

Towards the Delay and Synchronization Control for Networked Real-Time Multi-Object Multimedia Applications

Haining Liu and Magda El Zarki

School of Information and Computer Science, University of California, Irvine

Email: {haining,magda}@ics.uci.edu

Abstract

Due to the lack of QoS support, ensuring an acceptable application level QoS for the real-time delivery of multi-object multimedia presentations on the current Internet is very challenging. Currently, the only feasible solution is to artificially “equalize” the variable network delays. In this paper, we first investigate the application-level objective performance measures which reflect the user-perceived quality, and then propose an adaptive delay and synchronization control scheme making use of those metrics in real-time. The proposed scheme adopts a distributed timing model and can be configured with two inter-stream synchronization options based upon the application’s own characteristics. We take a wireline video conferencing application consisting of one audio stream and one video stream as an example. We detail the operation of the proposed scheme and investigate the resultant performance. The simulation results show that small synchronization phase distortions, low MDU loss percentages and low average end-to-end delays could be achieved simultaneously. We further discuss a more challenging issue raised by mobile wireless applications. A preliminary solution is proposed.

1. Introduction

The advances in communications and media coding technologies have led to the emergence of networked Real-time Multi-object Multimedia Applications (RM-MAs). People can now hold live video conferences consisting of both audio and video across the network by using tools such as MS Netmeeting. However, end users still often suffer from poor quality due to the unpredictable channel behaviour of packet-switching networks. Different from networked data applications, RMMAs usually generate continuous periodic traffic—multiple objects are periodically produced at the source, fragmented into media data units (MDUs), packetized and transported in succession to the destination in real-time. This feature naturally raises two application level QoS requirements from

the perspective of end users. First, the original temporal relationships, including both the temporal ordering among the MDUs in a media stream and the relative temporal positioning among the streams (e.g., lip-sync), need to be faithfully restored at the destination. This is referred to as *synchronization control* in the context of multimedia communication. Second, the latency, a.k.a. the end-to-end delay, should be kept low enough to ensure acceptable user interactivity. This requires appropriate *delay control*. Unfortunately, on a packet-switching network like the Internet, neither control can be easily achieved.

Considerable efforts have been made in different layers to fulfill the QoS requirements of the networked RM-MAs. Generally, there are two fundamental approaches: (1) to control the transmission performance (both the delay and the delay variation) in the network level by means of the service guarantees, and (2) to equalize end-to-end delays in the application level by artificially introducing an additional buffering delay and extending the playout deadline. Nonetheless, recent attempts to implement network QoS within Internet2 have proven the insurmountable difficulties of employing QoS models that offer hard QoS guarantees on the current network infrastructure [1]. This leaves delay equalization as the only feasible solution for current networked multimedia systems to maintain the synchronization.

Equalizing the end-to-end delay for networked RMMAs results in a conflict between the synchronization requirements and the delay preference. On the one hand, a larger equalization delay is favored to ensure more in-time MDU arrivals. For example, a typical on-line streaming application today has a 2-3 second buffering period. On the other hand, excessive end-to-end delay inevitably impairs the interactivity. A 2-3 second latency is not acceptable for end users demanding “real-time” experience. Thus, a solution that is adaptive, i.e., capable of striking a balance adaptively between the delay and the synchronization requirements, is desirable. The solution should be able to reduce the end-to-end delay as much as possible while maintain-

ing both types of synchronization.

Previous work on adaptive delay and synchronization control mainly focus on single-object networked multimedia applications [2]-[5]. Most approaches rely on a globally synchronized clock. With reference to the clock, some schemes record historical information and determine the amount of delay statistically [3][4], others adopt a deterministic strategy using run-time information (e.g. buffer-fill indication in [5]). There have only been a few attempts aimed at multi-object applications; each proposes an ad-hoc solution for a specific application scenario. For instance, a scheme assuming specific service for the master object from the underlying network is proposed for wireless PCS systems in [6]. In [7] and [8], an algorithm facilitating the occurrence of synchronization events is designed for an AV teleconferencing system. To this date, it still remains unclear how to apply the synchronization constraints, especially the inter-stream synchronization constraints, to an adaptive delay control scheme to achieve better performance for RMMAs.

In this paper, we extend our previous work in [9] and propose a generic delay and synchronization control scheme for networked RMMAs. The essence of this scheme is that it is receiver-based and application-level QoS oriented. The proposed scheme 1) employs distributed clocks by using a virtual timing mechanism, 2) monitors the synchronization errors in real-time and 3) piece-wisely adjusts the equalization delay to compensate for the delay jitters. We take a wireline video conferencing application consisting of one audio and one video as an example and investigate the performance based on the quality measurement features. A real conference session is emulated and the network trace collected during the session is used in our study. Through simulations, we show that small phase distortions, low MDU loss percentages and low average end-to-end delays can be achieved simultaneously using the proposed scheme. We further reveal a more challenging issue based on the traces recorded on a real WLAN. A solution is then proposed for mobile RMMAs running on WLAN.

