A Model for Specification and Communication of Data for Distributed Multimedia Systems

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

Abstract

As network technology provides the capability to handle multimedia traffic and the demand of multimedia services increases, protocols are required for effective communication of multimedia data in a distributed environment. Synchronization is one of the key issues in a multimedia system. Most of the current approaches do not support an integrated solution to the problem of synchronization. In this paper we propose a mechanism for synchronization of multimedia data in distributed environment where the accuracy of the protocol can be tailored to the application. The system model supports live and video-on-demand service. We present a scheme where the specification of the temporal requirements provided by the application can be directly mapped to obtain the information necessary to enforce the synchronization required. We present two examples of specifying the temporal requirements and process of obtaining the information and present performance results of our simulation studies.

1. Introduction

One characteristic of multimedia information systems is the need to compose data of various types and origin for presentation, storage and communication [LiGa90]. Data may be stored locally in singular database or remotely in distributed database servers. One of the greatest challenge ensuing from such a scenario is the need for synchronization to realize successful retrieval, communication, composition and presentation of multimedia objects.

Synchronization refers to making things happen in certain time order, or more specifically, coordinating the real-time presentations of information and maintaining the time-ordered relations among component media [QaWo93]. A specification model is needed that describes the objects completely by taking into account all the temporal relationships. Examples of such a model include the Object Composition Petri Net (OCPN) model proposed by [LiGa90].

Synchronization can be categorized into two types: intramedia synchronization and intermedia synchronization. Intramedia synchronization is concerned with delivering each object in time to meet the respective playout deadline. Policies used to ensure this include the generation of

a transmission schedule by deriving transmission deadlines from the playout deadlines after taking network and other delays into account [QaWo93]. Inter-media synchronization is required where some temporal relationship exists between two objects in a multimedia object which is required to be maintained. An example is lip-sync between an object of type audio and an object of type video. In this paper we shall be dealing primarily with intermedia synchronization issues.

The problem of maintaining synchronization among related multimedia streams is quite challenging in a monolithic environment. In a distributed scenario a number of parameters add up that make the problem even more tough; these include the network jitter, absence of a single clock and the difference in the clock speeds of the various devices at the destination. A comprehensive solution to this problem would call for a mechanism for specifying the temporal requirements at the application level. This specification model should be directly mapped to the generation, communication and the control of synchrony of the multimedia data.

Among the recent efforts that have been made to address the problem of synchronization is the scheme of multiplexing the media elements over the same virtual circuit connection [LeBa90]. It has the advantage of being simple to implement but has the disadvantage of hindering communication efficiency. Furthermore the approach is possible only if all the media streams are transmitted by a single source to one destination. The synchronization channel scheme described in [Shep90] uses an additional channel to carry the synchronization information and as a result the data need not be modified. However, it constitutes the overhead of an additional channel and messages. Another approach is to provide elastic buffers as suggested in [ShHu92]. An elastic buffer provides flexibility according to the burstiness of the stream. This scheme does not require synchronization markers since synchronization is done at the buffer level. However, the scheme has the drawback in that the destination should prepare sufficient buffer space on demand without having a priori knowledge of the maximum buffer size. In the method suggested by [LiGa91], the buffer requirement for each stream at the destination is estimated, and the estimation is used as a parameter for the communication connection.

Steinmetz [Stei90] and Little and Ghafoor [LiGa90] have discussed methods for formally describing the synchronization requirements in a multimedia environment. Steinmetz discussed the characteristics of a multimedia system and presented a set of constructs for expressing inter-media relationships; Little and Ghafoor evinced 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 [Hoep92] explored the synchronization of multimedia objects for presentation based on the Petri net model. Anderson and Homsy [AnHo91] described algorithms for synchronization among interrupt-driven media I/O devices, but they are only applicable to single site multimedia workstations. Nicolaou [Nico90] attempted 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 [EsDe92] presented an adaptive flow synchronization protocol that permits synchronizing in a distributed environment. The protocol however, operates under the assumption that global synchronized clocks are present which is not a very practical assumption considering that ATM networks will not provide a global synchronized clock [Rang92].

Rangan et. al [Rang92] proposed a feedback technique to detect asynchronization among the media streams in a 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 - that is the accuracy of detecting the asynchrony - is however bounded by the network jitter. In wide area networks in particular the jitter may be greater than the acceptable asynchrony among the media streams, thereby restricting the purview of the mechanism.

All the approaches mentioned above suffer from at least one or both of the following drawbacks.

- The approaches proposed do not support an *integrated solution* to the problem of synchronization. That is, schemes that allow the synchronization requirements among the media objects to be specified at the application level which may be automatically be translated to the generation, storage, communication of the data are not present.
- Secondly, the schemes for synchronization do not adapt to the application. In the face of the wide range of applications that multimedia systems are expected to reach, ranging from real-time remote surgery to entertainment applications, we believe that the accuracy of the protocols in detecting and resolving the asynchrony should be dependent on the application at hand. Though it may be argued that attempting to achieve high accuracy would be desirable irrespective of the application, greater accuracy would also mean greater demand for resources.
- Furthermore, most of the schemes for enforcing synchrony do not consider all the temporal relations that may exist among the media streams and hence the inability to incorporate synthetic media data.

The original OCPN model proposed by [LiGa90] was a model for specifying temporal requirements at the presentation level only. It could not be directly mapped to the transmission of data and the controlling of asynchrony among the media streams. An extended model (XOCPN) has been proposed by [QaWo93] that specifies the communication requirements and describes the communication and synchronization operations needed for retrieval, composition and presentation of multimedia objects. This approach is based on assigning synchronization interval units which are marked with synchronization interval number. These sequence numbers are used at the destination by the application to determine the asynchrony among the media streams. This model is highly restrictive in that it assumes that all the media streams arrive at a single destination. This may not be true for a wide range of applications as teleconferencing. Furthermore, this approach imposes the burden of synchronization on the receiver which may not be desirable.

In this paper we discuss a new model for guaranteeing synchronization among media streams where the accuracy of the scheme can be varied according to the application. We show how the specification of the temporal requirements at the application level are directly mapped to the actual generation, storage, retrieval and communication of data so as to maintain synchrony among the related media streams. The model offers support to live and video-on-demand service, does not assume the presence of a single global clock in a distributed environment and supports both natural and synthetic data. We also discuss the performance of our scheme based on the simulation results.

The rest of the paper is organized as follows. In Section 2 we discuss our scheme of generating, storing, communicating and eventually detecting the asynchrony among the media streams.

In Section 3 we discuss the different types of temporal relations that may exist among two or more time intervals. In Section 4 we discuss the OCPN model and illustrate how the model may be directly mapped to our synchronization scheme. We present two examples in Section 6 and discuss the performance of our scheme in Section 7. We conclude in Section 8.

2. Synchronization Model

The system model for the on-demand multimedia server comprises of a multimedia server (MMS) connected to the display units via a network [Rang92]. The various sites that generate the data to be stored in the MMS may also be connected over the network resulting in the system model of figure 1. The data stored in the multimedia server is retrieved on demand for display at the destination sites. The synchronization requirement is that the data at the destination sites should be displayed in the same time order as it was generated at the sources.

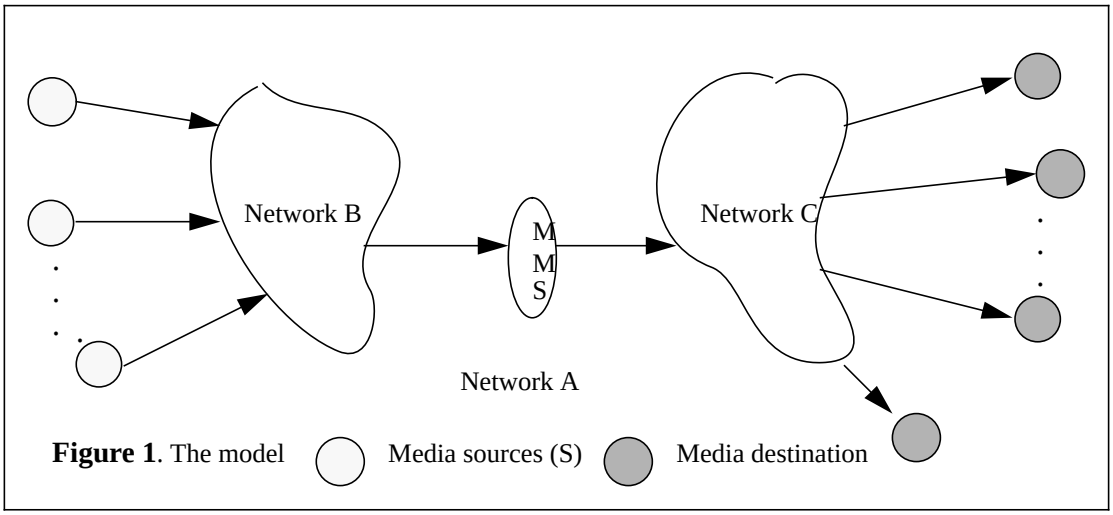

Figure 1. The model

We approach the problem of synchronization in four phases: determination of normalized clock times, the assignment of normalized relative time stamps to the media units, the detection of asynchrony among the media streams and the resolution of asynchrony if any among the media streams. In this chapter we describe each of these processes. Though our schemes enforces synchronization of all possible temporal relations, for the ease of understanding in this chapter we focus primarily on the equal temporal relation. Discussion of other temporal relations is done in Section 3.

