

Techniques for multimedia synchronization in network file systems

P Venkat Rangan, Srinivas Ramanathan, Harrick M Vin and Thomas Kaepfner

One of the unique features that distinguishes digital multimedia from traditional computer data is the presence of multiple media streams, whose display must proceed in a mutually synchronized manner. The design of techniques for synchronization of multimedia data at the time of storage, and retrieval from network file servers is the subject matter of this paper. We present algorithms by which a file server can create a relative time system and synchronize media units transmitted by different sources on a network to construct a multimedia object. These algorithms stay robust in the absence of global clocks, presence of transmission jitter and generation rate mismatches. We develop a feedback technique using which the file server can detect asynchronies in display devices during retrieval of multimedia objects, and even restore synchrony by deleting or duplicating media units destined for asynchronous destinations. We then present strategies by which the file server can actually predict the time in future when the asynchrony of a device is expected to exceed the permitted bound, and take gradual preventive action to nullify the asynchrony in advance. These algorithms can be generalized to heterogeneous multimedia networks in which there may be variations in sizes of media units generated, differences in network locations of sources and destinations, etc. We are currently implementing these techniques in a digital multimedia on-demand storage server being developed at the UCSD Multimedia Laboratory.

Keywords: multimedia synchronization, multimedia networks, multimedia file systems

Future advances in networking and storage^{1,2} will make it feasible to design computer systems capable of offering services such as multimedia mail, news distribution, advertisement, and entertainment^{3,4}. Supporting such

Multimedia Laboratory, Department of Computer Science and Engineering, University of California at San Diego, La Jolla, CA 92093-0114, USA (e-mail: venkat%cs@ucsd.edu)

Paper received: 10 February 1992; revised paper received: 14 May 1992

services requires the ability to capture multimedia objects digitally at their sources, transmit the objects to network file servers for storage, and later retrieve them for display at various destinations on the network. One of the unique features that distinguishes digital multimedia from traditional computer data is the presence of *multiple media streams*, whose display must proceed in a mutually synchronized manner. The design of techniques for synchronization of multimedia data at the time of storage, and retrieval from network file servers is the subject matter of this paper.

A multimedia object may consist of many components, such as audio, video and text. Generally, the components of a multimedia object are separated at the input (i.e. they go through different digitizers), and during storage they arrive at a file server as separate streams on a network. Retrieval of a multimedia object must proceed so as to not only maintain continuity of playback of each of its media components, but also to preserve the temporal relationships among them, even though each of the components may be played back at a different I/O device on the network. In future integrated networks, I/O devices that digitize and packetize media data may be connected directly (as opposed to via a host computer) to the network, e.g. Etherphone⁵, ISDN Telephone, etc. (see *Figure 1*). Such devices may not have the sophistication to run complex protocols or time synchronization mechanisms. Hence, any synchronization mechanism that involves elaborate handshaking protocols requiring frequent exchange of control messages may not be feasible. On the other hand, direct use of media data (such as its length) for synchronization may not be possible due to variable rates of video compression and silence elimination of audio, after which video and audio data no longer have lengths proportional to their period.

In this paper we have addressed the problem of media synchronization in the absence of globally

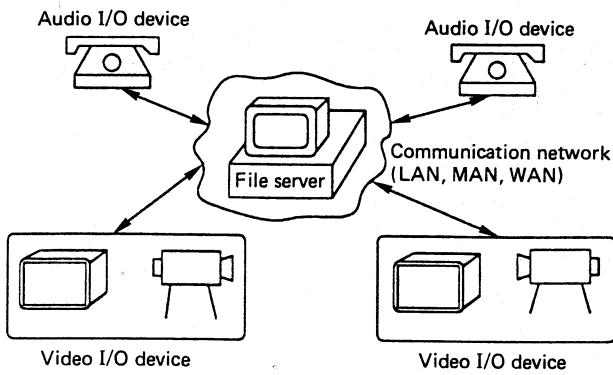


Figure 1 Communication network connecting a file server and media I/O devices

synchronized clocks but in the presence of jitter in transmission delays and digitization rates, and media losses on the network. We present algorithms by which, to facilitate synchronization, a file server can create a relative time system in which media units (from different sources on a network) whose generation times are within a bounded time window are assigned the same relative time stamp while they are being stored. The rate of advance of the relative time system is driven by a *master* media source that can be chosen on the basis of application (in the absence of an application-designated master, the file server itself serves as the master).

Synchronous retrieval is slightly more complex because media streams may be routed to different display devices on a network. These devices may have mismatches in playback rates, the network may have delay jitter, and the devices used for playback may be different from those used earlier for recording. We present techniques in which the file server uses feedback units periodically transmitted back to it by display devices to detect asynchronies between display devices*. The file server then restores synchrony by deleting or duplicating appropriate amounts of media data in the transmission queues destined for slow or fast devices, respectively. We develop algorithms that enable the file server to actually predict the time in future when the asynchrony of a device is expected to exceed the permitted bound, and take gradual preventive action in advance to remedy the asynchrony.

The rest of the paper is organized as follows. In the following section we introduce the concept of relative time system in multimedia network file servers. Algorithms for assigning relative time stamps to media data during storage are then presented, along with techniques for their synchronous retrieval on a multimedia network. These techniques are then generalized for heterogeneous multimedia networks. Finally we compare our work with other media synchronization efforts.

