Information Systems 34 (2009) 108–131

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

# Multimedia group and inter-stream synchronization techniques: A comparative study

Fernando Boronat *, Jaime Lloret, Miguel García

*Communications Department, Universidad Politécnica de Valencia, Escuela Politécnica Superior de Gandia—IGIC Institute Ctra. Nazaret-Oliva S/N, 46730 Grao de Gandia, Valencia, Spain*

ARTICLE INFO

*Article history:*
Received 13 May 2008
Accepted 13 May 2008
Recommended by: D. Shasha

*Keywords:*
Synchronization
Inter-stream synchronization
Group synchronization
Multimedia systems

ABSTRACT

This paper presents the most comprehensive analysis and comparison of the most-known multimedia group and inter-stream synchronization approaches. Several types of multimedia synchronization are identified but only inter-stream and group synchronization algorithms are considered. This is the first survey including group synchronization techniques. A classification of the main synchronization techniques included in most of the analyzed algorithms complements the paper. Finally, a table is presented summarizing the main characteristics of each analyzed algorithm according to those techniques and other critical issues.



## 1. Introduction

In the past decade, we have seen a spectacular growth of the distributed multimedia real-time systems, most of them characterized by one or several sources transmitting (unicast or multicast) multimedia streams to one or several receivers, playing one or several of the streams. Multimedia systems are characterized by the computer-controlled integration of the generation, communication, processing and presentation of different media streams. These media streams can be divided into two categories: continuous and static (or non-continuous). Continuous media, e.g. video and audio, have well defined temporal relationships between subsequent ***Media Data Units***[1] **(MDUs)**. Static media, e.g. text, slides, images and graphics, have no temporal properties within themselves. Blakowski and Steinmetz [1] defined **multimedia system** as '*that system or application supporting the integrated processing of several types of objects or information, being at least one of them time-dependent*'.

A **distributed multimedia presentation** (**DMP**) integrates multiple media streams, e.g., audio, video, image, and text media, and possesses timeliness requirement of media units with respect to the presentation. DMP systems require flexibility and good quality of service (QoS) for multimedia data presentations. To ensure flexible and satisfactory presentations of multimedia data, collaborations between servers (sources), network and clients (receivers) must be carefully designed to retrieve the data from the disk (or database) and transfer the data to the client (receiver).

Due to the time dependency between the different media objects, a coordination and organization in time of the different information streams is needed. Such a process of maintaining the temporal relationship and guaranteeing a time-ordered presentation between the MDUs of one or several media streams is called the **multimedia synchronization process**. In this paper we use the **multimedia synchronization** concept to refer to the process of integration, in the presentation instant (*playout point*), of different types of media streams (continuous and/or static). Hereafter, we will use the term '*synchronization*' to refer to that concept. On the other

* Corresponding author. Tel.: +34 962 849 341; fax: +34 962 849 309.
*E-mail addresses:* fboronat@dcom.upv.es (F. Boronat), jlloret@dcom.upv.es (J. Lloret), migarpi@posgrado.upv.es (M. García).

[1] We have found that a MDU is also called *media or medium unit* (MU), *logical data unit* (LDU), *information unit* (IU) or *stream granules* (*unit of stream granularity*) by different authors.

0306-4379/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2008.05.001

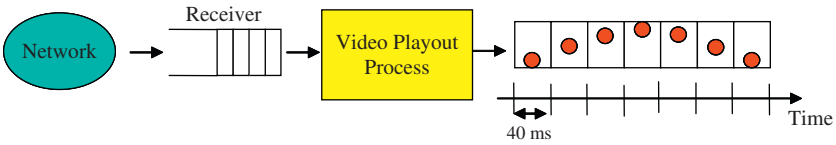


**Fig. 1.** Intra-stream synchronization.

hand, we use the term **multimedia synchronization algorithms** for protocols or solutions including the functionalities of maintaining the temporal relationships and guaranteeing a time-ordered presentation of one or several media streams. The solutions for synchronization use different techniques to coordinate the temporal and spatial object ordering, once the different media types have specific performance requirements.

Synchronization can be distinguished on different levels of abstraction. ***Event***[2]***-based synchronization*** must ensure a proper orchestration of the presentation of distributed multimedia objects. A multimedia object may be, for instance, a news cast consisting of several media objects, like audio and video. On a lower level, ***continuous synchronization*** or ***stream synchronization*** copes with the problem of synchronizing the playout of the data streams [2]. The classical example of stream synchronization is the synchronization between the audio stream and the associated lip movements in a speech (in this case a very accurate synchronization is required between both streams' playout processes), which is called ***lip-synchronization*** or ***lip-sync*** [3–5].

In continuous synchronization, we also can distinguish the temporal and MDUs relationships according to the use of either ***live*** or ***syntactic synchronization*** [1]. In the former case, the capturing and playback must be performed almost at the same time; in the latter case, samples are recorded, stored and played out at a later point of time. Live synchronization exactly reproduces the temporal relations made during the capturing process, while synthetic synchronization artificially specifies the temporal relations. Teleconferencing [6,7] is an example of a live synchronization application, while synthetic synchronization is often used in retrieval-based systems to rearrange multimedia objects to produce new combined multimedia objects (for example, video on demand (VoD) applications [5]). Temporal models allow specification of the temporal relations and operations of multimedia objects. These models can express complex operations by combining simple operations such as parallelized media streams, serialized multimedia objects and simple independent multimedia objects [1]. For live synchronization, the tolerable end-to-end delay is in the order of only a few hundred milliseconds. Consequently, the size of the elastic buffer must be kept small, and we can find trading-off requirements for jitter compensation against low delay for interactive applications. Synthetic synchronization of recorded media streams is easier to achieve than live synchronization: higher end-to-end delays are tolerable, and the fact that sources can be influenced proves to be very advantageous. For example, it is possible to adjust playback speed or to schedule the start-up times of streams as needed. However, as resources are limited, it is desirable for both kinds of synchronization to keep the required buffers as small as possible [8].

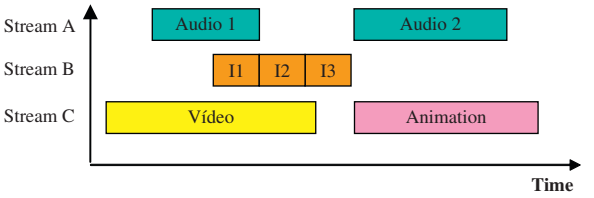


**Fig. 2.** Inter-stream synchronization.

In temporal synchronization we can distinguish between intra-stream (or intra-media) synchronization, inter-stream (or inter-media) synchronization and group (or inter-destination) synchronization. The distinction between these three types of synchronization will allow us to identify the different techniques or mechanisms to achieve them.

**Intra-stream synchronization** deals with the maintenance, during the playout, of the temporal relationship within each time-dependent media stream, i.e. between the MDUs of the same stream. As an example, we can cite the temporal relationship between MDUs of a video sequence. If we suppose the video was captured at a generation rate of 25 frames/s, each frame has to be displayed during 40 ms in the visualization device. In Fig. 1, synchronization requirement is shown for a video sequence showing a jumping ball. When reaching the receiver, the MDUs will be stored in a reception buffer to be able to guarantee the intra-stream synchronization. It will be necessary to guarantee the existence of MDUs in the buffer during the playing process (avoiding buffer underflow situations—*starvation*) and/or to guarantee that the buffer is not full when new MDU arrive (avoiding buffer overflow situations—*flooding*). Moreover, the playout process should be able to consume the MDUs with the same appropriate rate.

On the other hand, **inter-stream synchronization** refers to the synchronization, during the playout, of the playout processes of different media streams (time-dependent or not) involved in the application. An example of the temporal relationships between media streams in a multimedia application is shown in Fig. 2 (display-time bar char of a presentation's temporal schedule). It shows a playout starting with a video, followed by an audio sequence and several static images (slides), and, when all these sequences finish, there is an animation with related

[2] An event is an occurrence in time that can be instantaneous or can occur over some time period [61].

audio comments. In this case, first, intra-stream synchronization is also needed, and then, during the playout, some actions to correct possible deviations between playout processes should be taken to guarantee inter-stream synchronization.

As examples of inter-stream synchronization, we can cite *lip-synchronization* (*lip-sync*, [3–5]) as explained above, the synchronization between static images (slides) and an audio sequence describing the slides, etc. For example, if a presentation of video and audio streams is conducted without enforcement of synchronization, jitter will gradually accumulate between both streams. Such jitter may severely affect the performance of the presentation, especially in applications where speech is involved. Thus, the synchronization of multiple media streams becomes an essential prerequisite to any successful multimedia presentation application.

Manvi and Venkataram [9], present an inter-stream synchronization classification with three types: *point*, *real-time continuous* and *adaptive synchronization*. Point synchronization realizes that the start/completion time of the MDUs of the streams is synchronized with a certain specified synchronization point between the streams. In real-time continuous synchronization, MDUs are synchronized with a real-time axis. In adaptive synchronization, for example, the presentation time of the MDUs can be adjusted at regular intervals with change in network delays to reduce the losses.

From this point of view, we can also define a ***multimedia presentation*** as a set of media streams upon which synchronization constraints are specified on the display operations to enforce both intra and inter-stream constraints.

Apart from the above types of synchronization, in multicast communications, we can find another type of synchronization, called **group or inter-destination synchronization**, involving the synchronization of the playout processes of different streams in different receivers, at the same time, to achieve fairness among the receivers. We can cite the example of teleteaching applications in which a teacher could send (multicast) a video sequence (documentary or film—stored content stream) and, during the session, sometimes the teacher could make occasional comments about the video (live content stream). Network quizzes are other examples, in which the same multimedia question must appear at the same time to all the participants to guarantee fair play. In the first example, a simultaneous playout of the streams is important for both stored content and live content streams. Even if we only send the video stream (documentary or film), each video MDU (frame) should be played simultaneously in all the receivers (students) and then the students could comment the video content with other students. To guarantee that the initial playout instant (the playout beginning or starting point) should be the same for all the receivers. Once the playout processes have started simultaneously in each receiver, the temporal relationships between MDUs of the same stream should be maintained by the intra-stream synchronization process for that media stream. Nevertheless, due to the difference between end-to-end delays (due to different network delays, different consumption rates of the different receivers, etc.), resynchronization processes will be needed to maintain the receivers synchronized (group synchronization). In Fig. 3, we can see the playout of the above jumping ball sequence, synchronized in all the receivers.

In [10–16] another kind of synchronization (and solutions for it) is presented (***interactive synchronization***) for interactive distributed multimedia applications (IDMP). In some of these applications, VCR-like user interactions allow users to modify the presentation configuration at any time during the presentation. Typical user interactions include *fast forward (FF)*, *reverse (RR)*, *skip* and *pause*/*restart* the playout. There is a special difficulty in providing such user interactions due to the fact that they are issued dynamically and unpredictably during the presentation, which complicates the synchronization control.

The maintenance of temporal relationships within a stream or among the multimedia streams usually depends

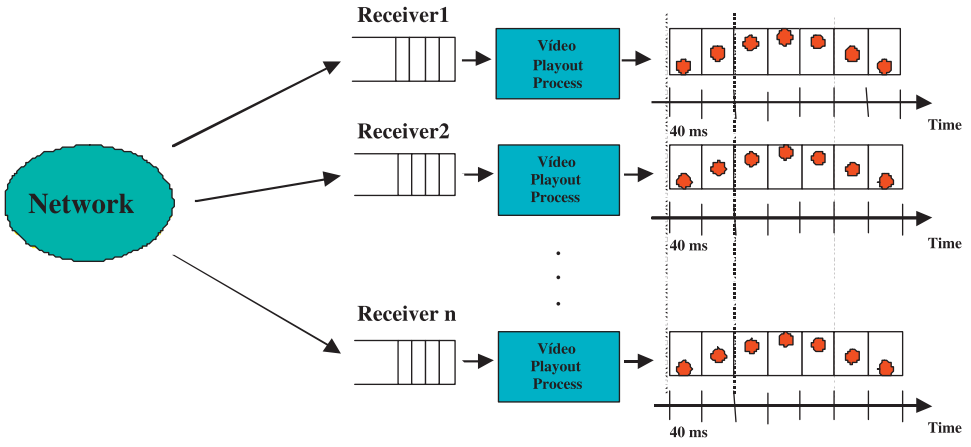


**Fig. 3.** Group synchronization.

on the following parameters, which can be tackled either individually or in an integrated manner [9]:

- *Network delays*: The delays experienced by the MDUs in the network to reach its receiver, which varies according to network load.
- *Network jitter*: It denotes the varying delay that stream packets experience on their way from the sender to the receiver network I/O device. It is introduced by buffering in intermediate nodes. It refers to the delay variations of inter-arrival of packets at the receiver because of varying network load.
- *End-system jitters*: Delay variations in presentation at the receiver because of varying workstation load and protocol processing delays. It refers to the variable delays arising within the end-systems, and is caused by varying system load and the packetizing and depacketizing of MDUs with variable size, which are passed through the different protocol layers.
- *Clock skew*: The clock time difference between the sender and the receiver.
- *Clock drift*: The rate of change of clock skew because of temperature differences or imperfections in crystal clocks.
- *Rate drift*: Change in generation and presentation rates because of server and receiver load variations.
- *Network skew*: Time difference in arrival of temporally related packets of streams, which is a differential delay among the streams.
- *Presentation skew*: Time interval in which the temporally related packets of the streams are presented.

