

Multimedia Group Synchronization Protocols for Integrated Services Networks

Ian F. Akyildiz, *Senior Member, IEEE* and Wei Yen, *Student Member, IEEE*

Abstract—Unlike traditional data traffic, real time multimedia traffic requires synchronization. Temporal relationships among media must be maintained. Yet delay jitter and the absence of a global clock may disrupt these temporal relationships. This paper introduces new group synchronization protocols for real time, multimedia applications, including teleconference, teleorchestration and multimedia on demand services. The proposed protocols achieve synchronization for all configurations (one-to-one, one-to-many, many-to-one, and many-to-many), and does so without prior knowledge of the end-to-end delay distribution, or the distribution of the clock drift. The only *a-priori* knowledge the protocols require is an upper bound on the end-to-end delay. The paper concludes with simulation experiments showing that the protocols work effectively in both LAN and WAN environments.

Index Terms—High speed networks, multimedia services, synchronization, asynchrony, delay jitter, local clock drift, initial collection time, initial playback times, performance evaluation.

I. INTRODUCTION

REAL TIME¹ multimedia applications present a critical problem to integrated services networks, i.e., synchronization that ensures the temporal related media to be presented in a proper temporal relation [8]. Such applications require both *intramedia* and *intermedia* synchronization. *Intramedia* synchronization defines the timing relationship of a single medium over a single connection. *Intermedia* synchronization defines those relationships for multiple connections, or for multiple media interleaved in a single connection.

Multimedia applications require synchronization in four configurations. The first configuration, known as *unicast*, is a one-to-one relation. In the unicast configuration, one user transmits temporally related media units and another user receives the media units and plays them out via different display devices. For example, visual telephony has the unicast configuration. The second configuration, *multicast*, is a one-to-many relation. One sender multicasts temporally related media units to many receivers. For example, some teleconferencing applications such as tele-lectures have the multicast configuration. The third case is the *retrieval* configuration that is a many-to-one relation. In this configuration, a receiver retrieves media from different senders. For example, a user may get video and voice from two different databases. Finally, there is

the *group* configuration that is a many-to-many relation. Each user in a communication group can be a sender, receiver, or both. Teleconferencing exemplifies the group configuration. Note that the unicast, multicast, and retrieval configurations are special cases of the group configuration.

In all four configurations, the following four sources of asynchrony can disrupt synchronization:

- a) *Delay Jitter* is the variation of end-to-end delay that consists of three parts [8]. First is the collection delay; the time needed for the sender to collect media units and prepare them for transmission. Collection may occur directly from media recorders such as camera, databases, or both. The second part is the network delay from the network boundary at the sender to the boundary at the receiver. Finally, delivery delay adds to the end-to-end delay. Delivery delay is the time the receiver needs to process the media units and prepare them for playback. Note that none of these delay components are necessarily constant. For example, the network delay in ATM networks varies because of different queueing delays caused by the unpredictable burstiness in the network and by the variable transmission bit rates. The collection delay and the delivery delay also vary from time to time due to different processing time (coding/decoding, segmentation/assembly etc.) of media units.

To see the effect of delay jitter, consider several temporally related media units sent from a sender to a receiver via different connections. These media units must be played back with appropriate temporal relationships. The presence of delay jitter, however, may destroy the temporal relationships that exist at the sender. When the receiver is ready to play a media unit, that unit may not be available yet because it experienced greater end-to-end delay than preceding units. In such a case, a playback *discontinuity* occurs at the receiver. Delay jitter becomes so significant in WAN environments that playback discontinuity is perceptible to humans.²

- b) *Local Clock Drift* arises when clocks at users run at different rates. Without a synchronization mechanism, the asynchrony gradually will become more and more serious. If the playback rate of the receiver is faster than the collection rate of the sender, the receiver may suffer *starvation*. As with delay jitter, playback discontinuity may result. Alternately, the receiver can become *flooded*

Manuscript received June 1, 1994; revised April 1, 1995.

The authors are with the Broadband and Wireless Networking Laboratory in the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA.

Publisher Item Identifier S 0733-8716(96)00231-4.

¹End-to-end delay must be less than a few hundred milliseconds for real time applications.

²End-to-end delay is also an important factor in WAN environments.

with media units if its playback rate is slower than the collection rate of the sender. Both of these problems can be considered to be aspects of the *asynchrony*.

- c) *Different Initial Collection Times*. In both of the retrieval and the group configurations, there are more than one sender in the communications. These senders must collect and transmit synchronously, otherwise, the temporal relationships among media units might be destroyed. For example, consider two media sources, one providing voice and the other video. If they start to collect their media units at different times, the playback of the media units of voice and video from two sources at the destination loses semantic meaning. Media units with no temporal correlation are replayed simultaneously resulting in the so-called "lip-sync" problem. Note that the different end-to-end delays and clock drift can also cause the "lip-sync" problem.
- d) *Different Initial Playback Times*. As receivers, users in a group must start to play temporally related media units simultaneously, so that each user initially perceives media units synchronously. If the initial playback times are different for each user, then asynchrony will arise. For some multicast applications in which fairness is the major concern, the playback times of media units of all receivers should be the same, otherwise, the earlier a receiver gets media units, the earlier he can react.

Note that only delay jitter and local clock drift arise in the unicast configuration.³ Multicast configurations must also deal with different initial playback times, while retrieval configurations may experience different initial collection times in addition to delay jitter and local clock drift. All four sources of asynchrony occur in the group configuration.

In addition to the classification by configuration, intermedia synchronization may be classified based on application. Such a classification results in *Lip* synchronization and *Synthetic* synchronization [14], [19]. Lip synchronization applications produce, transmit, and play media units in real time. Synthetic synchronization applications synchronize media units retrieved from databases. Typically, the temporal relationships among media units are static in lip-sync applications and are variable in synthetic-sync ones. For some synthetic-sync applications, data can be retrieved well ahead of their playout time so that the system can have the flexibility to schedule the communication task. Teleconferencing and on-line multimedia inquiry services are two examples of lip and synthetic synchronizations, respectively. All four sources of asynchrony affect both lip and synthetic synchronization applications.⁴

II. RELATED WORK

Recently there has been a growing interest in the development of synchronization protocols to solve asynchrony

³In some cases, the different initial collection time problem may also occur in unicast configurations. For example, a source sends a combined audio and video stream generated by two applications to a destination.

⁴In addition, there is another type of synchronization referred as event synchronization. Event synchronization deals with the unanticipated human interaction and signals from the underlying transmission system.

problems. Several papers have been published dealing with various types of asynchrony problems.⁵

The adaptive feedback protocol [17] solves the asynchrony in multimedia on-demand services, i.e., $1 - n$ configuration. It requires an additional connection for each sender and receiver pair to transmit feedback units. In the model [17], a multimedia server transmits media units to several mediaphones that are playback devices such as, audiophones and videophones. Among the mediaphones, one is assigned to be the master mediaphone and others are slave mediaphones. The protocol synchronizes slaves to the master by collecting and comparing the feedback units at a multimedia server. The multimedia server stores media units and provides them to mediaphones as they request them. The multimedia server estimates the asynchrony and instructs mediaphones to skip or pause media units accordingly. Adaptive feedback performs well in LAN environments but is not suitable for WAN's because its synchronization guarantees are limited by the maximum network delay.