* Each feedback unit is a replica of a media unit, except for the data part which is null; hence its transmission is lightweight and imposes little overhead.

RELATIVE TIME SYSTEM IN MULTIMEDIA FILE STORAGE

In its most general form, a multimedia object being stored on a network file server may be constituted of media components generated by different sources located at different places on the network. The individual media components are continuously digitized, converted into units of media display (such as frames), and transmitted on the network to the file server. Furthermore, the media components may be generated at different times, but their display may be required to be simultaneous (e.g. audio dubbing in movies). Hence, it is convenient to place all the media units constituting a multimedia object on a *relative time scale*, with the media units that are at the beginning of the object placed at zero on the scale. The position of a media unit on the relative time scale defines its *relative time stamp* (RTS). A unit of the relative time scale should not exceed the duration of the smallest media unit, so that no two media units originating from a source are associated with the same relative time, thereby avoiding ambiguities during retrieval. The periods of media units generated by different sources may be different, and choosing their common factor (such as their greatest common divisor (*GCD*)) as the unit of relative time scale affords the advantage that the RTS will only take integer values.

Each media unit (such as a video frame or an audio sample) should be associated with an RTS. During retrieval, all those media units with the same RTS must be displayed simultaneously. RTSs of media units can be determined in a straightforward manner by a file server when all the media sources are synchronized. We do not assume globally synchronized clocks for the following reasons:

- clock synchronization requires sophisticated protocols for negotiation and agreement among media sources. We assume that the media sources can digitize, packetize and transmit data on the network, but may not have the sophistication required for running complex protocols;
- media sources may belong to different organizational domains (each of which may possess internal but not external clock synchronization) that may not want to synchronize all of their clocks across all the domains solely for the purpose of the application being considered;
- even if clocks are synchronized there may be non-deterministic delays before audio/video units can be collected from the internal buffers of media devices. These delays may introduce inaccuracies in determining generation and playback times of units.

In the absence of globally synchronized clocks, the RTS is to be based on a master media source. The first

media unit from the master starts the RTS, and successive media units increment it. In general, the master clock may be that of any of the sources on the network, and its choice is application dependent. Some of the criteria that can be used for selecting a master are the accuracy of clock, and the significance of media source, etc. In the absence of a master, the file server itself serves as the master.

All other sources except the master are called *slave* sources. In order to associate an RTS with a media unit generated and transmitted by a slave source, the file server tries to determine a master's media unit that is generated at about the same instant as the slave media unit, and then the RTS of such a master's media unit is assigned to that slave media unit. In the absence of global clocks, it may not be possible to exactly decide whether the generation times of a slave and a master media unit are the same. The first problem, therefore, when the file server receives media units from different sources, is to determine a set of media units that are generated within a tolerable window of asynchrony, so that, such media units are assigned the same RTS and can be displayed simultaneously during retrieval. Such a set of media units is called a *synchronization set*, and is formally defined as follows:

Synchronization set: media units n_m and n_s , generated by a master and slave sources, S_m and S_s , respectively, are said to belong to a synchronization set if and only if:

$$|(g_{\text{real}}(n_m) - g_{\text{start}}(S_m)) - (g_{\text{real}}(n_s) - g_{\text{start}}(S_s))| \leq \mathcal{A}$$

where $g_{\text{real}}(n_m)$ and $g_{\text{real}}(n_s)$ are the actual generation times of units n_m and n_s , $g_{\text{start}}(S_m)$ and $g_{\text{start}}(S_s)$ are the start times of media streams from sources S_m and S_s (this is to accommodate media objects composed of media streams generated at different times), and \mathcal{A} represents the tolerable window of asynchrony in time.

In the remainder of this paper, we will assume that generation times of all media units are offset by the start times of their respective media streams, and using the substitutions $g(n_m) = g_{\text{real}}(n_m) - g_{\text{start}}(S_m)$ and $g(n_s) = g_{\text{real}}(n_s) - g_{\text{start}}(S_s)$, we obtain the following definition of a synchronization set:

$$|g(n_m) - g(n_s)| \leq \mathcal{A} \tag{1}$$

It is desirable that the asynchrony \mathcal{A} and hence, the differences between generation times of media units belonging to a synchronization set be very small. Synchronization sets are guaranteed to exist for each media unit only if \mathcal{A} is at least half the period of the largest media unit. On the other hand, they are guaranteed to be unique for each media unit only if \mathcal{A} does not exceed half the period of the smallest media

unit. The choice of the tolerable window of asynchrony \mathcal{A} and the master media source (both of which are chosen at the time of storage of the media streams) together determine the RTSs assigned to media units.

We now develop techniques for determining the synchronization sets when the file server receives media streams from a master source and several slave sources on the network. We assume that the network communication delay from a source to the file server, which includes transmission times on the network and queuing delays at intermediate nodes, is bounded, and that the periods of all the media units are identical. In the fourth section we extend the techniques to general cases.