Jitter is commonly equalized by the use of elastic buffers at the receivers (there are lots of buffering techniques).

Capture, reproduction and presentation of continuous media is driven by end-system clocks, but due to temperature differences or imperfections in the crystal clock, their frequencies can differ over a long period of time. The result is an offset in frequency to real time and to other clocks, which causes rate drifts. The problem of clock drift can be coped with by using time-synchronizing protocols within a network (for example, using the Network Time Protocol (NTP) or Global Positioning System (GPS) devices). Otherwise, if the problem of clock drift is neglected, buffer overflow (*flooding*) or buffer underflow (*starvation*) at the receiver will appear over a long period of time. The effect of clock drift is also known as *skew*, which is defined as an average jitter over a time interval.

Usually, networks are dynamic and have changing network conditions, not introduced by jitter, which are referred to a variation of connection properties (for example, an alteration of the average delay or an increasing rate of lost MDUs). These effects strongly depend on the QoS the underlying network can provide.

Apart from the network delay, MDUs also experience delay due to packetizing/depacketizing, the processing through the lower protocol layers, and the buffering at the transmitter and receiver sites. Fig. 4 shows the cumulative delay of two independent streams whose generation starts at the same time. Due to the different delays, the MDUs of the two streams (generated at the same time) would not be played out simultaneously if no synchronization techniques were used.

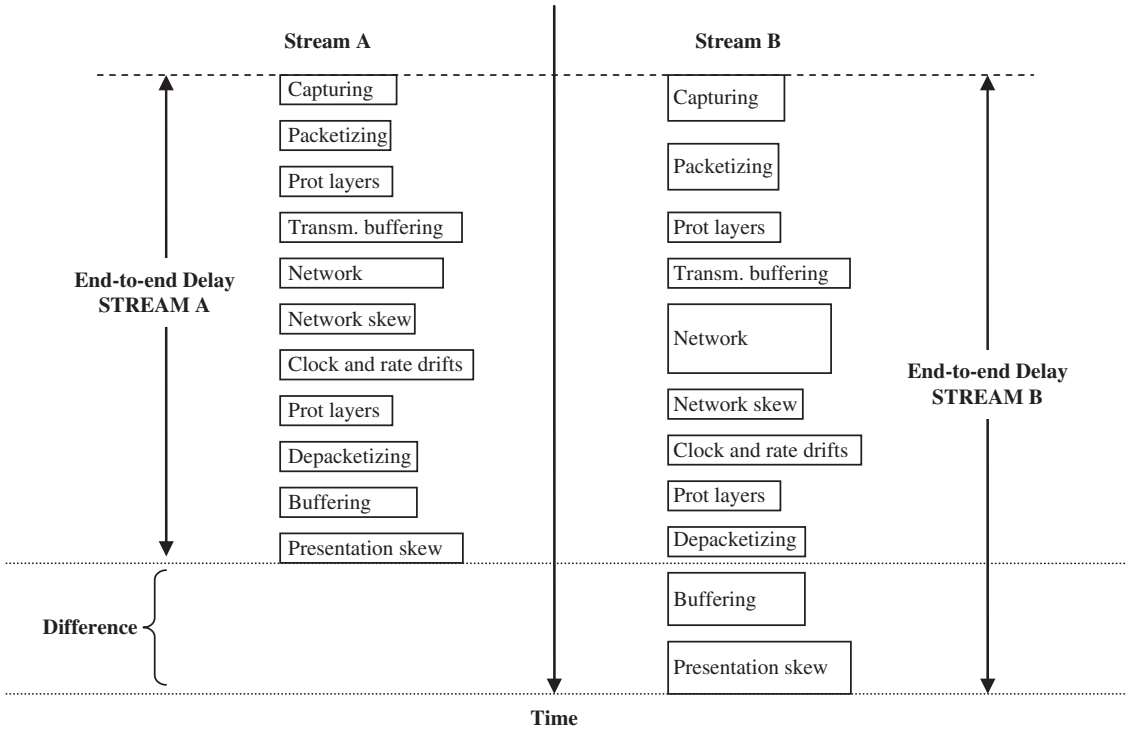


**Fig. 4.** Delays experienced by two independent media streams [8].

Synchronization mechanisms are needed to cope-up with all the above problems to ensure the time ordering of the stream/s and to maintain the presentation quality. Furthermore, the synchronization mechanism has to be adaptive regarding changes of the network conditions and to source drop-outs, which are a realistic assumption when using non-real-time operating systems.

There are lots of intra-stream synchronization solutions most of which try to avoid receiver buffer underflow and overflow problems. In [17], a comparative survey between intra-stream synchronization techniques can be found.

Over the last few years, new techniques have been developed to improve synchronization in multimedia systems, such as fixed and mobile agent-based techniques [9,18] and aspect-oriented programming-based techniques [19].

This paper is only focused on multimedia group and inter-stream synchronization techniques. We present the basic characteristics of them and analyze and compare qualitatively many of the solutions proposed in the past.

The rest of the paper is organized as follows. In the next section several past classifications found in the literature are presented which represent the basis of our classification. In Section 3, we present, as an example to make the understanding of the paper easier, a proposal developed by the authors to guarantee group and inter-stream synchronization using standard protocols. Section 4 presents the description and classification of the most common techniques for synchronization used by the studied solutions. In Section 5, a comparison table between those solutions (chronologically ordered, to indicate their appearance in time) is shown, according to several critical issues, very related to multimedia group and inter-stream synchronization. Finally, the paper ends by presenting our conclusions and the references.

## 2. Background

As far as we know, a common classification scheme for synchronization approaches does not exist. Most of the studied synchronization mechanisms are either application-specific or try to cover synchronization on a more abstract level independent of the application at hand. Surveys of multimedia synchronization mechanisms can be found in [8,16,20–23].

For this study, we have consulted the patterns in those references, followed by some authors in the past to compare multimedia synchronization techniques. In this work and henceforth the relation between algorithm/solution and techniques is the following: an algorithm can involve several control techniques.

Ehley et al. [21] present an early classification of the multimedia synchronization algorithms, grouping them in two synchronization schemes: distributed (in a network environment) and local (inside a workstation). Köhler and Müller [16] present another classification based on several factors: clocks (globally synchronized clocks and locally available clocks), synchronization mechanism location (it can be performed either at the sender or the receiver of continuous media information, but sender control requires some kind of feedback) and synchronization control techniques (only two are considered: clock frequency adjustment and skip and/or pause actions).

In [8,16], only three classification criteria (time, location and method) are used to summarize some solutions, which are systematized graphically in a 3D cube, with each criterion in a different axis. The solution space for playout synchronization consists of three almost orthogonal design criteria with two main choices in each dimension. The first decision is, whether the systems have an explicit common understanding of time or not. In the former case, some kind of clock synchronization takes place. The presentation time of a MDU can be calculated from an absolute or relative timestamp carried with every unit. If no clock synchronization takes place, playout synchronization can be achieved based on buffer control mechanisms. The second criterion is the location of synchronization actions (source or receiver). The third dimension distinguishes the methods that are used to correct asynchrony. Restoring synchronization could be done either by speeding up or slowing down the presentation or generation of MDUs, or by *stuffing*—well known method from bit or byte synchronization which means duplicating/deleting MDUs or pausing/skipping MDUs, respectively)

Perez-Luque and Little [23] develop a uniform, theoretical foundation for discussing multimedia synchronization and temporal specification. A temporal reference framework is proposed, focused on time models and it is used to compare some existing temporal specification schemes and their relationships to multimedia synchronization. That analysis explains why there are so many synchronization frameworks, how a multimedia scenario can be represented with different temporal specification schemes, and why some specification schemes cannot model all scenarios.

The classification presented by Ishibashi and Tasaka [22] is the most comprehensive one we have found and the one we have used as our starting point. In it only the intra-stream and inter-stream techniques used in each analyzed solution were taken into account, but address only 1:1 and $n$:1 communication.

In our comparative study, we have also included solutions for communications for 1:$n$ and $n$:$m$ communication and added new inter-stream synchronization solutions (previous (not considered in [22]) and later ones), such as those presented in [8,24–29,30,31] (some of them developed by the authors of this paper) [2,3,5,9,10–15,19,32–68]; and new versions of algorithms included in the original comparative survey, such as the ones in [5,38,69,70–75]. For example, in [69,72], the synchronization maestro scheme (SMS) for group synchronization, employed together with the virtual time rendering (VTR) media synchronization algorithm [76] has been enhanced so that the SMS scheme can be used efficiently in a P2P-based system and in a networked real-time game with collaborative work, respectively. Likewise, in [77], the scheme SMS and the distributed control scheme (DSC, defined in [47]), both used for group

synchronization, have also been enhanced, by taking into account the importance of the media objects.

In [20], the authors present (in Spanish language) a preliminary and shorter survey with 28 studied solutions. In this paper, we have completed and updated that survey with many new synchronization solutions (up to 53) and with new techniques and parameters to compare. In [22], we found eight algorithms that only included intra-stream synchronization. Those solutions and some others, which the authors have not been able to get, have not been included in this new study. Moreover, as an additional contribution, we have added other factors to the original comparative survey that we consider very relevant in the classification (for example, whether or not the solution uses a new specific protocol with new control messages or whether it uses a standard protocol, such as RTP/RTCP; whether it uses feedback from the receivers; the included group synchronization techniques, etc.). Another contribution has been the inclusion of the group synchronization approaches in the study (not included in [22]), such as the ones presented in [10,11,14,15,26,30,31,41–48,53,61,67–69,72,78]).

One of the main aims of the paper is to present only a qualitative comparative study, and not a quantitative comparative one, because the relations between the solutions we have found are not clear enough. One of the main reason is that the situations and environments those solutions have been developed for are very different. Moreover, no standard measurements have been defined to evaluate objectively the multimedia synchronization performance of the techniques. Nowadays, the quantitative relationships between the solutions seem quite chaotic.

## 3. Example of group and inter-stream synchronization algorithm

In this section, our proposal for multimedia group and inter-stream synchronization is qualitatively described as an example. More details regarding our solution and other components can be found in [30,31]. It is based on two previous protocols: the *Feedback Protocol* [79–84] and the *Feedback Global Protocol* [38]. The proposal, which uses the existing RTP/RTCP protocol suite (already used in most of the current multimedia applications for data transport and control), supposes the intra-stream synchronization is guaranteed by some technique (for example, one of the solutions classified in [17]) maintaining a correct and continuous playout of each multimedia stream (regardless of its nature), and includes two synchronization processes (group and inter-stream):

(1) *Group Synchronization*. This process has two phases: an *initial phase* in which all the receivers should start the playout of one of the streams (considered as the *master stream*), at the same instant (*Initial Playout Instant*); and then, the *second phase*, in which all the receivers should play that stream synchronously as continuously as possible.
(2) *Inter-stream Synchronization.* Inside each receiver there is an internal or local synchronization process between all the playout processes of all the streams the receiver is playing. This process is also called *multimedia fine inter-stream synchronization*. The playout processes of the other streams (considered as *slave*) will synchronize to the one of the master stream (also involved in the group synchronization process). This process will maintain the temporal relationships (the same as the ones existing at the generation time) between streams during the playout in each receiver.

A global time reference exists in all the sources and receivers (acquired, for example, using the NTP or GPS devices).

The set up is shown in Fig. 5. As it looks a bit confusing, we have divided it into several figures to make it clearer (Figs. 6–9).

In Fig. 6, we can appreciate the existence of a transmission (*multicast* or *unicast*), of several multimedia streams, using RTP (for data transmission) and RTCP (for control and feedback information transmission), from one or several sources to one or several receivers (our solution is general and covers the case in which a unique source sends multiple streams to one or several receivers). One of the streams has been chosen as the *master stream* (thicker lines and arrows) and, moreover, we have selected one receiver as the *master receiver* (dark in the figure), whose master stream playout point in every moment will be taken as the reference to determine the state (advanced or delayed form that one) of the playout of the other receivers (*slave receivers*). The master receiver can be selected taking into account several criteria beyond the scope of this paper [30,31].

Taking into account the RTCP feedback messages, the proposed synchronization algorithm, which will be executed at the master stream source (called *Synchronizer Source*), will use the feedback RTCP RR packets [85] which are extended [85] to include information useful for our algorithm, and new defined RTCP APP packets [85], used to determine and communicate the playout state of the master stream, both in and to all the receivers, respectively. In this way, the Synchronizer Source will be able to know the state of the playout processes during the session. This process is shown in Fig. 7.

