Real-Time Scheduling for Multimedia Services[1]

T.D.C Little and J.F. Gibbon

Multimedia Communications Laboratory
Department of Electrical, Computer and Systems Engineering
Boston University, Boston, Massachusetts 02215, USA
(617) 353-9877, (617) 353-6440 fax
tdcl@bu.edu, jfg@bu.edu

MCL Technical Report 09-02-1992

Abstract–High-speed networks and high-performance workstations are necessary but not sufficient to support distributed multimedia applications. A real-time scheduling system designed for multimedia data types is also required to orchestrate communications channels, disk storage units, output devices, and the CPU. These subsystems are coordinated to accommodate the special requirements of multimedia data: timely retrieval, transmission, and delivery with permissible levels of data loss and corruption.

In this paper we present our framework for the use of real-time scheduling disciplines to support time-dependent multimedia data in a distributed-data environment. Within this framework we propose the application of a statistical resource reservation mechanism and a real-time *session scheduler*. Furthermore, we relate scheduling and quality of service in a summary of the objectives of multimedia service provision and negotiation.

Keywords: Real-time scheduling, multimedia data delivery.

[1]In *Proc. SPIE Symposium OE/FIBERS '92, (High-Speed Networks and Channels II)*, Boston, Massachusetts, September 1992, SPIE Vol. 1784, pp. 320-331.

1 Introduction

Multimedia data such as audio and video require special considerations when supported by a computer system. These data have well-defined presentation timing constraints that must be satisfied by the system during presentation (or *playout*). Other data types (e.g., text, graphics) do not have implied timing like audio and video, but can also be assigned playout timing. To satisfy these timing requirements we draw upon the field of real-time systems, but more specifically, real-time scheduling.

The job of a real-time *scheduler* is to manage the assignment of execution resources to tasks awaiting execution within the timing constraints assigned to each task. The real-time scheduler must allocate resources to tasks in a predetermined manner based on some system of priority to ensure timing satisfaction. Allocated resources can be the CPU, storage systems, the communication channel, or other system devices. In this paper, we describe *bandwidth* allocation, and we will use it interchangeably for reference to any of a system's resources. For example, a real-time scheduler might reserve 50 % of the CPU bandwidth for executing a task prior to its completion deadline.

The playout timing for a complex multimedia object can be defined as a set of temporal relationships or as a playout schedule. Once this schedule is defined, a real-time scheduler must orchestrate the various resources for the desired presentation. However, the resource requirements can change at any time during the presentation because the presentation itself can change. A multimedia business presentation can be stopped, fast-forwarded, or appended at any time by the presenter through *temporal access control* operations. An educational system must produce the correct presentation in response to the student's last input. The scheduler must also account for system load changes in the data delivery path including the network and the CPU.

A multimedia scheduler can be more effective if system functions, such as copying data from one area of memory to another, have well defined temporal semantics with respect to a real-time operating system. However, a real-time operating system is not a substitute for a real-time multimedia scheduler. Even the priority system implemented in POSIX[9] does not adequately resolve the specific resource requirements present in a multimedia system. These requirements are considered when we discuss *quality of service* (QOS) in Section 4.

Along with dynamic inputs a real-time multimedia scheduler must also be able to contend with limited resource availability when establishing sessions. For example, in a diagnostic

imaging scenario, many physicians may simultaneously call up a patient's written record, X-rays, and a surgical video. This could overwhelm the available resources. A desirable solution to this problem is to provide session establishment and playout when less than nominal resources are available. The result is a degraded service to each session. This approach is called *graceful degradation* by the management of quality of service.

There is much recent literature covering real-time scheduling (e.g., Cheng et al.[1]). While providing a foundation for our proposed system, sparse research in real-time scheduling addresses the problems of varied performance characteristics needed to support multimedia delivery. We consider these characteristics in our discussion of QOS. Resource commitments have been integrated in systems such as real-time networks. In a multimedia system the resource commitments orchestrated by our resource reservation system are unique because they are for a specific period thus allowing for dynamic user input and system changes.

Data encoding formats have an important impact on real-time service for multimedia data. For example the MPEG-System proposal addresses real-time audio and video synchronization. This synchronization requires that both the audio and video come from the same source and are in MPEG format[7]. This method of synchronization cannot be applied to a multimedia data originating from multiple sources or to other encoding formats such as JPEG[8] compression. Our resource reservation system is designed specifically to adjust to system changes which cause skew in the playout of two synchronized streams[12]. It is also implemented considering distributed sources and is general enough to allow for various presentation formats.

The remainder of the paper is organized as follows. In Section 2 we present an overview of real-time scheduling theory and how it can be applied to multimedia data delivery. In Section 3 we describe our proposed limited *a priori* reservation system and associated session scheduler. Section 4 presents a framework for quantification of quality of service. Finally, Section 5 concludes the paper.

2 An Overview of Real-Time Scheduling

The use of real-time scheduling is essential in a multimedia system. We examine real-time scheduling theory by first describing some basic definitions and algorithms and their application to multimedia. We then consider real-time communications.

