Available online at www.sciencedirect.com

SCIENCE DIRECT®

The Journal of Systems and Software 79 (2006) 701–713

The Journal of Systems and Software

www.elsevier.com/locate/jss

An agent based synchronization scheme for multimedia applications

S.S. Manvi, P. Venkataram *

Protocol Engineering and Technology Unit, Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore 560 012, India

Received 27 October 2004; received in revised form 11 August 2005; accepted 12 August 2005
Available online 28 September 2005

Abstract

Synchronization of multimedia streams is one of the important issue in multimedia communications. In this paper, we propose an adaptive synchronization agency for synchronization of streams by using an agent based approach. The synchronization agency triggers one of the three synchronization mechanisms, *point synchronization or real-time continuous or adaptive synchronization* in order to adapt to the run-time and life-time presentation requirements of an application. The scheme or agency employs static and mobile agents for the following purpose: to estimate the network delays in real-time based on sustainable stream loss, to compute the skew, to monitor the loss and estimate the playout times of the presentation units. We have experimentally evaluated the scheme by using IBM Aglets and verified its functioning in terms of synchronization loss and mean buffering delays. The benefits of this agent based scheme are: asynchronous and autonomous delay estimation, flexibility, adaptability, software re-usability and maintainability.

Keywords: Multimedia; Agents; Stream synchronization; Delay estimation; IBM Aglets; Playout

1. Introduction

Multimedia communication deals with transfer of data over the network among multimedia systems. These systems include multiple sources of various media that are either spatially or temporally related to create composite multimedia documents. Spatial composition links various multimedia streams into a single entity. Temporal composition creates multimedia presentations by arranging the multimedia streams according to their temporal relationship, and the relationship is either loosely or tightly coupled (Little and Ghafoor, 1990; Elisa and Elena, 1998).

Continuous media (stream) such as audio and video are characterized by well defined temporal relationship between subsequent presentation units to be played (a presentation unit is a logical data unit that is perceivable by the user). Generally, the process of maintaining the temporal order of one or more media streams is called *multimedia synchronization*. The problem of maintaining continuity within a single stream is referred as *intra-stream or serial* synchronization, whereas the problem of maintaining continuity among the streams is called as *inter-stream or parallel* synchronization. The intra-stream and inter-stream synchronizations are necessary for both live stream(s) as well as for stored media stream(s) presentations (Blakowski and Steinmetz, 1996; Steinmetz and Klara, 1995). Human perceptions about the streams synchronization and the presentation requirements for different types of applications are discussed in Steinmetz (1996).

* Corresponding author.
E-mail addresses: sunil@protocol.ece.iisc.ernet.in (S.S. Manvi), pallapa@ece.iisc.ernet.in (P. Venkataram).
URL: http://pet.ece.iisc.ernet.in/pallapa (P. Venkataram).

0164-1212/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2005.08.010

The maintenance of temporal relationships within a stream or among the multimedia streams usually depends on the following parameters (Cosmos, 1990; David, 1991):

(1) Network delays: The delays experienced by the presentation units (PUs) in the network to reach its receiver, which varies according to network load.
(2) Network jitters: Delay variations of inter-arrival of PUs at the receiver due to varying network load.
(3) End-system jitters: Delay variations in presentation at the receiver due to varying CPU load and protocol processing delays.
(4) Clock skew: The clock time difference between the sender and the receiver.
(5) Clock drift: Rate of change of clock skew because of temperature differences or imperfections in crystal clocks.
(6) Rate drift: Change in generation and presentation rates due to server and receiver load variations.
(7) Network skew: Time difference in arrival of temporally related PUs of streams., i.e., differential delay among the streams.
(8) Presentation skew: Time interval in which the temporally related PUs of the streams are presented.

Synchronization mechanisms are needed to cope-up with these problems to ensure the temporal ordering of streams and to maintain the presentation quality.

1.1. Some of the existing synchronization mechanisms

The synchronization mechanisms for stream playout are broadly classified as point and continuous (real-time or adaptive) synchronization (Dick and Van-Liere, 1992). Harmony (Kazutoshi et al., 1993) uses point as well as real-time and adaptive synchronization playout for hypermedia objects. MHEG and HYTIME represent point and real-time synchronization which are used for synchronizing the hypermedia documents (Newcomb et al., 1991; Markey, 1991). The scheme proposed in (Panagiotis et al., 1996) estimates the reference playout points for the packets based on the observed packet delays. Some of the schemes given in Aidong et al. (2002), Ramanathan and Rangan (1993) and Feng and Krishnaswami (1998) use buffering and feedback control for multimedia transmission and synchronization. Enforcing synchronization in playout scheduling to deal with rate and delay variance of streams is proposed in Thomas and Aidong (1999). The concord algorithm (Shivkumar et al., 1995) defines a framework to deal with synchronization of streams to reduce buffering delays.

Flow synchronization protocol ensures that information in related flows is presented in temporal manner and adapts to changes in flow delays (Julio et al., 1994). Synchronization mechanisms based on presentation deadline approaches are discussed in Shahab et al. (1996). The work given in Kouhei et al. (2001) proposes adaptive playout control algorithms based on statistical analysis of packet delays. An adaptive synchronization protocol based on buffer level control mechanism is proposed in Kurt and Tobais (1997). The work given in Ramachandran et al. (1994) discusses about adaptive playout mechanisms for packetized audio. Multimedia synchronization for live video and audio streams is achieved by bounded buffer allocation scheme and applying forward re-synchronization policies such as restricted blocking and blocking to overcome synchrony anomalies (Chung-Ming and Ruey-Yang, 1996).

A stream synchronization protocol is proposed for news-on-demand application that allows synchronization recovery ensuring high quality presentation at receiver (Louise et al., 1996). The works given in Miguel and Paulo (1995) and Sangshin (1998) performs adaptive synchronization that adapts to acceptable QoS (Quality of Service) depending on CPU and network load variations. The research work in Ernst and Werner (1999) proposes adaptive synchronization playback mechanism that computes the buffer required to achieve both continuity within a single sub-stream and multiple sub-streams. RTP (real transport protocol) uses real-time control protocol (RTCP) to synchronize media streams prior to decoding operations by computing jitters (Schulzrinne et al., 1996). The work given in Zhuge (2002) presents about realization of semantic interconnectivity for multimedia data and their synchronization for soft devices by forming clusters of semantic grid. Synchronization schemes for VCR like operations is given in Chian and Chung-Ming (2004). Imperceptible and allowable ranges for inter-stream synchronization are employed for synchronization between haptic media and voice in collaborative virtual environments (Yutuka et al., 2004). The work given in Wang and Lin (2000) gives the use of agents in synchronization based on bandwidth allocation.

All the works discussed so far deal with life-time synchronization (i.e., same synchronization mechanism is used from beginning to end of an application) requirements for an application. None of the above schemes provide a flexible and adaptable mechanism of arbitrating among different types of synchronizations based on the run-time synchronization (synchronization mechanism changes depending on the semantic requirements of data from beginning to end of an application) requirements of an application. And also, existing schemes lacks extensibility and flexibility that is needed in current web-based systems to accelerate the communication software development.

