International Journal of Knowledge-based and Intelligent Engineering Systems 10 (2006) 139–153
IOS Press

An agent-based approach to intrastream synchronization for multimedia applications

S.S. Manvi* and P. Venkataram
Protocol Engineering and Technology (PET)-UNIT, Electrical Communication Engineering Department, Indian Institute of Science, Bangalore-560012, India
E-mail: {sunil,pallapa}@ece.iisc.ernet.in

Abstract. Multimedia synchronization control is essential to overcome the network delay variance problem in order to provide continuous and smooth playout of a multimedia stream. This paper proposes an agent-based intrastream synchronization scheme, which adapts to network delay fluctuations for continuous and smooth playout of a multimedia stream. The scheme employs three types of agents: application manager agent (AMA), user agent (UA) and negotiation/renegotiation mobile agents (NMAs/RMAs). AMA creates UA, NMAs and RMAs. UA monitors the synchronization parameters (delays, rate of change of delays, losses, etc.) as well as studies the user expectations. The NMAs/RMAs are used to negotiate/renegotiate synchronization parameters. The scheme operates in two phases: *start-up synchronization* and *resynchronization*. In *start-up synchronization* phase, a NMA is created by AMA to negotiate the delays and rate of change of delays with the intermediate nodes in the network on the basis of application requirements. In *resynchronization* phase, a RMA is created by AMA to renegotiate the delay parameters and resynchronize the presentation units of a stream, whenever UA reports violation of application requirements. Also, resynchronization phase takes care of link failures. The scheme is simulated in several network scenarios for verifying its operation effectiveness. It maintained the synchronization parameters well within the sustainable values. The benefits of the scheme are: asynchronous delay negotiation and adaptation, flexibility, adaptability and supports component based software development.

Keywords: Multimedia, synchronization, mobile agents, delays, jitters

1. Introduction

Multimedia applications such as Internet telephony, video phone, etc., demand strict real-time delivery of media units to support continuous and smooth playout at the receiver. However best effort service networks do not offer guaranteed bounded delays for the media units. Hence multimedia synchronization control is required to alleviate the problem of network delay variance. There are two types of synchronizations: *Interstream and Intrastream synchronization*. Interstream synchronization deals with maintenance of temporal relations among the multiple media streams whereas intrastream synchronization refers to maintenance of temporal relations between the media units of a stream [1]. For interstream synchronization issues, reader is referred to [2–6] since it is not in the scope of this paper.

The maintenance of temporal relationships within a stream depends on the following parameters: network delays and its variations, clock skew and drift, and end-system jitters. Network delay variations are caused due to unpredictable traffic in the network. Clocks can be synchronized using network time synchronization protocol [7]. Clock drift is eliminated by using stable crystal oscillators. End-system jitters occur due to variations in workstation loads.

In recent years, some of the static and dynamic intrastream synchronization methods have been proposed based on buffers and packet delay variations to overcome the network delay variations. The work given in [8] uses special measurement packets to estimate the

*Corresponding author.

ISSN 1327-2314/06/$17.00

voice packet delays and playout time. A detailed survey of intrastream synchronization methods for playout adaptation are discussed in [9]. The concord algorithm [10] computes the delays based on packet loss from packet delay distribution. The work given in [11] discusses adaptive playout mechanisms for packetized audio that adjusts to network delay fluctuations.

RTP (Real Time Protocol) along with RTCP (Real Time Control Protocol) are used to synchronize the media streams prior to decoding operations by computing jitters [12]. The scheme presented in [13] uses distributed jitter control at the network layer. A deadline based scheduling at each node is discussed in [14]. The work given in [15] describes the method of concealing smaller delay fluctuations by stretching voice segments and incorporating silence intervals. A video smoother is proposed that dynamically adapts various playout rates to compensate the delay variance [16]. An adaptive buffering scheme is proposed in [17], which manages the multimedia presentation by enforcing equalized delays to incoming media streams.

Quality of service (QoS) parameters (such as bandwidth, delays, etc.) reservations also help in synchronizing the presentation units of a multimedia stream. Several technologies have been developed in this context namely, ATM (asynchronous transfer mode) networks, MPLS (multiprotocol label switching) networks with constraint based label distribution, RSVP (resource reservation protocol) and Diffserv (differentiated services) [31–34].

From the literature survey, it is observed that existing intrastream synchronization schemes lacks flexibility, extensibility, customization, component reuse facility and maintainability which is needed in current Internet software development [18]. Agent technology is expected to provide the solutions to deal with these issues and also facilitate network programmability. This paper proposes an agent-based intrastream synchronization scheme that can adapt to network delay fluctuations.

1.1. Proposed work

The objective of proposed work is to provide increased flexibility in adaptation to network delay variations for continuous and smooth playout of a multimedia application. The scheme employed three types of agents: static application manager agent (AMA), static user agent (UA) and negotiation/renegotiation mobile agent (NMA/RMA). It works at the receiver/destination side. AMA creates UA, NMAs and RMAs. UA monitors the synchronization parameters as well as studies the user expectations. The scheme consists of two phases: *start-up synchronization* and *resynchronization*. In start-up synchronization phase, AMA sends a NMA from the destination to negotiate the node delays and rate of change of delays per every media unit (presentation unit). On successful start-up synchronization, transfer of media units takes place from source to destination. Even-though negotiation has been done, there is a possibility of some intermediate nodes violating the negotiation terms during running period of an application or there could be some link failures. In order to eliminate this problem, resynchronization is employed.

Here, AMA at the destination creates a RMA based on the UA recommendations and sends it across the network nodes of the path connecting source and destination to renegotiate the delays and rate of change of delays per every media unit. A UA understands expectations of a user and gets performance from the network. The creation of RMAs may be based on application requirements, which may fall under either of the following categories: 1) reaching certain acceptable threshold media unit loss; 2) variation of change in inter-arrival media unit time beyond the expected value; 3) variation in end-to-end delays beyond the sustainable delays at certain instant of time. Also, NMAs may be created upon getting link failure report from the intermediate nodes to negotiate synchronization parameters on the new path during resynchronization phase.

1.2. Paper organization

The following section presents a brief background and concept of agent technology. Section 3 describes the playout system model, proposed intrastream synchronization technique and some of the benefits of using mobile agents for synchronization. Simulation of the proposed work and its results are presented in Sections 4 and 5, respectively. Finally we conclude the proposed work in Section 6.

2. Agent technology

Agents are the autonomous programs situated within an environment (either host or network), which sense the environment and acts upon the environment 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* [19–21]. Agents can be classified based on properties they posses: *local or user interface agents, network agents, distributed AI* (*Artificial Intelligence*) *agents and mobile agents*. The network agents and user interface agents are usually termed as single agent systems whereas the other two agents are called as multiagent systems.

Mobile agent is an itinerant agent dispatched from a source node which contains program, data, and execution state information, migrates from one node to another node in the heterogeneous network and executes at remote host until it achieves its goals [22, 23]. Mobile code should be platform independent, so that, it can execute at any remote host in the heterogeneous network environment. Mobile agents code can be written in Java, TCL, Perl, or XML languages. Agents communicate and cooperate with other agents to achieve their goals. Inter-agent communication can be achieved by using message passing, remote procedure calls or common knowledge base (black board architecture). An agent platform supports the following services: agent creation, agent reception, agent execution, agent migration, inter-agent communication, persistence, fault tolerance and security [27]. Some of the popular agent platforms are IBM Aglets, Grasshopper, Voyager, Agent TCL, Tacoma and Mole. Agent based schemes offer several benefits as compared to traditional approaches: overcomes latency; reduces network traffic; encapsulate protocols; flexibility; adaptability; software reusability and maintainability and support component based software development [24, 25].