RTS ASSIGNMENT DURING STORAGE

Consider the storage of a multimedia object consisting of media streams originating from sources S_1, S_2, \dots, S_m (see Figure 2). Media units, which are generated at a constant rate with a nominal period of generation p , are transmitted to the file server over a network whose minimum and maximum communication delays are Δ_{min} and Δ_{max} , respectively. Let S_m be the master source on which the RTS is based, and S_s be a typical slave source. The file server tries to determine the RTS of a slave media unit as soon as it receives the media unit from the network. If τ_m and τ_s are the arrival times of media units n_m and n_s at the file server* from sources S_m and S_s , respectively, the earliest and latest possible generation times of n_m and n_s , denoted by $g_m^e(n_m)$, $g_m^l(n_m)$, $g_s^e(n_s)$, and $g_s^l(n_s)$, respectively, can be derived as follows:

$$g_m^e(n_m) = \tau_m - \Delta_{\text{max}} \tag{2}$$

$$g_m^l(n_m) = \tau_m - \Delta_{\text{min}} \tag{3}$$

$$g_s^e(n_s) = \tau_s - \Delta_{\text{max}} \tag{4}$$

$$g_s^l(n_s) = \tau_s - \Delta_{\text{min}} \tag{5}$$

Hence, measured on the file server's clock, the generation times $g_m(n_m)$ and $g_s(n_s)$ of media units n_m and n_s at

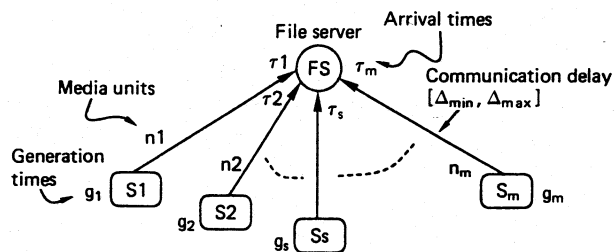


Figure 2 Recording of media components sources S_1, S_2, \dots, S_m and their transmission to a network file server FS for storage

*All times are measured on the file server's clock.

sources S_m and S_s belong to intervals $[g_m^e(n_m), g_m^l(n_m)]$ and $[g_s^e(n_s), g_s^l(n_s)]$, which are referred to as the *generation intervals* of n_m and n_s respectively. Since the exact generation times of media units are not known to the file server, it has to determine synchronization sets (and assign the same RTS to each of the media units that constitute a synchronization set) based on the above estimates of the generation intervals of media units transmitted by master and slave sources. This technique is formulated in the following theorem:

Theorem 1 *Media units n_m and n_s belong to a synchronization set if*

$$\delta g_{\max}(n_m, n_s) \leq \mathcal{A}$$

where

$$\delta g_{\max}(n_m, n_s) = \max(g_m^l(n_m) - g_s^e(n_s), g_s^l(n_s) - g_m^e(n_m)).$$

Proof Since $g_m^e(n_m) \leq g(n_m) \leq g_m^l(n_m)$ and $g_s^e(n_s) \leq g(n_s) \leq g_s^l(n_s)$, we get:

$$g_m^l(n_m) - g_s^e(n_s) \geq g(n_m) - g(n_s)$$

and:

$$g_s^l(n_s) - g_m^e(n_m) \geq g(n_s) - g(n_m)$$

Therefore:

$$\max(g_m^l(n_m) - g_s^e(n_s), g_s^l(n_s) - g_m^e(n_m)) \geq |g(n_m) - g(n_s)|$$

As a result, if:

$$\delta g_{\max}(n_m, n_s) = \max(g_m^l(n_m) - g_s^e(n_s), g_s^l(n_s) - g_m^e(n_m)) \leq \mathcal{A}$$

then:

$$|g(n_m) - g(n_s)| \leq \mathcal{A}$$

and hence media units n_m and n_s belong to a synchronization set. \square

Although, Theorem 1 defines a stronger condition than equation (1) (i.e. Theorem 1 is sufficient but not necessary) for determining a synchronization set, since the exact generation times of media units are unknown to the file server, determination of a synchronization set will have to be based on Theorem 1 rather than on equation (1). Hence, in the remainder of this paper, we use Theorem 1 as the condition for determining synchronization sets.

As soon as the file server receives a media unit n_s from a slave source S_s , the file server attempts to match n_s with the most recently received master's media unit,

n_m . In such a scenario, arrival time τ_m of n_m does not exceed the arrival time τ_s of n_s , and hence, $g_m^l(n_m) \leq g_s^l(n_s)$ and $g_m^e(n_m) \leq g_s^e(n_s)$. If $g_s^l(n_s) - g_m^e(n_m) < \mathcal{A}$, by Theorem 1, $\{n_m, n_s\}$ constitute a synchronization set.

On the other hand, if $g_s^l(n_s) - g_m^e(n_m) > \mathcal{A}$, it may seem likely that a synchronization set cannot be determined. Surprisingly, this is not true; even when $g_s^l(n_s) - g_m^e(n_m) > \mathcal{A}$, there are situations in which a later media unit $n_m + r$ from the master forms a synchronization set with n_s . To determine whether $n_m + r$ and n_s constitute a synchronization set, the file server has to estimate the $n_m + r$'s generation interval, and check if that generation interval together with that of n_s satisfy the conditions of Theorem 1.

