Mario Montagud; Fernando Boronat (2012). Enhanced adaptive RTCP-based inter-destination multimedia synchronization approach for distributed applications. Computer Networks. 56(12):2912-2933. doi:10.1016/j.comnet.2012.05.003.

Enhanced Adaptive RTCP-based Inter-Destination Multimedia Synchronization Approach for Distributed Applications

Mario Montagud, Fernando Boronat
Universitat Politècnica de València (UPV) - IGIC Institute
Grao de Gandia, Valencia (Spain)
mamontor@posgrado.upv.es; fboronat@dcom.upv.es

Abstract — Newer social multimedia applications, such as Social TV or networked multi-player games, enable independent groups (or clusters) of users to interact among themselvesand share services within the context of simultaneous media content consumption. In such scenarios, concurrently synchronized playout points must be ensured so as not to degrade the user experience on such interaction. We refer to this process as Inter-Destination Multimedia Synchronization (IDMS). This paper presents the design, implementation and evaluation of an evolved version of an RTCP-based IDMS approach, including an Adaptive Media Playout (AMP) scheme that aims to dynamically and smoothly adjust the playout timing of each one of the geographically distributed consumers in a specific cluster if an allowable asynchrony threshold between their playout states is exceeded. For that purpose, we previously had also to develop a full implementation of RTP/RTCP protocols for NS-2, in which we included the IDMS approach as an optional functionality. Simulation results prove the feasibility of such IDMS and AMP proposals, by adopting several dynamic master reference selection policies, to maintain an overall synchronization status (within allowable limits) in each cluster of participants, while minimizing the occurrence of long-term playout discontinuities (such as skips/pauses) which are subjectively more annoying and less tolerable to users than small variations in the media playout rate.

***Keywords* — Adaptive Media Playout; Inter-Destination Synchronization; Multimedia; RTP/RTCP; Simulation**

List of Abbreviations:

ACT	(RTCP APP) Action Packet
AMP	Adaptive Media Playout
APP	(RTCP) Application-Defined Packet
CBR	Constant Bit Rate
C-to-C	Cluster-to-Cluster
CMTS	Cable Mode Termination System
DSLAM	Digital Subscriber Line Access Multiplexer
FTP	File Transfer Protocol
IDMS	Inter-Destination Multimedia Synchronization
MOS	Mean Opinion Score
MSE	Mean Square Error
P2P	Peer-to-Peer
QoE	Quality of Experience
QoS	Quality of Service
RR	(RTCP) Receiver Report
RTP	Real Time Protocol
RTCP	Real Time Control Protocol
SR	(RTCP) Sender Report

1. Introduction.

1.1. *Multimedia Synchronization.*

Multimedia systems usually involve the integration of various independent media streams, including both continuous (audio or video) and discrete streams (text, images…), sent (unicast or multicast) by one or more sources to one or several receivers, which can be playing one or several of those streams. Due to the temporal, spatial or semantic relationships between the Media Units (MUs), such as video frames or voice samples, within or among the involved media streams, a precise mechanism of coordination and organization in time is needed in order to ensure a time-ordered presentation of the received MUs, in the same way as they were captured/generated by the media source. Such a process of maintenance and integration, in the presentation instant (or playout point), of the temporal (or spatial) relationships of the different types of media streams is referred to as multimedia synchronization [1].

We can distinguish three kinds of temporal multimedia synchronization techniques, namely: intra-stream, inter-stream and inter-destination (also known as group or multipoint) synchronization. Fig. 1 shows an example of each one of them. In it we can see a group of distributed receivers over an IP network, which are playing video, data (e.g. chat messages) and audio streams. First, intra-stream synchronization deals with the maintenance, during the playout, of the temporal relationships among subsequent MUs within each media stream. In Fig. 1, we can observe a proper and continuous playout process of each media stream in all the receivers, such as the evolution of a video sequence showing a jumping ball, together with audio and data streams. As an example, if the multimedia source captures a video sequence at 25 MUs (video frames) per second, they must be played out (displayed) during 40 ms (each one) at the receiver side. Inter-stream Synchronization refers to the preservation of the temporal dependences between playout processes of different, but correlated, media streams (time dependent or not) involved in the application. In remote user speech, the synchronization between the user's audible words and the associated movement of the lips, referred to as lip synchronization (*lip-sync*, [2] and [3]), is an example of this type of synchronization.

The above kinds of synchronization techniques are usually considered and implemented in typical multimedia applications. Nevertheless, a new type of synchronization is essential in a variety of emerging distributed multimedia applications, including both live and stored content streams, such as interactive Social TV, real-time distance learning and networked multi-player games. It is called Inter-Destination Multimedia Synchronization (IDMS), and involves the simultaneous synchronization of one or more playout processes of one or several media streams at geographically distributed receivers. As can be noticed in Fig. 1, at any moment during the multimedia session, all the receivers are playing the same MU of each media stream (IDMS). In this paper, we mainly focus on this kind of multimedia synchronization.

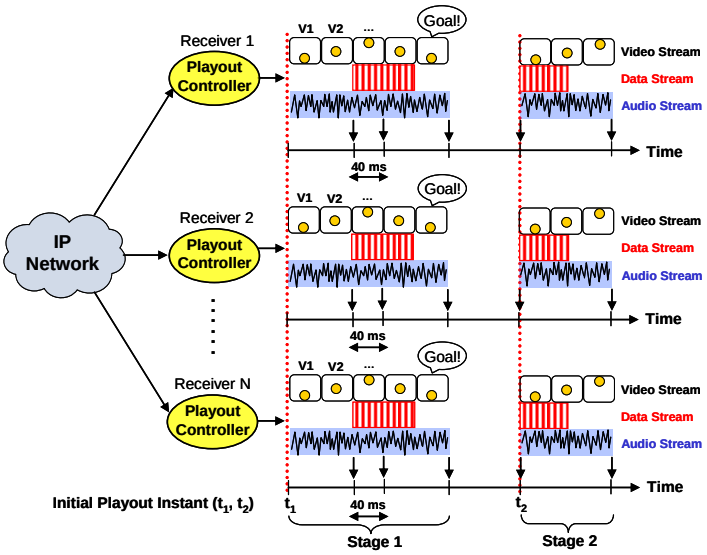


Fig.1. Multimedia Synchronization.

1.2. *Examples of Applications in Which IDMS is Needed.*

IDMS can be applied to any type and/or combination of streaming media, such as audio, video and scene information (e.g. chat, subtitles, etc.). Nowadays, we can find many distributed social multimedia applications in which the lack of IDMS may affect the user experience in many different ways [4]. Here we present some use cases:

1. *Synchronous e-learning*, in which an instructor can distribute a multimedia lesson to a group of students (that could be attending it from different locations), and he or she can occasionally make some comments or ask some questions about its content. Hence, it is crucial that each one of the students receives the multimedia question (possibly transmitted in several streams) at the same time, and, as a result, has a fair chance of answering.

2. *Networked real-time multiplayer games*. In such scenarios, multiple players often collaborate (as a team) with each other and fight against other multiple players (belonging to other teams). When each player presents output timing different from the other players, the fairness among them, or the efficiency of the collaborative work, can be damaged.

3. *Multimedia Cluster-to-Cluster (C-to-C) Applications* [5], including independent but semantically related data streams sent from end-systems located in one or more clusters[1] (sender clusters) to end-systems located in other distributed clusters (receiver clusters). For example, the sender cluster may consist of a collection of capture devices/sources (e.g.,

[1] We define a cluster as a collection of computing and communication end-systems sharing the same local environment

video cameras, microphones, etc.), each one producing an independent stream of data (video, audio, graphics or text media), and the receiver clusters might be a collection of display devices (e.g., digital light projectors, speakers, head-mounted displays, etc.) and computers that archive and reproduce the received data streams. Therefore, synchronization mechanisms must be provided in order to combine, process and present the incoming media streams in an integrated manner, so as to guarantee a high quality multimedia system, regardless of the number of receivers and streams played on the receiver clusters.

4. One of the most prominent use cases in which IDMS becomes indispensable is Social TV, which enables different groups of viewers, independently of their location and the network (and device) they are using, to interact among themselves and share services within the context of simultaneous media content consumption, by using immediate chat messaging, audio/video conferencing services, or for that matter any other sort of shared experience that is yet to appear [6]. In [6], a set of streaming media (IPTV or WebTV) applications providing synchronous shared experiences is presented. As an example, *Watchitoo*[2] is an emerging web-based application that enables not only chatting, but also audio and video conferencing while watching the same video content. A motivating scenario is when various friends are watching a live on-line football match at separate locations (*watching apart together*). In such a case, inter-stream synchronization must be performed between the involved time-dependent media streams, such as between the multimedia content the users are watching together (e.g. the video stream corresponding to a football match) and the associated streams corresponding to the chat messaging or voice/audio conferencing services. Moreover, a significant event (e.g. a goal) should be experienced by all the users almost simultaneously, even in all the associated media streams, so as not to degrade the user experience on such interaction (IDMS).

5. Other typical use cases are: networked quiz shows, distributed tele-orchestra, multi-conferencing with consumer-originated content sharing, video-Centered Communications, video wall, on-line election events, synchronization of multiple television outputs in a single physical location, synchronization of different networked speakers throughout an airport or museum, signaling applications, emergency applications, content distribution (replication) applications, etc.

[2] www.watchittoo.com

1.3. *Challenges and Motivation.*

The changing paradigm from single end-user media consumption to a group shared experience, in which social communication opportunities may be exploited (e.g. conferencing while watching television), faces lots of challenges, such as shared experience modelling, universal session handling, synchronization, and Quality of Service (QoS) [6]. Other essential technical challenges include scalability, noise reduction, presence awareness, design guidelines, privacy concerns, and social networking integration.