We distinguish between our approach and the approach adopted by Rangan [Rang93] as follows. Rangan adopts a scheme where the time the media units are generated at the source and subsequently displayed at the destination is predicted. This mechanism seems quite pragmatic in a distributed clock environment [Rang93]. However, the accuracy with which the relative timestamps are assigned is dependent on the network jitter and if the requirements for synchronization

for an application be stringent or we were to operate in a wide area network environment where the network jitter is quite large, the approach would not be useful. The scheme we propose however, is based on the normalization of clock times. The accuracy of assigning the relative time-stamps and eventually controlling the synchronization is a function of three parameters (discussed below). Altering any of these parameters would alter the accuracy of assigning the relative time-stamps and may be acclimatized to the application at hand. Furthermore, our scheme overcomes all the limitation mentioned is Section 1 that are associated with the current protocols.

2.1 Determination of Normalized Clock Times

Normalization of clock times is a process carried out by the MMS to establish a relation between the clock of the MMS and the clocks of the sources (destinations). This process is carried out during the assignment of normalized relative time-stamps and while detecting the asynchrony. The process of determining the respective clock times is carried out as follows.

A network session with a jitter bound is established from the MMS to the various sources S1, S2,..., Sn. A trigger packet is sent from the MMS to the various sources simultaneously and after an interval t another trigger packet is sent. On receiving the trigger packet the various sources sent their respective clock times to the MMS. The respective clock times are normalized to the MMS clock and the times thus obtained are referred to as the normalized clock times. The process is explained below.

Let x_0 and $x_0 + t$ be the instants with respect to the clock of the MMS at which the two trigger packets are sent by the MMS and y_0 and $y_0 + t + p$ be the instants at which these two trigger packets arrive at the source with respect to the clock of the source. Then an instant x' according to the clock of MMS will correspond to the instant y' with respect to the clock of S1 where

$$y' = \frac{(t+p)}{t} \times (x' - x_0) + y_0 \qquad (4.1)$$

The above equation would be correct if both the trigger packets experience the same delay in reaching the source from the MMS, that is, if there is no jitter in the network session. Since it is not practical to assume zero jitter, if δ_d be the jitter bound on the established, we obtain the maximum error or deviation from Equation (4.1) that may be introduced due to the jitter as:

$$\varepsilon = \frac{2\delta_d (x' - x_0)}{t} \qquad (4.2)$$

If different QOS sessions are established from the MMS to the various sources and the jitter bound of the different sessions is different, i.e. the network jitter of the sessions from S1 to MMS is different from the network jitter of the session from S2 to the MMS and so on, then the error is given by

$$\varepsilon = \frac{2 \times max(\delta_d)\ (x' - x_0)}{t} \qquad (4.3)$$

where $max(\delta_d)$ denotes the maximum of the jitter bounds among the sessions established. After receiving the response of both the trigger packets, the session is terminated, because this session, in which the timing information is obtained should have a very low jitter bound and is hence

expensive in terms of the resource utilization.

2.2 Assignment of Normalized Relative Time-Stamps

Of the various approaches for synchronization, we adopt the marker model owing to its simplicity and the advantage it offers for data storage. As media data is generated at the sources, *markers* are assigned at regular intervals to the media data which reflects the time they were generated at the sources. It may be observed that markers being inter-spread at equal time intervals need not necessarily correspond to equal amount of data being inter-spread between the markers. The primary reason for this is that many of the compression algorithms used to compress media data are sensitive to the content of the data. And hence different images which are of the same time interval may have different sizes in compressed form. By assigning the markers, a relative time order is established among the various media units and we refer to these markers as the *normalized relative time-stamps (NRTS)*. It is this relative time order that should be preserved among the media data from the source to the destination to meet the requirement of synchronization. The normalized relative time-stamps are distinguished from relative time-stamps in the manner in which these time-stamps are assigned. The process of assigning time-stamps would be very straight forward if we were to operate under the assumption of a single global clock, as the relative time-stamps could correspond to the actual clock times of the sources. However this assumption would make the model highly restrictive.

The aim in assigning RTS is to assign the same RTS to the media units in different media streams that observe the desired temporal relation at a particular instant of time. In this section we restrict the discussion to the “in-parallel” or the equal temporal relation. Thus we wish to assign the same RTS to the media units that are generated at the various sources at the same time instant. The other possible temporal relations are discussed in Section 3.

Consider the scenario depicted in figure 2 where media streams being generated at sources S_1, S_2..., S_n are stored at the multimedia server (MMS) along with the respective RTS. The protocol of assigning the RTS is executed in two phases. The first phase is to establish a relation between the various source clocks and the clock of the MMS which is achieved by the process discussed in Section 2.1. The second phase is to send the media data from the sources and assign the appropriate RTS at the MMS.

After the normalized clock times have been obtained, a new session with no restriction on the jitter bound is established between the media sources and the MMS. The media packets are then time-stamped by the respective sources and sent to the MMS. On receiving the media packets the MMS normalizes all the clock times using equation (1) to its own time reference. It can then be determined with an accuracy of ε if the media units were generated at the same instant at the various sources and if they should be assigned the same RTS. Note that time-stamps are the actual clock times according to the respective clocks at which the media packets are sent by the different sources while relative time-stamps represent an order among the different media units and are independent of the physical time.

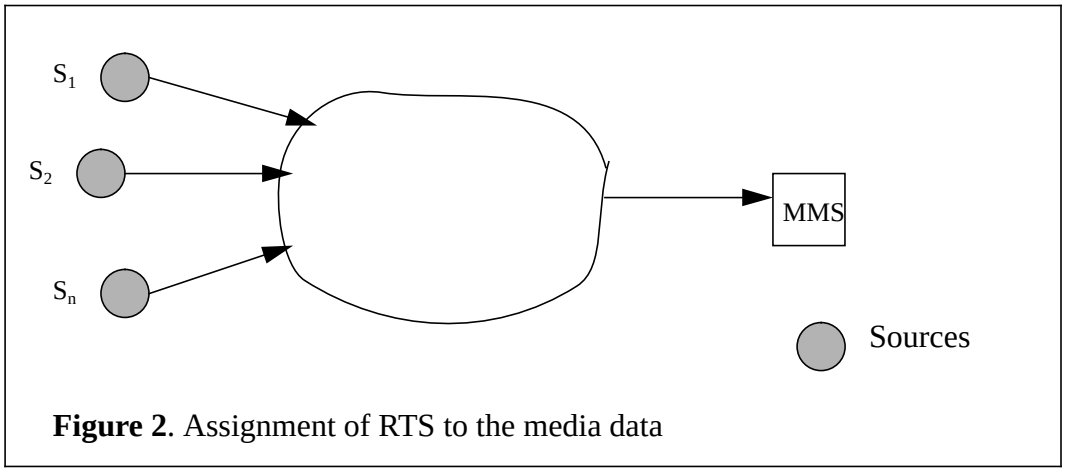

Figure 2. Assignment of RTS to the media data

The maximum error ε that may be introduced in a packet depends upon the time at which the previous trigger sequence had been generated. Later a media unit is sent out by the media sources *after* the termination of a trigger sequence, greater is the maximum error that may be introduced. Hence to ensure a low value of error, the trigger sequences are sent periodically, that is the process of obtaining the clock relationship is carried out periodically. It is evident from equation (2) and the periodicity of determining the clock relationships that the error is a function of three variables namely, the network jitter, the periodicity p with which the clock equations are obtained and the interval t between the sending of the first trigger packet and the second trigger packet. The error is directly proportional to the network jitter but inversely proportional to p and t. Hence altering any of these three parameters can control the accuracy. This renders the scheme very flexible, because depending upon the environment the appropriate parameter can be altered. For instance, if the jitter bound on the network session established is relatively high but the session can be retained for a longer period then t and p may be increased, leading to the same accuracy as a session with a lower network jitter bound and a shorter period.

The scheme has the advantage of being very adaptable, very simple to implement and constitutes very little overhead. Though our protocol has some characteristics of a synchronization protocol it is different in the following ways. Our protocol incurs very minimal overhead at the receiver. The only function that is expected of the receiver is to sent back the times at which they receive the RTS. The communication overhead is also very minimal. The only traffic besides the data are the trigger packets and the feedbacks from the receivers both of which carry very little amount of data. Our protocol unlike any other synchronization protocol regulates the accuracy of synchronization based on the needs of the application. It hence balances the resources required for ensuring synchronization and the accuracy necessary for the application. The drawbacks that are generally associated with a complex synchronization protocol [Rang 93] are hence dispensed.