3. Proposed synchronization technique

The proposed intrastream synchronization technique adapts to network delay variations by negotiating and renegotiating delays and rate of change of delays with the intermediate nodes in the path on the basis of application requirements and the user expectations. It assumes that the destination gets the information about the route (from source to destination), link capacities and link propagation delays from the route finding agency created by an application manager agent as and when required [30]. Discussion of route finding agency is beyond the scope of this paper since our intention is to describe the synchronization scheme. The scheme operates in two phases: *start-up* synchronization and *resynchronization* with a synchronization agency operating at the receiver side.

3.1. Playout system model

The periodic media units (presentation unit) of a stream emitted from a source that are transported to destination through intermediate nodes have to be synchronized and played back at the destination (see Fig. 1). The characteristics of the playout system model are as follows.

- The presentation unit (PU) of a stream is chosen as the unit perceivable by the human beings, for example, a frame is a perceivable PU of a video stream.
- The PUs of a stream are labeled with sequence numbers, and they are presented in same sequence.
- Some intermediate nodes may violate the delay agreements made with the hosts with certain probability due to congestion at a node and node unreliability.
- The queuing delay of a PU in each non agreement-violating intermediate node vary as per the agreed delays and rate of change of delays.
- The queuing delay of a PU in each agreement-violating intermediate node vary more than the agreed delays and rate of change of delays.
- Clock differences and drifts are assumed to be zero between the source and destination.
- Intermediate nodes use deadline based scheduling for each arrived PU depending on agreed delays and rate of change of delays. For example, for a non agreement-violating intermediate node, if agreed delay for a first $PU = 0.05$ seconds and rate of change of delay $= 0.001$ seconds per each PU, then the queuing delay for 100^{th} PU is: $0.05 + 0.001 * 99 = 0.0599$ seconds. In case of agreement-violating nodes, rate of change in delays may rise above the agreed values at regular intervals at most by 100%.
- An agent platform exists at every intermediate node and the end-systems. However, in case of agent platform not available, the agents use traditional message exchange mechanisms to perform its task.
- All the intermediate nodes support negotiation/ renegotiation of delay parameters. However, statistical delay parameter values will be considered for nodes that do not support reservations, in case of unavailability of a path with all nodes in the path supporting reservations.

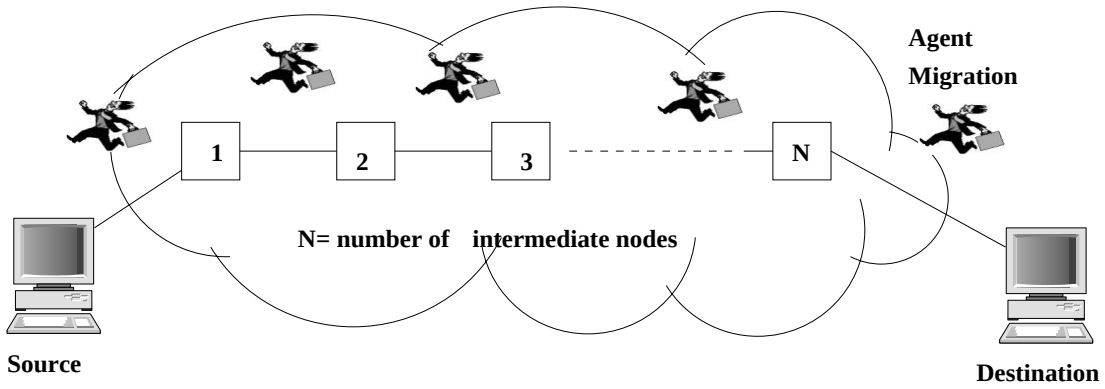

Fig. 1. Playout system model.

3.2. Synchronization scheme

On invoking a multimedia application, it first runs start-up synchronization for initial synchronization parameters negotiation with intermediate nodes at every node in the path. Later it performs resynchronization, if synchronization requirements are violated or link failure occurs. Synchronization agency at a receiver is depicted in Fig. 2. The scheme employs three types of agents: static user agent, static application manager agent and negotiation/renegotiation mobile agents. It uses a knowledge base for inter-agent communication that works based on blackboard principle. Components of the synchronization agency are explained.

Knowledge base: It consists of following data: application id, expected length of the session, PU size, playout duration of PU, number of PUs to be played, number of played PUs, playout time of the PUs, quality feedback from user (positive or negative value), acceptable end-to-end delay, sustainable end-to-end delay, acceptable loss, measured loss, jitters at the receiver, server transmission start time, fixed delays at server and client (includes packetization/depacketization, encoding/decoding), measured end-to-end delays, required and measured jitters at every node on the route, required and measured rate of change of delays at every node on the route, required and measured delays at every node on the route, available routes between source and destination, maximum capacity and propagation delays of links. This knowledge base is used by agents to share the synchronization related information and update with latest information by seeking permission from the application manager agent.

Application manager agent (*AMA*): It is a static agent created by an application which synchronizes the activities of other agents in the agency. User agent and negotiation/renegotiation agents and knowledge base are created by this agent. The agent acquires application requirements by communicating with the server, client and the user. It gets the route and its links information between source and destination by communicating with the route finding agency during synchronization and resynchronization. Agent computes average queuing delays required at each node to satisfy the delay requirement of an application and performs start-up synchronization by using negotiation agent and reserves certain amount of buffers for continuous playout. Later, it computes playout time of PUs, which will be used by the application for playout. On predicting the violation of application requirements, user expectations given by the user agent, it triggers renegotiation agent to perform resynchronization. Also, it creates a negotiation agent and sends on a new route for resynchronization if the link failures on the existing route are reported by intermediate nodes. Once negotiation/renegotiation is done, recomputes buffer requirement and the playout time of the PUs to be played.

User agent (*UA*): It is a static agent which monitors the PU losses, number of PUs played, end-to-end delays, rate of change of delays and jitters, and updates the knowledge base. The agent periodically interacts with the user to know his/her expectations about the quality of presentation by using its GUI (graphical user interface). If the user is not satisfied, UA advices the AMA to perform resynchronization either on the currently used route or another route (if current route is either approaching congestion mark or a link is failed). The routes are chosen by AMA in consultation with route finding agency but not by UA.

Negotiation/renegotiation mobile agent (*NMA/RMA*): This is a mobile agent which roams in the given route to negotiate/renegotiate the delays and rate of

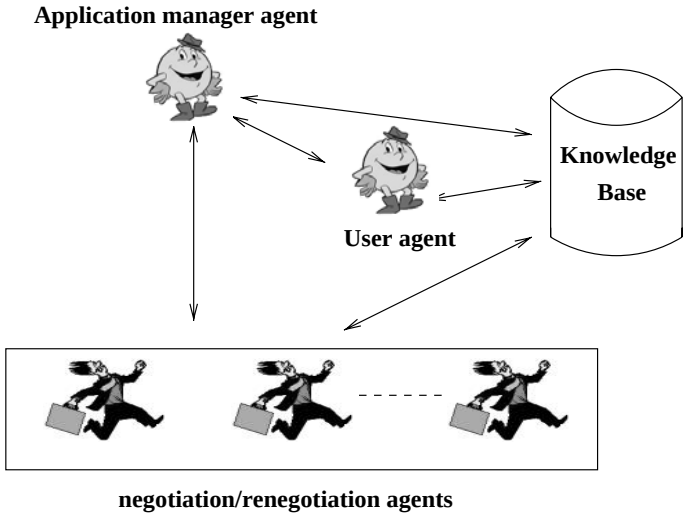

Fig. 2. Synchronization agency at the receiver.

change of delays such that end-to-end delays and jitters are within the limits of application requirements so as to provide good quality presentation. It updates the knowledge base with new reserved delays and rate of change of delays at every node in the path.