As mentioned above, this paper mainly focuses on the synchronization of the playout of media streams across multiple locations (IDMS). We present the implementation, optimization and evaluation, in a simulation framework, of an adaptive RTP/RTCP-based approach to acquire IDMS in distributed multimedia applications.

An important research question is to determine which ranges of asynchrony levels could be tolerated by users (i.e. the asynchrony limits that, if exceeded, are already noticeable and annoying to users) in specific use cases in which IDMS is needed. They should be obtained through very rigorous objective and subjective assessments in the near future. Here we present some preliminary conclusions extracted from previous works. Traditionally, 150 ms has been used as a rule of thumb, a value drawn from telecommunications research. This rule states that the maximum end-to-end one-way delay when talking remotely should not exceed 150 ms. Below this value a user cannot perceive the delay in communication, and therefore cannot detect differences on synchronization of shared video content [6]. The study in [7] provides a set of allowable asynchrony values between different types of media streams that may be tolerable to human perception, but only referred to inter-stream synchronization. Additionally, we have found some Social TV related studies in [4] and [8]. In [4] it is concluded that the requirements on inter-destination content synchronicity in interactive services may vary between 15 and 500 ms, depending on the type of service. In some cases, differences around 100 ms may already have an annoying effect on such interaction. More recently, the study in [8] aims to determine acceptable synchronization levels for Social TV scenarios. It was concluded that playout asynchronies up to 1 second might not be perceptible by users while communicating using either chat messaging or voice conferencing services. However, these results are largely dependent on several factors, such as the genre of the video content, the number of users, their activity and profiles, the communication channel, etc. Consequently, no statistically absolute user tolerance limits may be derived from these preliminary experiments, and more accurate asynchrony levels for IDMS should be achieved to avoid the user's frustration, and thus guarantee a satisfying shared experience in such synchronous media applications.

Nevertheless, these differences can be much larger (up to several seconds) in practical content distribution networks and newer delivery paradigms (e.g. IMS-based television broadcast channels) [4], mainly due to the existence of several uncontrollable factors, some of which can be either

related to the distribution network or to the end-system features, such as variable capturing, coding, encryption, packetization, network (traffic load, trans-coding or format conversion, fragmentation and re-assembly of packets, dynamic routing strategies, improper queuing policies, etc.), processing, depacketization, decoding, decryption, buffering, rendering and presentation delays, or packet losses, which can seriously disturb the original media timing at the receiver side, and result in different (and time-variant) end-to-end delays when multicasting one (or several) flows of information from one (or more) media sources to one (or multiple) destinations (that can be using different kind of terminals), possibly over different delivery chains (network architectures, cross-domain scenarios, etc.), as shown in Fig.2.

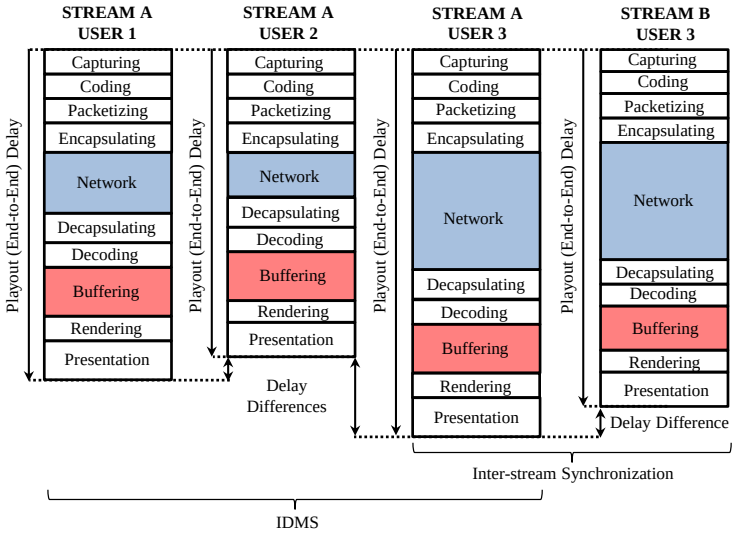


Fig.2. Inter-Stream and Inter-Destination Multimedia Synchronization (IDMS).

Delay and its variability become a serious problem when an interaction between the user and the media content, or interaction between different users in the context of specific content consumption is needed, because it could be detrimental to the Quality of Experience (QoE) in those synchronous social media applications and may prevent the inclusion of such advanced forms of interactivity. Thus, additional adaptive techniques must be provided to meet the above media sync requirements (especially IDMS) in practical content delivery networks.

1.4. *Goals and Contributions.*

The **main goal** of this research work is to **enhance a preliminary IDMS approach**, presented in [10], which is based on simple extensions to RTP/RTCP standard protocols [9], **by implementing and evaluating it in a simulation framework**. That IDMS solution was already implemented and tested satisfactorily, both objectively and subjectively, in a real WAN point-to-multipoint scenario between our University Campuses, but under controlled and limited network conditions [10]. Afterwards, in [5], the IDMS approach was adapted to be applied in one-way C-to-C applications.

However, we decided to extend and optimize that solution and to include it in a simulation framework in order test its performance in various network environments, under different network conditions, and to facilitate a **future** comparison with other existing synchronization approaches, as the ones detailed in [1].

Three main enhancements have been included in the evolved version of the IDMS approach presented in this paper**.** First, several **new dynamic policies for selecting the synchronization master reference** have been added**,** and their suitability and effectiveness have been examined, for specific network conditions and application requirements, according to the impact on the overall quality of the playout adjustments and the buffer fullness variations as the multimedia session goes on. Second, the evolved version of our IDMS approach is able to separately manage the playout processes of **different logical groups of receivers (clusters), independently of their location**. In addition, the feasible playout adjustments performed by each receiver to synchronize with respect to a reference playout point could originate a noticeable degradation of the user perception on the quality of the received media stream/s. Accordingly, another improvement is the **design and implementation of a novel Adaptive Media Playout (AMP) scheme** that aims to minimize the occurrence of long-term playout discontinuities by means of varying the media playout rate, within perceptually tolerable ranges, in order to smoothly acquire an overall synchronization status in a specific cluster every time an asynchrony threshold between the playout states of the receivers belonging to it is exceeded.

We chose Network Simulator 2 (NS-2[3]) for developing our research work. For this purpose, as simulations rely on the accuracy of the proposed models, and the NS-2 native implementation of **RTP/RTCP** neither defines nor strictly follows many of the attributes specified in [9], we firstly had to develop a **new NS-2 module with a more complete and accurate implementation for such protocols** ([11] and [12]). Secondly, this module was extended to include an **optional functionality with the implementation of our complete RTP/RTCP based IDMS approach**, by means of: i) extending and defining new RTCP packets; ii) implementing new control timers; iii) designing a proper **intra-stream synchronization solution** (playout buffering policy); iv) implementing reactive (playout adjustments) techniques to synchronize; etc.

The **simulation results,** presented in Section 7, show the ability and effectiveness of the proposed coordination of both IDMS and AMP techniques, for each one of the proposed master selection policies, to maintain the playout asynchrony between the distributed receivers within each cluster below allowable thresholds for typical multimedia applications, adding only a minimal network and computational load, while minimizing long-term playout discontinuities, such as skips and/or pauses, which are subjectively more annoying and less tolerable to users than short-term playout discontinuities or smooth variations in the media playout rate ([13]-[15]).

[3] Network Simulator 2 (NS-2) Home Page, http://www.isi.edu/nsnam/ns

1.5. *Structure of the Paper.*

This paper has been structured as follows. In the next section some related works are discussed. Section 3 presents the enhancements that have been included in our newly designed RTP/RTCP module for NS-2. In Section 4, the adopted playout buffering policy in order to guarantee intra-stream synchronization is described. Next, Section 5 introduces the effect of the receivers' playout rate imperfections over the local and global media synchronization. Section 6 describes the operation of our IDMS approach and the novel aspects that have been added to it. The evaluation of the IDMS approach is presented in Section 7. Finally, Section 8 is a conclusion and a discussion of future work.

2. Related Works

Over the last few years, many solutions for both intra-stream and inter-stream synchronization have been designed, but not so many for IDMS, despite the increasing relevance that this kind of synchronization is acquiring in a variety of emerging multimedia applications. On the one hand, [16] provides a comparative survey between many intra-stream synchronization techniques. On the other hand, in [1], authors presented an exhaustive qualitative comparison between the most recent inter-stream and IDMS proposals. Generally, three common techniques were employed to acquire the IDMS goal: i) the Master/Slave (M/S) Receiver Scheme; ii) the Distributed Control Scheme (DCS); and iii) the Synchronization Maestro Scheme (SMS).

In M/S Scheme [17], receivers are differentiated into master (one) and slave receivers (the rest). Only the master receiver sends feedback information about playout timing and the slave receivers adjust their playout processes accordingly.

In DCS ([18]-[21]), all the receivers can multicast feedback information about playout timing and each one of them decides the synchronization reference among its own playout timing or those of the other receivers. In [19], the use of '*local lag'* and '*time warp'* algorithms was proposed to avoid inconsistencies between users in replicated continuous applications, such as networked games. This solution has been recently enhanced and used in the iNEM4U platform[4] [20], which provides open, intelligent, and interoperable support services for social applications. In [21], these algorithms have been adapted to achieve coherent execution of a specific user's actions for all the clients, so that a consistent version of a shared video watching experience is perceived by all of them.

