Synchronization of Temporal Constructs in Distributed Multimedia Systems with Controlled Accuracy

Sang H. Son and Nipun Agarwal
Department of Computer Science,
University of Virginia,
Charlottesville, VA 22903
son@virginia.edu, nipun@virginia.edu

Abstract

With the inception of technology in communication networks such as ATM it will be possible to run multimedia applications on future integrated networks. Synchronization of the related media data is one of the key characteristics of a multimedia system. In this paper we present a scheme for synchronization of multimedia data across a network where the accuracy of detecting asynchronization and predicting the future asynchrony is variable and can be tailored to the intended application. The protocol has been designed keeping in mind characteristics of ATM networks such as the absence of global synchronized clocks and utilizing features as the QOS promised by them. The multimedia data when sent across the network may also be stored at an intermediate node and later retrieved for display. We extend the scheme and present a mechanism wherein synchronization of all the possible temporal constructs is supported and not restricted to the "in-parallel" construct which is only one of the thirteen possible temporal rela-

Keywords : Controlled Accuracy, Distributed System, Multimedia, Synchronization, Temporal Constructs.

1. Introduction

Multimedia has become an eye-catching term for many disciplines of computer science. Though it is commonly agreed that future computing systems should provide multimedia capabilities, there is some uncertainty about the term and what particular functions a multimedia system should provide.

Media flexibility is a major and agreed upon requirement of a multimedia system, which requires that the multimedia system combine both continuous (CM) and discrete media (DM). Though a necessary condition, the mere incorporation of these different media is not sufficient to achieve flexibility. A multimedia system should also be able to handle each type of media independently, providing the opportunity to combine them in arbitrary ways [1]. Hence the following definition has been arrived at [2]; *a multimedia system is characterized by the computer-controlled generation, manipulation, presentation, storage and communication of independent discrete and continuous media.*

The innovation that multimedia systems provide is the *integrated* manipulation of the multimedia information; the independent manipulation of each separate information medium does not provide anything new. This ability of integrated manipulation of the multimedia information provides the potential for dynamically creating new, user-directed composite data sets. However, operating systems and programming language interface have not kept up with the evolution of the user interface for supporting dynamic temporal data relationships. Hence, the challenge imposed by multimedia is not simply providing I/O support for the various media as audio, video and text. The challenge is of providing support for synchronizing otherwise autonomous data transfers within and across the computers [3].

The support for multimedia service is guided by two distinguishing features [5]:

- Support for continuous media: Unlike discrete media as text or graphics, continuous media requires special support while being processed or displayed. For instance CM data should be available within a certain time interval to be useful for an application, should be handled in a timely fashion to control jitter and should be obtained or rendered at the I/O devices at a certain rate to fulfill their operational requirements [4].

- Synchronization among media streams: During playback, the various media streams should not only be continuous but should also be temporally coordinated. The ability for synchronization refers to not only synchronization “in parallel” but synchronization of all the possible temporal constructs.

Synchronization is one of the most critical problems in real-time applications of ATM networks. The exploration of the issues of synchronization in a multimedia system and their solutions is in an incipient stage. Steinmetz [7] and Little and Ghafoor [8] have discussed methods for formally describing the synchronization requirements in a multimedia environment. Steinmetz discusses the characteristics of a multimedia system and presents a set of constructs for expressing inter-media relationships; Little and Ghafoor evince a strategy for formal specification and modeling of multimedia composition with respect to inter-

media timing, based on the logic of timed Petri nets. Hoepner [9] explores the synchronization of multimedia objects for presentation based on the petri net model. Anderson and Homsy [10] describe algorithms for synchronization among interrupt-driven media I/O devices, but these are only applicable to single site multimedia workstations. Nicolaou [11] attempts the synchronization problem in a two-level scheme; by defining explicit synchronization properties at the presentation level and by providing control and synchronization operations at the physical level. Escobar et. al [12] present an adaptive flow synchronization protocol that permit synchronizing in a distributed environment, but however operate under the assumption that global synchronized clocks are present; which is not a very practical assumption [5] considering that ATM networks will not provide a global synchronized clock. Rangan et. al [5] propose a feedback technique to detect asynchronization among the media streams in an distributed environment and to steer them to synchrony thereafter. The accuracy with which the multimedia server can detect the instant of playback of a media unit at the destination is however bounded by the network jitter. In a wide area network the jitter may be greater than the acceptable asynchrony among the media streams, thereby restricting the purview of the mechanism. For instance, if the maximum permissible asynchrony among the media streams for an application is τ and the network jitter is Ξ, and if $\Xi > \tau$, then this particular application cannot be sent over the network.

Furthermore in the approaches of both Escobar and Rangan, the synchronization assured is primarily synchronization “in parallel” among the various media streams. By synchronization “in-parallel”, we mean the capability to display the various media units simultaneously. Hence if the requirement of an application be that a segment of video be played κ time units after a audio clip - without any constraints on when the audio clip is to commence or when the video clip should end, this requirement cannot be easily or efficiently accomplished by the synchronization mechanisms which support only the parallel or simultaneous temporal constructs. We mention easy and efficient manner because it may be possible to express the above constraint for instance, by extending an empty video segment while the audio is playing and trailing empty audio segment for κ time units and for the duration the video data is being played, as shown in Figure 1. However, this method clearly is not suitable for practical applications owing to the over head incurred due to the addition of extended data and the complexity of the process.

