Evaluation of Multimedia Synchronization Techniques

Lynnae Ehley

IBM Corporation
Boca Raton, Florida 33431

Borko Furht and Mohammad Ilyas

Dept. of Computer Science and Engineering
Florida Atlantic University
Boca Raton, Florida 33431

Abstract

This paper presents a classification of the multimedia synchronization techniques used to control jitter among media streams. The inter-media synchronization techniques are described in regard to their topology. Two major classes of multimedia synchronization techniques are then introduced as being distributed or local. Distributed schemes are used in networking environments. Local schemes are used within a workstation. An overview of some of the more well known synchronization techniques is then provided with a concentration on the distributed schemes. An evaluation of all the techniques follows with noted advantages and disadvantages.

Keywords: inter-media synchronization and intra-media continuity, continuous and event-driven synchronization, jitter, and multimedia networks

1 Introduction

Multimedia is becoming largely incorporated into computer systems because of a number of technological advances such as high-speed communications, mass storage technologies and digital audio and video peripheral equipment. When multiple sources of media are played back in real-time either locally or in a distributed environment, maintaining the temporal relationships among the media streams can be difficult but is required for coherent playback. These temporal relationships have been defined as inter-media synchronization [1]. The streams incur skew due to a number of anomalies such as buffering delays and errors, as shown in Fig. 1. These skews are also known as jitter. Two types of inter-media synchronization have been defined as continuous or event-driven [2]. Continuous inter-media synchronization requires constant monitoring such as maintaining lip-synchronization between voice and video. Event-driven inter-media synchronization is less tightly knit such as the presentation of a slide show with blocks of audio allotted for each slide. It should be noted that all multimedia calls are required to maintain intra-media continuity. Intra-media continuity can be defined as a sequence of media quanta such as video frames or audio samples which convey meaning only when presented continuously in time [1].

Both the sources and the destinations are responsible for maintaining inter-media synchronization but most of the techniques rely more on the destinations.

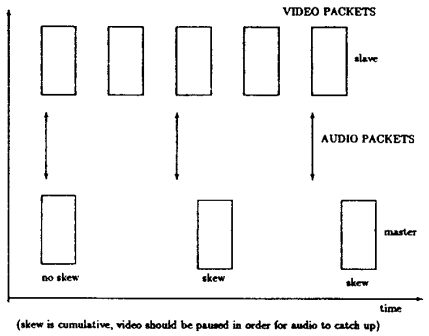

Figure 1: Example of jitter or skew

There are a number of different topologies that are used in specifying the type of synchronization required (if any). Bulterman et al. [3] describes four data location models, they are as follows:

1. Local single source - One source such as a CD-ROM distributes the media streams to the playback devices. As long as the devices maintain their playback speed, no synchronization technique is required.

2. Local multiple sources - More than one source distributes media streams to the playback devices. An example is a slide show played back with music or an audio tape. Synchronization is required within the workstation.

3. Distributed single source - One source such as a video tape distributes media streams across a network to one or more nodes' playback devices. An example is cable TV and again no synchronization is required other than maintaining the speeds of the playback devices.

4. Distributed multiple sources - More than one source (on one or more nodes) distributes media streams to multiple playback devices on one or more nodes.

0-8186-5530-5/94 $3.00

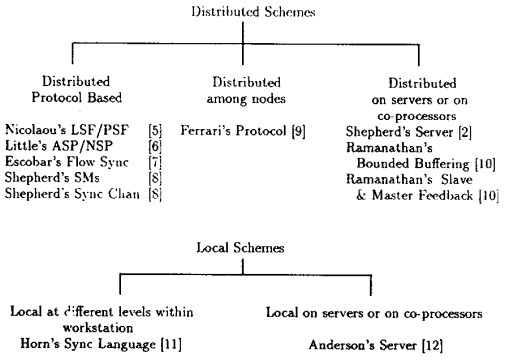

Figure 2: Classification of synchronization techniques

Due to the various inter-dependencies of multimedia streams and the different topologies of data locations, the requirements for synchronization are varied. Whenever multiple sources are used, some type of inter-media synchronization is required. The technique may be required to maintain either continuous inter-media synchronization or event-driven inter-media synchronization.

