Live Media Synchronization Quality of a Retransmission–Based Error Recovery Scheme

Shuji Tasaka, Toshiro Nunome and Yutaka Ishibashi
Department of Electrical and Computer Engineering,
Nagoya Institute of Technology, Nagoya 466–8555, Japan

Abstract – This paper proposes a transport–layer error recovery scheme using retransmission for live audio–video streams transferred over QoS non-guaranteed networks. The scheme employs an enhanced version of the *virtual–time rendering* (*VTR*) media synchronization algorithm, which adjusts media rendering–time according to the network condition, and is referred to as *RVTR* (*Retransmission with VTR*) in this paper. By experiment, we assess the synchronization quality of both intra–stream and inter–stream in RVTR and four other schemes for audio–video transmission. A comparison of the experimental results indicates that joint application of retransmission control and the VTR synchronization control as in RVTR is effective in the improvement of media synchronization quality and that either control alone does not produce much improvement.

1 Introduction

The explosively increasing popularity of the Internet demands various multimedia applications with the transmission of continuous media such as audio and video. Streaming audio and video for the World Wide Web (WWW), videotelephone, videoconference, and live concert broadcasting are good examples of the applications. In realizing them, we have to solve many technical problems.

One of the most important issues is how to cope with packet loss due to congestion and packet corruption, both of which are inevitable in *Quality–of–Service* (*QoS*) non-guaranteed networks like the Internet. Solutions to the problem include automatic retransmission request (ARQ), forward error correction (FEC), adaptive control of audio/video output rates, and error concealment. Among these techniques, ARQ is a basic and efficient one. This paper studies ARQ at the transport layer and proposes a retransmission–based error recovery scheme for audio–video transmission.

Previously, retransmission–based error control was considered not suitable for interactive audio–visual applications, since lost packets could not be recovered by retransmission before their output deadlines. Recently, however, the effectiveness of the technique has been demonstrated, and many protocols of this type are being introduced [1].

Another important technical issue in networked multimedia applications is the preservation of the media temporal relations, namely, *media synchronization* [2]. It can be classified into *intra-stream synchronization* and *inter–stream synchronization*. The former keeps the continuity of a single stream; that is, it outputs *media units* (*MUs*), which is the information unit for media synchronization, at the destination at the same intervals as the generation ones at the source. The latter is synchronization among plural media streams; a typical example is synchronization between spoken voice and the movement of the speaker's lips (i.e., video).

The temporal relations can be disturbed by various causes, e.g., by the fluctuation of available CPU–processing power during the media capturing/reconstructing process and by delay jitters during the transmission through communication networks. It should be noted that retransmission of MUs aggravates network delay jitters. Therefore, when we use some retransmission–based protocol for error recovery, we should give considerable thought to the issue of media synchronization.

A variety of studies on retransmission–based error control for continuous media transmission have been reported in [3]–[7]. However, we cannot find any systematic research result from the media synchronization point of view. All the previous works deal with either video or audio, and also the works assess the media synchronization quality only in insufficient ways in spite of strong influence of retransmission on delay jitters.

The purpose of this paper is twofold. First, we propose an efficient error recovery scheme based on retransmission for both audio and video transferred over QoS non–guaranteed networks like the Internet. The scheme utilizes a media synchronization algorithm previously proposed by the authors [8], which is referred to as the *virtual–time rendering* (*VTR*) algorithm [9]. The VTR algorithm adjusts the MU rendering–time according to the network condition; this can provide extra time for retransmission[1]. We refer to the proposed error recovery scheme as *RVTR* (*Retransmission with Virtual–Time Rendering*). Secondly, we quantitatively assess the synchronization quality of both intra–stream and inter–stream in five audio–video transmission schemes including RVTR by experiment; thereby, we show the effectiveness of RVTR.

This paper considers only live media, though RVTR is applicable to stored media. We first enhance the VTR algorithm so as to accommodate itself to MU loss and retransmission. We then introduce a selective retransmission mechanism that conducts retransmissions within limited time intervals specified by the VTR algorithm to preserve the real–time property. We also develop a simple experimental system for the quality assessment. In the system, we use RTP/RTCP [10] on top of UDP; RVTR is implemented over RTP/RTCP. Also, JPEG video is adopted for real–time transmission. In the assessment, we examine the effects of network loads and round–trip time between the source and the destination on the quality, since they affect MU loss and the efficiency of the retransmission.

The rest of the paper is organized as follows. Section 2 specifies the VTR algorithm enhanced for MU loss and retransmission. Section 3 proposes RVTR. Section 4 illustrates a methodology for the assessment of the media synchronization quality, including quality measures, an experimental system and an experimental method. Section 5 presents experimental results to demonstrate the effectiveness of RVTR.

2 The VTR Algorithm Enhanced for MU Loss and Retransmission

We consider the transmission of an audio stream and the corresponding video stream from a source to a destination over a QoS non–guaranteed network. The audio and video are transmitted as two separate transport streams, conforming to the specification of RTP; this corresponds to the *multi–stream approach* [11]. A video frame is defined as a video MU, and an audio packet consisting of a constant number of audio samples as an audio MU.

In this paper, we suppose that a retransmission mechanism is built on RTP/UDP. The retransmission is tried so that each retransmitted MU can arrive at the destination before its output deadline which is specified by the media synchronization algorithm employed at the destination. Consequently, MUs can be lost owing to insufficient time for retransmission, and the order of MU's arrival at the destination can be different from that of their generation at the source.