1.2. Proposed work

In the proposed work, we have designed synchronization techniques based on three types of synchronizations: point, real-time continuous and adaptive synchronization. These techniques are developed by using the agents to carry out continuous and smooth playout of multimedia streams. Point synchronization realizes that the start/completion time of PUs of the streams is synchronized with a certain specified synchronization point between the streams. In real-time continuous synchronization, PUs of streams are synchronized with real-time axis. In adaptive synchronization, presentation time of units of streams are readjusted at regular intervals with change in network delays to reduce the losses.

We propose a synchronization agency framework comprising of static and mobile agents that either arbitrates among the specified synchronization mechanisms depending on the run-time synchronization requirements of an application or maintains a specified synchronization mechanism throughout the life-time of an application. The proposed synchronization method runs at the receiver.

An example of run-time synchronization usage can be observed in on-line Internet based education systems where the lecture streams are distributed in different servers (Manvi and Venkataram, 2003). In this education systems, adaptive synchronization is used for regular viewing of lecture, real-time synchronization for random viewing of important concepts of missed lecture in the form of clips (whenever a user who has taken a break from the real-time lecture in between due to some reasons and rejoins the lecture), and point synchronization for viewing a particular missed concept in detail.

1.3. Organization of the paper

The following section gives a brief background about the agent technology. The proposed agent based synchronization mechanisms are described in Section 3. Experimental evaluation of the scheme is discussed in Section 4. Some of the benefits of using agents for multimedia synchronization are described in Section 5, finally we conclude in Section 6.

2. Agents

Agents are the autonomous programs situated within an environment, which sense the environment and acts upon the environment by using its knowledge base to achieve their goals. They have certain special properties which make them different from the standard programs such as *mandatory* and *orthogonal properties*. Mandatory properties of the agents are: *autonomy*, *reactive*, *proactive and temporally continuous*. The orthogonal properties are: *communicative*, *mobile*, *learning and believable* (Manvi and Venkataram, 2004). Agents can be classified based on properties they posses: *local or user interface agents*, *network agents*, *distributed AI* (*Artificial Intelligence*) *agents and mobile agents*.

Mobile agent is an itinerant agent dispatched from the source computer which contains program, data and execution state information, and migrates from one host to another host in the heterogeneous network, and executes at a remote host until it achieves its goals (Wong et al., 1999; Chess et al., 1995). Mobile code should be platform independent, so that, it can execute at any remote host in the heterogeneous network environment. Agent can update its information base while interacting with other agents during its travel. Inter-agent communication can be achieved by message passing or common knowledge base (black board architecture principle).

Some of the Java based agent platforms are (Menelaos et al., 1999): Aglets, Grasshopper, Concordia, Voyager and Odyssey. We have used IBM Aglets Work Bench (AWB) for experimental evaluation of proposed synchronization agency. The term aglet stands for "agent + applet". An aglet is a roaming applet that can move from one computer to another to achieve its goals (Danny and Mitsuru, 1998). Aglets class library provides a rich set of application program interfaces that facilitates the encoding of complex agent behavior. Agent services available in AWB for the developer are: persistence, security, communication messaging, collaboration and web enabled agents.

Agent based schemes comprising of static or mobile agents offer several advantages as compared to traditional approaches: reduces latency, reduces network traffic, encapsulates protocols, flexibility, adaptability, software re-usability and maintainability (Danny and Mitsuru, 1999; Nicholas, 2001). However, some problems have to be resolved in agent systems implementation: creation of a standardized agent platform for Internet, security to agents from hosts and vice versa.

3. Proposed synchronization technique

The proposed synchronization technique uses three synchronization mechanisms point, real-time and adaptive synchronizations based on the run-time and lifetime application requirements, and uses sequence numbers to identify the PUs to be synchronized.

Point synchronization realizes that the start/completion time of PUs is synchronized with a certain specified point on the real-time axis: using this mechanism, presentation of multimedia streams is started simultaneously (start point synchronization) and also the presentation of subsequent PU is made in synchronized manner whenever they are available. This kind of mechanism is suitable for applications like slide shows and cricket highlights display.

Real-time continuous synchronization is realized by presenting the PU of a stream synchronized with the real-time axis or PU of another stream. For example, the motion video with contents of 400 s should be presented for exactly 400 s or a motion video should be presented synchronized with its audio. If presentation of certain information of the stream lags behind normal presentation rate due to an unexpected heavy load on network/workstation, the mechanism skips over certain portions of presentation such as frames of video and catches up to correct position of the multimedia stream. This mechanism is suitable for entertainment applications if the estimated network delays does not change for the life-time of an application. It can also be used for applications which show random portions of detailed multimedia documents: watching several sets of movie trailers before making a decision to purchase a movie CD (Compact Disc) or to view a movie on Internet.

Real-time continuous synchronization has some deficiencies, because loss of certain presentation information is semantically very serious for application fields such as education and also does not offer good QoS for entertainment applications. To overcome this problem, adaptive synchronization mechanism is proposed to maintain the semantics of multimedia information and also ensure better QoS by reducing media losses. In this case, presentation time of units of media streams are readjusted at regular intervals with change in network delays and become synchronized with one another.

In this section, we discuss the synchronization mechanisms by considering a playout system model.

3.1. Playout system model

In this section, we describe a client server based playout system in which servers and clients are distributed throughout the Internet (see Fig. 1). Generally, playout system handles multimedia streams S_1 to S_i which needs to be synchronized for playout at client c. The characteristics of the playout system model are as follows:

- The PU of the streams are chosen as a unit perceivable by the human beings, for example, a frame is a perceivable PU of a video stream.
- The PUs of each stream are labeled with sequence numbers, and they are presented in same sequence.
- The network delay of a PU of each stream varies stochastically.
- PUs of each stream will not arrive in order at client c (since UDP transport protocol is used).

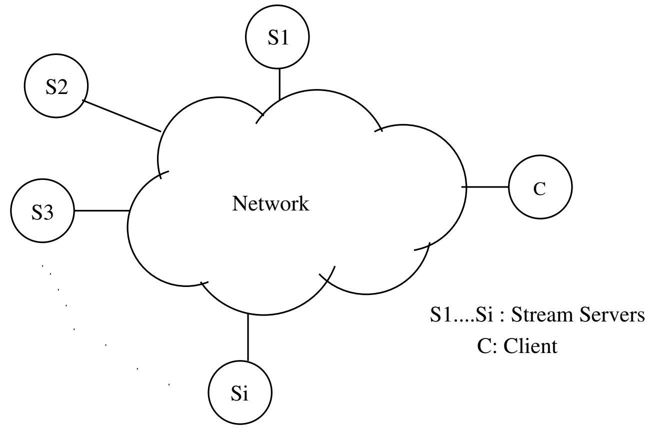

Fig. 1. Playout system model.

- Clock differences exist among the servers and client distributed in the network.
- Rate of clock drifts is zero since all the crystal oscillators are assumed to be stable.

The synchronization agency is run at the client which is described in the following subsection.

3.2. Synchronization agency framework

The proposed synchronization agency framework at the client comprises of set of static and mobile agents to perform multimedia streams synchronization. All the hosts connected to the network consists of an agent platform (IBM AWB in our experiment) to support the static and mobile agents.

The proposed synchronization agency framework at the client comprises of the following components (see Fig. 2): user interface agent, synchronization agent, mobile agents, and synchronization database. The functions of each of these components are given below.