2 Classification

A number of different techniques have been proposed for multimedia synchronization to satisfy the diverse requirements. The techniques implement the synchronization control at various locations in the sources and destinations depending on the requirements. The proposed classification scheme, consisting of five distinct approaches is shown in Fig. 2. The synchronization techniques are first classified as being either distributed or local. The distributed approaches implement network protocol-based synchronization. Most of the techniques suggest using broadband networks for their underlying network. The transfer mode of broadband networks is Asynchronous Transfer Mode (ATM). Jitter as described in [4] can be assumed to be bounded in ATM networks if traffic management principles such as admission control and resource reservation are employed. Since bounded jitter can be assumed, buffering of the media streams in skew circumstances can smooth out the temporal inconsistencies. These schemes are basically in agreement that the functionality should reside in either the session or the transport layer of the Open Systems Interconnection (OSI) model. The local approaches are used in single-sites for multimedia synchronization.

The first approach includes protocol-based synchronization at the sending and/or receiving processes while the second approach implements synchronization for network-based applications but scatters the synchronization functionality among all of the internetwork's gateway nodes. The third approach implements distributed synchronization within servers or on co-processors while the fourth approach implements synchronization at different levels within the local workstation (no networking is involved). Finally, the fifth approach implements local synchronization within servers or on co-processors.

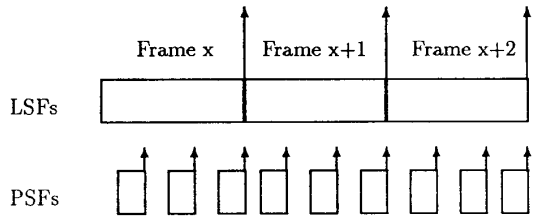

Figure 3: Two-level synchronization

3 Multimedia sync techniques

This section provides a more detailed description of some of the more widely known multimedia synchronization techniques. The major concentration is on the distributed synchronization techniques.

3.1 Nicolaou's LSF/PSF sync technique

Nicolaou's scheme belongs to the class of protocol-based synchronization techniques. An out-of-band synchronization protocol is proposed to deal with the three types of inter-media synchronization described, isochronous, error and pre-defined points. Pre-defined points include reset, start, and stop among others [5]. Nicolaou's scheme identifies two levels of data elements for defining synchronization points. Physical synchronization frames (PSF) are defined as the unit of synchronization for the communication subsystem. They refer to a physical unit of communication like a packet and are isochronous. Logical synchronization frames (LSF) are defined as the unit of synchronization for the controlling application. LSFs refer to error and pre-defined points. The LSFs are made up of a multiple of PSFs as shown in Fig. 3. By specifying the LSF as a QOS parameter, some degree of the Quality of Service (QOS) can be indicated for synchronization. A close relationship between the LSFs and the PSFs, like one-to-one implies tight synchronization. A more loose relationship is specified as n-to-one [5].

One of the pre-defined synchronization points that can be specified is a "stop" operation. When "stop" is requested in blocked mode, all of the PSFs are presented to the end of the current LSF. However, in non blocked mode, the PSF presentation is stopped immediately.

Synchronization points may refer to concurrent activities. These points are represented by synchronization variables. When a variable has been set, the synchronization point has been reached. Likewise, if the variable has not been set the synchronization point has not been reached. Synchronization variables are specified with three components, stream, synchronization point and value. Three primitives are defined for

the synchronization points, WaitForSV, SignalSV, and TestSV. WaitForSV is used by the application to control the constituent channel's streams playout. SignalSV is an upcall to the application that a variable has been set. TestSV is used by the application for determining the amount of data that has been played out on a stream.