[1]The technique of adaptive playback point with extended control time in [6] takes a similar approach; however, it produces the extra time by skipping P frames of MPEG, which is quite different from the adaptation mechanism of VTR.

The original VTR algorithm in [8] supposes a completely reliable transport protocol and so it assumes that every MU generated at the source is available in order at the *transport service access point* (*TSAP*) of the destination. Thus, in this paper, we are required to enhance the algorithm to accommodate itself to MU loss and retransmission.

This paper assumes that the *synchronization layer*, which offers the media synchronization services, is located on top of the transport layer as in [8] and that the transport layer here contains the retransmission mechanism.

Although the original VTR algorithm assumes only *locally available clocks* [8], the enhanced algorithm supposes *globally synchronized clocks*. This is because the current local times at the source and the destination should be identical for effective retransmission.

2.1 Definition of Notations

The VTR algorithm selects a media stream as the *master stream* and the others as *slave streams*, which are synchronized to the master. The algorithm exerts intra–stream synchronization control over both master and slave streams, while it performs inter–stream synchronization control only on slave streams after the intra-stream control. In this paper, we select audio (say *stream 1*) as the master and video (*stream 2*) as the slave since audio is more sensitive to intra–stream synchronization error than video.

For the description of the algorithm, we define the following notations for stream j ($j = 1$ and 2). First, we let $T_n^{(j)}$ ($n = 1, 2, \cdots$) denote the timestamp of the n-th MU in stream j, which is attached at the TSAP of the source, and define $\sigma_{n,m}^{(j)} \triangleq T_m^{(j)} - T_n^{(j)}$ ($n \le m; m = 1, 2, \cdots$). Secondly, we denote the *target output time* of the n-th MU by $t_n^{(j)}$; its exact definition will be given later. Also, let $\Delta_{\max}^{(j)}$ and $\Delta_{\min}^{(j)}$ be estimates of the maximum and minimum values, respectively, of the network delay in seconds for stream j, and define $J_{\max} \triangleq \max(\Delta_{\max}^{(1)} - \Delta_{\min}^{(1)}, \Delta_{\max}^{(2)} - \Delta_{\min}^{(2)})$. Then, $J_{\max}$ is an estimate of the maximum delay jitters; a rough estimate is sufficient for the algorithm to perform well. In addition, let $A_n^{(j)}$ and $D_n^{(j)}$ represent the arrival time of the n-th MU in stream j at the TSAP of the destination and the output time, respectively; this implies that we have the output waiting time[2] given by $\tau_n^{(j)} = D_n^{(j)} - A_n^{(j)}$, which enables us to absorb the network delay jitters.

In the intra–stream synchronization phase, the VTR algorithm calculates the *scheduled output time* $d_n^{(j)}$ by comparing $t_n^{(j)}$ and $A_n^{(j)}$. For the master stream (stream 1), we have $D_n^{(1)} = d_n^{(1)}$, while for the slave stream (stream 2) $D_n^{(2)}$ is determined by using $d_n^{(2)}$ in the inter-stream synchronization phase.

2.2 Intra–stream Synchronization Control

We make the enhancement only to the intra–stream synchronization; for inter–stream synchronization, we can use the same algorithm as that in [13].

[2]If $\Delta_{\max}^{(j)}$ and $\Delta_{\min}^{(j)}$ are the exact values of the maximum and minimum network delays, respectively, the intra–stream synchronization can be perfectly achieved by choosing the output waiting time so as to satisfy $\tau_1^{(j)} \ge \Delta_{\max}^{(j)} - \Delta_{\min}^{(j)}$ [12].

2.2.1 Output Time of First MU

We first determine the output time of the first MU in each stream, which is also used to obtain the time–origin for output control at the destination. The destination waits for the arrival of the first MU of audio and that of video and then selects the time of later arrival A_1, i.e., $A_1 \triangleq \max(A_1^{(1)}, A_1^{(2)})$. Defining $T_1 \triangleq \min(T_1^{(1)}, T_1^{(2)})$, we set the output time of the first MU in stream j ($j = 1$ and 2) to

$$D_1^{(j)} = A_1 + \tau_1^{(j)} \tag{1}$$

$$\tau_1^{(j)} = T_1^{(j)} - T_1 + J_{\max} \tag{2}$$

The time–origin for output control in each stream will be determined as the *ideal target output time* of the first MU in the stream.

2.2.2 Ideal Target Output Time

In order to preserve the real–time property of live media, we introduce the *maximum allowable delay* Δ_{al}. Then, we define the ideal target output time $x_n^{(j)}$ of the n-th MU in stream j as

$$x_1^{(j)} = \begin{cases} D_1^{(j)}, & \text{if } D_1^{(j)} - T_1^{(j)} \le \Delta_{\text{al}} \\ T_1^{(j)} + \Delta_{\text{al}}, & \text{otherwise} \end{cases} \tag{3}$$

$$x_n^{(j)} = x_1^{(j)} + \sigma_{1,n}^{(j)} \quad (n = 2, 3, \cdots) \tag{4}$$

Note that we have prepared two expressions for $x_1^{(j)}$ for the preservation of the real–time property. That is, the first MU may happen to suffer from a long delay; in this case, the time origin should be determined so as not to give bad effects on timely output of the succeeding MUs.