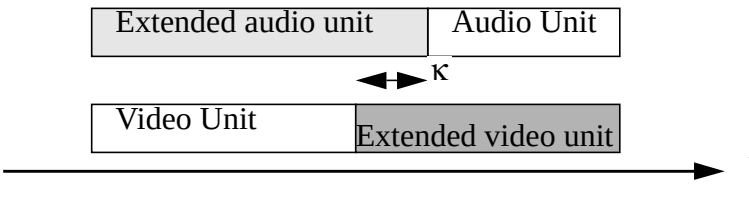

Figure 1. Synchronization of “after” temporal constructs using “in-parallel” construct.

In this paper, we present an algorithm for the synchronization of various media streams in ATM networks based on the synchronization marker concept which does not assume the presence of globally synchronized clock - though the algorithm is not restricted to ATM networks alone. The accuracy of detecting the asynchrony which is independent of the

network jitter, is variable and can thus be tailored to the intended application. We address the issue of assigning relative time stamps to the data originating from different media sources and discuss how our algorithm assigns the time-stamps with an accuracy that can be varied to suit the application, in the absence of global synchronized clocks. We then present a mechanism that guarantees the synchronization of any possible temporal relation among various media streams, not just the “in-parallel” temporal construct.

The remainder of this paper is organized as follows. Section 2 describes the system model used in this paper. Section 3 discusses the mechanisms used to assign relative time-stamps to the media data streams, generated by the media sources. Section 4 outlines the protocol to detect the asynchrony, if any, among the media streams when they arrive at the destination. The mechanisms for re-synchronizing the out-of-sync streams is discussed in section 5. Section 6 extends the proposed protocol to synchronize various data streams which can support all the temporal constructs. We conclude in Section 7 with some directions for future work.

2. System Model

Consider a simple scenario, where media data such as audio, video and text is being generated by a number of different media sources. This data is desired to be transmitted across the ATM network to a destination where it is to be played back. The different media data may be routed through different parts of the network which may experience different quality of service, such as jitter and delay. The devices at the destination which play back the media data, may also have mismatches in the playback rates and may differ from the devices used at the source. Further-more, the data may be stored on a physical storage at an intermediate site and may be retrieved for play back after some time.

In any of the above cases, it is required that the data be synchronized while it is being played at the destination devices - i.e. played back in the same time order as was played at the source. For instance when related audio and video data are sent over the network, it is required that they be played back at the destination in the same time order as they were at the source. A different scenario having the same synchronization requirement would be when data being sent from a single source is required to be received at multiple destinations simultaneously, e.g. in a teleconferencing application.

The above scenario is depicted in Figure 2. The goal is to send some multimedia data from the media sources to the media destinations over the network A. Let a node on the network, called the Multimedia server (MMS) be responsible for controlling the synchronization of media data at the destination and for storing the data if required by the application. The network A is logically divided into network B and network C. The data travels from network B to the MMS and from the MMS travels over network C to the media destinations. The MMS may hence be at any point on the network. Since the accuracy of detecting the asynchrony of our algorithm is independent of the network jitter, the MMS need not be necessarily physically close to the media destinations.

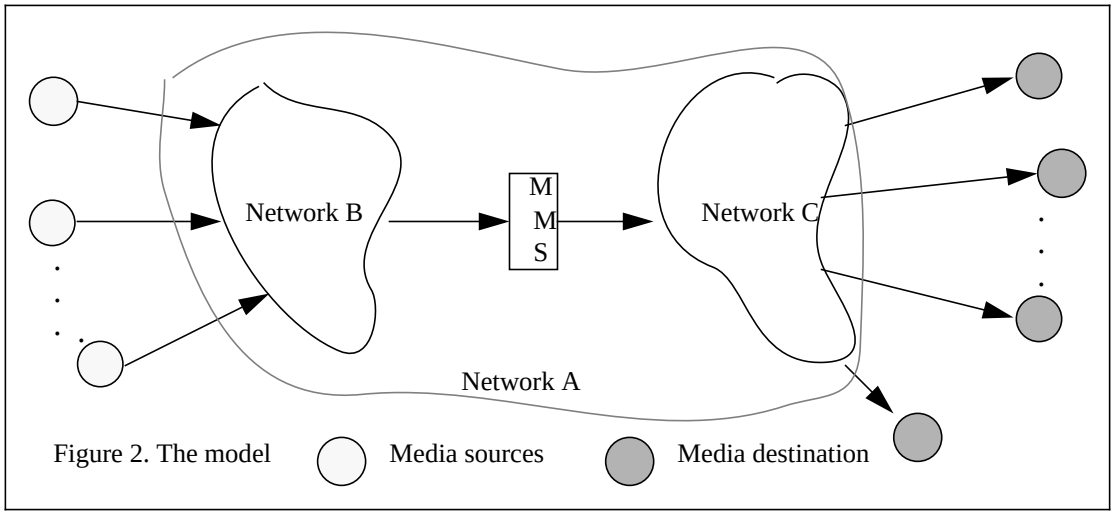

Figure 2. The model

3. Assigning the Relative Time-stamps

The aim of assigning relative time-stamps (RTS) is that media units in different data streams that have the same relative time-stamp are required to be synchronized during play back. Assigning RTS to the data could be performed quite conveniently and accurately by counting the number of bytes in each data stream, if compression is not involved. For instance, if a RTS is desired to be assigned at intervals of 1/30th of a second, and the frame rate be 30 fps; then calculating one frame size would give us the number of bytes after which a RTS should be assigned in the incoming video data stream. Similarly, calculating the amount of audio data generated every 1/30th of a second would give us the intervals after which RTS should be assigned to the incoming audio stream. The streams may then be sent to the playback sites where the media units in the audio and video stream with the same RTS are synchronized during playback.

