In IEEE Communications Magazine, Vol. 30, No. 7, July 1992, pages 56-65

Designing an On-Demand Multimedia Service

P. Venkat Rangan, Harrick M. Vin, and Srinivas Ramanathan

Multimedia Laboratory
Department of Computer Science and Engineering
University of California at San Diego
La Jolla, CA 92093-0114

E-mail: venkat%cs@ucsd.edu, Telephone: (619) 534-5419, Fax: (619) 534-7029

Abstract

Future advances in networking coupled with rapid advances in storage technologies will make it feasible to build a multimedia on-demand server capable of providing services similar to those of a neighborhood videotape rental store on a metropolitan-area network. In this paper, we study various admission control policies that permit such a multimedia server to satisfy multiple subscribers simultaneously without violating any of their continuous media playback requirements. We propose a quality proportional policy that retrieves media blocks at a rate proportional on an average to the playback rates of media streams, but uses a staggered toggling technique by which successive numbers of media blocks retrieved are fine-tuned individually to admit and service an optimal number of subscribers simultaneously. This policy permits dynamic additions and deletions of requests in a transparent manner (i.e., without causing discontinuity in the retrieval of any of the existing requests). Performance evaluation shows that the quality proportional policy is an order of magnitude more scalable compared to straightforward admission control policies such as servicing one subscriber per disk head and round robin servicing of subscribers. In order to ensure synchronous retrieval of multimedia objects requested by each of the admitted subscriber, we propose a feedback technique in which a multimedia server uses light-weight messages called *feedback units* generated by display devices and transmitted back to the server to detect asynchronies during playback. We study various resynchronization policies such as, aggressive, conservative, and probabilistic, and compare their performance for video/audio playback. Whereas aggressive policies perform best only at lower playback rates, conservative policies perform best only at higher playback rates. In contrast, probabilistic policies perform uniformly well at all playback rates. The policies and algorithms for admission control and synchronous retrieval presented in this paper form the basis for a prototype multimedia server being developed at the UCSD Multimedia Laboratory.

Introduction

Future advances in networking will make it feasible for digital computer networks to support multimedia transmission. Coupled with the rapid advances in storage technologies, they can be used to build a multimedia on-demand service over metropolitan area networks such as B-ISDN, that are expected to permeate residential and commercial premises in a manner similar to existing cable TV or telephone networks [21]. The design of a high-performance storage server capable of servicing a large number of multimedia on-demand retrieval requests from multiple users simultaneously is the subject matter of this paper.

A multimedia on-demand storage server, which we will refer to as a *Multimedia Server* in the rest of this paper, provides services similar to those of a neighborhood videotape rental store. It digitally stores multimedia information such as entertainment movies, educational documentaries, advertisements, etc., on a large array of extremely high-capacity storage devices such as optical or magnetic disks that are random accessible with a short seek time, and are permanently on-line. The multimedia server is connected to audiophones and videophones (both of which we will, henceforth, refer to as *mediaphones*) belonging to subscribers via a high-speed metropolitan area networks (see Figure 1). Subscribers can select multimedia objects through a variety of indices such as subject titles, and request their retrieval for real-time playbackq. The multimedia server, if it has the necessary resources, satisfies the subscriber requests by transmitting chosen media segments over high-speed networks to their respective mediaphones. The retrieval can be interactive, in the sense that subscribers can stop, pause, resume, and even record[1] and edit the media information if they have permissions to do so.

The above architectural vision of a multimedia server is feasible within the next several years (rather than decades). To see why, consider the storage and transmission capacities required for a multimedia server. Assuming real-time motion video to require a data rate of about 0.5 Mbytes/s, a 100 minute long movie requires 3 Gbytes. Storage of 1000 such videos requires a capacity of 3 terabytes. In comparison, capacities of currently available disks are about 5 GBytes, and are expected to increase to 25 Gbytes in a few years. Thus, with an array of 120 disks, a multimedia server can store 1000 popular movies simultaneously. As for the transmission capacities, fiber-optic networks offering gigabyte/sec bandwidths are already in place, and those offering terabyte/sec bandwidths are conjectured to be only a few years away. However, a key deciding factor for the feasibility of such a multimedia server is its economic viability. Assuming costs of about $ 4,000 per disk, the total expected installation cost of a multimedia server will be $ 0.48 million, which when amortized over 1000 subscribers is about $ 480 per subscriber, making it a viable alternative to owning a VCR.

The development of a multimedia on-demand service is guided by two distinguishing requirements of media playback:

[1] Throughout this paper, we will use the terms playback and retrieval, as well as recording and storage synonymously.

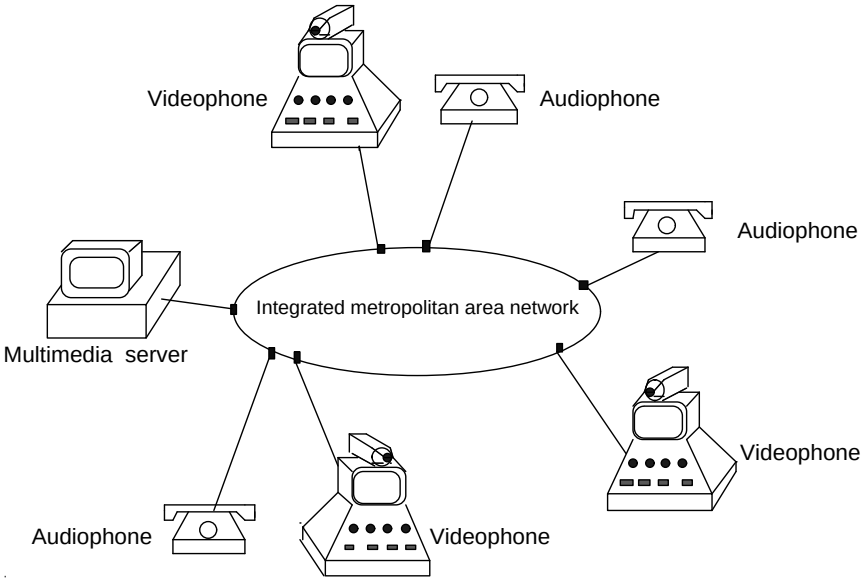

Figure 1: Configuration of a multimedia on-demand service

- *Time continuum of each media stream*: A media stream consists of a sequence of media quanta such as video frames or audio samples, which convey meaning only when presented continuously in time (this is unlike a textual object for instance, in which spatial continuity is sufficient). A multimedia server must ensure that the playback of each media stream proceeds at its real-time rate. Whereas ensuring the continuity of an isolated media stream is relatively straight-forward (given that the server's data transfer rate is at least as high as the playback rate of a media stream), satisfying multiple subscribers without violating the playback rates of any of their requested media streams necessitates the employment of sophisticated admission control mechanisms.