2.1 Basic Definitions and Algorithms

A real-time system requires either *hard* or *soft* scheduling. Hard real-time scheduling involves tasks with absolute deadlines; the system cannot recover if a task is not completed by an appointed time called its *deadline*. Soft real-time scheduling involves tasks which have deadlines that are not absolute but can only be missed a certain percentage of time.

We consider multimedia scheduling to be soft real-time scheduling. However, we can characterize the tolerance to missed deadlines and other errors in contrast with other soft real-time systems. For example, in most presentations if a graphic 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 missed playout deadline depends on the media and the presentation.

Another distinction of real-time systems is how the scheduling of tasks is performed, either *statically* or *dynamically*. In a static real-time system the information for all tasks that need to be scheduled is known *a priori*. A dynamic system must schedule tasks as they are requested without prior knowledge of their existence. The paradigm illustrated in this paper is of a multimedia system in which a session is requested and the session is divided into periods. Each period is scheduled statically just before its presentation. This allows our proposed limited *a priori* reservation system (Section 3) to accommodate recent user input or system load changes when creating the schedule for a period. For example, in an interactive educational application, different information is scheduled and presented based on a student's input. Similarly, rescheduling is required when additional students participate and new interactive sessions are added.

There are various types of both static and dynamic scheduling algorithms[1]. Static scheduling is often attempted with the straightforward *earliest-deadline-first* algorithm. This is useful when a small percentage of total available resources is used. In a multimedia system a simple slide show might be such an application. Another algorithm, the *rate-monotonic* algorithm, is used specifically for periodic tasks. This corresponds to the scheduling of tasks with the shortest period first.

Dynamic scheduling is a more difficult kind of scheduling because the arrival of tasks is not known *a priori*. For dynamic scheduling algorithms the run-time costs of the scheduler are important because they are a factor in meeting all the tasks deadlines. Some schedulers have an earliest-deadline first or earliest-deadline–earliest-*ready-time*-first policy. Ready time

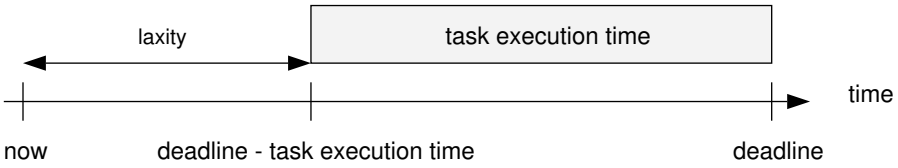

Figure 1: Laxity

is the earliest a task can be scheduled. An earliest-deadline–*least-laxity* scheduler is another possibility. *Laxity* is defined as the difference in time between the current time, and the latest time to start a task's execution and still satisfy its deadline (Fig. 1).

For multimedia data, an earliest-deadline–least-laxity discipline is an appropriate choice. An earliest-deadline-first scheduler processes tasks with the earliest deadline, even if they cannot possibly be complete before their deadlines. A least-laxity scheduler does not schedule tasks that cannot be completed on time. For continuous multimedia data such as video, this implies discarding frames that cannot be presented on time due to lateness in retrieval, transmission, decompression, etc. Fig. 2 (a) shows an earliest-deadline schedule in which each frame is late because the retrieval and decompression process for each frame is scheduled too late. With the earliest deadline discipline, each frame is scheduled regardless of the fact that they cannot meet their deadlines. The least laxity scheduler in Fig. 2 (b) takes into account the ability of a task to be accomplished by its deadline. It executes the retrieval and decompression of frame 7 after frame 5 because frame 6 has no chance of meeting its deadline.

2.2 Real-Time Communications

Real-time communications represent a special application of real-time systems theory. A real-time communications system must manage the communication of time-dependent data to provide timely and predictable data delivery. One way to schedule data transmission is to maintain statistics characterizing each communication channel. When the channel characteristics of the network change, the scheduler can adjust accordingly to maintain predictable service. This can be achieved by decreasing the demand on the network. For example, when a network becomes congested and the percentage of late data elements (missed deadlines) increases, dropping the demand on the network helps clear the congestion[4]. This effectively allows data elements scheduled for transmission to traverse the network and arrive on time rather than be lost due to lateness.