If the value of Δ_{al} is appropriate for a given network (i.e., if we have $x_1^{(j)} = D_1^{(j)}$) and if $J_{\max}$ is the exact value for the network[3], then we can set $D_n^{(j)} = x_n^{(j)}$ for all values of n. In reality, however, this is not the case; therefore, we cannot always output the MU at the ideal target output time. In order to cope with this situation, the VTR algorithm introduces the target output time $t_n^{(j)}$, which is obtained by adding/subtracting a delay (i.e., *slide time*) to/from the ideal target output time; this corresponds to the time expansion/contraction.

2.2.3 Target Output Time

Now, let us define the slide time of the n-th MU in stream j, which is denoted by $\Delta S_n^{(j)}$, as the difference between the *modified target output time* $t_n^{(j)*}$ and the original target output time $t_n^{(j)}$. Also, we denote the *total slide time* $S_n^{(j)}$ by

$$S_0^{(j)} = 0,\ \Delta S_1^{(j)} = 0 \tag{5}$$

$$S_n^{(j)} = S_{n-1}^{(j)} + \Delta S_n^{(j)} \ (n = 1, 2, \cdots) \tag{6}$$

Then, $t_n^{(j)}$ and $t_n^{(j)*}$ are given by

$$t_1^{(j)} = x_1^{(j)} \tag{7}$$

$$t_n^{(j)} = x_n^{(j)} + S_{n-1}^{(j)} \ (n = 2, 3, \cdots) \tag{8}$$

$$t_n^{(j)*} = t_n^{(j)} + \Delta S_n^{(j)} \ (n = 2, 3, \cdots) \tag{9}$$

In this paper, we assume that only the master stream (i.e., audio; $j = 1$) can modify the target output time for itself and accordingly the slave stream (video; $j = 2$) modifies it by the same amount at the same time. Therefore, we always have the identical total slide time for the master and slave streams.

[3]Note that even if we can get the exact value of $J_{\max}$, setting $J_{\max}$ to that value may destroy the real–time property.

2.2.4 Retransmission and Loss of MU

Before specifying how to set $\Delta S_n^{(1)}$, we consider the treatment of MU retransmission and loss in the algorithm. The retransmission is carried out when an MU is found absent at the destination; the procedure will be described in the next section.

Suppose that the n-th MU in stream j either arrives at TSAP or is found lost at the destination. In the implementation of the enhanced VTR algorithm, we regard the moment the n-th MU is read out from the receiver–buffer to exert media synchronization control as the arrival time $A_n^{(j)}$ of the MU at TSAP. The receiver–buffer has two queues: one for newly transmitted MUs and the other for retransmitted MUs. A received MU joins either queue according to its sort in the order of reception. To initiate the synchronization control of an MU for output, the destination first searches the queue of newly transmitted MUs and then the queue of retransmitted MUs if the MU is not in the former one. If neither queue has the MU, the destination waits for it until a *retransmission deadline*, at which time *retransmission timeout* occurs. Loss of the MU is identified by the retransmission timeout, which can be regarded as an arrival with $A_n^{(j)} = \infty$.

We now determine the retransmission deadline of the n-th MU in stream j, which is denoted by $W_n^{(j)}$. First, let the m-th MU ($m < n$) be the last MU output before the attempt to output the n-th MU. Also, let the MU with the smallest sequence number larger than n in the two queues be the k-th MU ($k > n$). Then, $W_n^{(j)}$ is set to an estimate of the target output time of the k-th MU: $W_n^{(j)} = t_m^{(j)} + \sigma_{m,k}^{(j)}$. If the destination receives the l-th MU ($n < l < k$) while waiting for the n-th MU, then $W_n^{(j)}$ is changed to the estimate for the l-th MU by setting $W_n^{(j)} = t_m^{(j)} + \sigma_{m,l}^{(j)}$.

2.2.5 Scheduled Output Time and Slide Time

We are now in a position to specify $\Delta S_n^{(1)}$. For that purpose, we first consider the scheduled output time $d_n^{(j)}$ for the n-th MU in stream j.

In order to determine $d_n^{(j)}$, the destination compares the arrival time $A_n^{(j)}$ and the target output time $t_n^{(j)}$. In this paper, we adopt the *quick recovery policy* [8], which gives

$$d_n^{(j)} = \begin{cases} t_n^{(j)*}, & \text{if } A_n^{(j)} \le t_n^{(j)} \\ A_n^{(j)}, & \text{otherwise} \end{cases} \tag{10}$$

The former case in the master stream has a possibility of advancing the target output time, namely, the time contraction (i.e., $\Delta S_n^{(1)} < 0$), while the latter has that of delaying it, the time expansion (i.e., $\Delta S_n^{(1)} > 0$). We first deal with the time expansion and then the time contraction.

(a) Time Expansion

We delay the target output time in two cases. One is the case where the arrival of the MU is too late but $A_n^{(1)} < \infty$, and the other is the case of MU loss, i.e., $A_n^{(1)} = \infty$. Both cases indicate network congestion. So, by delaying the target output time, we can produce extra time for possible retransmission of the succeeding MUs.

In the former case, we have $d_n^{(1)} = A_n^{(1)} >> t_n^{(1)}$. Therefore, if $d_n^{(1)} - t_n^{(1)}$ is larger than a threshold value[4] $T_{h2}^{(1)} > 0$,we modify the target output time by setting $\Delta S_n^{(1)} = d_n^{(1)} - t_n^{(1)}$.

[4]Since the target output time of the slave stream is not changed for itself in this paper, we set $T_{h2}^{(2)} = \infty$.