The rest of this paper is structured as follows. We discuss the performance metrics for an adaptive delay and control scheme in the next section. Section III details the design of the proposed scheme and Section IV presents the case study of a wireline video conferencing application. We discuss the challenging issue raised by the mobile RMMAs running on WLAN in Section V, and then summarize and conclude the paper in Section VI.

2. Application-Level QoS Metrics

Since our proposed scheme is application-level QoS oriented, we need to define the performance measures which can objectively reflect the ‘‘application quality’’.

The temporal inconsistency in presenting the periodically generated MDUs is defined as *synchronization phase distortion* (SPD) between the MDUs that are played back. For RMMAs, the SPD can be evaluated jointly by the Root Mean Square Error (RMSE) of the inter-sample time of the MDUs in one stream (intra-stream) and the RMSE of the playout time of the closest corresponding data units among streams (inter-stream). We denote the time when the MDU n of stream i is generated and played out by $t_g^i(n)$ and $t_p^i(n)$ respectively, then the intra-stream SPD of stream i is simply given by

$$\tau_i = \sqrt{\frac{\sum_{n=2}^{N_i} [(t_p^i(n) - t_p^i(n-1)) - (t_g^i(n) - t_g^i(n-1))]^2}{N_i}}, \quad (1)$$

where N_i denotes the total number of MDUs played out in stream i . Similarly, we can define the inter-stream SPD between two streams i and j by

$$\tau_{i,j} = \sqrt{\frac{\sum_{m=1}^{N_i} [(t_p^i(m) - t_p^j(n)) - (t_g^i(m) - t_g^j(n))]^2}{N_i}}, \quad (2)$$

where the stream i is the reference stream and the MDU n of stream j corresponds to the MDU m of stream i .

We are also interested in the *loss ratio* because it implicitly indicates the resultant presentation quality. For instance, a certain amount of audio MDU losses which exceeds a threshold value may suggest intolerable quality for the human perception system. The loss ratio of stream i is given by

$$l_i = \frac{(M_i - N_i)}{M_i}, \quad (3)$$

where M_i denotes the total number of MDUs generated in stream i . Note that in Equation (3), the MDUs skipped by the adaptive control are also considered as lost ones. Another important quantity is the average *latency*—the end-to-end delay. It is given by

$$d_i = \frac{\sum_{n=1}^{N_i} [\pi_{net}^i(n) + \pi_{buf}^i(n)]}{N_i} \quad (4)$$

where $\pi_{net}^i(n)$ and $\pi_{buf}^i(n)$ denote the network delay and the buffering delay experienced by MDU n played out in stream i respectively. Note that we do not include the encoding, packetization and decoding delays as they usually make up a constant component of the end-to-end delay.

3. Control Scheme

3.1. Timing Model

Timing is by all means critical for a synchronization control scheme which targets preserving the temporal property of a multimedia presentation. The common approach is to synchronize both ends (the source and the destination) to one reference clock. By doing so, network delays can be

measured accurately. Most proposed schemes assume that such a synchronized clock is available. However, in reality, it is not always feasible to tightly synchronize the clocks of a local and a remote system across a network. Our proposed scheme instead makes use of a distributed virtual timing model which does not require the cooperation from the source end.

The virtual timing model introduces a virtual clock in addition to the actual time. The virtual clock can be simply initialized to be the value of the time stamp carried by the first received MDU of a stream. Then, upon the arrival of each successfully received MDU, two values are available: the generation time, which is sampled at the source and carried in the time stamp, and the arrival time sampled according to the virtual clock. The destination can thus treat the generation time as the scheduled playout time of the MDU, compare the two time values and modify the playout time of the MDUs to achieve an adaptive playout timeline. For example, to increase the end-to-end equalization delay, the clock can be slowed down virtually by increasing the scheduled playout time value. By doing the reverse, the delay can be decreased. An obvious benefit of using this model is that the adaptive control can be safely performed by only referencing local timing information (at the destination end).