- *Synchronization between media streams*: Playback of multiple media streams constituting a multimedia object should not only be continuous, but also temporally coordinated. In the presence of rate mismatches among mediaphones and jitter in network delays, media streams may fall out of synchrony during their playback, and explicit resynchronization mechanisms are essential.

Most of the multimedia services that are being built or proposed make use of analog transmission and storage of video and audio by employing cable-TV style distribution network and crossbar switches for routing [12, 19]. On the other hand, multimedia services that are indeed digital and integrated into distributed computing systems, have focused mainly on still images and/or audio [1, 24, 8, 16, 13, 5, 23]. A qualitative proposal for a digital video on-demand service is presented by Sincoskie in [21]. However, a comprehensive quantitative study of optimal techniques for designing high-performance multi-user multimedia on-demand information servers has not received much attention.

In the recent past, many research projects have begun investigating storage issues in digital multimedia systems. The Cambridge Pandora project [10] and Matsushita's Real Time Storage System [14]

have begun investigating low level storage mechanisms for digital video. Anderson et al [2] and Gammell et al [7] have described file system designs for supporting multiple audio channel playback, and have proposed techniques for providing hard performance guarantees. A model for the design of a file system for storing real-time video and audio streams individually on magnetic disks have been presented by Rangan and Vin[20]. However, admission control policies for continuity-guaranteed servicing of multiple subscribers simultaneously has not received much attention.

The problem of inter-media synchronization has begun to receive attention only recently. Steinmetz [22] and Little et al. [11] have presented models for formally describing synchronization requirements among media streams. Nicoloau [15] proposes a two-level synchronization scheme in which synchronization requirements can be specified at a logical data level and implemented at a physical data level. Anderson et al. [3] describe algorithms for recovering from loss of synchrony among interrupt-driven media I/O devices, and is mainly applicable to single-site multimedia workstations. Network protocols for media synchronization have been presented by Escobar et al. [6]. These protocols, which can adapt to dynamic changes in network delays, assume the presence of globally sychronized clocks at all times. However, mediaphones directly connected to integrated networks may lack the sophistication to run elaborate clock synchronization protocols. Furthermore, these mediaphones may belong to different organizations, which may not want to synchronize their clocks. As a result, globally synchronized clocks may not exist in such environments. Sometimes, media streams recorded at different times may be required to be synchronized during playback, e.g., audio dubbing. Since there may not be any commonality in time of existence of the recording devices, there may be mismatches in recording rates. In such a case, media synchronization cannot be enforced merely by synchronizing the clocks of the mediaphones used for playback. Protocols for synchronous multimedia retrieval over integrated networks, in the face of non-deterministic variations in network delays and absence of globally synchronized clocks require immediate attention.

In this article, we formulate the problem of maintaining continuity of playback of each media stream in the presence of multiple subscriber requests, and present admission control algorithms that permit a multimedia server to satisfy the maximum number of subscribers simultaneously. In order to guarantee synchronous playback of media streams transmitted by the mulitmedia server to subscribers over metropolitan area networks, we present a feedback technique in which a multimedia server uses light-weight messages called *feedback units*, transmitted back to it by subscribers' mediaphones, to detect asynchronies among them, and to steer them back to synchrony thereafter.

1 System Architecture

The system architecture of a multimedia on-demand service comprises of a multimedia server connected to subscribers' mediaphones via a metropolitan area network (see Figure 2). Whereas the multimedia server is composed of a large system of randomly accessible storage disks (magnetic or optical),

Symbol	*Explanation*	*Unit*
η_{ms}	Granularity of media storage	media units/block
s_{mu}	Size of a media unit	bits/media unit
l_{ds}	Scattering parameter	sec
$\mathcal{R}_{dr}$	Disk data retrieval rate	bits/sec
θ	Nominal period of media at mediaphones	sec
ρ	Fractional drift in playback period at mediaphones	fraction
$\mathcal{R}_{pl}$	$\frac{1}{\theta*(1-\rho)}$, Fastest rate of media playback at mediaphones	media units/sec
Δ_{min}	Minimum network delay of a media unit	sec
Δ_{max}	Maximum network delay of a media unit	sec
$p(n)$	Playback initiation time of media unit n at a mediaphone	sec
$\mathcal{A}$	Maximum tolerable asynchrony	media units

Table 1: Symbols used in this article. A *media unit* represents a frame for video, and a sample for audio.

subscribers' mediaphones are simple media capture and display subsystems that are connected directly (as opposed to via a host computer) to the network. The architecture of each of these subsystems is elaborated next.

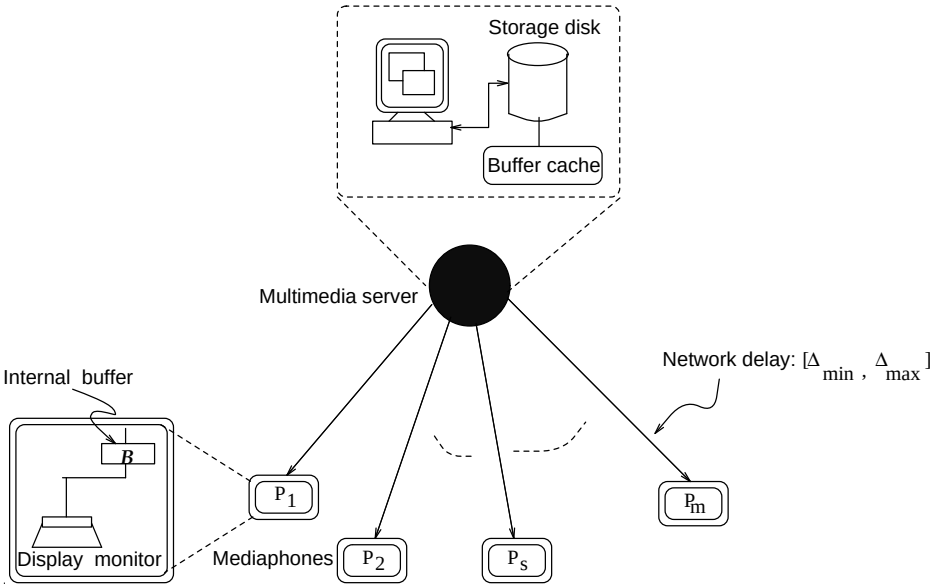

Figure 2: System architecture of a multimedia on-demand service

1.1 Storage Server Subsystem

The multimedia server organizes the storage of each media stream on the disk in terms of blocks. Most existing storage server architectures resort to random allocation of blocks on disk, resulting in unconstrained delay between retrievals of successive media blocks of a media stream. Alternatively, contiguous allocation of blocks of a media stream on disk can guarantee timely retrieval, but is fraught

with inherent problems of fragmentation, and can entail enormous copying overheads during insertions and deletions. Constrained block allocation, in which successive blocks of a media stream are placed on disk such that the separation between them does not exceed the playback duration of a media block, can guarantee timely retrieval without entailing the disadvantages of contiguous allocation scheme. Formally, using the symbols defined in Table 1, in which the term *granularity* denotes the size of a block and the term *scattering parameter* denotes the separation between consecutive blocks, it can be shown that, in constrained block allocation:

$$l_{ds} + \frac{\eta_{ms} * s_{mu}}{\mathcal{R}_{dr}} \leq \frac{\eta_{ms}}{\mathcal{R}_{pl}} \tag{1}$$

where, l_{ds} denotes the time to scan from the end of one media block to the beginning of the next media block of a media stream, $\frac{\eta_{ms} * s_{mu}}{\mathcal{R}_{dr}}$ denotes the time to retrieve a media block from the disk, and $\frac{\eta_{ms}}{\mathcal{R}_{pl}}$ denotes the playback duration of a media block. The multimedia server computes the relative values of granularity (η_{ms}) and scattering parameters (l_{ds}) for each media stream so as to satisfy Equation (1).