SMS ([22]) is based on the existence of a synchronization maestro (that can be the source or one receiver) which gathers the information about the playout processes of all the active receivers and corrects their playout timing by distributing control messages. In order to do this, each receiver sends (unicast or multicast) the information to the maestro, and then the maestro, after processing

[4] www.iNem4u.eu, Interactive Networked Experiences in Multimedia for You, FP7-ICT-2007-1-216647

such information, multicasts a control packet including a reference playout point to which the receivers should be synchronized. Moreover, in SMS, the receivers can be also classified into an M/S scheme, in which the playout timing of the master receiver is taken as the synchronization reference for adjusting the playout timings of the other (slave) receivers. The master receiver role could also be exchanged between receivers according to the network conditions, thus allowing M/S switching technique [1].

In all the above control schemes, an end-user device receiving a media stream reports on arrival time or presentation time of media packets of that stream, and (one or several) synchronization managers are used to collect those control reports and to compute temporal discrepancies among the clients. As a result, end clients must perform playout adjustments to acquire IDMS.

Unlike the above solutions, which are end-point or terminal based, hybrid network-based approaches can also be employed, as the one proposed in [23], in which the synchronization functionality is implemented in network edge nodes (e.g. a DSLAM - Digital Subscriber Line Access Multiplexer- or CMTS - Cable Modem Termination System-, or even higher up in the network hierarchy), each managing the output timing of the equipment of its domain users. The synchronization point (that consists of a synchronization buffer and control functionality) in the network is selected so that further downstream delays are considered acceptable for the combinational service. Further, at a higher level, a synchronization manager is used to control the output timing of the edge nodes. This network-based approach is suitable if a very large number of nodes belong to the same session, as in massive multi-player on-line games or broadcast IPTV channels.

The above end-based architectural solutions can be classified into centralized and distributed IDMS control schemes. Centralized schemes have advantages and disadvantages. On the one hand, they can preserve the causality more easily than distributed control schemes. Also, inconsistency between the states of the session members is less likely. However, centralized schemes have larger network delays, lower reliability, and poorer scalability. Distributed control schemes outperform the former in terms of fairness and interactivity. In [18], the advantages and disadvantages of SMS and DCS were discussed (in terms of reliability, control speed, control overhead…). In [24], both schemes for IDMS were 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 in the virtual environment) was introduced, in addition to the local importance (importance which is judged from the viewpoint of each user). In [25] and [26], an SMS-based approach for IDMS was enhanced to be used efficiently in a Peer-to-Peer (P2P) based system and in a networked real-time game with collaborative work, respectively. Likewise, in [27], an IDMS solution, using both SMS and DCS, was also enhanced by taking into account the importance of the media objects. In [28], the three above end-based IDMS schemes were compared and evaluated in a Multicast Mobile Ad-Hoc Network.

In [10], a preliminary version of an IDMS solution employing an SMS to carry out the synchronization control was presented. In this SMS-based approach for IDMS, the source acted as the synchronization maestro and it considered a fixed receiver as the master synchronization reference during the multimedia session lifetime. An extended SMS was presented in [3], by combining the selection of two reference playout points (the most advanced/lagged) over haptic media in networked real-time games. This study examined the influence of the determination methods of the reference output timings on the fairness among the players and the efficiency of the work. On the one hand, it was reported that the IDMS control improves the efficiency of the work in collaborative games if the output timing of the haptic media among multiple destinations is adjusted to the earliest output timing among the timings under control. This is because a player with earlier output timing can help other player with later output timings. On the other hand, it was concluded that the effectiveness of the IDMS control in competitive games, in terms of the fairness between the players, may be improved by adjusting the overall output timing to the latest one. However, in that work, the assessment of other determination methods for selecting the reference output timing (such as the average of the overall output timings) was left for further study. To the best of authors' knowledge, there is still no published study which addresses this issue. As a consequence, one contribution of this paper is to enhance the previously proposed SMS in [10], so that the synchronization maestro can handle several dynamic policies for selecting the master reference playout point to synchronize. This selection may have a quantitative impact on the synchronization effectiveness. However, there is not an optimal solution for all the possible scenarios because it depends on the environment conditions (network load, receiver's features, etc.) and application requirements. Those master reference selection policies are described and analyzed in Section 6 and 7, respectively.

In [25], the quality of *lip-synch* (inter-stream) was assessed, by combining different reactive control schemes, which were based on skipping, discarding, shortening and extension of output duration. Moreover, the relation between the results of the objective assessment (coefficient of variation of output interval, average playout rate and Mean Square Error -MSE- of inter-stream synchronization were evaluated) and those of the subjective assessment (Mean Opinion Score -MOS-) was investigated by multiple regression analysis. As a result, it was concluded that: i) the reactive control scheme based on shortening and extension of the output duration, for both voice and audio streams, produced the best quality of *lip-synch*; ii) the reactive control scheme based on skipping and discarding MUs was quite annoying for both media streams (especially in voice communications); and iii) average playout rates of voice and video and MSE of inter-stream synchronization were closely related to the MOS value, both for live and stored media streaming.

We do not carry out inter-stream synchronization control in this paper since we only consider the multimedia source transmitting a single media stream to the distributed clients. We already tested a solution for that purpose, both objectively and subjectively, in previous works with

satisfactory results ([10] and [5]). However, the results and conclusions of the work in [25], which were applied for inter-stream synchronization quality, may be extrapolated to our IDMS solution, because as in inter-stream synchronization control, receivers would be forced to adjust their playout timing to keep synchronization, causing probable playout discontinuities that could be annoying to users. As a consequence, we decided to design a novel AMP approach to alleviate those undesirable effects. The premise of AMP is that small variations in the media playout rate (short-term discontinuities) are subjectively more tolerable and less annoying to human perception than playout interruptions (long-term discontinuities). Previous works on AMP solutions have been mostly focused to mitigate the effect of network dynamics in order to improve the intra-stream synchronization quality of audio and video streaming applications (e.g. [15]-[17]) and, occasionally, for inter-stream synchronization purposes [25]. In intra-stream synchronization solutions, the triggering of the playout adjustments can be based either on buffer occupancy level or on buffer fullness variation [15]. Anyway, the goal is to keep the media stream playout as smooth as possible while avoiding buffer outage and/or playout interruptions. By manipulating the playout speed, AMP can reduce initial buffering delays in the case of pre-stored streams, decrease user-perceived latencies of live streams, and compensate packet delay variations in delay-sensitive interactive services, all without sacrificing playout reliability and getting the sender involved. However, in this paper we propose to extend the use of AMP for IDMS purposes, since this technique can enable the geographically distributed receivers belonging to each cluster, to smoothly adjust their playout timing (playing the media faster/slower than normal), according to instructions sent by the source, when the playout time discrepancy between them exceeds an allowable pre-specified threshold.

3. Enhanced RTP/RTCP Module for NS-2

Nowadays, NS-2 has become a widely adopted simulation tool for networking community as a way of designing and evaluating existing and new proposed protocols, algorithms or designs. Since its inception, NS-2 has been under constant improvement and, at present, it supports heterogeneous network architectures characterization, such as Mobile IP networks, WLAN, ad-hoc networks, sensor networks, grid architectures, satellite networks, and many others. Additionally, it contains modules for numerous network components such as MAC layer protocols, unicast and multicast routing algorithms, transport layer protocols, traffic source behaviour, queue management mechanisms, modules for statistics measurement, etc. Despite this variety, sometimes networking researchers need to adapt the existing NS-2 modules to their requirements or incorporate new simulation modules which are beyond the scope of the built-in NS-2 code. The simulator is open source; hence, it allows everyone to make changes to the existing code besides adding new protocols and functionalities to it. Therefore, the above reasons made us choose NS-2 as the simulation tool for developing our research goals.

Our research group is interested in the use of RTP (Real Time Protocol) and RTCP (Real Time Control Protocol) standard protocols, specified in [9], for multimedia streaming services (video, audio or data), their extension with synchronization purposes [10], and their evaluation through simulation techniques. These protocols have become a suitable framework to cope with the temporal and QoS requirements of real-time multimedia applications. While RTP takes care of data delivery, RTCP deals with the transport and management of feedback control reports from/to all the participants of an RTP session.

As simulations rely on the accuracy of the proposed models, and the NS-2 native implementation for RTP/RTCP is incomplete and imprecise, we needed to develop a new NS-2 module with a more precise and complete implementation for such protocols**.** As discussed, the legacy code neither implements nor strictly follows many of the attributes specified in [9]: i) it does not define all the RTCP packets, only RTCP SR packets are included, but its format is not complete (does not include payload type field, number of packets/octets sent fields, etc.); ii) since RR messages are not defined, neither QoS metrics (jitter, network delay, Round Trip Time -RTT-, and loss rate) monitoring nor reporting are provided; iii) the same packet header data structure is used for both RTP and RTCP packets construction; iv) these packet header fields are specified using incorrect variables' types and sizes; v) there is a bug for multicast transmissions configuration; vi) the code does not support multiple multicast streams on the same node; vii) the RTP Agent is only capable of generating Constant Bit Rate (CBR) traffic, etc.

As a consequence, our new RTP/RTCP module[5] includes the following enhancements respect to the legacy code: i) definition of all the types of RTCP packets with their exact format (Sender Report or SR, Receiver Reports or RR, Source Description or SDES, Application-defined or APP and BYE packets); ii) network-level metrics (such as one-way delay, jitter, RTT, throughput and packet loss) monitoring, processing and registering in simulation time; iii) capability of processing any kind of application traffic pattern supported by NS-2; iv) support for multiple multicast streams on the same node; and v) compatibility with the legacy code. More details about the implementation of the module and its use for the assessment of QoS metrics in a simple video streaming application can be found in [12].