- *User interface agent*: It is a static agent (aglet) which collects the application information such as application identification number (*appid*), stream servers address, required synchronization type (*syntype*) and the synchronization parameters (*synpar*). The synchronization parameters include sustainable and desired presentation rates, maximum allowed skew, and acceptable loss. It sets *syntype* value based on the life/run-time presentation requirements of an application. The value of *syntype* will be dynamically set for run-time presentation requirements whereas it will be set only once for life-time presentation requirements. The *syntype* values 0, 1 and 2 indicate point, real-time and adaptive synchronization, respectively. It creates a synchronization agent and a synchronization profile.
- *Synchronization agent*: This static agent (aglet) performs three types of synchronizations based on the

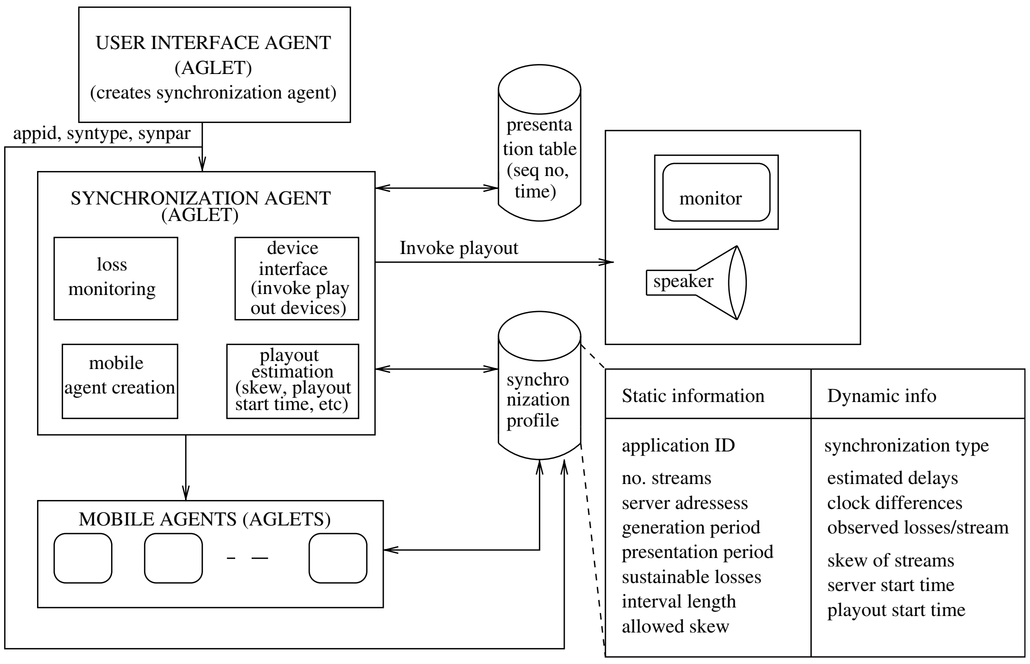

Fig. 2. Synchronization agency framework.

application presentation requirements. To identify the presentation requirements, it uses a variable *syntype* (0 = point, 1 = real-time and 2 = adaptive). It creates a mobile agent for each stream to estimate the clock difference and the network delays on the path which runs from the server to client. It computes skew, stream data transfer start time and the playout start time. Reference playout times of all the PUs of each stream is computed in case of real-time continuous synchronization. In case of adaptive synchronization, it periodically recomputes the playout times of all the PUs of each stream based on the recently estimated network delays and the monitored losses of each stream. Synchronization agent updates the synchronization profile and the presentation table with newly estimated parameters and the playout times respectively. Skew compensation mechanisms are used by the agent to provide continuous playout of the streams.

- *Mobile agents*: These agents (aglets) estimate the clock differences and the network delays among the servers and the client and updates the synchronization profile. They are also used to inform the servers about the start time of data transfer. They can be programmed to monitor the bandwidth, loss and delay parameters at the intermediate nodes and perform parameter negotiation at the servers, intermediate nodes and the client to facilitate better quality presentation.
- *Presentation table*: This table maintains the reference playout times of PUs of the streams (sequence number and playout time) as computed by the synchronization agent.
- *Synchronization profile*: This profile is a knowledge base of the synchronization agency. It stores the session information (static and dynamic) and facilitates sharing of information among agents. The static information is set by the user agent whereas dynamic information is updated by either mobile agents or synchronization agent. The static information stored are: application identification number, stream servers address, generation period, presentation period, sustainable losses, maximum allowed presentation skew and the length of interval (in terms of PUs) to reestimate the playout times. The dynamic information stored are: clock differences between the client and servers, estimated delays to servers from the client, skew between the streams, data transfer start time of the streams, synchronization type and the monitored losses of each stream.

The scheme does not require time-stamping of PUs since the presentation timing of a PU is computed by the agency itself. Network delays are estimated in real-time independent of PU arrivals enhancing the adaptive capabilities of the synchronization. Mobile agents encapsulate protocol for network delay estimation, thus estimation policy can be changed by just changing the code within it, facilitating flexible delay estimation. A delay estimator mobile agent can be extended to perform aggregate tasks such as bandwidth monitoring and allocation and QoS negotiation and renegotiation, etc. However, some overheads are associated with this scheme, such as, maintenance of synchronization profile and an agent platform.

3.3. Streams synchronization

Now we discuss how the synchronization agent computes the clock difference, network delay, skew, server data transfer start time, and the playout start time at the client side by considering a session with i number of streams. A mobile agent will be created for each stream by the client. Mobile agent of the respective stream makes specified round trips to stream server to estimate the network delays.

3.3.1. Clock difference computation

The clock difference between two hosts is computed as given in Wenyu and Schulzrinne (1999) Eq. (1) (assuming symmetric path delay assumption)

$$\Delta^{i,c} = ((T3^{i,z} - T1^{i,z})/2) - (T2^{i,z} - T1^{i,z}) + \delta \tag{1}$$

where $\Delta^{i,c}$ = clock difference between ith stream server and the client c, $T1^{i,z}$ = time at which mobile agent begins onward journey to stream server i during zth trip, $T2^{i,z}$ = time at which mobile agent reaches ith stream server during zth trip, $T3^{i,z}$ = time at which mobile agent returns to client from ith stream server after completing the zth trip. If $\Delta^{i,c}$ is negative, then client c is lagging, otherwise, client c is leading with respect to server i. Here we consider that a mobile agent of the stream in every trip does note the clock difference as well, and $z \in \{1, 2, \ldots, Z\}$ where Z = maximum number of trips. The estimated clock difference may not be accurate because of symmetric path delays assumption, hence we include an estimation error δ in Eq. (1) ($\delta \approx 150$ ms as observed in our experiment).

3.3.2. Delay estimation

In each trip, a mobile agent of stream i records the one way delay (assuming the symmetric paths) as given in Eq. (2)

$$d^{i,z} = T3^{i,z} - T1^{i,z}/2 \tag{2}$$

where, $d^{i,z}$ = one way delay incurred by mobile agent of ith stream during zth trip. The delays recorded in the mobile agent memory are used to construct Cumulative Distribution Function (CDF) of delay, $F(\cdot)$. Mobile agent of ith stream computes the delay $D^{i,t}$ in time interval t ($t = 0$ before stream transfer starts) after completion of Z trips, such that the losses are within the acceptable loss of each stream as given in Eq. (3).

$$F(d^{i,z} < D^{i,t}) \geqslant 100 - \mathrm{AL}^i \tag{3}$$

where, AL^i is the percentage of acceptable losses for stream i.