While servicing multiple subscribers, a multimedia server retrieves media blocks from disk, and transmits them over a network to subscribers' mediaphones for playback. A request for real-time playback of a media stream across a metropolitan area network may need to be admitted based on the availability of resources both at the network and at the multimedia server before the service can be rendered. In this article, we develop algorithms for admission control at the multimedia server. However, the disk schedules derived by the server admission control algorithms may not exactly match the transmission schedules obtained from network-specific admission control algorithms [9]. Differences in the disk access and network transmission schedules are smoothened out by buffering media blocks in an internal cache at the server prior to transmission on the network.

1.2 Network and Display Subsystem

The multimedia server and subscribers' mediaphones are interconnected by an integrated broadband metropolitan area network, such as B-ISDN, that may introduce a delay bounded between $[\Delta_{min}, \Delta_{max}]$ for each media unit.

The display subsystem of the multimedia on-demand service consists of mediaphones P_1, P_2,..., P_m, which are assumed to be simple devices capable of digitizing and transmitting, or receiving and playing back media units, but lacking the sophistication to run elaborate time rate synchronization protocols. Hence, the mediaphones may have mismatches in rates of recording and playback. We assume that the nominal period of recording and playback of a media unit at a mediaphone is θ, and the maximum fractional drift from the nominal period is bounded by ρ. These drifts are small enough for us to neglect higher powers of ρ and thereby approximate $\frac{1}{1-\rho}$ to $(1+\rho)$ and $\frac{1}{1+\rho}$ to $(1-\rho)$. As a result, the actual period $\theta(n)$ of a media unit n may take any value between $\theta * (1-\rho)$ and $\theta * (1+\rho)$.

2 Admission Control

A multimedia server may be required to service multiple subscriber requests simultaneously. In the best scenario, all the subscribers request the retrieval of the same media stream and the multimedia server only needs to retrieve the media stream once from the disk and then multicast it to all the subscribers. However, more often than not, different subscribers may request retrieval of different media streams; even when the same media stream is being requested by multiple subscribers, there may be phase differences among their requests (such as each subscriber retrieving a different portion of the stream at the same time). A simple scheme for guaranteeing real-time retrieval of each of the requests is to dedicate a disk head per request. However, this limits the total number of requests that can be serviced simultaneously to the number of disk heads, which may be orders of magnitude smaller than the number of subscribers needed to make the multimedia on-demand server economically viable.

In this section, we develop algorithms for maximizing the number of subscriber requests that can be serviced simultaneously, under the constraint that the retrieval of all of the requested media streams must proceed at their respective playback rates. In order to precisely formulate this requirement, let us suppose that a multimedia server has admitted r subscribers, each of whom is retrieving a media stream. The multimedia server multiplexes among all the r subscriber requests, retrieving a finite number of blocks k_i of each request $i \in [1, r]$, before switching to the next request. Each sequence of $k_1, k_2, ..., k_r$ retrievals constitutes a *service round*, and the multimedia server repeatedly executes service rounds until the completion of all the requests. Whereas the granularity and scattering parameters govern the rate of retrieval of successive media blocks of a media stream, switching from one media stream to another may entail an overhead of up to the maximum seek time to move the disk head from a block in one stream to a block of another stream (since the layout does not constrain the relative positions of different streams). Thus, the total time spent retrieving k_i blocks of i^{th} request in a service round consists of: (1) α: the overhead of switching from the $(i-1)$th request to the ith request, and then transferring the first block of the ith request, and (2) β: the time to transfer remaining $(k_i - 1)$ blocks of the ith request in this service round. If $\eta^1_{ms}, \eta^2_{ms}, ..., \eta^r_{ms}$ denote the granularities, $l^1_{ds}, l^2_{ds}, ..., l^r_{ds}$ denote the scattering parameters, and $\mathcal{R}^1_{pl}, \mathcal{R}^2_{pl}, ..., \mathcal{R}^r_{pl}$ denote the playback rates of the r streams being retrieved simultaneously, then the continuous retrieval of each of the requests can be guaranteed if and only if the total service time per round does not exceed the minimum of the playback durations of all the requests [20]. That is,

$$\sum_{i=1}^{r} (\underbrace{(l^{max}_{seek} + \frac{\eta^i_{ms} * s^i_{mu}}{\mathcal{R}_{dr}})}_{\alpha} + \underbrace{(k_i - 1) * (l^i_{ds} + \frac{\eta^i_{ms} * s^i_{mu}}{\mathcal{R}_{dr}})}_{\beta}) \leq \min_{i \in [1,r]} (k_i * \frac{\eta^i_{ms}}{\mathcal{R}^i_{pl}}) \quad (2)$$

The multimedia server can service all the r requests simultaneously if and only if $k_1, k_2, ..., k_r$ can be determined such that Equation (2) is satisfied. Since this formulation contains r variables and only one equation, determination of the values of $k_1, k_2, ..., k_r$ requires additional policies.

The simplest policy is to assign equal values to $k_1, k_2, ...,$ and k_r, yielding what is generally referred to as a *round robin* servicing algorithm [20]. Formally, if $k_1 = k_2 = \cdots = k_r = k$, Equation 2 yields the maximum number of subscriber requests that can be serviced in a round robin algorithm to be:

$$r^c_{max} = \frac{\eta^{avg}_{ms}}{\mathcal{R}^{max}_{pl} * (l^{avg}_{ds} + \frac{\eta^{avg}_{ms} * s^{avg}_{mu}}{\mathcal{R}_{dr}})} \tag{3}$$

Clearly, the number of subscriber requests that can be serviced by the round robin algorithm is limited by the request with maximum playback rate. Furthermore, the request with the maximum playback rate will have retrieved exactly the number of data blocks it needs for the duration of a service round, whereas other requests whose playback rates are smaller will have retrieved more data blocks than they need in each service round. Consequently, by reducing the number of data blocks retrieved per service round for such subscriber requests, it may be possible to accommodate larger number of subscribers.