Clearly, if data is compressed before being sent over the network, as is very likely in a multimedia environment, the above strategy cannot be employed because compression algorithms such as JPEG are sensitive to the content of the image and the size of the compressed data may be variable. The requirement for assigning relative time-stamps to the various media streams hence calls for a mechanism based on the generation times of the data at the respective sites. A similar approach has been employed in [13], however their approach may not be suitable if the application has some bounds on the accuracy with which the RTS should be assigned; because the accuracy with which their model can assign the RTS is bounded by the jitter of the network, which cannot be changed and may be large particularly in wide area networks.

In our approach for assigning the relative time-stamps, the data streams are time-stamped as they are generated at the media sources, according to the respective clocks at the media sources and sent to the multimedia server. Note that time-stamp refers to the actual clock time that the media data is sent/received, while relative time-stamp is used to indicate an order among the various media units. By a mechanism which we will describe below, the various clock times are converted to a single clock time which we refer to as the *normal-*

ized clock time. If the normalized clock time of the media units on the various media steams is the same, that is they were generated at the same time, they are assigned the same RTS. It may be observed that this mechanism does not necessitate global clock synchronization. If the absolute time of all the clocks involved is known at some point in time and the clock skews of the different clocks is known, then it is possible to normalize the various clock times and hence determine the corresponding points in the different data streams that were generated at the same time by the media sources. This may be achieved as follows. Consider the layout depicted in Figure 3.

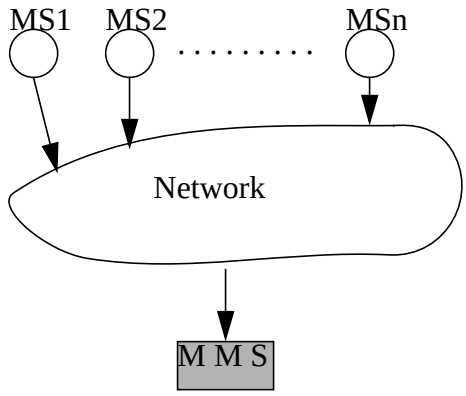

Figure 3. Assignment of the relative time stamps to media units

For the sake of simplicity, we consider only two active media sources in our discussion here, namely MS1 and MS2. A connection based on the quality of service (QOS) control model as described in [15] is established from the media sources (MS1 and MS2) to the multimedia server (MMS). Establishing a QOS session guarantees a lower bound on quality of service parameters as end-to-end delay, jitter bound etc., once a connection is established. When a new request for establishing a QOS session comes in, the request is accepted only if the network can satisfy the QOS parameters specified therein. This may result in the degradation of the QOS parameters of the other established sessions to their minimum value - i.e. degradation of sessions which were being rendered service better than the minimum promised. The only guarantees our protocol requires is on the upper bound on the delay and hence in this paper we shall refer to the quality of service to refer to the maximum packet delay that can occur. If the QOS parameters for the two connections established are not the same i.e. MMS to MS1 and MMS to MS2, deviation from the equations presented here is provided in the appendix.

A packet called the *trigger packet* is sent from the MMS to the media sources to trigger them to send their respective clock times to the MMS. The trigger packet does not contain much information and thus constitutes little overhead. After an interval *t* another trigger packet is sent to the media sources on the same QOS session. On receiving the trigger packets, the media sources send a message back to the MMS indicating the time - according to their respective clocks - at which the two trigger packets were received.

Let x_0 and $x_0 + t$ be the times when the two trigger packets were received by MS1, according to MS1's clock and y_0 and $y_0 + t + p$ be the times when the two trigger packets were received by MS2 according to MS2's clock; for some *p*, where $-\infty < p < \infty$. If the delay incurred in sending the various packets were the same, then corresponding to a point in time x_1 according to the MS1 clock, the point in time according to the MS2 clock would be

$$y' = \frac{(t+p) \times (x_1 - x_0)}{t} + y_0 \qquad (1)$$

However since the delay is not constant for the different channels/packets, the maximum difference in delay that may be incurred by the different media streams is $2\delta_d$, where $\delta_d = \delta_{max} - \delta_{min}$ i.e. the jitter of the QOS session. δ_{max} and δ_{min} are the maximum and the minimum delays respectively for the QOS session and are established at the time of setting up the QOS session. The maximum error ε that may be introduced due to the jitter, in determining the clock rates of the different sources is hence

$$\varepsilon = \left(\frac{(t+p+2\delta_d)\ (x_1 - x_0)}{t} + y_0\right) - \left(\frac{(t+p)\ (x_1 - x_0)}{t} + y_0\right) = \frac{2\delta_d (x_1 - x_0)}{t} \qquad (2)$$