3.2 Little's ASP/NSP sync technique

Little's ASP/NSP scheme also belongs to the class of protocol-based synchronization techniques. Little et al. proposed a two-level communication architecture to provide synchronization of multimedia calls [6]. The network synchronization protocol (NSP) maps to the session layer of the OSI Reference Model and the application synchronization protocol (ASP) maps to the application layer. The NSP is responsible for establishing and maintaining connections while providing some synchronization functionality. The ASP layer provides an integrated synchronization service for applications using multimedia. The protocol's theory is based on a few principles which determine the amount of buffering necessary to maintain synchronization with a given probability of failure for any set of arbitrary streams characterized by playout time instants, end-to-end delays and playout intervals [6].

An Object Composition Petri Net (OCPN) is used to describe the temporal relationships between media objects involved in a multimedia call. It captures both the precedence relations and the playout durations for the objects involved in the multimedia call.

The scheme can be used for applications that require live sources to be synchronized. Playout deadlines are generated on-the-fly from the stream of packets sent from the source(s).

3.3 Escobar's flow sync protocol

Escobar's flow synchronization protocol belongs in the class of protocol-based synchronization techniques too. Escobar's idea is similar to Ferrari's distribution technique [9] which is in the class of scattered synchronization among nodes [7]. The flow synchronization protocol suggests implementing the synchronization below the session layer, in the Multimedia Mechanisms Layer (MML) [5]. Escobar's approach as does Ferrari's relies on a clock synchronization protocol to maintain clock synchronization within the network. It should be noted that there is opposition to relying on synchronized clocks in a distributed network [10].

The protocol time-stamps data at the sources and delays all of the destinations by the equalization delay (largest amount of end-to-end delay expected on any destination). In this way, delay is equal across all streams and the streams are synchronized. The equalized delay can be fixed or adaptive. This scheme is used primarily for multimedia calls that have multiple sources and/or destinations.

Fig. 4 illustrates the protocol control. An initiator processor sends control messages to all source processors, $S^i s$ and all destination processors, $D^i s$ to initialize or change parameters. Event messages are sent by $S^i s$ to $D^i s$ to communicate flow parameters. Synchronization groups are formed by the distributed application that has access to all the processors. Period

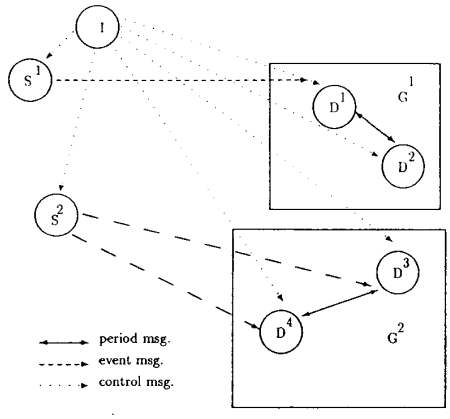

Figure 4: Protocol message flow

messages are sent among the processors in a synchronization group to compute the common synchronization delay for the group.

3.4 Shepherd's sync techniques

Shepherd's schemes also belong to the class of protocol-based synchronization techniques. The techniques for multimedia synchronization proposed by Shepherd et al. [8] also support the functionality residing below the session layer and suggest that it reside in the transport layer. The transport layer was chosen because it is the minimal layer required by future applications [8].

Two different techniques are suggested for intermedia synchronization, the synchronization marker (SM) concept and the synchronization channel concept. In addition, a server approach is also proposed. In the synchronization marker technique, the sender inserts 'synchronization markers', into the data. Buffering occurs at the transport layer for streams which have had their SM received. Data streams which have not had their SM received, continue to display any received data. When all streams' SMs have been received, buffering ceases and all streams display data again until subsequent SMs are received.

Some deficiencies exist with this scheme. First, massive amounts of buffering may be required particularly when delays in transmission occur. Second, modifications are made in the data streams and may be difficult to perform when using compressed video. Third, the scheme requires all streams to be synchronized and this may not always be desired. Suppose one stream is independent, special arrangements must be outlined in the protocol for these types of situations.