A straightforward manner in which the file server can determine the generation intervals of all future media units $n_m + r > n_m$ is by adding $r * p$ to the generation interval of n_m , where p is the period of generation of a media unit. However, the file server's clock may not be synchronized with that of the master, and hence measured on the file server's clock there may be variations in the generation rate of the master. If p is the nominal generation period of media units transmitted by all sources, and c the maximum fractional variation in p , using simple algebraic manipulations, the actual generation period (\bar{p}) of a media unit can be shown to be bounded by:

$$p(1 - c) \leq \bar{p} \leq p(1 + c) \quad (6)$$

Given these bounds on the variation in the generation period of media units, the file server can estimate the earliest and the latest generation times for media unit ($n_m + r_m$) as:

$$g_m^e(n_m + r_m) = g_m^e(n_m) + r_m * p * (1 - c) \quad (7)$$

$$g_m^l(n_m + r_m) = g_m^l(n_m) + r_m * p * (1 + c) \quad (8)$$

where, $[g_m^e(n_m), g_m^l(n_m)]$ is the generation interval of media unit n_m .

Given the above estimates of generation times of $n_m + r_m$, the following proposition derives the conditions under which $n_m + r_m$ can form a synchronization set with n_s :

Proposition 1 *Let the generation intervals for media units n_m and n_s , as estimated by the file server, be $[g_m^e(n_m), g_m^l(n_m)]$ and $[g_s^e(n_s), g_s^l(n_s)]$, respectively. Furthermore, let $g_m^l(n_m) \leq g_s^l(n_s)$ and $g_m^e(n_m) \leq g_s^e(n_s)$. If $g_s^l(n_s) - g_m^e(n_m) > \mathcal{A}$ then the file server can determine that media unit ($n_m + r_m$) forms a synchronization set with n_s , that is, $\delta g_{\max}(n_m + r_m, n_s) \leq \mathcal{A}$ if and only if*

$$\left[\frac{(g_s^l(n_s) - g_m^e(n_m)) - \mathcal{A}}{p * (1 - c)} \right] < r_m$$

$$< \left[\frac{\mathcal{A} - (g_m^l(n_m) - g_s^e(n_s))}{p * (1 + c)} \right]$$

Proof The file server can determine that media unit $(n_m + r_m)$ forms a synchronization set with n_s if $n_m + r_m$ and n_s satisfy the condition of Theorem 1, i.e. media unit $n_m + r_m$ should be chosen such that $g_m^e(n_m + r_m) \geq g_s^l(n_s) - \mathcal{A}$, and $g_m^l(n_m + r_m) \leq g_s^e(n_s) + \mathcal{A}$ (see Figure 3). Substituting for the estimates of $n_m + r_m$'s generation interval from equations (7) and (8), we obtain:

$$g_m^e(n_m) + r_m * p * (1 - c) \geq g_s^l(n_s) - \mathcal{A}$$

$$\Rightarrow r_m \geq \frac{(g_s^l(n_s) - g_m^e(n_m)) - \mathcal{A}}{p * (1 - c)} \quad (9)$$

and:

$$g_m^l(n_m) + r_m * p * (1 + c) \geq g_s^e(n_s) + \mathcal{A}$$

$$\Rightarrow r_m \leq \frac{(g_s^e(n_s) - g_m^l(n_m)) + \mathcal{A}}{p * (1 + c)} \quad (10)$$

Combining equations (9) and (10), together with the observation that r_m is an integer, we obtain:

$$\left\lceil \frac{(g_s^l(n_s) - g_m^e(n_m)) - \mathcal{A}}{p * (1 - c)} \right\rceil \leq r_m$$

$$\leq \left\lfloor \frac{\mathcal{A} - (g_m^l(n_m) - g_s^e(n_s))}{p * (1 + c)} \right\rfloor$$

Using the above proposition, the file server can determine synchronization sets and assign RTS to slave media units. If we define:

$$r_{min} = \left\lceil \frac{(g_s^l(n_s) - g_m^e(n_m)) - \mathcal{A}}{p * (1 - c)} \right\rceil$$

and

$$r_{max} = \left\lfloor \frac{\mathcal{A} - (g_m^l(n_m) - g_s^e(n_s))}{p * (1 + c)} \right\rfloor$$

the exact algorithm for assigning RTS to a slave media unit n_s , given a master media unit n_m , is as follows:

RTS assignment algorithm:

1. If $r_{max} - r_{min} = 0$, then $\{n_m + r_m, n_s\}$ is guaranteed to be a unique synchronization set, and $RTS(n_s) = RTS(n_m) + r_m$.
2. If $r_{max} - r_{min} > 0$, then all the media units $(n_m + r_m)$, $r_{min} \leq r_m \leq r_{max}$ form synchronization sets with n_s .

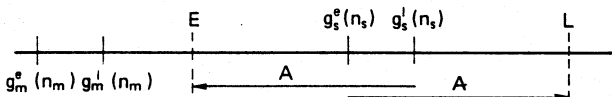


Figure 3 Determination of a media unit $n_m + r_m$ that can form a synchronization set with n_s ; the generation interval of $n_m + r_m$ must be within the interval $[E, L]$, i.e. $g_m^e(n_m + r_m) \geq g_s^l(n_s) - \mathcal{A}$ and $g_m^l(n_m + r_m) \leq g_s^e(n_s) + \mathcal{A}$

The set of possible RTSs for n_s is: $RTS(n_m) + r_m$, $r_{min} \leq r_m \leq r_{max}$. If the RTS is required to be unique, it can be chosen to be that of the master's media unit whose generation interval estimates are closest to n_s 's generation interval, i.e. $\delta g_{max}(n_m + r_m, n_s)$ is the smallest.