Pre-compiled scheduling protocol [18], [19] is designed for synthetic-sync applications. They assume media units are retrieved well ahead of transmission so that they can use an *Object Composition Petri Net* (OCPN) to capture the temporal relationships among media units. According to end-to-end delays and the OCPN, they calculate the playout and transmission schedules for receivers and senders, respectively. Senders and receivers will then follow the computed schedules to transmit and play the media units. As long as the network itself is synchronized, and the computed schedules are followed precisely, pre-compiled scheduling solves delay jitter, initial collection times and initial playback times. This method [18], [19] adds nearly all overhead to the beginning of the communication. It requires almost no overhead when the actual transmission starts. The critical problem with this approach is that the schedules cannot be calculated in advance if the temporal relationships among media are not known or predictable. In addition, the scheme does not consider the clock drift problem.

Although not specifically devised for multimedia synchronization, delay jitter control [7], [16] can be used to correct delay jitter problems. It distributes the synchronization responsibility among intermediate nodes between senders and receivers. The intermediate nodes that perform synchronization, however, are unlikely to be work-conserving. The resulting average network delay will become longer while the variance becomes smaller.

The flow synchronization protocol [8] can, in some cases, correct delay jitter, initial collection times, and initial playback times. The protocol assumes that a clock synchronization scheme exists to handle the local clock drift problem. The protocol lets users in a synchronization group exchange flow information periodically so that the common synchronization delay for the group can be calculated.⁶ The main problem of this protocol is the assumption of the underlying clock

⁵A study on the sensitivity of human perception for various multimedia applications has been done in [15].

⁶Users are identical to processors in [8].

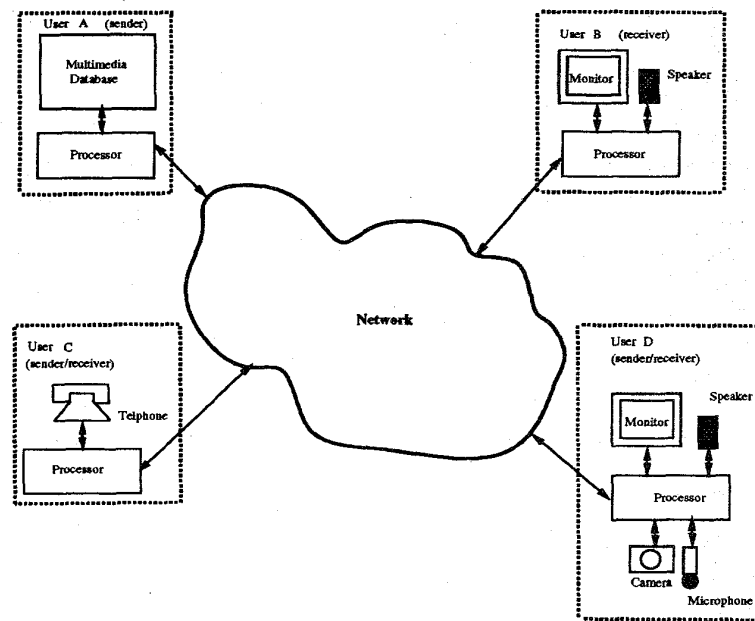


Fig. 1. User model.

synchronization scheme. In contrast, our protocol corrects local clock drift explicitly.

A real-time *Stream Synchronization Protocol* (SSP) aiming to provide multimedia news services was proposed in [10]. This protocol, first, determines an upper bound for asynchrony tolerance among media streams according to the quality of service requirements of applications. In order to synchronize media streams, this protocol adds intentional delay to those media streams experiencing less end-to-end delay without violating the upper bound of asynchrony tolerance. User interactions, including pausing, skipping, and scanning forward or backward, are also addressed in this paper. The local clock drift problem, however, is not explicitly mentioned in this paper, nor is the different initial playback times problem.

In addition to these five protocols, a media mixing algorithm was proposed in [11] to support multimedia conferences. The algorithm performs hierarchical mixing in a set of mixers, the root mixer multicasts the mixed media streams to all participants. This algorithm corrects delay jitter and local clock drift, but it is not clear how a multimedia conference is initiated synchronously. Media mixing leaves the problems of different initial collection times and different initial playback times unresolved. Furthermore, mixers add the additional end-to-end delay. The additional delay may be particularly critical in WAN environments. Furthermore, Steinmetz and Engler conducted experiments to explore the sensitivities of human perceptions to asynchrony in multimedia applications [15]. Their results define synchronization requirements for different types of multimedia applications.

In this paper, we develop synchronization protocols proposed in Section III to eliminate the four sources of asynchrony for the unicast, multicast, retrieval and group multimedia configurations in real time broadband integrated services net-

works. We assume that there exists an algorithm that maps the temporal relation among media into several synchronous flows. Our protocols should be placed at orchestration/synchronization layer to provide synchronization services to application layer [1], [9]. The application layer, however, needs to inform our protocols of the description of multimedia objects including the synchronous flows and their collection and playback periods. In Section IV, we demonstrate that our protocols perform equally well in LAN's and WANs without a global clock.⁷ In Section V we conclude the paper.

III. A PROTOCOL FOR GROUP SYNCHRONIZATION

A group in the network consists of a set of users of an application and of a set of connections among them. Moreover, each user has its own processor and may play roles of a sender, receiver, or both as Fig. 1 shows. A sender collects media units from multimedia sources such as media recorders or databases while a receiver plays out media units via media display devices such as a speaker or terminal. The users in the same group need to collect and playback media units synchronously. To better explain the configuration of group synchronization and its synchronization requirements, we demonstrate three interesting applications: teleconference, as shown in Fig. 2, teleorchestra, as shown in Fig. 3, and on-demand multimedia services, as shown in Fig. 4.

In Fig. 2, a set of users at different sites attempt to have a virtual meeting via an integrated services network. Each user transmits his own media units while receiving media units from the other participants. Media units collected by users in the conference at approximately the same time should be

⁷In some cases, LAN delay can be greater than WAN delay. In this paper, however, we assume end-to-end delay is less than 20 ms in LAN's and greater than 100 ms in WAN's.

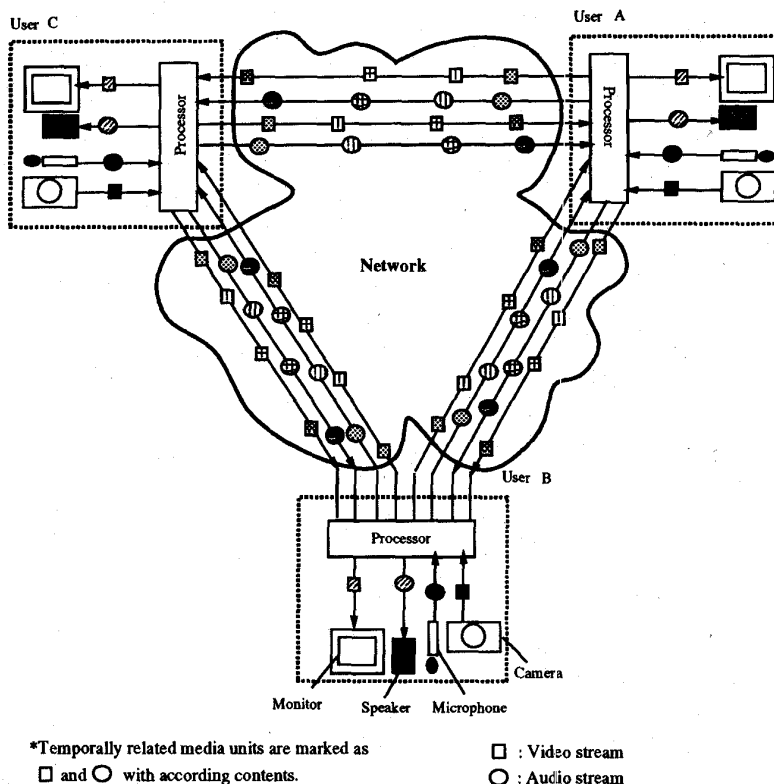


Fig. 2. Teleconference.