3.2.1. Start-up synchronization

The scheme computes delays and rate of change of delays to be negotiated along the nodes in the path connecting source and destination on the basis of following parameters: link capacities, fixed delays at source and destination (packetisation, encoding/decoding), expected duration of the call, sustainable end-to-end delay and better delivery end-to-end delays [35]. The worst case sustainable delays of an application can go even beyond the sustainable end-to-end delays depending on the perception of an individual.

In this phase, a NMA is sent from the destination on the path to negotiate delays and rate of change of delays in each intermediate node of the path. Negotiated delay parameters at each node in the path will be used to setup the node buffers. Algorithm 1 presents the pseudo-code of this phase. Considering the system model as shown in Fig. 1, a mobile agent makes visit to the nodes, *N, N-1,. . . , and 1*, in sequence for negotiating the delay parameters.

Algorithm 1. Start-up synchronization {*Nomenclature:* $lpd_i = i^{th}$ link propagation delay, $C_i =$ capacity of i^{th} link, $f_d =$ fixed delays (packetisation/depacketisation, encoding/decoding delays at source and destination), $S = PU$ size, $N =$ number of intermediate nodes, $bacd =$ better acceptable end-to-end delay requirement, $sd =$ sustainable end-to-end delay requirement, $l =$ expected length of the session, $pud =$ length of presentation/generation duration of a PU (if 20 PUs/sec, then PU duration is 50 ms), $eed =$ end-to-end delay, $jit =$ jitters, $sts =$ server transmission start time, $V =$ Visiting itinerary of a mobile agent, $src =$ source, $ptime^j =$ playout time of j^{th} PU bqd and sqd are better and sustainable average queuing delay at an intermediate node, respectively, D^i and dD^i are delay and rate of change of delay per PU at intermediate node i, respectively}

Begin

1. AMA updates the knowledge base with application requirements and the routes between the source and destination along with status information of links of the routes;
2. AMA computes the average queuing delays bqd and sqd to satisfy the delay requirements of an application;

$$sqd = (sd - \sum_{i=1}^{N+1} lpd_i + (2 * f_d) + \sum_{i=1}^{N}(S/C_i))/N \quad (1)$$

$$bqd = (bacd - \sum_{i=1}^{N+1} lpd_i + (2 * f_d) \quad (2)$$

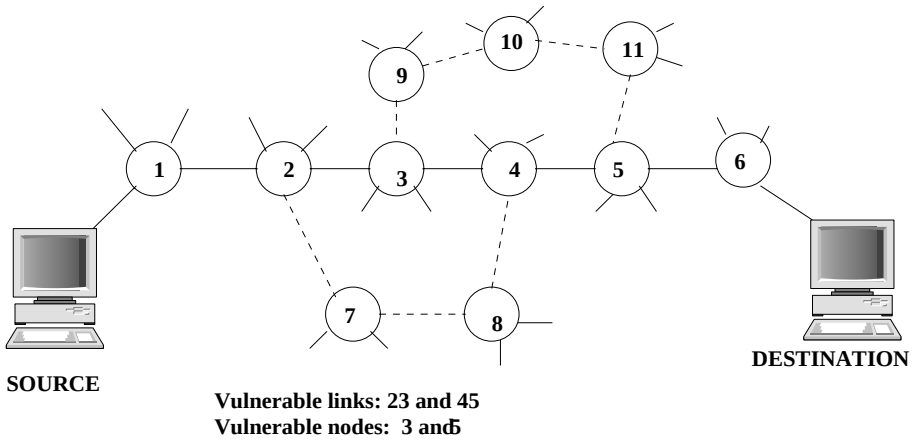

Fig. 3. Simulation model.

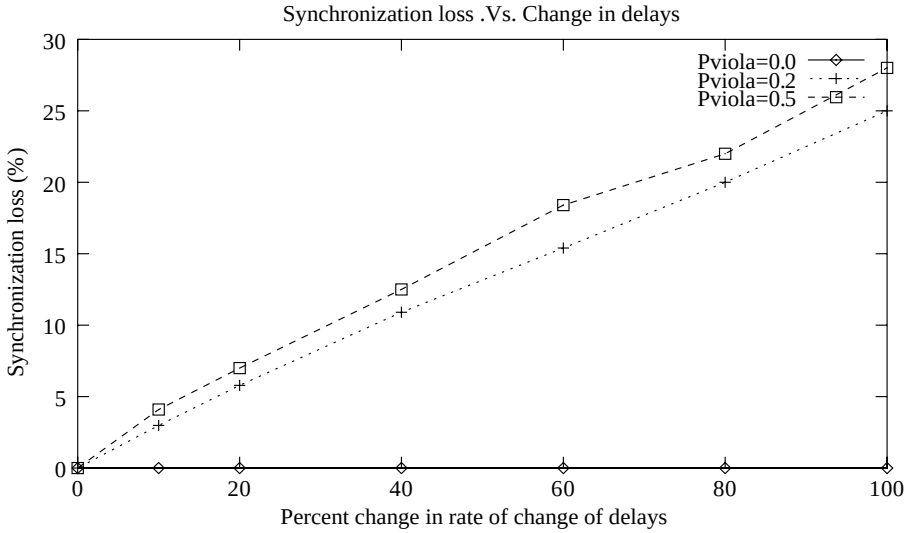

Fig. 4. CASE I: Synchronization loss (%) .Vs. Percent change in rate of change of delays.

$$+\sum_{i=1}^{N}(S/C_i))/N$$

{*Note:* The equations 1 and 2 consists of propagation delays across the path, the fixed delays, and transmission delay at the intermediate node.}

3. AMA creates a NMA to negotiate D^i and dD^i with N intermediate nodes of the path connecting source and destination
4. NMA initializes its visiting sequence (addresses of nodes to be visited, $V = \{v_n, v_{n-1}, ..., 1, src\}$), $jit = 0$, and $eed = 0$;
5. For $i = 1$ to N do /* agent travels from destination to source*/
   Begin
   - NMA migrates to upstream node i;
   - NMA negotiates the delay D^i within the range $[0, bqd]$;
   - NMA computes the rate of change of delay per PU, $dD^i = (sqd - D^i) * pud/l$ and negotiates this with the node i;
   - NMA updates jitters and end-to-end delays: $jit = jit + sqd - D^i$, and $eed = eed + D^i$

   End for i;
6. NMA after reaching the source, sends a message to AMA at the destination about the sts, D^i, dD^i, jit and eed, and requests server to start transmission, and then disposes;
7. AMA computes the total eed by including fixed, link propagation and transmission delays and up-