In the latter case, we have $D_n^{(1)} = d_n^{(1)} = \infty$, which means skipping of the MU, namely, loss of the MU. In this case, its target output time is delayed by $\Delta S_n^{(1)} = \max(\min(r, T_m^{(1)} - t_m^{(1)} + \Delta_{\mathsf{al}}), 0)$, where r is a positive constant, and the m-th MU is the last MU output before the attempt to output the n-th MU. At the same time, the target output time of the slave stream is changed by the same amount.

The meaning of the expression for $\Delta S_n^{(1)}$ is as follows. The destination tries to delay the target output time by r under constraint of the maximum allowable delay Δ_{al}; this would give $\Delta S_n^{(1)} = \max(\min(t_n^{(1)} + r, T_n^{(1)} + \Delta_{\mathsf{al}}) - t_n^{(1)}, 0) = \max(\min(r, T_n^{(1)} - t_n^{(1)} + \Delta_{\mathsf{al}}), 0)$. However, $T_n^{(1)} - t_n^{(1)}$ is unknown at the destination since the n-th MU is lost; therefore, we approximate it by $T_m^{(1)} - t_m^{(1)}$, which is known at the destination.

(b) Time Contraction

We can advance the target output time of the n-th MU if $A_n^{(1)} \le t_n^{(1)}$. However, we should do so when the network is not congested or when the target output time exceeds the limit specified by the maximum allowable delay Δ_{al}, i.e., when $t_n^{(1)} - T_n^{(1)} > \Delta_{\mathsf{al}}$. We check the former condition by confirming no MU loss for a certain period of time (say T_{noloss}) since the last loss. Thus, if either of the two conditions is satisfied when $A_n^{(1)} \le t_n^{(1)}$, we set $d_n^{(1)} = \max(t_n^{(1)} - r, x_n^{(1)})$; since $\Delta S_n^{(1)} = d_n^{(1)} - t_n^{(1)}$ in this case, we have $\Delta S_n^{(1)} = -\min(r, S_{n-1}^{(1)})$. At the same time, the target output time of the slave stream is advanced by the same amount.

3 Retransmission with Virtual–Time Rendering (RVTR)

We adopt a selective retransmission mechanism using negative acknowledgment on top of RTP/UDP. The retransmission is attempted during a limited time interval specified by the enhanced VTR algorithm. Below, we describe the control procedures at the destination and at the source.

3.1 Control at the Destination

We prepare two kinds of the control procedures: one for the first retransmission and the other for the second and subsequent retransmission.

First, let us explain the former. When the destination receives an MU, it memorizes the sequence number of the MU if the MU is newly transmitted, and it stores the MU into one of the two queues in the receiver–buffer. The failure in the first transmission of an MU is detected by discontinuity between the sequence numbers of newly transmitted MUs that have been received in succession. When the destination notices any failure in the first transmission, it sends a *request packet* to the source for the retransmission. The packet contains information the source needs for the retransmission.

To specify the information precisely, suppose that the destination has just received the m-th MU in stream j, which is newly transmitted, and stored it into the queue. Then, the destination checks the sequence number (say k) of the newly transmitted MU that had been received just before the m-th MU; note that k has been memorized. If k is different from $m-1$, the destination considers that the $(k+1)$-st through $(m-1)$-st MUs failed in the first transmission. We further assume that the h–th MU ($h \le k$) is the last MU output before the reception of the m-th MU. In addition, let $P_m^{(j)}$ denote the departure time of the m-th MU from the source, which is attached by the source. Then, the information consists of the

sequence numbers of the failed MUs (namely, $(k+1)$ through $(m-1)$), $t_h^{(j)} - T_h^{(j)}$ and $P_m^{(j)}$. Note that $t_h^{(j)} - T_h^{(j)}$ is the latest value of the time interval between the generation instant of an MU already output and its target output time.

Next, we describe the procedure for the second and subsequent retransmission. Each time the destination makes a retransmission request, it records the order of the request. When the destination receives a retransmitted MU, it refers to the request order recorded. If the reception order is different from the request order, then the destination considers that MUs corresponding to the gap in the request order failed in their retransmission, and it sends a request packet for the retransmission of the MUs in the same manner as the first procedure. However, no request is made for any MU whose sequence number is smaller than that of the MU which has been output just before the reception of the retransmitted MU.

3.2 Control at the Source

The source saves a copy of each newly transmitted MU in a buffer for a certain period of time[5] and searches the buffer on receiving a request packet from the destination.

Each time the source receives a request packet, it examines whether it has enough time to get each requested MU (say the n-th MU in stream j) arrived at the destination before a deadline. If the source judges the time enough, it carries out a retransmission. The deadline should be $W_n^{(j)}$ defined in the previous section. As a matter of fact, however, the source cannot know exactly which MUs have already arrived at the destination. That is, the source cannot calculate $W_n^{(j)}$. Therefore, we use an estimate of the target output time of the $(n+1)$-st MU instead. The estimate can be obtained from the information contained in the request packet as $T_{n+1}^{(j)} + t_h^{(j)} - T_h^{(j)}$. Note that the timestamp $T_{n+1}^{(j)}$ is known at the source. Also, let C_{time} and $R_{\min}$ denote the current time at the source and an estimate of the minimum round–trip time between the source and the destination, respectively. Then, the source judges the time enough if $T_{n+1}^{(j)} + t_h^{(j)} - T_h^{(j)} > C_{time} + R_{\min}/2$. Each time the source receives a request packet, it renews $R_{\min}$ in the following manner:

$$R_{\min} \leftarrow \min(C_{time} - P_m^{(j)}, R_{\min}).$$

We have selected the minimum value of the round–trip time since we should give as much opportunity as possible for retransmission to recover from MU loss.