[5] The RTP/RTCP module for NS-2 is publicly available at our wepbage: http://personales.gan.upv.es/~fboronat/

4. Intra-Stream Synchronization Approach

In current packet-switched networks it is not an easy task to handle real-time multimedia traffic due to some time-variant factors (such as the statistical nature of the data, dynamic routing strategies, network load, improper queuing mechanisms, probable packet losses, configuration errors, etc.) that can result in a data flow reception with irregular inter-packet delays or *jitter*, which can considerably degrade the quality of the real-time service and cause probably media playout interruptions. This phenomenon is sketched in the middle row of Fig. 3.

Technically speaking, jitter is the measure of the variability over time of the latency across a network. A widely used solution to alleviate/address the effects of jitter (but at the expense of additional delay) is to buffer the received data packets before playing them out in the correct temporal order they were generated. The goal is to obtain a sequence of playout instants, so the temporal distance between each couple of consecutive packets should be the same as at packet generation (uniformly spaced, if we assume CBR traffic), as illustrated in the bottom row of Fig. 3. This is the intra-stream synchronization control, which is necessary for the preservation of the temporal relationships between MUs (e.g. video frames) within a single media stream.

Let us consider the example illustrated in Fig. 3, which shows the transmission, buffering and playing phases of a non-continuous stream of MUs (the example is also valid for a continuous stream). Let t_n, $r_{n,i}$ and $p_{n,i}$ be the time when the *n-th* MU (of a specific burst) is sent, received by the playout buffer of the *i-th* receiver, and played out by that receiver, respectively. The *network dela*y or latency of the *n-th* MU for the *i-th* receiver is given by the difference between the reception instant to the playout buffer and the transmission or departure time, $l_{n,i}=r_{n,i}-t_n$. The *playout delay* for that *n-th* MU, $d_{n,i}$, is given by the time interval between its playout instant at the receiver side and its generation instant at the sender side, $d_{n,i}=p_{n,i}-t_n$. It should be maintained uniformly for each couple of *n-th* and *k-th* MUs in a smooth playout process, i.e. $(p_{n,i}-p_{k,i})\approx(t_n-t_k)$, at least for a group of packets corresponding to an entire burst. The playout delay is given by the sum of network and buffering delays (if coding, packetization, depacketization and decoding delays are considered as negligible), which instead are both variable. The playout delay for the first MU, d_{ini}, must be adequately set, and it is performed in the initial phase of our IDMS approach (discussed in Section VI). Thus, we call *Initial Playout Instant* ($p_{ini,i}$) to the playout time of the first MU (MU_{ini}) for each *i-th* receiver. Next, the playout controller of each *i-th* receiver will schedule the playout time of the successive MUs at $p_{n+1,i}=p_{n,i}+s_{n,i}$, where $s_{n,i}$ refers to the service time for the *n-th* MU, which is given by the difference between timestamps of the *n-th* and *(n+1)-st* MUs (if we assume CBR traffic, and the source transmits θ MU/s, $s_{n,i}$ should be equal to $1/\theta$ seconds).

Without loss of generality, we assume that $r_{n,i} = \infty$ for each lost packet due to the network conditions (packets *2* and *n* in Fig. 3). Furthermore, the playout buffering policy usually discards all packets that arrive later than their scheduled playout instants, i.e., packets with $l_{n,i}>d_{n,i}$, which

implies that all the packets with $r_{n,i} = l_{n,i} = \infty$ are discarded, regardless of the value of $p_{n,i}$ (packet *m* in Fig. 3). Some strategies have been designed in order to try to handle/conceal these losses [16].

In our simulation environment, we have implemented receiver buffers with a configurable capacity. When a packet is received, the playout buffer checks if the (portion of) MU included in it can be accommodated or if it should be dropped. An incoming MU is not added to the buffer if its playout time has already elapsed. The buffer could also implement the following dropping policies [16]: i) when the buffer is full, the newly received MU will be discarded; ii) when the buffer is full, the first MU to be dropped will be the next to be played out (i.e., the buffer would discard the oldest MUs, keeping the more recent ones). Finally, the buffering policy possibly reorders MUs so that they are played out in the order in which they were generated (using the sequence numbers contained in each RTP packet), as in packets *k-th* and *(k+1)-th* in Fig. 3.

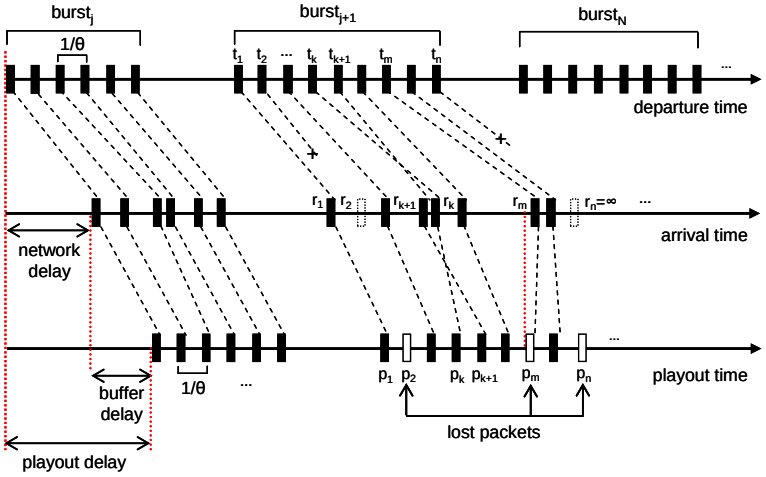


Fig.3. Intra-Stream Synchronization Approach (Playout Buffering Policy).

5. Playout Rate Imperfections

The maintenance of the temporal and causal (i.e. order relation) relationships within or among multiple media streams may be disturbed by several parameters, which can be tackled either individually or an integrated manner [1], such as: *delay, network and end-system jitter, clock and presentation skews/drifts, etc*.

Hence, the availability of a precise, high resolution and reliable local timing mechanism is a desirable feature in communication systems and protocols. Let us consider the effect of accuracy in the receivers' playout timing. A perfect receiver playout rate would mean the consumption of the incoming MUs with the same nominal rate (θ MU/s) as they were generated by the source (μ_1 in Fig. 4a). Nevertheless, real local playout timing mechanisms may not be completely precise. They could present a deviation trend or constant skew limited by $\pm\gamma$ (μ_2 –slow rate– and μ_3 –fast rate– in Fig. 4a), that is often expressed as a ratio in parts per million (*ppm*). Moreover, playout rates could

also present a nonlinear time-variant drift, given by $\omega(t)$, which can randomly oscillate over time between values bounded by a maximum factor of $\pm\ \varepsilon$ *ppm* (μ_4 in Fig. 4a). This fluctuation is very close related to the clock resolution, aging, oscillator stability, voltage changes, surrounding temperature and other environmental variables (e.g., noise), [29]. As a result, the instantaneous playout rate (in MU/s) of the *i-th* receiver can be formulated as:

$$\mu_i(t) = \theta \cdot (1 + \gamma_i + \omega_i(t)) \tag{1}$$

These undesirable factors, obviously, could affect to the receivers' playout asynchrony since their local timing are not generally in perfect agreement. If the receiver playout rate is faster than the sender generation rate, the receiver may suffer *underflow*. Conversely, the receiver can become *flooded* with MUs if its playout rate is slower than the generation rate of the sender.

Let us assume the source begins the transmission rate at t_{ini} instant, and the receivers start the playout of the first incoming MUs when they have buffered them a fixed global amount of time (b_{ini}). In that case we could find an initial playout time discrepancy (asynchrony) between all of them due to the different network delays from the source. As an example, specific MUs could experience a minimum/maximum network delay of l_{min}/l_{max} seconds to reach the nearest/furthest receiver. The worst case occurs when the nearest receiver (which will begin the playout process at $t_{min}=t_{ini}+l_{min}+b_{ini}$) plays out the stream at a maximum rate of $\theta\cdot(1+\gamma)$, whereas the furthest one (which begins the playout process at $t_{max}=t_{ini}+l_{max}+b_{ini}$) plays out the stream at a minimum rate of $\theta\cdot(1-\gamma)$ because, once the furthest receiver begins its playout process, the nearest one has already played out a certain number of MUs given by $[(t_{max}-t_{min})\cdot\theta\cdot(1+\gamma)]$ (first term of Eq. 2).