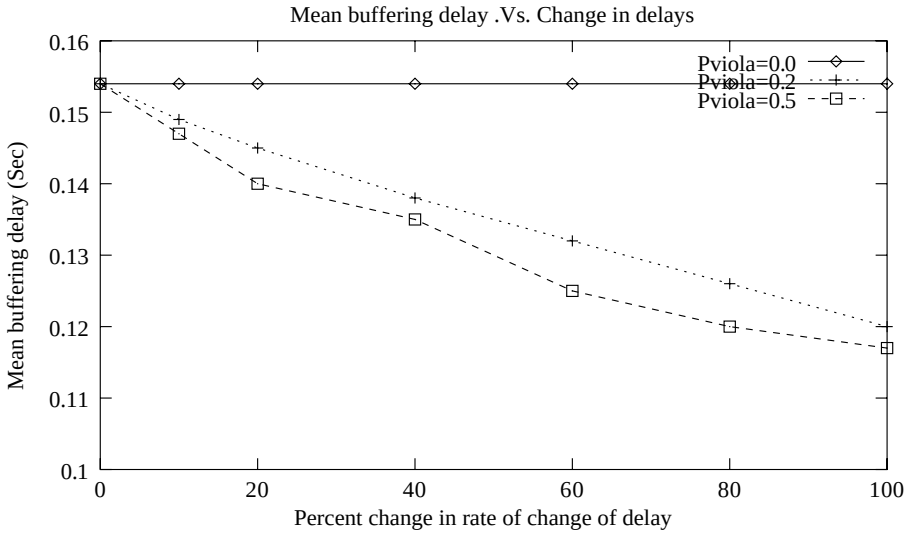

Fig. 5. CASE I: Mean buffering delays .Vs. Percent change in rate of change of delays.

dates the knowledge base:

$$eed = eed + \sum_{i=1}^{N+1} +(2 * f_d) + \sum_{i=1}^{N}(S/C_i) \quad (3)$$

8. AMA computes playout time for the PUs and updates the knowledge base

$$ptime^j = sts + (eed + jit) + (j-1) * pud \quad (4)$$

9. AMA requests UA to monitor the synchronization parameters (delays, rate of change of delays and PU losses) and also requests to interact with the user about the feedback of the presentation quality;
10. Application starts playing PUs only after buffering for a duration of jit time units;
11. Stop.

End.

3.2.2. Resynchronization

Even-though the intermediate nodes have agreed for negotiated delay parameters, some of the nodes in the path may be unreliable to provide the guaranteed delay service because of following reasons: 1) overloaded traffic passing through them; 2) node may be trying to maximize its revenue by promising guaranteed services and allowing more number of users than it can handle. Some times the links may fail. In such cases, the end-users will experience asynchrony in their media presentations: hence a poor quality presentation of data. In order to handle these types of problems, resynchronization phase is necessary to detect the delay parameter violations and link failures, and send a RMA to renegotiate the delay parameters either on the existing or alternate route and adapt to the current network environment by resynchronizing the media units of a stream.

The creation of RMAs may be based on following application requirements, which may fall under either of the following categories: 1) on detection of losses and reaching sustainable loss; 2) on detection of increase in rate of change of delays; 3) on detection of increase in end-to-end delays beyond the sustainable delays at certain instant of time. The NMA may also be created on getting link failure report from the intermediate nodes to negotiate the synchronization parameters on a new route. Algorithm 2 presents the pseudocode for resynchronization method, which uses sustainable loss parameter and link failures for invoking resynchronization.

Algorithm 2. Resynchronization { *Nomenclature:* al = acceptable loss (%), x = normalized value, npu = number of PUs to be generated/played (l/pud), $nplay$ = number of played PUs, $loss$ = percentage of lost packets, CD^i = current queuing delay of the recently transmitted packet at i^{th} node, cdD^i = current rate of change of delay per PU duration, $feed\_back$

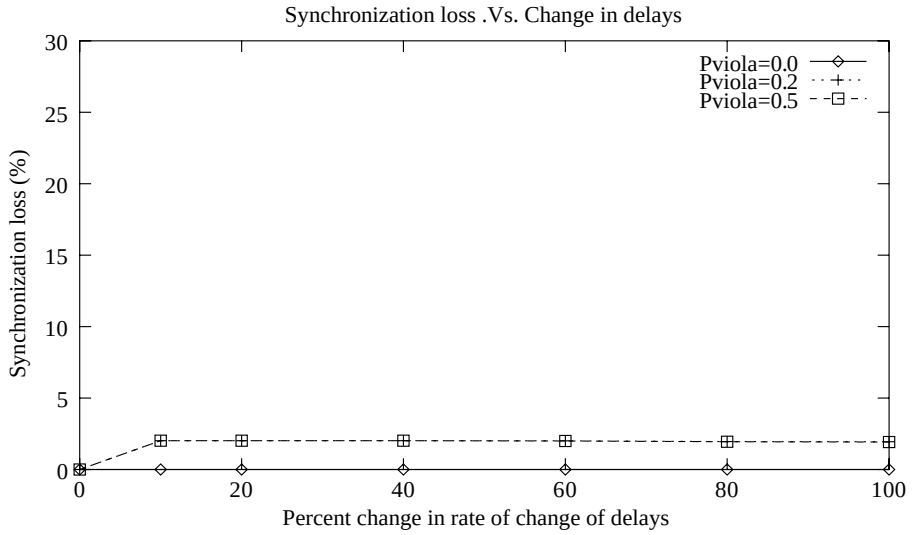

Fig. 6. CASE II: Synchronization loss (%) .Vs. Percent change in rate of change of delays.

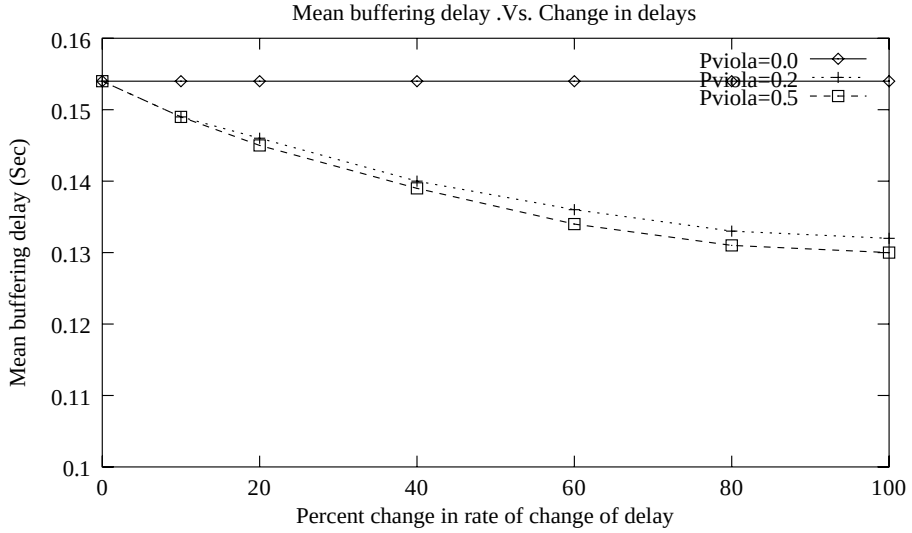

Fig. 7. CASE II: Mean buffering delays .Vs. Percent change in rate of change of delays.

feedback from the user (denotes negative or positive value)}

Begin

1. UA at the destination computes the percentage of late arrived ($loss$) PUs and also gets the feedback from the user and updates the knowledge base.
2. AMA performs following action: If (($loss \geqslant (x * al)$) and ($feed\_back =$ negative)), gets the current route from the knowledge base, then goto step 4;
3. If link failure is reported from intermediate node, AMA requests an alternate route from the routing agency that does not use the failed links, updates the knowledge base, then perform start-up synchronization (CALL Algorithm 1) for the remaining length of the session; goto step 9;
4. AMA creates a RMA to renegotiate the D^i, dD^i, and recompute the jit and eed} on the given route;
5. For $i = 1$ to N do
   Begin
   - RMA migrates to upstream node i.
   - RMA negotiates the delay D^i within the range $[D^i, CD^i]$ at node i.