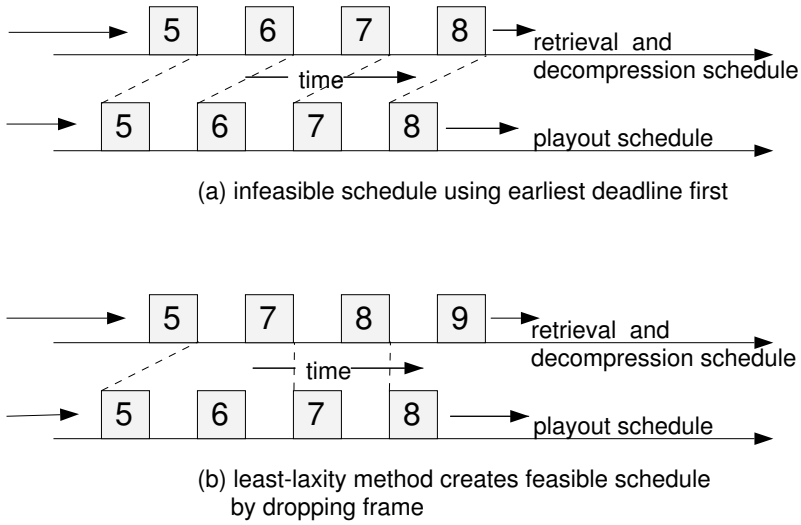

Figure 2: (a) Earliest-deadline versus (b) least-laxity scheduling

A real-time *network* provides performance guarantees for the delivery of data from a source to destination. These guarantees can be absolute though deterministic scheduling and resource allocation, or approximate by using statistical approaches. In either case, changes in network delay and bandwidth characteristics can have adverse effects on their behavior. With a real-time network, an agreement is negotiated between the application and the network controller. The application provides specific communication timing requirements and the network grants a connection assuming the existence of adequate resources. The following are representative of approaches to scheduling for real-time communications.

For asynchronous timesharing (ATS)[6], data traffic is divided into four classes. A control class C has the highest priority and experiences no loss or variable delay. Class I experiences no loss, but the user must specify a maximum end-to-end delay. Class II has a set maximum percent loss and a maximum consecutive loss. Finally, class III has zero loss with no maximum end-to-end delay, but a maximum average time delay and maximum average throughput can be established if requested by the user.

Another real-time network protocol handles performance requirements in a different manner[2,3]. In a connection request the user provides the network manager with maximum end-to-end delay, maximum packet size, maximum packet loss rate, minimum arrival time between packets, and maximum jitter, where jitter is defined as the difference in the delays experienced between two packets on the same connection. The application can request one of three types of channels each having a different expression for delay bound. *Deterministic*

bounds are expressed for channels which have absolute deadlines in the arrival of a packet. For a *statistical* channel, the probability that the delay is shorter than a given time D is greater than a factor Z. The third, *best-effort* channel, provides no guarantee for the percentage of arrived messages within the time D, however, the network manager attempts to meet the deadlines.

Statistical approaches to overcoming delay and bandwidth limitations rely on choosing an end-to-end control time T per packet that is larger than the delay experienced by a percentage of the transmitted packets. This delay represents a fixed latency in a multimedia session and is proportional to the number of data elements needed for buffering at the destination. The end-to-end delay for a fixed-size packet can be decomposed into a propagation delay D_p, a transmission delay D_t, and a variable delay due to queuing in the network D_v.

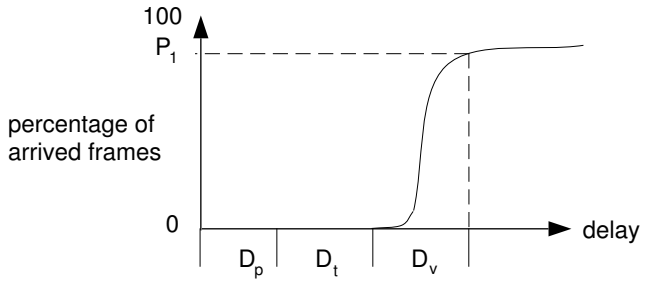

Figure 3: Delay characteristics

Fig. 3 illustrates the selection of a control time T_1 such that $T_1 > D_1$ to be assured that the desired percentage of packets P_1 will arrive on time. While such a delay function can accurately represent the delay characteristic over a given period of time, network loading can change, and a new delay function results. One way to accommodate this is to monitor the channel delay distribution and adjust the retrieval schedule accordingly. The channel can be characterized either by enforced network guarantees, or by traffic monitoring. Important parameters for real-time multimedia data include minimum, maximum, and average end-to-end delay delay; packet size, arrival time between packets, packet jitter, packet loss rate, consecutive packet loss rate. These QOS parameters are suitable for providing a target service objective provided by the network and negotiated by the application.

A statistical channel can be reserved based on the source and channel characteristics by using an exact or approximate source model. The exact model is applicable to data originating from stored sources, whose characteristics can be determined *a priori*. In this case, the transmission requirements of the entire source stream can be mapped to the available

channel bandwidth. For the approximate source model, the source data rate can be characterized by average and maximum values (e.g., live video from a camera). In either case, changes in the channel delay and bandwidth characteristics cannot be easily accommodated without disrupting the session.

In our proposed scheduling approach, we concentrate on exact source models typical of database applications. Changes in channel characteristics are managed by frequent reevaluation of the current loading, as we describe next.

