Hindawi Publishing Corporation
Advances in Multimedia
Volume 2011, Article ID 914062, 13 pages
doi:10.1155/2011/914062

Research Article

Real-Time Adaptive Content-Based Synchronization of Multimedia Streams

Imad H. Elhajj,[1] Nadine Bou Dargham,[1] Ning Xi,[2] and Yunyi Jia[2]

[1] *Department of Electrical and Computer Engineering, American University of Beirut, P.O. Box 11-0236, Riad El Solh, Beirut 1107 2020, Lebanon*
[2] *Department of Electrical and Computer Engineering, Michigan State University, East lansing, MI 48824-1226, USA*

Correspondence should be addressed to Imad H. Elhajj, ie05@aub.edu.lb

Received 21 January 2011; Revised 29 April 2011; Accepted 27 June 2011

Academic Editor: Chong Wah Ngo

Traditional synchronization schemes of multimedia applications are based on temporal relationships between inter- and intra-streams. These schemes do not provide good synchronization in the presence of random delay. As a solution, this paper proposes an adaptive content-based synchronization scheme that synchronizes multimedia streams by accounting for content in addition to time. This approach to synchronization is based on the fact that having two streams sampled close in time does not always imply that these streams are close in content. The proposed scheme primary contribution is the synchronization of audio and video streams based on content. The secondary contribution is adapting the frame rate based on content decisions. Test-ing adaptive content-based and adaptive time-based synchronization algorithms remotely between the American University of Beirut and Michigan State University showed that the proposed method outperforms the traditional synchronization method. Objective and subjective assessment of the received video and audio quality demonstrated that the content-based scheme provides better synchronization and overall quality of multimedia streams. Although demonstrated using a video conference application, the method can be applied to any multimedia streams including nontraditional ones referred to as supermedia like control signals, haptic, and other sensory measurements. In addition, the method can be applied to synchronize more than two streams simultaneously.

1. Introduction

Three types of applications are recognized over the Internet: asynchronous, synchronous, and interactive synchronous. Asynchronous applications do not involve simultaneous transfer of media streams, like file transfer and web browsing. Such applications do not require strict timing from the network. In synchronous applications, such as video clip viewing, multiple media streams (audio and video) are to be transferred simultaneously. In this type of applications, synchronous display of media streams is required. However, since no interactivity is involved, these applications do not enforce strict delay requirements on the network. In this case, applications can buffer data before starting to render audio and video to mask the network delays.

As for the interactive synchronous applications, they include real-time applications with an interactive nature such as video conferencing and networked gaming. They also include nontraditional applications using "supermedia" such as remote surgery and teleoperation [1, 2]. Due to their interactive nature these applications can only accept limited buffering of data before rendering the media. Therefore, their quality suffers considerably when operated over the Internet, and their performance is significantly degraded when used over long distances.

One reason for quality degradation is the lack of synchronization between different media/supermedia streams [1, 3]. To illustrate that, consider the digital audio and video case. The audio samples and the corresponding video frames should be played at specific time intervals to deliver a correct presentation. The conservation of these temporal relationships is known as time-based synchronization or lip synching [3, 4]. The main challenge is to synchronize the lip motion observed with the sound being heard, because the Internet

exhibits delays that are random in nature. In fact, the main challenge is not the presence of delay but the delay variation, called jitter. So each multimedia stream communicated via the Internet will face a different delay, and this would cause a desynchronization of the media streams at the receiving side. In their well-known Nature article, McGurk and MacDonald stated in 1976 that hearing a sound and viewing the lips tracing another sound would result in perceiving a third different sound [5, 6]. Authors in [1] discussed the importance of synchronization for supermedia streams in general. Therefore, methods should be applied to resynchronize these streams, or else a significant degradation of the perceived quality will be encountered.

Synchronization schemes are traditionally based on *temporal relationships* of inter- and intrastreams. Yet, these schemes do not always provide the best solution. Synchronizing multimedia streams based on temporal relationships alone does not provide the best synchronization quality in the presence of random delay conditions. As a solution, and to improve the quality, *content-based* synchronization is proposed. This approach does not totally ignore the temporal relationships between the streams. However, instead of synchronizing the streams based on their generation time only, they synchronize them based on both when they are generated and what they contain. This approach to synchronization is based on the fact that having two streams sampled close in time does not always mean that these streams are close in content and vice versa.

To illustrate this point, consider Figures 1 and 2 that represent real-time data collected from a video conferencing session. Figure 1 represents the audio rate of change versus time, and Figure 2 shows the video rate of change versus time. Two cases are shown here. In the first case, labeled (1), the person is talking but without moving; therefore, the rate of change of audio is high whereas the rate of change of video is very low. In the second case, labeled (2), the mobility of the person increased while talking; therefore, the rate of change of both audio and video is significant.

In the first case, the audio stream arrives far ahead the video stream with respect to time. The traditional synchronization schemes based on temporal relationships only would drop the video frame as it is considered relatively "old." But according to Figure 2, the video rate of change is very low. Therefore, from a content perspective, it is carrying "fresh" information and should not be discarded. The second case shows the opposite scenario, where the audio and video frames are close in time, so the time-based synchronization schemes would render the video frame, whereas Figure 2 shows that the video has exhibited significant changes and therefore, according to content, it is carrying "old" data and should be discarded. Following the time-based synchronization scheme would result in an inaccurate outcome since it does not take the content change into consideration.

In addition to correct synchronization, it is important to note that the video frame rate has a significant influence on the quality of multimedia applications. Clearly, the higher the video frame rate the better the quality obtained. In both time-based and content-based algorithms, dropped video frames consume resources without increasing the user perceived frame rate. Therefore, it is better to avoid transmitting the video frames that are likely to be dropped. This could be accomplished using an *adaptive frame rate*. Therefore, there is a need to apply real-time adaptive content-based synchronization of multimedia streams for the aim of providing better synchronization, hence delivering better quality. The method presented in this paper can be applied to any number of concurrent streams and to nontraditional media referred to as supermedia.

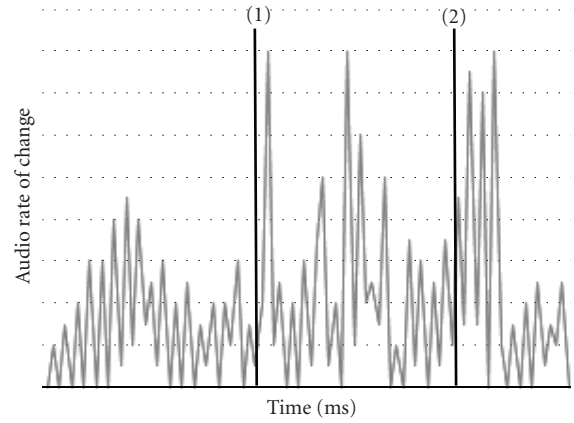

Figure 1: Audio rate of change versus time.

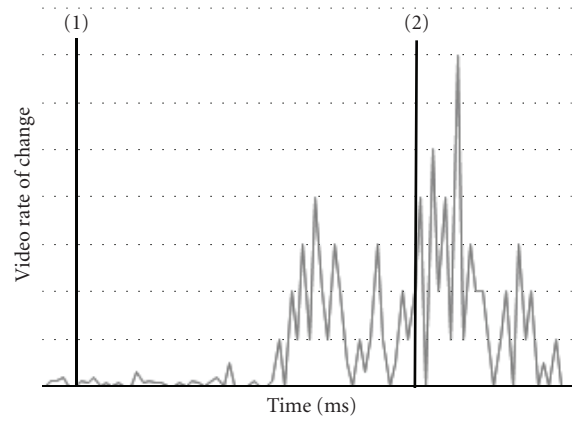

Figure 2: Video rate of change versus time.

The paper is organized as follows. Section 2 presents a literature review of the different related work. In Section 3, content-based synchronization and the adaptive scheme are detailed and analyzed. Section 4 presents the system implementation and explains the parameters used to assess the algorithm. Section 5 studies and compares the results obtained from applying the content-based synchronization algorithm to these obtained from the time-based scheme. Finally, Section 6 concludes with a summary of the paper and recommendations for future work.