At the other extreme is an *exhaustive* algorithm, which initially assigns the minimum value to each k_i, i.e., $k_1 = k_2 = \cdots = k_r = 1$, and then selectively increments the values of each k_i until the continuous retrieval equation (Equation (2)) is satisfied. The values of $k_1, k_2, ..., k_r$ thus obtained can be shown to be minimal [4], thereby guaranteeing that minimal time is spent on each subscriber and maximum number of subscribers are serviced during a service round. However, the computational overhead of such an algorithm can be prohibitive.

We propose a *Quality Proportional Multi-subscriber Servicing (QPMS)* algorithm, in which the number of blocks retrieved during each service round for each subscriber request is proportional to its playback rate. Thus, if k is the proportionality constant, we get, $k_1 = k * \mathcal{R}^1_{pl}$, $k_2 = k * \mathcal{R}^2_{pl}$, ..., $k_r = k * \mathcal{R}^r_{pl}$. Substituting the values of k_i in Equation (2) yields:

$$\sum_{i=1}^{r}((l^{max}_{seek} + \frac{\eta^i_{ms} * s^i_{mu}}{\mathcal{R}_{dr}}) + (k * \mathcal{R}^i_{pl} - 1) * (l^i_{ds} + \frac{\eta^i_{ms} * s^i_{mu}}{\mathcal{R}_{dr}})) \leq k * \min_{i \in [1,r]} \eta^i_{ms} \tag{4}$$

The variation of k with respect to r, as defined by Equation (4), is depicted in Figure 3. Furthermore, we can derive the maximum number of requests r^p_{max} that can be serviced simultaneously from Equation (4), and is given by:

$$r^p_{max} \leq \lfloor \frac{\eta^{avg}_{ms}}{\mathcal{R}^{avg}_{pl} * (l^{avg}_{ds} + \frac{\eta^{avg}_{ms} * s^{avg}_{mu}}{\mathcal{R}_{dr}})} \rfloor \tag{5}$$

Given the number of subscriber requests $r \leq r^p_{max}$, Equation (4) can also be used to determine k, from which we can obtain the values of k_1, k_2, ..., k_r. However, the values of k_i so obtained may not be integral. Since playback of media streams proceed in terms of quanta such as video frames, if each block is assumed to contain a quantum, then retrieval of a fraction of a block cannot be used for playback, causing the playback to starve until the remaining fraction arrives, possibly in the next service round. To avert such situations, the number of blocks retrieved in each service round must be integral for all the requests, to accomplish which we have proposed a staggered toggling technique in [25]. In this technique,

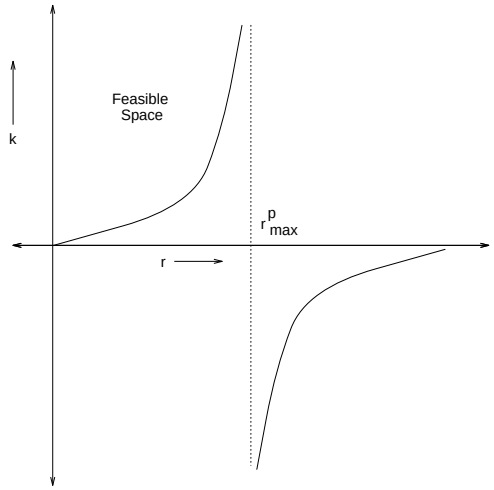

Figure 3: Variation in k with respect to the number of subscriber requests (r). For all values of $r > r^p_{max}$, Equation (4) yields $k < 0$, indicating that the multimedia server cannot service more than r^p_{max} requests simultaneously.

the number of blocks retrieved for each subscriber request i during each round toggles between $\lfloor k_i \rfloor$ and $\lceil k_i \rceil$, so that on an average, the retrieval rate is k_i blocks per service round. Furthermore, the toggling for various subscriber requests is staggered: the toggling up for one subscriber is matched by the toggling down for another subscriber, so that, over all, there is no net increase in the service time of any round and hence, no violation of continuity requirements. The subscribers for whom the number of blocks toggles up are exactly those who may face an impending starvation owing to an earlier toggling down in a preceding service round. This technique is shown to be provably correct, i.e., it satisfies the continuity requirements of each media stream in each service round [25].

It can be seen from Equation (5) that the QPMS algorithm can admit a much larger number of subscriber requests compared to the round robin algorithm, if there is a large variation in the playback rates of media streams requested by subscribers. In fact, it can be shown that the QPMS algorithm is provably optimal, that is, given any set of r subscriber requests a priori, if there is a set of $k'_1, k'_2, ..., k'_r$ satisfying the continuous retrieval equation (Equation (2)), then the QPMS algorithm is guaranteed to yield non-negative integers $k_1, k_2, ..., k_r$ that also satisfy Equation (2) [25].

If the total number of subscriber requests that need to be admitted are not known a priori, but may change dynamically, then continuity should not only be maintained during steady state, but also during transitions accompanying dynamic admission of new subscribers. To see why, suppose that a multimedia server receives a new request while servicing a set of r requests. If $r + 1 \leq r^p_{max}$, then the multimedia server can compute the new value of k, namely k^{new}, necessary for satisfying $(r+1)$ requests. If $k^{new} = k^{old}$ (where, k^{old} is the value of k using which the multimedia server has been servicing the existing r requests), then the multimedia server can immediately admit the $(r+1)$th subscriber request. However, if $k^{new} \neq k^{old}$, then $k^{new} > k^{old}$ (see Figure 3), and the multimedia server computes the new

values of k_i, namely $k_i^{new} = k^{new} * \mathcal{R}_{pl}^i$, for all $i \in [1, r+1]$, and begins transferring k_i^{new} blocks for each $i \in [1, r+1]$. However, during the first service round after admitting the $(r+1)^{th}$ subscriber, the number of blocks available to the ith subscriber for playback are those of the previous service round, namely k_i^{old}, the time for which may fall short of the time spent to transfer k_i^{new} blocks for each $i \in [1, r+1]$, leading to a violation of continuity requirement during this transitional service round. In order to avoid such a transitional discontinuity, the QPMS algorithm can be modified such that the time to transfer $(k_i + 1)$ blocks (i.e., $k * \mathcal{R}_{pl}^i$ instead of $k * \mathcal{R}_{pl}^i - 1$) is used in the left hand side of Equation (4), but the time to playback k_i blocks is used on its right hand side (i.e., no change in the right hand side). In such a case, it can be shown that $\forall r < r_{max}^p$, the difference between right hand side and the left hand side of Equation (4) increases with increase in k, as a consequence of which a smooth transition from k^{old} to k^{new} can be guaranteed in steps of 1, yielding a QPMS admission control algorithm that supports both transient and steady state continuity of media playback for subscribers.

