Proc. of the ACM Multimedia '94 Conference, Workshop on Multimedia DBMSs, October 1994, pages 63-70

The Calico Project for Continuous Media Services

Bruce K. Hillyer, Alexandros Biliris, Euthimios Panagos
AT&T Bell Laboratories
{bruce, biliris, thimios}@research.att.com

Abstract

Multimedia applications are becoming part of our daily interaction with computing systems. These applications need storage system support for both continuous media and conventional data, requiring advances in many areas, including scheduling, resource management, admission control, hierarchical storage, and the application/server interface. The Calico continuous media storage system, which is currently being implemented in AT&T Bell Labs, is a testbed to examine these issues. This paper provides an overview of a hierarchical resource model for the Calico server to manage resources and schedule disk and network requests to the operating system. Our goal is to support efficient, high quality continuous media service in distributed and somewhat unpredictable environments such as networked conventional UNIX workstations and PCs.

1 Introduction

Great demand has arisen for continuous media service on networks of conventional workstations and personal computers running common operating systems like UNIX. But the usual mechanisms in this environment provide fair resource allocation with no notion of time-sensitive behavior. Consequently, as the system load increases, a smooth playout of continuous media streams cannot be maintained because of loss of CPU cycles and network and disk bandwidth, leading to delayed transfers. The solution is improved resource management and scheduling.

This paper presents the efforts of the Calico project to support continuous media for conventional networked workstations and PCs. We describe user-level mechanisms and policies designed to give good, efficient multimedia service under the mild assumption that the operating system provides a preemptive real-time scheduling class that can be used to give CPU cycles to the multimedia processes in preference to other processes that are not time sensitive. Sun's Solaris and some versions of Hewlett-Packard's HP-UX provide this, as does Microsoft's Windows NT. We do not assume isochronous networks or operating system kernel modifications, and do not assume that applications can provide detailed descriptions of the resources they will consume in the server; it suffices for an application to state that it wants to play a particular stream of a type known to the server (e.g. a file containing MPEG-1 video 320x240 pixels, 8-bit color, 30 frames per second), or for the application to specify a frame rate and an index describing the offset of each frame in a file.

To give good continuous media service, resources need to be managed to avoid shortages that would cause interruptions. Three broad choices in the literature are

1. *Hard real-time scheduling.* This approach requires an environment where every latency and resource requirement is known [NRS+93, SZ92]. Hard real-time scheduling is expensive, and is impossible in general timesharing environments characterized by unpredictable resources and delays.

2. *Conservative resource usage estimates.* Admission control limits the service load to that known to be safe in the worst case, or known to be safe with high probability [JSZC92, AOG92, And93]. However, conservative estimates lead to low utilization in a general operating system setting.

3. *Resource modeling and reservation.* Admission control and resource scheduling is performed by a model of the resources available and the resource consumption demanded by a new request [LS93, WHN92, VR92]. One problem here is that a resource model of the system cannot be both highly accurate and inexpensively computed, particularly for general timesharing environments. Another problem is that the precise resource consumption implied by a new request for service is generally unknown.

The above approaches generally share the *admission control assumption*, namely that a request for service should either be refused immediately, or should

be guaranteed good service. (We do not discuss "best effort" systems that make no guarantees whatsoever.) To support the admission control assumption, the systems must place fairly strong requirements on the determinism and predictability of the system and workload.

Our algorithms schedule the issuing of I/O requests to the operating system on the server so as to manage the bandwidth of the disks, disk controllers, memory subsystem, and network to get the needed multimedia frames to the clients in the right timeframes. If the network supports bandwidth reservation, the network portion of the scheduling reduces to simple admission and monitoring. The techniques for resource management in the client/server environment are also applicable to systems playing continuous media from local disks. The network scheduling is applicable to (local area) network environments where the server send and client receive operations are closely coupled. A different and important area of research involving store and forward wide area networking has been studied elsewhere (e.g., in [HD92]).

The goals during the first phase of our work are the following:

1. A weaker form of the admission control assumption that is suitable for a general application environment such as UNIX workstations and PCs on networks that may not support bandwidth reservation.