2.3 Detection of Asynchrony

The media data stored at the MMS would be required to be displayed at the destinations at

some instant of time. The challenge is to playback the various media streams at the destination such that the various media units with the same RTS observe the temporal relation (that is media units with the same RTS are played simultaneously) - cases besides the equal temporal construct are discussed in section 3. To accomplish this, the process of normalizing clock times coupled with obtaining timed feedbacks from the media destinations is adopted.

Initially a quality of service session that has a low network jitter bound is established between the MMS and the various destinations. Two trigger packets separated by a time interval t are sent by the MMS to the various destinations simultaneously. The destinations on receiving the trigger packets send back their respective clock times to the MMS. Using equation (1), MMS normalizes the various clock times to its own time reference. The QOS session is then terminated and media data is sent to the display units. On receiving the media data the destination sites send back a feedback to the MMS that contains the time at which the media unit was displayed. The MMS can determine with an accuracy ε if the media streams are in synchronization. This is accomplished observing whether or not the RTS of the same value from different media streams are displayed at the destinations at the same instant. If not synchronized, necessary action is taken to resynchronize them. The error ε is the maximum error that may be introduced. The actual value of the error that is introduced (e) may range from a minimum of 0 to a maximum of ε. Owing to this *range* of error that is introduced we cannot for a certain interval determine if the streams are in synchrony or not. Specifically if z_1 and z' be the time instant, according to the clock of the MMS, of the data units of two streams reaching the respective destinations and ζ be the tolerable asynchrony, then

- Synchrony among the media streams is guaranteed if $|z_1 - z'| < |\zeta| - |\varepsilon|$.
- The streams are necessarily out of synchrony if $|z_1 - z'| > |\zeta| + |\varepsilon|$. (4)
- Synchrony may or may not exist for other times.

These cases are reflected in figure 3.

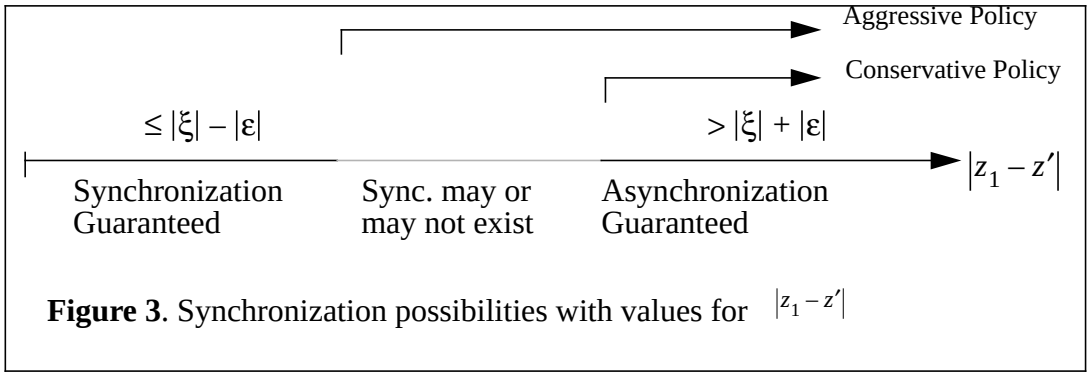

Figure 3. Synchronization possibilities with values for $|z_1 - z'|$

The policies used to trigger the synchronization mechanism if the streams are out of synchrony may be conservative or aggressive [Rang 93]. Conservative policies trigger the synchronization mechanism only if $|z_1 - z'| > |\zeta| + |\varepsilon|$, that is when it is assured that the streams are out of synchronization. Aggressive policies trigger the synchronization mechanisms whenever there is any possibility of the streams being out of synchrony, that is whenever synchrony is not assured.

3. Other Temporal Relations

Synchronization refers to making things happen in certain time order [Stei 90]. There are thirteen ways in which two time intervals may be interleaved of which 6 are inverse relations of each other [Hamb72], [Litt90]. For instance the relation *before* is the inverse relation of the relation *after.*

Most of the protocols for the synchronization of multimedia streams that have been discussed in the literature support the “in-parallel” or the “equal” temporal relation. Though it may be argued that other temporal relations can be expressed in the form of the “equal” relation, we believe that only a good appraisal of the other temporal constructs would enable an efficient solution for synchronizing them [SoAg93]. We shall illustrate how the other temporal relations may very efficiently be incorporated into our model.

One of the advantages that stems from introducing other temporal relations is its support to synthetic media data. A virtual reality application may for instance require that a certain audio sample be triggered *x* time units after the a video clip. The application places no constraints on the time at which the video clip commences. This type of a requirement we observe directly translates to the *after* temporal relationship.

The proposed incorporation of the other temporal relations would require some subtle modifications both in our approach of the assignment of the time-stamps and the detection of asynchrony. Unlike the conventional case where the time-stamps are assigned only at the front end, in our approach relative time-stamps are assigned both at the rear and the front end of the media unit. We define the *rear end* to be the right side of the media unit and the *front end* to be the left hand side of the media unit. So if the RTS of a media unit be *x* then the RTS *x* is placed both at the rear and the front end of the media unit. The process of detecting the asynchrony is analogous to the discussion in Section 2.2. The difference is that the destinations send back the respective clock times both when the rear or the front RTS arrive at the destinations. The decision whether the clock times associated with the arrival of the rear RTS or the front RTS is considered in the determination of asynchrony by the MMS depends on the temporal relation that exists among the media streams. For instance, if the temporal relation is A *meets* B then the arrival time of the front RTS of A and arrival time of the rear RTS of B are considered. These values may be substituted for z_1 and z' respectively in equation (4) to ascertain if the media streams are out of synchrony. Similar treatment can be met to the other temporal relations too. For the relation A *overlaps* B shown in figure 4, the arrival time of front RTS of both the A and B media units is considered. The left hand side of equation becomes $|z_1 - z' - t_i|$ where t_i denotes the time interval expressed in the relation *overlap*. The information regarding the temporal relations among the media streams is passed on to the MMS by the specification model provided by the application. We discuss the details of the specification model in Section 4. The reason that both the RTS are sent back and the decision of choosing one of them is left to the MMS is to ensure that the display units need not be concerned with the protocol and may all function in a uniform manner.