The Synchronizer Source, when deviations (asynchronies) are detected (receivers whose master stream playout process is either advanced or delayed with respect to the master receiver's process) exceeding a threshold value, will generate action messages to make them correct the master stream playout, by skipping or pausing MDUs in the playout processes. This way the synchronized master stream playout in all the receivers (group synchronization, between receivers) is guaranteed. This process is shown in Fig. 8.

Once the group synchronization between receivers has been achieved regarding the master stream playout, the proposal also has to guarantee the local inter-stream synchronization (for example, *lip-sync*, [3–5]). For this purpose, an internal inter-processes communication channel can be used (for example, *mbus* [86], also used in [28]). In each receiver, and locally, the playout process of the master stream (synchronized with all the other

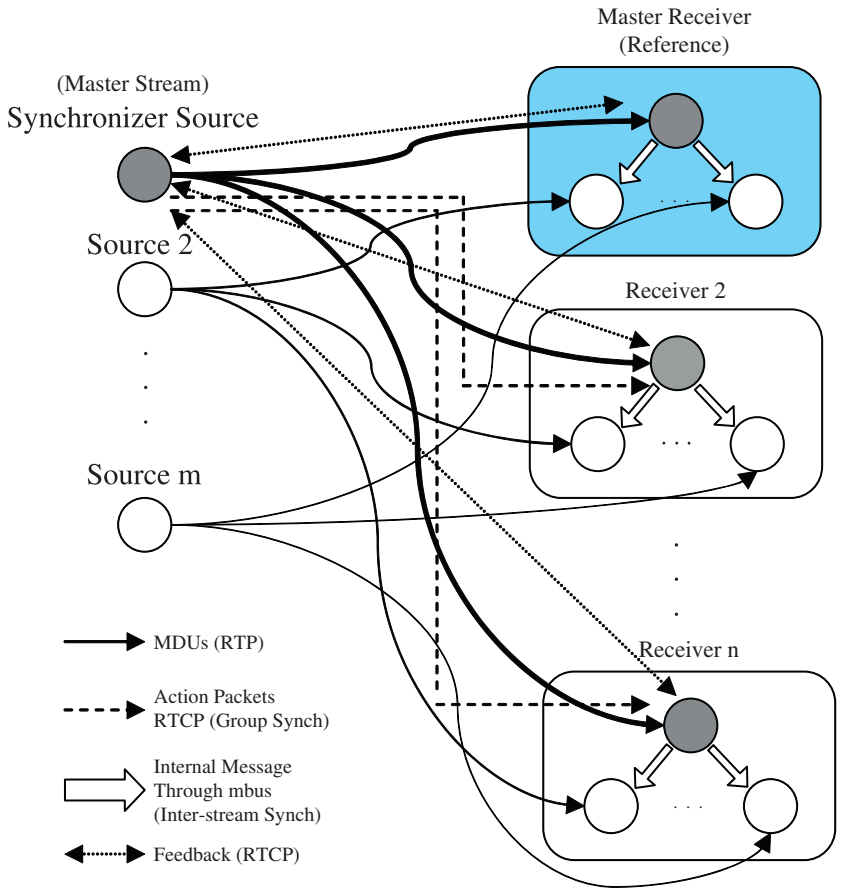


**Fig. 5.** Data and control streams.

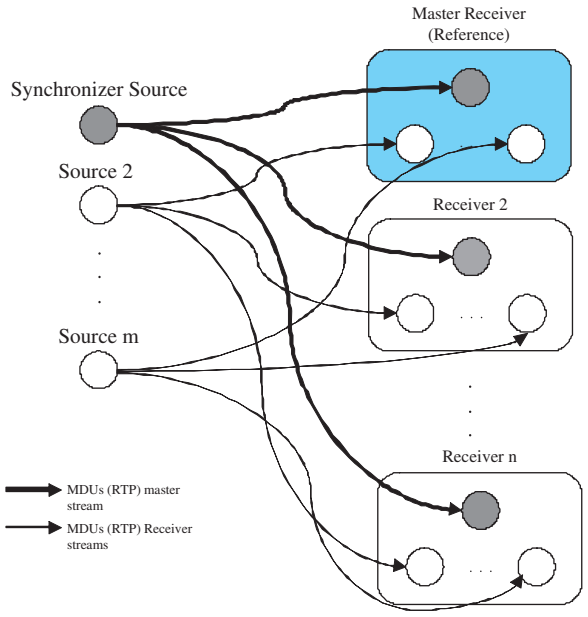


**Fig. 6.** Stream data transmission using RTP.

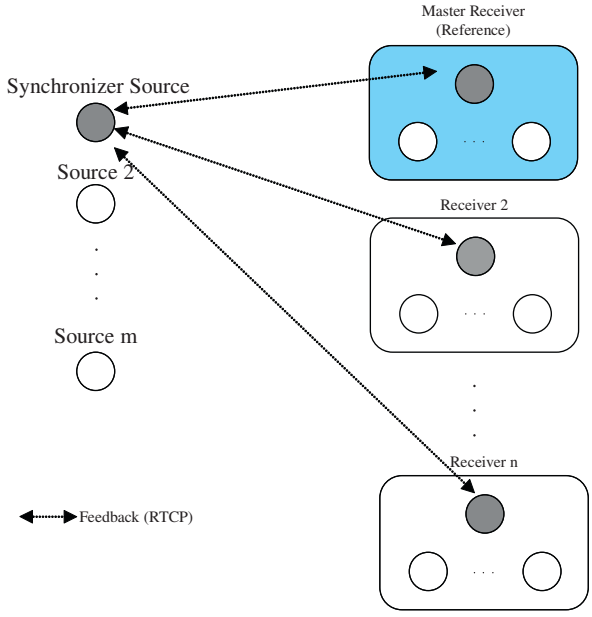


**Fig. 7.** Feedback messages (only for master stream).

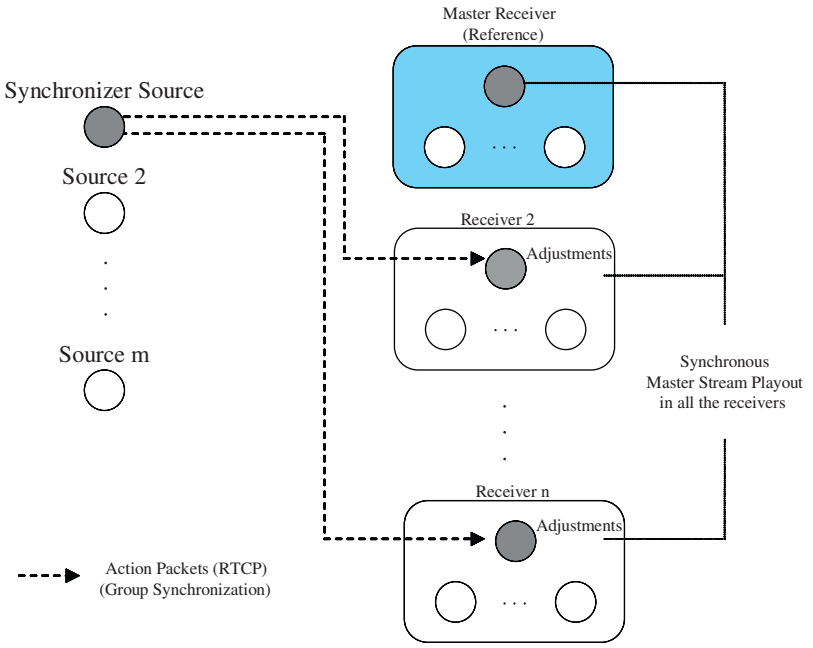


**Fig. 8.** Action RTCP messages sent by the Synchronizer Source to the slave receivers.

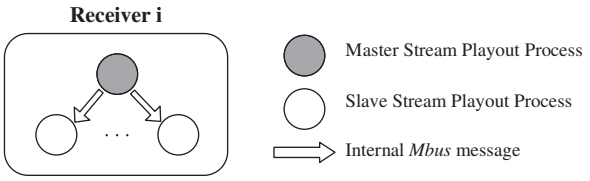


**Fig. 9.** Local inter-stream synchronization.

receivers using the above group synchronization process) will send its playout state periodically to the other slave streams' playout processes, to make them synchronize their playout states with synchronization actions, such as skips or pauses. This process is depicted in Fig. 9.

## 4. Synchronization techniques

Here, we present the main synchronization techniques we have found in the most representative solutions we have studied, classified into several categories. As several techniques can be included in a specific solution, each technique should be unique and indivisible (atomic), i.e. with no different functions from the multimedia synchronization point of view.

These techniques are summarized in Table 1. In Table 2, the main characteristics of all the studied synchronization solutions are presented, including the techniques each solution uses (right column).

The names of the techniques are presented in italics.

### 4.1. Specific techniques for group synchronization

Three of the most common techniques for group synchronization are: the *master/slave receiver scheme*, the *synchronization maestro scheme (SMS)* and the *distributed control scheme (DCS)*.

In the master–slave receiver scheme (initially presented in [78]), receivers are classified into a master receiver and slave receivers. None of the slave receivers send any feedback information about the timing of the playout processes. It adjusts the playout timing of MDUs to that of the master receiver. Only the master receiver sends (multicast) its playout timing to all the other (slave) receivers.

The SMS (initially presented in [42]) is based on the existence of a synchronization maestro (it can be the source or one of the receivers) which gathers the information on the playout processes from all the receivers and corrects the playout timing among the receivers by distributing control packets. In order to do this, each receiver sends (unicast) the information to the maestro, and the maestro sends (multicast) the corrected playout timing. The author's solution, presented in Section 3, follows this scheme.

In the distributed control scheme (initially presented in [47]), all the receivers can exchange (multicast) the control packets or use timestamps in media packets to calculate playout delays to achieve group synchronization. Each receiver decides the reference playout timing from among the output timing of itself and that of the other receivers. In [47,48], a distributed control scheme is proposed, which adaptively keeps the temporal and causal relationships according to the network load under distributed control. For group synchronization, the scheme adopts a group synchronization algorithm, which is called the *distributed control scheme*. In [11], a bucket synchronization mechanism is presented. In [53], the use of a local-lag and a timewarp algorithm is proposed to avoid inconsistencies between users in a replicated consinuous application (e.g., a network game application).

**Table 1**
Synchronization techniques

| Location | Technique | Technique's purpose |
|---|---|---|
| Basic Control | Source control | Add information useful for synchronization (timestamps, sequence numbers (identifiers) event information and/or source identifiers.) |
| | Receiver control | Buffering techniques |
| Preventive Control | Source control | Initial playout instant calculation<br>Deadline-based transmission scheduling<br>Interleave MDUs of different media streams in only one transport stream |
| | Receiver control | Preventive skips of MDUs (eliminations or discardings) and/or preventive pauses of MDUs (repetitions, insertions or stops)<br>Change the buffering waiting time of the MDUs<br>Enlarge or shorten the silence periods of the streams |
| Reactive Control | Source control | Adjust the transmission timing<br>Decrease the number of media streams transmitted<br>Drop low-priority MDUs |
| | Receiver control | Reactive skips (eliminations or discardings) and/or reactive pauses (repetitions, insertions or stops)<br>Make playout duration extensions or reductions (playout rate adjustments)<br>Use of a virtual time with contractions or expansions<br>Master/slave scheme (switching or not)<br>Late event discarding (Event-based)<br>Rollback techniques (Event-based) |
| Common Control | Source control | Skip or pause MDUs in the transmission process<br>Advance the transmission timing dynamically<br>Adjust the input rate<br>Media Scaling |
| | Receiver control | Adjust the playout rate<br>Data interpolation |

We can also classify the above schemes into centralized and distributed control schemes. The former schemes have their own advantages and disadvantages. For instance, they can more easily preserve causality and there is less possibility of inconsistency of state among the session members (receivers) occurring than with distributed control schemes. However, they have larger network delays, lower reliability and poorer scalability. Therefore, distributed control is also desirable for causality and media synchronization. In [47], the advantages and disadvantages of the *SMS* and the *distributed control scheme* are discussed in terms of reliability, control speed, control overhead, etc. In [77] both schemes for group synchronization have been enhanced by taking into account the importance of the media objects, for its application in networked virtual environments. In that work, the concept of *global importance* (importance which is judged from a point of view of all the users (receivers) in the virtual environment) is introduced in addition to the local importance (importance which is judged from a viewpoint of each user and used to change the intra-stream and inter-stream synchronization accuracy).

In [69,72] the SMS for group synchronization, employed together with the VTR media synchronization algorithm [76] has been enhanced so that that scheme can be used efficiently in a P2P-based system and in a networked real-time game with collaborative work, respectively. Likewise, in [77] the SMS and the DSC schemes, both used for group synchronization, also have been enhanced, by taking into account the importance of the media objects.

In [87], the three above schemes, based on the VTR algorithm, have been evaluated in a quite simple Multicast Mobile Ad Hoc Network (MMAHN).