To achieve a lower value of ε would require that δ_d be small and *t* be large. Lower value of δ_d reflects the need for a higher quality of service parameter which would require greater communication resources to be dedicated. Having a greater value of *t* would require that the QOS session be maintained for a longer period of time. Since the session is required to have a high quality of service, blocking the communication resources for the time period *t* may be wasteful, since the only communication that is done over the QOS session in the interval *t* is the sending of two trigger packets which, as we mentioned, do not carry much data. An alternative is to terminate the QOS session soon after the first packet is sent and to reestablish a new QOS session for sending the second packet after the interval *t*. The maximum error in this case would be given by

$$\varepsilon = \frac{(\delta_{d1} + \delta_{d2}) \times (x_1 - x_0)}{t} \qquad (3)$$

where δ_{d1} is the jitter for the first QOS session and δ_{d2} is the jitter for the second session.

Having determined the absolute clock times of the different sources and the clock skew - after having received the reply of the two trigger packets - the QOS session is terminated. The connection for sending the media data from the various media sources to the MMS is then established and the data being sent from the different media sources is time-stamped according to the respective clocks and sent to the MMS. The MMS on receiving the media streams, assigns the same RTS to the media units on different media streams that have the same *normalized* time-stamp. Since the time-stamps are assigned at the media sources according to the respective clocks (which may be asynchronous), equation (1) is used to convert the different clock times to one clock time, which we refer to as the normalized clock time. The maximum error that may be introduced in comparing the different clock times is ε. Note that this value depends on the QOS parameters at the time of determining

the clock times and the clock skew and is independent of the quality of session during the assigning of the relative time-stamps, and may hence be altered according to the requirement of the application.

In the above discussion it is assumed that the clock drift if any is very small compared to the network jitter and is hence not taken into account. If this is not the case the clock drift may be calculated as discussed in the appendix.

Considering the precision of human perception it is not always necessary that the various media streams be perfectly synchronized. For instance if audio and video be synchronized within 150 msec, the asynchrony is not apparent for some application to the user [7]. This value of the permissible maximum asynchrony (ξ) is dependent on the application. For instance audio applications usually have stringent synchronization requirements (and hence lower maximum asynchronization value) than the video applications. We refer to ξ as the *tolerable asynchrony.*

4. Detection of Asynchronization

Once the data is stored in the multimedia server, the challenge is to play the various media streams at the destination, in a manner such that the media units with the same RTS are played synchronously. For the discussion in this section, we consider only simultaneous display synchronization. Other possible temporal constructs will be discussed in Section 6. Two factors playing the predominant role that may lead to asynchronization are:

- The network jitter i.e. $\Delta_{max} - \Delta_{min}$, where Δ_{max} is the maximum delay bound and Δ_{min} the minimum delay bound for the network connection while the media data is being sent. These values are different from δ_{min} and δ_{max} which are the maximum and the minimum delay bounds for the QOS session which precedes the actual transfer of media data.
- The different clock rates of the various media devices at the destination.

If α be the nominal clock rate and ω be the fractional deviation from the nominal clock rate, then the maximum asynchronization that could occur when the slower media device is playing back the media unit τ_m is shown to be [5]:

$$A = \left(\frac{(\Delta_{max} - \Delta_{min}) + 2 \times \alpha \times \omega \times \tau_m}{\alpha \times (1 - \omega)}\right).$$

The steps followed for detecting asynchronization among the various media streams are similar to the ones used to assign the RTS to the media streams being emitted from the media sources. Consider the scenario of Figure 4.

Initially a QOS session is established between the MMS and the various media destinations (MD). Two trigger packets separated by time interval *t* are sent by the MMS to the media destinations. The MDs on receiving the packets note the time of arrival of the packets according to their respective clocks and send back this time information to the MMS. This information is used in equations (1) and (2) which are used to convert time from one clock scale to another and to estimate the maximum error that may have been introduced. The QOS session is then terminated.