Moreover, the difference between their playout rates ($[\theta\cdot(1+\gamma)-\theta\cdot(1-\gamma)]\cdot t$) will cause an increasing asynchrony, *A*, in MUs, between them as the multimedia session advances in time (Fig. 4b). This asynchrony, $A(t,\gamma)$, is given by Eq. 2 and 3 (compacted version)[6], where the first term is the contribution of the delay variability, which can be appreciated in the elevation of the left graph in Fig.4, and the second term represents the effect of the undesired playout rate deviations, γ, and the temporal evolution of the multimedia session, *t*:

$$A(t,\gamma) = \left[\left((t_{\max}) - (t_{\min})\right)\left(\theta\cdot(1+\gamma)\right)\right] + \left[\left(\theta\cdot(1+\gamma)\right)t - \left(\theta\cdot(1-\gamma)\right)t\right] \tag{2}$$

$$A(t,\gamma) = \left[(l_{\max} - l_{\min})\cdot\theta\cdot(1+\gamma)\right] + \left[2\cdot\theta\cdot\gamma\cdot t\right] \tag{3}$$

As can be appreciated in (the elevation of) the graph in Fig.4b, if there is a significant delay variability between the furthest and the nearest receiver (l_{min}-l_{max}=200 ms) and fixed buffering delays are set to them, there will be an initial playout asynchrony (200 ms = 5·40 ms →5·1/θ = 5 MUs), which would be already noticeable and annoying in some IDMS use cases, at the starting of the playout of the media stream ($t=0$), even without considering playout rate deviations ($\gamma=0$):

[6] $(t_{max}-t_{min}) = (t_{ini}+l_{max}+b_{ini})-(t_{ini}+l_{min}+b_{ini}) = (l_{max}-l_{min})$;
$([\theta\cdot(1+\gamma)-\theta\cdot(1-\gamma)]\cdot t) = \theta t + \theta\cdot\gamma\cdot t - \theta t - (-\theta\cdot\gamma\cdot t) = 2\cdot\theta\cdot\gamma\cdot t$.

$$A(t=0,\gamma=0)=\left[(250-50)\cdot 10^{-3}\cdot 25\cdot(1+0)\right]+\left[2\cdot 25\cdot 0\cdot 0\right]=5\ MUs$$

This behaviour is unacceptable in practical group shared media experiences and some techniques are needed to cope with all these impairments in order to control/correct the asynchrony between distributed receivers (IDMS). Furthermore, these solutions cannot only be based on buffering techniques but they need a control message interchange process to be adaptive regarding changes of the network conditions, receivers' imperfections, and source drop-outs, which are realistic assumptions when using non real-time operating systems.

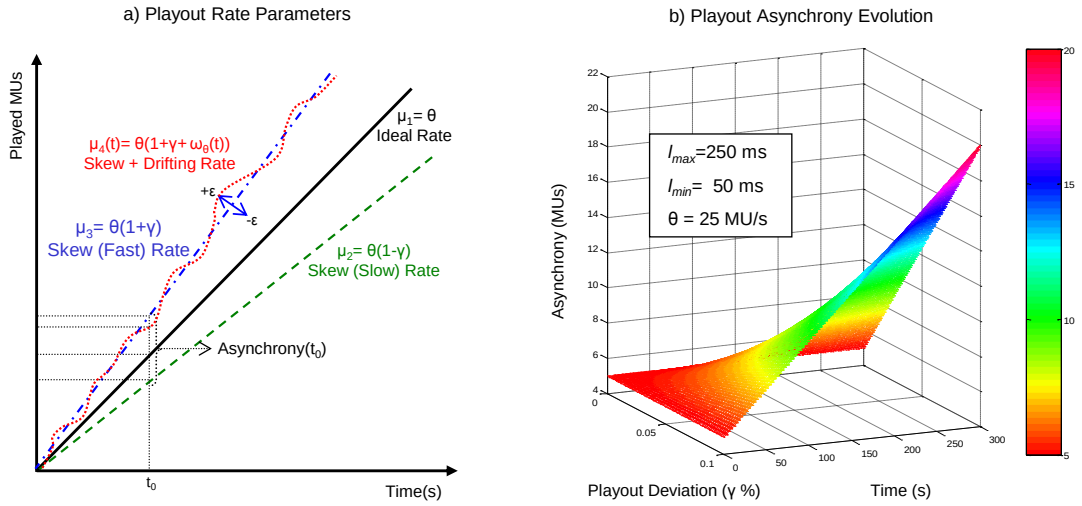


Fig.4. Playout Rate Parameters (left) and Playout Asynchrony Evolution (right).

6. Enhanced RTCP-based IDMS Approach

In this section, we describe the enhanced version of our IDMS approach [10], which in turn is based on two previous protocols: *Feedback Protocol* [30] and *Feedback Global Protocol* [31]. The former uses local clocks whereas the latter uses a global time reference. Both are adaptive, valid for multicast and use a Master/Slave (M/S) scheme and feedback techniques to exchange control information between sources and receivers.

Most of the existing IDMS solutions [1] define new proprietary protocols, with their own specific control packets, that should increase the network load. Currently, many multimedia applications make use of RTP/RTCP standard protocols [9]. On the one hand, the timestamps and sequence numbers mechanisms provided by RTP in each data packet are very useful to reconstruct the original media timing, to handle out of order packets and to detect some possible packet losses at the receiver side. On the other hand, the reporting features provided by RTCP make QoS monitoring possible and, subsequently, they can be used by service providers in multicast distribution services, such as IPTV, for troubleshooting or fault tolerance management [32].

Moreover, these protocols are intended to be tailored through modifications and/or additions in order to include profile-specific information required by particular applications, and the guidelines for doing so are specified in [33].

IDMS involves the collection, summarizing and distribution of RTP packet arrival and playout timings. As this information can be considered as QoS metric (it can reflect the effect of jitter, network load, packet losses, clock skews/drifts, presentation skews, CPU overload, etc.), RTCP becomes a promising candidate for carrying out IDMS. So, we decided to design a new IDMS approach taking advantage of the RTP/RTCP capabilities, by means of including/collecting playback timing in/from new defined and extended reports, which may facilitate implementation and deployment in typical multimedia applications.

Next, the main properties of our enhanced version of our RTCP-based IDMS approach are outlined in order to give an initial overview of its operation:

- It is based on the availability of a global time reference for all the distributed participants involved in a media session (e.g., provided by NTP, GPS, PTP or other solutions).
- The optimum transmission rate for the feedback control messages does not need to be computed, as required by most of the solutions in [1], because the RTCP feedback reports are exchanged, more or less periodically, between all the participants, and the report interval is dynamically adjusted according to the active number of participants and the session bandwidth, as specified in [9].
- Our IDMS proposal is not a full receiver-based solution, as are most of the solutions in [1]. Instead, it follows an SMS scheme, because the synchronization actions are performed by the receivers but in accordance with instructions sent by the source (maestro).
- The synchronization problem is tackled by dividing it into two main phases: first, to ensure that all the receivers start the playout process simultaneously (*Coarse Synchronization*); and then, to maintain the media stream playout processes in a synchronized way between all the receivers throughout the streaming session lifetime (*Fine Synchronization*).
- Every time an *out of sync*[7] situation is detected by the maestro, it can employ several dynamic policies for selecting a master reference playout point for IDMS. The maestro makes use of an M/S Scheme regarding the receivers for the IDMS control: one receiver (real or fictitious) will be selected as the IDMS reference and its playout process will be taken as the reference to determine the state of the playout processes (advanced or delayed from that one) of the other active (slave) receivers.
- Once distributed participants receive a new RTCP control message from the maestro, including playout setting instructions, two reactive synchronization techniques can be employed to restore the synchronicity (aggressive and smooth adjustments).

[7] We refer to this concept to as the situation in which the playout asynchrony (the one between the most advanced and the most lagged receiver of a synchronization group) exceeds an allowable threshold.

6.1. *Initial Playout Instant (Coarse Synchronization).*

In many applications it is required that all the receivers initiate the playout of the media stream at the same time (e.g. a scheduled tele-presentation). We can achieve this by ensuring that all the participants agree on the exact instant to begin the playout of the media stream (Fig. 1), referred to as *Global Initial Playout Instant* (p_{ini}).

To estimate it, once the receivers join to the session, we force them to send several control messages including their global timestamp (sending time). These messages are carried using new defined RTCP Application-Defined (APP) packets, which we called RTCP APP RET.

The source will receive these messages and register the network delay for each incoming packet (difference between source's global time and the one stamped in the received message). Thus, the source can estimate the maximum, minimum and mean (using a temporal window) network delay value for each receiver.

From these values, the source can calculate a proper p_{ini} for all the active receivers in the session. This p_{ini}, specified in relation to the global time reference, is sent to them using another RTCP APP packet, called RTCP APP TIN packet, prior to the transmission of the initial MUs. The process of calculation of the optimum p_{ini} must take into consideration some key factors such as the maximum and/or minimum estimated network delays, jitter values, expected duration of the session, or possible clock deviations, in order to allow the receivers to buffer enough MUs to start the playout and maintain it continuously, thus avoiding underflow and/or overflow situations (playout discontinuities) during the multimedia session lifetime [10]. Right away, the source will start sending MUs (encapsulated in RTP packets), which will be buffered by the receivers until their local clocks reach the p_{ini}. This way, this RTCP-based technique is able to counteract the effect of the delay variability (first term in Eq. 2 and 3) at the beginning of the playout of the media stream/s.

The format for both control packets is identical (see Fig. 5a), but they have different ASCII name (*"RET"* and *"TIN"*, respectively), and in TIN packets the NTP timestamp field is used to include the wall clock time when the receivers' playout processes must start.

Apart for the *Initial Playout Instant*, a multimedia presentation can be divided into several phases or stages, each one with a different *Initial Playout Instant*, which can be used to coarsely synchronize the receivers (Fig. 1). We call this process *Coarse Synchronization*. We assume that a phase or stage ends when the transmission of the media stream is stopped for longer than a specified interval (that can be set by the application) and assume that the next phase is initiated when the transmission of that media streams starts again. In addition, *Coarse Synchronization* actions may be forced in some cases, so as to avoid buffer underflow/overflow situations or to force synchronization adjustments at specific events (e.g. when a network quiz is sent to the contestants).