played by all users at the same time. Thus, all participants of the teleconference are in one group. For a teleorchestra, a conductor and a group of musicians at distributed locations play a symphony and broadcast the music to some audience in real time. Media units collected from the conductor must be played out to all musicians simultaneously; the playback time of the media units from the conductor must be synchronized at each musician. Failure to meet the requirement would cause the teleorchestra to become discordant and cacophonous. Note that the connections between musicians A and B are optional. Musician A can play along with the media units from the conductor and musician B, or musician B can play along with the media units from the conductor and musician A, if we can equalize the associated delay. If it is not possible to equalize the delay, then both musicians just play along with the media units from the conductor and there is no connection between musicians A and B. However, the playback times of the temporally related media units at each audience need not be the same because fairness is a concern in this case.

In Fig. 4, distributed multimedia databases provide on-demand multimedia services to an audience. The distributed databases playing the roles of senders need to collect media units synchronously. Although each audience needs to play out the media units from the databases synchronously, it is not necessary to synchronize the playback time of media units among the audiences, even when they order the same program. Thus, each audience playing the role of receiver belongs to a different group while the sending databases

belong to all groups. Note that Figs. 2 and 3 represent the group configuration while Fig. 4 demonstrates the retrieval configuration.

A. Compensating the Delay Jitter Effect

The asynchrony caused by delay jitter between sender i and receiver j can be solved by pre-fetched buffering with buffer size B_j given by the following formula [17]:

$$B_j = \left\lceil \frac{J_{i,j}}{\mu_j} \right\rceil \quad (1)$$

where $J_{i,j}$ is the delay jitter between i and j and is computed by

$$J_{i,j} = \Delta_{i,j} - \delta_{i,j} \quad (2)$$

with $\Delta_{i,j}$ as the maximum end-to-end delay between i and j ; $\delta_{i,j}$ is the minimum end-to-end delay between i and j ; μ_j is the playback period of a medium at receiver j , and $\lceil x \rceil$ is the smallest integer greater than x . Note that B_j is measured in terms of the media unit. The size of the media unit may be constant or various depending on the characteristics of the medium and the encoding scheme. Thus, B_j media units with the maximum size must be reserved at receiver j to prevent the asynchrony caused by delay jitter.

If $J_{i,j}$ can be kept small, then the required pre-fetched buffer size B_j will decrease as can be seen in (1). Since the values of $\Delta_{i,j}$, $\delta_{i,j}$, and μ_j can be determined in the negotiation

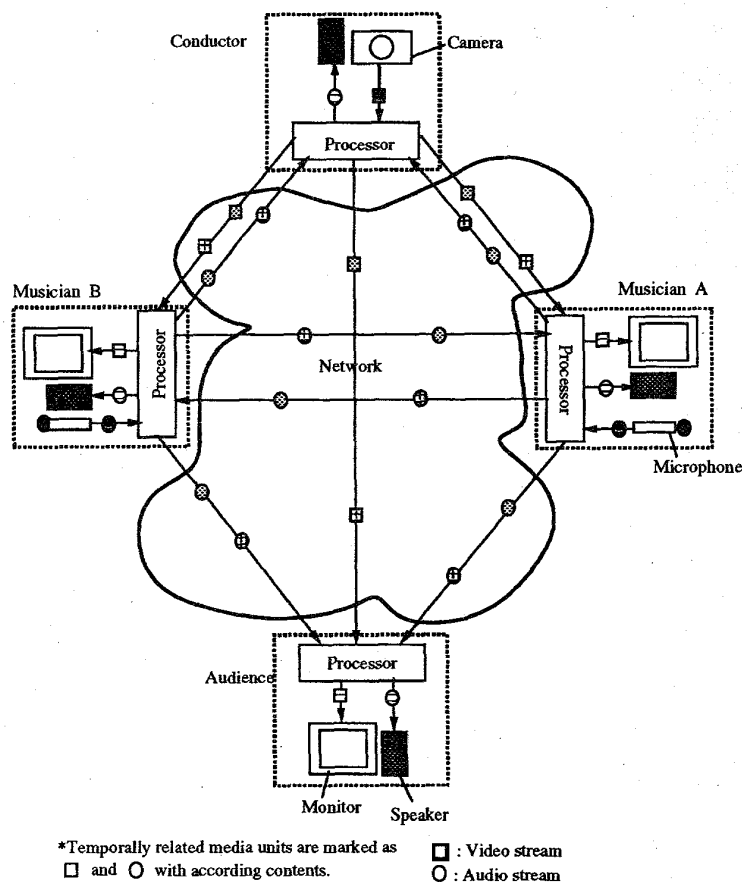


Fig. 3. Teleorchestra.

phase for multimedia communication between users i and j , the delay jitter effect can be solved by selecting an appropriate size of pre-fetched buffer at the receiver as given in (1).

B. Dynamic Period Adjustment Protocol for Local Clock Drift Problem

To compensate for local clock drift, our scheme synchronizes the clocks of all users in the same group according to a time reference [13] that we call *virtual global time* (VGT). The value of VGT is equal to the value of the local clock of the *chairman* elected by the distributed election algorithm presented below. The chairman distributes the time information, i.e., the value of the VGT, based on its local clock to all other users in the group (Section III.C). This time information is then used to synchronize the local clocks of all users to the VGT. First, we present the algorithm for the election of the chairman. Note that the algorithm is only required for the group and retrieval configurations because the sender is selected as chairman in the unicast and multicast configurations.

1) *Distributed Election Algorithm*: In the negotiation phase, each user of a group generates a one-byte random number between zero and $2^8 - 1$. The user broadcasts the random number with its own ID to all other users in the

group.⁸ After receiving all random numbers, each user then sorts these random numbers in a descending order. The user generating the largest random number is chairman. If more than one user generates the same largest number, then the algorithm selects the user with the largest ID. Once the chairman is selected, it creates a queue that contains all users in the group. The order of the users in the queue represents their priority to become the next chairman. The value of the local clock of the first user in the queue (the chairman) is used as the VGT. This queue is broadcast by the chairman to all other users in the group. If a user wants to leave the group, it is removed from the queue by all other users. If a user wants to enter the group, it is added to the end of the queue by all other users. Note that another election algorithm for a distributed clock synchronization system is given in [12]; where the existence of a master clock is assumed and the algorithm simply elects the next master clock when the original one fails.⁹

2) *Dynamic Period Adjustment Protocol*: Once the group has elected a chairman, a mechanism must synchronize the playback period at user j to the collection period at user i , given i is the chairman. We denote the initial playback period

⁸We assume that each user has a unique ID.

⁹Master clock and chairman are equivalent in [12].