2. Literature Review

The conducted literature review covered three categories: time-based audio and video synchronization, content-based synchronization, and content-based retrieval.

Most of multimedia synchronization algorithms are based on the temporal relationships between audio and video streams. One of the methodologies proposed is the timestamp exchange between master and slave clock, where all the clocks in the residential Ethernet must be synchronized by one clock called "Grand Master." In this method, the slave clock will synchronize to a master which will synchronize to its own master. This synchronization chain continues until reaching the Grand Master [7]. A second method proposes that the generating time of each object is time-stamped by the local Sample Clock (SC), and a set of PlayOut Clocks (POCs)—one for each incoming stream—is provided at the sink. In this method, an algorithm is employed to calibrate the PlayOut Clocks (POCs), which manages the presentations of multimedia streams [8]. The limitation of these synchronization mechanisms based on temporal relationships is that they do not provide the appropriate synchronization under different delay and jitter conditions. Therefore, time-based synchronization is not the best choice for interactive applications over the Internet.

An alternative way to provide multimedia synchronization is the content-based synchronization. Only a few content-based methodologies were proposed in the literature. Information-hiding-based synchronization embeds audio data in their corresponding video frame using high bit rate information hiding techniques, and the audio data is extracted from the video frame and played with it at the receiver [9]. Even though this method guarantees a reliable synchronization between audio and video, it requires significant processing on the sender side (embedding audio data within video frames) and the receiver side (extracting audio data from video frames). Therefore, this method can be used for synchronous applications, but it is not suitable for real-time interactive applications due to the strict timing constraints required. A different content-based multimedia synchronization scheme was proposed in which the media stream is hierarchically composed of smaller objects, logically structured based on their content, and the synchronization is achieved by deriving temporal relations among logical units of media object [10]. In this method, audio/video synchronization exhibits a significant time and involves intensive dispensation, so it is not feasible for interactive applications. The concept of content-based synchronization for supermedia streams was introduced in [1, 2]. This technique was primarily targeted for teleoperation applications and was discussed in the context of control. To the best of our knowledge, the work presented in this paper is the first use of the concept of content for synchronization in real-time interactive multimedia applications.

Other methods, not directly addressing synchronization but related to content-based processing of multimedia, are content-based retrieval which uses the content of multimedia to represent and index the data and content-aware play-out of streamed video. The contents of the media in the database are extracted and described by multidimensional feature vector (descriptors) that constitutes a feature dataset. To retrieve the desired data, users submit query examples to the retrieval system. The system then represents these examples with feature vectors. The distances (i.e., similarities) between the feature vectors of the query example and those of the media in the feature dataset are then computed and ranked. Retrieval is conducted by applying an indexing scheme to provide an efficient way to search the media database. Finally, the system ranks the search results and then returns the top search results which are the most similar to the query examples [11]. This method cannot be used via the Internet because it requires bounding and limiting the end-to-end delay and/or the jitter which is practically impossible. There are also schemes that can be used for fast similarity-based indexing and retrieval of both image and video databases in distributed environments. Such schemes are based on detecting shot changes using the temporal distribution of MPEG macroblocks, using algorithms to return images with identical DC coefficients, using relationships formed between images, text, and number of times retrieved to return a searched image [12–14]. In addition, there are papers proposing models that identify requirements for timeliness and accuracy of real-time content-based analysis [15]. Since many solutions require additional processing, classifying media objects using XML has been proposed as alternative lightweight solutions [16, 17]. However, all of the methods presented above require prior classification and/or significant processing. For these reasons, they are not suitable for synchronization of real-time interactive applications. Regarding content-aware play-out of streamed video, there is a significant amount of work. Adaptive content-aware play-out of video streamed over wireless network was introduced in [18]. Content-aware dropping of video packets during congestion was presented in [19]. Methods discussed in this area are related to managing one stream and not the synchronization among different streams.

3. Synchronization and Adaptation Algorithms

3.1. Content-Based Synchronization Algorithm. The developed content-based synchronization algorithm is demonstrated using a video conferencing application with both audio and video streams. However, this does not limit the algorithm from being adapted to other types or combinations of media streams. The reason audio and video were chosen is because they are more commonly used and their synchronization and quality can be intuitively evaluated. The flow chart of the content-based synchronization algorithm is presented in Figure 3.

From a design perspective, it is worth noting that video conference applications typically apply buffering (fixed or adaptive) to the arriving media segments in the order of tens of milliseconds. This is typically employed to allow the larger video frames to arrive with their corresponding audio segments and to reduce the effect of jitter. Irrespective of the buffering scheme used, the proposed technique is applied to media segments being dequeued from the buffer after being delayed.

The developed content-based synchronization algorithm takes decisions by accounting for content in addition to time. In the algorithm the audio stream is assumed to be the master stream. This means that audio would be played to the user upon arrival as long as it is in order, regardless of the state

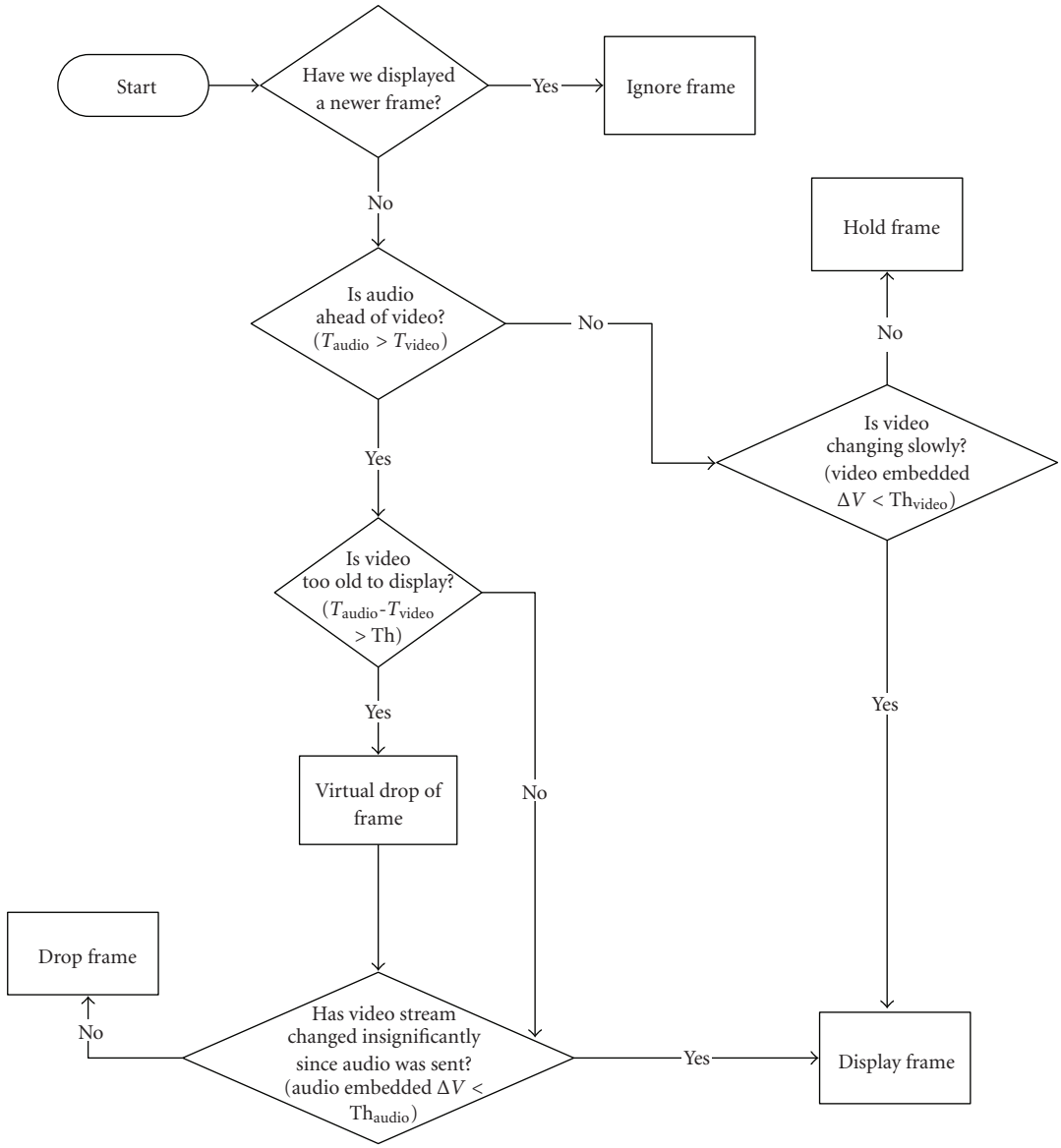

Figure 3: Flow chart of content-based synchronization algorithm for video conferencing application.