3.3.3. Skew computation

Skew ($\mathrm{sk}^{i,t}$) of ith stream with reference to other streams is defined as inter-arrival time difference among the PUs of different streams in time interval t ($t = 0$ before stream transfer starts). It is computed as given in Eq. (4)

$$\mathrm{sk}^{i,t} = \max(D^{i,t}, \forall i) - D^{i,t} \tag{4}$$

3.3.4. Stream start time computation

Stream start time, sstart^i, is defined as the time at which streams start flowing into the network. It is computed as given in Eq. (5).

$$\mathrm{sstart}^i = \begin{cases} \mathrm{cstart} + \Delta^{i,c} + \theta & \text{if } \Delta^{i,c} \leqslant 0 \\ \mathrm{cstart} - \Delta^{i,c} + \theta & \text{otherwise} \end{cases} \tag{5}$$

where $\mathrm{cstart} = \mathrm{tcur} + \epsilon + \max(D^{it}, \forall i)$ for $t = 0$, and, cstart = time at which the client plans to ask the servers to start the stream transfer, tcur = current clock time at client once all the mobile agents return after Z trips, θ = PU generation period at the source, ϵ = time to compute the playout time at the receiver (set $\epsilon = \theta$).

3.3.5. Playout start time computation

The playout start times, spt^i, is defined as the time at which the playout of the ith stream commences. It is computed as given in Eq. (6) for the time interval $t = 0$

$$\mathrm{spt}^i = \begin{cases} \mathrm{sstart}^i + \Delta^{i,c} + D^{i,t} + \theta + \mathrm{sk}^{i,t} & \text{if } D^{i,t} < \max\limits_{\forall i} D^{i,t} \\ \mathrm{sstart}^i + \Delta^{i,c} + D^{i,t} + \theta & \text{otherwise} \end{cases} \tag{6}$$

3.4. Point Synchronization

Algorithm 1 describes the point synchronization mechanism used by the synchronization agent.

Algorithm 1 (*Point synchronization*).
{*To describe the mechanism consider a session with sm streams*}

```
Begin
  (1) User interface agent creates the synchronization
      agent and updates the synchronization profile with
      session information.
  (2) For $s = 1$ to sm do
      Begin
        • Synchronization agent creates a mobile agent
          for stream $s$ for estimation of clock difference
          and network delays.
        • Mobile agent of stream $s$ makes $Z$ round trips to
          stream server $s$ and records the clock difference
          and the one way delay between the server $s$ and
          the client (Eqs. (1), (2)), computes the delay (Eq.
          (3)), and updates the synchronization profile.
      Endfor $s$;
```

```
(3) For $k = 1$ to sm do
    Begin
    • Synchronization agent computes the skew for the stream $k$ and updates the synchronization profile (Eq. (4)).
    • Synchronization agent computes the stream $k$ start time and sends a mobile agent to server $k$ to convey start time information, and updates the synchronization profile (Eq. (5)).
    • Synchronization agent computes playout start time of the stream $k$ and updates the presentation table (Eq. (6)).
    Endfor $k$;
(4) For $s = 1$ to sm do {Synchronization agent begins playout of the stream $s$};
(5) Stop;
End.
```

In point synchronization, if the PUs to be presented do not arrive within the duration of skew tolerance (allowed skew values may be approximately 100–200 ms), synchronization agent uses skew compensation mechanisms. These mechanisms are: restricted blocking (display the last frame to deal with the losses and delayed frames) and blocking (do not play anything) for video and audio streams respectively. The late arrived PUs of each stream will be skipped.

3.5. Real-time synchronization

This mechanism uses the similar steps as discussed in point synchronization mechanism to compute the starting playout time of the streams. In addition to this, synchronization agent computes the reference playout times for all the contents of the streams as given in Eq. (7) (at $t = 0$).

$$\mathrm{refpt}^{i,j} = \begin{cases} \mathrm{sstart}^i + \Delta^{i,c} + D^{i,t} + \theta + \mathrm{sk}^{i,t} + \xi * (j-1) \\ \quad \text{if } D^{i,t} < \max\limits_{\forall i} D^{i,t} \\ \mathrm{sstart}^i + \Delta^{i,c} + D^{i,t} + \theta + \xi * (j-1) \\ \quad \text{otherwise} \end{cases} \tag{7}$$

where $\mathrm{refpt}^{i,j}$ = reference playout point of jth PU of ith stream, $\xi = \theta$ is the presentation period of a PU. Algorithm 2 describes the real-time synchronization mechanism.

Algorithm 2 (*Real-time synchronization*).

{*To describe the mechanism consider a session with sm streams*}

```
Begin
(1) User interface agent creates the synchronization agent and updates the synchronization profile with session information.
(2) Perform operations as given in steps 2 and 3 of Algorithm 1.
(3) For $s = 1$ to sm do {Synchronization agent computes the reference playout time of all the PUs of the stream s (Eq. (7))}.
(4) For $s = 1$ to sm do {Synchronization agent begins playout of the stream $s$}.
(5) Stop.
End.
```

Synchronization agent displays the media units that arrive within the estimated playout time. Skew compensation mechanisms are used by the synchronization agent to overcome the playout gap problems caused due to losses or late arrivals.

3.6. Adaptive synchronization

This mechanism uses similar steps of point synchronization to compute the stream playout start times. Synchronization agent sends a mobile agent of each stream periodically (at regular time intervals) for $Z/2$ trips to estimate the network delays over the path from client to server. These estimated delays and the observed PU losses are used to readjust the reference playout times of the PUs. The gaps created due to readjustment of reference playout times are compensated by using skew compensation mechanisms. The reference playout times are computed for every k presentation units (k is the length of time interval for delay adaptation).

For the first k presentation units, i.e., for the first time interval $t = 1$, Eq. (8) is used in computing the playout time, $\mathrm{puplay}^{i,j,t}$, for the jth PU of a stream i, where $j \in \{1, \ldots, k\}$.

$$\mathrm{puplay}^{i,j,t} = \begin{cases} \mathrm{sstart}^i + \Delta^{i,c} + D^{i,t-1} + \theta + \mathrm{sk}^{i,t-1} + \xi * (j-1) \\ \quad \text{if } D^{i,t-1} < \max\limits_{\forall i} D^{i,t-1} \\ \mathrm{sstart}^i + \Delta^{i,c} + D^{i,t-1} + \theta + \xi * (j-1) \\ \quad \text{otherwise} \end{cases} \tag{8}$$

For the successive time intervals, $t \in \{2, \ldots, T\}$, where $T = \lceil N/k \rceil$ = number of intervals and N = number of PUs in each stream, the PU sequence number ranges are $k+1$ to $2k$ for $t = 2$, $2k+1$ to $3k$ for $t = 3$, and so on. The playout start time of a PU in the current interval (t) depends on the time of the last displayed PU and the observed losses in the previous interval ($t - 1$). The observed losses of a stream i in the previous interval as computed by the synchronization agent is given as $\mathrm{obl}^{i,t-1} = (\mathrm{npul}^{i,t-1}/k) * 100.0$, where, $\mathrm{obl}^{i,t-1}$ = percentage of observed losses of a stream i in the $t-1$ interval (previous interval), $\mathrm{npul}^{i,t-1}$ = number of PUs lost in a stream i in the previous interval.