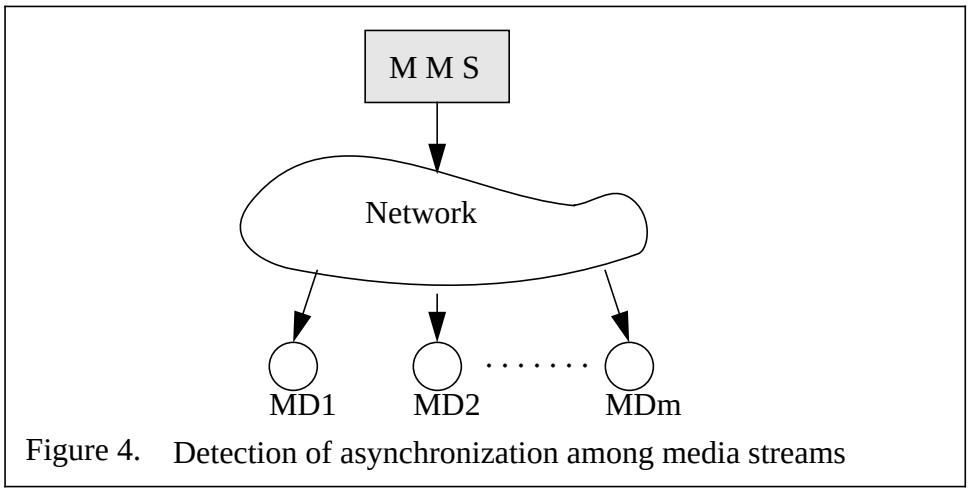

Figure 4. Detection of asynchronization among media streams

In the next phase, after the QOS session has been terminated, the media data is sent to the various MDs by the MMS and the time at which the data is received at the MDs by the respective clocks is sent back to the MMS. Since the timings are obtained relative to different clocks, equation (1) is used to convert x_1, measured by clock of MD1 to y' which is the reference clock MD2. The maximum error that may have been introduced in the conversion is bounded by ε evaluated in equation (2).

If it were possible to determine precisely the time at which the various media units with the same RTS on different media streams were displayed at the media destinations, it would enable us to determine with certainty if the media streams were synchronized or not. However, some error λ is introduced in evaluating equation (1) due to the network delay. Moreover, this value λ is variable and depends on the network traffic conditions and may vary from a minimum value 0 to the maximum value ε. Hence there would exist an interval for which it cannot be said with certainty if the media streams are synchronized or not.

Let the maximum permissible asynchrony between the media streams be ξ; and y_1 and y' be the times, according to the clock of MD2, of the media units from the different media streams with the same RTS reaching the media destinations. We have the following cases:

- Synchronization is guaranteed if $|y_1 - y'| \leq ||\xi| - |\varepsilon||$ (4)
- Asynchronization guaranteed if $|y_1 - y'| > |\xi| + |\varepsilon|$.
- Synchronization may or may not exist otherwise.

Figure 5 illustrates the above conditions.

As mentioned in Section 3, the value of ξ is dependent on the application. Certain applications, for instance those involving audio, have tighter synchronization requirements and hence lesser ξ value than others, for instance those involving video. The desired value of ξ is provided by the application and ε can be computed from equation (3). It may be observed that the values for the various intervals mentioned in equation (4) needs to be

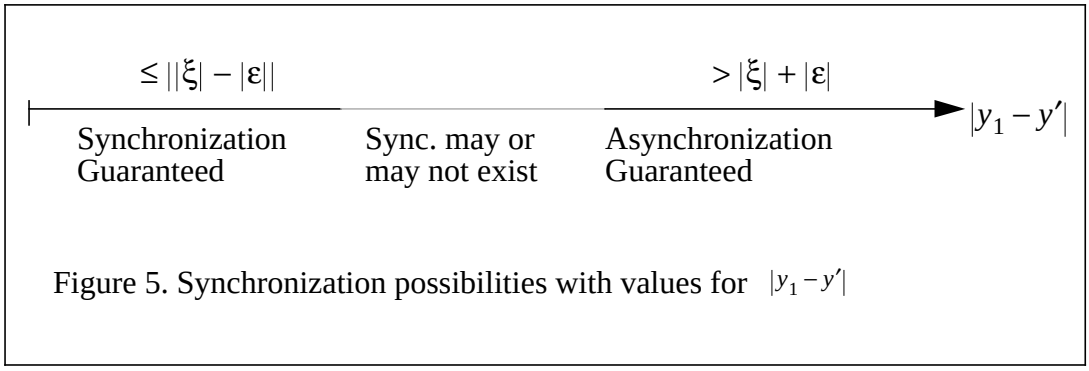

Figure 5. Synchronization possibilities with values for $|y_1 - y'|$

computed only once - after the termination of the QOS session - and hence does not constitute much overhead.

5. Re-synchronization

Having detected asynchrony if any among the various media streams, the task is to re-synchronize them. Primarily two ways exist to re-synchronize the streams, namely to duplicate the data on the leading stream or to skip data on the lagging stream. In order to enforce the above mechanisms, one of the streams is deemed the master and the other streams the slaves. It is the media units of the slaves that are subjected to skipping or deletion of the data. The choice of the master depends on the application; in a teleconferencing application for instance, if a particular site is of greater importance then the media stream to that site is deemed to be the master.

The policies for deciding when to trigger the synchronization mechanisms may be conservative, aggressive or probabilistic [5]. Conservative policies would trigger synchronization mechanisms only when $|y_1 - y'| > |\xi| + |\varepsilon|$, that is when asynchronization is guaranteed. Aggressive and probabilistic policies both trigger in the dotted region of Figure 5, i.e. whenever there is any probability of the streams being out of synchronization. However the probabilistic policies base their decision on the statistical distribution of network delays and the playback periods compared to the aggressive policies which trigger the synchronization mechanisms whenever synchronization is not guaranteed.

The time taken from the moment the asynchronization is detected to the time the streams are played back in synchronous fashion at the media destinations depends on the maximum network delay; since the asynchronization information has to travel from the media destinations to the MMS, the necessary action taken by the MMS and then the data is sent to the media destinations. If the upper bound on the network delay is too large the time taken may be unacceptable depending on the application. Hence an approach for predicting the time when the media streams could fall out of synchronization is desirable and has been suggested in the literature [13].