### 4.2. Generic synchronization techniques

Although many ways to classify the synchronization techniques can be found, we have chosen, and extended, the ones described by Ishibashi and Tasaka [22] and Liu [50], based on each techniques' purpose and the locations at which they are employed. We classify the synchronization techniques into four groups:

(a) *Basic control techniques*, needed in most of the solutions and essential to preserve the temporal structures.
(b) *Preventive control techniques*, needed to avoid the asynchrony (situation of out of synchrony), before it appears.
(c) *Reactive control techniques*, needed to recover from asynchrony after it has been detected.
(d) *Common control techniques*, which can be used for both prevent (prevention) and/or correct (reaction) situations of asynchrony.

Any one of these techniques, either alone or in combination with others, can be employed to achieve the desired synchronization for a targeted application. In most of the solutions, the Basic Control techniques are usually complemented, at the same time, with preventive and/or reactive control techniques. The preventive control techniques cannot usually avoid completely the appearance of asynchrony, so the combination with reactive control techniques is also needed.

Generally, the above synchronization techniques can be applied at the source (multimedia server) and/or the receiver. On the one hand, some control techniques are always needed at the receiver side because of the existence of network jitter. On the other hand, control techniques at the source side need *feedback* information from the receivers or from the network to let the source/s know the synchronization and/or network QoS in each moment to proceed consequently. In some techniques, source/s and receivers cooperate to control the synchronization processes. We divide the above four techniques into two groups *according to their location* (source or receiver), as in [16]. In some solutions for mobile

**Table 2**
Multimedia inter-stream and group synchronization solutions

| NAME | Clock | Delay limits | MDU generation | Stored or live contents | Intra, inter and group synchronization | Group syncronization techniques | *Master/Slave (M/S)* | *Feedback utilization* | Synchr. information | Location | RTP use | Synchronization techniques |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ACME [93] | Global | – | Periodical | Both | Intra and inter | – | M/S streams and without relationship | – | Timestamps | Receiver | No | Reactive skips and pauses<br>Virtual time expansion<br>Transmission rate adjustments<br>Playout rate adjustments<br>Data interpolation |
| [91,92] | Global or local | – | Periodical | Stored | Inter | – | Without relationship | Yes (but not used for synch.) | Seq. number | Receiver | No | Decreasing the number of media streams<br>Preventive and reactive skips (discardings) and pauses (duplicates) to change the playout rate<br>MDU insertions |
| [95] | Global | Known | Periodical | Live | Inter | – | Without relationship | No | Seq. number | Source and receiver | No | Decreasing the number of media streams<br>Reactive skips and pauses |
| FP [79–84] | Local | Known | Periodical | Stored | Intra and inter | – | M/S streams | Yes | Timestamps Seq. number | Source and receiver | No | Skips and pauses at source side |
| MCP [67,68] | Global | – | Periodical and no Periodical | | Inter and group | Token-based | – | No | Timestamps Seq. number context information | Receiver | No | Skips and pauses at receiver side<br>MDU discarding |
| [18,57,58] | Local | Unknown | Periodical and no Periodical | Stored | Intra and inter | – | Without relationship | No | Seq. number | Source and receiver | No | Initial transmission and playout instants<br>Deadline-based transmission scheduling<br>Insertion of synchronization points (coarse synchronization)<br>Reactive skips and pauses |
| Fsp [36] | Global | Unknown | Periodical and no Periodical | Both | Inter | – | – | No | Timestamps | Receiver | No | Playout duration extensions or reductions |
| VTR [70,71,73,74,76,78,90] | Global and local | Unknown | Periodical and no Periodical | Both | Intra and inter and group [78] | M/S receiver scheme [78] | M/S streams | Yes | Timestamps Seq. number | Source and receiver | In [70,71,74] | Change of the buffering time according to the delay estimation<br>Decreasing the number of media streams<br>Preventive pauses<br>Reactive skips and pauses<br>Skips at the source side<br>Playout duration extensions or reductions<br>Virtual local time expansions or contractions<br>Media scaling<br>Interleaving MDUs [73,74] |

Table 2 (*continued*)

| NAME | Clock | Delay limits | MDU generation | Stored or live contents | Intra, inter and group synchronization | Group syncronization techniques | *Master/Slave (M/S)* | *Feedback utilization* | Synchr. information | Location | RTP use | Synchronization techniques |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ASP [2,62] | Global | – | – | Both | Intra and inter | – | M/S streams | – | Timestamps | Receiver | No | Initial transmission and playout instant<br>Reactive skips and pauses (duplicates)<br>Change of the buffering time according to the delay estimation<br>Playout duration extensions or reductions<br>Master/slave switching<br>Playout rate adjustment |
| Concord Algorithm [63] | Global and local | Known | Periodical – | – | Intra and inter | – | M/S streams and without relationship | – | Timestamp in 1st packet or especial packets | – | No | MDU discarding<br>Pauses in the slave stream playout |
| [94] | Local | Unknown | – | Both | Intra and inter | – | M/S streams | Yes | Timestamps Seq. number | Source and receiver | No | Transmission and playout rate adjustments<br>Skips and pauses at source site<br>Reactive skips and pauses<br>Playout duration extensions or reductions<br>Late MDU discarding |
| [26] | Global | Known | – | – | Intra, inter and group | Distributed control scheme | M/S streams<br>M/S receivers (chairman) | – | Timestamp in 1st packet | – | No | Initial transmission and playout instant<br>Playout rate adjustments (receiver's clock)<br>Master/slave receiver swhitching (chairman) |
| [27] | – | Known | – | Stored | – | – | Without relationship | No | Playout deadline | Source | No | Deadline-based transmission scheduling |
| [8,29] | Local | Unknown | – | Stored | Intra and inter | – | Without relationship | Yes | Timestamps Seq. number | Source and receiver | Optional. | Initial transmission instant synchronization to achieve, subsequently, the initial playout instant synchronization<br>Playout duration extensions<br>Preventive skips and pauses at source side |
| MultiSynch [96] | – | – | Periodical | – | Intra and inter | – | Without relationship | – | Timestamps | Receiver | No | Reactive skips (discardings) and pauses<br>Low priority MDU dropping |
| [4] | Global | Unknown | Periodical | Live | Intra and inter | – | M/S streams and without relationship | – | Timestamps | Receiver | Yes | Master/slave switching<br>Playout duration extensions or reductions |
| SSP [89] | Local | Known | Periodical and no Periodical | Stored | – | – | – | Only at the initial stage | Timestamps | Source and receiver | No | Initial playout instant<br>Deadline-based transmission scheduling |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | Low priority MDU dropping<br>Preventive skips (selective discardings) and pauses (duplicates) or deliberated delays<br>Virtual time expansion |
| [6] | Local | Unknown | Periodical and no Periodical | Without distinction | Intra and inter | – | Without relationship | – | Timestamps | Receiver | No | Reactive skips (discarding) and pauses |
| [99] | Global | Known | Periodical | Stored | Intra and inter | – | M/S streams and without relationship | – | Timestamps (in some MDUs) | Source and receiver | No | Reactive skips and pauses at source and/or receiver side |
| [64] | Local | Unknown | – | Both | Inter | – | – | Yes | Timestamps | Source (Synchronization Server) | No | Skips and pauses at source side |
| MSTP [100] | Local | Unknown | Periodical and no Periodical | Both | Inter | – | M/S streams | – | Seq. number. Seq. number of the master stream MDU to synchronize with | Receiver | No | Reactive skips and pauses in slave streams' playout<br>Slave streams MDU discarding<br>Late MDU discarding |
| [101] | Local | Unknown (maximum Jitter) | Periodical | – | Inter | – | Without relationship. | Yes | Timestamps Seq. number | Source | No | Skips and pauses (noise MDU insertion) at source side |
| [98] | Local | Known | Periodical | Both | Intra and inter | – | M/S streams | No | – | Receiver | No | Reactive skips and pauses<br>Master/slave switching |
| SMS [14,15,41–46,72] | Global | Known | Periodical and no Periodical. | Both ([78] for stored and [41] for live) | Group (VTR is used for Intra and inter) | Synchronization maestro scheme | M/S streams M/S receivers | Yes | Timestamps Seq. number | Receiver | No | VTR [76] techniques<br>Initial transmission instant (only in [42]) |
| [5,60] | Local | – | Periodical | Stored | Inter | – | M/S streams | Yes | Timestamps | Source and receiver | No | Initial transmission rate adjustment<br>Playout rate adjustments<br>Skips and pauses<br>Slave stream's MDU discarding |
| [12–14] | Local | Unknown | Periodical and no Periodical | Stored | Intra and inter | – | M/S streams | Yes | Seq. number | Receiver | No | Initial playout instant<br>Late and useless/ redundant MDU discarding<br>Reactive skips and pauses in slave streams' playout<br>Master/slave switching |
| [97] | Local | Unknown | Periodical | Both | Intra and inter | – | M/S streams | Yes | Seq. number | Receiver | No | Playout rate adjustments<br>Playout duration extensions or reductions<br>Reactive skips (discardings) and pauses (duplicates) |
| [88] | – | – | Periodical | Both | Intra and inter | – | M/S streams | No | Seq. number Synchr. marker | Receiver | No | Reactive skips and pauses<br>Playout rate adjustments |

Table 2 (*continued*)

| NAME | Clock | Delay limits | MDU generation | Stored or live contents | Intra, inter and group synchronization | Group syncronization techniques | *Master/Slave (M/S)* | *Feedback utilization* | Synchr. information | Location | RTP use | Synchronization techniques |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BS [11] | Global | Known | Periodical and no Periodical | Live | Inter and group | Distributed control scheme | Without relationship | No | Timestamps Seq. number | Receiver | Yes | Skips (discardings) and pauses (duplicates)<br>Late events are dropped |
| [54–56] | Global | Unknown | Periodical and no Periodical | Stored | Intra and inter | – | – | Yes | Timestamps Seq. number | Source and receiver | Yes | Transmission (or playout) rate adjustments<br>Low priority MDU dropping.<br>Late MDU discarding.<br>Playout rate adjustments.<br>Dynamic playout scheduling |
| [7] | Local | Unknown | Periodical and no Periodical | Live | Intra and inter | – | M/S streams | No | Timestamps | Receiver | No | Reactive skips and pauses<br>Virtual time expansion and contraction<br>Master/slave switching |
| DCS [47,48] | Local | Unknown | Periodical and no periodical | Both | Group (VTR is used for Intra and inter) | Distributed control scheme | M/S streams | No (Timing information exchange between receivers) | Timestamps | Receiver | No | VTR [76] techniques<br>Playout rate adjustments |
| PARK [39,40] | Local | Unknown | Periodical and no periodical | Stored | Intra and inter | – | M/S streams | Yes | Seq. number | Source and receiver | No | Initial transmission instant<br>Transmission rate adjustments<br>Late slave streams' MDU discarding<br>Reactive skips and pauses in slave streams' playout<br>Master/slave switching |
| [25] | – | Unknown | Periodical and no periodical | Stored | Intra and inter | – | – | No | – | Source | – | Deadline-based transmission scheduling<br>Initial transmission instant<br>Transmission rate adjustments.<br>Low priority MDU dropping (by source) |
| [37] | Global or local | Known | Periodical and no periodical | Live | Intra and inter | – | – | Only at the initial stage | – | Receiver | No | MDU discarding<br>Playout duration extensions or reductions (reduction of the silent periods duration and 'Gap' insertion) |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RVTR [75] | Global | Unknown | Periodical and no periodical | Both | Intra and inter | – | M/S streams | Yes | Timestamps Seq. number | Source (retrans.) and receiver | Yes | VTR [76] techniques<br>Playout duration extension or reduction |
| FGP [38] | Global | Known | Periodical | Both | Intra and inter | – | M/S streams | Yes | Timestamps Seq. number | Receiver | No | Reactive skips and pauses |
| [49] | Global | Unknown | Periodical | Live | Intra and inter | – | – | Yes | Timestamps Seq. number | Receiver | Yes | Playout rate adjustments |
| RTP-FGP [30,31] | Global | Unknown | Periodical and no periodical | Both | Inter and group | Synchronization maestro scheme | M/S streams<br>M/S receivers | Yes | Timestamps<br>Source id. | Source and receiver | Yes | Initial playout instant<br>Reactive skips and pauses at the receiver side<br>Playout rate adjustment<br>Virtual time expansion<br>Master/slave receiver switching (group synchronization) |
| ALPSMS [66] | Global | Unknown | Periodical and no periodical | Both | Inter | – | Without relationship | – | Timestamps Seq. number | Receiver | No | Change of the buffering time according to the delay estimation<br>Late MDU discarding |
| [35] | – | Unknown | Periodical and no periodical | Both | Inter | – | M/S streams | Yes | Event number<br>Timestamps | Source and receiver | – | Event-based synchronization control (no time-based control is used)<br>Late MDU discarding |
| MoSync [32–34] | Local | Unknown (but Jitter bounded) | Periodical and no periodical | Both | Intra and inter | – | Without relationship | Yes | Timestamps Seq. number server/ source number | Source, network base station and mobile receiver | – | Initial transmission instant<br>Deadline-based transmission scheduling (achieved by adopting various transfer rates)<br>Dynamic playout scheduling |
| [50–52] | Local | Unknown | Periodical | Live | Intra and inter | – | M/S streams | No | Timestamps | Receiver | Yes | Virtual local time expansions or contractions<br>MDU playout duration extensions or reductions<br>Reactive skips |
| LSA [3] | – | – | Periodical | Both | Inter | – | M/S streams | No | Timestamps | Receiver | – | Playout rate adjustments |
| LL-TW [53] | Global | Unknown | Periodical and no periodical | Live | Group | Distributed control scheme | – | No | Timestamps | Receiver | – | Event-based synchronization control<br>Playout duration extension<br>Rollback-based techniques |
| [19] | Global | Unknown | Periodical and no periodical | Both | Intra and inter | – | – | No | Timestamps Seq. number | Receiver | Yes | Dynamic playout scheduling<br>Late MDU discarding |
| TSS [10] | Global | Unknown | Periodical and no periodical | Live | Inter and group | Distributed control scheme | – | No | Timestamps | Receiver | – | Event-based synchronization control<br>Playout duration extension<br>Rollback-based techniques |