PU sequence numbers in interval t ranges from $(t-1)*k+1$ to $t*k$. If $\text{obl}^{i,t-1} > \text{AL}^i$ (observed losses are greater than the acceptable loss), the start time of the stream i for the interval t is computed by using, $\text{puplay}^{i,(t-1)*k,(t-1)}$ (the time at which the last PU $(((t-1)*k)$th) of a stream i of the interval $t-1$ is displayed). The playout times for all the PUs in interval t is computed as given in Eq. (9)

$$\text{puplay}^{i,j,t} = \text{puplay}^{i,(t-1)*k,t-1} + (D^{i,t} - D^{i,t-1}) + (\text{sk}^{i,t} - \text{sk}^{i,t-1}) + \xi * (j \bmod k - 1) \tag{9}$$

where $j \in \{((t-1)*k+1), \ldots, (t*k)\}$. If observed losses of a stream i are within the AL^i in the previous interval, the playout times of PUs of successive interval are computed by using the previous interval playout ending time as given in Eq. (10)

$$\text{puplay}^{i,j,t} = \text{puplay}^{i,(t-1)*k,t-1} + \xi * (j \bmod k - 1) \tag{10}$$

The detailed description of the adaptive synchronization mechanism is given in Algorithm 3.

Algorithm 3 (*Adaptive synchronization*).
{*To describe the mechanism consider a session with sm streams*, *T time intervals*, *k PUs in each time interval*}

```
Begin
  (1) User interface agent creates a synchronization agent and updates the synchronization profile with session information.
  (2) Perform the operations as given in steps 2 and 3 of Algorithm 1.
  (3) For $t = 1$ to $T$ do
       Begin
      • If $t = 1$, synchronization agent computes the playout times of $k$ PUs using Eq. (8).
      • If losses in the interval $t - 1$ are greater than the acceptable loss and $t > 1$ {synchronization agent recomputes the playout times of $k$ PUs of all the streams in the interval $t$ (Eq. (9))}.Else {compute the playout times of $k$ PUs in the interval $t$ (for $t > 1$) using Eq. (10)};
      • For $s = 1$ to $sm$ do
      •    Begin
         – Synchronization agent starts playout of the PUs of stream $s$ in the time interval $t$.
         – Synchronization agent monitors the losses of stream $s$ in time interval $t$ and updates synchronization profile.
         – Synchronization agent sends a mobile agent for $Z/2$ trips to estimate the network delays between the stream server $s$ and the client and updates synchronization profile.
           Endfor $s$;
     Endfor $t$;
  (4) Stop;
End.
```

4. Experimental evaluation

We experimentally evaluated the scheme by implementing the synchronization agency by using IBM aglets work bench (Version 2.0) on the hosts distributed across PET-UNIT network which is an in-house network (see Fig. 3). We considered applications with different presentation requirements, and

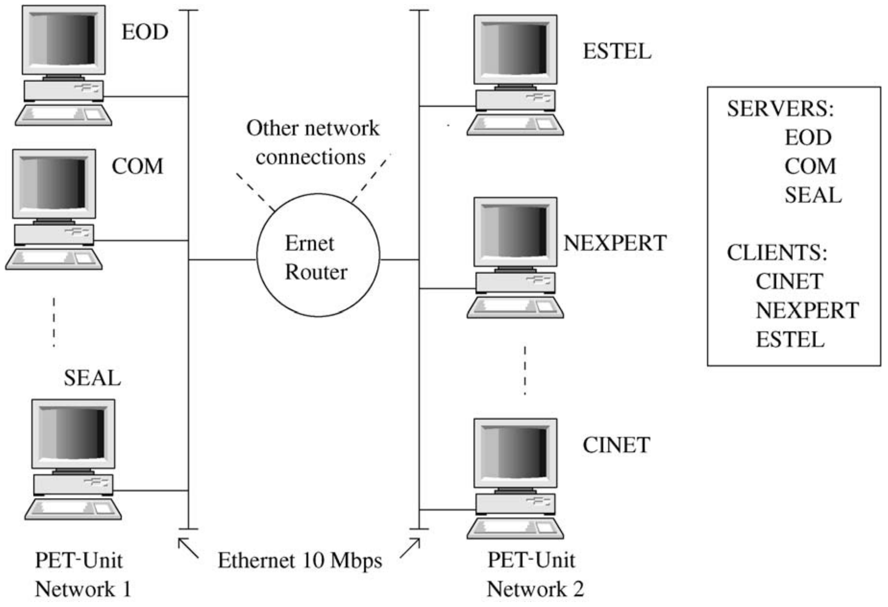

Fig. 3. Experimental network topology.

percentage of network delay variations for experimental evaluation.

The multimedia data is distributed stream-wise among the servers. We tested the designed scheme with two servers (EOD and SEAL among the shown three servers) and the multiple clients. Usefulness of the scheme is tested for applications like slide show, movie trailers and the Internet education. For Internet education we used run-time synchronization: adaptive synchronization for regular viewing of study material; real-time synchronization for random viewing of study material; point synchronization for viewing a particular concept in detail. Movie trailers used real-time synchronization as life-time synchronization requirement. Slide show used point synchronization as life-time synchronization requirement.

4.1. Experimental procedure

We artificially injected delays for each arriving PU of a stream to create a multi-hop environment and disorder the PUs, since hosts in the experimental setup uses only one hop. Several parameters used in the experimental setup are as follows.

- Network delays injected for arriving PUs at the client and the stream mobile agents are normally distributed with mean μ and variance σ^2.
- The delays rise are injected at every regular intervals (interval in terms of number of PUs) to create sharp fluctuations in delays with probability P.
- Increase in mean of the PU delays due to P is $x * \mu$, where x is the percentage rising factor.
- Increase in delay variance due to P is $V * (\mu + x * \mu)$, where V is the percentage rising factor.
- Applications specify: the number of PUs (N), sustainable losses, PU generation and presentation period, and the skew tolerance in terms of PUs.

The performance parameters evaluated in the experiment are as follows:

- *Stream PU loss*: It is defined as the percentage of PU losses in a stream.
- *Synchronization loss*: It is defined as the percentage of PU loss of either of the streams in a presentation period.
- *Mean buffering delay*: It is defined as the mean waiting time of presented PUs of a stream in the receiver.

The experimental procedure adopted is as follows:

(1) Evaluated the scheme with $P = 0$, to find out the number of optimal trips Z by observing the PU losses and mean buffering delays by applying point synchronization.
(2) Real-time synchronization mechanism is evaluated by injecting PU delay variations periodically.
(3) Adaptive synchronization mechanism is evaluated in cases of network delay fluctuations.
(4) Run-time synchronization mechanism is evaluated by creating network delay variations.

The inputs considered to discuss the experimental results are: $N = 20{,}000$; injected network delays (in milliseconds) of stream 1 and stream 2 are normally distributed with $\mu_1 = 450$, $\mu_2 = 350$, $\sigma_1^2 = 100$ and $\sigma_2^2 = 100$; P is varied from 0 to 1.0; size of interval for considering delay rise $(k) = 200$ PUs, and the burst length of PUs for which delay rise is affected is randomly distributed between 1 and 200; x is randomly distributed within range 10–100%; V is randomly distributed within range 5–50%; $\theta = \xi = 80$ ms; and sustainable loss of stream 1 and stream 2 are 10%.