2. A hierarchical resource scheduling algorithm that, based on simple conservative resource availability and consumption functions, identifies the initial set of transfers to be issued to the operating system at the beginning of each schedule cycle.

3. An algorithm for dynamic slack determination and filling that observes actual resource consumption during a cycle and dynamically initiates additional transfers to improve utilization.

In an environment where the available resources cannot be characterized precisely, and the resource demands implied by client requests for continuous media service are not accurately known, our algorithms give good service to the most important[1] continuous media jobs, focusing resource insufficiency on the least important jobs. This is the best that can be done without stronger assumptions on the predictability of the system and workload. To our knowledge none of the existing approaches do this.

[1] Most important is determined by an arbitrary *prioritization policy* that places a rank ordering on the jobs.

Currently, the focus of our work is on playback of continuous media (e.g. audio and video). The recording of continuous media is not a time sensitive activity: a server having sufficient bandwidth to accommodate the stream can use buffering and delayed writes to handle recording of continuous media at its convenience.

The environment of interest is quite different from that of specialized multimedia systems, such as video on demand, which are generally characterized by isochronous or reservation networks connecting special-purpose servers with specialized client systems. In that environment, the workload characterization and system performance controls are sufficiently tight that deterministic scheduling can give high quality with high system utilization.

2 Overview of the Idea

When a new job arrives and requests service, a simple admission triage generates one of three responses

- Trivial admit. The system is very underloaded and can provide the requested service.
- Trivial reject. The system is very overloaded so service is refused.
- Conditional admit. Service will be attempted.

The guarantee given by admission is not unconditional good service. It is that the service to an admitted stream will not be degraded until all "less important" streams have first had their service cut off, where "less important" is defined by some importance ranking policy. Overload handling and service degradation is discussed further in section 2.1.

The server issues batches of disk and network transfer requests to the operating system to gain the benefits of disk scheduling and overlaps. But this means the requests will complete in unpredictable orders. Cyclic (periodic) scheduling is used to establish time frames that can be monitored to detect tardiness before jobs actually miss their deadlines.

For each cycle the server performs relatively simple conservative resource modeling to grant bulk admission to this cycle for the most important jobs that can safely be predicted to complete on time. During the cycle the server does online tracking of actual resource consumption, and dynamically admits additional transfers to this cycle when possible. The idea is to only pay the cheap cost of simple modeling, but to achieve a good utilization by dynamically filling slack

regained from conservative estimates by the early completion of transfers.

The server does not actively manage CPU cycles. We assume it runs in a real time process class so the continuous media service can prevent other jobs (that are not as time-sensitive) from starving the continuous media streams for CPU cycles. Given this, we assume that the system bottleneck is somewhere in the disk, network, or memory subsystems.

The scheme does not depend on a balanced architecture: the resource scheduling dynamically adjusts to the bottleneck resource, whether that be a disk, disk controller, network interface, or the memory subsystem. Consider, for instance, fast workstation with Ethernet vs. fast workstation with gigabit ATM networking.

The network is not required to support bandwidth reservations, but if they are available then the system efficiency should increase because more streams will be granted bulk admission to a cycle, and fewer will be served by absorbing slack dynamically freed during the schedule. In addition, clients obtain better assurance of high service quality because the hazard of network bandwidth shortages has been eliminated. Of course for non-reservation networks like Ethernet the clients will experience degradation if the network bandwidth goes away. Several schemes are described in the literature for obtaining good service in practice by buffering ahead and coping with temporary bandwidth reduction by techniques such as frame dropping or audio dilation [JSTS92, AH91, LK92, KR94].

2.1 Overload Handling and Quality of Service Degradation

If the system is not totally isochronous so it can be perfectly scheduled, or if scheduling is not maximally conservative, then there will be overloads. Consider for example variable bit rate compressed video, or scripted presentations that have peaky bandwidth demands: given that the user interface provides VCR functions including pause, fast forward, and rewind, then an adversary can cause the peaks to coincide in the worst possible way.