Note that the destination could also adopt a policy of judging whether the retransmission request should be made or not. In this paper, however, we have not taken the policy for simplicity of the implementation and for small overhead of a request packet compared to the size of an MU.

4 Methodology for Quality Assessment

In order to demonstrate the effectiveness of RVTR, we evaluate the performance of RVTR and four other schemes and compare their media synchronization quality. The four schemes are: (1) Retransmission control using the VTR algorithm but without the change of the target output time, which is referred to as *RSync* here, (2) No–retransmission control using the VTR algorithm with the change of the target output time, which we denote here simply by *VTR*, (3) No–retransmission control using the VTR algorithm but without the change of the target output time, which we call *Sync*, and (4) Neither retransmission control nor media synchronization control; that is, no–control, which is represented by *NC*.

[5]It is three seconds in the implementation in this paper.

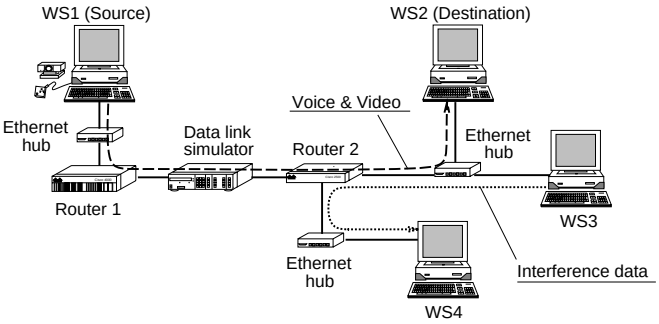

Figure 1: Configuration of the experimental system.

In the comparison, we examine detrimental effects of network loads and the round–trip time between the source and the destination on the quality. This is because MU loss occurs owing to heavy network loads and the success probability of the retransmission depends on the round–trip time as well as the network load.

In this section, we first define measures for quality assessment. We then show an experimental system developed for the quality assessment and also illustrate a method for the experiment.

4.1 Measures for Quality Assessment

There is no authorized measures for quantitative assessment of media synchronization quality. Therefore, we employ measures the authors have introduced and used in their previous studies on media synchronization [9], [11], [13].

For the quality assessment of intra–stream synchronization for audio or video, we first evaluate the *coefficient of variation of output interval*, which represents the smoothness of output of a media stream. In addition, we use the *MU loss rate*, which is the ratio of the number of MUs lost to the total number of MUs generated, to investigate the efficiency of retransmission.

For the inter–stream synchronization quality, we calculate the *mean square error*[6], which is defined as the average square of the difference between the output time of each slave MU and its *derived output time*. The derived output time of each slave MU is defined as the output time of the corresponding master MU plus the difference between the timestamps of the two MUs [8].

Finally, the *average MU delay* is a key measure for live media; it is the average time in seconds from the moment an MU is generated until the instant the MU is output.

4.2 Experimental System

Figure 1 illustrates the configuration of the experimental system. It comprises four workstations (WS1 through WS4), three 10Base–T Ethernet–hubs, two routers (Router 1 and Router 2) and a data link simulator.

WS1 and WS2 are SUN Ultra 2 workstations (200 MHz) each of which has a main memory of 128 Mbytes and a JPEG video board (PowerVideo from Parallax Graphics, Inc.) for the NTSC–standard. WS3 and WS4 are SUN Ultra 1 workstations (143 MHz) each with a 64 Mbyte main memory. WS1 runs Solaris 2.6 with OpenWindow 3.6, while WS2, WS3 and WS4 run Solaris 2.5.1 with OpenWindow 3.5.1.

Routers 1 and 2 are Cisco System's 4700–M and 2514, respectively. They are connected to each other by a V.35 serial line through the data link simulator (ADTECH SX/12). The transmission rate of the serial line is set to 4 Mbps in our experiment.

The data link simulator can simulate various transmission error patterns and propagation delay values. In our experiment,

[6]If at least either a slave MU or the corresponding master MU is lost, the pair of MUs are excluded in the calculation.

Table 1: Specifications of voice and video.

item	voice	video
coding scheme	ITU–T G.711 μ-law	JPEG
image size [pixels]	—	320 × 240
average MU size [bytes]	400	3527
original average MU rate [MU/s]	20.0	
original average inter–MU time [ms]	50.0	
original average bit rate [kbps]	64.0	563.0
measurement time [s]	243.3	

we utilize the capability of producing a propagation delay which can take any value in the range from 0 to 2 seconds by ms; thereby, we can set various values of the round–trip time.

4.3 Method of the Experiment

Our experiment in this paper focuses on lip synchronization, and we use a lady's voice and her head view video as the audio stream and video stream, respectively. Table 1 shows the specifications of voice and video.

In the experiment, WS1 and WS2 are used as the source of the voice and video streams and its destination, respectively. WS1 inputs the voice and video streams from a video cassette recorder in order to generate the media traffic of the same amount in each experimental run. Using RTP/RTCP, WS1 transfers the voice and video as two separate transport streams to WS2. An RTCP packet is transmitted at intervals of 5 seconds; however, no function of RTCP packets is utilized in this paper. As in [11], we can perform dynamic resolution control of video by utilizing the value of some field (e.g., "fraction lost") of sender report (SR)/receiver report (RR) RTCP packets; we have already implemented and evaluated this scheme. Experimental results of this subject will be covered in another paper.