4.2. Results

Optimal number of trips required by an mobile agent for delay estimation are determined without injecting the artificial delay rise to arriving PUs (see Figs. 4 and 5). We

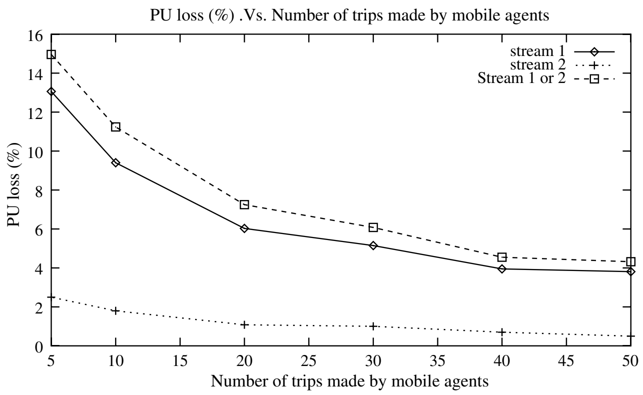

Fig. 4. PUs loss (%) vs Number of trips made by mobile agents with $P = 0$ (without injecting delay rise).

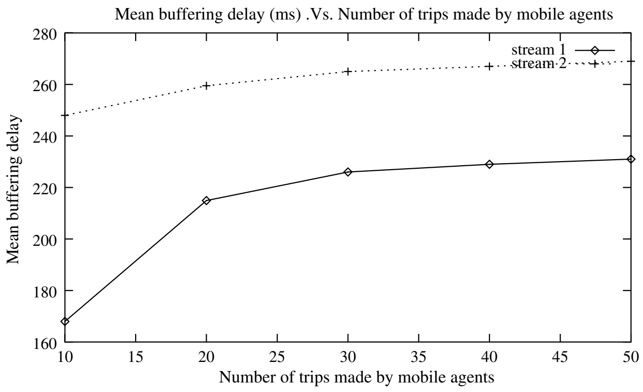

Fig. 5. Mean buffering delay for a PU vs Number of trips made by mobile agents with $P = 0$ (without injecting delay rise).

observe that the synchronization losses and the mean buffering delays almost remain stable for trips = 30 (the PU loss and buffering delay variations are not much significant with trips ranging from 35 to 50 as compared to when trips = 30). Hence, we initially fixed the mobile agent trips to be 30 for real-time and adaptive synchronization.

We observe (see Figs. 6 and 7) that the PU losses increase with delay fluctuations and the mean buffering delays of PU of the streams reduce with increase in losses and the delay fluctuations. Huge losses occur when $P = 1.0$, thus real-time synchronization mechanism is not suitable when the loads are heavily fluctuating in the network.

In adaptive synchronization, we notice (see Figs. 8 and 9) that the synchronization losses are maintained well within the desired limit (10%), and the mean buffering delays are slightly higher for both the streams as compared to other plots (see Figs. 5 and 7). Due to continuous adaption to changes in delay rises, PUs will be buffered for more time to avoid the PU loss. The mean buffering delays reduce with increase in losses and P. Hence adaptive synchronization mechanism is suitable for applications that require high quality of service.

The PU loss and the mean buffering delays for the run-time synchronization for an application is shown in Figs. 10 and 11 which is run for 30 min, where point synchronization is employed for 5 min in the session,

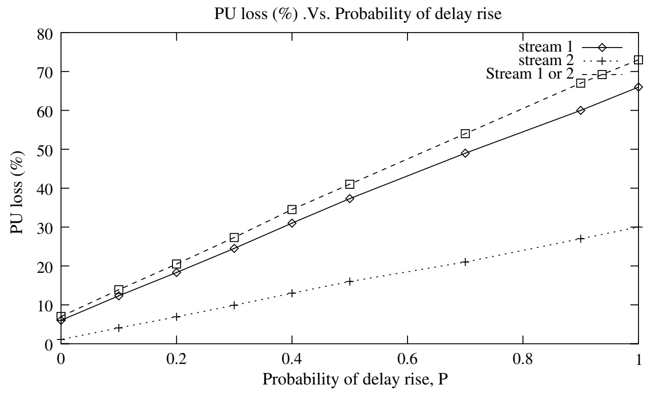

Fig. 6. Presentation units loss (%) vs Probability of delay rise (P) for real-time synchronization.

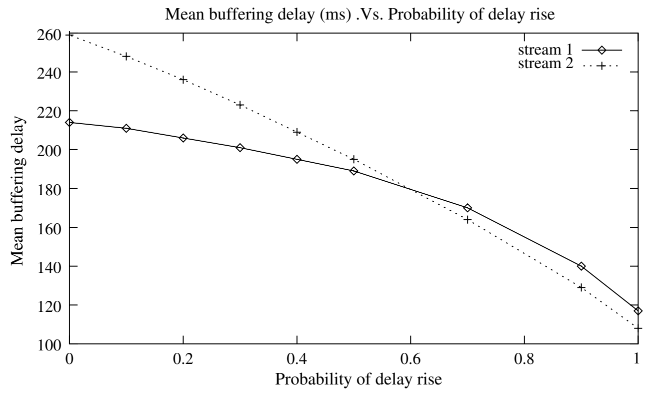

Fig. 7. Mean buffering delay for a PU vs Probability of delay rise (P) for real-time synchronization.

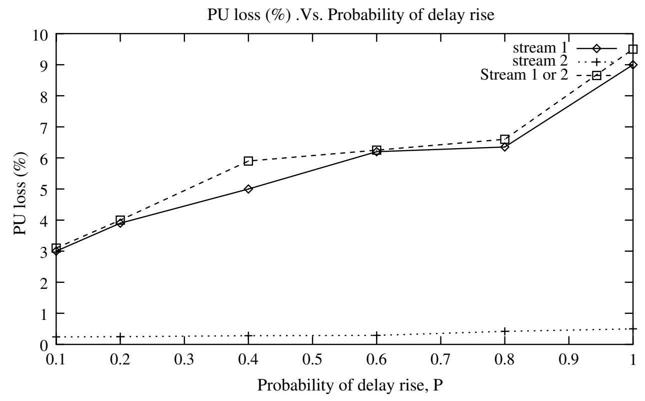

Fig. 8. Presentation units loss (%) vs Probability of delay rise (P) for adaptive synchronization.

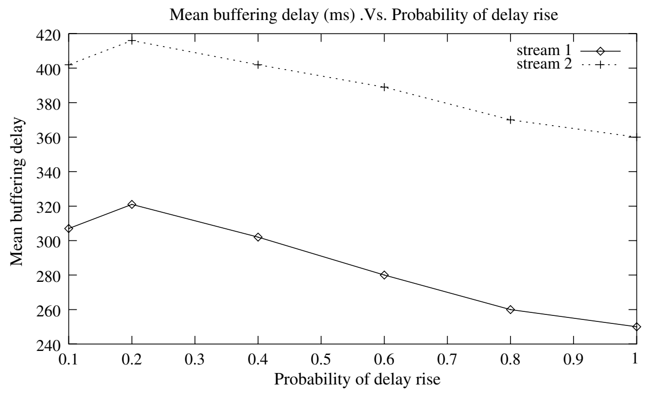

Fig. 9. Mean buffering delay for a PU vs Probability of delay rise (P) for adaptive synchronization.

