# On the Synchronization Mechanisms for Multimedia Integrated Services Networks

**Wei Yen and Ian F. Akyildiz**

School of Electrical and Computer Engineering
Georgia Institute of Technology; Atlanta, GA 30332
Tel: 1-404-894-5141; Fax: 1-404-853-9410
E-mail: wei@eecom.gatech.edu; ian@armani.gatech.edu

**Abstract.** With the advance of communication technology, integrated services networks make possible real-time, multimedia applications. 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 it 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 mechanisms work effectively in both LAN and WAN environments.

*Key Words: High Speed Networks, Multimedia Services, Synchronization, Asynchrony, Delay Jitter, Local Clock Drift, Initial Collection Time, Initial Playback Times, Performance Evaluation.*

## 1 Introduction

Real-time[1], multimedia applications present a critical problem to integrated services networks – synchronization. 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.

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

For example, video/voice mail 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 which is a many-to-one relation. In this configuration, a receiver retrieves media from different senders. A user may, for example, get video and voice from two different databases. Finally, there is the *group* configuration which is a many-to-many relation. Many users in a communication group, each play the role of sender and receiver. Teleconferencing examplifies the group configuration. Note that the unicast, multicast, and retrieval configurations are special cases of the group configuration.

In all four configurations, four sources of asynchrony can disrupt synchronization: (*a*) Delay jitter, (*b*) Local clock drift, (*c*) Different initial collection times and (*d*) Different initial playback times.

*(a) Delay Jitter.* Delay jitter is the variation of end-to-end delay which consists of three parts [6]. 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 or from 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 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/assembling 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 preceeding units. In such a case, a playback *discontinuity* occurs at the receiver. Delay jitter[2] becomes so significant in WAN environments that playback discontinuity is perceptible to humans.

*(b) Local Clock Drift.* 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* 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*.

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

*(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, then 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. 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 [12, 17]. 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.

## 2 Related Work

Recently there has been a growing interest in the development of synchronization protocols to solve asynchrony problems. Several papers have been published dealing with various types of asynchrony problems[3].

The adaptive feedback protocol [15] 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 [15], a

[3] A study on the sensitivity of human perception for various multimedia applications has been done in [13].

multimedia server transmits media units to several mediaphones which 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 WANs because its synchronization guarantees are limited by the maximum network delay.

Pre-compiled scheduling protocol [16, 17] 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 [16, 17] 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 [5, 14] can be used to correct delay jitter problems. It distributes the synchronization responsibility among intermediate nodes between senders and receivers. However, the intermediate nodes which perform synchronization are unlikely to be work-conserving. The resulting average network delay will become longer while the variance becomes smaller.

The flow synchronization protocol [6] 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 [4] in a synchronization group exchange flow information periodically so that the common synchronization delay for the group can be calculated. The main problem of this protocol is the assumption of the underlying clock 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 [8]. This protocol, first, determines the upper bound of tolerant asynchrony among media streams according to the QoS requirements of applications. In order to synchronize media streams, this protocol adds "intentional delay" to those media streams which experience less end-to-end delay without violating the tolerant asynchrony bound. User interac-

[4] "users " are identical to "processors" in [6].

tions, including pausing, skipping, and scanning forward or backward, are also addressed in this paper. However, the local clock drift problem 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 [9] 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 unresolved the problems of different initial collection times and different initial playback times. 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 [13]. Their results define synchronization requirements for different types of multimedia applications.

In this paper, we develop synchronization protocols proposed in section 3 to eliminate the four sources of asynchrony for the unicast, multicast, retrieval and group multimedia configurations in real-time broadband integrated services networks. Our protocols should be placed at orchestration/sychronization layer to provide synchronization services to application layer [1, 7]. However, application layer needs to inform our protocols of the description of multimedia objects including the synchronous flows and their collection and playback periods. In section 4, we demonstrate that our protocols perform equally well in LANs and WANs[5] without a global clock. In section 5 we conclude the paper.

## 3 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. Each user in the group has full duplex connection to all other users in the group. Moreover, each user has its own processor and may play roles of a sender or a receiver or both as Figure 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.

In teleconference, 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 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

[5] In some cases, LAN delay can be greater than WAN delay. However, in this paper, we assume end-to-end delay is less than $20ms$ in LANs and greater than $100ms$ in WANs.

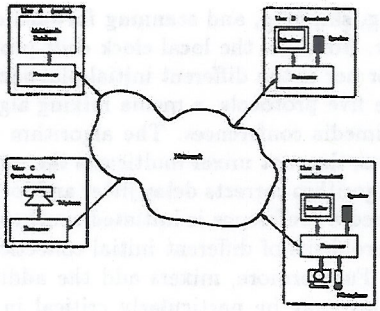

Fig. 1. User Model

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. However, the playback times of the temporally related media units at each audience need not to be the same because fairness is not the concern in this case.