So the question is how to handle overloads. Numerous techniques have been proposed for load shedding. Some systems such as [HZ94, And93] serve three classes of clients: continuous media, interactive response, and background; the latter two can have their service rate reduced to save resources for continuous media. Vin et al. [VGGG94] suppose that some streams are pre-identified as degradable, i.e. able to cope with reduction of bandwidth, and their system degrades these streams first, based on measures of how loss tolerant each client is, spreading the degradation proportional to loss tolerance. Other proposals for load shedding include the use of scalable digital coding and service degradation by reducing resolution [CZ94], the deferral of processing for requests that the application has indicated that are less important [WHN92, CT94], and the reduction of bandwidth by reducing resolution or frame rate [CT92]. The Pegasus operating system forcibly revokes resource reservations to reduce the bandwidth provided to the client [LMM93]; the client is presumed able to deal with this.

Our position is that these are *policies*, so the server has no business selecting which of these techniques to apply. We assert that if a stream is missing its deadlines because of system overload, an application-selected policy should be applied to that stream. The server just provides the mechanisms.

A strategy of degrading service to cope with overload will be perceptually significant, because slight degradation of the streams won't yield significant resource savings. For instance, if the server disk subsystem is the bottleneck, killing 10% of the streams will shed 10% of the load, but dropping 10% of the frames in all streams sheds much less than 10% of the load because disk seek overhead becomes relatively higher. Omitting a few frames from the middle of a whole track read may result in no mitigation of overload whatsoever [VGGG94].

The conventional admission control assumption leads to policies that accept new clients for service, even if this means degrading the quality of service provided to existing clients, up to some threshold beyond which new clients are refused. By contrast, our admission control does not need to predict the degradation that would result from admitting the next client, because clients are admitted subject to the condition that any degradation required to handle a resulting overload will be focused on the least important streams. In particular, our scheduling algorithms provide a mechanism to protect a subset of the streams, and use a *prioritization policy* that explicitly chooses which streams to protect and which others to degrade. We can use any policy that gives a rank ordering over all the streams. A simple policy that appears useful for the server is to protect the oldest streams, degrading the newest.[2] Thus streams that have been running

[2] Jones and Hopper [JH93] give good arguments that on the client side, when overloaded, a desirable prioritization policy is to degrade old streams in favor of the newest one. (Scenario: accept a video call, let it degrade the movie playing in the background; the movie will again receive good service when the video call is over.)

for a long time will tend to be protected from degradation, while streams that recently started may be interrupted if it turns out that there really isn't sufficient capacity to serve them. A new client who requests service can quickly see whether the quality of service is satisfactory and decide whether to continue or quit. Thus resource insufficiency is focused on unsupportable newly added load, stimulating client-driven load shedding. Furthermore, a client can state how it would like be degraded (e.g. lower frame rate or resolution) if it would otherwise be suspended because of resource insufficiency. Note that this has been a discussion of load shedding policy (who to hurt during overload), not how conservative to make admission control (to set the tradeoff between system utilization and probability of degradation).

2.2 Composite Streams

Consider composite streams such as video with audio, or scripted presentations consisting of a time-related collection of continuous media streams and discrete data objects such as images that will be synchronized at the client for joint presentation. From a human factors standpoint, loss of audio is a much greater failure than loss of video. Therefore, scheduling should consider more than a frame's intended presentation time: in systems that can't provide hard real-time guarantees, it is wise to send small valuable frames well ahead of time, and buffer them on the client side. This can be hidden in the client-side libraries that handle composite audio-video streams by prefetching audio data to the client buffer far in advance. Transfer scheduling to the client is distinct from presentation scheduling at the client.

The server of Long et al. [LOC93] sends data at a slightly faster frame rate than specified by the client. This builds up the amount of data buffered at the client until buffer space limitations apply backpressure, establishing a cushion to protect against jitter and modest stream stalls. Other systems such as [KR94] use dynamic feedback control of the transfer rate to maintain a satisfactory client buffer size.