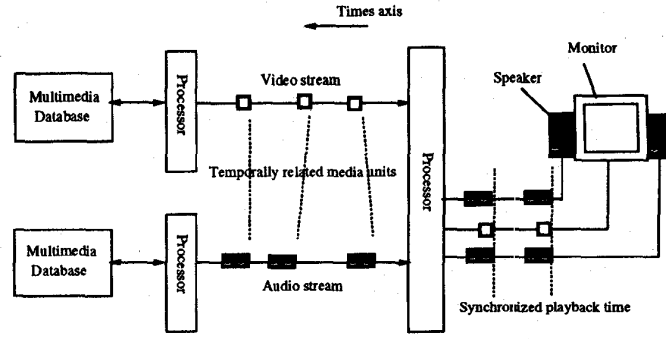


Fig. 4. Multimedia on-demand services.

decided by i and j in the negotiation phase by $\mu_j^{(init)}$ and the adjusted playback period we want to achieve by μ_j . Note that the initial playback period $\mu_j^{(init)}$ at j and the collection period η_i at i have the same value. User j needs to obtain the time information of user i so that he can use it to adjust its playback period μ_j .

We stamp the first packet of each media unit k from i to j with the timestamp $\phi_{i,j}(k)$ at which i begins to collect the media unit. The delaystamp $\psi_{i,j}(k)$ is the sum of all times spent by this first packet in the nodes (switches) of the network plus the collection delay at i and the delivery delay at j [21]. Since the first packet of each media unit contains the timestamp and delaystamp, it is necessary to protect the first packet from corruption. In other words, for the first packet of each media unit, user j either receives it correctly or loses it completely. The k th media unit sent from user i is available to be played by j at time $S_{i,j}(k)$ that is

$$S_{i,j}(k) = \phi_{i,j}(k) + \psi_{i,j}(k) + \theta_{i,j} \quad (3)$$

where $\theta_{i,j}$ is the propagation delay between users i and j .

Let

$$\sigma_i(k) = S_{i,j}(k+1) - S_{i,j}(k) \quad (4)$$

represent the value of the interarrival time between media units k and $k+1$ at user j referring to local clock of i .

Substituting (3) into (4) we obtain

$$\sigma_i(k) = \phi_{i,j}(k+1) - \phi_{i,j}(k) + \psi_{i,j}(k+1) - \psi_{i,j}(k). \quad (5)$$

Ideally, the *intercollection* time $[\phi_{i,j}(k+1) - \phi_{i,j}(k)]$ in (5) is constant for all $k = 1, 2, \dots$ media units and is equal to the collection period η_i at user i , i.e.,

$$\eta_i = \phi_{i,j}(k+1) - \phi_{i,j}(k). \quad (6)$$

In this case (5) can be rewritten as

$$\sigma_i(k) = \eta_i + \psi_{i,j}(k+1) - \psi_{i,j}(k). \quad (7)$$

If the *intercollection* time is not constant, then we can use (5) to estimate $\sigma_i(k)$.

In order for user j to find out the local clock drift between i and himself, our protocol requires a timer at user j . Let $\nu_j(k)$ be the value of this timer that counts the interarrival times between two consecutive media units, say k and $k+1$, using the local clock of j .

Now we want to adjust the negotiated playback period $\mu_j^{(init)}$ at j to μ_j so that the time user i spends from zero to the collection period η_i is the same as the time user j spends from zero to the adjusted playback period μ_j . Therefore, the ratio of the interarrival time $\nu_j(k)$ referring to user j 's clock and the interarrival time $\sigma_i(k)$ referring to user i 's clock is equal to the ratio of the adjusted playback period μ_j at user j and the collection period η_i at user i

$$\frac{\nu_j(k)}{\sigma_i(k)} = \frac{\mu_j}{\eta_i}. \quad (8)$$

By substituting (7) into (8), we obtain

$$\mu_j = \frac{\eta_i \nu_j(k)}{\eta_i + \psi_{i,j}(k+1) - \psi_{i,j}(k)}. \quad (9)$$

User j plays media units according to the adjusted playback period μ_j obtained from (9) referring to its own local clock. Note that (9) assumes that the intercollection time is constant. Equation (9), however, can easily be modified in the case of variable intercollection time using (5).

Let $A_{i,j}(k)$ define the dynamic period adjustment factor as

$$A_{i,j}(k) = \frac{\mu_j - \mu_j^{(init)}}{\mu_j^{(init)}} \quad (10)$$

where $\mu_j^{(init)}$ is the negotiated playback period at user j and μ_j is the adjusted playback period, (9). $A_{i,j}(k)$ measures the drift between the local clocks of users i and j .

By substituting (9) into (10) and using the fact that the value of $\mu_j^{(init)}$ is equal to η_i , then $A_{i,j}(k)$ can be computed from

$$A_{i,j}(k) = \frac{\nu_j(k)}{\eta_i + \psi_{i,j}(k+1) - \psi_{i,j}(k)} - 1. \quad (11)$$

The local clock of each user in the group can be synchronized to the chairman's clock by periodically applying the dynamic period adjustment protocol to all users pair (*chairman*, j) where j is any user in the group except the chairman. Intermediate nodes along the connections can apply this protocol as well, so that their clocks are synchronous to that of the chairman. Hence, the delaystamps computed by these nodes are measured according to the chairman's clock. The period of adjusting μ_j depends on the synchronization requirements of applications and the level of clock

drift. The performance of the dynamic period synchronization mechanism with different adjustment periods will be shown in Section IV.

3) Remarks:

- It is possible that some media units are corrupted during the transmission. According to different applications and compressing schemes, re-transmission may or may not be necessary to compensate the corrupted media units. For example, the compressed video stream may require re-transmission when a corrupted video frame is received because the errors in the corrupted frame can propagate over the next several frames. On the other hand, the uncompressed video stream may not require re-transmission because the errors do not propagate. Regardless of whether the retransmission is required or not, it is important to assure that the dynamic period adjustment protocol works appropriately in both cases.

In fact, the following solution could be used for synchronization. Suppose user j receives the first packet of the k th media unit, then finds that the media unit is corrupted. In this case, j is still able to calculate $A_{i,j}(k)$ by using (11) because it receives the first packet of the k th media unit. In other words, j can synchronize its local clock to the VGT provided by user i . Furthermore, j can play, in some cases, the $(k-1)$ th media unit at the playback time of the k th media unit. Hence, no serious problem is caused by the corrupted media unit. If user j does not receive the first packet of the k th media unit, then the previously computed $A_{i,j}(k-1)$ is used to synchronize the clock between i and j . Also, the $(k-1)$ th media unit is used to replace the corrupted k th media unit. However, the timer at j keeps ticking until it receives the first packet of $(k+1)$ th media unit and $v_j(k)$ represents interarrival time between the $(k-1)$ th and $(k+1)$ th media units, then $A_{i,j}(k)$ can be calculated by slightly modifying (11), i.e., replacing $\psi_{i,j}(k+1)$ by $\psi_{i,j}(k+2)$. The dynamic period adjustment protocol updates $A_{i,j}(k)$ only when user j receives the first packet of a media unit.

- The dynamic period adjustment protocol presented above sends the time information implicitly instead of explicitly as it is the case of other clock synchronization protocols such as network time protocol (NTP) [4]. Our protocol can reduce the network messages without adding intolerant complexity to the nodes in the network.
- It is not necessary to perform a per-packet delaystamp in our protocol. The delaystamp is used only when the dynamic period adjustment protocol is performed and, as shown in Section IV, one period synchronization action for every 50 media units is good enough to keep the asynchrony within a few milliseconds. In other words, the complexity required to compute the delaystamp at each node is minor. Moreover, if the nodes provide the delay jitter control, then the delaystamp is computed automatically.
- To accurately measure the delaystamp $\psi_{i,j}$, the operating system at each node must be able to support a clock accuracy in millisecond level or better. Note that traditional multiprogramming operating systems (such

as the UNIX BSD 4.3) do not guarantee the required accuracy, mainly due to the fact that system calls are non-preemptable. Solution approaches include modifications to the operating system kernel by adding preemption points to the source code of operating system calls [20], or use real time operating systems [6], [5].

