starting from Step 1.

The FS algorithm produces an optimal solution in a timely manner. One of its drawback 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. For example, 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 SIU which could be a combination of previous and new requests, and continues to schedule SIUs as long as the pool is not empty.

OFS Algorithm:

1. Sort SIU_s in S using EDD rule with the LPT rule for the tie breaking
2. Initialize L_j and α_j = 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,
   $L_j = L_j \cup \{SIU_i\}$ if $j = \text{argmin}\{\alpha_k + s(i)/c(k) + \delta_k\}$ $1 \le k \le m$
   Set $Tx^j_{(1)} = \alpha_j$ and $\pi^j_{(1)} = s(i)/c(j)$
   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 SIU_s) with the new coming SIU_s
7. Go to the first step and continue the scheduling until finished
8. T_{max} is then $\max\{0, A_i - d_i\}$ for $1 \le i \le n$

5 The Simulator

We have developed a simulator to test 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 signaling for low connection setup latency . The communication channels between a client and the server are setup 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 A TM-based network which can support multiple virtual channels with various QoS.

The pre-created multimedia document structures are based upon XOCPN. Anisochronous data (for example, text and images) needs to be available to the play-out devices at the destination prior to their play-out deadlines. In contrast, isochronous objects (for example, video and audio) can be decomposed into smaller presentation units (SIUs) which are meaningful to the play-out device. Therefore, anisochronous data occupies exactly one SIU while isochronous data occupies one or more SIUs depending on the play-out duration of that object. Each client gets its own set of virtual channel 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 is designed to run long enough to give stable results and to eliminate the bias of the initialization stage.

6 Experiments and Results

In our experiments we used multimedia documents that include video-clips, audio-clips, and images. 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 transform it to the transmitter - XOCPN model. Each object has associated with it 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 tinit 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 d1 + d2 where d1 represents an artificial delay that is introduced in order to account for the time gap between two successive SIUs in transmission; and d2 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.

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(Pv,Pa ,Pi ,Pt) where Pv, Pa, Pi, and Pt indicate the probabilities for video, audio, image, and textual data respectively (see the following table).

Table 2: Distribution of multimedia objects

Set number\Probabilites	Pv	Pa	Pi	Pt
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

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 kilo bits to 285 kilo bits. An audio object is assumed to be stored without any post-processing. 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 kilo bits to 15 kilo bits. Figure1 shows an example of an OCPN generated in our simulation.

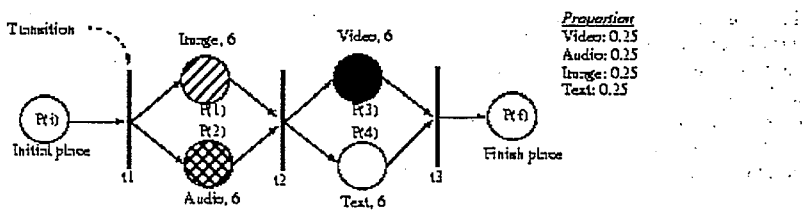

Figure 1: An example of a randomly generated OCPN model.

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.

$$\rho \leq 1, \mu \geq \lambda, \ \lambda/\mu = \rho$$

In other words: 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 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 executed each scheduling algorithm without the simulator for all of our test data and measured the actual execution times of all algorithms. Table 4 summarizes the results in terms of the scheduling time per SIU.

Table 3: Scheduling algorithms benchmarks (second per SUI)

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

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 proportion of the SIUs being scheduled by the RRS, then the time for scheduling n SIUs using SSS is given by

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

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. Theses experiments are done is for mixed media services, such as computer based training (CBT) with different combinations of objects from of all streams.

For 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.

Table 4: Experimental design for simulation

	Values used	# 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 (Pv, Pa, Pi, Pt)	(0.25,0.25,0.25,0.25) (0.4,0.4,0.1,0.1) (0.1,0.1,0.4,0.4)	3
Distribution of the transit delay	U[0,0.0.05]	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

Figures 2 and 3 depict the results of the mixed media using different algorithms. For the results shown in Figure 2, 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 deadline missing rate. In terms of response time, OFS outperforms both FS and BS. On the 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 Figure 2(b)) 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 Figure 2(c)). 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 Figure 3. 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 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 over head imposed by BS (more SIUs cannot meet their deadlines in time after the start of the presentation). However , BS performs better after a certain critical point. The best performance is still yielded by OFS.