WS3 and WS4 are used to generate a traffic flow of interference with which MU loss in the voice and video streams occur. WS3 sends fixed–size data messages of 1472 bytes each to WS4 under the UDP protocol at exponentially distributed intervals. The amount of the interference traffic is adjusted by changing the average of the interval.

The parameter values in the VTR algorithm are set as follows[7]: $T_{h2}^{(1)}$=80 ms, $T_{h3}^{(1,2)}$=80 ms, $T_{h4}^{(1,2)}$=160 ms, r=20 ms, and T_{noloss}=5 seconds. We have selected the values of $T_{h3}^{(1,2)}$ and $T_{h4}^{(1,2)}$, referring to results of subjective assessment of a lip synchronization experiment conducted by Steinmetz [14]: He reports that a time difference of between -80 ms and $+80$ ms leads to synchronization of high quality, whereas a time difference beyond $\pm$160 ms corresponds to asynchrony. The values of $T_{h2}^{(1)}$, r and T_{noloss} were determined after we had tried several values for each.

Furthermore, we set $J_{\max}$=100 ms and Δ_{al}=300 ms. We roughly chose $J_{\max}$=100 ms, considering the current experimental environment. The choice of Δ_{al}=300 ms was made on the basis of ITU–T Recommendation G.114 [15], which regards delays of 150 to 400 ms as acceptable provided that Administrations are aware of the transmission time impact on the transmission quality of user applications.

5 Experimental Results

We first examine detrimental effects of the interference data load (namely, effects of MU loss) on the quality when the propagation delay of the data link simulator, which we call *additional delay* here, is set to 0; this corresponds to a LAN environment. We then study the effects of the additional delay on the quality when the data load is kept constant.

[7] $T_{h3}^{(1,2)}$ and $T_{h4}^{(1,2)}$ are threshold values for inter–stream synchronization.

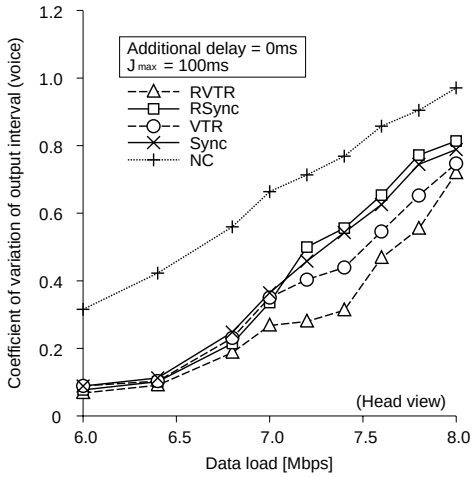

Figure 2: Coefficient of variation of output interval versus data load for voice.

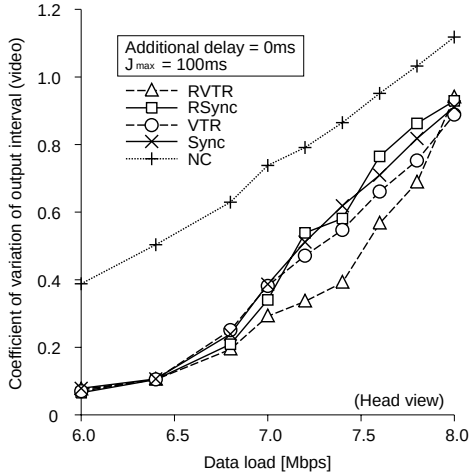

Figure 3: Coefficient of variation of output interval versus data load for video.

5.1 Effect of Data Load with No Additional Delay

We first assess the intra–stream synchronization quality. Figures 2 and 3 show the coefficient of variation of output interval for voice and video, respectively, as a function of the data load for the five schemes. In these figures we observe that for both voice and video RVTR provides the minimum coefficient of variation and VTR the second minimum for almost all the data loads here, while NC gives the largest. Thus, we see that media synchronization control without the change of the target output time is not so effective in improving the smoothness of output and that joint application of retransmission and media synchronization control with the change of the target output time is effective.

We next examine the MU loss rate. Figure 4 displays this measure for video; the result of voice shows the same tendency. We then find that RVTR achieves the best performance and

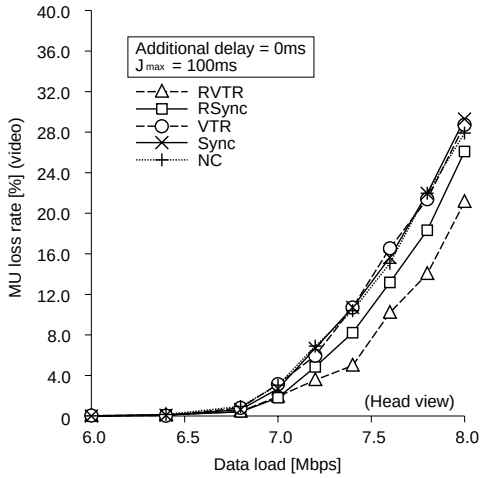

Figure 4: MU loss rate of video versus data load.

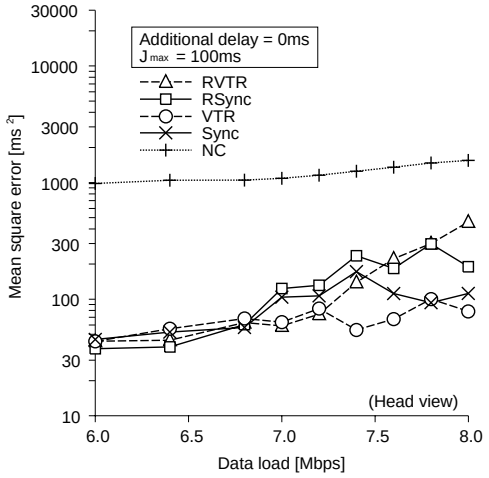

Figure 5: Mean square error of inter–stream synchronization versus data load.