C. Synchronization Protocol for Initial Collection Time

In some applications, such as teleconference, it is important that users start to play and collect at the same time to maintain the temporal relationships among media units, i.e., the semantic meaning of communication. In this subsection, we present the mechanism to synchronize the initial collection time of all users in a group. The combination of the initial collection time synchronization mechanism and the period synchronization mechanism introduced in Section III.B assures that users provide media units in a synchronous manner.

Initial collection time synchronization is either explicitly stated or implicitly assumed in some of the related works [8], [11], [19]. However, the methods devised to guarantee initial collection time synchronization require some sort of initiator who sends the initialization message to users in the group [8], [19]. The initiator may or may not be one of the users in the synchronization group. The initiator will not send the initialization message until all users in the group are ready, so users in the group need to inform the initiator when they are ready to collect the media units. Those methods delay the global initial collection time of the group because of the message exchange among the users and the initiator. In our approach the initial sending time is computed and decided in a distributed manner and it is the earliest time that users can initiate their collection of media units.

Our method requires the recovery of VGT at all users in the group. If the propagation delay $\theta_{i,j}$ between the chairman and every user in the group is known, then each user can compute the VGT value by using the (3). In the following, we explain how to measure the propagation delay. Note that the propagation delay is the time spent by a packet at the intermediate links along the connection while the delaystamp is the sum of collection delay, delivery delay and queuing delay, the time a packet spent at intermediate nodes along the connection.

In the negotiation phase, user i transmits a packet α to user j with the timestamp $\phi_{i,j}(\alpha)$. The delaystamp $\psi_i(\alpha)$ is the total time spent by the packet in the network plus the collection and delivery delays as we described in Section III.B. After receiving the packet α , user j adds the time that α spends there to the delaystamp $\psi_{i,j}(\alpha)$ and sends α back to user i . So, the packet α comes back to user i at time $T_{i,j}(\alpha)$ which is

$$T_{i,j}(\alpha) = \phi_{i,j}(\alpha) + \psi_{i,j}(\alpha) + 2\theta_{i,j} \quad (12)$$

where $\psi_{i,j}(\alpha)$ is the accumulated delaystamp for packet α and $2\theta_{i,j}$ is the round-trip propagation delay between i and j . Since $T_{i,j}(\alpha)$, $\phi_{i,j}(\alpha)$, and $\psi_{i,j}(\alpha)$ are known, the propagation delay $\theta_{i,j}$ of the packet α for the connection can be obtained by rewriting (12).

$$\theta_{i,j} = 0.5[T_{i,j}(\alpha) - \phi_{i,j}(\alpha) - \psi_{i,j}(\alpha)]. \quad (13)$$

In the negotiation phase, each user measures the propagation delay $\theta_{i,j}$ between itself and all other users in the group. Once the chairman is appointed, user i can use the estimated propagation delays $\theta_{ch,i}$ between itself and the chairman to compute the value of VGT using (3). According to (3), the estimated propagation delay $\theta_{ch,i}$ and delaystamp $\psi_{ch,i}$ affect the accuracy of computed VGT. If intermediate nodes provide precise measurement on $\theta_{ch,i}$ and $\psi_{ch,i}$, then we can expect the accuracy is within millisecond range. The users (excluding the chairman) increase the VGT with the speed of their local clocks and confirm the VGT values with the time information carried implicitly in some packets sent from the chairman. After the chairman starts to distribute VGT, each user i can inform all other users in the group of its initial collection time I_i in terms of VGT.

Note that the initial collection time I_i of user i must be bounded as

$$I_i \geq \max_j \{\Delta_{i,j}\} + \tau_i \quad (14)$$

where $\max_j \{\Delta_{i,j}\}$, is the largest among the maximum end-to-end delays of the user pairs (i,j) in the network and τ_i is the time in terms of VGT at which user i broadcasts its initial collection time I_i to all other users. If I_i does not satisfy the inequality (14), then there exists a user j with the maximum end-to-end delay $\Delta_{i,j}$ such that $\Delta_{i,j} + \tau_i > I_i$. Namely, it is possible that j is still waiting for I_i at the moment other users start to collect media units.

After each user receives the initial collection times I_i for all users i in the group, the largest initial collection time is picked by users in the group as the global initial collection time G_c .

$$G_c = \max_i \{I_i\} \quad (15)$$

where I_i is obtained from (14). Each user begins to collect his media units at G_c .

D. Synchronization Protocol for Initial Playback Time

Once the initial playback times are synchronized given by the protocol in the previous section, the dynamic period synchronization mechanism described in Section III.B will adjust the collection/playback periods according to the VGT. The combination of initial playback time synchronization protocol and the period synchronization protocol assures that users play media units in a synchronous manner.

In this subsection, we show the derivation of the global initial playback time G_p . Without loss of generality, we assume that the sending times of a media unit from user i to all other users in the group are different. Fig. 5 shows the sending times of a media unit to n users. In the negotiation phase of the connections, the maximum end-to-end delay $\Delta_{i,j}$ is decided by the QOS requirements of applications and available resources [1], [3].

By assuming the maximum collection and delivery delay are known, we can determine the maximum end-to-end delay $\Delta_{i,j}$. The minimum end-to-end delay $\delta_{i,j}$ is equivalent to the propagation delay $\theta_{i,j}$ between users i and j plus the minimum collection delay and minimum delivery delay. We can estimate

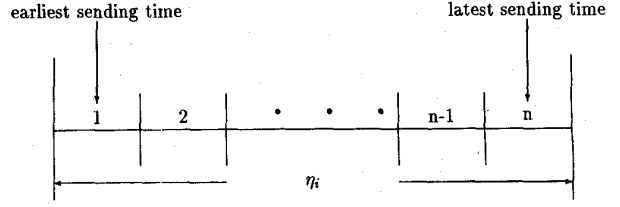


Fig. 5. Possible sending time for a media unit.

the propagation delay $\theta_{i,j}$ estimated in the negotiation phase by the mechanism we introduced in Section III.C.

With the knowledge of the two parameters $\delta_{i,j}$ and $\Delta_{i,j}$, we can calculate the pre-fetched buffer sizes $B_j(m)$ for any medium m of user j using (1). For the connections between i and j , the latest time $L_{i,j}$ that j could start to playback media units is equal to

$$L_{i,j} = I_i + \max_m \{B_j(m) \eta_i(m) + \Delta_{i,j}(m)\} \quad (16)$$

where I_i is the initial collection time of the media units from i ; $B_j(m)$ is the pre-fetched buffer size of medium m at receiver j ; $\eta_i(m)$ is the collection period of a medium m of user i ; $\Delta_{i,j}(m)$ is the maximum end-to-end delay between users i and j for the medium m .

In (16), we see that $L_{i,j}$ is the latest time that the $B_j(m)$ th media unit is ready to be played out at user j for each medium m sent from user i . It is obvious that receiver j gets at least $B_j(m)$ media units for all media by the time $L_{i,j}$. If receiver j starts to playback media units before $L_{i,j}$, then discontinuity may occur due to delay jitter and insufficient pre-fetched buffer size. On the other hand, if j starts to play media units from i after the latest playback time $L_{i,j}$, then j must buffer more media units than it needs, a waste of network resources.