The second technique is the synchronization channel concept. In this scheme, a separate synchronization channel is to be used to transmit synchronization information to the receiver. Three types of presentation controls can be requested by the user for media streams, in series, in parallel or in any order. The in-

formation carried on the SC contains the presentation control, followed by the references to synchronization points for all of the data streams. The synchronization points refer to a count of the Transport Protocol Data Units (TPDUs) or packets that are to be transmitted for synchronization.

It is suggested that the synchronization channel run at the same speed as the fastest data channel. This is the preferred method so that synchronization information does not lag behind the data. It should be noted that some overhead is incurred with the use of this synchronizing channel and that more processing is now required in the transport layer.

The server approach uses a transputer-based multiprocessor front-end unit for multimedia support and network protocol support. The synchronization manager provides the synchronization service. Event driven synchronization is handled in the front-end by the interpretation of scripts. Continuous synchronization is handled by orchestrator processes in the front-end unit [2].

3.5 Ramanathan's sync techniques

Ramanathan's synchronization techniques belong to the class that implements synchronization within servers or on co-processors. Ramanathan et al. [10] presents feedback techniques for synchronization in distributed multimedia systems. A multimedia server provides distribution of videos over the BISDN to mediaphones. Mediaphones are described as devices that have minimum capability to playback media and lack the sophistication to run any type of synchronization mechanism [10]. In order to maintain intra-media continuity and inter-media synchronization, feedback units are sent back to the multimedia server from the mediaphones. Selected feedback units (a percentage to guarantee synchronization [1]) are sent back concurrently with the playback of the data parts of these media units. The media units are lightweight messages (part of the header) that contain only the number of the media unit.

With the feedback for maintaining intra-media continuity, servers can estimate the earliest and latest times at which playback of the units could have been initiated. This feedback then aids in the estimation of subsequent media units (u') earliest and latest playback times. The server attempts to estimate the earliest time at which u' can be transmitted to avoid buffer overrun and the latest time at which u' can be transmitted to avoid mediaphone starvation.

Feedback is also used as an aid in maintaining inter-media synchronization. In order to determine the temporal relationships between related streams, relative time stamps (RTSs) are assigned to each media unit. These stamps signify the relative time of playback. When the RTS of two media units are equivalent, they are to be played back simultaneously. The multimedia server is responsible for maintaining synchronization by skipping or pausing the slave in order to synchronize playback with that of the master.

Two inter-media synchronization techniques are proposed: one uses bounded buffering with feedback from the master, and the other uses feedback from all devices. In bounded buffering, when the multimedia server transmits media units to the master, it simultaneously transmits all media units with the same RTS to the slaves. Therefore, if a slave is ahead of the master, the slave is starved for a period of time while the master catches up. Likewise, if a slave is behind the master, media units are buffered until overrun occurs. Upon overrun, the slave is then forced to skip media units to catch up with the master. It should be noted that feedback units are transmitted only by the master to maintain intra-media continuity. Slaves' output is considered acceptable as a discontinuous flow.

There are drawbacks with this technique. Skipping at slaves is determined only by the amount of buffering available at the slave. Likewise, pausing of slaves is dependent only on the amount of buffering available at the master. Since the buffering capacity at the master is more of a continuity issue, it should not have any impact on the inter-media synchronization. This technique can be improved by using the multiple feedback technique.

In this technique, both master and slaves transmit feedback units to the multimedia server. The feedback units sent by the master are primarily used to maintain intra-media continuity. The feedback units sent by the slave are used to maintain inter-media synchronization. The multimedia server estimates the earliest and latest playback times of the media units sent by the master and slaves. The server then compares the estimated playback times of the master and slaves to determine if any asynchrony exists. When asynchrony exists, the multimedia server forces the slave to skip or pause by dropping or duplicating data in order to regain synchronization with the master.