3. If $r_{max} - r_{min} < 0$, then the available information is insufficient for the file server to determine a synchronization set (and hence to assign an RTS to n_s). However, the file server may be able to do so when it receives a more recent media unit from the master.

□

SYNCHRONOUS RETRIEVAL

During retrieval of a multimedia object, the media streams constituting the object may have to be routed to different display devices on the network. Furthermore, each media stream may have been generated by one (input) device, but may be displayed on another (output) device, and there may be non-deterministic rate mismatches between the two devices. Synchronous retrieval can be guaranteed if the display of all the media streams starts simultaneously, and proceeds at identical rates. However, in practice, mismatches in clock rates of display devices, media losses resulting on the network or during protocol processing (e.g. buffer overruns), jitters in communication delays, excessive delays (which may lead to idling of display devices), and other anomalies can cause display devices to go out of synchrony very soon after the start of retrieval.

The RTS that is assigned during storage can be used to mutually synchronize the display of media streams during retrieval. At any instant, asynchrony can be detected by comparing the RTSs of units being displayed by the slaves with the RTSs of media units being displayed by the master. Mismatches in the RTSs of media units being played back concurrently at master and slave devices are symptomatic of asynchrony between slaves and the master. Synchrony can then be restored by having a slower slave skip the required amount of data to catch up with the master, or a faster slave pause to wait for the master (the master device always continues its display at its natural rate).

The file server, since it has to transmit media streams to all the display devices, is best suited to handle synchronization during retrieval with little additional overhead. To permit the file server to detect asynchrony between slaves and the master, we present a mechanism in which all of the display devices transmit *feedback* units to the file server. Whenever a device schedules a media unit for display, it also schedules a feedback unit for transmission back to the file server. Each feedback unit contains the RTS of the media unit that is concurrently scheduled for display. By comparing the

RTSs of feedback units received from the slaves and the master, the file server detects the exact amount of asynchrony, after taking into account the jitter in communication delay and mismatches in generation rates between slaves and the master. The file server then takes appropriate corrective actions by deleting or duplicating media units in the media streams scheduled for transmission to the slaves, and thereby brings them back to synchrony. Furthermore, the file server tries to predict the time in future when the asynchrony of a device is expected to exceed the permitted bound, and take preventive action (which may be gradual instead of abrupt) in advance to remedy the expected asynchrony.

Feedback technique for detecting asynchrony

Let D_m be the master and D_s be a slave display device on the network to which media streams of a multimedia object are being retrieved from the file server. D_m and D_s create and transmit feedback units, f'_m and f'_s , respectively, back to the file server, whenever they initiate the display of a media unit in the media stream that they receive from the file server. Given a communication delay jitter of $[\Delta_{min}, \Delta_{max}]$ on the network, when the file server receives feedback units f'_m and f'_s at arrival times τ_m and τ_s , the file server can easily estimate their generation intervals $[g_m^c(f'_m), g_m^l(f'_m)]$ and $[g_s^c(f'_s), g_s^l(f'_s)]$, respectively, using equations (2-5). Based on these generation intervals, the file server can use Proposition 1 to determine a set of feedback units $\{f_m, f_s\}$ such that $\delta g_{max}(f_m, f_s) \leq \mathcal{A}$. We refer to such a set as a *playback set**. The file server uses playback sets to determine the asynchrony in the display between D_m and D_s .

Each feedback unit contains the RTS of the media unit whose display had been initiated concurrently with the transmission of that feedback unit. Hence, the earliest and latest generation times of feedback units f_m and f_s are also the estimates of the earliest and latest times at which the display of their corresponding media units could have been initiated. Therefore, the feedback units in a playback set correspond to media units that are being displayed within a time difference of \mathcal{A} . Consequently, if feedback units corresponding to media units with different RTSs are determined to belong to the same playback set, the file server decides that there is asynchrony between the master and the slave, and it deletes or duplicates media units (or instructs the slave to do so) corresponding to the difference in those RTSs. Precisely, if $\{f_m, f_s\}$ form a playback set and RTS(f_m) and RTS(f_s) are the RTSs of media units corresponding to feedback units f_m and f_s , the asynchrony is given by $RTS_{m-s} = RTS(f_m) - RTS(f_s)$, and the file server takes the following actions:

- if $RTS_{m-s} = 0$, then D_m and D_s are in synchrony of display;
- if $RTS_{m-s} > 0$, then the slave is consuming media units slower than the master. The file server *deletes* RTS_{m-s} media units from the transmission queue destined for the slave, so as to bring the slave's display back to synchrony with that of the master;
- if $RTS_{m-s} < 0$, then the slave is consuming media units faster than the master. The file server *duplicates* $|RTS_{m-s}|$ media units in the transmission queue destined for the slave so as to bring the slave's display back to synchrony with that of the master.

As an alternative policy, in cases when the file server may have already transmitted future media units in advance to be buffered for display at the slave, instead of the file server directly deleting or duplicating media units, it can instruct the slave to carry out the necessary deletion or duplication in the slave's display buffers.