3 Inter-Media Synchrony During Retrieval

During the simultaneous retrieval of multiple media streams (such as video and audio) constituting a multimedia object, it is required to not only maintain continuity of each media stream (using admission control mechanisms described earlier), but also preserve the temporal relationships that existed among those media streams at the time of their recording. When the rates of all the mediaphones are perfectly matched (because of their matching clock rates), and the network delays experienced by media units between a multimedia server and the mediaphones are deterministic, synchronization between media can be guaranteed if the multimedia server first instructs the mediaphones to begin their playback after preset delays following the reception of the first media unit, and then transmits media units to these mediaphones at a constant rate. However, in the face of network jitter and rate mismatches, the mediaphones may go out of synchrony soon after the commencement of playback. To see why, suppose that the multimedia server starts the transmission of media streams to a mediaphone at time T. Network jitter may cause the mediaphone to begin playback as early as $T + \Delta_{min}$ or as late as $T + \Delta_{max}$. Therefore, an initial asynchrony of at most $(\Delta_{max} - \Delta_{min})$ may exist between any pair of mediaphones. Owing to rate mismatches, the maximum fractional values of which can be $\pm\rho$, some mediaphones may playback at the fastest rate (with a period of $\theta * (1 - \rho)$), whereas some others may playback at the slowest rate (with a period of $\theta * (1 + \rho)$). Asynchrony between two mediaphones which playback at the fastest and slowest rates, respectively, will increase as playback progresses, reaching a maximum of:

$$\mathcal{A} = \lceil \frac{(\Delta_{max} - \Delta_{min}) + 2 * \theta * \rho * n_s}{\theta * (1 - \rho)} \rceil \tag{6}$$

where $\mathcal{A}$ represents the maximum asynchrony between mediaphones at the time the slowest mediaphone is playing back media unit n_s [17]. It may be observed from Equation (6) that the maximum asynchrony increases linearly with progression of media playback (i.e., n_s), and is unacceptable in practice. Hence, additional mechanisms are required for enforcing synchronization between media streams.

In order to facilitate synchronous retrieval, the multimedia server determines and notes down temporal relationships between media streams at the time of their storage. Since the media streams may be recorded at different mediaphones, possibly at different times, but may be required to be played back synchronously, it is convenient to represent the temporal relationships in the form of *Relative Time Stamps* (RTSs). We assume that at the time of recording, the multimedia server assigns (RTSs) to all media units of the media streams. Simultaneity of playback of a set of media units is indicated by equality of RTSs associated with them. (Techniques for assigning RTSs to media units are elaborated in [18].) The multimedia server, since it maintains information about the temporal relationships between media streams, is best suited to handle synchronization during retrieval with little additional overhead. In order to resynchronize mediaphones that have gone out of synchrony, the multimedia server may have to speed up some mediaphones and slow down some others, thereby causing breaks in continuity of their playback. The playback of at most one stream, which we will call as the master, can be spared from such discontinuities. While the master always plays back at its natural rate, all other streams, which take on the role of slaves, may be subject to skips and pauses in order to remain synchronized with the master. The choice of the master stream is dependent on the application. For example, when viewing a multimedia document, if smoothness of audio playback is of utmost importance, the audio stream serves as the master and drives the playback. The video stream, being the slave, may be subject to deletions or duplications of frames in order to synchronize its playback with that of audio. We now present a feedback mechanism in which the multimedia server uses feedback units, periodically transmitted back to it from master and slave mediaphones, to estimate playback asynchronies between those mediaphones, and steer them back to synchrony.

3.1 Feedback Mechanism for Detecting Asynchrony

Mediaphones generate feedback units concurrently with the playback of *selected* media units (but *not* necessarily with playback of every media unit), and transmit them back to the multimedia server (see Figure 4). Each feedback unit is a light-weight message containing only the RTS of the media unit that was concurrently played back at the time of the feedback unit's generation; hence, its transmission imposes little overhead on the network.

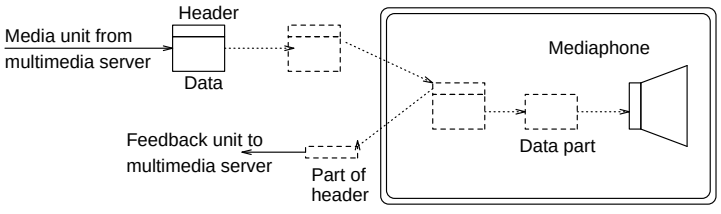

Figure 4: Feedback transmission at a mediaphone: The mediaphone transmits part of the headers of selected media units back to the multimedia server concurrently with the transfer of data parts of these media units to the display monitor for playback.

Using the bounds, Δ_{min} and Δ_{max} on network delays, the multimedia server, upon receiving a feedback unit f_{n_m} (corresponding to media unit n_m) at time $\tau(f_{n_m})$ from the master mediaphone P_m, can estimate the earliest and latest possible times, $p^e(n_m)$ and $p^l(n_m)$ at which playback of media unit n_m could have been initiated to be:

$$p^e(n_m) = \tau(f_{n_m}) - \Delta_{max} \tag{7}$$

$$p^l(n_m) = \tau(f_{n_m}) - \Delta_{min} \tag{8}$$

Media units are played back periodically, with the playback period varying between $\theta * (1 - \rho)$ and $\theta * (1 + \rho)$, where θ is the nominal playback period and ρ is its maximum fractional drift. Therefore, the multimedia server can predict the earliest and latest possible times of playback of all future media units $n_m + \nu_m$ as follows:

$$p^e(n_m + \nu_m) = p^e(n_m) + \nu_m * \theta * (1 - \rho) \tag{9}$$

$$p^l(n_m + \nu_m) = p^l(n_m) + \nu_m * \theta * (1 + \rho) \tag{10}$$

Similar computations can be carried out when the multimedia server receives a feedback unit f_{n_s} from a slave mediaphone. Using the above estimates, the multimedia server can determine media units that may be played back concurrently at the master and slave mediaphones (here, concurrently means closest in time, and it can be shown that the closest separation can be at most half the display period, which can be at most $\frac{\theta*(1+\rho)}{2}$). To see how, observe that the earliest and latest playback times of any media unit $n_m + \nu_m$ which may possibly be played back concurrently with media unit n_s should be such that (see Figure 5):

$$p^l(n_m + \nu_m) \geq p^e(n_s) - \frac{\theta * (1 + \rho)}{2}$$

and

$$p^e(n_m + \nu_m) \leq p^l(n_s) + \frac{\theta * (1 + \rho)}{2}$$

Substituting for $p^e(n_m + \nu_m)$ and $p^l(n_m + \nu_m)$ from Equations (9) and (10) in the above equations and solving for $n_m + \nu_m$, we have

$$\nu_m \geq \frac{(p^e(n_s) - p^l(n_m)) - \frac{\theta*(1+\rho)}{2}}{\theta * (1 + \rho)} \tag{11}$$

and

$$\nu_m \leq \frac{(p^l(n_s) - p^e(n_m)) + \frac{\theta*(1+\rho)}{2}}{\theta * (1 - \rho)} \tag{12}$$

Using the earliest and latest playback times of media units n_m and n_s, the multimedia server can thus determine a range of media units, $[n_m + \nu^e_m, n_m + \nu^l_m]$ which could possibly be played back concurrently with media unit n_s, where $n_m + \nu^e_m$ is the smallest unit satisfying Equation (11) and $n_m + \nu^l_m$ is the largest media unit satisfying Equation (12).