Given the rate of generation of the media units (α) the accuracy with which the asynchrony may be predicated is dependent on the primarily on the fractional deviation of the clocks from the nominal clock rate (ω) and the error bound in determining the asyn-

chrony of the current media unit. This error bound is fixed and bounded by the network jitter in [13] and hence cannot be altered or reduced if required. However since our scheme can control the error bound ε, it is possible to predict the asynchronization of the future media units with controlled accuracy and may hence be tailored to the application.

6. Synchronization of Temporal Constructs

Synchronization refers to making events happen in a certain time order [7]. Most of the approaches dealing with synchronization in a distributed networked environment however, present synchronization in context of synchronization of all streams in parallel. In this section we describe how the mechanisms presented in earlier sections may be extended for synchronizing other possible temporal constructs in an environment discussed in Section 2.

6.1. Assignment of the relative time-stamps

The mechanism for assigning relative time-stamps to media units of the different data streams is similar to the one proposed in Section 3. However the RTS are assigned both at the front and the rear end of the media units as shown in Figure 6. As is shown, if the RTS

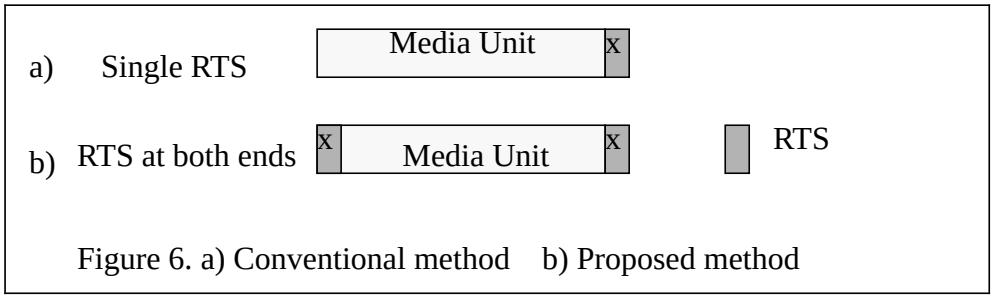

Figure 6. a) Conventional method b) Proposed method

of a media unit is υ then υ is the assigned RTS at both the front and rear end of the media unit.

6.2. Detection of asynchrony

The process of detecting asynchrony, if any, among the media streams is in confluence to the discussion in Section 4. After the relationship between the clocks of the different media destinations is established, the media units P_a and P_b are sent on the (different) communication channels. At the media destination the times at which the front end and the back end RTS are received is noted, according to the respective clocks, and the information sent back to the MMS. The MMS on receiving this information and noting the type of synchronization desired can with the help of an extension of equation (4), described below, ascertain if the media units are in synchronization or not. It may be noted here that the arrival time of both the front and rear end markers is noted and sent back, unlike the in-parallel synchronization scenario where the only time stamp of the front marker is sent back.

It has been shown that there are thirteen distinct ways in which two time intervals may be related [8], [17]. Since six of them are inverse relations of the other [8], we discuss the details of the process of synchronization carried out by the MMS for only seven of these temporal relations. Consider Figure 7. The first temporal relation is P_a *before* P_b. If the intervals P_a and P_b be swapped, we obtain the relation P_a *after* P_b. And hence, the relation *after* is the inverse of the relation *before*. Similar treatment can be performed for all the other temporal relations barring the *equal* relation, since equal is the inverse of itself.