User i can compute the latest playback time $L_{i,j}$ for any user j in the communication group from (16). However, if we want to synchronize the initial playback times M_i of media units from user i at all other users in the group, then we must select a time instant based on VGT. We can select the synchronous initial playback time of the users as

$$M_i = \max_j \{L_{i,j}\} \quad (17)$$

where M_i represents the maximum $L_{i,j}$ among the connections associated with user i . Assume that user i is aware of the initial collection time I_i in the negotiation phase, it can calculate the latest initial playback time $L_{i,j}$, (16), for user j in the communication group. Then, i can get M_i from (17) and distribute it to every user in the group.

We can compute M_i for each user i in the group. In order to get the global initial playback time, however, the largest M_i will be selected by all users in the group. The initial collection time of each user i in the group can be synchronized by using (15). Note that each user in the group plays both roles of a sender and a receiver. From the senders' point of view, M_i can be obtained from the global initial collection time G_c , (15). Thus, each user i in the group distributes its playback waiting period λ_i in terms of VGT to all users in the group where

$$\lambda_i = M_i - I_i. \quad (18)$$

In the playback waiting period λ_i , all users fill their pre-fetched buffer with media units from user i and wait for the synchronous playback of the media units. After receiving the playback waiting period λ_i from all users, each user in the group will pick up the largest playback waiting period $\max_i\{\lambda_i\}$. The global initial playback time G_p is then computed from

$$G_p = G_c + \max_i\{\lambda_i\} \quad (19)$$

where G_c is the global initial collection time obtained from (15), and $[\max_i\{\lambda_i\}]$ is the largest playback waiting period.

For each user j in the group, a timer ω_j is set up with the initial value given by (20). The timer ω_j starts to count down when the first media unit from the chairman is received.

$$\omega_j = G_p - \rho_j \quad (20)$$

where ρ_j is the time in terms of VGT that the first media unit from the *chairman* is ready to be played at j .

The timer keeps decreasing and corrects its value according to the dynamic period adjustment factor $A_{i,j}(k)$, given i is the chairman. When the timer ω_j reaches zero, user j starts to playback. Once the receivers begin playback, only the dynamic period adjustment protocol of Section III.B is involved.

E. Negotiation Process for Synchronization

In Fig. 6, we show the negotiation process for synchronization between any two users i and j in the same communication group. First, user i runs the chairman election algorithm and sends the packet α to user j with timestamp $\phi_{i,j}(\alpha)$ in order to estimate the propagation delay $\theta_{i,j}$. After receiving the random numbers generated by all users in the group, user j knows the identity of the chairman and synchronizes its timer $\nu_j(k)$ to the local clock of the chairman. User j also adds the time that the packet α spends there to the delaystamp $\psi_{i,j}(\alpha)$ and sends α back to user i who then determines the propagation delay $\theta_{i,j}$ between himself and j by using (13). Assuming the maximum and minimum collection and delivery times are known, negotiation of the minimum end-to-end delay $\delta_{i,j}$ and the maximum end-to-end delay $\Delta_{i,j}$ can be achieved by the users and the network. User i can compute the pre-fetched buffer size B_j and the playback waiting period λ_i from (1) and (18), respectively. User i sends these values with the initial collection time I_i , (14), to user j . After receiving B_j , λ_i , and I_i from all users in the group, user j can decide on the global initial collection time G_c , (14), and global initial playback time G_p , (15), and start to collect or playback media units accordingly.

From Fig. 6 it is easy to see that the time κ required to initiate the negotiation process for the synchronization is greater than three times the longest end-to-end delay, i.e.,

$$\kappa \geq 3 \max_{i,j} \{\Delta_{i,j}\} \quad (21)$$

where $\max_{i,j} \{\Delta_{i,j}\}$ is the longest end-to-end delay among all connections in the communication group; the factor three is the number of messages sent either by user i or j in the negotiation phase.

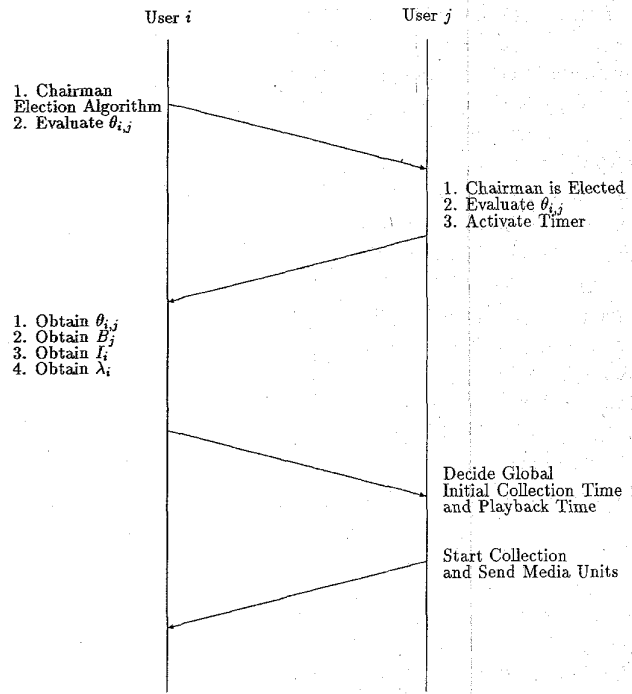


Fig. 6 Negotiation process.

Example: Let us assume that there are three users, say A , B , and C , who would like to have a teleconferencing. The negotiation process performs as follows:

- 1) User A sends a packet containing a random number, timestamp, and delaystamp to users B and C , respectively. Users B and C send this same information to users $\{A, C\}$ and users $\{A, B\}$, respectively.
- 2) Upon receiving the packet from other users, user A :
 - a) Records the arrival time of the packet.
 - b) Records the timestamp, $\phi_{B,A}(k)$ or $\phi_{C,A}(k)$, and delaystamp, $\psi_{B,A}(k)$ or $\psi_{C,A}(k)$, of the packet.
 - c) Adds the time that the packet spent at A to the delaystamp in the packet.
 - d) Sends the packet back to its original sender.
Users B and C follow the same procedures (a-d) upon receiving packets from other users.
- 3) After receiving all packets from other users, users A , B , and C can determine the chairman according to the random numbers received. Let us assume that user C is selected as the chairman. Users A and B will start their timers and assign the arrival time of the packet from C recorded in step (2.a) as the initial value of their timers after determining C is the chairman.
- 4) Upon receiving the packets originally from themselves, users A , B , and C , can calculate the propagation delays according to (13). For example, user A can calculate $\theta_{A,B}$ and $\theta_{A,C}$ after receiving the packets sent back from B and C in step (2.d). In the same manner, users B and C can calculate $\theta_{B,A}$, $\theta_{B,C}$ and $\theta_{A,C}$, $\theta_{B,C}$, respectively.