Table 2 (*continued*)

| NAME | Clock | Delay limits | MDU generation | Stored or live contents | Intra, inter and group synchronization | Group syncronization techniques | *Master/Slave (M/S)* | *Feedback utilization* | Synchr. information | Location | RTP use | Synchronization techniques |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ILA [61] | Global | Unknown | Periodical and no periodical | Both | Group | Distributed control scheme | – | No | Timestamps | Receiver | – | Event-based synchronization control<br>Preventive MDU/event discarding<br>Reactive events discarding |
| [9] | Local | Unknown | Periodical and no periodical | Both | Intra and inter | – | – | No | Timestamps Seq. number | Receiver | – | Initial transmission and playout instants<br>Reactive skips and pauses (silences for audio)<br>Playout duration extensions or reductions<br>Late MDU discarding |
| [24] | Local | Unknown | Periodical and no periodical | Stored (video and metadata) | Inter | – | M/S streams | No | Synchronization information in MDUs | Source and receiver | – | Periodical insertion of synchronized information in the video stream to achieve media synchronization of video and metadata<br>Skips and pauses |
| ESMS [69] | Global | Known | Periodical and no periodical | Both | Intra and group (VTR for intra) | Synchronization maestro scheme | M/S receivers | Yes | Timestamps Seq. number | Source and Receiver | No | Skipping and VTR [76] techniques for intra-stream synchronization<br>Same techniques than in SMS [14], but with two reference output timings and different methods |
| [28] | Global | Unknown | Periodical | Live | Intra and inter | – | M/S streams | No | Timestamps | Receiver | Yes | Preventive modification of the length of the silent periods for intra-stream synchronization<br>Coarse synchronization at the beginning of a talkspurt<br>Playout duration extensions or reductions<br>M/S streams switching |
| ECSA [65] | Global | Unknown | Periodical and no periodical | Live | Inter | – | M/S streams | No | Event information<br>Timestamps | Source and receiver | – | Event-based synchronization control<br>Rollback-based techniques |

ACME, abstractions for continuous media; ALPSMS, application-level protocol for synchronized multimedia sessions; ASP, application synchronization protocol; BS, bucket synchronization; DSC, distributed control scheme; ECSA, event correlation synchronization algorithm; FGP, feedback global protocol; FP, feedback protocol; FSP, flow synchronization protocol; ILA, interactivity-loss avoidance; LL-TW, local-lag and timewarp algorithms; LSA, lip-synchronization algorithm; MCP, multi-flow conversation protocol; MoSync, mobile synchronization algorithm; MSTP, multimedia synchronization transport protocol; PARK, paused-and-run k-stream multimedia synchronization control scheme; RTP-FGP, RTP-based feedback-global protocol; RVTR, retransmission with VTR; SMS, synchronization maestro scheme; SSP, stream synchronization protocol; TSS, trailing state synchronization; VTR, virtual time rendering algorithm.

environments, some techniques described for sources in this paper are used by local base stations in the cells where the mobile receiver is located (as in [32–34]).

#### 4.2.1. Basic control techniques

Basic control, which consists of appending synchronization information (timestamp, sequence number, etc.) to MDUs and buffering the data at the receiver side, is essential for all algorithms. The following techniques have been found in all the solutions we have found.

(a) Source control
The basic control techniques executed by the source can consist of introducing some *information useful for synchronization* in the headers of the MDUs, such as *timestamps*, *sequence numbers* (*identifiers*), *sequence marking* (streamlined time stamps), *event information and/or source identifiers* (see column '*Synchr. Information*' in Table 2). The use of *timestamps* is not needed when, for example, the generation of MDUs is periodical, and the use of only sequence numbers would be sufficient.
Moreover, in some cases, the source can include temporal or event marks to force the inter-stream resynchronization at specific instants of time in the playout process (we call this process *Coarse* [38,57–59,76,88] or *Event* [10,11,35,53,61,65] *synchronization*). In the example presented in Section 3, the solution uses timestamps and sequence numbers, included in RTP/RTCP packets' headers.

(b) Receiver control
Nearly all the solutions use *buffering techniques* at the receiver side. The reception buffers are used to keep MDUs until their playout instants arrive, according to certain synchronization information, and to smooth out the effects of the network jitter.

#### 4.2.2. Preventive control techniques

Preventive control consists of techniques used to avoid asynchrony. We have found the following ones, according to their location.

(a) Source control
For stored media content, source usually will be able to transmit MDUs according some synchronization information (for example, timestamps). This technique is used in most of the studied solutions. The source can draw up a schedule for transmission according to deadline times [25,27,32–34,57–59,89], but this technique is only valid for stored content transmission. We call this technique *deadline-based transmission scheduling*. If the source is able to know, for each MDU, its size, deadline transmission and the network delay limits (maximum and minimum delays or at least the probability distribution function of the delay), it will be able to schedule the transmission of MDUs according to those temporal requirements. In [32] both the server and local base station (wireless environment) schedule the transmission of MDUs. Boukerche et al. propose that the closest base station can schedule the packets for the playout of MDUs on the mobile receiver in several queuing policies (a FIFO [32], PQ, RR or WFQ [34] orders) according to the network conditions and applications' requirements.
Another technique to prevent asynchrony consists of the *initial transmission and/or playout instant calculation*. As we explained above (example in Section 3), the source can prevent from an initial asynchrony situation at the starting of the playout of the different media streams. It can do it by calculating the *initial playout instant* of the presentation (common for all the receivers and streams) and communicating it to all the receivers before the transmission of media streams starts. So, all the receivers will start the playout at the same time. The initial playout instant calculation technique has been used in [2,8,9,12–14,26,29–31,57–59,62,89]. The initial transmission instant calculation technique has been used in [2,8,9,25,29,32–34,39,40,47,57–59,62].
The source can also use a technique based on *interleaving MDUs* of different media streams in only one transport stream (as in [73,74]). This technique improves the inter-stream synchronization quality but may degrade the intra-stream synchronization quality in those streams sensitive to network jitter.

(b) Receiver control
In some cases, the receiver's playout process will do *preventive skips of MDUs (eliminations or discardings) and/or preventive pauses of MDUs (repetitions or insertions)* depending on the playout buffer's size [16,61,70,71,73,74,76,78,89–92]. It is also possible to insert dummy (noise) data, instead of '*pausing*' (or stopping) the playout process. When using multilevel coding systems, such as MPEG, the receiver can discard some MDUs with low priority (e.g., B-frames in MPEG), according to the receiver buffer occupation.
If the receiver can estimate the network delay experienced by the MDUs, it may *change the buffering waiting time of them* [76,2,62,66]. Some authors propose to enlarge or shorten the silence periods of the audio streams of the applications [28]. This technique is not valid for musical applications, where the silences are as important as the sounds.

#### 4.2.3. Reactive control techniques

Reactive control is used to recover synchronization after asynchrony occurs. Approaches such as reactive skipping and pausing, shortening/extending playback duration and virtual local time contraction or expansion (referred as virtual local time control hereafter) all belong to reactive control. We have found the following reactive techniques:

(a) Source control
On the one hand, the source can *adjust the transmission rate (timing)* changing the transmission period. If the source is capable of knowing the existing asynchrony between the streams' playout processes

[5,9,25,39,40,54–56,60,93,94], it will be able to change the transmission timing (for example, by changing the transmission period or by skipping or pausing MDUs). For example, in the PARK[3] approach [39,40], according to a TCP-like scheme, (a) the source could quickly decrease its transmission rate when network congestion is detected in order to release the congestion situation; (b) the source could slowly increase its transmission rate when the network congestion is released in order to utilize available bandwidth as much as possible.

On the other hand, if the source detects that recovery of synchrony is difficult for the receiver, it can *decrease the number of media streams transmitted* [70,71,73,74,76,78,90–92,95]. For example, in audio and video synchronization case, if asynchrony is detected and this situation persists, the source could stop the transmission of the video stream temporarily, and, so, when it detects the receiver has recovered, the source could restart the transmission of the video stream.

When using multilevel coding systems, such as MPEG, the source can *drop low-priority MDUs* (for instance, B-frames in MPEG), as in [4,25,54–56,89], according to some QoS parameters, such as the network congestion or MDU loss rates.

(b) Receiver control

From the receiver point of view, it can take some actions for recovering from detected asynchrony situations. On the one hand, the most popular technique, due to its easy implementation, and used by the solution described in Section 3, consists of *reactive skips (eliminations or discardings) and/or reactive pauses (repetitions or insertions).* This technique has been used by many authors. As an example, we can cite the case in which a receiver detects the playout point of the MDU it is processing has expired, because it has arrived too late. Then, it can choose between to playout it and discard consecutive MDUs already received (for example, [2,7,15,62,70,71,73–76,78,90,93,95]), or to discard it directly (for example [9,12–14,19,30,31,35,39,40,54–56,66,96]). For audio streams, to deal with loses and delayed MDUs, a solution can be not to play anything [9]. In [66], the MDUs of several streams sent at the same instant (with the same timestamp) are considered as a synchronous group, so the playout of a MDU of a stream is *stopped* until the MDUs of other streams in the same group do not reach the receiver.

Experimental experience has demonstrated that abrupt skipping or pausing can result in playout gaps. Such gaps may dramatically degrade presentation effectiveness, especially when lip-synchronization is involved.

Some authors (for example, [57–59]) use the following nomenclature: blocking and restricted blocking policies. In a **Blocking Policy**, for slow streams, i.e. streams in which the MDUs do not arrive at the receiver in time to meet their respective playout deadlines, the playout process can be *blocked* or *suspended*, until the late MDU arrives. In a **restricted blocking policy**, the playout process blocks for a pre-specified period of time only, while it waits for the current MDU to arrive. After the waiting period with non-arrivals of late MDUs, it may playout the most recent stored MDUs, skipping the late MDUs altogether.

On the other hand, when receiver buffer starvation is detected, to avoid playout gaps, the receiver can opt to *playout repeatedly the last MDU* until the next one arrives [30,31,57–59] or to make *playout duration extensions or reductions* (*playout rate adjustments*) rather than abruptly skipping or pausing the presentation. To gradually recover from an asynchrony situation without degrading the playout quality (not noticed by application end users), the receiver can shorten or extend the playout duration of each MDU until the synchronization has been recovered [2–5,8–10,26,28–31,36,37,41,48–56,60,62,70,71,73–76,78,88,90,93,94,97]. If the duration is shortened, it implies an acceleration in the playout (fast-forwarding), but without skips of MDUs, meanwhile an extension of the duration implies to slow down the playout (but without MDUs repetitions). This technique is also used to adapt the playout of a stream to the playout of another stream (inter-stream synchronization). In the case of voice streams, and in specific applications, only the playout of the MDUs of silent periods could be shortened or extended to affect as little as possible to the quality perceived by the users.

The extension of the playout duration technique is similar to the use of preventive pauses, because the latter can be made by enlarging the output or playout time of the MDUs. Nevertheless, the former is a reactive technique while the latter is preventive.

Some authors propose the *use of a virtual time with contractions or expansions* to get the desired synchronization. This way the MDUs are played using a virtual time axis different from the real-time axis. This technique has been used in many proposals [7,15,22,30,31,50–52,70,71,73–76,78,89,90,93]. Virtual time expands or contracts according to the amount of delay jitter of the received MDUs.

In [3], *timestretching* the audio at the beginning of each utterance[4] is proposed for lip-synchronized videoconference systems. The audio stream can be time-stretched by re-sampling and interpolating the original audio packet.

For example, in Fig. 10, the playout of stream A and stream B becomes asynchronous at instant $t_0$ after MDU 30. The receiver recovers resynchronization control at point $t_1$ such that the presentation becomes synchronous when MDU 37 of both streams is being played out at the same time. The stream A playout process has played

[3] *Paused-and-run k-stream multimedia synchronization control scheme.*

[4] The beginning of an utterance is defined as the moment when the audio volume exceeds a silence threshold, the maximum measured audio volume when the user is not talking. The end of an utterance is defined as the moment when the audio volume is less than the silence threshold [3].

out duplicated MDUs and has adjusted the playout rate, whereas the stream B playout process has stopped its playout during a short interval of time (silence).