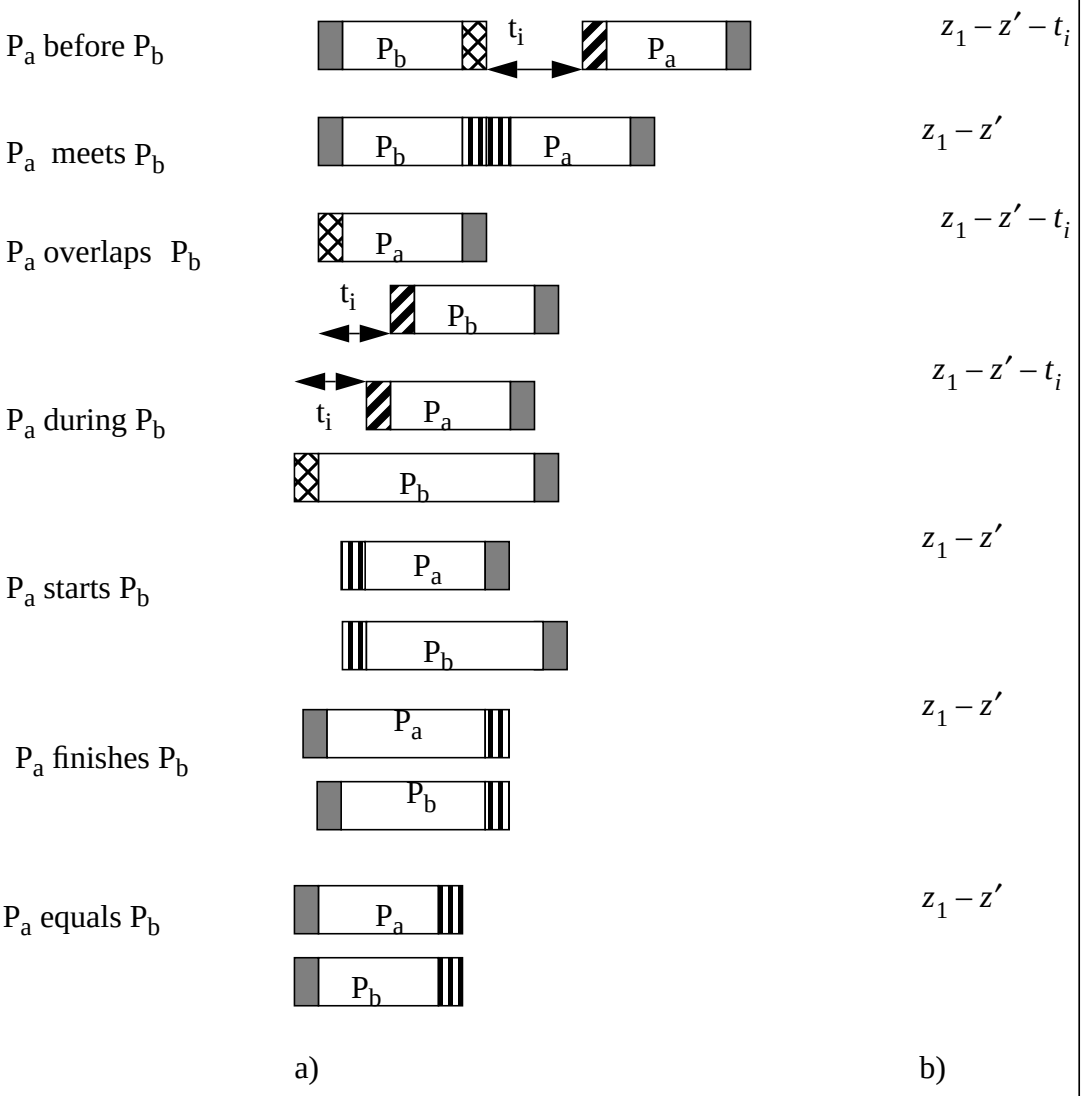

Figure 4. a) Temporal relations b) Equations corresponding to LHS of eq. (4)

4. Specification Model

In this section we discuss a model of specification and show how temporal relations expressed in this form of specification can be used to control the asynchrony among the different media streams.

We choose the OCPN model suggested by [LiGa90] for specifying the temporal relations among the various media objects. We believe that the model is powerful enough to effectively

specify any temporal relations that may exist among the various media streams. Furthermore, some of the recent work indicates that the OCPN model can be enhanced to facilitate modelling of multimedia synchronization characteristics with dynamic user participation [PaRa93]. We give a brief description of the Petri net and the OCPN model as proposed in [LiGa90].

The Petri net is defined as a bipartite, directed graph N = (T,P,A) where

$T = \{t_1, t_2, t_3, ..., t_n\}$
$P = \{p_1, p_2, p_3, ... , p_n\}$ and
$A : \{T \times P\} \cup \{P \times T\} \rightarrow I$
$I = \{1, 2, ...\}$

T, P and A represent a set of transactions, a set of places and a set of directed arcs, respectively [HoVe85]. A marked Petri net model N_M = (T,P,A,M) includes M where M assigns tokens to each place in the net.

$M: P \rightarrow I$,
$I = \{0,1,2, ...\}$

The Object Composition Petri Net (OCPN) augments the conventional Petri net model with values of time, as durations, and resources utilization on the places in the net [LiGa90]. The OCPN is hence defined as C_{OCPN} = {T,P,A,D,R,M,} where

D: $P \rightarrow R$ and
R: $P \rightarrow \{r_1, r_2, r_3, ... , r_k\}$

D and R are mappings from the set of places to the real numbers (durations) and from the set of places to a set of resources respectively. Also, associated with the definition of the Petri net is a set of firing rules that govern the semantics of the model.

It has been shown that given any two atomic processes specified by temporal intervals, there exists a Petri net (OCPN) representation for their relationship in time [LiGa90]. For example the Petri net for the temporal interval overlaps is shown below.

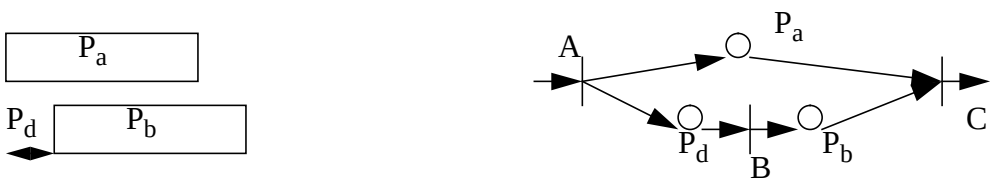

Figure 5. A Temporal Relation and the corresponding Petri net.

The above specification of the Petri net model would correspond for instance, to an application where the display of the audio data occurs after P_d time units after the display of the video data. It can be shown that any temporal requirement can be expressed in the OCPN representation. We shall present more examples of OCPN representations in Section 5.

4.1 Determination of feedbacks

From the discussion in Section 3 we observe that the primary difference in dealing with different types of temporal relations is in deciding the RTS whose feedback should be considered by the MMS. For instance in the *during* relation the feedbacks of the front markers of both the media streams needs to be considered by the MMS, while in the relation *before* the feedbacks of the front RTS of one media stream and the rear RTS of the other media stream need to be considered. Given temporal relations that exist among media streams, the challenge is hence to communicate to the MMS the feedbacks that it should consider in different media streams. This information can be obtained from the specification provided by the user. In this section we propose an algorithm to determine the feedbacks to be considered by the MMS, given the OCPN specification provided by the user.

When we traverse an OCPN model from a transition (bar) A to a transition B, we define present_place to be the circle that we just traversed through. In figure 5, on reaching transition B (from A) the present_place would be P_d. next_place is defined to be the place (circle) that would be traversed if we were to traverse the arc emanating from the present transition. This circle should be uniquely identified if there is more than one arc emanating from the transition. Each place in the OCPN model has two components - the name of the circle as P_i and the value of the circle which represents the time interval that the circle represents. The value of the circle P_i is denoted by val(P_i). *E* is an empty data unit which is introduced in an OCPN model to introduce some delay. t_i is the variable referred in Section 3. The algorithm used to determine the feedbacks to be considered is given below.

```
if at a transition only one outgoing arc
  move to the next transition
  II_marker := rear of present_place
  move to the next transition
  if present_place ≠ E
    ti := 0                        /* meets */
    I_marker := front of present_place
 else ti := val(E)
      I_marker = front of next_place    /* before */

else     /* if more than one outgoing arc */
if neither next_circle == E
   I_marker := front_Pa
   II_marker := front_Pb   /* equal or start relation */
   ti := 0
else
   ti := val(E)
   move to the next pair of transitions    /* during or overlap relation */
   I_marker := front of present_place (Pm) which is not E
   II_marker := front of the next_place (Pn) which is E
```

```
if val(Pm) == val(Pn)+val(E) then
 ti := 0       /* finish relation */
 I_marker := rear of present_place
 II_marker := rear of next_place
```

The algorithm as described above has two restrictions. Firstly it deals with an OCPN model where there are at most two arcs emanating from any transaction. Secondly the model as described does not discuss the case where we begin traversing the OCPN model from a state different from the start state. We define a transition A to be a *start state* if there is no place preceding the start state (as in figure 5).

To account for the first restriction and to extend the algorithm for any OCPN we use the subnet replacement strategy as mentioned in [LiGa90]. Let O_1 be an OCPN with three arcs emanating from the start transition A leading to the places P_1, P_2 and P_3. The OCPN O_1 is replaced by OCPN O_2 where there are two arcs out of transition A leading to subnet S_1 and S_2. The subnets S_1 and S_2 each contain two out of the three places. This approach is similar to the bottom-up approach often used in software development.

We shall illustrate this concept in an example in Section 5. The solution to the second limitation is straight forward. If we begin traversing the model from a transition A and there exists a transition A' such that there is an arc from A' to A (in other words there is a place P_x between A' and A) then the rear marker of P_x is also considered.

Following the above rules, the appropriate feedbacks to be considered for any given Petri net can be ascertained.

5. Examples

In this section we present two examples which illustrate the schemes presented in the preceding sections. We first present the temporal requirements and show how the feedbacks that need to be considered by the MMS can be determined from the specification. We then illustrate how this information is used by the MMS to control the asynchrony among the media streams.

Let A_1, A_2 and A_3 be three media streams which are being generated at three sources in a scenario as depicted in figure 1. The relation between A_1, A_2 and A_3 is shown in figure 6a. As the media units of the three media streams arrive at the MMS they are assigned the relative time stamps at the rear and the front end. The media units are then sent to the destination where the arrival times of the rear and the front RTS are sent to the MMS. Let A_i^r and A_i^f denote the rear and front RTS of a media unit of the stream A_i respectively. t_i^r and t_i^f denote the arrival time of the rear and front of the media unit of stream A_i respectively. n_i^r and n_i^f denote the normalized clock times that are obtained corresponding to t_i^r and t_i^f respectively, using equation (1). Let the maximum tolerable asynchrony for the application be m.