of the video. Audio was chosen as the master stream for two reasons. The first reason is because audio samples are typically smaller than video frames; thus, audio samples usually arrive before their video counterparts. Secondly, humans are more sensitive to degradation in audio quality than in video [20]. Therefore, the audio stream is played as long as the audio sample is newer than a previously played sample. As for video frames, decisions are based on both content and time.

First, the algorithm checks if a newer video frame had been rendered and if so the frame is immediately dropped as there is no reason to step back within the video stream. Otherwise, the timestamp of the video is compared with that of the audio to be played in parallel with to determine which one is ahead. Let T_{video} be the timestamp of the video frame being considered for rendering, and let T_{audio} be the timestamp of the audio segment being played in parallel. Typically, since video frames are larger than audio samples, audio samples arrive ahead of video frames ($T_{audio} > T_{video}$), but for completeness the case of video frames being ahead of their corresponding audio samples ($T_{video} > T_{audio}$) has to be considered. If audio is ahead of video ($T_{audio} > T_{video}$), the algorithm checks how far ahead it is by calculating the difference in their timestamps, $T_{audio} - T_{video}$. Here, two cases are present: in the first case, if $T_{audio} - T_{video} \leq$ Th, where Th is a configured threshold indicating the video frame is unacceptably old, the video average rate of change since this frame was sampled is examined. Clearly this requires knowledge about frames that have not arrived yet. That is why each audio sample is required to carry the average video rate of change ΔV (defined in Section 3.1.2) at the time it was sampled, "Audio Embedded ΔV." In the case where $T_{audio} > T_{video}$, Audio

Embedded ΔV gives insight into the future state of the video stream. If this value is lower than a threshold, Th_{audio}, it will be known that the video was changing slowly between when these audio and video were sampled. Therefore, the video frame being considered still has "fresh" information and should be displayed to the user. Otherwise, the frame is considered to have "old" information and it is dropped.

In the second case, where the time difference is too high, $T_{\text{audio}} - T_{\text{video}} > \text{Th}$, the video frame being considered is significantly older than the audio sample. Typically, the frame is dropped without making further decisions (ignoring its content). But what if the rate of change of video is very low and, therefore, this video frame is still holding "fresh" data, even when $T_{\text{audio}} - T_{\text{video}}$ is greater than the threshold? Dropping the video frame instead of rendering it in this case will degrade the synchronization quality. For this reason, the proposed algorithm suggests not dropping the video frame before checking its content, even when the time difference between audio and video is significantly high. So in case of a large difference in time between audio and video frames, the algorithm drops the video frame *virtually* and then checks for ΔV. Virtually dropping a video frame means counting the frame as dropped, that is, increasing the percentage of dropped frames, but without dropping it in reality. This will affect the behavior of the adaptive algorithm in controlling the frame rates as will be seen later. The decision on dropping the video frame or rendering it will not be taken until studying the video rate of change. The video frame will not be dropped in reality if the difference in content is acceptable; it is only dropped if $\Delta V > \text{Th}_{\text{audio}}$. A slightly different decision-making process must be followed in the less likely event that video arrives ahead of audio ($T_{\text{video}} > T_{\text{audio}}$). Since the content of this frame shows something that will happen in the future of the stream, it cannot be dropped. Dropping it could possibly allow an audio sample to be played without a corresponding video frame, thus decreasing synchronization quality. Since the audio stream is the master stream, there is no need to check if it is too old to render as it is played to the user regardless of the state of the video stream. Instead, the algorithm looks at the content of the video stream and how fast it is changing. In the previous case, the ΔV considered was the one embedded in the audio packet since the audio was arriving ahead of video. In this case, the algorithm cannot use this value since the video is ahead of the audio being rendered. The ΔV within the audio packets provides information about the state of the video stream in the past, which cannot be used to make synchronization decisions. Instead the algorithm considers the ΔV embedded in the video frame itself, "Video Embedded ΔV." If the stream had changed faster than a threshold, Th_{video}, the algorithm holds the frame. Holding a frame would keep it in memory until an audio packet closer to it in time arrives at which point the algorithm makes a decision regarding the frame. Otherwise, the stream had not changed significantly and it is appropriate to render the frame to the user.

3.1.1. Time Threshold "Th". As discussed earlier, traditional synchronization methods only consider the temporal difference between samples being rendered to the user. For example, given an audio sample and a video frame with timestamps T_{audio} and T_{video}, respectively, the frames would be rendered together if the difference $T_{\text{audio}} - T_{\text{video}}$ is lower than a certain threshold. For the best synchronization quality possible with such an algorithm, the threshold should be less than the level at which humans detect frames as being out of sync. This level was found to be around 130 ms [20]. While this does drop frames that are too old, it does so indiscriminately. It ignores the content or behavior of the stream.

For this reason, in content-based synchronization, the threshold for frames to be marked as too old to display, Th, is left higher than that in the traditional time-based synchronization algorithms: 500 ms in content-based versus 150 ms in time-based. This allows the content-based algorithm to take the decisions based on content change more than on time and virtually drop the frames, that is, change the frame rate, when they really are "very old." This threshold can be modified according to the master stream defined in a specific application, and it could also be made adaptive itself depending on network conditions. The literature contains significant research on human tolerance to delay for multimedia streams [20].

3.1.2. Video Rate of Change "ΔV". The video is coded as MPEG, where motion estimation between two consecutive frames is one of the concepts it is based upon [21]. MPEG accomplishes motion estimation by a search algorithm which determines the sum of absolute differences (SADs) for each macroblock in the search area, where macroblocks in MPEG can be thought of as the pieces making up the whole frame. The macroblock with the lowest SAD value is determined to be the closest match. For example, if the minimum SAD value between two macroblocks is 0, then the blocks are the same. Since the higher minimum SAD values imply more change, these values can be used to estimate quantitatively the total amount of change that has occurred from one frame to the other. In other words, the sum of minimum SAD values gives an estimate of the total change between frames, and dividing this number over the difference in time gives the rate of change. Therefore,

$$\begin{aligned}&\text{Estimated Video Rate of Change}\\&= \frac{(\text{Total Change Between Frames})}{\Delta T}\\&= \frac{\sum(\text{Change Between all Macroblocks})}{\Delta T}\\&= \frac{\sum \text{MIN (SAD)of all Macroblocks}}{\Delta T},\end{aligned} \tag{1}$$

where ΔT is time difference and

$$\text{SAD} = \text{SAD}(U, V) = \sum_{x=0}^{15}\sum_{y=0}^{15} |\, U(x, y) - V(x, y) \,|, \tag{2}$$

where x, y are pixel spatial coordinates and U, V are adjacent image frames.