Distributed multimedia databases provide on-demand multimedia services to an audience. The distributed databases playing the roles of senders need to collect media units sychronously. 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.

### 3.1 Compensating the Delay Jitter Effect

The asynchrony caused by delay jitter between sender $i$ and receiver $j$ can be solved by prefetched buffering with buffer size $B_j$ given by the following formula [15].

$$B_j = \lceil \frac{J_{i,j}}{\mu_j} \rceil \tag{1}$$

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

$$J_{i,j} = \Delta_{i,j} - \delta_{i,j} \tag{2}$$

with $\Delta_{i,j}$ is the maximum end-to-end delay between $i$ and $j$; $\delta_{i,j}$ is the minimum end-to-end delay between $i$ and $j$; $\mu_j$ is the playback period of a medium at receiver $j$; and $\lceil x \rceil$ is the smallest integer greater than x. If $J_{i,j}$ can be kept small, then the required prefetched buffer size $B_j$ will decrease as can be seen in equation (1).

Since the values of $\Delta_{i,j}$, $\delta_{i,j}$ and $\mu_j$ can be determined in the negotiation phase for multimedia communication between users $i$ and $j$, the delay jitter

effect can be solved by selecting an appropriate size of prefetched buffer at the receiver as given in equation (1).

### 3.2 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 [11] which 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 3.3). 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.

*Distributed Election Algorithm.* In the negotiation phase, each user of a group generates a one-byte random number between 0 and $2^8$ and broadcasts it to all other users. 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 ignores that number and selects the next largest number from the remaining set. The search for a chairman continues until a unique maximum is found. If the algorithm fails to pick a chairman by using the set of random numbers generated by the users in the first round, then it proceeds with another round in the same fashion. Once the chairman is selected, it creates a queue which 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 [10]; there the existence of a master clock[6] is assumed and the algorithm simply elects the next master clock when the original one fails.

*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 decided by $i$ and $j$ in the negotiation phase by $\mu_j^{(init)}$ and the adjusted playback period which we want to achieve by $\mu_j$. Note that the initial playback period $\mu_j^{(init)}$ at $j$ and the collection period $\eta_i$ 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 $\mu_j$.

We stamp the first packet of each media unit $k$ from $i$ to $j$ with the timestamp $\phi_{i,j}(k)$ at which $i$ began to collect the media unit. The delaystamp $\psi_{i,j}(k)$ is the sum of all times spent by the packet in the nodes (switches) of the network plus

[6] "Master clock" and "chairman" are equivalent in [10].

the collection delay at $i$ and the delivery delay at $j$ [18]. The $k$-th media unit sent from user $i$ is available to be played by $j$ at time $S_{i,j}(k)$ which is

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

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

Let

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

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

Substituting equation (3) into equation (4) we obtain

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

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

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

In this case equation (5) can be rewritten as:

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

If the *intercollection* time is not constant, then we can use equation (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 which 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 $\mu_j$ so that the time user $i$ spends from zero to the collection period $\eta_i$ is the same as the time user $j$ spends from zero to the adjusted playback period $\mu_j$. 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 $\mu_j$ at user $j$ and the collection period $\eta_i$ at user $i$:

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

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

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

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

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

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

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

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

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

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. The period of adjusting $\mu_j$ 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 4.

### 3.3 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 3.2 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 [6, 9, 17]. 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 [6, 17]. 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 every user can compute the VGT value by using the equation (3). In the following we explain how to measure the propagation delay.

In the negotiation phase, user $i$ transmits a packet $\alpha$ 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 3.2. After receiving the packet $\alpha$, user $j$ adds the time that $\alpha$ spends there to the delaystamp $\psi_{i,j}(\alpha)$ and sends $\alpha$ back to user $i$. So, the packet $\alpha$ comes back to user $i$ at time $T_{i,j}(\alpha)$ which is:

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

where $\psi_{i,j}(\alpha)$ is the accumulated delaystamp for packet $\alpha$; $2\theta_{i,j}$ 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 $\alpha$ for the connection can be obtained by rewriting equation (12).

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

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 equation (3). According to equation (3), the estimated propogation 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:

$$I_i \geq \max_j\{\Delta_{i,j}\} + \tau_i \tag{14}$$

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 $\tau_i$ 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 $G_c$.

$$G_c = \max_i\{I_i\} \tag{15}$$

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

### 3.4 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 3.2 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.

Now 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. 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 the propagation delay $\theta_{i,j}$ estimated in the negotiation phase by the mechanism we introduced in section 3.3.