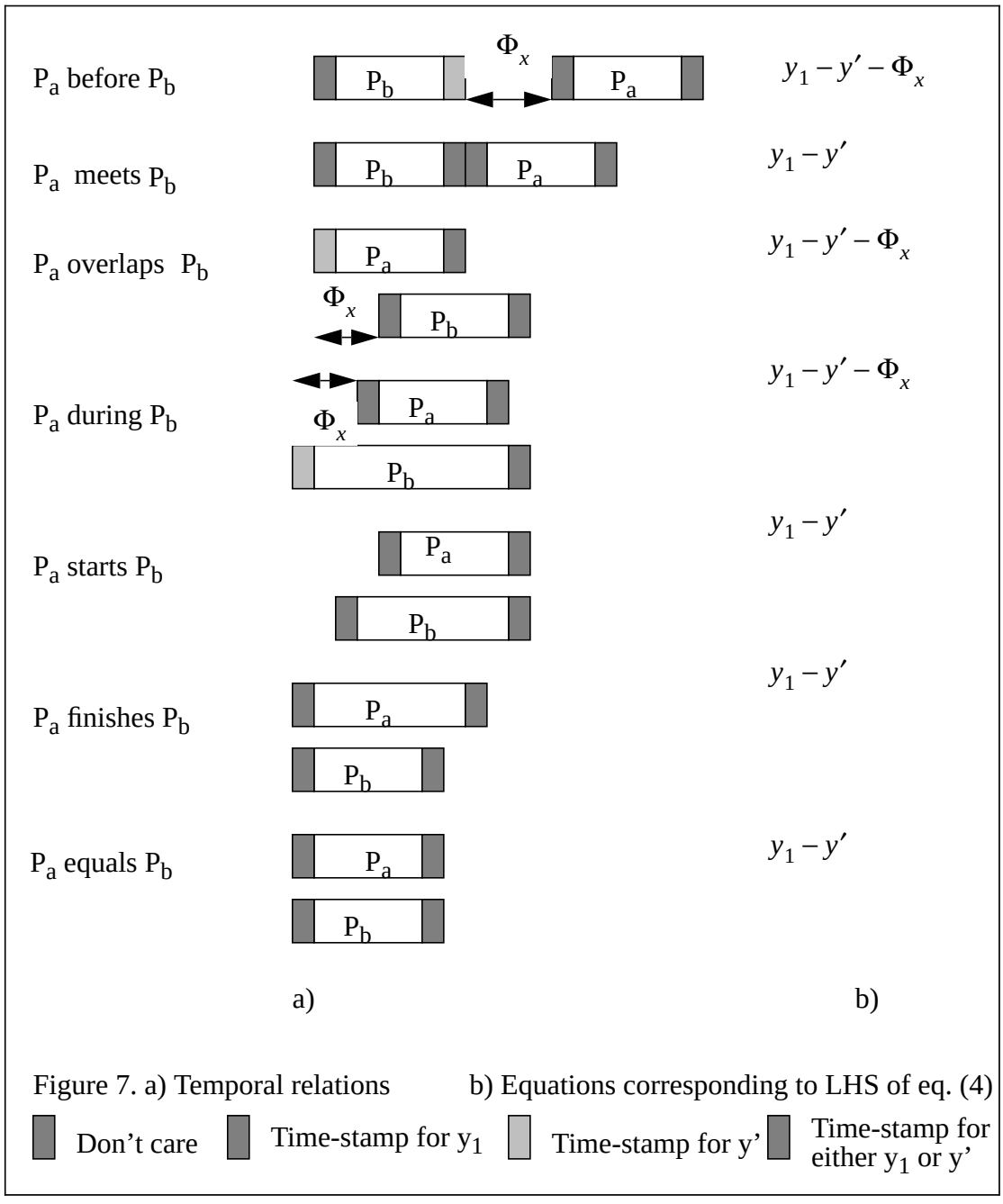

Figure 7. a) Temporal relations b) Equations corresponding to LHS of eq. (4)

Figure 7 shows the seven different temporal relations and the corresponding equations that have to be used on the L.H.S. of equation (4). Consider the first temporal relation P_a *before* P_b. The requirement of this temporal relation is that P_a be rendered Φ_x time units *before* P_b. Since there is no time restriction as to when P_a begin or P_b end, the MMS needs to know the time-stamps (arrival time at the destination) of only the rear marker of P_a and the front marker of P_b. Since the front marker of P_a and the rear marker of P_b are not required for the determination of asynchrony they are marked *don't care*. For the equations (1) and (4), the time-stamp of the rear marker of P_a is substituted for the value of y_1 and the time-stamp of the front marker of P_b is substituted for the value of y'.

Substituting $|y_1 - y' - \Phi_x|$ for $|y_1 - y'|$ in equation (4), gives the equation for detecting the asynchrony if any for the temporal relation, where y_1 is the time-stamp of the arrival of the rear RTS of the first media unit (P_a) and y' is the time-stamp of the arrival of the front RTS of the second media unit namely P_b.

That is if: (4 a)

- $|y_1 - y' - \Phi_x| \leq ||\xi| - |\varepsilon||$, then the media streams are guaranteed to be in synchronization
- $|y_1 - y' - \Phi_x| > |\xi| + |\varepsilon|$, then the media steams are assuredly out of synchronization
- Otherwise, it cannot be said with certainty if the streams are in synchronization or not.

Similarly the equations for all the other temporal relations are given in figure 7b). Re-synchronization of the other temporal constructs can be achieved in a manner similar to the one for synchronizing the parallel temporal construct.

In Figure 7, the *don't cares* refer to the markers whose time-stamp is not required by the MMS to detect asynchrony. As another example the temporal construct P_a *meets* P_b, the conditions for determining asynchronization remain the same as equation (4), with the only difference being the markers whose time-stamps are taken into consideration. We observe that for all the possible temporal relations only two forms of equations (4) exist, namely equation (4) itself and equation (4 a), with the difference among each of these equations being the markers that are considered for the substitution of y_1 and y'.

7. Conclusions

We have proposed a scheme for synchronization of related media streams in a ATM networked environment, where the accuracy of determining the asynchrony can be controlled, and hence tailored to the intended application. The idea of synchronizing media streams without global clock synchronization has been proposed in the literature, but the accuracy with which detection of asynchronization is achieved is fixed (e.g. equal to the jitter of the network [5]) and may hence not be suitable for certain applications particularly in wide area networks. We have also proposed a mechanism where we ensure synchronization of all the possible temporal constructs and not just synchronization in parallel, which is only one out of the thirteen possible temporal relations.

We are presently looking into mechanisms to express the temporal relationships while sending the data, so that different temporal relationships may be interleaved and synchronized at the destination. Also we are simulating the protocol so that we may establish a mapping between the jitter guarantees provided by the network and the maximum error with which the asynchrony may be detected.

References:

[1] R. G. Herrtwich, R. Steinmetz, "*Towards Integrated Multimedia Systems: Why and How*", IBM European Networking Center technical report # 43.9101.

[2] Ralf Steinmetz et. al , "*Multimedia-Systeme*", Informatik Spektrum, Springer Verlag, Vol. 13, No. 5, 1990, Pages 280-282.