When the RTSs of slave media units are not unique, there are several choices for deciding upon the conditions under which to resynchronize. For instance, suppose that $\{f_m, f_s\}$ constitute a playback set, with f_s corresponding to a set of RTSs: $RTS_s = [RTS_s^1, RTS_s^2, \dots, RTS_s^i]$, and f_m corresponding to a set of RTSs: $RTS_m = [RTS_m^1, RTS_m^2, \dots, RTS_m^i]$. Some of the conditions under which the file server may decide to resynchronize are: (i) first, there is at least one element of RTS_s that does not belong to RTS_m ; (ii) none of the elements of RTS_s belongs to RTS_m , and so on.

Predictive resynchronization

In practice, it is more advantageous to be able to predict the time in future when the asynchrony between the master and a slave is expected to exceed the permitted bound, and carry out remedial deletions or duplications of media units gradually in advance to nullify the predicted asynchrony. To do so, given the generation intervals of a feedback unit f_s from a slave, the file server can estimate the generation intervals of future feedback units ($f_s + r_s$) to be:

$$g_s^c(f_s + r_s) = g_s^c(f_s) + r_s * p * (1 - c)$$

$$g_s^l(f_s + r_s) = g_s^l(f_s) + r_s * p * (1 + c)$$

Using the above estimates of the generation interval of a future unit $f_s + r_s$ from the slave, and the estimate for the generation interval of the most recently received feedback unit f_m from the master, the file server can use Proposition 1 to determine a unit $f_m + r_m$ from the master such that $f_m + r_m$ forms a playback set with ($f_s + r_s$). The file server can then predict the asynchrony $RTS_{m-s} = RTS(f_m + r_m) - RTS(f_s + r_s)$ that is expected to occur at the time of playback of media unit $f_s + r_s$ at the slave, and initiate appropriate remedial

*As we will see shortly, feedback units in a playback set correspond to media units that are displayed simultaneously, hence the name *playback set*.

deletions or duplications to gradually pull the slave back to synchrony with the master.

However, carrying out the above determination of playback sets at the file server for every feedback unit (and hence, for every media unit displayed) in real-time can incur a lot of computational overhead. To avoid this computational overhead, recall that if $\{f_m, f_s\}$ constitute a playback set, then $\delta g_{\max}(f_m, f_s) \leq \mathcal{A}$. The difference, $\mathcal{A} - \delta g_{\max}(f_m, f_s)$ represents the extent of asynchrony that can be accommodated in the display of later media units before resynchronization becomes necessary. By using equations similar to equations (7) and (8) for each successive pair of feedback units we obtain that the maximum possible asynchrony (given by the difference between the latest generation time of one unit and the earliest generation time of another) increases at most by $2 * p * c$, and hence, for feedback units $(f_m + r)$ and $(f_s + r)$, this increase is at most $2 * p * c * r$. Thus, given that feedback units f_m and f_s form a playback set, feedback units $(f_m + r)$ and $(f_s + r)$ are guaranteed to form a playback set only if $\delta g_{\max}(f_m, f_s) + 2 * p * c * r \leq \mathcal{A}$. If we define the recomputation point as

$$r_{\text{comp}} = \left\lfloor \frac{\mathcal{A} - \delta g_{\max}(f_m, f_s)}{2 * p * c} \right\rfloor$$

then, for all values of $r \leq r_{\text{comp}}$, it is guaranteed that, units $(f_m + r)$ and $(f_s + r)$ constitute a playback set, and the file server can thereby avoid lengthy computation of these playback sets. For each of these playback sets (i.e. for each value of $r \leq r_{\text{comp}}$), the file server has to compare RTS $(f_m + r)$ with RTS $(f_s + r)$. If RTS $(f_m + r) \neq$ RTS $(f_s + r)$, then it is referred to as a *resynchronization point*, and the first such point is denoted by r_{sync} . Resynchronization points refer to instances *in future* at which asynchrony is expected to exceed permitted bounds. Thus, asynchrony is detected at the earliest possible instant, and the file server initiates remedial deletions or duplications gradually in advance so as to prevent the RTS mismatch at r_{sync} . Note that the effect of such remedial action percolates to all later resynchronization points, i.e. remedial action at r_{sync} will reduce RTS mismatches at all later resynchronization points.

At the recomputation point r_{comp} , the file server reaches the limit for predicting playback sets with the available information, and for all values of $r > r_{\text{comp}}$, the file server has to wait to receive future feedback units in order to continue with prediction of playback sets. (The proof of this claim is simple and proceeds by showing that if the file server can predictively compute a playback set $[f_m + r, f_s + r]$, then $[f_m + r - 1, f_s + r - 1]$ must also be a playback set.) If the reception of a new feedback unit advances the recomputation point, the file server repeats the procedure for predicting playback sets.

Predictive resynchronization, not only prevents

asynchrony in advance, but also enables reduction in the transmission overhead of feedback units. This is because, the file server can compute the playback sets of numerous future media units when it receives just one feedback unit. Hence, feedback units need not be transmitted every time when one of these media units is displayed, but only when it gets close to the recomputation point.

DISCUSSION

The above methods for predicting future playback sets can also be directly applied to predicting future synchronization sets during RTS assignment, which can significantly reduce computational overhead during storage of multimedia objects. There are some realistic generalizations of the above synchronization problem. For instance, the masters used for RTS assignment and display may be different, and the periods of media units of various sources may be different. We now extend our techniques to these general cases.