With the knowledge of the two parameters $\delta_{i,j}$ and $\Delta_{i,j}$, we can calculate the prefetched buffer sizes $B_j(m)$ for any medium $m$ of user $j$ using equation (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

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

where $I_i$ is the initial collection time of the media units from $i$; $B_j(m)$ is the prefetched buffer size of medium $m$ of receiver $j$ (equation (1)); $\eta_i(m)$ 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 equation (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 prefetched 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 wastes of network resources.

User $i$ can compute the latest playback time $L_{i,j}$ for any user $j$ in the communication group from equation (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:

$$M_i = \max_j\{L_{i,j}\} \tag{17}$$

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 $L_{i,j}$, equation (16),

for user $j$ in the communication group. Then, $i$ can get $M_i$ from equation (17) and distribute it to every user in the group.

We can compute $M_i$ for each user $i$ in the group. However, in order to get the global initial playback time, 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 equation (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 $G_c$, equation (15). Thus, each user $i$ in the group distributes its playback waiting period $\lambda_i$ in terms of VGT to all users in the group where

$$\lambda_i = M_i - I_i \tag{18}$$

In the playback waiting period $\lambda_i$, all users fill their prefetched buffer with media units from user $i$ and wait for the synchronous playback of the media units. After receiving the playback waiting period $\lambda_i$ 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

$$G_p = G_c + \max_i\{\lambda_i\} \tag{19}$$

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

For each user $j$ in the group, a timer $\omega_j$ is set up with the initial value given by equation (20). The timer $\omega_j$ starts to countdown when the first media unit from the *chairman* is received.

$$\omega_j = G_p - \rho_j \tag{20}$$

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

Afterwards, 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 $\omega_j$ reaches zero, user $j$ starts to playback. Once the receivers begin playback, only the dynamic period synchronization mechanism of section 3.2 is involved.

## 4 Performance

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

### 4.1 Performance of the Prefetched Buffering Protocol

Our simulation model consists of one sender and one receiver where the sender transmits 1000 video frames with the collection period $\eta_i = 25ms$ and the receiver consumes media units with the playback period $\mu_j = 25ms$. Each frame

contains $0.2 \sim 1.5$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 1 we show the number of discontinuities occuring 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 the total discontinuity time represents the sum of the persistent time of all discontinuities $j$ suffers. As it is clear in Table 1, both the number and the total time of discontinuities increase monotonically with the increasing value of $J_{i,j}$. It can also be seen in Table 1, that no discontinuity occurs if the prefetched buffer scheme is applied. In Table 1 we also show the required buffer size by using equation (1).

| $\Delta_{i,j}$ $(ms)$ | $\delta_{i,j}$ $(ms)$ | 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 1.** The discontinuity caused by delay jitter

### 4.2 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 $\Delta_{1,2} = 150ms$, $\delta_{1,2} = 80ms$, $\Delta_{1,3} = 50ms$, $\delta_{1,3} = 40ms$, $\Delta_{2,3} = 200ms$, $\delta_{2,3} = 100ms$. The collection and playback periods of a medium are $\eta_i = \mu_j = 30ms$ for $i = 1, 2$ and $j = 3$. We assume that user 1 and 2 start to send 10000 media units to user 3 at the same time. The local clocks at user 1 and 2 are 1 in the beginning and start drifting after transmission. The drifts[7] distribute uniformly between $-0.0005 \sim 0.0005$ per collection period. Once the clocks reach the boundaries, 0.99 or 1.01, they jump back to 1. 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 2, 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.

[7] If the drifts are constant, then we need to perform the dynamic period adjustment protocol only once instead of doing it periodically.

| 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 |

**Table 2.** The average asynchrony with different adjustment period

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 $49.9ms$ if the dynamic period adjustment mechanism is not used in the simulation model.

According to the quality of service (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 asychrony within a few milliseconds while the period of adjustment action is 50 media units.

### 4.3 Performance of the Initial Collection Time Synchronization Protocol

Here we simulate a teleconference with 3 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 1. 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 equation (15), the local clock drift among them will cause the small amount of asynchrony as shown in Table 3. The asynchrony is so small that it is far below human perception.

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

**Table 3.** The asynchrony when the initial collection time synchronization scheme applied

### 4.4 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 4 the asynchrony is caused by different speeds of timers $\omega_j$ for $j = 1, 2, 3$, which are used to countdown the initial playback time. Note that the asynchronies in Table 3 and 4 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 LANs and WANs.