3 Real-Time Scheduler for Multimedia Data

A fault with prior statistical reservation approaches is their inability to adapt to system load changes or the dynamic behavior of an interactive multimedia session. Because statistical scheduling relies on a commitment from the network, changes in loading cannot be tolerated by the application. Our proposal is to use a hybrid statistical resource reservation approach that only performs scheduling based on a relatively short interval over the life of a session. By restricting the scheduling period we benefit in the following ways:

- reduced set of deadlines to evaluate,
- rapid static scheduling performance,
- minimization of initial scheduling latency,
- adaptation to changes in resource allocation (e.g., bandwidth),
- responsiveness to dynamic user interaction,
- close matching of required to available resources, and
- tolerance to changing quality of service

In this section we describe our proposed scheduling mechanism which provides a balance between static and dynamic scheduling for multimedia object retrieval, transmission, and playout. We call the approach *limited a priori* (LAP) scheduling, being based on a static, *a priori* scheduling approach[11], but supporting dynamic user input and system load changes by periodic schedule recomputation. The LAP scheduling approach is comprised of a static resource reservation mechanism, and a dynamic, run-time executor of the LAP-produced

schedule. We call these components the *LAP reservation mechanism* and the *session scheduler*, respectively.

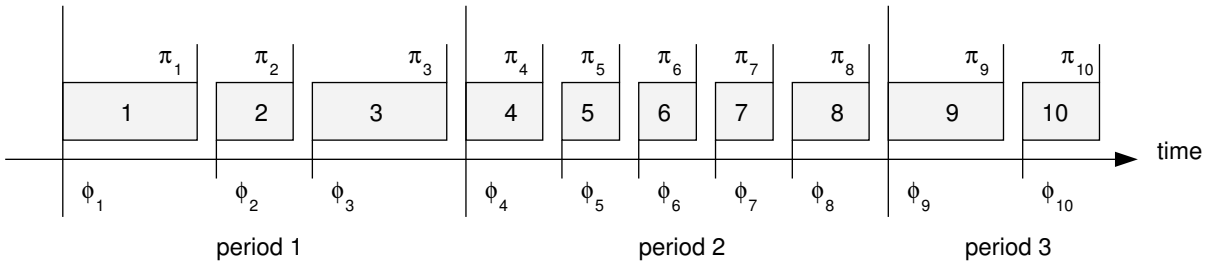

Figure 4: Schedule decomposed into periods

The essence of the LAP scheduling is as follows. A multimedia session is decomposed into periods of similar resource utilization that can be scheduled independently (Fig. 4). Each period is allocated resources and scheduled using statistical resource reservation. The resultant schedule is then executed, along with other similar schedules, by the session scheduler.

The session scheduler is responsible for evaluating the schedule generated by the LAP reservation mechanism, responding to dynamic user input, and managing the execution of other non-real-time tasks. In the remainder of the section we elaborate on the scheduling approach and its relation to real-time static and dynamic scheduling.

3.1 Limited *a Priori* Scheduling

The LAP resource reservation mechanism operates on a sequence of deadlines corresponding to the playout of multimedia data elements. These elements are identified and transformed in the process of static schedule generation.

3.1.1 Initialization

Upon user selection of a multimedia object for presentation, the playout timing must be identified to facilitate retrieval and playout scheduling. We use a temporal-interval-based[11] approach to managing object time dependencies that supports TAC operations (fast-forward, reverse playout, etc.) and indexing on individual object components for general database access. This representation is transformed into a playout schedule based on the access mode selected. The resultant playout schedule, $\Pi = \{\pi_i\}$ represents a monotonically increasing set

of deadlines that are appropriate for the LAP scheduling approach. Once a playout schedule is created, in part or in its entirety, the LAP reservation mechanism is applied to produce a feasible retrieval schedule.

3.1.2 Static Schedule Generation

Like the aforementioned channel reservation service (Section 2.2), the LAP reservation mechanism calculates the retrieval schedule by evaluating the playout sequence Π corresponding to a multimedia object for the required source-to-destination communication path. Delay and bandwidth of the channel, CPU and storage devices involved in data retrieval are used to compute the resultant LAP retrieval schedule. However, instead of interpreting the entire playout schedule, a reservation algorithm[11] (not shown here) is applied to groups of playout deadlines assigned to periods of evaluation (Fig. 5). The result is a set of retrieval deadlines, $\Phi = \{\phi_i\}$ for each period which are based on the current resource allocation and the original playout sequence. Once a retrieval schedule is generated, it is executed by the session scheduler that redirects arriving data to the subsystems. The individual subsystem devices merely receive data as they are released by the session scheduler. While the session scheduler executes a schedule generated for the current playout period, the LAP reservation mechanism generates the retrieval schedule for the next group of playout deadlines (Fig. 5).

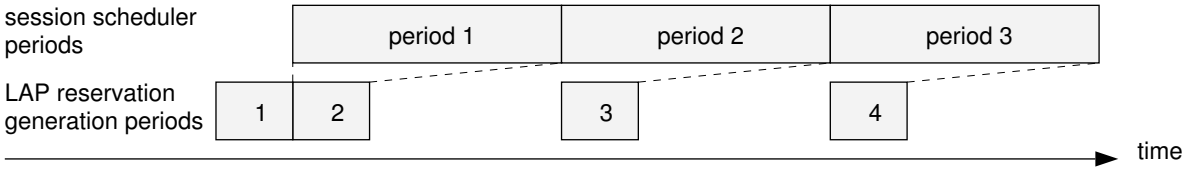

Figure 5: Interleaving of LAP and session scheduling

3.1.3 Changing Resource Commitments

A major advantage of the LAP scheduling approach is its ability to create a schedule which is closely matched to the current resource commitment (e.g., allocated communication bandwidth). At the beginning of each period, subsequent retrieval/transmission scheduling is performed using the latest resource utilization characteristics. Changes that occur to the committed bandwidth only affect the LAP schedule that has been released to the session scheduler, i.e., the playout schedule under active execution. Subsequent periods use the

newer resource allocation information and are not affected by the change. In a similar manner, the bandwidth requirements for a session are not overstated because they are done on a period by period basis.