Playback is synchronous if media units that are being played back concurrently at the master and slave have the same RTS. On the other hand, any mismatch in RTSs of media units being played

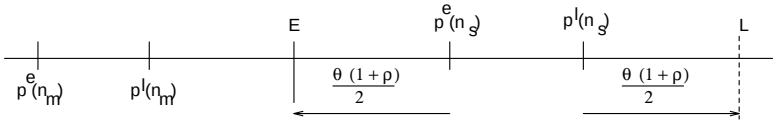

Figure 5: Determination of a media unit $(n_m + \nu_m)$ that could have been played back concurrently with n_s; The playback interval of $(n_m + \nu_m)$ must be within the interval $[E, L]$, where $E = p^e(n_s) - \frac{\theta*(1+\rho)}{2}$ and $L = p^l(n_s) + \frac{\theta*(1+\rho)}{2}$.

back concurrently at the master and slave is symptomatic of asynchrony between the master and slave mediaphones. In such a case, the multimedia server may have to take corrective action to bring the slave back to synchrony with the master. Since the multimedia server determines a range of master media units (as opposed to a unique media unit), $[n_m + \nu_m^e, n_m + \nu_m^l]$ that could be played back concurrently with a media unit n_s of a slave, there may be some situations in which asynchrony could possibly exist and some others in which asynchrony is guaranteed to exist. Between these two extremes there are a whole range of situations in which the multimedia server can detect and correct asynchrony. The policies for deciding when to trigger resynchronization can range from deterministic ones, which take into account only the bounds on network delays and rate mismatches, to probabilistic ones, which resynchronize based on statistical distributions of network delays and playback rates. These policies are elaborated below:

- *Conservative* policies trigger remedial resynchronization only when asynchrony is guaranteed to exist, i.e. when $\mathsf{RTS}(n_s) \notin [\mathsf{RTS}(n_m + \nu_m^e), \mathsf{RTS}(n_m + \nu_m^l)]$. The slave lags or leads the master depending on whether $\mathsf{RTS}(n_s) < \mathsf{RTS}(n_m + \nu_m^e)$ or $\mathsf{RTS}(n_s) > \mathsf{RTS}(n_m + \nu_m^l)$, respectively.

- *Aggressive* policies, on the other hand, trigger resynchronization even if there is a slight chance of asynchrony i.e., when there exists $n'_m \in [n_m + \nu_m^e, n_m + \nu_m^l]$, such that $\mathsf{RTS}(n'_m) \neq \mathsf{RTS}(n_s)$. The slave lags or leads the master depending on whether $\mathsf{RTS}(n_s) < \mathsf{RTS}(n'_m)$ or $\mathsf{RTS}(n_s) > \mathsf{RTS}(n'_m)$, respectively.

- *Probabilistic* policies use statistical distributions of network delays and playback periods to estimate the probability distributions of playback times of media units from the observed arrival times of feedback units. The probability distribution of asynchrony, which is the difference between playback times of media units at the master and the slave, can then be computed as a convolution of the individual probability distributions of the playback times. Resynchronization is triggered whenever the probability of asynchrony exceeds a threshold value, which can, of course, be application-specific.

In order to resynchronize, the multimedia server duplicates or deletes an appropriate number of media units in its transmission queues destined for a leading or a lagging slave device, respectively (the slave devices themselves can take this action, but in general, they may not be sophisticated enough to handle such control functions). Whereas the conservative policy represents a "no risk" approach and

initiates resynchronization only when it is guaranteed not to accentuate asynchrony, the aggressive policy is a "high risk" approach, whose potential advantage is detection of asynchrony at the earliest possible chance. The probabilistic policies, since they take into account the delay characteristics of networks, can be potentially more effective than either purely conservative or purely aggressive policies.

3.2 Minimizing Feedback Transmission

The multimedia server, since it may not be able to determine exactly the playback times of media units at the master and slave mediaphones, may be unable to completely resynchronize the master and slave mediaphones. The residual asynchrony, γ that may exist between the master and slave mediaphones immediately following a resynchronization, is dependent on the precision of the multimedia server's estimates of playback times at the mediaphones and on the resynchronization policy being employed. Starting from this residual value, asynchrony of a slave may again increase with progression of playback, owing to network delay jitter and variations in playback periods. Such an increase may be allowed upto a maximum tolerable limit, by when the multimedia server must have received the next feedback unit from the mediaphone and initiated the next resynchronization. In order to minimize additional overheads due to feedback transmission, it is desirable to maximize the interval between successive resynchronizations; the maximum allowable interval is directly determined by the difference between the maximum tolerable asynchrony and the residual asynchrony immediately following resynchronization, γ. The minimum rate at which feedbacks can be transmitted, so as to still restrict the maximum possible asynchrony to within tolerable limit is expressed as the *minimum feedback ratio*, which is defined as the ratio of number of feedback units transmitted to media units played back at a mediaphone. Given the maximum tolerable asynchrony $\mathcal{A}$ (that can be specified by an application based on human perception needs), the minimum feedback ratio, $\mathcal{F}$ can be computed to be [17]:

$$\mathcal{F} = \lfloor \frac{1}{\frac{\mathcal{A}*\theta*(1-\rho)^2-\theta*(1-\rho)-3*\Delta_{max}+\Delta_{min}}{\theta*(1+\rho)} - \frac{\gamma*(1-\rho)}{2*\rho}} \rfloor \tag{13}$$

4 Buffering Requirements

Whereas the arrival of a feedback unit from a mediaphone enables a multimedia server to estimate the playback instant of the feedback unit's corresponding media unit to within an accuracy of delay jitter ($\Delta_{max} - \Delta_{min}$), the accuracy of prediction of playback instants of later media units decreases by twice the drift for each successive media unit (see Equations 7-10). In order to guarantee starvation-free playback at a mediaphone, the multimedia server must transmit each media unit so as to ensure its availability at a mediaphone prior to its earliest predicted playback time. Since network delays can be as high as Δ_{max}, a media unit must be transmitted by the multimedia server at least Δ_{max} prior to that unit's earliest predicted playback time. However, if the actual network delay experienced by media units is less than Δ_{max}, or if the actual playback instants of media units are later than their earliest

predicted playback times, media units will be received at their mediaphones earlier than their scheduled times of playback, and will have to be buffered. We will now compute the maximum buffering that is needed at display devices.