Normally, to obtain an inter-stream synchronization, a *master/slave scheme* is used, defining one of the streams as the *master stream*, and considering the others as the *slave streams* (as in the example described in Section 3 [2–5,7,12–14,24,26,28,30,31,35,38–41,48,50–52,60,62,63,65,69–71,73–76,78–84,88,90,93,94,97–100]. There are some proposals in which the master and slave roles are exchanged dynamically, allowing master–slave switching [2,4,7,12–14,28,30,31,39,40,62,77,98]. This way, when the asynchrony of a slave stream exceeds a threshold value, the receiver can switch the roles, and consider that stream as the new master one, making the needed required adaptations. In [77], the master role is switched between streams according to their *global importance* in virtual environments.

Also in group synchronization, a master/slave schema can be used but regarding the receivers [15,26,30,31,43,78]. The master receiver playout timing is taken as the reference for updating the playout timings of the other receivers (slaves). As in the above case, in some algorithms the roles can be switched between receivers [26,30,31].

Fig. 11 shows an example of a master/slave inter-stream synchronization control. When the playout process of the slave stream ends its playout at a synchronization point and the playout process of the master stream has not finished yet, the playout process of the slave stream pauses, repeating the playout of the last MDU, or stopping its playout (blocking) until the master stream playout process finishes (Fig. 11a). When a slave stream playout process has not played some MDUs (because they have not arrived yet or because they arrived too late), and the master stream playout process has already finished its playout at a synchronization point, the slave stream playout process discards the late MDUs to maintain synchronization with the master stream playout process (Fig. 11b).

Ishibashi et al. [77] carried out subjective and objective assessment of the lip-synchronization quality of nine receiver-based synchronization control schemes, which consist of combinations of the following four reactive control techniques: skipping, discarding, shortening and extension of output duration, and virtual time contraction and expansion. In this paper, those nine schemes are used for inter-stream synchronization purposes, whereas the VTR algorithm [76] is used for intra-stream synchronization. They conclude that a scheme which uses the shortening and extension of output duration and the virtual time contraction and expansion together for voice and the shortening and extension of output duration for video produces the best quality of lip-synchronization. They also confirmed that the skipping and discarding control is not suited to voice.

On the other hand, in [10,11,35,53,61,65], the concept of **event-synchronization** is introduced (in [35], the case of a tele-robotic system is presented, but this kind of synchronization is usually used in networked game applications). It is similar to inter-stream synchronization, but it uses event-based action reference instead of the time. Therefore, event-synchronization control is coarser-grained than inter-stream synchronization [100]. In order to keep a consistent view of the state of the application, some mechanisms to guarantee a global ordering of events are necessary. This can either be done by avoiding disordering (by waiting for all possible events to arrive), or by having mechanisms to detect and correct disordering. To achieve this goal, **conservative and optimistic synchronization algorithms** have been devised. These algorithms usually consist of a collection of logical processes that communicate by exchanging timestamped messages or events. In conservative algorithms [11], receivers are not allowed to advance their virtual clocks until all other receivers have acknowledged that they have completed the computation for the current time period. It is impossible for inconsistencies to occur since no receiver performs calculations until it has the same exact information as everyone else. Unfortunately, with this scheme it is impossible to guarantee any relationship between virtual time (game time) and wall-clock time. Optimistic algorithms [10,35,53,61,65] execute events before they know for sure that no earlier events could arrive, and then repair inconsistencies when they are wrong. Algorithms of this type are far better suited to interactive situations (i.e. networked games). In contrast to conservative approaches that avoid violations of the local causality constraint, optimistic methods allow violations to occur, but are able to detect and recover from potential inconsistency. We can find several different reactive techniques: receivers can *discard late events* [11] or use *rollback techniques*, such as maintaining late events and using them to compensate for inconsistency at the receiving end (in the Timewarp algorithm [53] this can cause an extra overhead in terms of memory space and computation for inconsistency compensation). In [53] *Rollback based techniques* are also exploited to reestablish the consistency of the game state. Copies of the states are maintained after command executions and events received after their playout time are stored locally instead of being dropped and used to compensate for the inconsistency among receivers' views. Then, visual rendering of significant events can be delayed (to avoid inconsistencies if corrections occur). The problem, in this case, is that the use of these realignment techniques may further impact on the responsiveness of the system. Trailing State Synchronization (TSS [10]) is another optimistic synchronization algorithm, which uses *dynamically changing states as the source of rollbacks* as opposed to static snapshots, which is the fundamental difference between it and Timewarp [53]. It maintains several instances of the applications running with different synchronization delays. In [61], a proactive event discarding mechanism relying on the discrimination of obsolete events is used (obsolete events are discarded with a probability depending on the level of interactivity). In TSS inconsistencies are detected by detecting when the leading state and the correct state diverge, and are corrected at that point. In [65], state rollback is executed only when there exists an event whose timestamp is within the rollback time and the event is related with others that should be executed

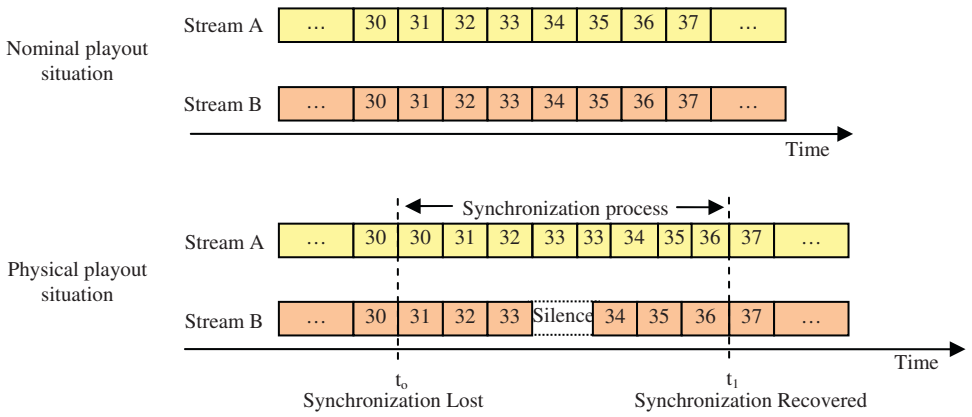


**Fig. 10.** Example of reactive control techniques at the receiver site.

after it but has been previously performed (correlated events).

#### 4.2.4. Common control techniques

We have found several techniques that can be used as a means to prevent (preventive) or correct (reactive) asynchrony, from the source or the receiver side.

(a) Source control

Some authors [8,25,29,76,80] propose that the source *skips or pauses MDUs in the transmission process*, according to feedback information from the receivers (to prevent and/or correct the asynchrony situations). Moreover, the source could send empty MDUs, instead of skipping, when the generation rate is lower than the transmission rate [101].

When we consider stored contents [25], the source can *advance the transmission timing dynamically depending on the network delay estimation* (made by both the source and/or the receivers). For example, the timing can be advanced by skipping MDUs in the transmission.

In [93], the *adjustment of the input rate* is proposed. The source can vary the clock frequency of the input device, according to the obtained synchronization quality. In this work the *interpolation of data* is also proposed to adjust the effective input rate.

Another technique that can be used is the *Media Scaling*. Layered multicast is an example of it and more or less streams can be transmitted depending on the network conditions [74,102]. For example, the temporal or spatial resolution of the video stream can be changed depending on the network load.

(b) Receiver control

One of the techniques included in this Group is the *adjustment of the playout rate* by modifying the playout device's clock frequency, according to the obtained synchronization quality. In [2,62,93] the receivers adjust the playout rate according to the size of the playout buffer.

In [93], another technique is proposed: the *data interpolation* in the receiver side to adjust the effective output rate. Table 1 summarizes all the techniques described above.

## 5. Comparison

We have found numerous approaches to the modeling and execution of multimedia synchronization scenarios. Unfortunately, these approaches are difficult to compare and evaluate due to their varied theoretical bases and modeling techniques [23].

In Table 2 we summarize, chronologically ordered, all the identified synchronization solutions, presenting the above-described techniques, and other factors of interest, such as the following ones:

- *Clocks*: Table indicates if the clock signal used by the algorithm is globally synchronized (global reference) or if it is available only locally (local reference).
- *Network delay limits*: The need for the solution to know in advance these limits or their probability distribution function is indicated.
- *MDU generation periodicity*: The solution can have been developed to work with transmission of streams in which the generation of MDUs is periodical or not.
- *Stored or live contents*: Some solutions have been developed for transmission of stored content, live content or for both content types.
- *Synchronization type included* in the solution (intra-stream, inter-stream and/or group). In this case, as mentioned before, only those solutions which include inter-stream or group Synchronization have been classified. Solutions for group synchronization were not included in the classification presented in [22]. Those which only include intra-stream synchronization have not been considered.
- *Master/slave relationship*: The existence of master/slave relationships (between streams or receiver) is indicated in the table.
- *Group synchronization techniques*: In the solutions which include group synchronization techniques, the technique/s included are also indicated (*master/slave receiver scheme*, *SMS* and/or *distributed control scheme*).

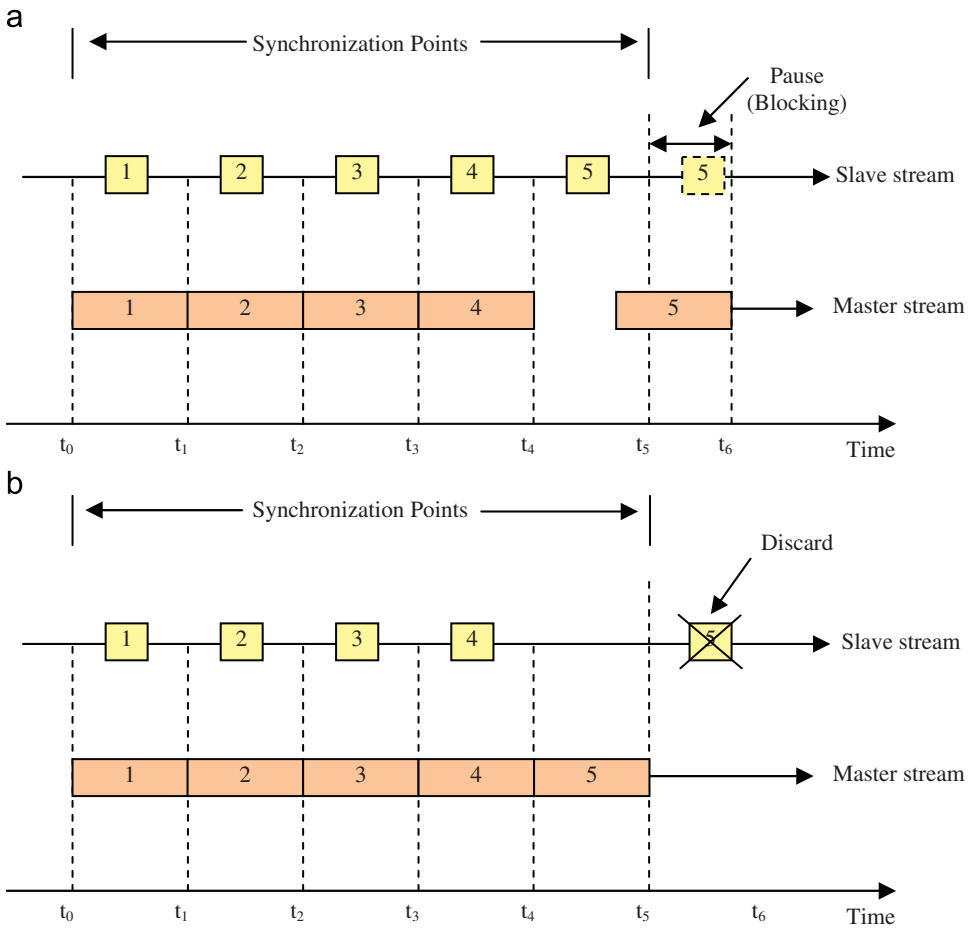


**Fig. 11.** Example of master/slave inter-media synchronization techniques. (a) Pausing MDUs playout or stopping MDUs playout technique; (b) discarding MDUs technique.

- *Feedback utilization*: Some solutions use feedback information included in the messages sent back from the receivers to the sources. The use of feedback techniques for synchronization purposes is indicated.
- *Synchronization information*: The information useful for synchronization included in the transmitted MDUs (if there is any) is indicated.
- *Location of the synchronization techniques*: The location of the synchronization techniques is important. The synchronization control can be done by the source/s or by the receiver/s.
- *RTP use*: New proposals use or allow the use of RTP/RTCP protocol [85]. Older proposals do not use it.
- *Synchronization techniques*: The most representative techniques included in each solution have been indicated in the table.