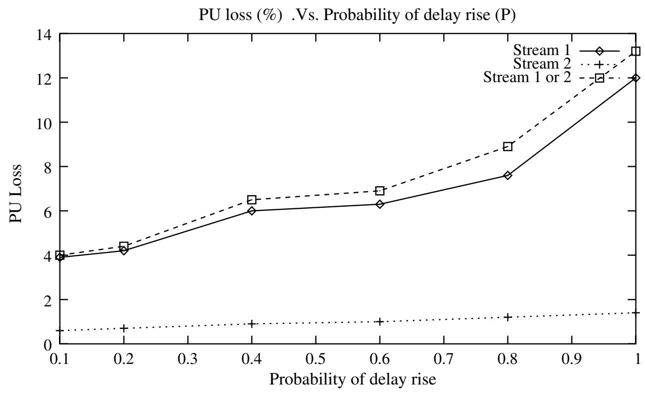

Fig. 10. Presentation units loss (%) vs Probability of delay rise (P) for run-time synchronization.

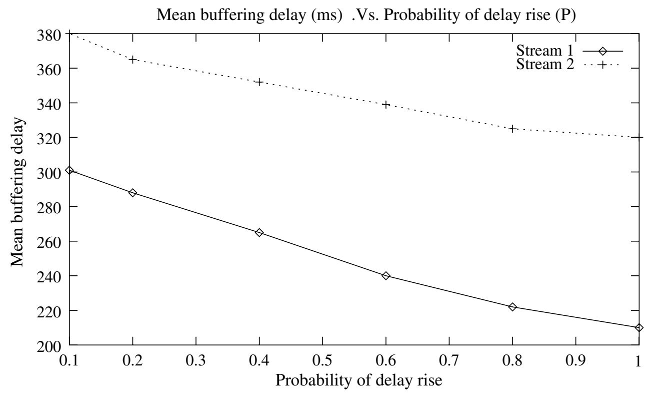

Fig. 11. Mean buffering delay for a PU vs Probability of delay rise (P) for run-time synchronization.

real-time synchronization is used for 5 min in the session, and the rest of the session is adaptively synchronized.

5. Benefits of using agents

In this section we would like to highlight some of the benefits we get by using agents for multimedia synchronization purpose as compared to traditional methods. Agent oriented programming facilitates CBSE which is needed in today's software development of web-based systems (Griss and Pour, 2001). As observed from the literature (Manvi and Venkataram, 2000; Ajay et al., 2003; Manvi and Venkataram, 2005), mobile agents are used in various ways to support multimedia communications: they go through the intermediate nodes to gather the bandwidth and delay information; they can identify the congested nodes in the network and suggest different routes; they can intelligently vary the bandwidth at the source depending on the network environment. We can encode all these functionality along with the delay estimation within a single mobile agent and improve the performance of multimedia communication systems which justifies our use of mobile agents in delay estimation.

We observe in our experiments that agent based synchronization scheme offers flexibility, adaptability, reusability and maintainability. Even though it is difficult to quantify these features, we explain below how they are achieved by the synchronization agency:

Flexibility: Synchronization agent demonstrates its flexibility in changing the synchronization scheme based on the information received from application/user interface agent and presents the PUs using appropriate skew compensation techniques. It can also be programmed to dynamically arbitrate among synchronization mechanisms based on the network conditions learned by the mobile agent. Mobile agents show flexibility in delay estimation by applying different policies for estimation (for example, delay estimation based on actual round trip or artificial round trip times, agent can intelligently over estimate when it encounters consecutive delay rises).

Adaptability: Synchronization agent adapts the presentation schedule as per the network dynamics (delay variations captured by mobile agents) in case of adaptive synchronization and uses appropriate skew compensation techniques to maintain continuous playout.

Re-usability: To demonstrate the software reuse, the mobile agent components was reused with some slight modifications to incorporate bandwidth estimation along with delay estimation and used this bandwidth estimation to dynamically allocate bandwidth using spatial resolution (changing resolution of presentation unit) techniques.

Maintainability: We can easily debug the agent components and also replace the old agent components with a new one without affecting the other components of the agency. For example, if we develop a new scheme of delay estimation in future with distributed time and delay server agents, we can replace the mobile agent component with this new component without affecting the other components of the agency.

6. Concluding remarks

In this paper, we analyzed the synchronization problems in multimedia communications and proposed an agent based synchronization frame work to handle three synchronization mechanisms (point, real-time and adaptive) at application service level depending on the life/run-time presentation requirements of the multimedia applications. Adaptive synchronization mechanism adjusts playout times in accordance with changes in network conditions and offers better quality presentation by maintaining the sustainable losses. The scheme was experimentally evaluated using IBM aglets work bench in a wired network. The results were demonstrated in terms of percentage PU loss and the mean buffering delays. Agent based architectures provide flexible, adaptable and asynchronous mechanisms for multimedia communications.

The proposed technique can also be used for synchronization of multimedia streams in mobile ad-hoc networks that are used in university campuses, disaster recovery operations with slight extensions to the agents in the agency that covers mobility features and resource scarcity of the nodes. In case of heterogeneous environment, a proxy connected to wired network can employ an agency on behalf of mobile users that provides continuous and smooth playout at the mobile devices.

References

Aidong, Z., Yuqing, S., Markus, M., 2002. Netmedia: streaming multimedia presentations in distributed environments. IEEE Multimedia, 56–73.

Ajay, S., Venkataram, P., Manvi, S.S., 2003. QoS routing scheme by using mobile agents. In: Proceedings of the Indian International Conference on Artificial Intelligence (IICAI), Hyderabad, India.

Blakowski, G., Steinmetz, R., 1996. Media synchronization survey: reference model, specification, and case studies. IEEE JSAC (Journal Selected Areas in Communications) 14 (1), 5–35.

Chess, D., Harrison, C., Kershenbaum, A., 1995. Mobile agents: are they a good idea? IBM Research Division, T.J. Watson Research Center, Yorktown Heights, New York, Technical report.

Chian, W., Chung-Ming, H., 2004. Synchronization schemes for controlling VCR like Interactions in interactive multimedia on demand. Computer Journal 47 (2), 140–152.

Chung-Ming, H., Ruey-Yang, L., 1996. Achieving multimedia synchronization between live video and live audio streams using QoS controls. Computer Communications Journal 19, 456–467.

Cosmos, N., 1990. An architecture for real time multimedia communication systems. IEEE JSAC 8, 391–400.

Danny, B.L., Mitsuru, O., 1998. Programming and Deploying Java Mobile Agents with Aglets, Pearson Technology Group Publishers.

Danny, B.L., Mitsuru, O., 1999. Seven good reasons for mobile agents. Communications of ACM 42, 88–89.

David, L.M., 1991. Internet time synchronization: network time protocol. IEEE Transactions on Communications 39, 1482–1493.

Dick, C.A.B., Van-Liere, R., 1992. In: Herrtwich, R. (Ed.), Multimedia Synchronization and Unix, LNCS 614. Springer-Verlag.