Streams A_1 and A_2 observe the *start* relationship and A_2 and A_3 observe the *finish* relationship. These requirements may be expressed by the OCPN shown in figure 6b. The OCPN of figure 6b may be reduced to the OCPN shown in figure 6c, where subnet 1 contains the places A_1, E and

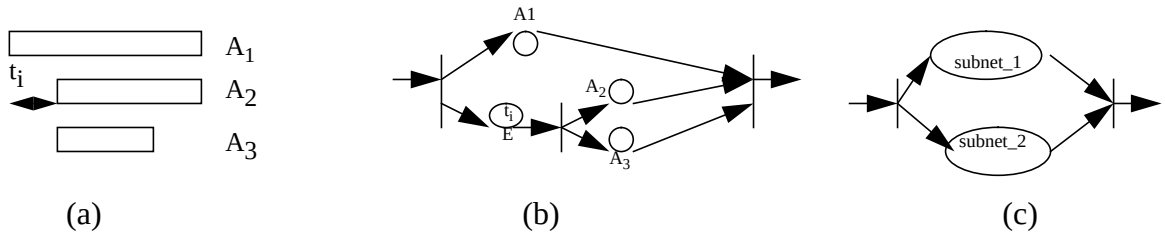

Figure 6 (a) A temporal requirement (b) Specification of requirement (c) Reduction of problem

A_2 and subnet 2 contains the places A_2 and A_3. Using the algorithm given in Section 4, it can be determined that the feedbacks that need to be considered are those of the rear RTS of A_1 and A_2 and the front RTS of A_2 and A_3. The times associated with these feedbacks are t_1^f, t_2^f, t_2^r and t_3^r. The normalized times that would be obtained are hence n_1^f, n_2^f, n_2^r and n_3^r. If ε be the maximum error that may be introduced as explained in Section 2.1, the equations obtained for detecting the asynchrony are:

$$\left|n_1^{\,r} - n_2^{\,r}\right| > |m| + |\varepsilon| \quad \text{and}$$

$$\left|n_2^{\,f} - n_3^{\,f}\right| > |m| + |\varepsilon| \,.$$

If the above equations are true, the streams are necessarily out of synchrony and the remedial action would be required. The action may be to drop frames of one of the streams or to duplicate frames of the other stream. If the above inequalities are not true then the following equations are checked to see if the streams are in synchrony.

$$\left|n_1^{\,r} - n_2^{\,r}\right| < |m| - |\varepsilon| \text{ and}$$

$$\left|n_2^{\,f} - n_3^{\,f}\right| < |m| - |\varepsilon|$$

If the above inequalities do not hold either then it may not determined with certainty if the streams are in synchrony or not, as explained in Section 2.2.

The second example we present is from the requirements of one of the interfaces of the multimedia information system that we designed for the department of echocardiography of the University of Virginia medical center [AgSo93]. For this example we only describe the process of finding the appropriate feedbacks. The process of detecting the asynchrony after the feedbacks have been obtained is similar to explanation given for the previous example. The interface consists of a video display showing the motion of the heart of a patient. The motion is captured from different views and each view is played for a predetermined period. Corresponding to each view is a still image and some text which are to be displayed for a certain time interval. An audio stream which corresponds to the annotation of a physician and a text message are displayed throughout the period the video is displayed.

The above requirements correspond to the temporal relations shown in figure 7. The specification of the these temporal requirement can be specified by the application using the OCPN model as shown in figure 8.

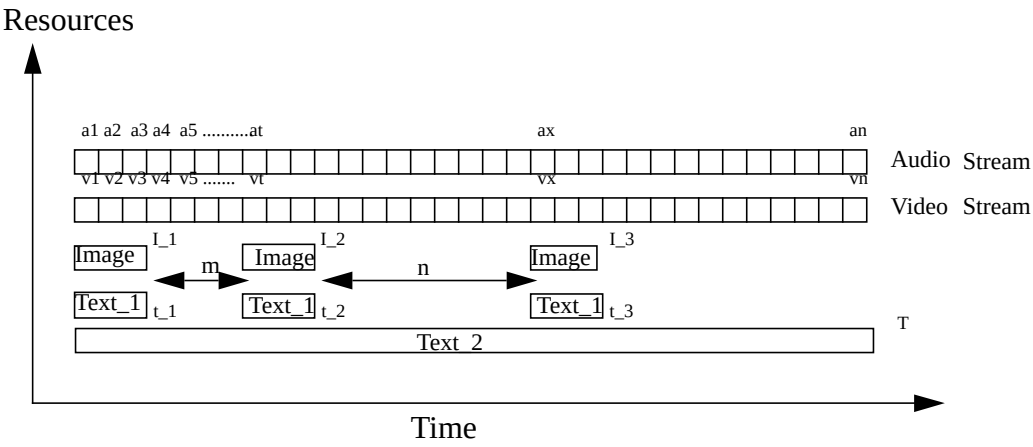

Figure 7 Temporal requirements for a multimedia medical information system

The relationships shown in figures 7 and 8 can be divided into intermedia relationships and

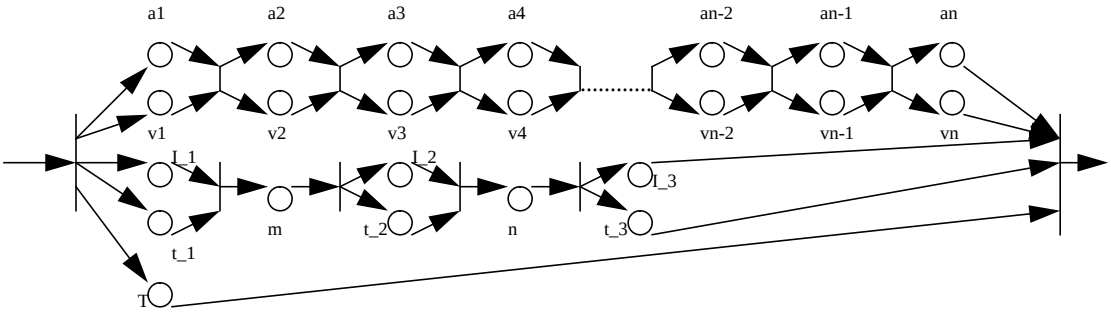

Figure 8 Specification of the requirements expressed in figure 7.

intra media relationships. Intermedia relationships deal with determining the feedbacks which need to be considered by the MMS. Considering the intermedia relationships we have the following relations: (a_1, v_1, I_1, t_1, T), (a_t, v_t, I_2, t_2), (a_x, v_x, I_3, t_3), (a_2,v_2), (a_3,v_3), (a_4,v_4) ..... (a_n,v_n). All of these observe the start relationship. Following the algorithm in Section 4, it is found that the feedbacks to be considered are the front RTS of the media units that appear in the relations above. Having found the feedbacks which need to be considered by the MMS, the MMS can determine if the media units of the different relations maintain the synchrony. The process is similar as described for example 1.

The relations for the intramedia synchronization are: (a_1,a_2), (a_2,a_3) .... (a_{n-1}, a_n), (v_1,v_2), (v_2,v_3) .... (v_{n-1},v_n), (I_1,I_2), (I_2, I_3), (t_1,t_2), (t_2,t_3). A step toward achieving this goal is to generate a transmission schedule by deriving transmission deadlines from the playout deadlines after taking the network and other delays into account [LiGh91]. To compensate for anomalies, appropriate handling schemes are required at the destination.

6. Extension of the System Model

The system model discussed so far portrays a scenario where data is sent across the network and is stored in a multimedia server to be later retrieved for display at the destination sites and it is required that the synchrony among the media streams be maintained during the display. This model can be efficiently extended to support synchronization of live multimedia data transfer across the network, where data storage is not involved.

Consider figure 9. The sources A_1, A_2,..., A_n send media data across the network to the destination sites B_1, B_2,...,B_n and we wish to guarantee synchronization among the media streams while they are displayed at the destination. The difference between this model and the one discussed in figure 2 is that in this model media data is not stored at any intermediate site. One of the merits resulting from this model is that the multimedia servers which are very expensive devices can now be replaced by a simple inexpensive computer to accomplish the same functionality. The cost associated with the MMS is primarily owing to their storage media [Rang 92].