Our server supports applications that wish to trade more aggressive degradation of some streams in exchange for better protection of others. Applications can specify tradeoffs in the prioritization policies of the component streams of a composite or scripted job. A newly arriving job can boost the priority of some components while antiboosting the priority of others, provided (a) that the bandwidth-weighted average priority does not increase (the job must give up more than it gets), and (b) it doesn't injure established streams unless these streams have antiboosted priority (the job only hurts those who gave permission to be hurt in exchange for their own boosts). In other words, a stream is only permitted to boost if it (a) does not cause an overload, or (b) causes an overload that only degrades antiboosted streams.

It may be necessary to prevent an application from boosting the priority of a scripted component now in exchange for hurting another component in the future, otherwise it could cheat by appending dummy components to the end of a script, boosting to get good service now in exchange for degradation of the dummy portion that won't really be used anyway. Two ways to handle this are (1) an economic solution: charge for the entire composite, including dummy portions, or (2) pre-verification or policing of the entire schedule: during no cycle does the bandwidth-weighted boost exceed the antiboost. Similarly, an application should not be able to pad composite streams with dummy streams that exist only to be degraded. Two ways to handle this are (1) economic: charge for the entire composite including dummy portions, or (2) discriminate against higher bandwidth jobs so that adding a dummy stream and degrading it maximally, boosts the remaining streams only to the priority they would have obtained had the dummy stream not been a member of the composite in the first place.

Similarly, the overall approach of favoring older streams encourages a client to improve its priority by opening a stream hours ahead of time, but (`pause`) not starting the actual data transfer until much later, at which point it could have amassed sufficient priority to displace clients that had been running smoothly for the past tens of minutes. This may be ok if a client is charged from the time the stream is opened: a client can buy better service by holding open an idle line, and the server has been charging a client that has been generating no server load.

3 Alternative Techniques

In this section we compare various approaches described in the literature with the Calico efforts.

3.1 Admission Control

Many systems incorporate the idea of job admission control as the mechanism to obtain a good quality of service. The idea is to protect the existing jobs by refusing new jobs that could result in system overload. If the admission control is too conservative, the result is system underutilization. Too

liberal admission leads to overloads and consequent degradation of the existing jobs. This sort of admission control doesn't *guarantee* quality of service, and it requires considerable knowledge about the capacity of the system and resource consumption of the clients. Specific systems include the following. Vin et al. [VGGG94] perform admission control based on current measured load and the resource requirements of the new stream. [JSZC92] also perform admission control of a new client based on the current traffic measurements. The GRAMS system [HZ94] classifies requests into three categories, one for continuous media, a second for demand-driven retrieval by interactive clients, and a third for retrievals that are not time sensitive. During overload conditions, service to the latter two classes is reduced or suspended. Overload is determined by a starvation counter technique based on buffer underflow. [SW93] gives a systematic overview of issues in constrained latency retrieval. Systems are classified by size and predictability of requests, and the authors present a table indicating that quality of service guarantees are generally unobtainable.

Our approach refuses admission only when the system is clearly in overload: admission control can effectively limit the server load, but generally cannot guarantee quality of service. Quality of service can't be *guaranteed* unless the system and applications are sufficiently predictable that hard real-time scheduling techniques apply. In systems that are nearly that predictable, statistical guarantees can be obtained. For the less predictable environment of networked workstations and PCs, a weaker criterion is needed. Our approach is to provide good quality of service for jobs that fall within the capacity of the system by a scheduling and resource monitoring approach that focuses the resources on the most important jobs. Important streams are protected by focusing degradation on the minimal number of least important streams. Our scheduling is robust because of the dynamic slack monitoring, whether overloads are the result of too many jobs admitted, overcommitted schedules, or external resource loss. This approach can detect and proactively handle impending cycle faults, with the potential to recover before any degradation of important streams occurs.