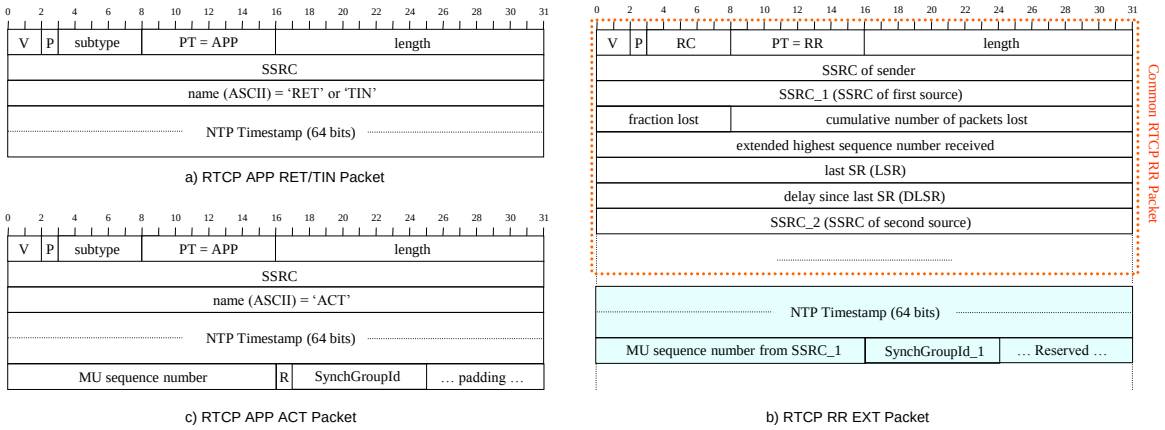


Fig.5. Format of the proposed RTCP Packets.

6.2. *Fine Synchronization (Playout Information Distribution and Setting Instructions).*

Once an *Initial Playout Instant* has been chosen or a *Coarse Synchronization* has been achieved, a mechanism to correct the effect of network fluctuations and/or system dynamics (e.g. clock skews/drifts, network and CPU congestion, network and end-system jitter, etc.) in the playout processes of all the receivers will be required. We call this *Fine Synchronization* because it is not a local synchronization between different streams (inter-stream), but rather a synchronization of the states of the playout processes of the media stream in each receiver (we also refer to this as *Distributed Media Synchronization*). To tackle this problem, we again base our approach on the existence of a global time reference and on the ability to define new extensions to the existing RTCP packets and new types of RTCP messages.

Typically, during an RTP session, when the source starts sending RTP media packets, active receivers regularly send RTCP RR to inform about QoS (network delay, RTT, jitter or loss rate). As [9] allows it, and following the guidelines specified in [33], we decided to extend these feedback reports to facilitate synchronization between receivers, calling them RTCP RR EXT packets. This extension includes the local playout point of each *i-th* receiver: *i)* the 16-bit sequence number of the current MU being played by that receiver (MU_i); *ii)* the 64-bit wall clock time at which the receiver started the playout of that MU, i.e. its playout time (p_i); and *iii)* the 8-bits synchronization group (*cluster*) identifier to which the receiver belongs. This involves quite a short extension of three 32-bit words (although the required fields need 88 bits, the extension must be done in 32-bit words, $3 \cdot 32 = 96 > 88$) which results on a low traffic overhead. Its complete format is illustrated in Fig. 5b, in which the extended block is emphasized in color. Note that these reports (without the extension) would also be sent even if our IDMS approach is not applied.

Once the source has collected the playout information of all the active receivers in a specific cluster, with that information (88 bits per receiver) and a track of the mean limits of the network delay (they can be easily measured using feedback information in each RR packet [9]), it will run a simple algorithm for selecting one receiver as the synchronization (master) reference (discussed

later) and calculate the maximum asynchrony (playout time discrepancy) between all of them, by making a comparison among their local playout points. If the detected asynchrony exceeds an allowed threshold (τ_{max}), i.e. the receivers are out-of-synchronization, the source will multicast a new defined *'action message'* to make them adjust their playout processes. This message is a newly defined RTCP APP packet, called RTCP APP ACT (see Fig. 5c), which includes a target playout point (i.e., a MU sequence number -MU_{ACT}-, the global time at which that MU should be played by all the receivers in a specific cluster -p_{ACT}-, which will depend on the selected reference playout point, and the cluster identifier to which the packet is sent). The total length of this packet is six 32-bit words. The flow chart of the designed synchronization control scheme is sketched in Fig. 6.

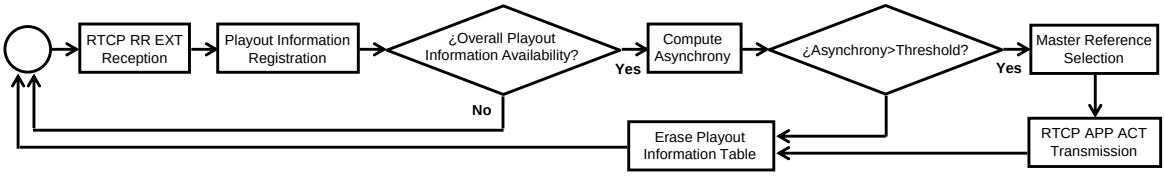


Fig.6. Flow Chart of the Designed Synchronization Maestro Scheme (SMS) for IDMS.

6.3. *Fault Tolerance.*

If a specific RTCP RR EXT packet (just one) sent by a receiver is lost, the synchronization algorithm will not be affected drastically because the source will wait for the reception of the next report from that receiver, since the RTCP messages are sent regularly, as specified in [9]. If more than one successive RTCP RR EXT packets are lost during the session, the source could have to wait for an excessive period of time in order to collect the playout timing from all the receivers. So, we have included a control timer in this evolved version of our IDMS solution that aims to regulate the transmission of the RTCP APP ACT control messages (including playout setting instructions to the receivers). This way, a new RTCP APP ACT message can also be sent as a result of a timeout event of this timer. Every time a new action packet is sent, the timer is newly updated.

However, if the maestro does not receive RTCP feedback reports from a specific receiver for a long period of time, it will consider that receiver has left the session, and it will not take into consideration the playout status of that receiver in the calculation of the synchronization target playout point.

To summarize, an RTCP APP ACT packet will be sent either as a result of an asynchrony (over a threshold) detection or as a result of a timeout event of the control timer.

6.4. *Master Reference Selection Policies.*

We have enhanced our previously proposed SMS, [10], in which a fixed receiver was considered as the synchronization (master) reference, by introducing four new dynamic master reference selection policies: *i)* synchronization to the slowest receiver; *ii)* synchronization to the fastest receiver; *iii)* synchronization to the mean playout point; and *iv)* synchronization to the source nominal rate.

The first strategy consists of selecting the *slowest receiver*, regarding the media stream playout process, as the synchronization reference. Next, we present the pseudo-code for this algorithm. Initially, the first receiver in the receivers' information table (first row) is taken as the provisional master receiver. Then, we compare its playout process with the ones of all the other receivers in order to select the slowest one (MU_i refers to the media unit the *i-th* receiver was playing when it sent the RTCP RR EXT packet, and p_i indicates the global time at which MU_i started its playout process). In such an algorithm, if the playout process of the receiver we are comparing is delayed to the one of the provisional master, that receiver will be selected as the new master. The *i* sub index indicates the *i-th* analysed receiver; *N* is the number of active receivers in the session; and θ reflects the sending/playout rate ($\theta \approx \mu$, in MU/s):

```
MU_master = MU_1              /* MU the i-th receiver is playing */
p_master = p_1                /* playout time for that MU */
SSRC_master = SSRC_1          /* receiver identifier */
for (i = 2, i <=num_receivers, i++)
        if (MU_i > MU_master) THEN
      Diff_MUs_time = (MU_i – MU_master)* (1/θ)
      if (p_i >p_master) then
           Diff_playout_times = p_i –p_master
           if (Diff_playout_times > Diff_MUs_time) then
                MU_master = MU_i
                p_master = p_i
                SSRC_master = SSRC_i
           end if
      end if
        else
      Diff_MUs_time = (MU_master - MU_i)* (1/θ)
      if (p_master <= p_i) then
           MU_master = MU_i
           p_master = p_i
           SSRC_master = SSRC_i
        else
           Diff_ playout_times = p_master - p_i
           if (Diff_ playout_times <= Diff_MUs_time) then
                MU_master = MU_i
                p_master = p_i
                SSRC_master = SSRC_i
        end if
      end if
   end if
end for
```

Such an algorithm can easily be performed by estimating the playout rate skew (see Section 5) of each *i-th* receiver (γ_i) from its local playout point[8], as it is reflected in Eq. 4, and then identifying the minimum value of each one of them (γ_{min}):

$$\gamma_i = [(MU_i - MU_{ini})/(p_i - p_{ini})]/\theta - 1 \quad (4)$$

As a result, the source will calculate a target playout point for IDMS considering the (probably deviated) playout rate of the most lagged receiver (i.e. $\gamma_{master}=\gamma_{min}$):

$$p_{ACT} = p_{ini} + (MU_{ACT} - MU_{ini})/(\theta \cdot (1+\gamma_{master})) \quad (5)$$

Using this method there will not be any skips in the receivers' playout processes, with the consequent discontinuities, since slave (faster) receivers will be forced to pause their playout processes (waiting for the slowest one). So, this strategy could be suitable for multimedia applications with flexible delay requirements and it could enable the inclusion of interactive error recovery techniques through retransmission requests. Besides, in distributed scenarios, where users can compete among themselves (e.g. in battle multi-player on-line games or in network quizzes shows), this policy would be appropriate to guarantee fairness between them [3]. However, if the playout rate of the master (most lagged) receiver is slower than the sender nominal rate, it could suppose the progressive filling of the receivers' playout buffers, which could even overflow. This way, loss of real time sensation could be noticed, affecting to the overall quality. To avoid such a situation, some additional buffering monitoring and control (or Coarse Synchronization) techniques should be employed (out of the scope of this paper).