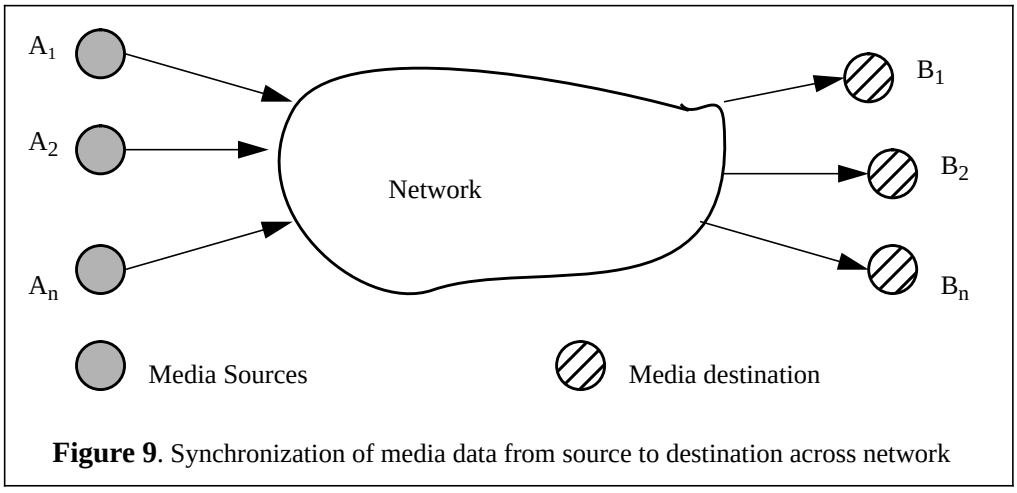

Figure 9. Synchronization of media data from source to destination across network

Consider a *node* on the network which is any simple computer. We call this node the *synchronization server* and it is designated the task of executing a synchronization protocol which is similar to the ones discussed in the earlier sections. Note that the synchronization server is different from the multimedia server in that it does not have to store the media data and since the processing of the synchronization protocol does not constitute much overhead, and the synchronization server (SS) may be any node on the network, unlike a MMS which has a very expensive storage and is dedicated to the task of managing the media data.

The synchronization scheme is carried on as follows. The media data from the different sources is routed through the network to a node which assumes the functions of a SS. The various media units are time-stamped at the sources and send to the SS across the network, and in the same manner discussed in Section 2.1, the SS assigns RTS to the media units. The media units once time-stamped are not stored at the SS but are forwarded to the destination sites. To detect asynchrony if

any at the destination sites, the SS establishes a network session with a low network jitter bound with the destination sites and obtains the clock timing. The session is then terminated and a feedback mechanism is used to detect the asynchrony. The process carried out is similar to the one discussed in Section 2.2, in context of the multimedia on-demand type of a service.

The maximum error that may be introduced either while assigning the time-stamp or while the data is delivered at the destinations is given by equation (2). The total error that may be introduced in detecting asynchrony in the process of sending the data and delivering it at the destination is hence $2 \times \varepsilon$.

A drawback which seems imminent with the above approach is that the SS may become a bottleneck. However, the overhead introduced due to the assignment of the time-stamps to the data units and the monitoring of the feedback units is not large and thus multiple nodes can be assigned the task of the SS - or may also be adaptable depending upon the load at the various nodes. The only requirement is that all the media streams that need to be *mutually* synchronized should be sent to the same SS. The architecture depicted is now similar to that shown in figure 10. It may further be pointed out that establishing a session between the SS and the media sources and the subsequent sending of data does not require that the various sessions established have the same route. Different routes with different jitter bounds may be used and the error equation may be calculated from equation (3).

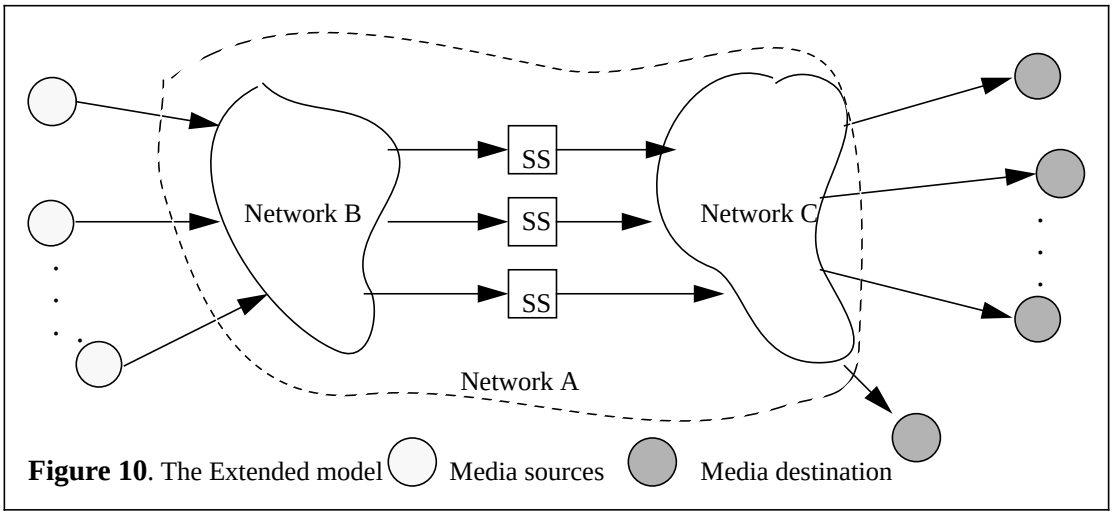

Figure 10. The Extended model

7. Simulations

The simulation of our protocol was done on a SUN Sparc station using the SES simulation environment. The primary goals of the simulation study were to verify the correctness of our scheme, to observe the effect of varying some parameters on the error that is introduced and to verify that the actual error introduced is with in the bounds that we predict.

The parameters on which the error introduced depends is given by Equation (2). These are the jitter of the QOS session, the periodicity of sending the trigger packets, and the time interval between the sending of the two trigger packets. The effect of the varying the delay of the network while the data is being transmitted was also studied.

We assumed that the bound on the network delay and the jitter bound of the QOS session followed a uniform distribution. We believe that changing the distribution will not make alter any results that we have obtained. Without any lack of generality, the data packets were generated at regular time intervals. The clocks of the MMS and the destinations ran independently and we did not assume the presence of clock skews.

In the first simulation experiment the jitter of the quality of service session established for determining the normalized clock times was varied. The trigger packets were sent after every 10 packets. We observe that the plot of the error introduced versus the packet number sent is a saw tooth shaped curve (Figure 11). The error rises steadily reaching a maximum value just before the next sequence of trigger packets is sent. Having sent the next sequence of trigger packets, the error drops steeply. This is in agreement with the performance that would be predicted as given by Equation (2). In our simulations the interval between sending the packets was a multiple of t.

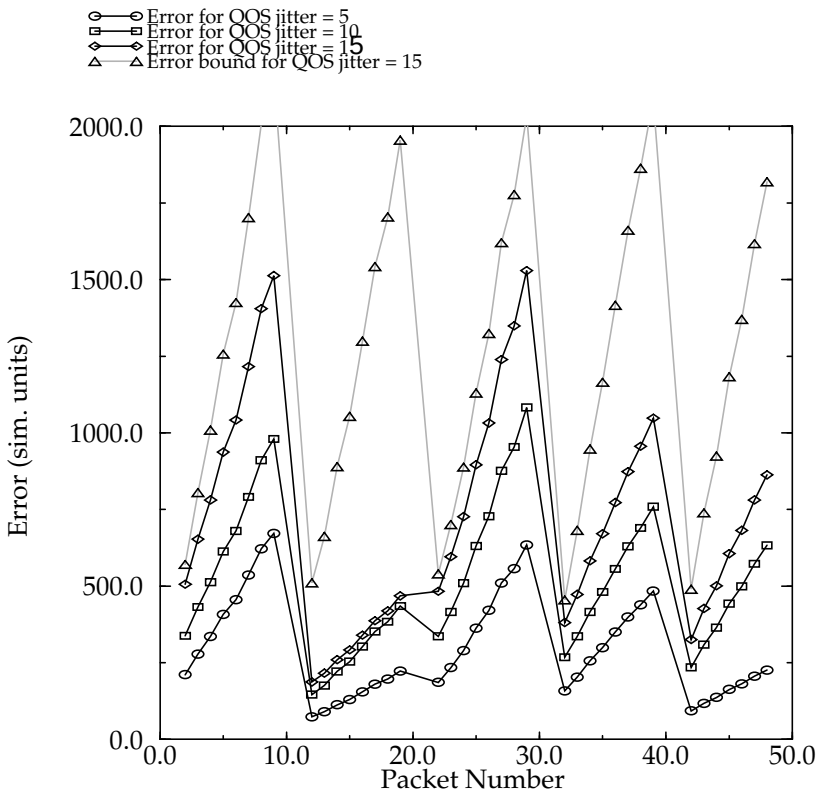

Figure 11 Error Introduced with Increasing Packet Numbers

We plot the value of the error introduced and the error bound for varying values of the jitter of the QOS session to study the impact of the QOS jitter. Both the error bound and the error introduced rise with the increase in the jitter bound. As we observe from Figure 12 the increase in the error is a linear function of the jitter of the QOS session. The plots of the error bound are much more symmetrical than the plots for the error introduced owing to the fact that the delay (and the jitter) that is introduced in the various data packets is uniformly distributed. Since the error introduced depends on the actual jitter introduced for a packet, the curves are not as symmetrical as the ones for the error bounds. It is observed that the actual error introduced is well within the error bounds predicted.