Another benefit is that the virtual timing values can be used directly to calculate the resultant SPD. If we denote the delay jitter between the MDU $(n - 1)$ and the MDU n in the stream i as ξ , and denote the clock adjustment (if happened between the two arrivals) as Δ , we can get $t_p^i(n) - t_p^i(n - 1) = T_p^i(n - 1) + \Delta + \xi - T_p^i(n - 1) = T_p^i(n) - T_p^i(n - 1)$, where $T_p^i(n)$ denotes the playout time of the MDU n with reference to the virtual clock. That is, we can obtain τ_i as follows:

$$\tau_i = \sqrt{\frac{\sum_{n=2}^{N_i} [(T_p^i(n) - T_p^i(n-1)) - (t_g^i(n) - t_g^i(n-1))]^2}{N_i}} \quad (5)$$

3.2. Intra-Stream Synchronization Control

In order to reduce the intra-stream SPD caused by variable delays, two possible scenarios, namely “in-time arrival” and “late arrival”, need to be handled carefully. For an arrived yet deadline-missed MDU, the solution would be just skipping it, but this could result in excessive skipings. Considering the fact that the human perception system can tolerate a certain amount of rendering jitter (phase distortion) of multimedia objects, we can allow those MDUs not missing the schedule by too much to be rendered with careful control. We propose to partition the arrival time of the MDUs in a stream into two regions—“playout” and “discard”, as illustrated in Figure 1. We denote $T_a^i(n)$ as the arrival time of the MDU n by referencing the virtual clock, and $T_g^i(n)$ as the scheduled playout time. Note that using

the virtual timing model, we have

$$T_g^i(n) = t_g^i(n). \quad (6)$$

On the virtual time axis, we extend the schedule deadline by a discard boundary δ_i for MDUs in the stream i . If $T_a^i(n) \leq T_g^i(n) + \delta_i$, the MDU n falls into the “playout” region and can be rendered. If $T_a^i(n) > T_g^i(n) + \delta_i$, the MDU n falls into the “discard” region and is skipped to avoid error. Note, that to simplify the implementation, the following condition must hold

$$T_g^i(n-1) < T_g^i(n-1) + \delta_i < T_g^i(n). \quad (7)$$

The actual playout time of the MDU n is determined as follows. If n arrives early, then it is played out at its scheduled time. Otherwise, it depends on how $(n - 1)$ was played out. If the MDU $(n - 1)$ was rendered on time, then MDU n is played back immediately at the time it arrives. If $(n - 1)$ was rendered later than its scheduled time, then α_i , a smoothing parameter, is applied to ensure that $(n - 1)$ is rendered with enough time to minimize the SPD caused by this lateness, i.e., $T_p^i(n) = \max[T_p^i(n - 1) + T_g^i(n) - T_g^i(n - 1) - \alpha_i, T_a^i(n), T_g^i(n)]$.

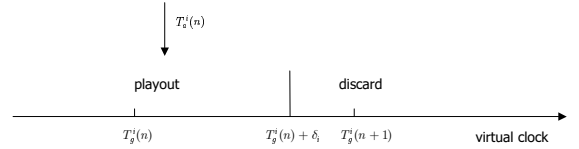


Figure 1. Temporal partition of MDU arrival on virtual time axis

Since the playout time determination process could potentially introduce large SPD, we need to exert tight intra-stream synchronization control for each stream. Meanwhile, the resultant MDU losses (including those skipped) also need to be controlled. We hence monitor both the SPD and the MDU loss ratio in real-time using a sliding window for each stream. Based on the monitoring results, the equalization delay is regulated piecewisely by adjusting the virtual clock. If either one has exceeded the threshold value, we slow down the clock to increase the buffering time. If none of the errors occur, we speed up the clock to balance the latency requirement. For stream i , the measured SPD is given by

$$\hat{\tau}_i = \sqrt{\frac{\sum_{n=2}^{W_i} [(T_p^i(n) - T_p^i(n-1)) - (T_g^i(n) - T_g^i(n-1))]^2}{W_i}} \quad (8)$$

where W_i is the current window size and \bar{W}_i is the maximum window size. W_i ranges from 2 to \bar{W}_i , and it is incremented by 1 upon the playout of each MDU. The measured

loss ratio is defined as

$$\hat{l}_i = \frac{\psi_i}{W_i}, \quad (9)$$

where ψ_i is the loss count within the current monitoring window. Our scheme controls the virtual clock at the destination end based on a real-time calculation of the synchronization errors as defined by Equations (8) and (9).

3.3. Inter-Stream Synchronization Control

The existence of inter-stream synchronization constraints among the media streams in a RMMA makes the whole control scheme more complicated. Two issues need to be addressed: how to apply the inter-stream synchronization constraints, and how to integrate two types of synchronization control into one scheme. As the inter-stream synchronization control concerns mutual temporal relationship, we must identify which stream (object) plays the major role. In accordance with this principle, we classify the streams that need to be inter-synchronized as either a *master* or a *slave* considering their capability of tolerating delay jitters. For example, in an AV presentation, the audio which is more sensitive to the delay variations, is the master. A master dominates the inter-stream synchronization control, and vetos the request of clock speed-up from the slaves.