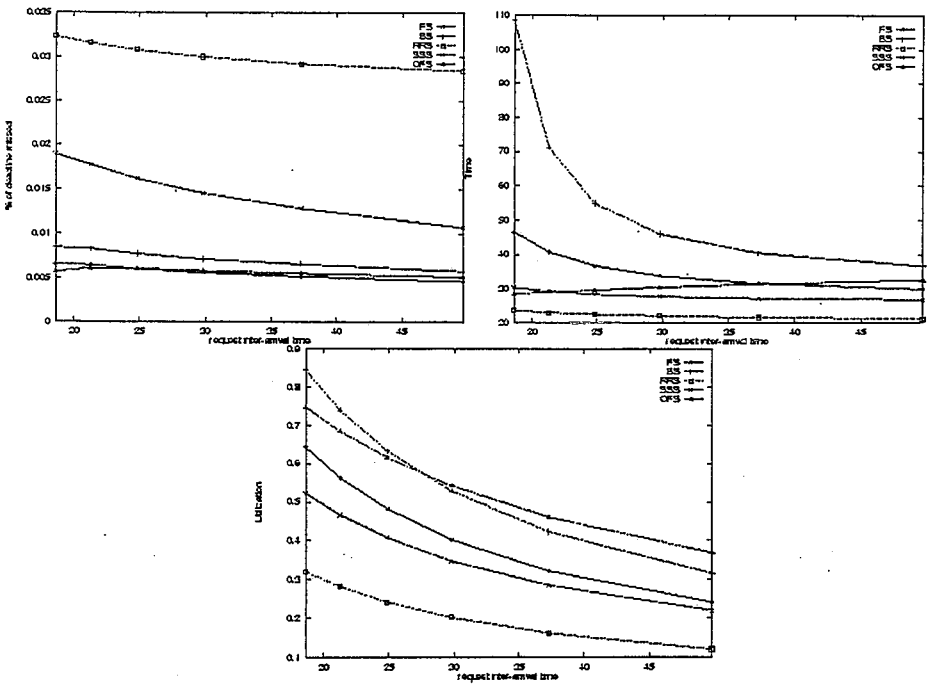

Figure 2: Comparison of algorithms(mixed media)
(a)Deadline missing rate (b)Client response time (c)Scheduler utilization

7 Conclusions

The newly two proposed algorithms outperformed 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.

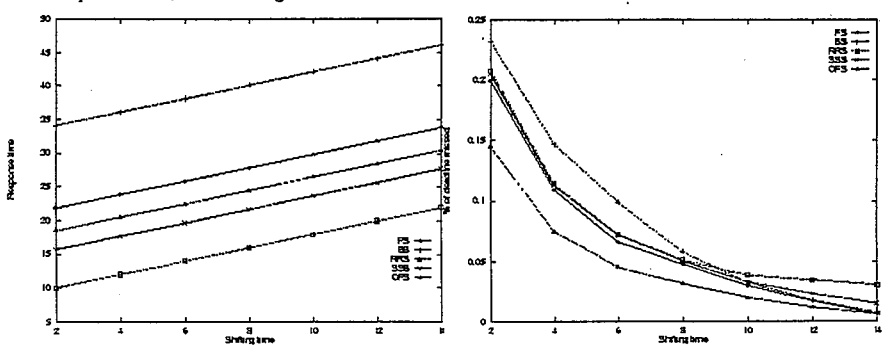

Figure 3: Deadline missing probability and response time vs shifting time
(a)Response time vs shifting time (b)Deadline missing vs shifting time

References

[1] 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 , pp. 54-61, May 1990.

[2] S. Baqai, M. Farrukh Khan, M. Woo, S. Shinkai, A. A. Khokhar, and A. Ghafoor, "Quality-Based Evaluation of Multimedia Synchronization Protocols for distributed Multimedia Information Systems," IEEE Journal on selected areas in communications , vol. 14, no.7, pp. 1388-1403, Sep96.

[3] J. Blazewicz, K. Ecker, G. Schmidt, and J. Weglarz, "Scheduling in Computer and Manufacturing Systems," Berlin Heidelberg : Springer-Verlag, 1993.

[4] Y. F. Day, S. Dagtas, Mitsutoshi Iino, A. Khokhar, and A. Ghafoor, "Spatio-Temporal Modeling of Video Data foron-Line Object-Oriented Query Processing," Proceedings of the International Conference on Multimedia Computing and Systems (Cat. No.95TH8066) (1995) p98-105.

[5] Y. Deng and S. Chang, "A Framework for the Modeling and Prototyping of Distributed Information Systems," International Journal of Software Engineering and Knowledge Engineering , vol. 1, pp. 203-226, 1991.

[6] M.L. Escobar-Moiano, S. Gandeharizadeh, and D. Lerardi, "An Optimal Resource Scheduler for Continuous Display of Structured Video Objects," IEEE Transactions on Knowledge and Data Engineering , vol. 8, no. 3, pp. 508-511, June 1996.

[7] S.D. Liman, S. Ramaswamy, "Earliness-tardiness Scheduling Problems with a Common Delivery Window," Operations Research Letters vol. 15, no. 4 (May 1994) p195-203.

[8] T.D.C. Little and A. Ghafoor, "Synchronization and Storage Models for Multimedia Objects," IEEE J. on Select. Areas in Comm. , vol. 8, pp. 413-427, Apr. 1990.

[9] Y. Masunaga, "An Object-Oriented Multimedia Database Management System," Journal of Information Processing , vol. 14, pp. 60-74, 1991

[10] R. Steinmetz, "Analyzing the Multimedia Operating System," IEEE Multimedia , pp.68-84, Spring 1995.

[11] P. Stotts and R. Furuta, "Petri-Net-Based Hypertext: Document Structure with Browsing Semantics," ACM Transactions on Office Automation Systems , vol. 7, pp. 3-29, January 1989.

[12] F. Tompa, "A Data Model for Flexible Hypertext Database System," Information Services and Use , vol. 7, pp. 85-100, January 1989.

[13] M. Woo, "A Synchronization Framework for Networked Multimedia Services," Ph.D Thesis, Purdue University, Dec. 1995.

[14] M. Woo, N.U. Qazi, and A. Ghafoor, "A Synchronization Framework for Communication of Pre-orchestrated Multimedia Information," IEEE Network , pp. 52-61, January/February 1994.