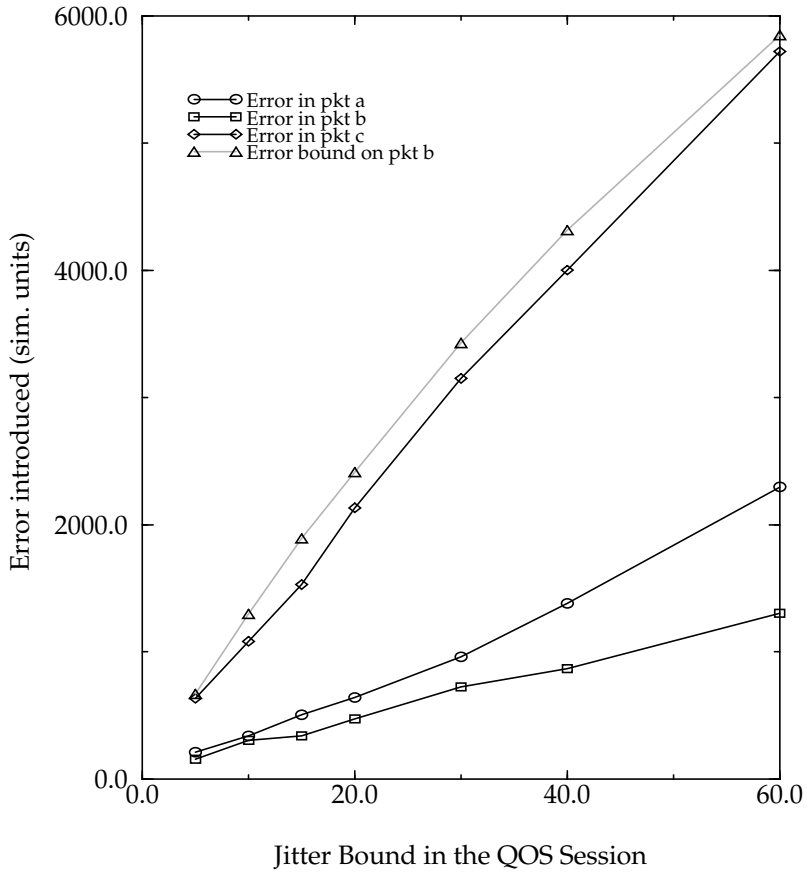

Figure 12 Effect of QOS Jitter on Error

Next the effect of altering the time interval between the sending of the two trigger packets was studied. As the time interval between the sending of two trigger packets is increased the error introduced drops. This drop in error is very marked in the initial interval and decreases for larger values of the time interval (Figure 13). This interval corresponds to t in the error equation.

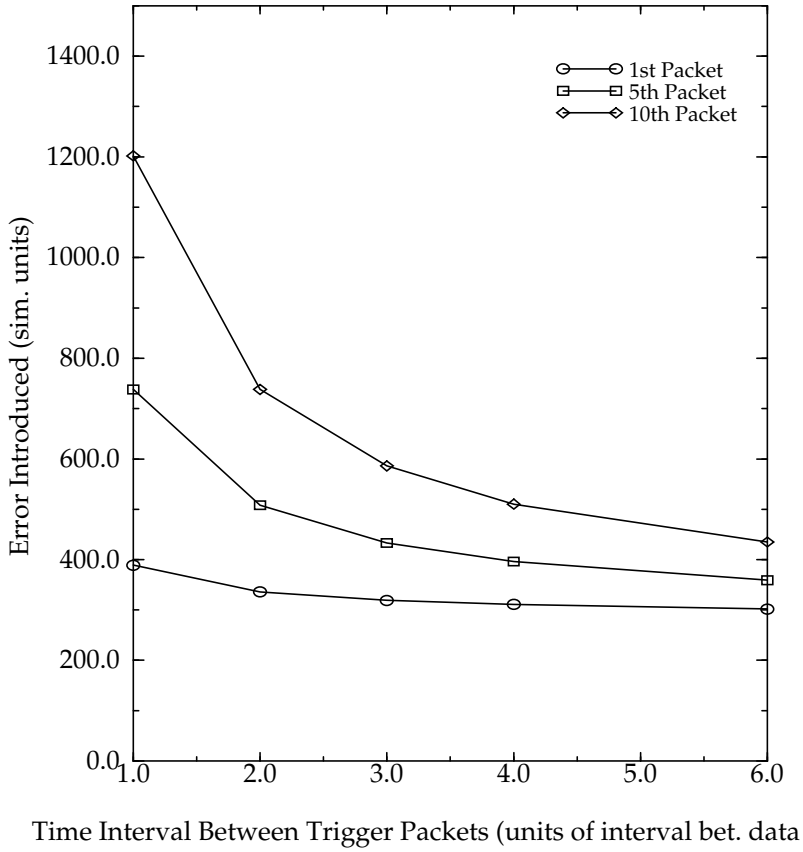

Figure 13 Effect of Time Interval on Error Introduced

The effect of varying the periodicity of the trigger interval is quite interesting. The plot of the error introduced versus a data packet does not seem to follow any particular pattern. The reason is that, as we notice from Figure 12, the error introduced in a packet is dependent upon the time relative to the sending of the last trigger sequence. A packet sent immediately after a trigger sequence would have lower error bound than a packet sent after some time after the sending of the trigger sequence. For instance if the trigger packets be sent every ten packets starting at packet number one, then the error bound for the nineteenth packet would be higher than the error for the twenty third packet. It is this observation that has to be taken into account while studying the effect of periodicity and hence a plot of error versus the modulo of the packet by the period is used. As shown in Figure 14, the error increases steadily with the increase of the period.

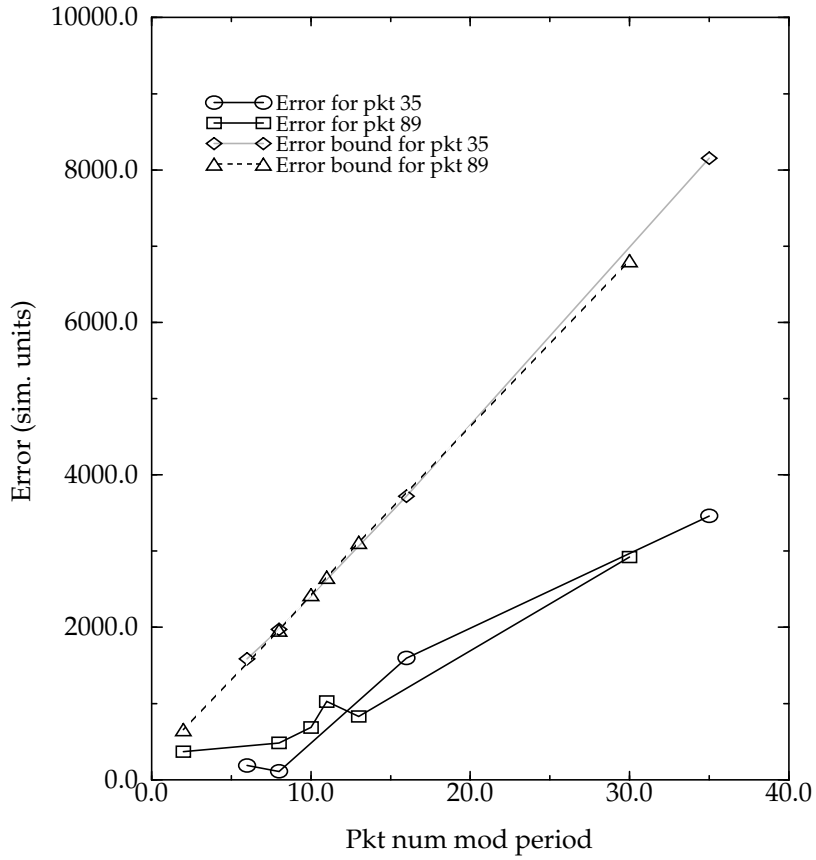

Figure 14 Effect of Periodicity on Error

Finally, the effect of the change of the delay bound on the regular network session (while data is being transmitted) is studied. Though our protocol does not assume the existence of any such delay bounds while the data is being transmitted, it is useful to study the effect *if* such guarantees could be made. It is interesting to note that the error introduced decreases with the existence/ decrease of the error bound. This may not be intuitive at the first sight. The reason for this behavior is that greater delay would mean that a packet that has been sent from the MMS to the destination would be expected to arrive later than on a session with lower network delay bound. And since the error bound is dependent upon the arrival times of the data units at the destination sites, a greater delay bound would result in a higher error bound for a packet.

The simulation results for the performance evaluation of our protocol verify the correctness of the synchronization scheme. We observe that varying any of the parameters, namely the jitter of the QOS session, the time interval between sending two trigger packets, or the periodicity leads to an appropriate change in the value of the error bound and the error that is introduced. Depending on the environment and the resource availability, the system can alter one or some combination of these parameters to obtain error bounds that may be appropriate to the application at hand.