There are two options to choose from on how to apply the inter-stream synchronization constraints. The first one is rather straightforward—applying the inter-stream synchronization control to MDUs of a slave. Only one virtual clock is maintained for a group of streams that need to be inter-synchronized, and the playout time of slave MDUs are checked in an MDU-by-MDU fashion to satisfy the inter-stream synchronization requirement. We define this strategy as *hard enforcement*. Figure 2 illustrates this approach using a RMMA consisting of one master and one slave. Details regarding playout time adjustment of slave MDUs can be found in our previous work [9].

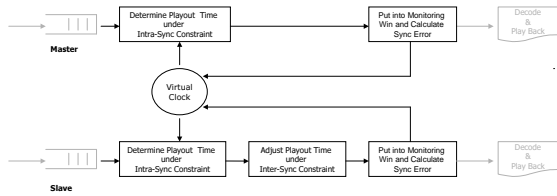


Figure 2. An example synchronization control scheme using hard enforcement

Another option is to allow each stream to maintain its own virtual clock, and the inter-stream synchronization constraints are only applied when clock adjustment is needed. To be specific, after each clock adjustment, the offset between the virtual clocks of the master and the slave stream must always satisfy

$$abs(o^m - o^s) \leq \bar{r}_{m,s} - max(\delta_m, \delta_s), \quad (10)$$

where o^m and o^s denote the offset between the source clock and the virtual clock of the master and the slave respectively. The inter-stream synchronization is maintained in the sense that the clock offset between the master and the slave does not exceed the threshold value. Since this strategy maintains the inter-stream synchronization in a more relaxed form, we define it as *soft enforcement*. Compared with the hard enforcement strategy described above, less comparison operations are involved when using soft enforcement approaches. The only checkpoints are the time instants when clock adjustment is needed, and this results in less operational overhead when implemented in a real system. Again, we take a RMMA consisting of one master and one slave as the example, and show this approach in Figure 3. Note that using either option, the clock adjustment is driven by the measured intra-stream synchronization errors.

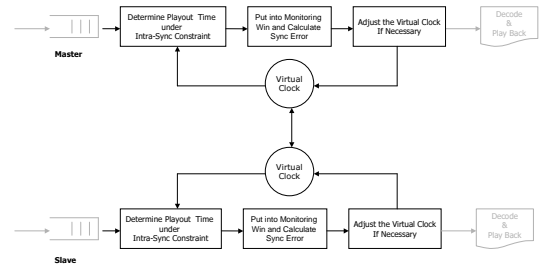


Figure 3. An example synchronization control scheme using soft enforcement

3.4. Clock Adjustment

In the proposed scheme, adjusting the virtual clock is equivalent to adjusting the buffering delay to compensate for the delay jitters. It appears in the form of expanding/shortening the playout duration of one MDU, or proactively skipping received MDUs. When the measured synchronization errors (SPD and MDU losses) are all zero over the maximum monitoring window, we can speed up the virtual clock by Δ_1^i . Once either type of the measured synchronization errors (SPD or MDU losses) has exceeded the

threshold value, the virtual clock needs to be slowed down, i.e. the buffering delay needs to be increased to compensate for it. If it is the measured SPD which triggers the clock adjustment, we slow down the virtual clock by Δ_2^i . If it is the measured loss ratio that has exceeded the threshold value, then the clock should be slowed down by Δ_3^i . Figure 4 illustrates the actual effect of the virtual clock adjustment. The scheme adjusts the discard boundary of MDUs dynamically according to their delay probability distribution. We take a wireline video conferencing application consisting of one audio and one video as an example and instantiate the proposed scheme in the next section.

4. Case Study (Wireline Video Conferencing)

We show how the proposed scheme operates in a wireline video conferencing application in this section. A real network delay trace is used to investigate the performance.

4.1. Clock Adjustment Parameters

We assume the network delay process of wireline networks is "stationary", i.e., over any two fairly large sampling periods, the network delay p.d.f. stay the same. As such, when clock speed-up is needed, the adjustment amount Δ_1^i is calculated by

$$\Delta_1^i = \max_{n \in W_i} \{abs[T_a^i(n) - T_g^i(n)]\}. \quad (11)$$

It is the minimum buffering delay that MDUs of stream i in the monitoring window experienced. After the conservative adjustment, we can expect that MDUs that arrive during the next monitoring period satisfy the intra-synchronization constraint.