[3] D. C. A. Bulterman, R. van Liere, "*Multimedia Synchronization in UNIX*", Proceedings of the Second International Workshop on Network and Operating System Support for Digital Audio and Video, Heidelberg, Germany, Nov 18-19, 1991, Pages 108-120.

[4] R. G. Herrtwich, "*Time Capsules: An Abstraction for Access to Continuous-Media Data*", IEEE Real-Time systems Symposium, Lake Buena Vista, 1990, IEEE press, Pages 11-20.

[5] P. V. Rangan et. al, "*Designing an On-Demand Multimedia Service*", IEEE Communications Magazine, July 1992, Pages 56-64.

[6] Nipun Agarwal, Sang H. Son et. al., "*Model Multimedia Workstation for Echocardiography*", to appear in the 17th Annual Symposium on Computer Applications in Medical Care, Washington D.C., Nov 1, 1993.

[7] Ralf Steinmetz, "*Synchronization Properties in Multimedia Systems*", IEEE Journal on Selected Areas in Communication, Vol 8, No. 3, Pages 401-411, April 1990.

[8] Thomas D. C. Little, Arif Ghafoor, "*Synchronization and Storage Models for Multimedia Objects*", IEEE Journal on Selected Areas in Communication, Vol. 8, No. 3, Pages 413-426, April 1990.

[9] Petra Hoepner, "*Synchronizing the presentation of multimedia objects*", IEEE Computer Communications, Vol. 15, No. 9, Nov 1992, Pages 557-564.

[10] David P. Anderson, George Homsy, "*A Continuous Media I/O Server and Its Synchronization Mechanism*", IEEE Computer, Pages 51-57, October 1991.

[11] C. Nicolaou, "*An Architecture for Real-Time Multimedia Communication Systems*", IEEE Journal on Selected Areas in Communication, Pages 391-400, Vol. 8, No. 3, April 1990.

[12] Julio Escobar, D. Deutsch, C. Partridge, "*Flow Synchronization Protocol*", Globecomm 1992.

[13] P. V. Rangan et. al, "*Techniques for Multimedia Synchronization in Network File Systems*", IEEE Computer Communications, Vol. 16, No. 3, March 1993, Pages 168-176.

[14] Nipun Agarwal, Sang H. Son, "*Experiences with the Development of a Multimedia Information System for Medical Applications*", Tech Rep. # CS-93-39, Dept. of Computer Science, University of Virginia, July12, 1993.

[15] Hideyuki Tokuda et. al., "*Continuous Media Communication with Dynamic QOS Control Using ARTS with an FDDI Network*", SIGCOMM 1992, Pages 88-98, August 17-20, Baltimore, MD.

[16] Ralf Steinmetz, "Multimedia Synchronization Techniques: Experience Based on Different System Structures", IEEE Multimedia Workshop 1992, Monterey, April 1992.

[17] C. L. Hamblin, "*Instants and Intervals*", Proc. of 1st International Soc. for the Study of Time, J.T.Fraser et. al, Eds., New York: Springer-Verlag, 1972, Pages 324-331.

[18] D. Ferrari et. al., "A Scheme for Real-Time Channel Establishment in Wide-Area Networks", IEEE journal on Selected Areas in Communication, Vol. 7, No. 3, April 1990, Pages 368-379.

Appendix

Consider the scenario when a QOS session is established between the MMS and two media devices say D_a and D_b and the QOS session remains established for the duration *t* (as defined in Section 3). If the QOS maximum delay parameter is not the same for the two QOS sessions then, the maximum error ε would be given as:

$$\varepsilon = \frac{2 \times max(\delta_{da}, \delta_{db}) \times (x_1 - x_0)}{t}$$

where δ_{da} is the maximum jitter for the QOS session from the MMS to the device D_a and δ_{db} is the maximum jitter for the QOS session from the MMS to the device D_b.

The function $max(m, n)$ is defined as the maximum of the values m and n.

If the QOS session be terminated soon after the sending the first trigger packet and a new QOS session be established after the time interval *t* then,

$$\varepsilon = \frac{2 \times max(\delta_{da1} + \delta_{da2}, \delta_{db1} + \delta_{db2}) \times (x_1 - x_0)}{t}$$

where δ_{da1} and δ_{da2} are the maximum jitter bounds for the first and the second QOS session respectively from the MMS to the media device D_a. δ_{db1} and δ_{db2} are the maximum jitter bounds for the first and the second QOS session respectively from the MMS to the media device D_b. The function $max(m, n)$ is the same as defined above.

Determination of Clock Drift

If the clock drift is not very small compared to the network jitter and needs to be accounted for, it may be determined by sending an additional trigger packet. That is the multimedia server sends three trigger packets in all. The first two packets as we observe in the protocol are separated by time interval *t*. The third packet may be sent after a time interval of *k* time units after the sending of the first packet. If the times at the clock under consideration be m, $m + \alpha t$ and $m + \gamma$ corresponding to the times the three trigger packets are sent by the multimedia server then the clock drift is given by:

$$drift = \frac{\gamma - (\alpha \times k)}{k}.$$

The maximum error introduced in the determination of the clock drift is also equal to the jitter bound and hence can be controlled by altering the QOS parameter of the session.