In the first column (*Name*) we present the name the authors assigned to their proposal and its main references. If there is no name, we only put the references.

The gaps in the table indicate that the factor related to the column is not considered or included by the solution or, possibly, we have not found any mention of the use or the inclusion of that factor in the references in which the solution is described.

We can see that there are a large variety of synchronization solutions, and they differ in terms of goals and application scenarios. They have been defined for many different conditions and environments; therefore, in each one a different combination of several synchronization techniques is used. It has not been an easy job to try to find the relationships between them and to make the qualitative comparison. Perez and Little [23] explain why there are so many synchronization frameworks, how a multimedia scenario can be represented with different temporal specification schemes, and why some specification schemes cannot model all scenarios.

The solution proposed by the authors [30,31], described in Section 3 has been emphasized (in bold type).

## 6. Conclusions

In this paper, a comprehensive qualitative comparison between the 53 most relevant multimedia synchronization

solutions has been described. As far as the authors know this is the largest one that has been published. It has been done taking into account some critical issues in the multimedia synchronization field. Once the three types of multimedia synchronization (intra-stream, inter-stream and group) and the main synchronization techniques have been described, only those solutions that provide inter-stream and group synchronization have been considered in our survey. They have been studied and we have specified which techniques are included in each solution. All this information has been tabulated, with the solutions chronologically ordered. For a better understanding of the paper we have included our solution as an example of multimedia group and inter-stream synchronization proposal, which includes some of the described techniques and uses modified standard protocols, such as RTP/RTCP.

Although some of the references at the end of the paper may appear dated, the authors have chosen to use the references of the papers in which the solutions were described for the first time. Many of the solutions have been used and tested subsequently in new environments by their authors and those papers have also been referenced. As an example, among others, we have found the VTR algorithm [76] being used (slightly enhanced in some cases) recently in media synchronization between voice and movement of avatars in networked virtual environments [70], for group synchronization control for haptic media in networked virtual environments [44,45], in media games [48,69], in collaborative work scenarios [45,69,71], for media synchronization in wireless networks [46,87], and in a remote haptic drawing system [72]. Despite all these papers, we have maintained the VTR's original reference of 1995 [76].

Our main aim has not been to classify the solutions from best to worst because, as discussed before, they all differ in terms of goals and application scenarios (each one has been developed and is suitable for specific scenarios and application conditions). For this reason it is difficult (if not impossible) to compare all the solutions quantitatively. Moreover, in the references there are papers in which we can find the evaluation in the specific scenarios (some solutions have been evaluated and compared with other solutions in the related papers—hyphenated along the text of the paper). It would be very complicated (if not impossible) to implement and evaluate all them in the same scenario in order to compare the results about efficiency or synchronization QoS. In many cases we have only found a quite short paper describing the solution. For this reason we have chosen some objective issues that allow us to compare them, qualitatively at least.

In Table 2, the solutions have been ordered chronologically. A very important issue to emphasize is the fact that most of the modern solutions use feedback and time information (timestamps) included in the RTP/RTCP protocols, as the authors' proposal, described in Section 3, does. Until RTP was chosen as an standard for the time-dependent multimedia streams transmission protocol, most of the solutions did not follow nor use any standard protocol but defined new synchronization protocols with new data packet formats including time information and new control messages for feedback. Using RTP, the solutions take advantage of the use of control RTCP report packets for including feedback information or useful information for multimedia synchronization purposes. Modern solutions, as the ones described in [30,31,49], use RTP/RTCP and, moreover, some old solutions have been implemented subsequently using RTP (for example, VTR in [74,75]).

Our main aim is that this comparison should be very useful to new researchers in multimedia fields to understand quickly the most common synchronization techniques and which ones are used by each solution. Novel researchers have a very valuable starting point to choose and study the solutions they are interested in and to develop new solutions using the described techniques they choose.

## Acknowledgments

Authors want to thank Denis Shasha, the Editor-in-Chief of the journal, and the reviewers for their comments that helped us improve the quality of this paper substantially.

## References

[1] G. Blakowski, R. Steinmetz, A media synchronization survey: reference model, specification and case studies, IEEE J. Sel. Areas Commun. 14 (1) (1996) 5–35.
[2] K. Rothermel, T. Helbig, An adaptive stream synchronization protocol, in: Proceedings of the Fifth International Workshop on Network and Operating System Support for Digital Audio and Video, Durham, New Hampshire, USA, April 1995, pp. 189–202.
[3] M. Chen, Low-latency lip-synchronized videoconferencing system, Proceedings of the Conference on Human Factors in Computing Systems (CHI2003), Fort Lauderdale, FL, USA 5 (1) (2003) 465–471.
[4] I. Kouvelas, V. Hardman, A. Watson, Lip synchronization for use over the internet: analysis and implementation, in: Proceedings of the IEEE Conference on Global Communications, GLOBECOM'96, vol. 2, London, UK, November 1996, pp. 893–898.
[5] L. Qiao, K. Nahrstedt, Lip synchronization within an adaptive VoD, SPIE Multimedia Computing and Networking 1997, San José, CA, USA, February 1997, pp. 170–181.
[6] C. Liu, Y. Xie, M.J. Lee, T.N. Saadawi, Multipoint multimedia teleconference system with adaptative synchronization, IEEE J. Sel. Areas Commun. 14 (7) (1996) 1422–1435.
[7] Y. Xie, C. Liu, M.J. Lee, T.N. Saadawi, Adaptive multimedia synchronization in a teleconference system, Multimedia Syst. 7 (4) (1999) 326–337.
[8] E. Biersack, W. Geyer, Synchronized delivery and playout of distributed stored multimedia streams, Multimedia Syst. 7 (1) (1999) 70–90.
[9] S.S. Manvi, P. Venkataram, An agent based synchronization scheme for multimedia applications, J. Syst. Software (JSS) 79 (5) (2006) 701–713.
[10] E. Cronin, B. Filstrup, S. Jamin, A.R. Kurc, An efficient synchronization mechanism for mirrored game architectures, Multimedia Tools Appl. 23 (l) (2004) 7–30.
[11] C. Diot, L. Gautier, A distributed architecture for multiplayer interactive applications on the Internet, IEEE Network 13 (4) (1999) 6–15.
[12] C.M. Huang, C. Wang, J.M. Hsu, Formal modeling and design of multimedia synchronization for interactive multimedia presentations in distributed environments, in: International Conference on Consumer Electronics 1998, ICCE 1998, Digest of Technical Papers, June 1998, pp. 458–459.
[13] C.M. Huang, C. Wang, Synchronization for interactive multimedia presentations, IEEE Multimedia 5 (4) (1998) 44–62.

[14] C.M. Huang, C. Wang, C.H. Lin, Interactive multimedia synchronization in the distributed environment using the formal approach, IEE Proc. Soft. 147 (4) (2000) 131–146.
[15] Y. Ishibashi, S. Tasaka, H. Miyamoto, Joint synchronization between stored media with interactive control and live media in multicast communications, IEICE Trans. Commun. E85-B (4) (2002) 812–822.
[16] D. Köhler, H. Müller, Multimedia playout synchronization using buffer level control, in: Proceedings of the Second International Workshop on Advanced Teleservices and High-Speed Communication Architectures, LNCS, vol. 868, Heidelberg, Germany, September 1994, pp. 167–180.
[17] N. Laoutaris, I. Stavrakakis, Intrastream synchronization for continuous media streams: a survey of playout schedulers, IEEE Network Mag. 16 (3) (2002) 30–40.
[18] Y.H. Wang, H.Z. Lin, Cooperating intelligent mobile agents mechanism for distributed multimedia synchronization, in: Proceedings of the International Conference on Multimedia and Expo (ICME'00), vol. 2, New York, USA, July/August 2000, pp. 743–746.
[19] A. Furfaro, L. Nigro, F. Pupo, Multimedia synchronization based on aspect oriented programming, Microprocessors Microsystems 28 (2) (2004) 47–56.
[20] F. Boronat, J.C. Guerri, Analysis and comparison of multimedia inter-stream and group synchronization algorithms, IEEE Latin Am. Trans. 3 (5) (2005) 23–32 (in Spanish).
[21] L. Ehley, B. Furht, M. Ilyas, Evaluation of multimedia synchronization techniques, in: Proceedings of the International Conference Multimedia Computing and Systems, ICMCS 94, Boston, MA, USA, May 1994, pp. 514–519.
[22] Y. Ishibashi, S. Tasaka, A comparative survey of synchronization algorithms for continuous media in network environments, in: Proceedings of the 25th IEEE Conference on Local Computer Networks, Tampa, FL, USA, November 2000, pp. 337–348.
[23] M.J. Perez-Luque, T.D.C. Little, A temporal reference framework for multimedia synchronization, IEEE J. Sel. Areas Commun. 14 (1) (1996) 36–51.
[24] H. Abe, H. Shigeno, K. Okada, Media synchronization method for video hypermedia application based on extended event model, in: Proceedings of IEEE International Conference on Multimedia and Expo, Toronto, Canada, July 2006, pp. 1289–1292.
[25] Z. Ali, A. Ghafoor, C.S. George Lee, Media synchronization in multimedia web using a neuro-fuzzy framework, IEEE J. Sel. Areas Commun. 18 (2) (2000) 168–183.
[26] I.F. Akyildiz, W. Yen, Multimedia group synchronization protocols for integrated services networks, IEEE J. Sel. Areas Commun. 14 (1) (1996) 162–173.
[27] S. Baqai, M.F. Khan, M. Woo, S. Shinkai, A.A. Khokar, A. Ghafoor, Quality-based evaluation of multimedia synchronization protocols for distributed multimedia information systems, IEEE J. Sel. Areas Commun. 14 (7) (1996) 1388–1403.
[28] I. Bartoli, G. Iacovoni, F. Ubaldi, A synchronization control scheme for videoconferencing services, J. Multimedia 1 (4) (2007) 1–9.
[29] E. Biersack, W. Geyer, C. Bernhardt, Intra- and inter-stream synchronization for stored multimedia streams, in: Proceedings of the IEEE International Conference on Multimedia Computing and Systems, Hiroshima, Japan, June 1996, pp. 372–381.
[30] F. Boronat, J.C. Guerri, M. Esteve, J.M. Murillo, RTP-based feedback global protocol integration in Mbone tools, EUROMEDIA 2002, Modena, Italy (Best Paper Award) April 2002.
[31] F. Boronat, Specification and evaluation of a Multimedia group synchronization algorithm, Ph.D. Thesis, Polytechnic University of Valencia (UPV), Spain, April 2004, 410pp (in Spanish).
[32] A. Boukerche, S. Hong, T. Jacob, An efficient synchronization scheme of multimedia streams in wireless and mobile systems, IEEE Trans. Parallel Distributed Syst. 13 (9) (2002) 911–923.
[33] A. Boukerche, S. Hong, T. Jacob, Synchronization and handoff management schemes for wireless multimedia systems, Comput. Networks: Int. J. Comput. Telecommun. Networking 41 (3) (2003) 347–362.
[34] A. Boukerche, H. Owens, Media synchronization and QoS packet scheduling algorithms for wireless systems, Mobile Networks Appl. 10 (1–2) (2005) 233–249.
[35] I. Elhajj, N. Xi, B. Song, M.-M. Yu, W.T. Lo, Y.H. Liu, Transparency and synchronization in supermedia enhanced Internet-based teleoperation, in: Proceedings of the 2002 IEEE International Conference on Robotics & Automation, Washington, USA, May 2002, pp. 2713–2718.
[36] J. Escobar, C. Partridge, D. Deutsch, Flow synchronization protocol, IEEE/ACM Trans. Networking 2 (2) (1994) 111–121.
[37] A.J. González, H. Abdel-Wahab, Light-weight stream synchronization framework for multimedia collaborative applications, in: Proceedings of the Fifth IEEE Symposium on Computers and Communications, Antibes-Juan Les Pins, France, July 2000, pp. 398–403.
[38] J.C. Guerri, M. Esteve, C.E. Palau, V. Casares, Feedback flow control with hysteresial techniques for multimedia retrievals, Multimedia Tools Appl. 13 (3) (2001) 307–332.
[39] C.M. Huang, H.Y. Kung, J.L. Yang, PARK: a paused-and-run k-stream multimedia synchronization control scheme, in: Proceedings of the 20th International Conference on Distributed Computing Systems, Taipei, Taiwan, April 2000, pp. 272–279.
[40] C.M. Huang, H.Y. Kung, J.L. Yang, Synchronization and flow adaptation schemes for reliable multiple-stream transmission in multimedia presentations, J. Syst. Software 56 (2) (2001) 133–151.
[41] Y. Ishibashi, S. Tasaka, A group synchronization mechanism for stored media and its measured performance in a heterogeneous network, IEICE Trans. Commun. E82-B (7) (1999) 1009–1018.
[42] Y. Ishibashi, S. Tasaka, A group synchronization mechanism for live media in multicast communications, in: Global Telecommunications Conference, 1997 (IEEE GLOBECOM' 97), November 1997, pp. 746–752.
[43] Y. Ishibashi, S. Tasaka, Y. Tachibana, Adaptive causality and media synchronization control for networked multimedia applications, in: Proceedings of the IEEE International Conference on Communications, vol. 3, Helsinki, Finland, June 2001, pp. 952–958.
[44] Y. Ishibashi, T. Hasegawa, S. Tasaka, Group synchronization control for haptic media in networked virtual environments, in: Proceedings of the 12th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Chicago, USA, March 2004, pp. 106–113.
[45] H. Kaneoka, Y. Ishibashi, Effects of group synchronization control over haptic media in collaborative work, in: Proceedings of the 14th International Conference on Artificial Reality and Telexistence (ICAT'04), Coex, Korea, November/December 2004, pp. 138–145.
[46] S. Tasaka, Y. Ishibashi, M. Hayashi, Inter-destination synchronization quality in an integrated wired and wireless network with handover, in: Global Telecommunications Conference, GLOBECOM'02, vol. 2, Taipei, Taiwan, November 2002, pp. 1560–1565.
[47] Y. Ishibashi, S. Tasaka, A distributed control scheme for group synchronization in multicast communications, in: Proceedings of the International Symposium Communications, Kaohsiung, Taiwan, November 1999, pp. 317–323.
[48] Y. Ishibashi, S. Tasaka, A distributed control scheme for causality and media synchronization in networked multimedia games, in: Proceedings of the 11th International Conference on Computer Communications and Networks, Miami, USA, October 2002, pp. 144–149.
[49] C.C. Kuo, M.S. Chen, J.C. Chen, An Adaptive transmission scheme for audio and video synchronization based on real-time transport protocol, in: Proceedings of the IEEE International Conference on Multimedia and Expo, August 2001.
[50] H. Liu, M. El Zarki, A synchronization control scheme for real-time streaming multimedia applications, in: Proceedings of the 13th Packet Video Workshop, Nantes, France, April 2003.
[51] H. Liu, M. El Zarki, On the adaptive delay and synchronization control of video conferencing over the internet, in: Proceedings of the International Conference on Networking 2004 (ICN'04), February 2004.
[52] H. Liu, M. El Zarki, Adaptive delay and synchronization control for Wi-Fi based AV conferencing, in: Proceedings of the First International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks, Dallas, TX, USA, October 2004, pp. 84–91.
[53] M. Mauve, J. Vogel, V. Hilt, W. Effelsberg, Local-lag and timewarp: providing consistency for replicated continuous applications, IEEE Trans. Multimedia 6 (l) (2004) 47–57.
[54] M. Mielke, A. Zhang, A multi-level buffering and feedback scheme for distributed multimedia presentation systems, in: Proceedings of the Seventh International Conference Computer Comm. and Networks, Lafayette, LA, USA, October 1998, pp. 219–226.
[55] T. Vinod, A. Zhang, Dynamic playout scheduling algorithms for continuous multimedia streams, Multimedia Syst. 7 (4) (1999) 312–325.
[56] A. Zhang, Y. Song, M. Mielke, NetMedia: streaming multimedia presentations in distributed environments, IEEE Multimedia 9 (1) (2002) 56–73.