Using the estimated video rate of change would give instantaneous changes between frames, but what is of interest

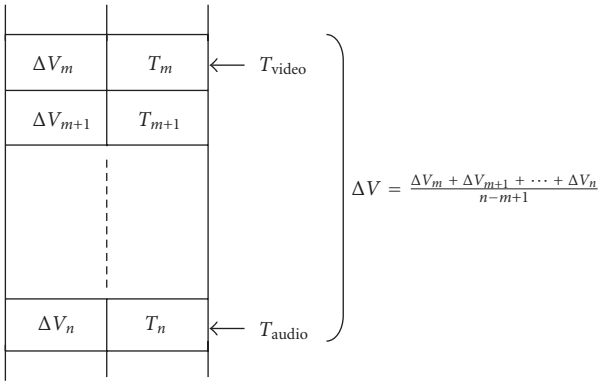

Figure 4: Average video rate of change calculation.

is the trend of change that the video has gone through. Therefore, the algorithm uses the *average video rate of change*, ΔV, which is the average of the estimated video rate of change between when the audio and the video were sampled.

Figure 4 illustrates an example of the average video rate of change (ΔV) calculation when the audio is ahead of the video. When the audio arrives ahead of the video, the audio embedded ΔV should be computed. This is performed by maintaining a table containing the instantaneous rates of change of video and the corresponding audio times. When the video frame arrives, the average of the ΔVs between T_{audio} and T_{video} is computed. A similar approach is followed when the video is ahead of the audio, thus the video embedded ΔV should be computed. In this case the instantaneous rate of change of the video and the corresponding video timestamp should be maintained in the table. When the audio frame arrives, the average of the ΔVs between T_{video} and T_{audio} is computed from the table.

It is worth noting, that choosing this definition of rate of change requires minimal additional processing as the SADs are already being computed by the MPEG encoder and the proposed method just extracts these values from the encoder and computes the corresponding average between when the audio and the video were sampled. Clearly, this definition of change in media stream can be modified for video depending on the encoding or even if a different media stream is used. Note that the rate of change in MPEG video has also been defined in the literature as a function of the motion vectors [21].

3.1.3. Video Rate of Change Thresholds: "Th_{audio}" and "Th_{video}". The thresholds indicating the allowable content variation, Th_{video} and Th_{video}, are important to the overall performance of the synchronization algorithm. If too low of a threshold is chosen, frames will be dropped too often. If too high, most frames will be rendered, regardless of their temporal relation with the audio stream. Both of these scenarios will result in degradation of quality. Thus, choosing an appropriate threshold is crucial. Using a constant threshold seems desirable at first, but closer examination of the problem of synchronization shows that this is not the case. Since the ultimate goal of synchronization is to render frames in such a way that they accurately depict what was recorded on the transmitting machine, it becomes desirable to be more forgiving of content variation for frames that are close in time to the audio sample. For example, consider a case yielding an average difference of 80 ms between a rendered video stream and its corresponding audio stream. Recalling that the goal of synchronization is accurate depiction of the recorded stream, it becomes evident that, if a frame and a sample have a timestamp difference less than the average, they are closer together than usually encountered. This indicates that rendering them should be favored. Consider the opposite scenario, where a frame and a sample are being considered for rendering with a timestamp difference larger than the average of 80 ms. Rendering these frames together would be less desirable since they are further apart than average, which results in a lower quality. To allow the synchronization algorithm to adapt in situations like these, the thresholds described above are adjusted in real time. The above discussion suggests an inverse relationship between the threshold and the timestamp differences. Smaller timestamp differences should allow more content variation, while larger timestamp differences than average should allow less content variation. Thus, the threshold is calculated using

$$\text{Th}_{\text{audio}} = \text{Th}_{\text{video}} = \frac{\Delta T_{\text{avg}}}{\Delta T_{\text{inst}}} * \Delta V_{\text{avg}}, \tag{3}$$

where ΔT_{inst} is the actual timestamp difference calculated between an audio sample and a video frame, ΔT_{avg} is the average timestamp difference, and ΔV_{avg} is the average content change variation. This results in the required behavior: when ΔT_{inst} is smaller than ΔT_{avg} more content variation is allowed, and when ΔT_{inst} is larger than ΔT_{avg} less content variation is allowed.

ΔT_{avg} could have been set to a constant value but that would not allow the synchronization conditions to adapt to the network conditions. ΔT_{avg} is initialized to 50 ms when the audio is ahead of the video and to 97 ms in the opposite scenario. These values are chosen from computing the average time difference between audio and video of fifteen video scenes and calculating the corresponding average:

$$\Delta T_{\text{avg}} = \frac{\sum_1^n \Delta T_{\text{avg}} \text{ per video}}{n}. \tag{4}$$

ΔT_{avg} will then be updated dynamically using the following formula:

$$\Delta T_{\text{avg}} = \alpha * \Delta T_{\text{avg}} + (1 - \alpha)\Delta T_{\text{inst}}, \tag{5}$$

where $\alpha = 0.875$ and ΔT_{inst} is the instantaneous time difference between audio and video frames. This dynamic behavior allows ΔT_{avg} to vary depending on the instantaneous time difference between audio and video, making the decision more accurate and realistic and the application more dependent on the present network conditions.

The choice of ΔV_{avg} will depend on what is considered an acceptable rate of content variation on average. To choose the appropriate value of ΔV_{avg}, twenty video scenes of medium motion activity were played, and the average content change

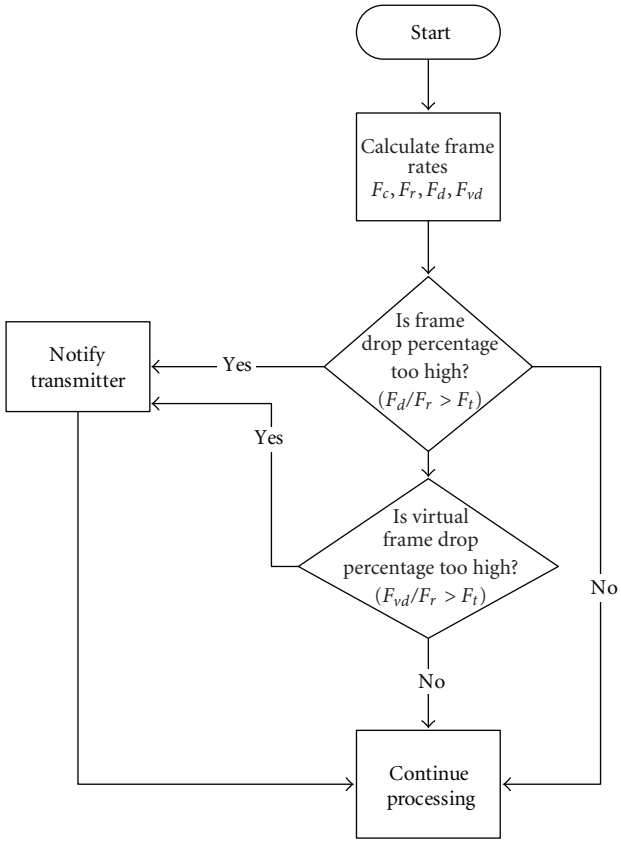

Figure 5: The adaptive frame rate algorithm (receiving machine).

difference was computed for each of these videos. Then the corresponding average value was calculated. The chosen video scenes consisted of drama sections that included few people who are standing, walking, and running. These scenes are considered to have a moderate motion activity [22]:

$$\Delta V_{\text{avg}} = \frac{\sum_1^n \Delta V_{\text{avg}} \text{ per video}}{n}. \tag{6}$$

Following this approach, the value of ΔV_{avg} was found to be 1050. This makes the algorithm more aggressive when fast video is played, that is, when the content change difference of video frames is high. The video frame in this case is rendered if the instantaneous time difference ΔT_{inst} is smaller than the average time difference ΔT_{avg}, whereas the algorithm is more forgiving when the content-change difference is low like in the low motion activity videos, allowing larger time difference between the audio and video frames.

3.2. Adaptive Algorithm. Figure 5 presents the flow chart of the adaptive algorithm at the receiving machine, given that F_c is the capture frame rate from a camera or other device, F_r is the rate at which video frames are being received from the network, F_d is the rate at which frames are being dropped, and F_{vd} is the rate at which the video frames are virtually dropped.