- 5) Assume that user A knows the maximum end-to-end delay $\Delta_{A,B}$ and $\Delta_{A,C}$, user B knows $\Delta_{A,B}$ and $\Delta_{B,C}$, and user C knows $\Delta_{A,C}$ and $\Delta_{B,C}$, then they can calculate the associated buffer sizes by using (1). (The propagation delay found in step 4 can be used as the minimum end-to-end delay.)
- 6) With the maximum end-to-end delay and associated buffer size, users A , B , and C can calculate I_i and λ_i , where $i = A, B, C$, respectively by using (14), (16)–(18).
- 7) User A sends I_A and λ_A to users B and C . User B sends I_B and λ_B to users A and C . User C sends I_C and λ_C to users A and B with timestamp, $\phi_{C,A}(k+1)$ or $\phi_{C,B}(k+1)$, and delaystamp, $\psi_{C,A}(k+1)$ or $\psi_{C,B}(k+1)$, that will be used A and B to synchronize their clock to C 's clock.
- 8) Upon receiving all I_i and λ_i where $i = A, B, C$, users A, B , and C can determine initial collection time G_c and initial playback time G_p by (15), (19), respectively, and start to collect and play media units accordingly.
- 9) Users A and B will synchronize their own local clock to C 's clock by using the information of their timers ($\nu_A(k)$ or $\nu_B(k)$), the timestamp ($\phi_{C,A}(k+1)$ or $\phi_{C,B}(k+1)$), and delaystamp ($\psi_{C,A}(k+1)$ or $\psi_{C,B}(k+1)$) contained in the packets sent by C in step 7.
- 10) Once the communication starts, the dynamic period adjustment protocol in Section III.B takes over the responsibility of synchronizing the clocks.

IV. PERFORMANCE

In this section we present simulation results to evaluate the performance of these protocols.¹⁰

A. Performance of the Pre-Fetched Buffering Protocol

Our simulation model consists of one sender and one receiver where the sender transmits 1000 video frames with the collection period $\eta_i = 25$ ms and the receiver consumes media units with the playback period $\mu_j = 25$ ms. Each frame contains 0.2 ~ 1.5 Mb data. Suppose that the speeds of the local clocks of the sender and the receiver are perfectly matched, i.e., no local clock drift exists. The maximum and minimum end-to-end delays are used as input variables.

In Table I we show the number of discontinuities occurring based on different maximum and minimum end-to-end delays and the total discontinuity time. The number of discontinuities counts the times that j has no media units to play while the total discontinuity time represents the sum of the persistent time of all discontinuities j suffers. As it is clear in Table I, both the number and the total time of discontinuities increases monotonically with the increasing value of $J_{i,j}$. It can also be seen in Table I, that no discontinuity occurs if the pre-fetched buffer scheme is applied. In Table I we also show the required buffer size by using (1).

¹⁰The simulation is conducted on a Sun workstation without actual media transmissions.

TABLE I
DISCONTINUITY CAUSED BY DELAY JITTER

$\Delta_{i,j}$ (ms)	$\delta_{i,j}$ (ms)	Number of discontinuities	Total discontinuity time (ms)	No. of discontinuities with prefetched buffer	Buffer size required (videoframes)
50	40	3	9.98	0	1
100	75	6	14.92	0	1
150	90	10	36.17	0	3
200	100	10	88.67	0	4

TABLE II
AVERAGE ASYNCHRONY WITH DIFFERENT ADJUSTMENT PERIOD

Adjustment Period (m.u.)	Maximum Asynchrony (ms)	Minimum Asynchrony (ms)	Average Asynchrony (ms)
1	0.303	-0.363	-0.269
2	0.459	-0.951	-0.252
10	1.611	-3.856	-1.04
50	4.577	-18.076	-3.341
100	13.459	-39.564	-8.747
200	9.628	-46	-16.598

B. Performance of the Dynamic Period Adjustment Protocol

Here, our simulation model consists of three users. Users 1 and 2 send media units to user 3 who is the chairman. Both users 1 and 2 suffer clock drift between 1.01 to 0.99. The maximum and minimum end-to-end delay among users are $\Delta_{1,2} = 150$ ms, $\delta_{1,2} = 80$ ms, $\Delta_{1,3} = 50$ ms, $\delta_{1,3} = 40$ ms, $\Delta_{2,3} = 200$ ms, $\delta_{2,3} = 100$ ms. The collection and playback periods of a medium are $\eta_i = \mu_j = 30$ ms for $i = 1, 2$ and $j = 3$. We assume that user 1 and 2 start to send 10 000 media units to user 3 at the same time. The local clocks at user 1 and 2 are one in the beginning and start drifting after transmission. The drifts distribute uniformly between $-0.0005 \sim 0.0005$ per collection period.¹¹ Once the clocks reach the boundaries, 0.99 or 1.01, they jump back to one. Note that the typical local clock drift rate is less than 0.001. In our simulations, we use the large clock drift rate to show that our mechanisms can handle the worst cases.

In Table II, we show the simulation results for the maximum, minimum, and average asynchrony with different adjustment periods. We observe that the average asynchrony increases monotonically when the adjustment period increases. The difference between maximum and minimum asynchrony implies that the variance of the asynchrony also increases with the increasing adjustment period. Note that the average asynchrony is 49.9 ms if the dynamic period adjustment mechanism is not used in the simulation model.

According to the QOS requirements of applications and the level of clock drift, we need to choose appropriate adjustment periods to meet the QOS requirements while keeping the synchronization overhead low. Note that our simulation model, as described, spans in a WAN environment. The dynamic period adjustment mechanism is still able to keep the average asynchrony within a few milliseconds while the period of adjustment action is 50 media units.

¹¹If the drifts are constant, then we need to perform the dynamic period adjustment protocol only once instead of executing it periodically.

TABLE III
ASYNCHRONY WHEN THE INITIAL COLLECTION
TIME SYNCHRONIZATION SCHEME IS APPLIED

$\Delta_{i,j}$ (ms)	$\delta_{i,j}$ (ms)	Asynchrony (ms)
50	40	0.0096
100	75	-0.016
150	90	-0.018
200	100	0.01

TABLE IV
ASYNCHRONY WHEN THE INITIAL PLAYBACK
TIME SYNCHRONIZATION SCHEME IS APPLIED

$\Delta_{i,j}$ (ms)	$\delta_{i,j}$ (ms)	Asynchrony (ms)
50	40	-0.024
100	75	-0.013
150	90	0.015
200	100	0.016

C. Performance of the Initial Collection Time Synchronization Protocol

Here, we simulate a teleconference with three users. We assume that user 3 is the chairman and users 1 and 2 have local clock drift between 1.01 and 0.99 and user 3 has local clock with speed one. The maximum and minimum end-to-end delays $\Delta_{i,j}$ and $\delta_{i,j}$ among three users are assumed to be the same and are used as simulation variables.

Although the global initial collection G_c can be computed for all three users from (15), the local clock drift among them will cause the small amount of asynchrony as shown in Table III. The asynchrony is so small that it is far below human perception.

D. Performance of the Initial Playback Time Synchronization Protocol

The simulation model has the same input parameters as in the previous case. As shown in Table IV the asynchrony is caused by different speeds of timers ω_j for $j = 1, 2, 3$, used to countdown the initial playback time. Note that the asynchronies in Table III and IV are small and almost unchanged while we vary the values of $\Delta_{i,j}$ and $\delta_{i,j}$. In other words, the two synchronization mechanisms for initial collection and playback times perform well in both LAN's and WAN's.