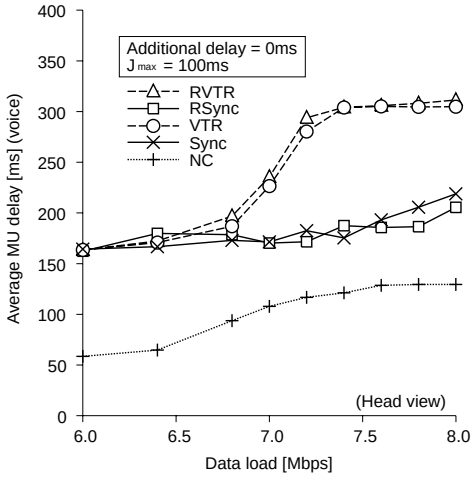

Figure 6: Average MU delay of voice versus data load.

RSync the second best and that there is no substantial difference among the others. Thus, we can confirm the effectiveness of the retransmission control.

Figure 5 plots the mean square error of inter–stream synchronization versus data load. We notice that although NC has much larger values than the four schemes with synchronization control, the values for all the schemes are much smaller than 6400 ms^2 (=$80^2 ms^2$), which is a threshold of high inter–stream synchronization quality reported by Steinmetz [14]. High quality of inter–stream synchronization even with no control is a characteristic of live media [11].

Figure 6 presents the average MU delay of voice. The average MU delay of video is approximately the same as that of voice owing to high quality of inter–stream synchronization. We find that both RVTR and VTR produce larger MU delays than the others at heavy data loads because of the modification of the target output time. However, it should be noted that the MU delays of RVTR and VTR are restricted approximately to Δ_{al}=300 ms; this implies that the upper bound of the average MU delay is controllable with the VTR algorithm.

5.2 Effect of Additional Delay at a Constant Data Load

It is clear that the retransmission control is not suitable for environments of long round–trip time. Therefore, we varied the additional delay from 0 to 100 ms in the experiment. We also kept the data load at 7.4 Mbps in the figures to be shown below. Since the results of video are very similar to the corresponding results of voice, we show the results of voice only.

Figure 7 plots the coefficient of variation of output interval for voice as a function of the additional delay. The figure indicates that as the additional delay increases, the coefficient of variation for RVTR also increases, while those for the others hardly do. We see that RVTR has the best quality for additional delays up to about 40 ms.

Figure 8 depicts the MU loss rate of voice. In the figure, we observe that RVTR and RSync are affected by increase in the additional delay, whereas the others are hardly affected. This is, of course, due to the difference in capability of retransmission control. The figure indicates that RVTR can improve the performance for additional delays up to about 60 ms.

We present the mean square error of inter–stream synchronization in Fig. 9, from which we see that the additional delay hardly affects the quality in all the schemes.

Figure 10 displays the average MU delay of voice. In this figure, we observe that for all the additional delays shown here, the average MU delays of both RVTR and VTR take an approximately constant value, namely, Δ_{al}=300 ms, while those of the other schemes linearly grow with increase in the additional delay.

6 Conclusions

We proposed RVTR, which exerts retransmission control with the VTR media synchronization algorithm. By experiment, we assessed the media synchronization quality of five schemes including RVTR for various network loads. We then saw that joint application of the retransmission control and the VTR algorithm is effective in the improvement of media synchronization quality and that either control alone does not produce much improvement. We also noticed that although the VTR algorithm incurs increment of the average MU delay, its upper bound is controllable by setting Δ_{al} to a desirable value.

We further examined the effects of the round–trip time on the quality and found that the quality improvement obtained by RVTR decreases as the round–trip time increases.

As the next step of our research, we need to investigate how the threshold and parameters in RVTR affect the media synchronization quality under a wide variety of conditions and thereby establish a method for setting their appropriate values in practical situations. Our future work also includes extensions

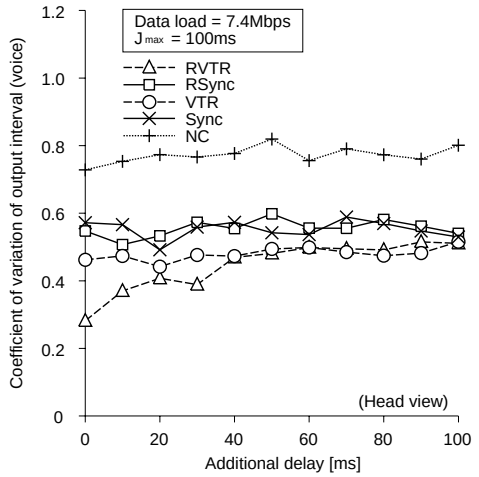

Figure 7: Coefficient of variation of output interval versus additional delay for voice.