When the measured SPD exceeds the threshold value, we slow down the virtual clock by Δ_2^i . It is calculated as follows. We assume a "heavy-tail" network delay distribution as shown in Figure 4, and assume that the adjustment happens when o^i , the offset between the source clock and the virtual clock, is located at the tail portion of the delay p.d.f.

We further assume that within a monitoring window W_i , every MDU of stream i arrived later than the scheduled playout time but ahead of the discard boundary δ_i (a worst-case scenario), and uniformly distributed within this range. That is, RV $X^i(n) = T_a^i(n) - T_g^i(n)$ conforms to a uniform distribution within $[0, \delta_i]$, where $n \in W_i$. Then according to the Central Limit Theorem, we can get

$$\sum_{i=1}^{W_i} [X^i(n) - X^i(n-1)]^2 \sim N(W_i \mu, W_i \sigma^2), \quad (12)$$

where μ and σ^2 are the expected value and the variance of RV $[X^i(n) - X^i(n-1)]^2$, respectively. With a probability

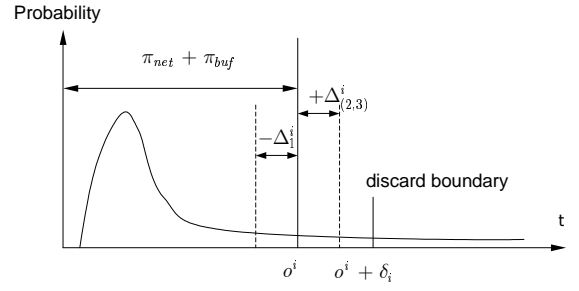


Figure 4. The assumed delay distribution with a heavy tail

of 1, we can further get

$$\frac{\sum_{i=1}^{W_i} [X^i(n) - X^i(n-1)]^2}{W_i - 1} \leq \frac{(W_i \cdot \mu + 3.1 \cdot \sqrt{W_i} \sigma)}{W_i - 1}. \quad (13)$$

If the network delay is an i.i.d. process, then after the clock is adjusted by Δ_2^i , we should expect

$$\frac{(W_i \cdot \mu + 3.1 \cdot \sqrt{W_i} \sigma)}{W_i - 1} \geq \frac{(\bar{W}_i \cdot \mu' + 3.1 \cdot \sqrt{\bar{W}_i} \sigma')}{W_i - 1}, \quad (14)$$

where μ' and σ'^2 are the expected value and the variance of RV $[X'(n) - X'(n-1)]^2$ ($X'(n)$ conforms to a uniform distribution within $[0, \delta_i - \Delta_2^i]$). On the basis of a worst case, after the clock is slowed down by Δ_2^i , we obtain

$$\sqrt{\frac{W_i - 1}{6}} \cdot \delta_i = \sqrt{\frac{\bar{W}_i - 1}{6}} \cdot (\delta_i - \Delta_2^i). \quad (15)$$

Hence,

$$\Delta_2^i = \left(1 - \sqrt{\frac{W_i - 1}{\bar{W}_i - 1}}\right) \cdot \delta_i. \quad (16)$$

If the measured loss ratio has exceeded the threshold value, then after the clock is slowed down by Δ_3^i , the following equation

$$\frac{\left\{ \max_{n \in W_i} [T_a^i(n) - T_g^i(n)] - \Delta_3^i \right\}}{\left\{ \max_{n \in W_i} [T_a^i(n) - T_g^i(n)] - \delta_i \right\}} = \frac{W_i}{\bar{W}_i} \quad (17)$$

should hold to ensure that the measured loss ratio over the next window should not exceed the threshold value. As a result,

$$\Delta_3^i = \max_{n \in W_i} [T_a^i(n) - T_g^i(n)] \left(1 - \frac{W_i}{\bar{W}_i}\right) + \delta_i \cdot \frac{W_i}{\bar{W}_i}. \quad (18)$$

4.2. Simulations

To make the study more accurate, we use a real network trace instead of pseudo network delay generated by theoretical models. Assuming that the video conferencing application uses the RTP/UDP/IP protocol stack, we emulated

a real conference session on a residential network that experiences congestion during certain hours of the day. We further assumed the application uses G.723.1 as the audio codec and MPEG-4 as the video codec. The size and the sending interval of the packets were set the same as those of the real MDUs. The round trip time (RTT) of each media MDU was then recorded and post-processed to obtain the one-way network delay¹. Since we are concerned primarily with delay variations, such a simplification (using RTT) is valid for our study. For MDUs transmitted in multiple packets, the largest packet delay value is used. The cumulative probability distribution of the delays recorded in this trace is shown in Figure 5. Note the “heavy-tail” feature displayed by the delay distribution.