[57] R. Mussauer, F. Paiva, Gonçalves, P.A. da S. Gonçalves, O.C. Muniz, SAMM: an integrated environment to support multimedia synchronization of pre-orchestrated data, in: IEEE International Conference on Multimedia Computing and Systems 1999, vol. 2, Florence, Italy, 06/07/1999–06/11/1999, pp. 929–933.
[58] N.U. Qazi, M. Woo, A. Ghafoor, A synchronization and communication model for distributed multimedia objects, in: Proceedings of the ACM Multimedia'93, Anaheim, CA, August 1993, pp. 147–56.
[59] M. Woo, N.U. Qazi, A. Ghafoor, A synchronization framework for communication of pre-orchestrated multimedia information, IEEE Network 8 (1) (1994) 52–61.
[60] K. Nahrstedt, L. Qiao, Stability and Adaptation Control for Lip Synchronization Skews, Technical Report, Department of Computer Science, University of Illinois at Urbana-Champaign, May 1998.
[61] C.E. Palazzi, S. Ferretti, S. Cacciaguerra, M. Roccetti, On maintaining interactivity in event delivery synchronization for mirrored game architectures, in: IEEE Global Telecommunications Conference Workshops, Dallas, TX, USA, November/December 2004, pp. 157–165.
[62] K. Rothermel, T. Helbig, An adaptive protocol for synchronizing media streams, ACM/Springer Multimedia Syst. 5 (5) (1997) 324–336.
[63] N. Shivakumar, C. Sreenan, B. Narendran, P. Agrawal, The concord algorithm for synchronization of networked multimedia streams, in: Proceedings of the Second IEEE International Conference on Multimedia Computing and Systems, Washington, DC, USA, May 1995, pp. 31–40.
[64] S.H. Son, N. Agarwal, A model for specification and synchronization of data for distributed multimedia applications, Multimedia Tools Appl. 3 (2) (1992) 79–104.
[65] S. Xiang-bin, L. Fang, D. Ling, Z. Xing-hai, X. Yuan-sheng, An event correlation synchronization algorithm for MMOG, in: Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, vol. 1, Qingdao, China, July/August 2007, pp. 746–751.
[66] C.C. Yang, Design of the application-level protocol for synchronized multimedia sessions, in: Proceedings of the IEEE International Conference on Communications, 2002 (ICC'02), New York, USA, April/May 2002, pp. 2518–2522.
[67] R. Yavatkar, MCP: A protocol for coordination and temporal synchronization in collaborative applications, in: Proceedings of the 12th International Conference Distributed Computing Systems, Yokohama, Japan, June 1992, pp. 606–613.
[68] R. Yavatkar, K. Lakshman, Communication support for distributed collaborative applications, Multimedia Syst. 2 (2) (1994) 74–88.
[69] T. Hashimoto, Y. Ishibashi, Group synchronization control over haptic media in a networked real-time game with collaborative work, in: Proceedings of the Fifth ACM SIGCOMM workshop on Network and System Support for Games, Singapore, October 2006.
[70] Y. Ishibashi, I. Inazumi, S. Tasaka, T. Asano, Media synchronization between voice and movement of avatars in networked virtual environments, in: Proceedings of the 2004 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, Singapore, June 2004, pp. 134–139.
[71] Y. Ishibashi, T. Kanbara, S. Tasaka, Inter-stream synchronization between haptic media and voice in collaborative virtual environments, in: Proceedings of the 12th annual ACM international conference on Multimedia, New York, USA, October 2004, pp. 604–611.
[72] Y. Kurokawa, Y. Ishibashi, T. Asano, Group synchronization control in a remote haptic drawing system, in: Proceedings of the IEEE International Conference on Multimedia and Expo, Beijing, China, July 2007, pp. 572–575. Utiliza la rec ITU-R Rec. BT. 500 para evaluación subjetiva
[73] S. Tasaka, Y. Ishibashi, Media synchronization in heterogeneous networks: stored media case, IEICE Trans. Commun. E81-B (8) (1998) 1624–1636.
[74] S. Tasaka, Y. Ishibashi, A Performance Comparison of Single-Stream and Multi-Stream Approaches to Live Media Synchronization E81-B (11) (1998) 1988–1997.
[75] S. Tasaka, T. Nunome, Y. Ishibashi, Live media synchronization quality of a retransmission-based error recovery scheme, in: Proceedings of the IEEE International Conference on Communications, New Orleans, LA, USA, June 2000, pp. 1535–1541.
[76] Y. Ishibashi, S. Tasaka, A synchronization mechanism for continuous media in multimedia communications, in: Proceedings of the IEEE INFOCOM'95, Boston, MA, USA, April 1995, pp. 1010–1019.
[77] Y. Ishibashi, K. Tomaru, S. Tasaka, K. Inazumi, Group synchronization in networked virtual environments, in: Proceedings of the 38th IEEE International Conference on Communications 2003, vol. 2, Alaska, USA, May 2003, pp. 885–890.
[78] Y. Ishibashi, A. Tsuji, S. Tasaka, A group synchronization mechanism for stored media in multicast communications, in: Proceedings of the Sixth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), vol. 2, Kobe, Japan, April 1997, pp. 692–700.
[79] S. Ramanathan, P.V. Rangan, Continuous media synchronization in distributed multimedia systems, in: Proceedings of the Third International Workshop on Network and Operating System Support for Digital Audio and Video, La Jolla, CA, USA, November 1992, pp. 289–296.
[80] S. Ramanathan, P.V. Rangan, Adaptive feedback techniques for synchronized multimedia retrieval over integrated networks, IEEE/ACM Trans. Networking 1 (2) (1993) 246–259.
[81] S. Ramanathan, P.V. Rangan, Feedback techniques for intra-media continuity and inter-media synchronization in distributed multimedia systems, Comput. J. 36 (1) (1993) 19–31.
[82] P.V. Rangan, S. Ramanathan, Designing an on-demand multimedia service, IEEE Commun. Mag. 30 (7) (1992) 56–64.
[83] P.V. Rangan, S. Ramanathan, T. Kaeppner, Performance of inter-media synchronization in distributed, heterogeneous multimedia systems, Comput. Networks ISDN Syst. 27 (4) (1995) 549–565.
[84] P.V. Rangan, S. Ramanathan, S. Sampathkumar, Feedback techniques for continuity and synchronization in multimedia information retrieval, ACM Trans. Inf. Syst. 13 (2) (1995) 145–176.
[85] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: a transport protocol for real-time applications, RFC-3550, July 2003.
[86] D. Kutscher, J. Ott, The Message bus. a communication-integration infrastructure for component-based systems, White Paper, January 2000. 〈http://www.mbus.org/mbuswp.pdf〉.
[87] T. Nunome, S. Tasaka, Inter-destination synchronization quality in a multicast mobile ad hoc network, in: Proceedings of the IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications, vol. 2, Berlin, Germany, September 2005, pp. 1366–1370.
[88] M.C. Yuang, B.C. Lo, Y.G. Chen, P.L. Tien, A synchronization paradigm with QoS guarantees for multimediacommunications, in: Global Telecommunications Conference, GLOBECOM '99, vol. 1, Rio de Janeiro, Brazil, 12/05/1999–12/09/1999, pp. 214–220.
[89] L. Lamont, L. Li, R. Brimont, D. Georganas, Synchronization of multimedia data for a multimedia news on demand application, IEEE J. Sel. Areas Commun. 14 (1) (1996) 264–278.
[90] Y. Ishibashi, S. Tasaka, Y. Tachibana, Media synchronization and causality control for distributed multimedia applications, IEICE Trans. Commun. E84-B (3) (2001) 667–677.
[91] T.D.C. Little, F. Kao, An intermedia skew control system for multimedia data presentation, in: Proceedings of the Third International Workshop on Network and Operating System Support for Digital Audio and Video, La Jolla, CA, USA, November 1992, pp. 121–132.
[92] T.D.C. Little, A framework for synchronous delivery of time-dependent multimedia data, Multimedia Syst. 1 (2) (1993) 87–94.
[93] D.P. Anderson, G. Homsy, A continuous media I/O server and its synchronization mechanism, IEEE Comput. 24 (10) (1991) 51–57.
[94] M. Correia, P. Pinto, Low-level multimedia synchronization algorithms on broadband networks, in: International Multimedia Conference, Proceedings of the Third ACM international conference on Multimedia, San Francisco, CA, USA, November 1995, pp. 423–434.
[95] K. Ravindran, V. Bansal, Delay compensation protocols for synchronization of multimedia data streams, IEEE Trans. Knowl. Data Eng. 5 (4) (1993) 574–589.
[96] H.Y. Chen, J.L. Wu, MultiSync: a synchronization model for multimedia system, IEEE J. Sel. Areas Commun. 14 (1) (1996) 238–248.
[97] S. Yoo, Realization of the synchronization controller for multimedia applications, in: Proceedings of the IEEE GLOBECOM'98, vol. 2, Sydney, Australia, December 1998, pp. 798–803.

[98] C.M. Huang, R.Y. Lee, Achieving multimedia synchronization between live video and live audio streams using QoS controls, Comput. Commun. 19 (5) (1996) 456–467.
[99] P.V. Rangan, S. Srihari, S. Rajan, Continuity and synchronization in MPEG, IEEE J. Sel. Areas Commun. 14 (1) (1996) 52–60.
[100] C.C. Yang, J.H. Huang, A Multimedia synchronization model and its implementation in transport protocols, IEEE J. Sel. Areas Commun. 14 (1) (1996) 212–225.
[101] P.N. Zarros, M.J. Lee, T.N. Saadawi, Interparticipant synchronization in real-time multimedia conferencing using feedback, IEEE/ACM Trans. Networking 4 (2) (1996) 173–180.
[102] S. McCanne, Van Jacobson, M. Vetterli, Receiver-driven layered multicast, in: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, ACM SIGCOMM, Palo Alto, CA, USA, August 1996, pp. 117–130.