3.1.4 Response Time to Dynamic User Input

By scheduling on a periodic basis, input through user interaction which might invalidate a precomputed schedule can be accommodated. A schedule for the playout of a digital motion picture requires resource allocation and the generation of a suitable retrieval schedule over limited bandwidth constraints. For example, if a user decides to stop and reverse the playout of a motion picture, its playout sequence and precomputed schedule become invalid. On the other hand, if the schedule is computed in blocks corresponding to the scheduling period, then little schedule computation is lost. Furthermore, latency associated with initial scheduling is reduced to short intervals.

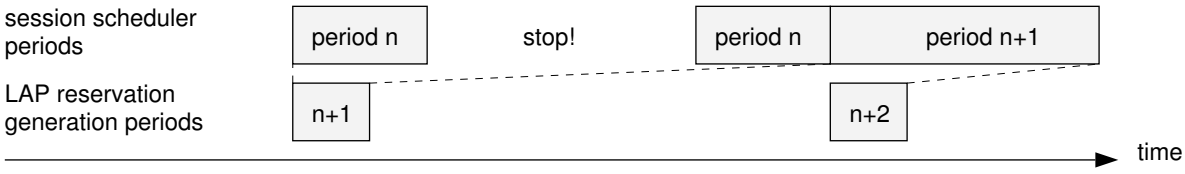

Figure 6: Response to *stop* signal

Response time to user interaction depends on the TAC operation that is invoked. A simple *stop* of playout halts the presentation of the current block and can stop the computation of future playout intervals (Fig. 6). A *reverse* or *fast-forward*, however, requires recomputation of both the playout and LAP schedules. Response time for these operations will be equal to the single period computation time for creating the playout sequence and LAP schedule (Fig. 7).

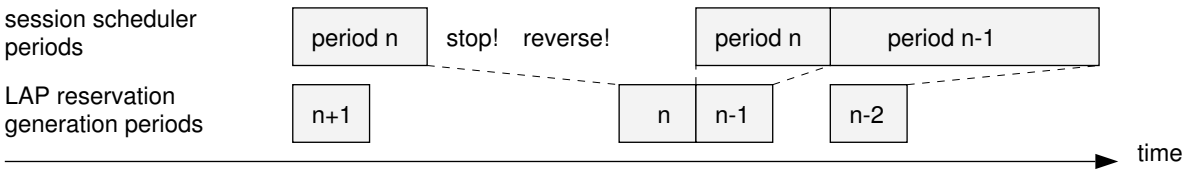

Figure 7: Response to $reverse$ or $fastforward$ signal

The creation of a schedule must always begin before a schedule is executed. This amount of time is the same for all schedules if the schedules being created are of similar length in terms of estimated execution time. This similarity leads to a constant load on the session scheduler throughout the set of periods comprising the session. This approach could be made more responsive by allocating enough CPU processing time so that currently active schedules can be recalculated and reissued when there is a user input or system change. An advantage of this LAP scheduling is that the amount of processing time needed for the calculation of a schedule predictable, allowing for effective real-time scheduling.

3.1.5 Bulk Retrieval

Another advantage of scheduling over an interval is the potential for retrieval of data in groups. By grouping a set of data elements, a more efficient request can be queued at a storage subsystem instead of issuing requests for individual data elements as they are scheduled. This consideration has more impact on the session scheduler which must interpret the retrieval schedule generated by the LAP reservation mechanism.

It is also desirable to choose appropriate block sizes in other contexts. The various components involved in the delivery and use of multimedia data are influenced in different ways by data block size. Physical subsystems such as a rotating-disc storage devices, audio subsystems, and video subsystems have different optimal data block sizes. However, these block sizes also depend on functionality. For example, continuous data retrieval from a storage device has an optimal interleaving factor depending on the device, the media, and the access functionality[10]. Additional application functionality can affect data block sizing, including support for fast or reverse playout, and the granularity of editing operations: between frames or sets of frames. The process of cutting and pasting segments of video can create less than optimal storage patters affecting subsequent multimedia presentation.