We use the performance threshold values reported in previous work [10] and [11] in our simulations. For the audio, we set the $\bar{\sigma}_a$ (intra-stream SPD threshold) and \bar{l}_a (loss ratio threshold) to be 2 ms and 0.02, respectively. The smoothing factor α_a is set to be 10ms, which in turn guarantees that each rendered audio MDU lasts for at least 30-10=20 ms. For the video, $\bar{\sigma}_v$ and \bar{l}_v are set to be 5 ms and 0.03 respectively. The smoothing factor α_v is set to be 16.667 ms. $\bar{\tau}_{a,v}$, the maximum allowed inter-stream skew between audio and video, is set to be 80 ms. We set the maximum sliding window size of both audio and video to be 1000, which guarantees a sufficiently large sampling space for monitoring. For both options of inter-stream synchronization control, we ran the simulations with different sets of $\bar{\sigma}_a$ and $\bar{\sigma}_v$. The final results are presented in Figures 6-12.

4.3. Performance

From Figures 6-12, it can be concluded that in our proposed scheme, the discard boundary δ_i (δ_a or δ_v in the video conferencing application) is the tradeoff between the resultant MDU losses, the overall intra-stream synchronization errors and the average end-to-end delays. Smaller δ_a and δ_v clearly lead to smaller intra-stream SPD as well as smaller average end-to-end delays for both the audio and the video streams. At the same time, higher amount of losses of both the audio MDUs and the video MDUs result. Larger δ_a and δ_v , on the other hand, allow more late MDUs to be played out and result in relatively larger intra-stream SPD, larger average end-to-end delays and smaller loss ratio. However, despite the choice of the discard boundary, the proposed scheme clearly demonstrates the ability to ensure a certain application layer quality, i.e., control the resultant SPD and MDU losses, by regulating the equalization delay. Although

¹ The UDP payload size of each audio MDU was set to be 36 bytes (24 bytes data at 6.3 kb/s + 12 bytes RTP header). The sending interval of audio packets was 30 ms. The UDP payload sizes of the video MDUs were set to be the same as the values in the QCIF sequence “Forman” encoded at 64kb/s. The MDU/frame rate was 15fps.

the actual network delay values vary from a few milliseconds to several hundred milliseconds, the proposed scheme can find an appropriate equalization delay value to well balance the end-to-end delay and the synchronization performance.

Figure 8 shows that, with the same set of parameters, using hard enforcement option generally results in much better inter-stream synchronization performance than when using soft enforcement option. But noticeably, applying a set of large δ_a and δ_v , which in turn allows only small clock adjustment, the proposed scheme using soft enforcement option can yield reasonably good inter-stream synchronization performance (inter-stream SPD below 30 ms). This finding suggests that a specific networked RMMA can select the option (inter-stream synchronization strategy) based on its own characteristics. For example, a RMMA consisting of many multimedia objects could trade a small amount of inter-stream synchronization performance for the reduction of the overhead in a real implementation by using soft enforcement.

It also can be seen that, given the delay characteristics of current wireline IP networks, using a set of large δ_a and δ_v for the proposed scheme does not result in a dramatic increase in the final average end-to-end delay. This further proves that our closed-loop synchronization control effectively “absorbs” the occasional large network delay jitter.

Overall, we have observed that, using the proposed delay and synchronization control scheme with either inter-stream synchronization control option, small synchronization phase distortions, low MDU loss percentages and low average end-to-end delays can be achieved simultaneously.

5. Mobile RMMAs over WLAN

Recently, wireless networking technologies, particularly WLAN (IEEE802.11x), have been maturing. It is projected that RMMAs for portable and embedded devices on campus-wide mobile networks with wi-fi hotspots will become popular in the near future. However, the channel behavior of the WLAN makes the task of ensuring some application-level QoS rather complicated.

Recall that in the previous section we assumed that the network delay process for wireline RMMAs is a stationary process. Such an assumption can be safely made considering the fact that wireline networks usually do not experience frequent performance fluctuation. In fact, our tracing experiments have shown that the probability distributions of wireline networks over different time scales show roughly the same characteristics. Yet the experiments on a WLAN have revealed much different characteristics. Due to the constantly changing wireless channel quality, various moving speed and traffic context, the behavior of the transport channels across WLAN is highly “non-stationary”. Given

this, we could again argue that an adaptive delay and synchronization control scheme is desperately required for mobile RMMAs running over the wireless network since any deterministic schemes configured with pre-determined parameters will simply fail. On the other hand, we have to incorporate more functionality to the proposed scheme. As shown in Figure 13, frequent clock adjustment could result if we simply apply the scheme proposed for wireline RMMAs to wireless mobile applications. Accordingly, extra synchronization errors, which should be avoided, have been brought to the final presentation.