Choice of master

Master devices for media storage and retrieval are chosen depending on the application. During storage, the RTS of masters' media units always advances by a constant factor, whereas those of slaves' media units may have jumps or plateaus. During retrieval, the master device always continues to display at its natural rate, while other devices may be required to delete or duplicate media units to remain synchronous with the master. Hence, the master (for both storage and retrieval) may be chosen on the basis of relative importance of smoothness and continuity of its display. For instance, in the case of audio/video retrieval, if smoothness of audio playback is more important than the smoothness of video display, then video frames may be deleted or duplicated as needed to maintain synchrony with audio. On the contrary, in the case of video editing, all the video frames must be displayed without omissions. Other criteria that may be used for the selection of the master are: accuracy of display rate, minimality of jitter on network path, etc. A special case is when the file server itself is the master, in which case, the estimate of master's generation interval does not increase (by $2 * p * c$) for every future media unit, thereby allowing the file server to predict a larger number of playback or synchronization sets.

The master device for retrieval need not be the same as that used for RTS assignment during storage. In fact, many a time, output devices that playback media streams constituting a multimedia object may be entirely different from input devices that generated the multimedia object, and there may be non-deterministic

rate mismatches between the two sets of devices. Masters may also be changed on the fly during storage or retrieval based on policies such as round-robin, statistical performance criteria, etc.

The choice of the master for retrieval and the maximum tolerable asynchrony \mathcal{A} (both of which are application specified) governs the progress of display at all the display devices, and hence determine the synchronization rules. For example, if the master is chosen to be the device which always displays at the slowest rate, all other display devices are only subjected to pauses during the retrieval. Similarly, the choice of the maximum tolerable asynchrony, \mathcal{A} will determine the instants and frequency of resynchronization; the smaller the value of \mathcal{A} the greater is the frequency of resynchronization. Whereas the feedback technique, as described here, permits the master to be changed on-the-fly, the maximum asynchrony \mathcal{A} is used for determining synchronization sets and hence, needs to be decided at the time of recording. This scheme can be easily extended so as to make the determination of synchronization sets independent of \mathcal{A} and to use the maximum asynchrony only at the time of retrieval, thereby permitting the asynchrony to be changed at any time by the application. In this scheme, the time window used for determining synchronization sets can be chosen to be the smallest permitted value; the restriction on the size of the window arises from the network jitter, $\Delta_{\max} - \Delta_{\min}$, which limits the precision with which generation times of media units are estimated, and the duration of display of a media unit p because the synchronization set is guaranteed to contain at least one media unit of the master and slave streams if the window is chosen to be at least $p/2$.

M-ary synchronization

In the RTS assignment algorithm, all the slave media units whose generation times differ from that of a master's media unit by a maximum asynchrony of \mathcal{A} are assigned the same RTS as the master's media unit. However, two such slave units from two different slaves, each of which differs from a master media unit by a maximum of \mathcal{A} , may themselves differ by an asynchrony larger than \mathcal{A} . To see how, consider the maximum asynchrony between two slave units, n_{s1} and n_{s2} that are assigned the same RTS as a master unit, n_m . Since $\delta g_{\max}(n_{s1}, n_m) < \mathcal{A}$ and $\delta g_{\max}(n_{s2}, n_m) < \mathcal{A}$, it is guaranteed that $g_{s1}^e(n_{s1})$, $g_{s1}^l(n_{s1})$, $g_{s2}^e(n_{s2})$, and $g_{s2}^l(n_{s2})$ belong to the interval $[g_m^l(n_m) - \mathcal{A}, g_m^e(n_m) + \mathcal{A}]$. Thus, the length of this interval is the maximum asynchrony between the two slave units, and is given by $\Theta = 2 \cdot \mathcal{A} - (g_m^l(n_m) - g_m^e(n_m)) = 2 \cdot \mathcal{A} - (\Delta_{\max} - \Delta_{\min})$, which may actually be higher than \mathcal{A} . This is mainly because the RTS assignment algorithm is *binary* in nature in the sense that it considers each pair of [master, slave]

separately in assigning RTS, rather than considering all the slaves together with the master.

A better synchronization policy is to restrict the maximum asynchrony between any two units that are assigned the same RTS to \mathcal{A} , whether master-slave or slave-slave. In other words, all units with the same RTS must be guaranteed to be within a window of \mathcal{A} . Such an RTS assignment algorithm is said to be *m-ary*. It can be shown that, if there are m sources, the smallest window of asynchrony \mathcal{A} within which all sources are guaranteed to have generated a media unit is $p - p / (2^m - 1)$, which reduces to p if m is unbounded.

Heterogeneity in periods of media units

RTS assignment becomes slightly tricky when the periods of media units of various sources are different. In such a situation, the master and each of the slave units are regarded as composed of imaginary smaller media units, each of period equal to their greatest common divisor (GCD). Thus, if the master's and slaves' media unit periods are $p_m, p_{s1}, p_{s2}, \dots, p_{sm}$, master's and slaves' media units are regarded as composed of $p_m / \text{GCD}(p_m, p_{s1}, p_{s2}, \dots, p_{sm})$, $p_{s1} / \text{GCD}(p_m, p_{s1}, p_{s2}, \dots, p_{sm})$, \dots , $p_{sm} / \text{GCD}(p_m, p_{s1}, p_{s2}, \dots, p_{sm})$ imaginary media units, respectively, each of period, $\text{GCD}(p_m, p_{s1}, p_{s2}, \dots, p_{sm})$. The master's imaginary media units are then assigned RTS, synchronization sets are determined containing imaginary units of the master and slaves (the generation times of imaginary units are estimated using equations (7) and (8) after proportionally reducing p and c), and RTSs are mapped back to the real media units of the master and slaves. The GCD or its factor can also be used as the granularity of the RTS. Clearly, since resynchronization during retrieval is done by comparing RTSs and deleting or duplicating (whole) media units of display, the granularity of RTS should be a common factor of the periods of media units of all sources and destinations.