– RMA negotiates the dD^i to original agreed dD^i, if cdD^i is greater than dD^i at node i.

End for i;
{*Note*: The non agreement-violating intermediate nodes will negotiate with probability $1/nt$, where nt is the number of times renegotiation has taken place for the session. The agreement-violating nodes will negotiate with probability 1.}

6. RMA after reaching the source, recomputes jit and eed and informs the AMA at the destination about the renegotiated D^i and dD^i and disposes. The jit and eed are computed as follows:

$$eed = \sum_{i=1}^{N+1} lpd_i + (2 * f_d) + \sum_{i=1}^{N}(S/C_i) + \sum_{i=1}^{N} D^i \tag{5}$$

$$jit = \sum_{i=1}^{N} dD^i * (npu - nplay) \tag{6}$$

7. AMA computes the playout time for remaining PUs by using renegotiation completed time, eed, jit, and pud as given in step 8 of Algorithm 1. Updates the knowledge base;
8. Application updates its jitter buffers based on jit and plays out the subsequent PUs;
9. Stop.

End.

Resynchronization methods can also be invoked by considering the rate of change of delay variations at the destination as said earlier. For this purpose, step 2 has to be modified in algorithm 2, i.e., if AMA observes increase in rate of change of interarrival time of consecutively received packets for certain amount of time, it invokes the RMA for renegotiating the delays and rate of change of delays in the nodes along the path.

In a situation, where at random intervals of time, the sustainable end-to-end delay requirement vary (may vary from 90% to 150% of the sustainable delays) for an application because of unavailability of resources at intermediate nodes, resynchronization is invoked based on the current sustainable end-to-end delay requirements. For this purpose the modifications to algorithm 2 should be done in steps 1 and 2. In step 1, determine the current end-to-end delays of the PUs and sustainable end-to-end delays, and in step 2, goto step 4, if current end-to-end delay is greater than the sustainable end-to-end delays.

If the PUs to be presented does not arrive within the duration of skew tolerance, application uses skew compensation mechanisms for playout. These mechanisms are: restricted blocking (display the last frame to deal with the losses and delayed frames) and blocking (do not play anything or stretch the playout duration of existing PUs) for video and audio streams, respectively [26].

3.3. Benefits of using agents

Agent oriented programming facilitates component based software engineering (CBSE) which is needed in today's software development of web-based systems [18,25]. In future there will be enormous number of agents (agents are next generation components) which have to coordinate with each other to provide multimedia information searching, retrieval and communication services, once the agent platform becomes standardized.

As observed from the literature [28–30], 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 agent and improve the performance of multimedia communication systems.

Some of the benefits of using the agents for synchronization purpose are as follows.

– *Flexibility*: Flexibility in delay adaptation policies by changing agent code facilitates personalizing of customer services, i.e., adaptation policies may be coded to depend on current and predicted network traffic as well as on the link/node failures. NMA/RMA can be even coded to negotiate based on some pricing models as used by the customers.
– *Adaptability*: The agency used in synchronization copes up with a dynamic, heterogeneous and open environment which is characteristic of today's Internet. The agents dynamically either adapt or renegotiate to maintain the synchronized presentations based on application requirements and network problems such as congestion and link failures.

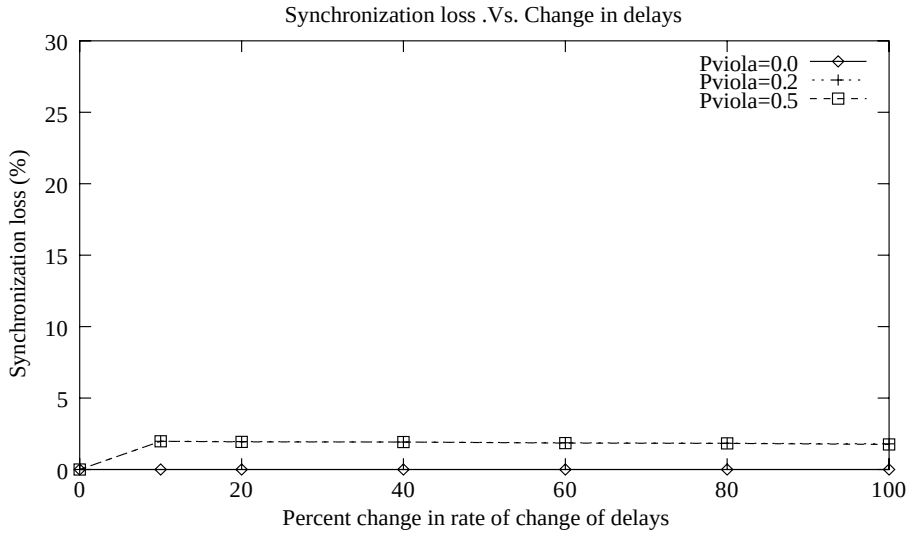

Fig. 8. CASE III: Synchronization loss (%) .Vs. Percent change in rate of change of delays.

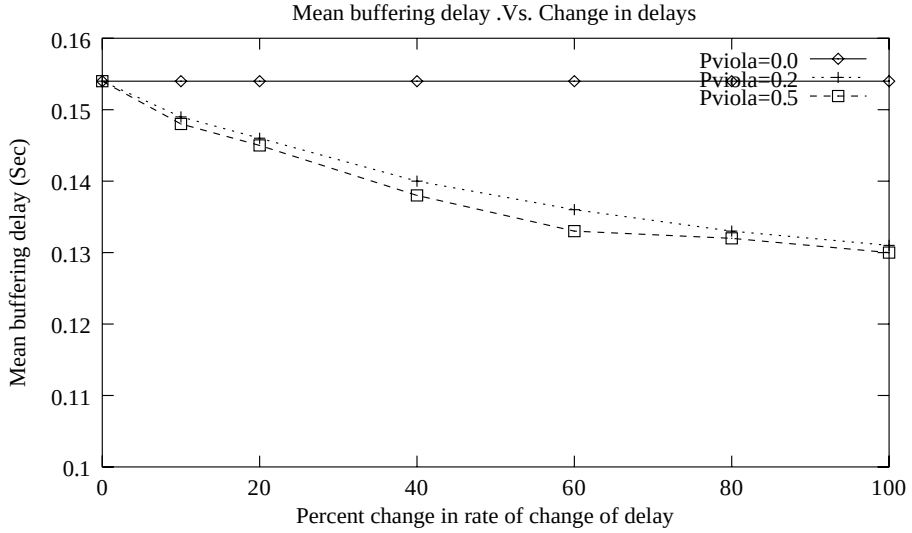

Fig. 9. CASE III: Mean buffering delays .Vs. Percent change in rate of change of delays.