When synchronizing the multimedia stream, statistics are kept related to the state of the stream and the quality. These quality statistics are various frame rates: F_c, F_r, F_{vd}, and F_d. When the drop frame rate or the virtual drop frame rate surpasses a certain value, the algorithm concludes that frames are being dropped at an unacceptable rate and thus the sending side can be alerted to this fact.

A critical parameter at the receiving side in the performance of the adaptive frame rate algorithm is the acceptable frame drop rate. If too many frame drops are allowed, the effects of the adaptive frame rate algorithm will be nullified, since the sending rate is never lowered to allow a remote host to "catch up." If too little frame drop is allowed, the transmitting side will be notified of dropping frames too often, which will force the frame rate to remain unnecessarily low. However, making decisions based on the frame drop rate alone might be misleading. Consider an application that is receiving video frames from the network at a rate of about two per second ($F_r = 2$). In this case, dropping even one frame per second could be disastrous to the perceived quality of the application, since half of the video data being transmitted is being dropped. Now, consider another application receiving video frames at a rate of about 25 per second ($F_r = 25$). In this case, dropping one frame per second would be much more acceptable than in the previous case, since 96% of the video data being transmitted is being displayed. This hints that the threshold for considering a drop rate, F_d, to be unacceptable is more suitably expressed as a percentage than as an integer value. Therefore, deciding on whether too many frames are being dropped should be done while accounting for the frame receive rate, F_r. This is accomplished by considering the percentage of dropped frames per received frames, F_d/F_r, as well as the percentage of virtually dropped frames per received frames, F_{vd}/F_r. If this percentage is higher than a certain threshold, F_t, an unacceptable percentage of frames are being dropped and the transmitting side is alerted. In this implementation, F_t was chosen to be 50%. The choice of this percentage is a compromise between being too aggressive and too passive and could be chosen to be different for the drop and virtual drop ratios.

The adaptive algorithm at the sending machine is similar to the source-based rate control algorithm based on additive increase and multiplicative decrease (AIMD) approach. The AIMD works as follows.

If $p \leq P\text{th}$

$$r = \text{MIN}\{(r + \text{AIR}), \text{Max}\, R\} \tag{7}$$

else

$$r = \text{MAX}\{(\alpha * r), \text{Min}\, R\}, \tag{8}$$

where p is the packet loss ratio, Pth is the threshold for the packet loss ratio, r is the sending rate at the source, AIR is the additive increase rate, Max R and Min R are the maximum rate and the minimum rate of the sender, respectively, and α is the multiplicative decrease factor [23]. In a similar fashion, the adaptive algorithm works as presented in Figure 6. Once alerted that the current frame rate is unacceptable, the transmitting side reduces its frame rate multiplicatively or linearly, in an effort to avoid sending frames without them

being displayed. The frame rate is multiplicatively reduced to its half when the high drop rate is a result of high difference in content change between audio and video frames, that is, when $F_d/F_r > F_t$, whereas it is linearly decreased by 1 fps when the sender is alerted that the number of virtually dropped video frames is above the threshold, that is, when $F_{vd}/F_r > F_t$. This allows the algorithm to be more aggressive in the case where high content change occurs than in the case of high time difference between audio and video frames.

Eventually, the reduction in the sending frame rate will allow the receiving machine to drop fewer frames. The sender can then reduce its capture and send rate, in an effort to avoid sending frames without them being displayed. This gives the receiving machine time to "catch up." Contributing to the reduction in dropped frame rate is improvement in network conditions and/or a reduction in the rate of content change. Once the dropped frame rate is reduced, it becomes desirable to increase the frame rate to take advantage of the improving conditions. In the developed algorithm, speeding up the frame rate is a passive action where the receiving side does not initiate an increase in frame rate by sending a packet. Instead, the sending side monitors how long it has been since it has received a request to slow down. If it has been long enough between requests to slow down, the frame rate is automatically increased by 1 fps (linear increase). An upper bound for the frame rate, Max R, can be set in case of limited uplink capacity. In the experiments conducted the upper bound was limited by the camera's maximum capture rate of 25 fps. In the developed application, the interval the sending side will wait for before assuming it is safe to begin increasing the frame rate is set to 5 seconds. This value is another parameter which helps configure the behavior of the algorithm. If this waiting duration is set to a small value, the frame rate will be controlled more aggressively by speeding up the frame rate quicker than if the value is set to a larger interval. The behavior of this adaptive algorithm is similar to rate control in networking protocols like TCP/IP, which has proven to be very stable and robust. Note that having explicit notification from the receiver sent back to lower the rate reduces the amount of feedback overhead. Otherwise, the receiver would have to continuously feed back status information to the sender in order for the sender to make a decision. The contribution regarding the adaptive behavior is in the inclusion of content in the decision making process via the use of the virtual drop concept.

4. System Implementation

The system was assessed using remote tests conducted between the American University of Beirut (AUB), Lebanon, and Michigan State University (MSU), USA. The remote test setup is shown in Figure 7. Note that the machine at AUB is connected to the Internet using a wireless broadband setup with uplink bandwidth of 64 kbps and downlink bandwidth of 256 kbps. This allowed for the assessment of the algorithm over long distances via the Internet and under relatively limited bandwidth conditions. For the purpose of the test, a frame grabber used to capture video frames from a DVD player was installed in the PC. This allows for playing the same video scene for both time-based and content-based algorithms and therefore to perform a consistent and repetitive test resulting in a fair comparison between the two schemes.

The remote test was performed over video segments of different intensities of motion activity. Video activity levels are generally classified as low, moderate, and high. Example of low activity content involves talking heads like an anchor or an interview with still and low motion background. Moderate activity content may be a drama, educational, or news scene, with standing or still people, walking or running, whereas a high activity content may be a dance or music scene with high motion and lighting changes [22].

Video portions with low, medium, and high motion were tested to compare the behavior of the content-based synchronization algorithm under different content change variations. During each remote test, the round trip delays (RTDs) were recorded. The following were assessed: the average timestamp difference resulting in a render and drop of video frames and the average content difference resulting in a render and drop of frames. Objective and subjective assessment of the received videos was performed in order to compare the video quality. Objective methods aim to mathematically estimate the impairment introduced to the video while the subjective assessment helps in the statistical analysis of sample ratings generated by humans. These methods were applied using the "Video Quality Studio 0.32" software. The "Video Quality Studio 0.32" software uses the root mean square error- (RMSE-) based metric (SNR or PSNR) and the discrete cosine transform- (DCT-) based video quality metric (VQM) to objectively compare the video quality. In the RMSE-based metric, the software computes the difference between two frames. It is applied to digital video by finding the average of results for every frame. The software then computes the peak-signal-to-noise ratio (PSNR) value for each of the Y, U, and V frames as follows (see [24]):

$$\text{PSNR} = 20 * \log_{10}\left(\frac{225}{\text{RMSE}}\right) \text{dB}. \tag{9}$$

This equation shows that PSNR value is higher for lower values of RMSE. That is, a better quality of video is obtained when the difference between the original video sequence and the impaired video sequence is smaller.

Human vision and visual perception are complex mechanisms. The video quality definition varies with how we perceive color and motion. For this reason, objective video quality metrics that calculate the distortion, like RMSE, may fail to match with the subjective perception of humans since they do not take into consideration the spatial and temporal properties of the human visual perception. DCT-based VQM is an alternative video quality assessment method that takes into account the properties of the human visual perception. In fact, the eye sensitivity to special temporal patterns decreases with high spatial and temporal frequency. Based on this fact, high spatial and temporal information can be represented with less precision since the eye is not very sensitive to the loss of information. This property is exploited by DCT quantization [25].