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

**Table 4.** The asynchrony when the initial playback time synchronization scheme applied

## 5 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 mechanism 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.
- Faster and more accurate response to asynchrony.
- In real-time multimedia applications, users are free to enter/leave without affecting the existing connections.
- No a priori knowledge of the distribution of end-to-end delay and of the clock drift rate are required.
- Network messages are significantly reduced by implicit transmission of time information and prefetched buffering.

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

*Acknowledgement: We would like to thank Hui Zhang, Julio Escobar, Tom Little, and Venkat Rangan for their suggestions and constructive comments.*

## References

1. A. Campell, G. Coulson, F. Garcia, D. Hutchison, and H. Leopold, "Integrated Quality of Service for Multimedia Communications," *Proc. of the INFOCOM '93 Conference*, pp. 732-740, Apr. 1993.
2. D. C. Verma, H. Zhang and D. Ferrari, "Delay Jitter Control for Real-Time Communication in a Packet Switching Network," *Proc. of the TRICOMM '91 Conference*, pp. 35-43, Dec. 1991.
3. D. Ferrari and D. C. Verma, "A Scheme for Real-Time Channel Establishment in Wide-Area Networks," *IEEE Journal on Selected Areas in Comm.*, Vol. 8, No. 3, pp. 368-379, Apr. 1990.
4. D. L. Mills, "Internet Time Synchronization: The Network Time Protocol," *IEEE Transactions on Communications*, Vol. 39, No. 10, pp. 1482-1493, Oct. 1991.
5. H. Zhang and D. Ferrari, "Rate-Controlled Static-Priority Queueing," *Proc. of the INFOCOM '93 Conference*, pp. 227-236, Apr. 1993.
6. J. Escobar, D. Deutsch, and C. Partridge, "Flow Synchronization Protocol," *IEEE Global Communications Conference*, pp. 1381-1387, Dec. 1992.
7. L. Besse, L. Dairaine, L. Fedaoui, W. Tawbi and K. Thai, "Towards an Architecture for Distributed Multimedia Applications Support," *Proc. of ICMCS'94*, Boston, May 1994.
8. L. Lamont and N. D, Georganas, "Synchronization Architecture and Protocols for a Multimedia News Service Application," *Proc. of ICMCS'94*, Boston, May 1994.
9. P. V. Rangan and H. M. Vin, and S. Ramanathan, "Communication Architectures and Algorithms for Media Mixing in Multimedia Conferences," *IEEE/ACM Transactions on Networking*, Vol. 1, No. 1, pp. 20-30, Feb. 1993.
10. R. Gusella and S. Zatti, "An Election Algorithm for a Distributed Clock Synchronization Program," *IEEE 6th International Conference on Distributed Computing Systems*, pp. 364-371, Boston, May 1986.
11. R. Gusella and S. Zatti, "The Accuracy of the Clock Synchronization Achieved by TEMPO in Berkeley UNIX 4.3BSD," *IEEE Transactions on Software Engineering*, vol. 15, no. 7, pp. 847-853, Jul. 1989.
12. R. Steinmetz, "Synchronization Properties in Multimedia Systems," *IEEE Journal on Selected Areas in Comm.* vol. 8, no. 3, pp. 401-412, Apr. 1990.
13. R. Steinmetz and C. Engler, "Human Perception of Media Synchronization," IBM European Networking Center, Technical Report 43.9310, 1993.
14. S. J. Golestani, "A Framing Strategy for Congestion Management," *IEEE Journal on Selected Areas in Comm.*, Vol. 9, No. 7, pp. 1064-1077, Sep. 1991.
15. S. Ramanathan and P. V. Rangan, "Adaptive Feedback Techniques for Synchronized Multimedia Retrieval over Integrated Networks," *IEEE/ACM Transactions on Networking*, Vol. 1, No. 2 , pp. 246-260, Apr. 1993.
16. T. D. C. Little and A. Ghafoor, "Synchronization and Storage Models for Multimedia Objects," *IEEE Journal on Selected Areas in Comm.*, Vol. 8, No. 3, pp. 413-427, Apr. 1990.

17. T. D. C. Little and A. Ghafoor, "Multimedia Synchronization Protocols for Broadband Integrated Services," *IEEE Journal on Selected Areas in Comm.*, Vol. 9, No. 9, pp. 1368-1382, Dec. 1991.
18. W. A. Montgomery, "Techniques for Packet Voice Synchronization," *IEEE Journal on Selected Areas in Comm.*, Vol. SAC-1, No. 6, pp. 1022-1027, Dec. 1983.