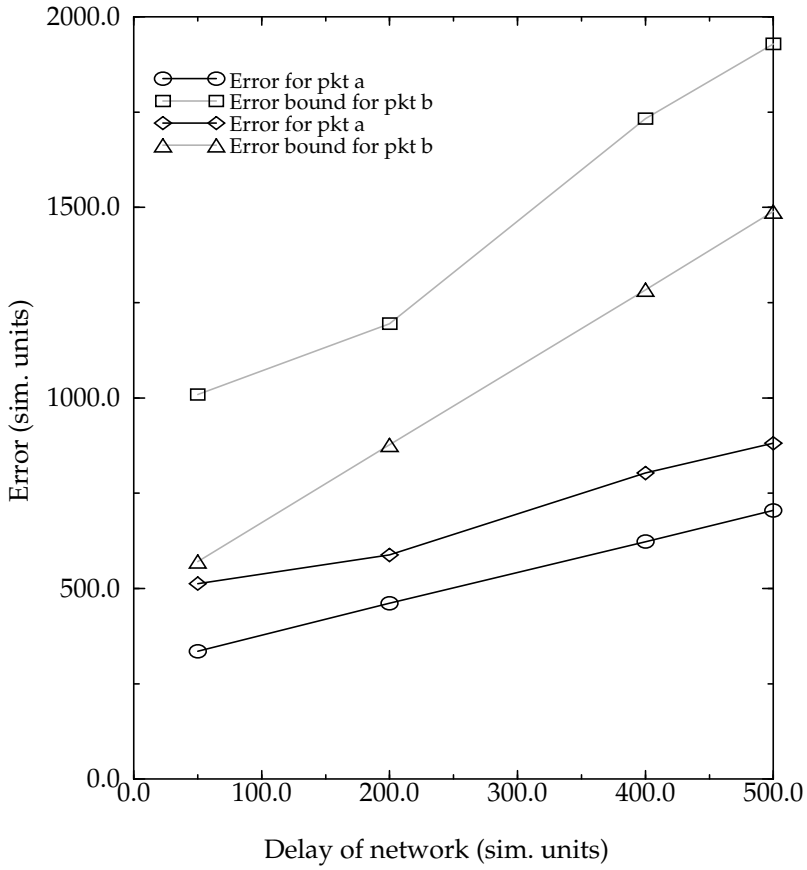

Figure 15 Effect of Network Delay on Error

8. Discussion

8.1 Clock Synchronization

It is observed that assuming the presence of a global clock would make the process of assigning the time-stamps and determining the time of arrival of the various media units at the destination very straightforward. However global clock synchronization mechanism have the following drawbacks associated with them [Rang93].

- Clock synchronization requires complex and sophisticated protocols.
- Communication overhead associated with synchronization is large.
- The clocks may belong to different domains and may not want to synchronize their clocks and may not want to incur and synchronization overhead.

Our approach for determining the normalized clock times has the following advantages over a typical clock synchronization protocol.

The complexity of our protocol is limited to the establishing the QOS session and the calculation of normalized clock times. The overhead that does occur is at a synchronization server and not at the source or the destination sites. The overhead in communication is the sending of two trigger packets and their replies; hence restricting it to four packets which carry very small amount of data. The strength of our protocol stems from the fact that depending on the applications requirements, the protocol can alter the requirements of the resources that are required to obtain the normalized clock times. This kind of an adaptive scenario is very beneficial in the area of multimedia where the requirements of the applications are expected to differ vastly [Stei93].

8.2 Comparison of Error Bounds

We mention that one of the benefits that stems from our scheme is the ability to provide with a synchronization quality of service - the ability of change the error bound depending on the application. It would however be interesting to compare the error bounds introduced by our approach and those of the existing protocols.

The error bound for our scheme is given by

$$\varepsilon = \frac{2\delta_d (x' - x_0)}{t}$$

Let the jitter bound for the regular network session be j - if such a bound were to exist. Let the ratio of the network jitter bound to the jitter bound of the QOS session be k_1, that is $j / \delta_d = k_1$ and if the time between generating subsequent media packets be *i*, then let the ratio between the generation times of the two trigger packets and i be k_2.

That is $t / i = k_2$.

If x_0 corresponds to the instant when the first packet is sent and x' the instant when the nth packet is sent, then x' - $x_0 = n \times i$.

The equation for error bound reduces to

$$\varepsilon = \frac{2jn}{k_1 k_2}$$

The error bound in [Rang92] is equal to the network jitter, it such a bound were to exist. Hence the ratio of the error bound in their case to our scheme's error bound is

$$R = \frac{k_1 k_2}{2n} .$$

9. Conclusions and Future Work

We have proposed an integrated scheme for communication of multimedia data in a distrib-

uted environment. Considering the wide range of applications multimedia systems intend to cover, the accuracy of detecting the asynchrony and the overhead incurred in our protocol can be tailored to the application. We have demonstrated how our scheme can be used to maintain the relative time-order among various streams for all the possible temporal relations. A scheme where the specification of the temporal requirements given by the application can be directly mapped to enforce the synchronization policy is presented. Our simulation results support our analytic work for the correctness of the protocol and the expected change in the error bounds with the change of parameters.

We identify some of the future work that may be pursued in the direction of our work. Our protocol is primarily geared towards ensuring inter-media synchronization. Intra-media synchronization deals with ensuring that the media units in a particular media stream observe the desired delivery behavior. The solution to the problem would involve a scheduling mechanism which takes into account the characteristic of the network and the receiver.

We have provided bounds for the error that may be introduced by our scheme and our simulations show that the actual error introduced is well within the bounds of the error predicted. It would be interesting to study if tighter bounds for a media unit can be provided on the error by observing the error characteristics of previous media units.

The periodicity of sending trigger packets is decided a priori in our protocol. An adaptive approach where the network behavior may be regularly monitored to vary the parameters set a priori and yet meet the required quality of service may be feasible.

References:

[AgSo93] Nipun Agarwal, Sang H. Son et. al., "*Model Multimedia Workstation for Echocardiography*", 17th Annual Symposium on Computer Applications in Medical Care, Washington D.C., Page 860, Oct 30 - Nov 3, 1993.

[AnHo91] David P. Anderson, George Homsy, "*A Continuous Media I/O Server and Its Synchronization Mechanism*", IEEE Computer, Pages 51-57, October 1991.

[BuLi91] 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.

[EsDe92] Julio Escobar, D. Deutsch, C. Partridge, "*Flow Synchronization Protocol*", Globecomm 1992.

[Hamb72] 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.

[Hoep92] Petra Hoepner, "*Synchronizing the presentation of multimedia objects*", IEEE Computer Communications, Vol. 15, No. 9, Nov 1992, Pages 557-564.

[HoVe85] Holliday, M.A, Vernon, M.K., "*A generalized timed Petri net model for performance*

analysis", Proceedings of Int. Conf. Timed Petri Nets, Torino, Italy, July 1985, Pages 181-190.

[LeBa90] Leung, W.H., Baumgartner, T.J., Hwang, Y.H., Morgan, M.J., Tu, S.C., "*A Software Architecture for Workstations Supporting Multimedia Conferencing in Packet Switching Networks*", IEEE Journal on Selected Areas in Communications, Vol. 8, No. 3, Apr 1990, Pages 380-390.

[LiGa90] 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.

[LiGa91] Thomas D. C. Little, Arif Ghafoor, "*A Multimedia Synchronization Protocols for Broadband Integrated Services*", IEEE Journal on Selected Areas in Communication, Vol. 9, No. 9, December 1991, Pages 1368-1382.

[Nico90] 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.

[PaRa93] Prabhakaran, B. and Raghavan, S.V. , "*Synchronization Models for Multimedia Presentation with User Participation*", Proceedings of the ACM Multimedia 1993, Anaheim, CA, Pages 157-166.

[QaWo93] Qazi, N.U., Woo, M., Ghafoor, A., "*A Synchronization and Communication Model for Distributed Multimedia Objects*", Proceedings of the ACM Multimedia 1993, Anaheim, CA, Pages 147-155.

[Rang92] P. V. Rangan et. al, "*Designing an On-Demand Multimedia Service*", IEEE Communications Magazine, July 1992, Pages 56-64.

[Rang93] 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.

[Shep90] Shepherd D., Salmony M, "*Extending OSI to Support Synchronization Required by Multimedia Applications*", Computer Communications, Vol. 13, No. 7, Sep 1990, pages 399-406.

[ShHu92] Shepherd D., Hutchinson, D., Garcia, F., and Coulson, G., "*Protocol Support for Distributed Multimedia Applications*", Computer Communications, Vol. 15, No. 6, July 1992, Pages 359-366.

[SoAg93] Sang H. Son and Nipun Agarwal, "*Synchronization of Temporal Constructs in Distributed Multimedia Systems with controlled accuracy*", Tech Rep. # CS-93-57, Dept. of Computer Science, University of Virginia, Oct 28, 1993.

[Stei90] Ralf Steinmetz, "*Synchronization Properties in Multimedia Systems*", IEEE Journal on Selected Areas in Communication, Vol 8, No. 3, Pages 401-411, April 1990.