The second method is the opposite of the previous one and consists of selecting the *fastest receiver* as the synchronization reference. Thus, the target playout point will be calculated from (5) by taking the most advanced playout point (i.e. $\gamma_{master}=\gamma_{max}$). As an example, in networked collaborative real-time games, the efficiency of the work may be improved by adjusting the later playout timings to the earliest one [3]. Nevertheless, we could find the case in which any slower receiver, with bad conditions (long delays, jitter, network load, latency of the operating system, clock deviation, CPU overload, etc.), would be constantly skipping MUs to achieve synchronization, and the continuity of its playout process could be seriously affected. In such a case, if the playout rate of the master is significantly faster than the source nominal rate, the playout buffers of all the receivers may suffer *underflow* (progressive emptying of their buffer occupancy) as the session advances in time, and its application would require the use of novel adaptive buffering control techniques, as in the previous policy.

Note that both algorithms are dynamic processes since the master and slave roles can be exchanged during the session, depending on the aforementioned uncontrollable factors, allowing M/S switching (technique described in [1]).

[8] The local playout point (including its playout deviation) of each i-th receiver is given by: $MU_i=MU_{ini}+\theta\cdot(1+\gamma_i)\cdot(p_i-p_{ini})$.

Another solution for selecting the synchronization reference is to define a virtual playout point (fictitious receiver), obtained as the *mean playout point* of all the active receivers. Thus, the target playout point will be calculated from (5) by averaging the overall estimated playout deviations (i.e. $\gamma_{master}=\gamma_{mean}$). Using this method, the playout processes would be more continuous and smoother since the values of the adjustments would be lower than in the previous policies. However, its use does not guarantee the playout buffer filling (overflow) or emptying (underflow) avoidance since the receivers' playout rate imperfections and system dynamics are in fact unpredictable (there could be a higher/smaller proportion of fast receivers than slow receivers with different deviation values). Also in this case, and in order to avoid this, some buffering control techniques should be employed (out of the scope of this paper).

In all the previous discussed policies, the reporting of an erroneous playout point, either accidental or malicious, may lead to undesired behavior. According to the model adopted, extremely advanced/delayed playout points would produce high adjustments on the receivers' playout processes with the consequent significant loss of real-time/continuity perception. Therefore, the synchronization maestro (source) should consider inconsistent playout information (exceeding configured limits) as a malfunction service and reject that information in the calculation of the IDMS target playout point.

Hence, a new solution is introduced in order to minimize such effect: synchronization to the source nominal rate. In such an algorithm, the source will also act as a virtual master receiver with an *ideal target playout time* (as it knows the value of the global p_{ini}), which is defined as the time at which a specific MU should be played if there are no network delay jitter and playout rate deviations, i.e. setting $\gamma_{master}=0$ in (5). Therefore, the maximum playout asynchrony will be calculated taking also into consideration the playout point of this virtual ideal receiver as an additional receiver in the session. Using this method, if network conditions are quite stable, underflow/overflow situations would be avoided since the playout states of the deviated receivers would be adjusted according to this *ideal playout point* every time τ_{max} is exceeded. Furthermore, this algorithm benefits the accurate receivers because the minor deviation in their playout states means that lower adjustments would be needed to acquire IDMS.

Note that all the above presented policies are appropriate for network scenarios with controlled/bounded delays, i.e. for each *n-th* MU, $l_{min}\leq l_n\leq l_{max}$. Otherwise, we should force several *Coarse Synchronization* phases during the session, or consider AMP techniques to dynamically adjust the playout processes to the network fluctuations (and system dynamics) in order to avoid buffer underflow/overflow situations. This is left for further study.

Once an RTCP APP ACT packet is received, the playout process in each receiver will compare the target playout point (included in that message) with the local one. Consequently, it must adjust its playout process. It can be done by following two possible methods. The first one is based on

simple reactive actions such *'skips/pauses'* (aggressive adjustments), while the second one makes use of AMP (smooth adjustments) to achieve the IDMS goal.

6.5. *Agressive Playout Adjustments (Skips & Pauses).*

Let us consider the *i-th* receiver is playing a specific MU -MU_i- at p_i instant (local playout point). That receiver would consume the successive MUs with a (possibly deviated) playout rate of $\mu_i \approx \theta \cdot (1+\gamma_i)$ MU/s. So, it would play out the MU_{ACT} at p'_{ACT} instant, which possibly does not match with p_{ACT} instant (IDMS target). Let $\Delta_{n,i}$ denote the asynchrony, for each *n-th* MU, between the evolution of the local playout point of the *i-th* receiver (p'_{ACT}) and the target playout point included in each action message (p_{ACT}):

$$p_{ACT} = p'_{ACT} + \Delta_{n,i} = \left[p_i + \frac{1}{\mu_i}(MU_{ACT} - MU_i) \right] + \Delta_{n,i} \quad (6)$$

If $\Delta_{n,i}>0$, the playout process of the *i-th* receiver is advanced to the target playout point. So, that receiver must *'pause'* (stop playing) its playout process during $\Delta_{n,i}$ seconds to synchronize, by setting the service time for the current *n-th* MU to $s_{n,i}+\Delta_{n,i}$ seconds, causing a probable *freezing effect* (Fig. 7a). Consequently, the playout delay for the next MU, d_{n+1}, will be increased (i.e. p_{n+1} will be delayed). Otherwise, if $\Delta_{n,i}<0$, the playout process of the *i-th* receiver is lagged. In that case, it must *'skip'* (jump or move forward) a certain number of MUs until the detected asynchrony is reduced to a lower value than the service/presentation time for one MU. Thus, every time an *'action message'* is received, the number of skipped MUs will be given by the integer value from the result of dividing $(\Delta_{n,i})/(s_{n,i})$. This way, d_{n+1} will be reduced (i.e. p_{n+1} will be advanced). As shown in the evaluation section, these reactive actions will result in an overall synchronization status (within acceptable limits).

6.6. *Smooth Playout Adjustments: Adaptive Media Playout (AMP)*

The above reactive playout adjustments could originate a noticeable degradation of user perception as regards the quality of the received media streams because, on the one hand, some important information may not be presented to the users (due to the *skipped* MUs) and, on the other hand, a sensation of loss of continuity may also be noticed (due to the *paused* MUs). Thus, a novel AMP scheme has been adopted to minimize the occurrence of the above long-term playout discontinuities.

The flow chart of this algorithm is sketched in Fig. 8. Initially, the playout controller deals to play out the buffered MUs at a non-adaptive playout rate given by $\mu_{n,i}=1/(s_{n,i})$, as they were generated by the source. Once each *n-th* MU finishes its presentation period, the next *(n+1)-th* MU must be played out and the buffer occupancy must be updated.

Active receivers include their current local playout point (MU_i, p_i) in each RTCP RR EXT they send to allow the source collect the overall playout status. Once an RTCP APP ACT is received,

the target playout point is registered and the AMP process is triggered. At this point, the AMP approach will attempt to either fasten or loosen the playout rate in order to distribute $\Delta_{n,i}$ among all the remaining MUs to reach the target playout point, as can be observed in the Fig. 7b. It can be done by means of increasing/decreasing the playout time of each *n-th* MU a value of $\delta_{n,i}=(\Delta_{n,i})/(MU_{ACT}-MU_i)$ seconds (as a consequence, $s_{n,i}+\delta_{n,i}$).

However, to perform the AMP scheme we must consider the allowed ratio within which the video playout speed can be varied without degrading the media quality. We are primarily concerned with AMP for video streaming. Issues related to AMP for audio streaming have been addressed elsewhere [13], but are not considered in this work. Subjective tests have shown that playout speed variations of up to 25% are often *unnoticeable* and, depending on the content and the frequency of the adjustments, variations up to 50% are sometimes *acceptable* ([13]-[15]). Thus, we will assume in our simulation tests that video playout adjustments up to 25% lead to unnoticeable quality impairments, and we define a playout factor for each *n-th* MU in each *i-th* receiver ($\varphi_{n,i}$) to specify this variation ratio, whose optimal value is computed, as smooth as possible, combining (6) and (7), using equation (8):

$$p_{ACT} = p_i + \frac{1}{\mu_i(1+\varphi_{n,i})}(MU_{ACT} - MU_i) \tag{7}$$

$$\varphi_{n,i} = \frac{1}{1+(\delta_{n,i}/s_{n,i})} \tag{8}$$

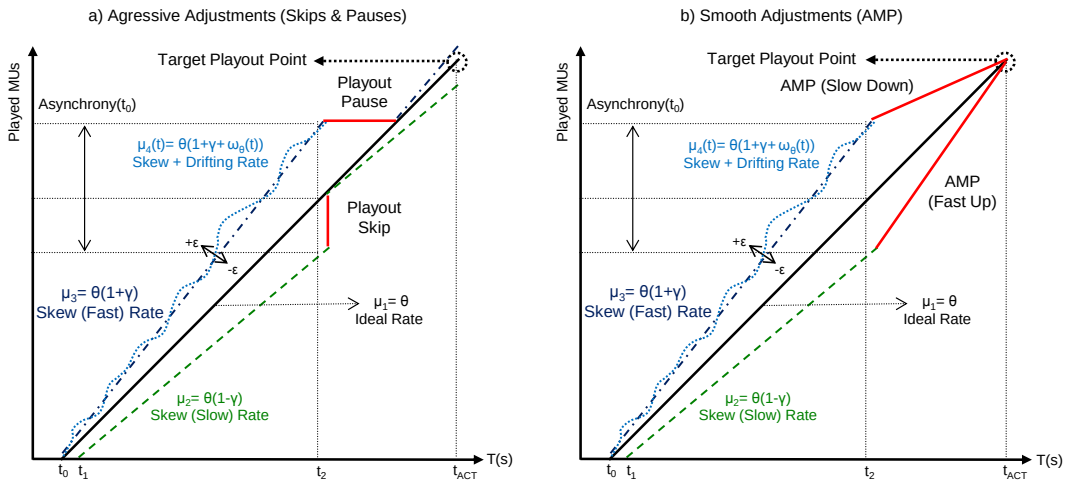


Fig.7. Playout Adjustments to Acquire IDMS.