V. CONCLUSION

Our work provides synchronization protocols for multimedia traffic in integrated services networks. Using proposed synchronization protocols for initial collection time and initial playback time, the global initial collection time G_c and global initial playback time G_p can be determined in a distributed manner. Then the dynamic period adjustment protocol takes over the task of resynchronizing local clocks, and maintains the synchronization achieved by synchronizing G_c and G_p . With the cooperation of intermediate nodes in delay estimation, the

dynamic period adjustment protocol provides fast estimation of VGT. The frequency of performing the dynamic period adjustment protocol should be added to the QOS requirements of applications to further reduce the overhead caused by the protocol. In addition, our protocols have the following advantages:

- No additional connection is needed for transmission of feedback units from the receivers to the sources in the unicast, multicast, and retrieval configurations.
- Faster and more accurate response to asynchrony.
- In real time multimedia applications, users are free to enter or leave without affecting the existing connections.
- No *a-priori* knowledge of the distribution of the clock drift rate is required.
- No *a-priori* knowledge of the distribution of the end-to-end delay, except the upper $\Delta_{i,j}$ and lower $\delta_{i,j}$ bounds, is required.
- Network messages are significantly reduced by implicit transmission of time information and pre-fetched buffering.

We also point out that our protocols rely heavily on a reliable broadcast election protocol and precise measurement on delaystamp at intermediate nodes.

ACKNOWLEDGMENT

The authors would like to thank Hui Zhang, Julio Escobar, Tom Little, Venkat Rangan, and referees for their constructive comments.

REFERENCES

- [1] A. Campbell, G. Coulson, F. Garcia, D. Hutchison, and H. Leopold, "Integrated quality of service for multimedia communications," in *Proc. IEEE INFOCOM Conf.*, Apr. 1993, pp. 732-740.
- [2] D. C. Verma, H. Zhang, and D. Ferrari, "Delay jitter control for real-time communication in a packet switching network," in *Proc. TRICOMM Conf.*, Dec. 1991, pp. 35-43.
- [3] D. Ferrari and D. C. Verma, "A scheme for real time channel establishment in wide area networks," *IEEE J. Select. Areas Commun.*, vol. 8, no. 3, pp. 368-379, Apr. 1990.
- [4] D. L. Mills, "Internet time synchronization: the network time protocol," *IEEE Trans. Commun.*, vol. 39, no. 10, pp. 1482-1493, Oct. 1991.
- [5] H. Tokuda and C. W. Mercer, "ARTS: a distributed real time kernel," *ACM Operating Syst. Rev.*, vol. 23, no. 3, July 1989.
- [6] H. Tokuda and T. Nakajima, and P. Rao, "Real time mach: toward a predictable real time system," in *Proc. USENIX Mach Workshop*, Oct. 1990.
- [7] H. Zhang and D. Ferrari, "Rate-controlled static-priority queueing," in *Proc. IEEE INFOCOM Conf.*, Apr. 1993, pp. 227-236.
- [8] J. Escobar, C. Partridge, and D. Deutsch, "Flow synchronization protocol," *IEEE/ACM Trans. Networking*, pp. 111-1121; Apr. 1994.
- [9] L. Besse, L. Dairaine, L. Fedatoui, W. Tawbi, and K. Thai, "Toward an architecture for distributed multimedia applications support," in *Proc. ICMCS*, Boston, May 1994.
- [10] L. Lamont and N. D. Georganas, "Synchronization architecture and protocols for a multimedia news service application," in *Proc. ICMCS*, Boston, May 1994.
- [11] P. V. Rangan, H. M. Vin, and S. Ramanathan, "Communication architectures and algorithms for media mixing in multimedia conferences," *IEEE/ACM Trans. Networking*, vol. 1, no. 1, pp. 20-30, Feb. 1993.
- [12] R. Gusella and S. Zatti, "An election algorithm for a distributed clock synchronization program," in *Proc. IEEE 6th Int. Conf. Dist. Comp. Syst.*, Boston, May 1986, pp. 364-371.
- [13] R. Gusella and S. Zatti, "The accuracy of the clock synchronization achieved by TEMPO in Berkeley UNIX 4.3BSD," *IEEE Trans. Software Eng.*, vol. 15, no. 7, pp. 847-853, Jul. 1989.

- [14] R. Steinmetz, "Synchronization properties in multimedia systems," *IEEE J. Select. Areas Commun.*, vol. 8, no. 3, pp. 401-412, Apr. 1990.
- [15] R. Steinmetz and C. Engler, "Human perception of media synchronization," IBM European Networking Center, Technical Report 43.9310, 1993.
- [16] S. J. Golestani, "A framing strategy for congestion management," *IEEE J. Select. Areas Commun.*, vol. 9, no. 7, pp. 1064-1077, Sept. 1991.
- [17] S. Ramanathan and P. V. Rangan, "Adaptive feedback techniques for synchronized multimedia retrieval over integrated networks," *IEEE/ACM Trans. Networking*, vol. 1, no. 2, pp. 246-260, Apr. 1993.
- [18] T. D. C. Little and A. Ghafoor, "Synchronization and storage models for multimedia objects," *IEEE J. Select. Areas Commun.*, vol. 8, no. 3, pp. 413-427, Apr. 1990.
- [19] T. D. C. Little and A. Ghafoor, "Multimedia synchronization protocols for broadband integrated services," *IEEE J. Select. Areas Commun.*, vol. 9, no. 9, pp. 1368-1382, Dec. 1991.
- [20] T. Fisher, "Real time scheduling support in Ultrix-4.2 for multimedia communication," in *Proc. 3rd IEEE Workshop Network Operating Support Digital Audio Video*, Nov. 1992, pp. 282-288.
- [21] W. A. Montgomery, "Techniques for packet voice synchronization," *IEEE J. Select. Areas. Commun.*, vol. SAC-1, no. 6, pp. 1022-1027, Dec. 1983.



Wei Yen was born in Kaohsiung, Taiwan, ROC. He received the B.S. degree from the Tatung Institute of Technology, Taiwan, in 1989, and the M.S. degree from the Georgia Institute of Technology, Atlanta, GA, in 1993. He is currently a Ph.D. student in the School of Electrical and Computer Engineering at the Georgia Institute of Technology.

His current research interests include communication protocol design and analysis for multimedia applications, multicast routing, and ATM networks.



Ian F. Akyildiz (M'86-SM'89) received the B.S., M.S., and Ph.D. degrees in computer engineering from the University of Erlangen-Nuernberg, Germany, in 1978, 1981, and 1984, respectively.

He is a Professor with the School of Electrical and Computer Engineering at the Georgia Institute of Technology, Atlanta, GA. He has held visiting professorships at the Universidad Tecnica Federico Santa Maria, Chile, as well as the Université Pierre et Marie Curie and the Ecole Nationale Supérieure Telecommunications, both in Paris, France. He is an

editor for *Computer Networks and ISDN Systems Journal*, *IEEE TRANSACTIONS ON COMPUTERS*, *ACM-Baltzer Journal of Wireless Networks*, as well as for *ACM-Springer Journal for Multimedia Systems*. He has guest-edited several special issues including; "Parallel and Distributed Simulation Performance" for *ACM Transactions on Modeling and Simulation* and "Networks in the Metropolitan Area" for the *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*. His current research interests include; B-ISDN (ATM) networks, wireless networks, performance evaluation, and parallel simulation.

Dr. Akyildiz is a member of ACM (SIGCOMM), and has been a national lecturer for ACM since 1989. He received the Don Federico Santa Maria Medal for his services to the Universidad of Federico Santa Maria, Chile, and he is listed in *Who's Who in the World (Platinum Edition)*. He received the ACM Best Distinguished Lecturer Award for 1994.