3.1.6 Interval Size Selection

The LAP scheduler performs static, *a priori* resource reservation for groups of data elements comprising a multimedia session. The groups of data elements correspond to periods of evaluation. At one extreme, we can use an evaluation period of minimum length, i.e., we can evaluate the playout deadline of each data frame. This corresponds to totally dynamic scheduling. As each frame is required, the scheduler can evaluate the system loading and attempt to allocate sufficient bandwidth (communication or computational). Unfortunately,

such a scheduling policy results in the loss of benefit from lookahead. Periods of highly parallel activity in a multimedia session can then overload the available resources. By choosing a nonzero period, these high activity intervals can be spread out across several periods and accommodated. At the other extreme, a single period encompasses an entire multimedia session and static scheduling results. Since the whole schedule can be interpreted, and optimal retrieval schedule can be computed. However, if the schedule is computed based on changing statistical bandwidth properties the result will either be wasted resources, or the invalidation of the schedule and the inability of the system to satisfy the presentation.

We envision periods to be of approximately equal length in terms of resource utilization to simplify session scheduling. That is, suppose the playout sequence $\Pi = \{\pi_i\}$ consists of m playout deadlines each with characteristic resource requirement e_i. The sequence can be divided into p periods of multiple playout deadlines with approximately equivalent bandwidth requirements, or $E_k \simeq E_l$ where $E_l = \sum_j e_j$ in period l. However, each period need not have the same number of tasks. Basically, we can perform scheduling by splitting our object into periods of approximately equal execution time and effectively smoothing or averaging the load over a longer interval. The larger the window, the more averaging is possible, within the constraints of buffering at the destination.

Other factors influencing the size of the evaluation interval are the time to reevaluate system load changes and monitor subsystem utilization. A certain percentage of CPU resources can be allocated to calculate schedules and negotiate levels of service from subsystem managers during each period. The creation of a period's schedule is begun early enough for resource negotiation and close enough to the schedule enactment to be responsive to recent user inputs or system changes. Also to be considered are the constraints on dynamic buffer size as this also relates to the maximum period. An important consideration in the implementation of the LAP reservation mechanism is the policy for evaluation of consecutive intervals. A totally static scheduler can interpret the entire set of deadlines to be scheduled. The LAP reservation mechanism can only look ahead within a period. A compromise is to allow scheduling periods to overlap, thereby averaging the discontinuities that result from a period-by-period static scheduling approach. This amounts to a sliding period of evaluation.

The complexity of the static scheduling is proportional to the number of deadlines to be scheduled. By restricting the static scheduler to an interval. We benefit by reduced complexity in static schedule creation. Furthermore, the schedule generation performance will be enhanced by the management of fewer deadlines. The penalty is the reevaluation of resource changes, but only when they occur. There is also the benefit of closely matching

the allocated resources to the required ones to prevent resource wasting.

3.2 Session Scheduler

The primary responsibility of the session scheduler is to execute the static schedules after they have been created by the LAP reservation mechanism. In addition, we envision this process to manage multiple sessions, monitor resource commitments, and schedule background tasks.

The session scheduler maintains the status of current resource commitments. As sessions are added, the commitments are changed. For example, the percentage of available CPU computational bandwidth is reduced. At the next period, these changes are interpreted by the LAP reservation mechanism and the result is a redistribution of available resources, subject to session resource priority. In a similar manner, as sessions terminate, released resources can be reapplied to existing sessions to improve their quality.

To allocate CPU bandwidth, we define an average CPU utilization averaged over a number of LAP periods. The remaining and available bandwidth can be used in scheduling new sessions. As more sessions are added, predicted execution bandwidth for operations such as retrieval, transmission, and decompression decreases as the total utilization of the resource increases. The LAP reservation mechanism applies the new values of utilization in the generation of individual session schedules.

3.3 Scheduling of Subsystems

A session will have specific demands on the each subsystem. The session scheduler requests a level of performance from each subsystem through process called a *subsystem manager*. Each subsystem manager can either commit to that level or offer a lower level of service. This commitment is based on QOS parameters. Once the resource commitments are retrieved, the session scheduler reconciles each QOS offer, and issues a confirmation of the negotiated service levels.

This conciliation between the session scheduler and the subsystem manager is necessary in many cases. For example, when a video subsystem only has capacity for 15 frames/s (due to the existence of another video session in progress), there is no reason to commit the session scheduler to the full rate of video retrieval (30 frames/s). Based on the level of commitment from each of the subsystems the session scheduler can decide not to execute

the presentation. A factor in this decision is the lowest level of quality of service that is acceptable to the user as predetermined at session initiation. If the resulting presentation is unacceptable, the user can attempt improve the session by altering the QOS parameters which would then take effect during the next scheduling period.

4 Scheduling and Quality of Service

An important consideration for system support for time-dependent multimedia data is the ability to manipulate the spatial and temporal resolution of data prior to, and during presentation. By giving the system the ability to manipulate these characteristics, a multimedia session can be adapted to available processing, communication, storage, and display resources. Spatial resolution can be interpreted as instantaneous resolution characteristic of imagery but also applicable to audio data. Image quality can be controlled, as a QOS parameter, by approaches such as progressive image resolution transmission. Temporal resolution defines the granularity of data presented over time. Because video streams have ample temporal redundancy, it is feasible manipulate temporal resolution by dropping frames to control its required bandwidth.