– *Reusability*: Part of the agent software can be reused for allocating bandwidth and buffers required for any kind of multimedia application. Reusing of code is possible because of autonomous operation of agents involved in multimedia services.
– *Maintainability*: Since the agents developed for synchronization purpose is developed on a modular approach in combination with other agents involved in the multimedia services, it is easy to debug and update the agent software.
– *Customizability*: The agent software can be customized to the user needs and the application requirements by encoding the delays and rate of change of delays in the agent.
– *Scalability*: The agent sent from a host of a network for delay parameters negotiation or renegotiation can be made to perform aggregate tasks for other hosts of the network that are connected to the same source.
– *Asynchronous operation*: RMAs/NMAs sent across the network do not need permanent connections to the hosts. As and when required the agents will send the information to AMA. This feature

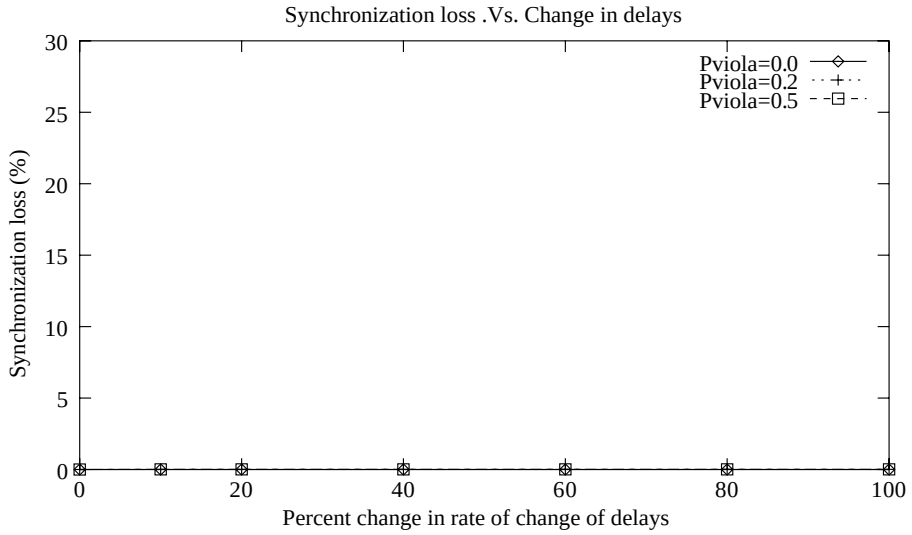

Fig. 10. CASE IV: Synchronization loss (%) .Vs. Percent change in rate of change of delays.

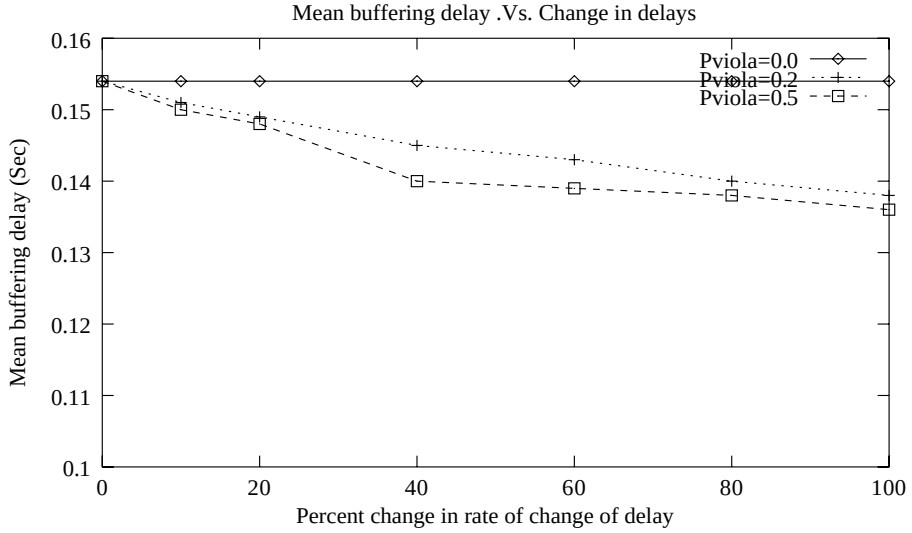

Fig. 11. CASE IV: Mean buffering delays .Vs. Percent change in rate of change of delays.

is particularly useful in mobile networks, where no permanent connection wastes bandwidth and computation time of the manager agent.

4. Simulation

In this simulation, we consider invoking of RMAs during resynchronization phase on the basis of application categories as discussed in Section 3.2.2. We generated several applications, which fall into above mentioned categories. In this section we give a network traffic model to test the proposed scheme, simulation procedures and the results.

4.1. Network model

We consider a network simulation model as shown in Fig. 3 for testing the proposed intrastream synchronization scheme. The model has 13 nodes in which primary path between source and destination is 1-2-3-4-5-6. There are 2 unreliable intermediate nodes (3 and 5) which may violate the delay and rate of change of delay agreements. Also, there are two unreliable links

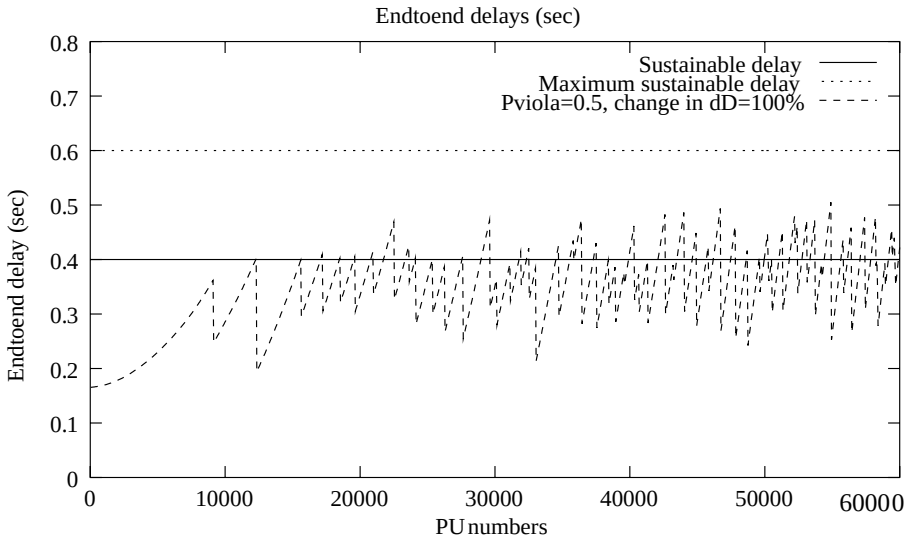

Fig. 12. End-to-end delays (sec) for each PU (only the first 60,000 PUs delays are shown) considering better delivery delays = 200 ms, sustainable delays = 400 ms, and maximum sustainable delays = 600 ms (1.5 times of 400 ms).

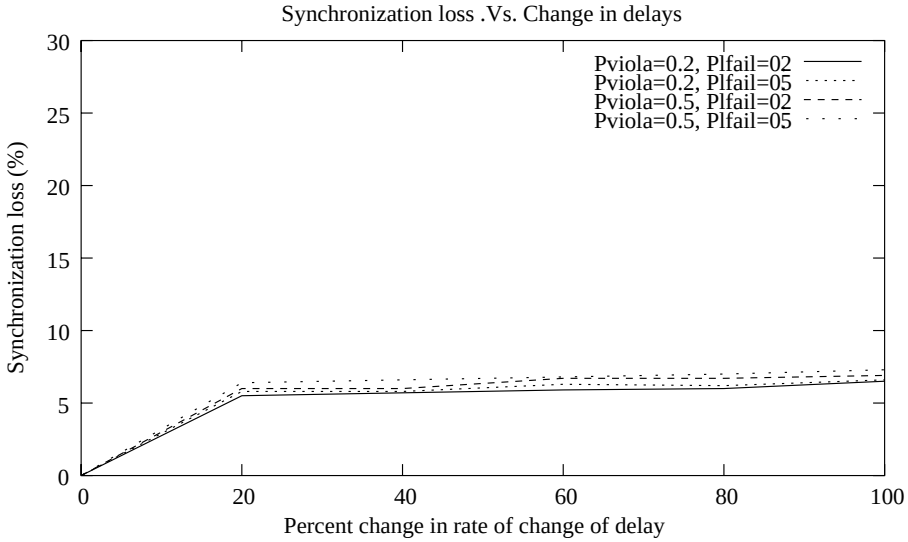

Fig. 13. CASE V: Synchronization .Vs. Percent change in rate of change of delay.