3.2 System Bottleneck

Many proposals are targeted at systems with the bottleneck in a specific subsystem, most often the disk subsystem, assuming fast networks such as ATM. Among these are the following. NTT's VTSS technique [FIKS94] uses rate based fetching, and rearranges client requests to improve startup response time. Any slack is used to fill the buffers with the least remaining playout time, so that the scheduling cycle period can be lengthened to drive up the system efficiency. The MOD server of [KCS94] features whole track transfers, no constraints on data placement placement, and sufficiently long disk queues that the SCAN disk scheduling algorithm makes the seek times negligible. The server of [DSK94] schedules batches of 8 concurrent requests to each disk controller; the controllers perform their own internal fetch scheduling. Each 64 KB disk block contains a pointer to the next, to avoid directory query operations. Disk striping is used. Admission control only considers disk utilization, and unfortunately, appears to be linear in the length of the movie requested. [Dai94] gives a formal analysis of disk scheduling for continuous media. They break huge disk transfers into track sized ones to make the disk more preemptable, which increases the effectiveness of the real-time scheduling algorithm they apply. They coalesce all periodic tasks into a superperiod and schedule this with a fixed period but different buffer sizes depending on the bandwidth of each stream.

Our system is not designed with one particular resource assumed to be the bottleneck. The hierarchical resource scheduling technique dynamically determines the bottleneck subsystem, and schedules and slack-fills to highly utilize that resource.

3.3 Hardware Organization

Some systems use outboard subsystems to handle continuous media separately from the computer processor and memory. This prevents interference between the multimedia and computing, but supports presentation rather than application processing of multimedia streams. Other systems use architectures that decouple the workstation subsystems into semi-independent components on a desk area network. Among these are Pandora [JH93] and the system of [BCC+93] that use attached network hardware to handle the routing of multimedia outside the workstation's memory bus. [LMM93] use the desk area network approach in Pegasus for the same reason.

Our system handles continuous media as first-class data on conventional workstations and PCs, transferred and processed with the same operating system mechanisms and privileges as any other application data. There is considerable sentiment that the interesting new algorithms want to manipulate the continuous media bits, not just issue VCR commands.[BCC+94]

3.4 Service Degradation During Overload

Many systems implement specific policies for degradation of continuous media streams during system overload, including those discussed with respect to admission control in section 3.1 above and in the discussion of load shedding in section 2.1.

Our system provides a collection of mechanisms, and lets applications specify the disposition policies to be applied to their own streams if they cannot be served on time.

3.5 Scheduling Approach

Many systems use scheduling algorithms from the real-time scheduling theory, including least laxity first, earliest deadline first, and rate-monotonic scheduling, but in general, these systems have insufficient information to guarantee deadlines.

Since we know we can't provide real-time guarantees, our system performs scheduling by an efficient greedy bin packing technique, and manages resource consumption dynamically, based on observation of the system performance actually achieved.

3.6 Client/Server Interface

Many systems state a specific API that defines the interworking between client and server. For instance, [Bul92] describes a support request from application to the Amsterdam Multimedia Framework based on a multimedia document specification that describes the document component's synchronization needs, resource needs, and priority. The system responds with trivial accept, trivial reject, or negotiated accept that starts a process of negotiation for service with fewer resources.

We just state a few requirements on the interworking, because our techniques are broadly applicable and largely independent of any particular API.

4 Concluding Remarks

We have provided an overview of our efforts in the Calico project for handling continuous media efficiently and with good service quality on conventional computers and operating systems, given a real-time scheduling priority such as in Sun's Solaris or Microsoft's Windows NT. The key ideas are (1) to perform hierarchical resource scheduling based on conservative estimates of resource consumption, and based on dynamic monitoring of actual resource consumption, improve utilization by slack filling, (2) protect the quality of service for the maximal number of important streams by focusing degradation during periods of resource insufficiency on the streams deemed least important according to some prioritization policy, and (3) use application-specified policies to degrade streams, rather than a common server-specified policy.

10 Mb Ethernet is common, but also is a severe bottleneck by comparison with typical disk subsystems. Both slow ISDN and fast ATM networking may play a role in our future systems, so it is important to have an architecture that can handle gross imbalances between available disk and net bandwidth. Calico's approach dynamically determines the bottleneck resource and schedules it for high utilization.

With respect to general storage servers, we see three useful classes of data service: continuous media, transactional storage (including persistent objects), and file service. It may be useful to have an integrated server that can handle all three classes, which is a possible future direction of Calico.