In a similar manner, QOS characterization quantifies the importance of data losses and time delay in the delivery and use of all multimedia data types. For a resource reservation and real-time scheduling system, we ultimately desire a system which will allow flexibility managing system resources for a set of multimedia sessions, but will also provide for a constant user-perceived QOS. To this end, we seek a "smooth" QOS function on which to operate. This function would guide the tradeoff between different system parameters (e.g., a range of temporal versus spatial resolutions which yield a constant QOS). We have not formalized such a QOS function at this time. However, we have identified some of the basic QOS considerations. In Table 1, we summarize some of the QOS parameters affecting the various subsystems of a multimedia delivery system.

In Table 2, we show values of various real-time parameters for each data type. When initially allocating resources, these QOS parameters can be considered for each medium and applied in the creation of feasible schedules. For example, when scheduling the communication channel audio data must arrive closer to their deadlines than image data. However, a major deficiency in this system of resource allocation is its inflexibility. It allows no means of deciding which parameters are the most important, or which should be compromised in limited resource situations to minimize the impact on total quality of service. Therefore, Ta-

Table 1. Quality of service parameters for various subsystems

subsystems	**quality of service parameters**
CPU	dynamic inputs, number of sessions, inter-playout time
storage devices	throughput, block size, organization, editability
network	bandwidth, delay, jitter, and error
audio subsystem	complexity of decompression, resolution, tolerance to loss
display subsystem	number of windows, color maps, size, resolution
video compression subsystem	complexity of images, tolerance to loss, size, resolution
user	cost, ease of use, flexibility

ble 1 includes a priority assingment based on relative importance to total QOS. These rough factors can be modified to emphasize various aspects of the presentation. These priorities can be further adjusted when considering the the application or any interaction between two portions of session. For example, a video window almost entirely occluded by another video window will have a lower priority than the fully visible window.

Table 2. Characteristic QOS parameter values for different data types (adapted from Hehmann et al.[5]).

Relative priority show in parentheses (4 lowest).

Data Types	**Maximum delay Jitter (ms)**	**Acceptable bit error rate**	**Acceptable packet loss rate**	**Maximum Consecutive Loss**
Voice	10 (1)	0.064 (1)	$< 10^{-1}$ (2)	(2)
Video (TV Quality)	10 (2)	10^{-2} (2)	10^{-3} (3)	(2)
Compressed video	1 (2)	10^{-6} (2)	10^{-9} (2)	(2)
Text	- (4)	0 (1)	0 (1)	(1)
Control Data	- (3)	0 (1)	0 (1)	(1)
Image / Graphic	- (4)	10^{-4} (2)	10^{-9} (1)	(1)

QOS parameters can be used to make changes in resource allocation with little change in the overall quality of a session. This is especially important when there is a run-time change in the resource load. The proposed system minimizes the impact of decreasing resource availability and maximizes the impact of newly allocated resources. Such a change can occur when there is dynamic user input or an increase in the amount of computer processes not involved in the current presentation. Services can also need to be provided by the system for additional multimedia sessions. A user may want to expand the current session or even add additional sessions. In this instance, QOS parameters are important because they aid the system in making decisions on the execution of all sessions given the significantly different set of resource requirements.

QOS parameters are also important for allocating resources within the subsystems. In our proposed LAP scheduling system, the level of service provision is negotiated between the session and the subsystem manager. The subsystem managers can evaluate the parameters intrinsic to their media (e.g., video playout subsystem), the priority of the parameter to the application, and the relative priority of the session when multiple sessions are involved. The relative importance of a session can be supplied by the user and may take into account the desire to reserve a percentage of resources available for other sessions. Future research will allow this priority to be further adjusted by considerations such as video window occlusion and focus of attention of the current session. Psychophysical analysis can also aid in deciding which session requires more data to impart its content. For example it is more important in videotelephony for a talking head to have a high image quality than a one that is not talking or otherwise participating in an interactive session.

Clearly it is difficult to quantify quality as it is a subjective attribute. However, we can characterize some QOS parameters. In Fig. 8, we show the relative cost penalty for jitter affecting various media. In each case, the cost increases with a larger value of jitter. For audio and video, the cost drops to zero after some interval when data discarding occurs, as permitted when temporal redundancy exists. For still images and text, jitter is less significant as is reflected in the illustration. However, these data have little temporal redundancy and cannot be discarded.