2–3 and 4–5, which may fail. Alternate paths between source and destination are 1-2-7-8-4-5-6 and 1-2-3-9-10-11-5-6 for failed links 2–3 and 4–5, respectively. Probability of a link failure may be designated as P_{lfail} for every 100 PUs delivery.

An agreement-violating node may violate with a maximum probability of P_{viola} at every regular time interval. Rate of change of delays may change randomly in an agreement violating node. The fixed delays of an application are *fd*. The maximum capacity *C* of all the links are same. The propagation delays *lpd* of all the links are equal. Other parameters of the the application requirements are acceptable delay ($bacd$), sustainable delay (sd), worst case sustainable delay by application by using playout compensation mechanisms (msd), PU size (S), PU presentation duration (pud), length of the session (l), and acceptable loss (al). The agreement-violating node varies rate of change of delays at regular intervals of x PUs. Agent creation, migration, and execution time is distributed uniformly in the range $(50, 200)$ ms. as observed in experiment conducted using IBM aglets work bench [26,27]. Rate of

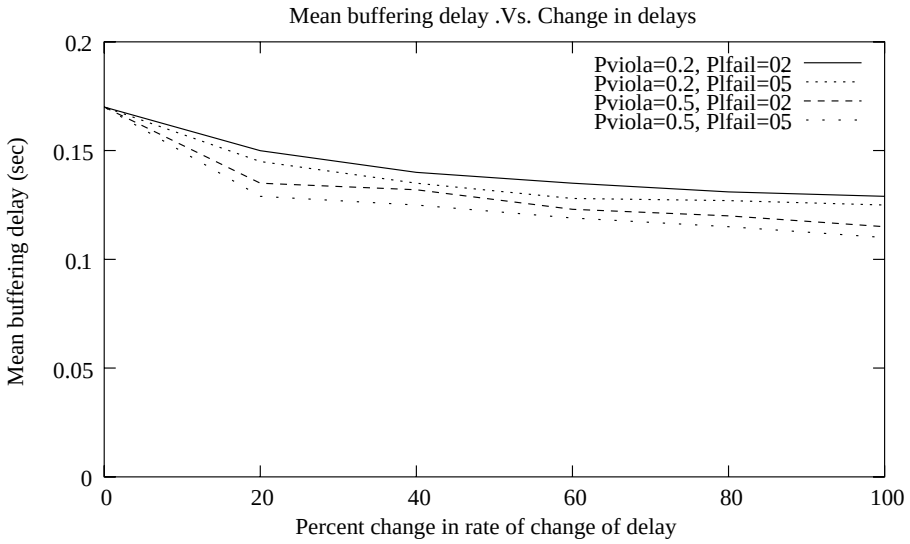

Fig. 14. CASE V: Mean buffering delay .Vs. Percent change in rate of change of delay.

change of delay at a node per PU duration varied from LV to HV.

4.2. Simulation procedure

The inputs considered in the simulation are: $bacd = 200$ ms [35], $sd = 400$ ms., $msd = 600$ msec; $lpd = 1$ ms., $C = 50$ Mbps; $f_d = 10$ ms, $al = 2.0\%$; $S = 1024$ bytes, $pud = 50$ ms., P_{viola} (violation probability) is considered for different cases as 0.0, 0.2 and 0.5 for nodes 3 and 5 respectively, $LV = 5\%$ to $HV = 100\%$; $l = 600$ sec., and $x = 100$ PUs, P_{lfail} is considered for cases 0.0 and 0.5.

Begin

- Run the simulation program for following cases by changing the rate of change of delay variations and evaluated the performance parameters.
  - CASE I: Uses only start-up synchronization.
  - CASE II: Uses start-up synchronization, and resynchronization is invoked at the instant of losses exceeding acceptable PU losses.
  - CASE III: Uses start-up synchronization, and resynchronization is invoked at the instant of losses exceeding 90% of the acceptable PU losses.
  - CASE IV: Uses start-up synchronization and invokes resynchronization whenever rate of change of end-to-end delays vary more than agreed values for consecutive number of PUs.
  - CASE V: Uses CASE III with link failures
- The simulation program also ran for a large session of 100 minutes, in which resynchronization is invoked whenever end-to-end delays of the PUs exceed the certain percentage of sustainable delays at every regular intervals of time. The percentage of required sustainable delays is uniformly distributed (initially it starts with 90% of sustainable delays).

End.

The performance parameters evaluated in the simulation are as follows.

- *Synchronization loss*: It is defined as the percentage of PU loss (late arrivals) of the stream in a presentation period;
- *Mean buffering delay*: It is defined as the mean waiting time of presented PUs of a stream in the receiver;
- *Agent overheads*: It is defined as the number of agents created during resynchronization.
- *Latency*: It is the time required for renegotiation either during application requirement violations or link failures.

5. Results

We observe that without applying the resynchronization (see Figs 4 and 5) for an application, synchronization loss increases and the mean buffering delay reduces with increase in probability of agreement violation and

percent changes in rate of change of delays. The user experiences degradation in the presentation quality due to more losses if the nodes violate the agreements and delays crop up abruptly.

If resynchronization is used, we notice that the synchronization losses are well within the desired/ sustainable losses (within 2% for CASE III and around 2% for CASE II) even with the increase in probability of agreement violation and the rate of change of delays (see Figs 6 and 8). The buffering delays (see Figs 7 and 9) in CASE II and CASE III are little higher as compared to CASE I so as to avoid the losses.

If resynchronization is invoked at the instant of increase in rate of change of delay losses will be almost nearing zero (see Fig. 10) even in case of agreement violation and rise in rate of change of delay at the expense of little more agent overheads as compared to CASE II and CASE III. Mean buffering delays (Fig. 11) are more here as compared to other cases.

The agent overheads observed during resynchronization are: 39, 35 and 52 for CASE II, CASE III and CASE IV, respectively with $P_{viola} = 0.2$; 74, 74 and 96 for CASE II, CASE III and CASE IV, respectively with $P_{viola} = 0.5$.

Finally we conducted the simulation of the synchronization scheme (using $P_{viola} = 0.5$ and rise in rate of change of delay = 100%) for a larger session duration of 100 minutes, in which resynchronization is invoked if observed end-to-end delays of PUs exceed the certain percentage of sustainable delays (sustainable delays are assumed to be uniformly distributed within the range $[0.9 * Sd, 600]$) at every intervals of 100 PUs. This simulation is conducted to view the variation in end-to-end delays of the PUs (see Fig. 12). We observe that end-to-end delays of the PUs are well controlled by the resyncronization phase. The delays vary within the best delivery delay (200 ms) to maximum sustainable delays (600 ms).