The two factors that contribute to accumulation of media units in buffers at mediaphones are: (1) delay jitter, and (2) fractional drift in playback periods. Buffering due to delay jitter attains a maximum value when media units experience minimum delays (Δ_{min}). This component of buffering can be shown to be [17]: $\frac{2*(\Delta_{max}-\Delta_{min})}{\theta*(1+\rho)}$. Buffering due to drift in playback period of media units reaches a maximum value when the playback period of each media unit is largest (i.e., $\theta*(1+\rho)$) and increases by $\frac{2*\rho}{1+\rho}$ for each media unit transmitted after the arrival of the most recent feedback unit. The arrival of a subsequent feedback unit from a mediaphone enables the multimedia server to more accurately revise (to within delay jitter $\Delta_{max}-\Delta_{min}$) its estimates of the playback instant of the very next media unit at that mediaphone, allowing the multimedia server to delay the transmission time of that very next media unit until media units accumulated so far at that mediaphone's buffers are consumed. The total maximum buffering needed is governed by the feedback ratio $\mathcal{F}$, and can be shown to be given by [17]:

$$\mathcal{B} = \lceil \frac{2*(\Delta_{max}-\Delta_{min})+4*\Delta_{max}*\rho}{\theta*(1-\rho^2)} + \frac{2*\rho}{\mathcal{F}*(1-\rho)} \rceil \quad (14)$$

The available buffering must equal at least the maximum needed buffering $\mathcal{B}$, so as to avert anomalies such as buffer overruns (which lead to media losses).

5 Experience and Performance Evaluation

In order to experimentally evaluate the performance of policies and algorithms for admission control and synchronous retrieval, we are developing a prototype multimedia server at the UCSD Multimedia laboratory (see Figure 6). The multimedia server is being implemented on a 486-PC with multiple gigabytes of storage. Media streams are recorded and played back at multimedia stations, each of which consists of a computing workstation, an audiophone, and a videophone. Audiophones digitize audio signals at 8 Kilobytes per second and the videophones digitize and compresses motion video at real-time rates.

We have carried out preliminary performance estimations of the admission control algorithm and the resynchronization policies. We have assumed a multimedia server containing an array of 120 disks, each with a data transfer bandwidth of 1 Gigabyte per second. Network delays of media units are assumed to be exponentially distributed between 50 and 60 ms (values beyond 60 ms were approximated to 60 ms). The maximum fractional drift in playback periods at mediaphones is assumed to be $\rho = 10^{-4}$.

At an average playback rate of 22.5 frames per second, which is the case for a pool of requests with half the requests are at a playback rate of 30 frames per second and the other half at a rate of 15 frames per second, the maximum number of simultaneous subscriber requests that can be admitted by a multimedia server were found to be 10666 and 8000 using the QPMS and round robin algorithms,

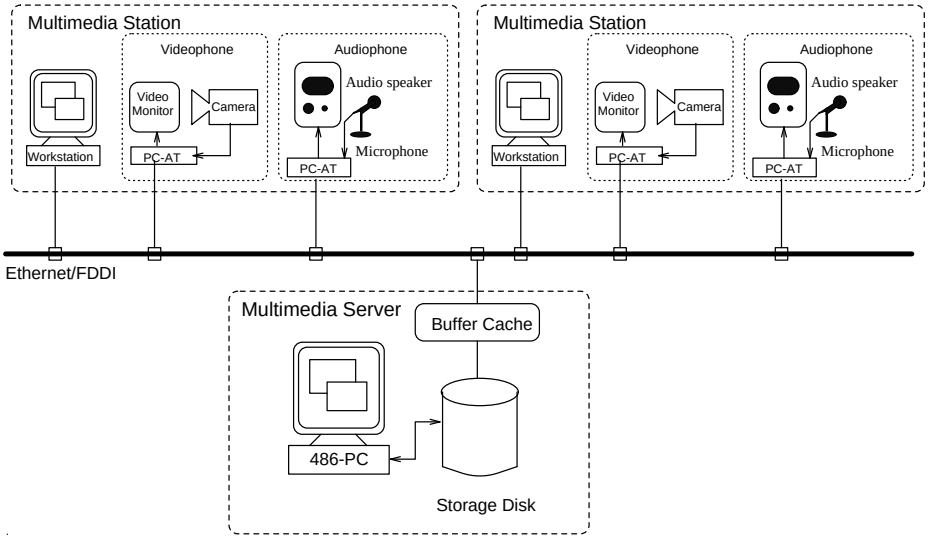

Figure 6: Hardware configuration of our multimedia server prototype

respectively. These values are two orders of magnitude greater than 120 subscribers supported by straightforward multiplexing techniques such as one subscriber per disk head, thereby demonstrating the scalability of the QPMS algorithm.

A comparison of the effectiveness of conservative, aggressive and probabilistic resynchronization policies for audio and video playback with audio as the master stream reveals that the aggressive policy performs best only at higher values of the tolerable asynchrony. This is because, lower the tolerable asynchrony, greater is the ratio of network jitter to the tolerable asynchrony and hence, greater are the number of instances in which the network jitter may, by itself falsely trigger resynchronization. In such cases, the aggressive policy over reacts, and introduces excessive deletions and duplications (of media units) that tend to have an oscillatory impact on the asynchrony, thereby increasing the average asynchrony. The conservative policy, unlike the highly reactive aggressive policy, reacts only if it can conclude with certainty that media units with the same RTS are not being played back concurrently. Hence, it does not trigger false resynchronizations like the aggressive policy and consequently performs better than the aggressive policy at lower asynchrony. At higher asynchronies, however, the conservative policy, since it delays resynchronization until an asynchrony equal to the maximum tolerable value is guaranteed to exist, entails larger average asynchrony than the aggressive policy. In contrast, the probabilistic policy adapts best to changing values of tolerable asynchrony and exhibits moderate reactivity at all asynchronies.

6 Concluding Remarks

It is envisaged that future advances in networking and storage technologies will make it feasible to build multimedia on-demand servers providing services similar to those of a neighborhood videotape

rental store. We have investigated the problems of providing continuous and synchronous access to such multimedia on-demand services. We have presented various admission control policies, ranging from a straightforward round robin policy to an optimal quality proportional policy, that can be employed by a multimedia server to service multiple subscribers simultaneously, the same time guaranteeing that each of their retrieval requests proceeds at its real-time rate. In order to ensure synchronous retrieval of multiple media streams to the admitted subscribers, we have proposed a feedback technique in which a multimedia server uses light-weight messages called *feedback units* generated by mediaphones and transmitted back to it to detect asynchronies during playback. We have presented various resynchronization policies such as, aggressive, conservative and probabilistic. The policies and algorithms for admission control and synchronous retrieval presented in this paper form the basis of a prototype multimedia server being developed at the UCSD Multimedia Laboratory.