Elisa, B., Elena, F., 1998. Temporal synchronization models for multimedia data. IEEE Transactions Knowledge and Data Engineering 10, 612–631.

Ernst, B., Werner, G., 1999. Synchronized delivery and playout of distributed stored multimedia streams. Multimedia Systems Journal 7, 70–90.

Feng, W., Krishnaswami, B., 1998. Proactive buffer management for streamed delivery of stored video. In: Proceedings of the ACM Multimedia Conference, New York, pp. 285–290.

Griss, M.L., Pour, G., 2001. Accelerating development with agent components. IEEE Computer 34 (5), 37–43.

Julio, E., Craig, P., Debra, D., 1994. Flow synchronization protocol. IEEE/ACM Transactions Networking 2, 111–121.

Kazutoshi, F., Shinji, S., Toshio, M., Shojiro, N., Hideo, M., 1993. The synchronization mechanisms of multimedia information in the distributed hypermedia system harmony. In: Proceedings of the International Conference on Multimedia Modeling, Singapore, pp. 275–289.

Kouhei, F., Shingo, A., Masayuki, M., 2001. Statistical analysis of packet delays in the Internet and its application to playout control for streaming applications. IEICE Transactions Communications E84-B, 1504–1512.

Kurt, R., Tobais, H., 1997. An adaptive protocol for synchronizing media streams. Multimedia Systems Journal 5, 324–336.

Little, T.D.C., Ghafoor, A., 1990. Synchronization and storage models for multimedia objects. IEEE JSAC 8 (3), 413–427.

Louise, L., Lian, L., Renaud, B., Nicolas, G., 1996. Synchronization of multimedia data for a multimedia news on demand application. IEEE JSAC 14, 264–277.

Manvi, S.S., Venkataram, P., 2000. QoS management by mobile agents in multimedia communication, In: Proceedings of the Database and Expert Systems, Workshop on Agent-Based Intelligent Systems, Greenwich, UK, pp. 407–411.

Manvi, S.S., Venkataram, P., 2003. A multimedia synchronization model for on-line Internet based education systems. Journal of Computer Science and Informatics 33 (2), 31–42.

Manvi, S.S., Venkataram, P., 2004. Applications of agent technology in communications: a review. Computer Communications Journal 27 (15), 1493–1508.

Manvi, S.S., Venkataram, P., 2005. An agent based adaptive bandwidth allocation scheme for multimedia applications. Journal of Systems and Software 75, 305–318.

Markey, B.D., 1991. Emerging hypermedia standards: hypermedia market place prepares for HyTime and MHEG. In: Proceedings of the USENIX Conference, pp. 59–74.

Menelaos, K.P., Fotis, G.C., Iakovos, S.V., Gennaro, M., 1999. Mobile agent standards and available platforms. Computer Networks Journal 31, 1999–2016.

Miguel, C., Paulo, P., 1995. Low level multimedia synchronization algorithms on broadband networks. In: Proceedings of the ACM Multimedia Conference, California, USA, pp. 423–434.

Newcomb, S., Kipp, N., Newton, V., 1991. The HyTime hypermedia/time-based document structuring language. Communications of ACM 34, 67–83.

Nicholas, R.J., 2001. An agent-based approach for building complex software systems. Communications of ACM 44, 35–41.

Panagiotis, N.Z., Myung, J.L., Tarek, N.S., 1996. A synchronization algorithm for distributed multimedia environment. Multimedia Systems Journal 4, 1–11.

Ramachandran, R., Jim, K., Don, T., Schulzrine, H., 1994. Adaptive playout mechanisms for packetized audio applications in wide area networks. In: Proceedings of the IEEE Infocom, Canada, pp. 680–688.

Ramanathan, S., Rangan, P.V., 1993. Feedback techniques for intra-media continuity and inter-media synchronization in distributed multimedia systems. The Computer Journal 36, 19–31.

Sangshin, Y., 1998. Realization of the synchronization controller for multimedia applications. In: Proceedings of the IEEE Globecom, pp. 798–803.

Schulzrinne, H., Frederick, R., Jacobson, V., 1996, RTP: a transport protocol for real time applications, RFC 1889.

Shahab, B., Farrukh, K.M., Miae, W., et al., 1996. Quality-based evaluation of multimedia synchronization protocols for distributed multimedia information systems. IEEE JSAC 14, 1389–1403.

Shivkumar, N., Sreenan, C.J., Narendra, B., Agrawal, P., 1995. The concord algorithm for synchronization of networked multimedia streams. In: Proceedings of the IEEE ICMCS.

Steinmetz, R., 1996. Human perception of jitter and media synchronization. IEEE JSAC 14, 61–72.

Steinmetz, R., Klara, N., 1995. Multimedia Computing Communications and Applications. Prentice Hall, Englewood Cliffs, NJ.

Thomas, V.J., Aidong, Z., 1999. Dynamic playout scheduling algorithms for continuous multimedia streams. Multimedia Systems Journal 7, 312–325.

Wang, Lin, 2000. Cooperating intelligent mobile agents mechanism for distributed multimedia synchronization. In: Proceedings of the International Conference on Multimedia and Expo (ICME).

Wenyu, J., Schulzrinne, H., 1999. QoS measurement of Internet real-time multimedia services, Technical report CUCS-015–99, Columbia University.

Wong, D., Paciorek, N., Moore, D., 1999. Java based mobile agents. Communications of ACM 42, 41–48.

Yutuka, I., Takeshi, K., Shuji, T., 2004. Inter-stream synchronization between haptic media and voice in collaborative environments. In: Proceedings of the 12th ACM Conference on Multimedia, New York, USA, pp. 604–611.

Zhuge, H., 2002. Clustering soft-devices in semantic grid. IEEE Computing in Science and Engineering 4 (6), 60–62.

S.S. Manvi received M.E Degree in Electronics from U.V.C.E, Bangalore university, India in 1993, Ph.D. in Electrical Communication Engineering from Indian Institute of Science, Bangalore, India in 2004.

He is working as Professor and Head of Electronics and Communication Engineering Department, Basaveshwar Engineering College, Bagalkot, India. He is a programme committee member for international conferences, IICAI 2003, ICHMI 2004, IICAI 2005. He has coauthored a book on "Communication Protocol Engineering" published by Prentice Hall of India in 2004. His areas of interest include multimedia communications and networking, applications of mobile agents, mobile ad-hoc networks and protocol engineering. He has several national and international publications in referred conferences/journals. He is a Fellow of IETE, India and a member of IEEE, USA.

P. Venkataram received M.Sc. degree in Mathematics from Sri. Venkateswara University, Tirupathi, India, in 1973 and Ph.D. degree in Information sciences from University of Sheffield, UK, in 1986. He is currently Professor of Electrical Communication Engineering Department, Indian Institute of Science, Bangalore, India. His areas of research include wireless networks, computational intelligence in communication networks, protocol engineering and multimedia systems. He has visited several universities in India and abroad as visiting scientist and professor. He has authored a book on "Communication Protocol Engineering" published by Prentice Hall of India in 2004. He has more than 150 paper publications in referred conferences/journals, chapters in two books and edited a book. He has served in various capacities in many IEEE and ICCC conferences and workshops. He is a fellow of IEE, UK, a fellow of IETE, India, and a senior member of IEEE computer society, USA.