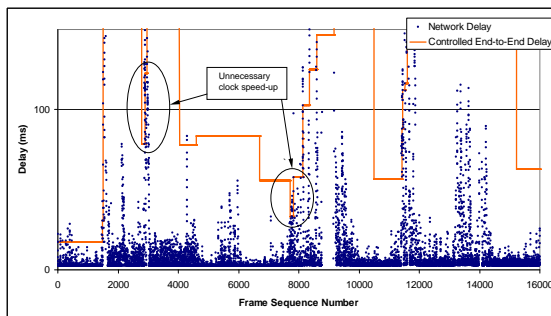


Figure 13. An example operation of the proposed scheme (without further modification) on the audio trace in a simulated mobile video conferencing session

We hence propose to add a status estimator to the virtual clock control. The arrival status of each MDU, including the playout state and the timing information, is fed into the status estimator to estimate the current state of the network delay process—either “stable” or “transient”. Each clock adjustment request will then be re-evaluated in accordance with the estimated state. For example, if the estimated state is currently “transient”, then a clock speed-up request will be ignored. The design of the status estimator is a work in progress.

6. Conclusions and Future Work

Due to the lack of QoS support on the current Internet, artificially introducing an equalization delay to compensate for the delay variation has become the only feasible solution to ensure an acceptable application-level QoS for networked RMMAs. In this paper, we investigated the objective performance measures which reflect the application level quality users perceive, and proposed an adaptive delay and synchro-

nization control scheme for networked RMMAs making use of these metrics in real-time. The proposed scheme adopts a distributed timing model and can be configured with two inter-stream synchronization control options based upon the application’s own feature. We took a wireline video conferencing application consisting of one audio and one video as an example, and illustrated the details of the scheme’s operation. A real conference session was emulated and a real network delay trace was recorded. The simulation results show that small synchronization phase distortions, low MDU loss percentages and low average end-to-end delays could be achieved simultaneously. We further investigated the characteristics of mobile wireless RMMAs, and discussed the feasibility of applying the proposed scheme. A preliminary version of the modified scheme was presented. Future work include the in-depth study of the delay and loss characteristics of mobile RMMAs running over WLAN, and the design of the transport channel status estimator.

References

- [1] B. Teitelbaum, Qbone architecture (v1.0), *Internet2 QoS Working Group Draft*, August 1999.
- [2] C. J. Sreenan, J.C. Chen, P. Agrawal, and B. Narendran, “Delay reduction techniques for playout buffering,” *IEEE Trans. Multimedia*, vol. 2, pp. 100-112, June 2000.
- [3] S. B. Moon, J. Kurose, and D. Towsley, “Packet audio playout delay adjustment: performance bounds and algorithms,” *ACM/Springer Multimedia Systems*, Jan. 1998.
- [4] Y. J. Liang, N. Frber, and B. Girod, “Adaptive playout scheduling using time-scale modification in packet voice communications,” in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2001.
- [5] J. Escobar, D. Deutsch, and C. Partridge, “Flow synchronization protocol,” in *Proc. IEEE INFOCOM’92*, pp. 1381-1387.
- [6] H. Liu and M. El Zarki, “Delay and Synchronization Control Middleware to support Real-Time Multimedia Services over Wireless PCS Networks,” *IEEE JSEC*, Vol. 17, No. 9, pp.1660-1672, 1999.
- [7] C. Liu, Y. Xie, M. Lee, and T. Saadawi, “Multipoint multimedia teleconference system with adaptive synchronization,” *IEEE JSEC*, vol. 14, pp. 1422-1435, Sept. 1996.
- [8] Y. Xie, C. Liu, M. J. Lee, and N. Saadawi, “Adaptive multimedia synchronization in a teleconference system”, *ACM/Springer Multimedia Systems*, vol. 7, pp. 326-337, 1999.
- [9] H. Liu and M. El Zarki, “A synchronization control scheme for real-time streaming multimedia applications,” in *Proc. 13th Packet Video Workshop*, France, 2003.
- [10] D. Miras, “A survey on network QoS needs of advanced internet application,” *Working Document of Internet2 QoS Working Group*, 2002.
- [11] R. Steinmetz and C. Engler, “Human perception of jitter and media synchronization,” *Internal Report #43.9310*, IBM European Networking Center, Heidelberg, Germany 1993.
- [12] D. kotz and K. Essien, “Analysis of a campus-wide wireless network,” *Mobicom 2002*, Atlanta, USA.