Subjective assessment is performed using the mean opinion score (MOS) method. It provides a numerical indication

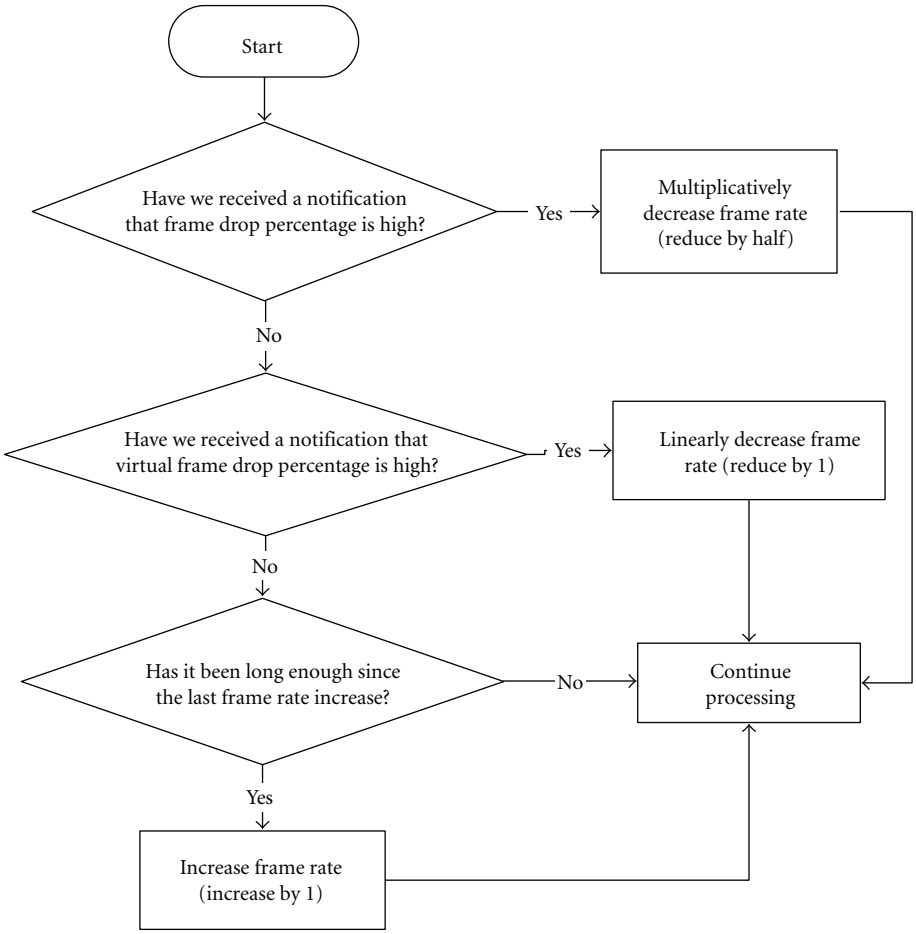

Figure 6: The adaptive frame rate algorithm (sending machine).

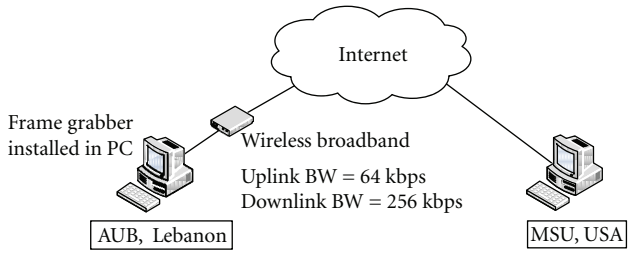

Figure 7: Remote test setup.

of the perceived quality of received media after transmission. The MOS is expressed as a number in the range (1) to (5), where (1) is the lowest perceived audio-video quality and (5) is the highest perceived quality. The MOS is generated by averaging the results of a set of standard, subjective tests where a number of persons (fifteen in our case) rate the audio and video quality of a given scene. Each person is required to give a rating using the rating scheme shown in Table 1.

Fast, moderate, and slow motion activity videos were played to different subjects using both the time-based and the content-based synchronization algorithms. The slow motion activity video consisted of an anchor speaking with a still background; the moderate motion activity consisted of a scene representing two people walking and talking at the same time, whereas the fast motion activity video consisted of several people dancing under changing light conditions

Table 1: Mean opinion score (MOS) interpretation.

MOS	Quality	Impairment
(5)	Excellent	Imperceptible
(4)	Good	Perceptible but not annoying
(3)	Fair	Slightly annoying
(2)	Poor	Annoying
(1)	Bad	Very Annoying

and different camera effects like explosion and fire. These scenes have been chosen based on the definition of the intensity of motion activity levels stated in [22]. Each subject was asked to assess the videos and record her/his opinion of the synchronization between audio and video and the overall quality of the scene. The MOS was then generated by averaging the recorded results.

5. Experimental Results and Analysis

Remote testing was conducted using a video conferencing application modified to support adaptive content-based synchronization. The application encodes video as H.263 and audio as GSM6.1. The application inserts after the UDP header a customized header which includes the content change information and the timestamp. In addition, in order for the buffering scheme or duration not to affect or bias the outcomes, the experiments were conducted under 0 ms buffering. However, this does not exclude our method from being used with buffering of any form. Clearly, in a practical application content-based synchronization would have to be applied in conjunction with buffering in order to optimize the overall application performance.

Remote testing with videos of different motion activities (low, moderate, and high motion activities) was conducted to assess the behavior of the proposed algorithm when the content change is low (low activity videos), medium (moderate activity videos), or high (high activity videos). In each case, the same video segment was used for both algorithms. For each test, the round trip delay and the percentage loss were recorded through pinging the remote machine during the test. This allows for the detection and assessment of the network conditions during the test. Table 2 presents the average and the standard deviation of the RTD obtained during each test, as well as the percentage of packet loss. This table is provided to detail the actual experimental conditions which are a function of the network conditions during each particular test.

The obtained values of the average timestamp difference resulting in a render and drop of video frames and the average content difference resulting in a render and drop of frames for both time-based and content-based algorithms are compared in Table 3.

All scenarios exhibit the same behavior; therefore, for simplicity the analysis will focus on the fast motion case. The table shows that, in the time-based synchronization algorithm, frames are only dropped when they are significantly skewed in time from audio. For this scenario, the average timestamp difference between audio and video at which a render has occurred is about 73 ms and the average timestamp difference between audio and video when video was dropped is about 3338 ms. Content in the time-based synchronization algorithm is not taken into consideration. This is evident from the fact that the average value of ΔV at which a render has occurred is higher than that for which a drop has occurred (average ΔV render = 1340 and average ΔV drop = 1267). This implies that many video frames have been rendered even though their corresponding content change difference is much higher than that of other frames dropped by the algorithm. This demonstrates that the time-based synchronization scheme renders and drops video frames by only comparing the time difference between the audio and video streams, regardless of the corresponding content change difference. As will be shown in the objective and subjective assessment, this behavior degrades the synchronization quality between audio and video and therefore degrades the quality of service offered to the user.

Table 2: RTD and percentage of packet loss information.

Sync type	Motion activity level	Average RTD (ms)	RTD standard deviation	Percentage loss
Content	Fast	583	666.3970	8%
Time	Fast	410	149.9826	0%
Content	Medium	374	113.1367	6%
Time	Medium	419	139.9904	4%
Content	Low	408	138.5518	0%
Time	Low	412	159.9281	2%

As for the content-based synchronization algorithm, frames are rendered at lower amounts of content variation than in the time-based synchronization scheme (1156 with content-based algorithm versus 1340 with time-based). Also, frames are dropped at significantly higher levels of content variation (20547 with content-based algorithm versus 1267 with time-based algorithm). This demonstrates the content-based algorithm behavior in rendering video frames with low content change and dropping those with high content change. As for the time difference, the table shows that the average timestamp difference resulting in a render (900 ms) is lower than that resulting in a drop (2203 ms). This demonstrates that the content-based synchronization algorithm also takes the timestamp difference between audio and video streams into account to make the rendering and dropping decisions. The table also shows that video frames are rendered if the difference between audio and video streams is on average 900 ms even though the threshold used to decide whether the frame is too old or not is 500 ms. The reason is that in the content-based synchronization, when the time difference between audio and video is greater than the threshold, the video frame is counted as if it is dropped in order for the sender to reduce the sending rate; however, this frame is not dropped in reality if the content change difference is acceptable. So the high value of the average time difference at which a render occurred is due to the presence of many video frames that are far in time from the corresponding audio streams and yet exhibit small content variation.