RELATED WORK

The problems in media synchronization are just being recognized. Whereas Steinmetz⁶ presents a set of programming constructs for expressing inter-media relationships, Little *et al.*⁷ propose Petri net-based models for formally describing synchronization requirements among media streams. Nicolaou⁸ presents 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.*⁹ describe algorithms for recovering from loss of synchronization between interrupt-driven media input/output devices. Escobar *et al.*¹⁰ present mechanisms for controlling media synchronization parameters in multimedia applications such as conferencing. Using these

protocols, synchronization is enforced by negotiation of playout delays between destinations, based on the network delays; the playout delays are dynamically adapted based on changes in network delays. These are only a representative sampling of the many multimedia projects that are investigating issues in media synchronization.

Both Steinmetz and Little *et al.* address the problem of media synchronization at a much higher level than us. Whereas Nicolaou mainly proposes mechanisms and abstractions for representation and transfer of synchronization requirements between various levels and protocol layers, the work of Anderson *et al.* is mainly applicable to single-site multimedia workstations. Escobar *et al.* develop synchronization protocols for multimedia applications on networks assuming the existence of globally synchronized clocks. In comparison, our work has mainly focused on mechanisms and algorithms for synchronization between media streams during file storage and retrieval on multimedia networks, and our techniques stay robust in the absence of global clocks, presence of transmission jitter and generation rate mismatches.

CONCLUDING REMARKS

We have addressed the problem of media synchronization in future integrated networks and presented algorithms by which a file server can create a relative time system and synchronize media units transmitted by different sources on a network to construct a multimedia object. We have developed a feedback technique using which the file server can detect asynchronies in display devices during retrieval of multimedia objects, and restore synchrony by deleting or duplicating media units destined for asynchronous destinations. We have also presented strategies by which the file server can actually predict the time in future when the asynchrony of a device is expected to exceed the permitted bound, and take gradual preventive action to nullify the asynchrony in advance. We are generalizing these algorithms to heterogeneous multimedia networks in which there may be variations in sizes of media units generated, differences in network locations of sources and destinations, etc. We are

studying the performance including success rates of these synchronization techniques in a testbed digital multimedia on-demand storage server that is being developed at the UCSD Multimedia Laboratory¹¹.

We are extending the synchronization mechanism so as to minimize the transmission of feedbacks (by *selective* transmission of feedbacks), but at the same time restricting the maximum possible asynchrony to within tolerable limits. In multimedia environments spanning large distances, it might be more efficient to implement the synchronization mechanisms at intermediate sites on the network, rather than at a central network file server. We are developing distributed and hierarchical synchronization protocols towards such multimedia environments. We are also investigating the synchronization requirements when the individual components themselves of a multimedia object may be stored at different file servers on the network.

REFERENCES

- 1 Cochrane, P and Brain, M 'Future optical fiber transmission technology and networks'. *IEEE Commun. Mag.* (November 1988) pp 45-60
- 2 Gait, J 'The optical file cabinet: a random-access file system for write-once optical disks'. *IEEE Comput.*, Vol 21 No 6 (11-22 June 1988)
- 3 Luther, A C *Digital Video in the PC Environment*, McGraw Hill, New York (February 1989)
- 4 Skrzyzpczak, C S 'The intelligent home of 2010'. *IEEE Commun. Mag.* (December 1987) pp 81-84
- 5 Rangan P V and Swinehart, D C 'Software architecture for integration of video services in the etherphone environment'. *IEEE J. Sel. Areas Commun.*, Vol 9 No 9 (December 1991) pp 1395-1404
- 6 Steinmetz, R 'Synchronization properties in multimedia systems'. *IEEE J. Sel. Areas Commun.*, Vol 8 No 3 (April 1990) pp 401-412
- 7 Little, T D C and Ghafoor, A 'Synchronization and storage models for multimedia objects'. *IEEE J. Sel. Areas Commun.*, Vol 8 No 3 (April 1990) pp 413-427
- 8 Nicolaou, C 'An architecture for real-time multimedia communication system'. *IEEE J. Sel. Areas Commun.*, Vol 8 No 3 (April 1990) pp 391-400
- 9 Anderson, D P and Homsy, G 'A continuous media I/O server and its synchronization mechanism'. *IEEE Comput., Special Issue on Multimedia Information Systems* (October 1991) pp 51-57
- 10 Escobar, J, Deutsch, D and Partridge, C A *Multi-Service Flow Synchronization Protocol*. BBN Systems and Technologies Division, USA (March 1991)
- 11 Rangan, P V and Vin, H M 'Designing file systems for digital video and audio'. *Proc. 13th Symposium on Operating Systems Principles (SOSP'91)*, *Operating Systems Rev.*, Vol 25 No 5 (October 1991) pp 81-94