This technique must be integrated into the OSI model to support distributed systems. The session layer is chosen to implement the feedback techniques because it deals with control of end-to-end connections.

3.6 Horn's sync language

Horn defines a synchronous language called Esterel to be used in supporting multimedia object synchronization. It uses broadcast valued signals for communication between parallel processes. Synchronous languages provide real-time primitive support. Media objects are defined as objects that encapsulate the processing and synchronization associated with a type of media. The behavior of objects is defined in scripts using invocations of the object's methods [11].

3.7 Anderson's CM I/O server

Anderson's continuous media I/O server belongs to the class of local synchronization techniques within servers or on co-processors. Anderson describes algorithms for recovering from asynchrony among interrupt-driven media I/O devices which are connected to a continuous media (CM) I/O server. This technique is mainly applicable to single-site workstations. The server (runs on SUN Sparcstation) described is called ACME and it communicates over transmission-control protocol (TCP). The ACME synchronization mechanism described is called a logical

time system (LTS). Clients use the LTS to synchronize devices that share temporal relationships [12].

In order to specify how to synchronize temporally related devices, one device in the LTS is chosen as the master. The device chosen as master should have the most critical intra-media continuity criteria. The temporal relationships are specified as time stamps, or sequence numbers. Data units with the same time stamp are to be played out at the same time. All of the other devices in the LTS are considered to be slaves. When a slave lags behind a master, some of the media units buffered in the slave may be skipped (for output devices) to align the slave with the master. Also, if a slave goes faster than the master, pausing (for output devices) on media units in the slave will occur in order to align the slave with the master.

Other methods for maintaining synchronization include, varying the hardware I/O rate of the physical device and interpolating data to adjust the effective I/O rate. Skipping/pausing and interpolating data are preferred over manipulating the hardware because of the ease in programming. In these synchronization schemes, rate adjustments on streams using the same physical device but with different LTSs can be accomplished. Anderson chose skipping/pausing over interpolating data because it is a simpler mechanism to support.

4 Analysis of the sync techniques

The majority of the research done for multimedia synchronization has focused on distributed networks. The reason for this is that the further the streams must travel, the more skew can occur. It should be noted that no one synchronization technique can be chosen to handle every application's requirements. However, it is worthwhile to take note of the schemes that have shown success. A comparative analysis of the various techniques is given in Table 1.

More research is required in this area. With the advent of BISDN, the number and complexity of multimedia applications will increase dramatically. The technology must be available to handle the complicated issues of intra-media continuity and inter-media synchronization of multimedia applications in an efficient and reliable manner.

5 Conclusion

A lot of research has been devoted to multimedia synchronization techniques within the last few years. By classifying the techniques and studying the varied requirements both in the data location and the type of synchronization (continuous or event-driven) it is clear that the complexity is vast. The techniques which have had the most success seem to have offloaded the processor to provide the multimedia synchronization support or have specified the use of special operating systems designed to handle the speeds and vast amounts of media that are to be supported. A bottleneck found in more than one multimedia study has been the system bus. Until these problems are overcome, multimedia systems can not be fully implemented.

References

[1] P. Venkat Rangan, S. Ramanathan, H. M. Vin, and T. Kaeppner, "Techniques for Multimedia Synchronization in Network File Systems", Technical Report Number CS92-229, January 1992.

[2] D. Shepherd, D. Hutchinson, F. Garcia and G. Coulson, "Protocol Support for Distributed Multimedia Applications", Computer Communications, Vol. 15, No. 6, pp. 359-366, July/August 1992.

[3] D. C. A. Bulterman, and R. van Liere, "Multimedia Synchronization and UNIX", Network and Operating System Support for Digital Audio and Video. Second International Workshop Proceedings, Nov. 1991.

[4] G. M. Woodruff, and R. Kositpaiboon, "Multimedia Traffic Management Principles for Guaranteed ATM Network Performance", IEEE Journal on Selected Areas in Communications, Vol. 8, No. 3, pp. 437-446, April 1990.