Table 3 also shows that, when the content-based algorithm is applied, the time difference between audio and video streams resulting in a render is smaller in the case of higher motion activity videos than in the case of lower motion activity videos (900 ms for fast motion video, compared to 1495 ms for medium motion video and 2295 ms for low motion video). The reason is that the content change difference ΔV in the lower motion videos is less than that in faster videos, thus the algorithm is more forgiving and allows

Table 3: Performance results of content-based and time-based synchronization algorithms obtained for videos with different levels of activities.

Sync type	Motion level	Avg timestamp difference render	Avg timestamp difference drop	Avg ΔV render	Avg ΔV drop
Content	Fast	900 ms	2203 ms	1156	20547
Time	Fast	73 ms	3338 ms	1340	1267
Content	Medium	1495 ms	3225 ms	1066	18680
Time	Medium	70 ms	2128 ms	1235	1234
Content	Low	2295 ms	4179 ms	996	16516
Time	Low	62 ms	3063 ms	1321	1123

Table 4: Objective assessment results for a 5-minute video with different motion activity levels.

Sync type	PSNR Y	PSNR U	PSNR V	Distortion in Y
Content	13.6220 dB	32.9594 dB	28.4345 dB	10.5532
Time	11.6838 dB	31.9429 dB	27.5583 dB	11.8282

larger time difference between audio and video streams when slower video motion activities are in play.

All these results given in Table 3 simply demonstrate that the algorithm is functioning as designed; however, they do not demonstrate that this would result in a better quality. The benefit of applying content-based synchronization is illustrated using objective and subjective quality assessment measures. The PSNR values of Y, U, and V frames obtained from RMSE-based model and the distortion level in Y frames obtained from the DCT-based VQM model are presented in Tables 4 and 5. These values are gathered from playing a 5-minute video with different motion activity levels (Table 4), then playing fast, moderate, and low activity 1-minute video scenes separately (Table 5). The results are recorded for both time-based and content-based schemes.

The table shows that the PSNR values of the Y, U, and V frames are higher when the content-based algorithm is applied than when the time-based synchronization algorithm is used. Also, it shows that the distortion in the Y frame is smaller with the content-based than with the time-based synchronization algorithm. This quantitatively demonstrates that the content-based scheme provides 18% better quality of service than the time-based synchronization algorithm since the difference between PSNR values of content-based and time-based algorithms is approximately 1.5 dB.

Table 5 shows that, for all levels of motion activity, the content-based scheme performs better than the time-based scheme. The PSNR of Y, U, and V frames is larger and the distortion in Y frame is smaller when the content-based synchronization algorithm is used. Also, the table shows the relation between PSNR values and the motion activity level regardless of the synchronization type used. The PSNR values are larger when the motion activity level decreases (using content-based synchronization algorithm, PSNR of Y frames is 17.30891 dB for fast activity video compared to 22.65270 dB for medium activity videos and 27.79932 dB for slow activity videos) which quantitatively means that the video quality is enhanced by about 80% when decreasing the motion activity level. Similarly, the table shows that the video exhibits less distortion when its motion activity level decreases (the distortion in Y frames is 8.768407 for fast video, 5.756310 for moderate video, and 3.139498 for low activity video). This is justified by the fact that the more activity in the video the more likely for frames to be dropped as per the algorithm. Also, more motion usually results in lower compression by the encoder and thus more bandwidth requirements resulting in degradation of performance. Another note is that the difference in PSNR values between the videos played using content-based and those played using the time-based synchronization algorithm is the largest when the video motion activity level is high. For example, the difference between the PSNR values of U frames between content-based and time-based schemes is $32.50701 - 29.60511 = 2.9019$ dB when fast motion video is played (which means that content-based synchronization algorithm provides about 40% better quality than time-based algorithm). This difference decreases with the motion activity level to become 0.69415 dB for moderate activity level (8.3% better quality with content-based synchronization) and 0.04 dB for low activity video (0.46% better quality with content-based synchronization). The same analysis applies to the distortion level differences. The difference in the distortion level between content-based and time-based schemes decreases as the motion activity level decreases. This is explained by the fact that the slower the motion level is the more forgiving the stream is to desynchronization. Since the difference in content between frames is relatively low for most frames and thus accounting for this difference would not affect much the quality perceived.

To assess the performance of content-based synchronization as perceived by the user, subjective tests were performed with fifteen subjects. The same fast, medium, and slow motion activity video scenes assessed objectively were saved at the receiver side (MSU) during tests using time-based and content-based synchronization schemes. The scenes were then rendered to the subjects to record their opinion regarding the synchronization and the overall quality over a scale of 1 to 5, 5 being the best and 1 being the worst performance. The average perceived synchronization and overall quality (MOS values) are recorded in Figure 8.

The bar chart shows the advantage of the content-based synchronization over the time-based synchronization scheme in terms of both the synchronization and overall quality. For example, in the case of moderate motion activity video, the MOS value of the synchronization quality is 3.57 when content-based scheme is applied, compared to

Table 5: Objective assessment results for 1 minute video scenes with low, moderate and high activity levels.

Sync Type	Motion Level	PSNR Y(dB)	PSNR U (dB)	PSNR V (dB)	Distortion in Y
Content	Fast	17.3089	32.5070	28.4788	8.76840
Time	Fast	14.4529	29.6051	24.1076	11.18156
Content	Medium	22.6527	37.3771	32.7195	5.75631
Time	Medium	21.6807	36.6830	32.0665	6.29525
Content	Slow	27.7993	41.8539	35.6949	3.13949
Time	Slow	27.5295	41.8142	35.6723	3.14671

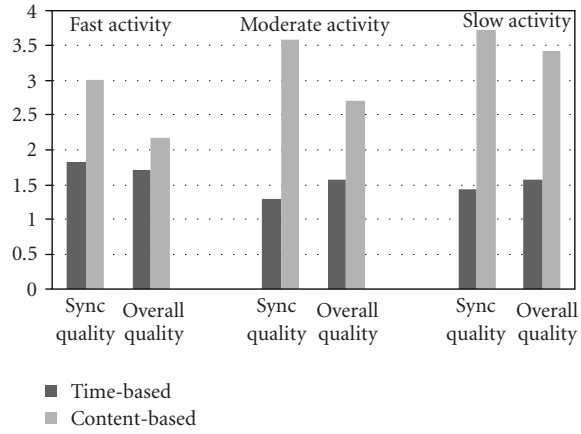

Figure 8: Perceived synchronization and overall quality.

1.285 when time-based algorithm is applied. And the MOS value of the overall quality is 2.71 for content-based, compared to 1.57 for the time-based algorithm. The same results were obtained for the other motion activity levels. This means that the evaluators perceived a better quality when content-based synchronization scheme is applied for different levels of motion activity. The values in Figure 8 also show the relation between the perceived quality and the motion activity level. The evaluators perceived better synchronization and overall quality when the video activity level is lower. For example, the synchronization quality when content-based synchronization is applied is perceived to have an average value of 3.71 when slow motion activity is tested versus an average value of 3.57 for moderate activity videos and 3 for fast activity videos. This is because when more activity is involved, the frames are more likely to be dropped and therefore the perceived quality degrades.

To summarize, the adaptive content-based synchronization algorithm constitutes an improvement over the traditional time-based synchronization algorithm. The limitation of the proposed algorithm is the choice of the scaling factor in the thresholds. Although, the thresholds are adaptive, there is a need to scale content change to the time difference scale and this depends on the definition of content change itself which depends on the stream being encoded and the encoder used. In addition, the proposed scheme introduces some overhead caused by sending the instantaneous content change difference, ΔV, with each packet. However, this overhead is not significant as it is of size byte which is very small compared to the size of the packets sent. Specifically, the smallest packets sent are audio packets which are of size 33 bytes. Therefore, the instantaneous content change difference, ΔV, represents only 3% overhead on throughput of audio and thus does not affect the quality of service. In addition, the table where the values of content change are recorded may grow but this problem can be resolved by periodically deleting old entries in the table after computing the needed parameters.