The agent overheads noticed in this simulation is 283. The overheads are very much minimal as compared to the flexible services offered to the users. For an agent with size of 4 KB, bandwidth required by the agents is 4KB*283 = 9.27 Mb for a session of duration 6000 sec: hence 1.54 Kbits/sec is the bandwidth overhead for an application. Suppose, if there are 500 applications using this scheme over the path, the overheads will amount to 770 Kbits/sec which is very small percentage of the path bandwidth (7*50 Mbps = 350 Mbps).

It is noticed from the Figs 13 and 14 that the synchronization losses are little increased with link failures since some time is required to setup a new route and resynchronize the presentations. During this period some PUs will be lost. However, the scheme immediately renegotiates on another route (with the assumption that the new route has enough resources). In case of resources not available on a new route, application is terminated by AMA on getting information from NMA. Latency in resynchronization during link failures varied from 600 ms. to 800 ms.

6. Conclusions

The paper presented an agent-based intrastream synchronization scheme, which synchronizes the media units of a stream under the conditions of network delay variance problems and link failures. The scheme used an agency employing agents to negotiate, renegotiate and adapt the delays and rate of change of delays with the intermediate nodes in the network based on customized parameters of an application (sustainable loss, sustainable delays, sustainable rate of change of delays) for resynchronizing the presentation units within the stream. The simulation results showed that the scheme maintains the synchronization parameters of an application well within the sustainable values even under variations of network delays and link failures. Agent-based schemes facilitate software reuse and maintenance. However, some problems have to be resolved in agent system implementation since it is in infant stage: creation of a standardized agent platform for Internet to facilitate development of agent based applications, security to agents from hosts and vice versa, and agent creation tools. There are some efforts made by standard bodies such as FIPA (Foundation of Intelligent Physical Agents) and OMG (Object Management Group) [19]. It is soon expected that agent technology will become a reality within few years especially in the areas of network management, information management, e-commerce and peer to peer computing.

References

[1] G. Blakowski and R. Steinmetz, Media Synchronization Survey: Reference Model, Specification, and Case Studies, *IEEE JSAC* **14**(1) (1996), 5–35.

[2] A. Zhang, Yuqing and M. Mielke, Netmedia: Streaming Multimedia presentations in distributed environments, *IEEE Multimedia mag* (2002), 56–73.

[3] K. Rothermel and T. Helbig, An adaptive protocol for synchronizing media streams, *ACM Multimedia Systems* **5** (1997), 324–336.

[4] C. Huang and R. Lee, Achieving multimedia synchronization between live video and live audio streams using QoS controls, *Computer Communications* **19** (1996), 456–467.
[5] E. Biersack and W. Geyer, Synchronized delivery and playout of distributed stored multimedia streams, *ACM Multimedia systems* **7** (1999), 70–90.
[6] L. Lamont, L. Li, R. Brimont and N. Georganas, Synchronization of Multimedia data for a Multimedia News on demand Application, *IEEE JSAC* **14** (1996), 264–277.
[7] D.L. Mills, Internet time synchronization: Network time protocol, *IEEE trans. communications* **39** (1991), 1482–1493.
[8] F.A. Bertran, F. Oller and J.M. Selga, Voice synchronization in packet switching networks, *IEEE Network. mag* (1993).
[9] N. Laoutaris and I. Stavrakakis, Intrastream synchronization for continuous media streams: A survey of playout schedulers, *IEEE network mag* (2002).
[10] N. Shivkumar, C.J. Sreenan, B. Narendra and P. Agrawal, The concord algorithm for synchronization of networked multimedia streams, *Proc. ICMCS* (1995).
[11] R. Ramjee, J. Kurose, D. Towsley and H. Schulzrinnie, Adaptive playout mechanisms for packetized audio applications in Wide Area Networks, *Proc. IEEE Infocom*, Canada, 1994, 680–688.
[12] H. Schulzrinne, R. Frederick and V. Jacobson, RTP: A transport protocol for real time applications, *RFC 1889* (1996).
[13] S. Kadur, F. Gashing and B. Millard, Delay-jitter control in multimedia applications, *Multimedia systems Journal* **4** (1996), 30–39.
[14] D. Pern, H. Zhong and D. Ferrari, Delay jitter control for real time communication in a packet switching network, *Proceedings of IEEE Tricomm* (1991), 35–43.
[15] F. Liu, J. Kim and C. Jay Kuo, Adaptive delay concealment for Internet voice applications with packet based time scale modification, *IEEE ICASSP* (2001), 1461–1465.
[16] M.C. Yuang, S.T. Liang, Y.G. Chen and C. L-Sen, Dynamic video playout smoothing method for multimedia applications, *Proc. IEEE ICC* (1996), 1365–1369.
[17] Y. Xie, C. Liu, M.J. Lee and T.N. Saadwi, Adaptive multimedia synchronization in a teleconference system, *Multimedia System Journal* **7** (1999), 326–337.
[18] M.L. Griss and G. Pour, Accelerating development with agent components, *IEEE Comp. Mag* **34**(5) (2001), 37–43.
[19] S.S. Manvi and P. Venkataram, Applications of agent technology in communications: A review, *Computer communications* **27**(15) (2004), 1493–1508.
[20] Cetus links, http://www.cetus-links.org/00_mobile_agents.html.
[21] D. Chess, N. Benjamin, C. Harrison, D. Levine and C. Paris, Itinerant agents in mobile computing, *IEEE personal communication* (1995), 35–49.
[22] D. Wong, N. Paciorek and D. Moore, Java based mobile agents, *Communications of ACM* **42** (1999).
[23] D. Chess, C. Harrison and A. Kershenbaum, *Mobile Agents: Are They a Good Idea?* IBM Research Division, T.J. Watson Research Center, Yorktown Heights, New York, March 1995.
[24] D.B. Lange and M. Oshima, Seven good reasons for mobile agents, *Communications of ACM* **42** (1999), 88–89.
[25] N.R. Jennings, An agent-based approach for building complex software systems, *Communications of ACM* **44** (April, 2001), 35–41.
[26] S.S. Manvi and P. Venkataram, Multimedia synchronization model for Internet based education systems using agents, *Journal of Comp.Sc. and Informatics* **32**(2) (June, 2003), 30–41.
[27] IBM Aglets work bench, http://www.trl.ibm.co.jp/aglets.
[28] S.S. Manvi and P. Venkataram, QoS Management by Mobile Agents in Multimedia Communication, *Database and Expert Systems* (*DEXA 2000*), *Agent-Based Intelligent Systems ABIS-2000*, Greenwich, U.K, Sept. 2000, 407–411.
[29] S.S. Manvi and P. Venkataram, Mobile Agent based Online Bandwidth Allocation Scheme in Multimedia Communications, *IEEE GLOBECOM*, San Antonio, USA, Nov. 2001, 2622–2626.
[30] K. Oida and M. Sekido, An agent-based routing system for QoS guarantees, *Proc. IEEE Int. Conf SMC* (1999), 33–38.
[31] S. Keshav, *An engineering approach to computer networks*, Addison wesley, 1997.
[32] D. Awduche, J. Malcolm et al., *Requirements of traffic engineering over MPLS*, RFC 2702, Sep. 1999.
[33] R. Guerin, S. Blake and S. Herzog, *Aggregate RSVP-based QoS request*, Internet Draft, 1997.
[34] G. Lu, Issues and technologies for supporting multimedia communication over Internet', *Computer communications* **23** (2000), 1323–1335.
[35] Kouhei Fujimaoto, Shingo Ata and Masayuki Murata, Statistical analysis of packet delays in the Internet and its application to playout control for streaming applications, *IEICE Trans. commn.* **E84-B** (June, 2001), 1504–1512.