[5] C. Nicolaou, "An Architecture for Real-Time Multimedia Communication Systems", IEEE Journal on Selected Areas in Communications, Vol. 8, No. 3, pp. 391-400, April 1990.

[6] T. D. C. Little, and A. Ghafoor, "Multimedia Synchronization Protocols for Broadband Integrated Services", IEEE Journal on Selected Areas in Communications, Vol. 9, No. 9, pp. 1368-1382, December 1991.

[7] J. Escobar, D. Deutsch and C. Partridge, "Flow Synchronization Protocol", 1992 IEEE Globecom Vol. 3, pp. 1381-1387, 1992.

[8] D. Shepherd and M. Salmony, "Extending OSI to support synchronization required by multimedia applications", Computer Communications, Vol. 13, No. 7, pp. 399-406, September 1990.

[9] D. Ferrari, "Delay Jitter Control Scheme for Packet-Switching Internetworks", Computer Communications, Vol. 15, No. 6, pp. 367-373, July/August 1992.

[10] S. Ramanathan, and P. V. Rangan, "Feedback Techniques for Intra-Media Continuity and Inter-Media Synchronization in Distributed Multimedia Systems", Computer Journal, Vol. 36, No. 1, pp. 19-31, 1993.

[11] F. Horn, and J. B. Stefani, "On Programming and Supporting Multimedia Object Synchronization", Computer Journal, Vol. 36, No. 1, pp. 4-18, 1993.

[12] D. P. Anderson, and G. Homsy, "A Continuous Media I/O Server and Its Synchronization Mechanism", IEEE Computer, pp. 51-57, October 1991.

Table 1: Evaluation of Multimedia Synchronization Techniques

CLASS	TECHNIQUE	ADVANTAGES	DISADVANTAGES
1	**Nicolaou's LSF/PSF** – Sync protocol that defines 2 levels for sync points, LSFs and PSFs	Constant feedback is provided to application	Ideas are sketchy, a hierarchical scheme for applying to data elements
1	**Little's ASP/NSP** – Protocol that determines a schedule and amount of buffering for playback based on the probability of link failures	Good for stored data retrieval and for real-time	Requires clock sync, unexpected anomalies not covered, and not applicable if media streams have different playback rates
1	**Esocobar's Flow Synch Protocol** – Source time stamps data and destination delays playback by the equalization delay	Easy to implement	Requires global clock sync, mainly used for multiple sources and/or destinations in MM
1	**Shepherd's SMs** – Sources insert SMs and destinations buffer streams that have had their SMs received	Only minor modifications to protocols required	Requires massive buffering, difficult to insert SMs and independent streams difficult to implement
1	**Shepherd's Sync Channel** – Synchronization information is sent on another channel as a count of the TPDUs of the bytes to be received for streams	Handles independent streams and is flexible	Sync channel requires overhead and more overhead in transport layers
2	**Ferrari's distributed protocol**– Internetwork gateway nodes delay packets until the canonical arrival time	Does not require increasing amounts of buffer space as distance from the source grows	Requires clock sync and complicates network switches
3	**Shepherd's server** – Sync manager uses scripts and orchestrator processes	Offloads processor, uses elastic buffers and application has full control of streams	Expensive H/W setup
3	**Ramanathan's Bounded – Buffering** – Server transmits media units to slave at same time that they are sent to master	Easy to implement and is flexible	Dependent on the amount of buffering in devices
3	**Ramanathan's Master and Slave Feedback** – Server uses feedback units to compare estimated play-back times of master and slaves	Constant feedback and is flexible	Support is required in all devices
4	**Horn's Sync Language** – Defines language requirements for providing sync, i.e. Esterel	General approach, simplifies MM sync programming and has support to ensure that sync is met	Requires O/S with real-time features
5	**Anderson's Server** – Uses a CM I/O server to synchronize local devices using a LTS	Constant feedback is provided to application	Described only for single site work-stations and not suited for today's O/Ss