References

[AH91] D. P. Anderson and G. Homsy. A continuous media i/o server and its synchronization mechanism. *IEEE Computer*, 24(10):51–57, October 1991.

[And93] D. P. Anderson. Metascheduling for continuous media. *ACM Transactions on Computer Systems*, 11(3):226–252, August 1993.

[AOG92] D. P. Anderson, Y. Osawa, and R. Govindan. A file system for continuous media. *ACM Transactions on Computer Systems*, 10(4):311–337, 1992.

[BCC+93] G. Blair, A. Campbell, G. Coulson, F. Garcia, D. Hutchison, A. Scott, and D. Shepherd. A network interface unit to support continuous media. *IEEE Journal on Selected Areas in Communications*, 11(2):264–275, February 1993.

[BCC+94] G. S. Blair, A. Campbell, G. Coulson, N. Davies, F. Garcia, and D. Shepherd. Summary of the 4th international workshop on network and operating system support for digital audio and video (nossdav '93). *Operating Systems Review*, 28(2):22–33, April 1994.

[Bul92] D. C. A. Bulterman. Synchronization of multisourced multimedia data for heterogeneous target systems. In *Proceedings 3rd International Workshop on Network and Operating*

System Support for Digital Audio and Video, La Jolla CA, pages 119–129, November 1992. published as *Lecture Notes in Computer Science 712* P. Venkat Rangan (ed.), Springer-Verlag, Berlin.

[CT92] S. T.-C. Chou and H. Tokuda. System support for dynamic qos control of continuous media communication. In *Proceedings 3rd International Workshop on Network and Operating System Support for Digital Audio and Video* La Jolla CA, pages 363–368, November 1992. published as *Lecture Notes in Computer Science 712*, P. Venkat Rangan (ed.), Springer-Verlag, Berlin.

[CT94] C. L. Compton and D. L. Tennenhouse. Collaborative load shedding for media-based applications. In *Proceedings International Conference on Multimedia Computing and Systems* Boston MA, pages 496–501. IEEE Computer Society Press, Los Alamitos CA., May 1994.

[CZ94] E. Chang and A. Zakhor. Scalable video placement on parallel disk arrays. In *Proceedings Storage and Retrieval for Image and Video Databases II,* San Jose CA, February 1994. published as *SPIE Proceedings Series Vol. 2185*, SPIE, Bellingham WA.

[Dai94] S. J. Daigle. Disk scheduling for multimedia data streams. In *Proceedings High-Speed Networking and Multimedia Computing* San Jose CA, pages 212–223, February 1994. published as *SPIE Proceedings Series Vol. 2188*, SPIE, Bellingham WA.

[DSK94] J. K. Dey, C.-S. Shih, and M. Kumar. Storage subsystem design in a large multimedia server for high-speed networking environments. In *Proceedings High-Speed Networking and Multimedia Computing* San Jose CA, pages 200–211, February 1994. published as *SPIE Proceedings Series Vol. 2188*, SPIE, Bellingham WA.

[FIKS94] H. Fujii, A. Ishikawa, N. Kotani, and N. Sakurai. Multimedia server for on-demand services. In *Proceedings High-Speed Networking and Multimedia Computing* San Jose CA, pages 179–188, February 1994. published as *SPIE Proceedings Series Vol. 2188*, SPIE, Bellingham WA.

[HD92] R. G. Herrtwich and L. Delgrossi. Beyond ST-II: Fulfilling the requirements of multimedia communication. In *Proceedings 3rd International Workshop Network and Operating System Support for Digital Audio and Video* La Jolla CA, pages 25–31, November 1992. Published as *Lecture Notes inComputer Science 712*, P. Venkat Rangan (ed.), Springer-Verlag, Berlin.

[HZ94] J. Y. Hui and J. Zhang. Grams: a distributed multimedia service system. In *Proceedings High-Speed Networking and Multimedia Computing* San Jose CA, pages 189–199, February 1994. published as *SPIE Proceedings Series Vol. 2188*, SPIE, Bellingham WA.

[JH93] A. Jones and A. Hopper. Handling audio and video streams in a distributed environment. In *Proceedings of the 14th ACM Symposium on Operating Systems Principles* Asheville NC, pages 231–243, December 1993.