References

[1] C. Abbott. Efficient Editing of Digital Sound on Disk. *Journal of Audio Engineering*, 32(6):394–402, June 1984.

[2] D. Anderson, Y. Osawa, and R. Govindan. Real-time Disk Storage and Retrieval of Digital Audio and Video. *To appear in the ACM Transactions on Computer Systems.*

[3] D. P. Anderson and George Homsy. A Continuous Media I/O Server and Its Synchronization Mechanism. *IEEE Computer, Special Issue on Multimedia Information Systems*, 24(10):51–57, October 1991.

[4] D. P. Anderson, Yoshitomo Osawa, and Ramesh Govindan. Real-Time Disk Storage and Retrieval of Digital Audio and Video. *Technical Report No. UCB/CSD 91/646*, Computer Science Division, University of California, Berkeley, California 94720, August 8, 1991.

[5] S. Angebranndt, R. L. Hyde, D. H. Luong, N. Siravara, and C. Schmandt. Integrating Audio and Telephony in a Distributed Workstation Environment. In *Proceedings of Summer 1991 USENIX Conference, Nashville, TN*, pages 419–436, June 10-14, 1991.

[6] J. Escobar, D. Deutsch, and C. Partridge. A Multi-Service Flow Synchronization Protocol. *BBN Systems and Technologies Division*, March 1991.

[7] J. Gemmell and S. Christodoulakis. Principles of Delay Sensitive Multimedia Data Storage and Retrieval. *ACM Transactions on Information Systems*, 10(1):51–90, 1992.

[8] S. Gibbs, D. Tsichritzis, A. Fitas, D. Konstantas, and Y. Yeorgaroudakis. Muse: A Multi-Media Filing System. *IEEE Software*, 4(2):4–15, March 1987.

[9] Zygmunt Haas. Adaptive Admission Congestion Control. *Computer Communication Review*, 21(5):58–76, 1991.

[10] A. Hopper. Pandora – an experimental system for multimedia applications. *ACM Operating Systems Review*, 24(2):19–34, April 1990.

[11] T.D.C. Little and A. Ghafoor. Synchronization and Storage Models for Multimedia Objects. *IEEE Journal on Selected Areas in Communications*, 8(3):413–427, April 1990.

[12] W. E. Mackay and G. Davenport. Virtual Video Editing in Interactive Multimedia Applications. *Communications of the ACM*, 32(7):802–810, July 1989.

[13] Sun Microsystems. Multimedia File System. *Software Release*, August 1989.

[14] Y. Mori. Multimedia Real-Time File System. Technical report, Matshushita Electric Industrial Co., February 1990.

[15] Cosmos Nicolaou. An Architecture for Real-Time Multimedia Communication System. *IEEE Journal on Selected Areas in Communication*, 8(3):391–400, April 1990.

[16] B.C. Ooi, A.D. Narasimhalu, K.Y. Wang, and I.F. Chang. Design of a Multi-Media File Server using Optical Disks for Office Applications. *IEEE Computer Society Office Automation Symposium, Gaithersburg, MD*, pages 157–163, April 1987.

[17] P. Venkat Rangan and Srinivas Ramanathan. Rate-Based Feedback Techniques for Continuity and Synchronization in Multimedia Retrieval over High-Speed Networks. *Technical Report No. CS92-230, Dept. of Computer Science and Engineering, University of California, San Diego*, February 1992.

[18] P. Venkat Rangan, Srinivas Ramanathan, Harrick M. Vin, and Thomas Kaeppner. Media Synchronization in Distributed Multimedia File Systems. In *Proceedings of Multimedia'92 - 4th IEEE COMSOC International Workshop on Multimedia Communications, Monterey, California*, April 1-4, 1992.

[19] P. Venkat Rangan and D. C. Swinehart. Software Architecture for Integration of Video Services in the Etherphone Environment. *IEEE Journal on Selected Areas in Communication*, 9(9):1395–1404, December 1991.

[20] P. Venkat Rangan and Harrick M. Vin. Designing File Systems for Digital Video and Audio. In *Proceedings of the 13th Symposium on Operating Systems Principles (SOSP'91), Operating Systems Review, Vol. 25, No. 5*, pages 81–94, October 1991.

[21] W. D. Sincoskie. System Architecture for a Large Scale Video on Demand Service. *Computer Networks and ISDN Systems, North-Holland*, 22:155–162, 1991.

[22] R. Steinmetz. Synchronization Properties in Multimedia Systems. *IEEE Journal on Selected Areas in Communications*, 8(3):401–412, April 1990.

[23] D.B. Terry and D.C. Swinehart. Managing Stored Voice in the Etherphone System. *ACM Transactions on Computer Systems*, 6(1):3–27, February 1988.

[24] R.H. Thomas, H.C. Forsdick, T.R. Crowley, R.W. Schaaf, R.S. Tomlinsin, V.M. Travers, and G.G. Robertson. Diamond: A Multimedia Message System Built on a Distributed Architecture. *Computer*, 18(12):65–78, December 1985.

[25] Harrick M. Vin and P. Venkat Rangan. Designing a Multi-User HDTV Storage Server. *To appear in the IEEE Journal on Selected Areas in Communications*, 11(1), January 1993.

Biography

P. Venkat Rangan directs the Multimedia Laboratory at the University of California, San Diego, where he is an Assistant Professor of Computer Science since 1989. He has been a visiting scientist at Xerox Palo Alto Research Center (PARC). He serves on the editorial boards of IEEE Network, Journal of Interactive Multimedia, Journal of Organizational Computing Systems, as the Program Chair for the 1992 International Workshop on Network and Operating System Support for Digital Video and Audio, and on the ACM steering committee on multimedia. Dr. Rangan earned a Ph.D. in the Computer Systems Research Group at the University of California, Berkeley, in 1989. Dr. Rangan has received numerous awards including the Powell Foundation and NCR Research Innovation Awards for establishing multimedia laboratory at UCSD, IBM doctoral fellowship for outstanding Ph.D. research at U.C. Berkeley, and the "President of India Gold Medal" for the best undergraduate academic record at Indian Institute of Technology, Madras, India, in 1984.

Harrick M. Vin is a doctoral candidate in the Department of Computer Science and Engineering at the University of California, San Diego. His research interests are in multimedia computer systems and ultra-high speed networking with emphasis on conferencing and storage architectures for digital video and audio. He is a recipient of an IBM Doctoral Fellowship and NCR Innovation Award. He received his B.Tech in Computer Science and Engineering from the Indian Institute of Technology, Bombay, India, in 1987 and his MS from Colorado State University in 1988.

Srinivas Ramanathan is a doctoral student in the Department of Computer Science and Engineering at the University of California, San Diego. His research interests are in multimedia communication and synchronization. He received his B.Tech in Chemical Engineering from Anna University, Madras, India, in 1988 and M.Tech in Computer Science and Engineering from the Indian Institute of Technology, Madras, India, in 1990.