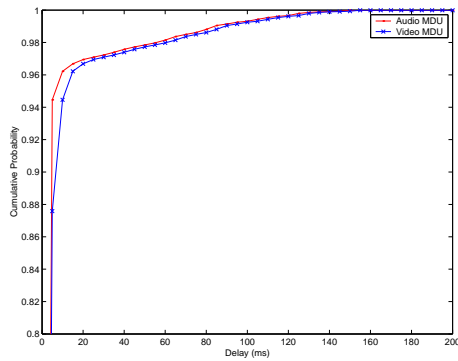


Figure 5. The cumulative probability distribution of the network delay trace

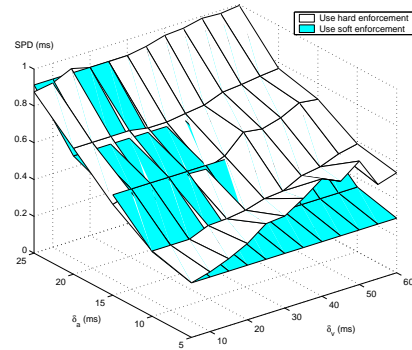


Figure 6. Intra-stream synchronization performance of the audio

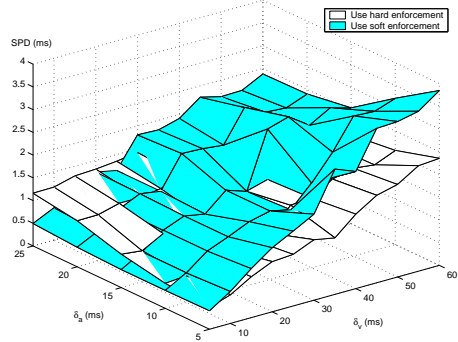


Figure 7. Intra-stream synchronization performance of the video

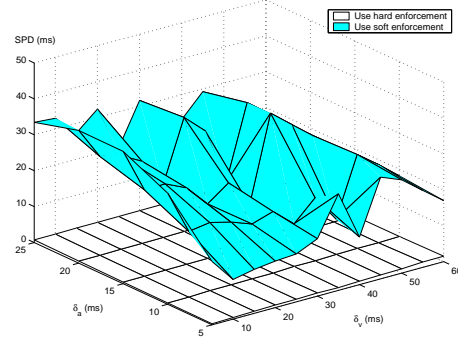


Figure 8. Inter-stream synchronization performance between the audio and the video

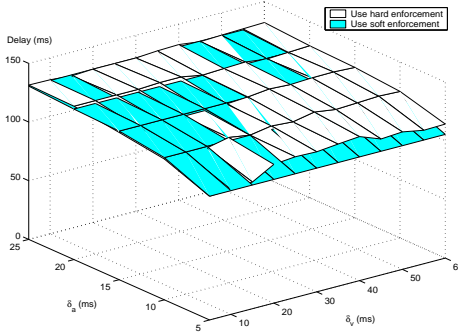


Figure 9. Average end-to-end delay of the audio

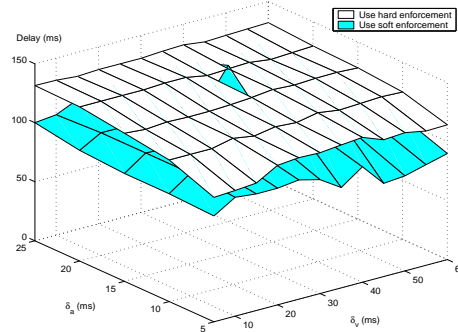


Figure 10. Average end-to-end delay of the video

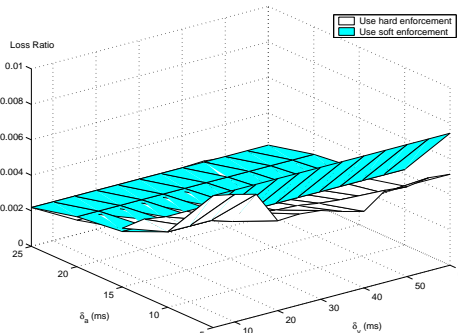


Figure 11. Resultant MDU loss ratio of the audio

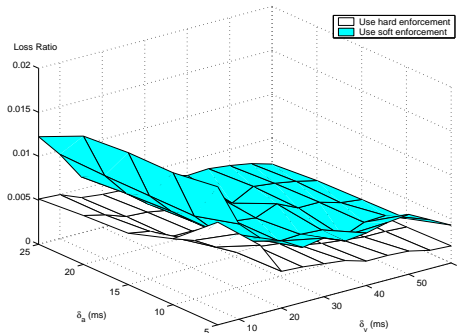


Figure 12. Resultant MDU loss ratio of the video