[JSTS92] K. Jeffay, D. L. Stone, T. Talley, and F. D. Smith. Adaptive, best-effort delivery of digital audio and video across packet-switched networks. In *Proceedings 3rd International Workshop Network and Operating System Support for Digital Audio and Video* La Jolla CA, pages 3–14, November 1992. published as *Lecture Notes in Computer Science 712*, P. Venkat Rangan (ed.), Springer-Verlag, Berlin.

[JSZC92] S. Jamin, S. Shenker, L. Zhang, and D. D. Clark. An admission control algorithm for predictive real-time service. In *Proceedings 3rd International Workshop Network and Operating System Support for Digital Audio and Video* La Jolla CA, pages 349–356, November 1992. published as *Lecture Notes in Computer Science 712*, P. Venkat Rangan (ed.), Springer-Verlag, Berlin.

[KCS94] D. D. Kandlur, M.-S. Chen, and Z.-Y. Shae. Design of a multimedia storage server. In *Proceedings High-Speed Networking and Multimedia Computing* San Jose CA, pages 164–178, February 1994. published as *SPIE Proceedings Series Vol. 2188*, SPIE, Bellingham WA.

[KR94] H. P. Katseff and B. S. Robinson. Predictive prefetch in the nemesis multimedia information service. In *Proceedings ACM Multimedia '94*, October 1994. (to appear).

[LK92] T. D. C. Little and F. Kao. An intermedia skew control system for multimedia data presentation. In *Proceedings 3rd International Workshop Network and Operating System Support for Digital Audio and Video* La Jolla CA, pages 130–141, November 1992. published as *Lecture Notes in Computer Science 712*, P. Venkat Rangan (ed.), Springer-Verlag, Berlin.

[LMM93] I. M. Leslie, D. McAuley, and S. J. Mullender. Pegasus–operating system support for distributed multimedia systems. *Operating Systems Review*, 27(1):69–78, January 1993.

[LOC93] D. D. E. Long, C. Osterbrock, and L.-F. Cabrera. Providing performance guarantees in an fddi network. In *Proceedings 13th International Conference on Distributed Computing*

Systems Pittsburgh PA, pages 328–336, May 1993.

[LS93] P. Lougher and D. Shepherd. The design of a storage server for continuous media. *The Computer Journal*, 36(1):32–42, 1993.

[NRS+93] D. Niehaus, K. Ramamritham, J. A. Stankovic, G. Wallace, C. Weems, W. Burleson, and J. Ko. The Spring scheduling co-processor: design, use, and performance. In *Proceedings Real-Time Systems Symposium* Raleigh-Durham NC., pages 106–111, December 1993.

[SW93] R. Staehli and J. Walpole. Constrained-latency storage access. *IEEE Computer*, 26(3):44–53, March 1993.

[SZ92] K. Schwan and H. Zhou. Dynamic scheduling of hard real-time tasks and real-time threads. *IEEE Transactions on Software Engineering*, 18(8):736–748, August 1992.

[VGGG94] H. M. Vin, A. Goyal, A. Goyal, and P. Goyal. An observation-based admission control algorithm for multimedia servers. In *Proceedings International Conference on Multimedia Computing and Systems* Boston MA, pages 234–243, May 1994.

[VR92] H. M. Vin and P. V. Rangan. Admission control algorithms for multimedia on-demand servers. In *Proceedings 3rd International Workshop on Network and Operating System Support for Digital Audio and Video* La Jolla CA, pages 56–68, November 1992. published as *Lecture Notes in Computer Science 712*, P. Venkat Rangan (ed.), Springer-Verlag, Berlin.

[WHN92] G. A. Wall, J. G. Hanko, and J. D. Northcutt. Bus bandwidth management in a high resolution video workstation. In *Proceedings 3rd International Workshop on Network and Operating System Support for Digital Audio and Video* La Jolla CA, pages 274–288, November 1992. published as *Lecture Notes in Computer Science 712*, P. Venkat Rangan (ed.), Springer-Verlag, Berlin.