It is worth noting that, even though our method could be combined with buffering-based methods to improve the overall performance as stated earlier, if a direct comparison is to be done, the proposed method will outperform variable buffering techniques, as these methods only rely on delays to vary buffering with no regard to content. Considering the performance criteria we used, the buffering methods would have lower performance as content and its variation is not being considered. So two frames with large or small content change will be treated the same by all variable buffering techniques. These techniques just consider the delay (estimate, maximum, or minimum) of packets received in previous talk spurts [26], with no measure of content or stream behavior.

6. Conclusion and Future Work

In this paper, adaptive content-based synchronization of multimedia steams was presented. The primary contribution of the paper is the synchronization scheme which allows synchronization decisions to be made based on content as well as temporal proximity. It renders the media exhibiting low content change difference and drops those with high content variations, without ignoring the temporal relationships between the different streams. The secondary contribution is the adaptive scheme proposed which also accounts for content via the introduced concept of virtual drop. The algorithm was tested remotely between the American University of Beirut and Michigan State University to study its performance under real network conditions. Different video motion activity scenes were used to study and compare the behavior of the content-based synchronization algorithm under different content change variations. The objective assessment of the content-based and the traditional time-based synchronization algorithm showed that, for all motion activity levels, the video frames (Y, U, and V) have higher PSNR and the Y frame exhibits less distortion when the content-based algorithm is applied. The subjective assessment showed that, for all motion activity levels, subjects perceived better synchronization and overall quality when the content-based synchronization algorithm was applied.

Future work includes extending the proposed synchronization technique to other multimedia streams such as video, force, or other sensory measurements. In addition, video and audio encoding schemes can be further investigated and enhanced to provide a better quality of video and audio and to potentially use other measures of content change such as vectors of motion in MPEG.

Acknowledgment

The paper is supported by the American University of Beirut University Research Board.

References

[1] I. Elhajj, N. Xi, W. K. Fung, Y. H. Liu, Y. Hasegawa, and T. Fukuda, "Supermedia enhanced internet based telerobotics," *Proceedings of the IEEE*, vol. 91, no. 3, pp. 396–421, 2003.

[2] I. Elhajj, H. Hummert, N. Xi, and Y. H. Liu, "Synchronization and control of supermedia transmission via the internet," in *Proceedings of the International Symposium on Intelligent Multimedia, Video and Speech Processing*, Hong Kong, 2001.

[3] C. M. Huang and C. Wang, "Synchronization for interactive multimedia presentations," *IEEE Multimedia*, vol. 5, no. 4, pp. 44–61, 1998.

[4] S. Khanvilkar, F. Bashir, D. Schonfeld, and A. Khokhar, "Multimedia networks and communication," in *The Electrical Engineering Handbook*, W. Chen, Ed., Academic Press, 2004.

[5] H. McGurk and J. MacDonald, "Hearing lips and seeing voices," *Nature*, vol. 264, no. 5588, pp. 746–748, 1976.

[6] Arnte's sound site, 2002, http://www.hf.uio.no/imk/personer/arntm/McGurk_english.html.

[7] G. M. Garner, F. F. Feng, E. H. S. Ryu, and K. Den Hollander, "Timing and synchronization for audio/video applications in a converged residential ethernet network," in *Proceedings of the 3rd IEEE Consumer Communications and Networking Conference*, vol. 2, pp. 883–887, January 2006.

[8] Y. Xie, C. Liu, M. J. Lee, and T. N. Saadawi, "Adaptive multimedia synchronization in a teleconference system," *Multimedia Systems*, vol. 7, no. 4, pp. 326–337, 1999.

[9] M. Yang, N. Bourbakis, Z. Chen, and M. Trifas, "An efficient audio-video synchronization methodology," in *Proceedings of th IEEE International Conference on Multimedia and Expo*, pp. 767–770, Beijing, China, July 2007.

[10] D. Young, S. SampathKumar, and P. Rangan, "Content-based inter-media synchronization," in *Proceedings of the Multimedia Computing and Networking*, vol. 2417, pp. 202–214, San Jose, Calif, USA, February 1995.

[11] C. H. Wei and C.T. Li, "Design of content-based multimedia retrieval," in *Proceedings of the 5th WSEAS International Conference on Circuits, Systems, Electronics, Control & Signal Processing*, Dallas, Tex, USA, 2006.

[12] B. Furht and P. Saksobhavivat, "A fast content-based multimedia retrieval technique using compressed data," in *Proceedings of the Conference on Multimedia Storage and Archiving Systems III*, pp. 561–571, Boston, Mass, USA, 1998.

[13] J. Calic, S. Sav, E. Izquierdo, S. Marlow, N. Murphy, and N. E. O'Connor, "Temporal video segmentation for real-time key frame extraction," in *Proceedings of the IEEE International Conference on Acoustic, Speech, and Signal Processing*, pp. IV/3632–IV/3635, , Orlando, Fla, USA, May 2002.

[14] S. C. Chen, M. L. Shyu, N. Zhao, and C. Zhang, "An affinity-based image retrieval system for multimedia authoring and presentation," in *Proceedings of the ACM International Workshop On Multimedia Databases*, New Orleans, La, USA, 2003.

[15] V. S. W. Eide, F. Eliassen, O. C. Granmo, and O. Lysne, "Supporting timeliness and accuracy in distributed real-time content-based video analysis," in *Proceedings of the 11th ACM International Conference on Multimedia*, pp. 21–32, Berkeley, Calif, USA, November 2003.

[16] A. J. Perrott, A. T. Lindsay, and A. P. Parkes, "Real-time multimedia tagging and content-based retrieval for CCTV surveillance systems," in *Proceedings of the Internet Multimedia Management Systems III*, Boston, Mass, USA, 2002.

[17] H. Blanken, T. Grabs, H.-J. Schek, R. Schenkel, and G. Weikum, "Intelligent search on XML data," in *Lecture Notes in Computer Science (LNCS)*, vol. 2818, pp. 217–230, Springer, Berlin, Germany, 2003.

[18] Y. Li, A. Markopoulou, J. Apostolopoulos, and N. Bambos, "Content-aware playout and packet scheduling for video streaming over wireless links," *IEEE Transactions on Multimedia*, vol. 10, no. 5, Article ID 4543842, pp. 885–895, 2008.

[19] Y. Li, Z. Li, M. Chiang, and A. Robert Calderbank, "Content-aware distortion-fair video streaming in congested networks," *IEEE Transactions on Multimedia*, vol. 11, no. 6, Article ID 5235177, pp. 1182–1193, 2009.

[20] R. Steinmetz, "Human perception of jitter and media Synchronization," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 1, pp. 61–72, 1996.

[21] F. Halsall, *Multimedia Communications: Applications, Networks, Protocols and Standards*, Addison-Wesley, Swansea, UK, 2001.

[22] K. Peker and A. Divakaran, *Framework for Measurement of the Intensity of Motion Activity of Video Segments*, Mitsubishi Electric Research Laboratories, 2003.

[23] D. Wu, Y. T. Hou, and Y. Q. Zhang, "Transporting real-time video the challenges and approaches," *Proceedings of the IEEE*, vol. 88, no. 12, pp. 1855–1874, 2000.

[24] I. Richardson, *Video Codec Design: Developing Image and Video Compression Systems*, John Wiley & Sons, 3rd edition, 2002.

[25] A. B. Watson, "Toward a perceptual video quality metric," in *Proceedings of the Conference on Human Vision and Electronic Imaging III*, Proceedings of SPIE, San Jose, Calif, USA, 1998.

[26] M. Narbutt and L. Murphy, "Adaptive payout buffering for audio/video transmission over the internet," in *Proceedings of the IEE UK Teletraffic Symposium*, Dublin, Ireland, May 2001.