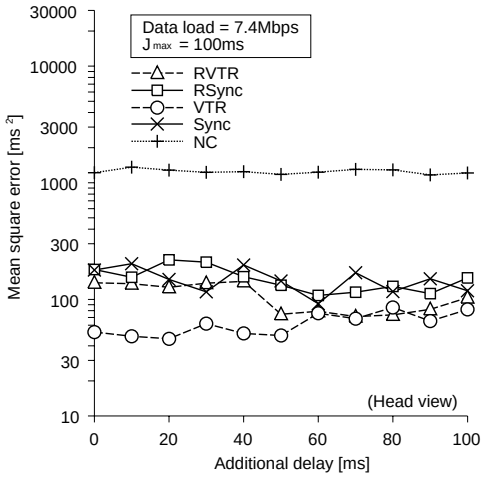

Figure 9: Mean square error of inter–stream synchronization versus additional delay.

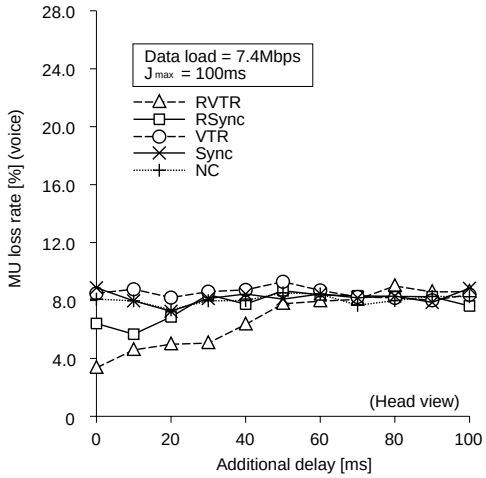

Figure 8: MU loss rate of voice versus additional delay.

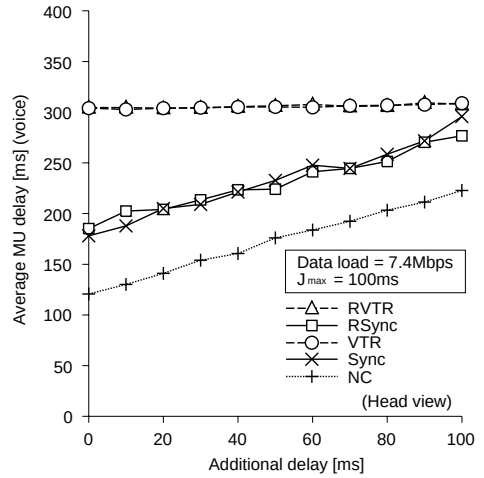

Figure 10: Average MU delay of voice versus additional delay.

of RVTR to multicast communications and interframe coded video.

References

[1] G. Carle and E. W. Biersack, "Survey of error recovery techniques for IP–based audio–visual multicast applications,"*IEEE Network*, vol. 11, no. 6, pp.24–36, Nov./Dec. 1997.

[2] G. Blakowski and R. Steinmetz, "A media synchronization survey: Reference model, specification, and case studies," *IEEE J. Sel. Areas in Commun.*, vol. 14, no. 1, pp. 5–35, Jan. 1996.

[3] G. Ramamurthy and D. Raychaudhuri, "Performance of packet video with combined error recovery and concealment," in *Proc. INFOCOM'95*, Apr. 1995, pp.753–761.

[4] B. J. Dempsey, J. Liebeherr and A. C. Weaver, "On retransmission–based error control for continuous media traffic in packet–switching networks," *Comp. Networks and ISDN Sys.*, vol. 28, no. 5, pp.719–736, Mar. 1996.

[5] S. Pejhan, M. Schwartz and D. Anastassiou, "Error control using retransmission schemes in multicast transport protocols for real–time media,"*IEEE/ACM Trans. Networking*, vol. 4, no. 3, pp.413–427, June 1996.

[6] X. Li, S. Paul, P. Pancha and M. Ammar, "Layered video multicast with retransmission (LVMR): Evaluation of error recovery schemes," in *Proc. NOSSDAV'97*, May 1997.

[7] T. Hasegawa, T. Hasegawa, T. Kato and K. Suzuki, "Applying reliable data transfer protocol to real time video retrieval system," *IEICE Trans. Commun.*, vol. E80–B, no. 10, pp. 1482–1492, Oct. 1997.

[8] Y. Ishibashi and S. Tasaka, "A synchronization mechanism for continuous media in multimedia communications," in *Proc. INFOCOM'95*, Apr. 1995, pp.1010–1019.

[9] S. Tasaka, H. Nakanishi and Y. Ishibashi, "Dynamic resolution control and media synchronization of MPEG in wireless LANs," in *Conf. Rec. GLOBECOM'97*, Nov. 1997, pp.138–144.

[10] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson,"RTP: A transport protocol for real–time applications," RFC 1889, Feb. 1996.

[11] S. Tasaka and Y. Ishibashi, "Single–stream versus multi–stream for live media synchronization", in *Conf. Rec. ICC'98*, June 1998, pp. 470–476.

[12] H. Santoso, L. Dairaine, S. Fdida, and E. Horlait, "Preserving temporal signature: A way to convey time constrained flows," in *Conf. Rec. GLOBECOM'93*, Dec. 1993, pp. 872–876.

[13] Y. Ishibashi, S. Tasaka and E. Minami, "Performance measurement of a stored media synchronization mechanism: Quick recovery scheme," in *Conf. Rec. GLOBECOM'95*, Nov. 1995, pp.811–817.

[14] R. Steinmetz, "Human perception of jitter and media synchronization," *IEEE J. Sel. Areas in Commun.*, vol. 14, no. 1, pp. 61–72, Jan. 1996.

[15] ITU–T Recommendation G.114: "Transmission systems and media, general characteristics of international telephone connections and international telephone circuits: One–way transmission time," Feb. 1996.