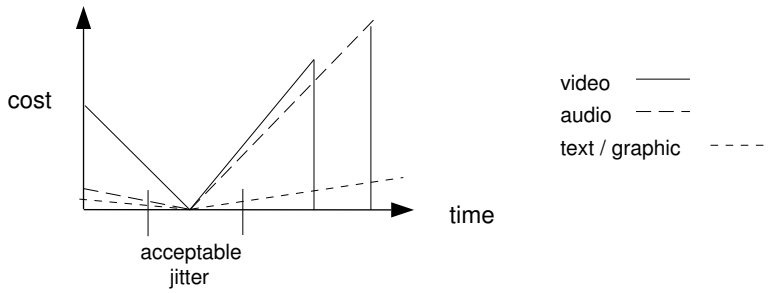

Figure 8: Cost versus time for delay jitter variation

Application-specific factors must also be considered in QOS evaluation. A graphic must have a low jitter value if it is displayed as part of a high-speed animation. In a multimedia slide presentation, the same graphic can tolerate a larger jitter value. The spatial location of visual information is also important. A video stream that is 70% occluded by another image does not need premium delivery service, in contrast to a diagnostic X-ray which requires the delivery of full spatial resolution. QOS parameters mut also be identified for compressed

data formats which have additional penalties for losses. For intra-frame coding (e.g., JPEG compression), losses of any fraction of a frame result in the loss of the entire frame. For inter-frame coding (e.g., MPEG compression), a loss within a frame can cause the loss of multiple frames.

Once a complete QOS characterization is attained, a real-time scheduling system can operate within a range of acceptable QOS by adapting to available resources. Although our initial characterization is primitive, we are progressing to a more complex framework.

5 Conclusion

In this paper we have described the use of real-time scheduling disciplines for supporting the delivery of time-dependent multimedia data in a distributed-data environment. We have also proposed a limited *a priori* scheduling mechanism that is designed to overcome some of the deficiencies of static, statistical bandwidth reservation. The proposed scheduler allows for dynamic user interaction and changes in system loading to be accommodated at regular intervals of schedule reevaluation. We emphasize that this work is evolving from the design stage and is currently unproven and untested. We are presently evaluating the feasibility of our approach and will likely introduce significant changes. Therefore, there are many current and future research issues to address.

With respect to session scheduling, we seek an appropriate scheduling discipline (e.g., earliest-deadline). Although we believe that the least-laxity approach is the most suitable, it is likely that another method will be more appropriate for regular, periodic events associated with time-dependent data. We also intend to further investigate QOS for multimedia data types as we seek the elusive "smooth surface" to support graceful service degradation. This requires a better understanding of user tolerances to spatial and temporal resolution, as well as system parameter changes including delays and losses. Further, we plan on better characterizing delays in system components to support resource allocation decisions. We are now able to characterize data retrieval delay distributions from rotating disc-type storage devices[10].

6 Acknowledgements

We would like to thank Prof. Azer Bestavros for his helpful discussions on real-time scheduling.

7 References

1. S. Cheng, J. A. Stankovic, and K. Ramamritham, "Scheduling algorithms for hard-real time systems - a brief survey," *Hard Real-Time Systems*, J. A. Stankovic and K. Ramamritham, Ed., pp. 150-173, IEEE Computer Society Press, Washington D.C., 1988.

2. D. Ferrari, "Design and application of a delay jitter control scheme for packet-switching internetworks," *Proc. 2nd Intl. Workshop on Network and Operating Support for Digital Audio and Video*, Heidelberg, Germany, November 1991.

3. D. Ferrari and D. C. Verma, "A scheme for real-time channel establishment in wide-area networks," *IEEE Journal on Selected Areas in Communications*, Vol. 8, No. 3, pp. 368-379, April 1990.

4. M. Gilge and R. Gusella, "Motion video coding for packet-switching networks - an integrated approach," *SPIE Conference on Visual Communications and Image Processing*, Boston, November 1991.

5. B.B. Hehmann, M.G. Salmony, and H.J. Stuttgen, "Transport services for multimedia applications on broadband networks," *Computer Communications*, Vol. 13, No. 4, pp. 197-203, May 1990.

6. A. A. Lazar, A. Temple and R. Gidron, "An architecture for integrated networks that guarantees quality of service," *International Journal of Digital and Analog Cabled Systems*, Vol. 3, No. 2, 1990.

7. D. Le Gall, "MPEG: a video compression standard for multimedia applications," *Communications of the ACM*, Vol. 34, No. 4, pp. 46-58, April 1991.

8. B.I. Szabo and G.K. Wallace, "Design considerations for JPEG video and synchronized audio in a Unix workstation environment," *Proc. USENIX*, pp. 353-368, Nashville, TN, 1991.

9. Technical Committee on Operating Systems of the IEEE Computer Society, "Realtime extension for portable operating systems," P1003.4/Draft 10, February 1991.

10. T.D.C. Little and H.-J. Chen, "A file system for multimedia applications," working paper, Multimedia Communication Laboratory, Boston University.

11. T.D.C. Little and A. Ghafoor,"Scheduling of bandwidth-constrained multimedia traffic," *Computer Communications* (Special Issue: Multimedia Communications), Vol. 15, No. 6, pp. 381-387, July/August 1992.

12. T.D.C. Little and J.F. Gibbon, "Management of time-dependent multimedia data," to appear in the *Proc. SPIE Symposium OE/FIBERS'92, (Enabling Technologies for Multi-Media, Multi-Service Networks)*, Boston, MA, Sept. 1992.

13. W. A. Montgomery, "Techniques for packet voice synchronization,"*IEEE Journal on Selected Areas in Communications*, Vol. SAC-1, No. 6, pp. 1022-1028, December 1983.