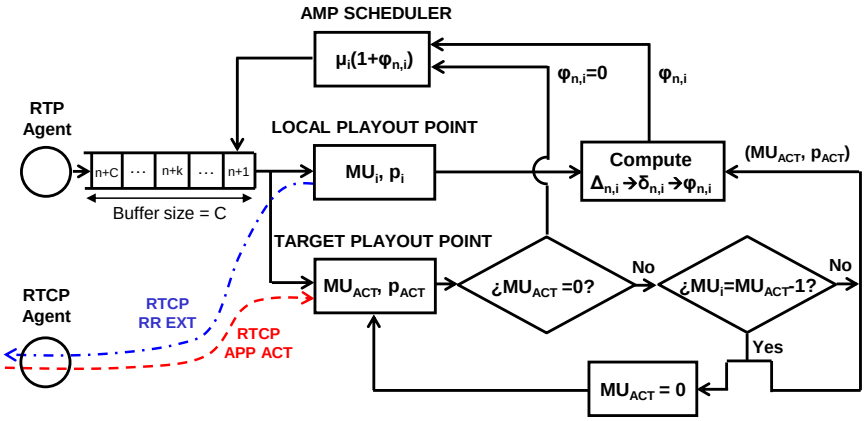


Fig.8. Flow Chart of the Designed AMP Scheme.

Note that if the calculated $\varphi_{n,i}$ is higher than 25%, it will be bounded to that maximum scaling ratio (i.e. $|\varphi_{max}|\leq 0.25$). It may occur when τ_{max} is set too high or when there are not enough buffered MUs to smoothly distribute the detected asynchrony between them. So, a proper election of p_{ini}, τ_{max}, MU_{ACT} or the master selection policy must be accomplished to minimize noticeable playout interruptions. The AMP process will be finished once the target playout point (MU_{ACT}, p_{ACT}) is reached (i.e. MU_{ACT}, $\varphi_{n,i}$ and $\delta_{n,i}$ will be set to zero) and will not be performed again until the reception of a new RTCP APP ACT packet.

As a summary, all the different symbols that have been defined in this paper and their meaning are in Table 1.

Symbol	Units	Meaning
θ	MU/s	Source Nominal Rate
γ_i	%	Playout Rate Skew (Deviation) of the *i-th* Receiver
$\omega_i(t)$	%	Playout Rate Drift of the *i-th* Receiver
ε	%	Maximum Playout Rate Drift (Fluctuation)
$\mu_i(t)$	MU/s	Playout (Service) Rate of the *i-th* Receiver
t_n	Time Unit	Transmission time of the *n-th* MU
$r_{n,i}$	Time Unit	Reception time of the *n-th* MU in the *i-th* receiver
$l_{n,i}$	ms	Network delay of the *n-th* MU in the *i-th* receiver
$b_{n,i}$	ms	Buffering delay of the *n-th* MU in the *i-th* receiver
$d_{n,i}$	ms	Playout delay of the *n-th* MU in the *i-th* receiver
$p_{n,i}$	Time Unit	Playout time of the *n-th* MU in the *i-th* receiver
τ_{max}	ms	Maximum allowable asynchrony threshold
$\Delta_{n,i}$	ms	Detected playout asynchrony for the *n-th* MU in the *i-th* receiver
$\delta_{n,i}$	ms	Distributed asynchrony to correct for the *n-th* MU in the *i-th* receiver
$\varphi_{n,i}$	%	Playout Factor (adjustment rate) for the *n-th* MU in the *i-th* receiver

Table 1: Nomenclature Used in this Work

7. Evaluation

7.1. *Simulation Models, Scenario and Setup*

Modeling and simulations were conducted using NS-2. We tested our IDMS approach in a multicast scenario with seven geographically distributed receivers, with variable network delays to the source, belonging to two different clusters (Table 2). The source transmitted a media stream with a specific rate of $\theta=25$ *MU/s.* Additionally, different intensive background traffic was configured over the network topology in order to cause jitter variability. The receivers' playout rate imperfections (skews and drifts) were configured as shown in Table 2. These values were set larger than reasonable deviations in inexpensive oscillators (which can vary between 10-100 *ppm* [34]), in order to force higher asynchronies between the receivers, and to test if they were successfully handled by our IDMS approach. Moreover, in order to corroborate the M/S switching capabilities of our approach, those values were intentionally changed in two of the receivers belonging to Cluster 1 (C1) at *300-th* second, midpoint of the simulation (Table I). The duration of each simulation was set to 10 minutes and the value of τ_{max} (maximum allowable asynchrony threshold) was set to 80 ms in order to trigger playout corrections slightly before reaching an asynchrony of 100 ms that can be already perceivable and annoying in some cases, according to the studies in [4].

Receiver	Cluster	Mean RTT (ms)	Rate Skew, γ (%)	Rate Drift, ε (%)
R1	C1	288	0.03 %	0.02 %
R2	C1	125	- 0.02 % → - 0.03 %	0.02 %
R3	C1	44	- 0.05 % → - 0.02 %	0.02 %
R4	C1	125	- 0.015 %	0.02 %
R5	C2	82	0 %	0.02 %
R6	C2	288	- 0.02 %	0.02 %
R7	C2	161	0.01 %	0.02 %

Table 2: Receivers' Parameters

7.2. *Simulation Results*

1) Intra-Stream Synchronization. Fig.9a illustrates the buffering and playout delays for one of the receivers (R1), when there were not playout rate deviations ($\gamma_1=0, \varepsilon_1=0$), in two different network load cases. Initially, we ran a single simulation without background traffic. In that case only RTP traffic and the associated RTCP feedback reports were exchanged. As a result, both the buffering ($b_1 \approx p_1 - l_1 \approx 500-(288/2) \approx 500-144 \approx 356$ *ms*)[9] and playout ($p_1 \approx 500$ *ms*) delays for that receiver were quite stable during the multimedia session, because the jitter values for the media traffic were negligible, as it can be observed in the figure.

[9] As shown in Table 2, the mean RTT for R1 was 288 ms, thus we assume that its network delay is around $l_1 \approx (288/2) \approx 144$ *ms*

Next, the network load was increased by transmitting intensive background traffic (different Pareto, CBR and FTP traffic flows were configured over the network topology) in order to assure that the total amount of network traffic (RTP data stream + background traffic) was near the links' capacity at some instants during the session. As expected, as the network traffic increased, the buffering delay for that receiver presented significant fluctuations during the session, mainly due to the higher queueing delays at the intermediate routers (as the RTP stream must contend with the background traffic). This way, the jitter values for the received media stream were much more variable than in the previous case (the incoming MUs were not received in a timely synchronized manner). However, in both cases, the designed playout buffering policy was able to smooth out the effect of the time-variant jitter values, by delaying the incoming MUs at the playout buffer, and then removing them from the buffer queue with the same rate as they were generated by the source (constant rate in this case), thus reconstructing the original media timing and presenting a uniform playout delay (high quality of intra-stream synchronization).

The effect of the playout rate imperfections has been reflected in Fig.9b. This figure illustrates the playout delay evolution for one of the receivers, when different playout rate parameters were configured to it in the simulation setup. As can be observed, when that receiver presented an ideal playout rate ($\gamma=0$, $\varepsilon=0$), the incoming MUs were played out at a constant rate (the temporal pattern of the incoming media stream is CBR), thus presenting a uniform playout delay during the session. When that receiver had a drifting rate ($\gamma=0$, $\varepsilon\neq0$), the playout rate presented slight fluctuations over the previous ideal playout rate due to the random oscillations in the playout interval periods. Finally, if the playout rate of the receivers is slower/faster than the sender nominal rate, i.e. it presents a constant playout rate skew ($\gamma\neq0$), the playout buffer of the receiver may become *flooded/emptied* with MUs if such effect is not corrected/handled during the streaming session. In the third case of Fig.9b., it can be observed that the playout delay for that receiver was progressively increasing due to the configured playout rate deviations to it ($\gamma=0.05\%$, $\varepsilon=0.03\%$), probably causing a buffer overflow situation if the multimedia session had a long duration. The effect of these playout rate imperfections can drastically increase the asynchrony between distributed receivers. Nonetheless, these undesirable effects can be handled or reduced if an effective IDMS solution is applied.

2) Global Initial Playout Instant. Despite the different Round Trip Time (RTT) values for each receiver in both clusters, measured from each RTCP RR sent by them during the simulations (Table 2), we can notice in the zoom view of Fig. 10a, and also in some of the other presented graphs, that all of them were perfectly synchronized at p_{ini}, because the source indicated to them a pre-specified *global initial playout delay* (d_{ini}) of 500 ms in the initial phase of our IDMS approach.

3) IDMS Control. Fig. 10a illustrates the playout delay evolution of three distributed receivers belonging to C1 (Table 2) when the designed IDMS solution was enabled. It can be observed that

the asynchrony between the playout states of those receivers progressively increased mainly due to the configured deviations in their local playout rates (Table 2). As a result, every time the source detected a playout time discrepancy greater than τ_{max} (80 ms), it sent an RTCP APP ACT packet to make the receivers adjust their playout timing, according to a target playout point included in that control message, which was calculated taking into consideration the collected playout information from one of them (R1 in that case). Fig. 10b illustrates the same process but, in this case, a new receiver (R4) joined to the on-going multimedia session at *60-th* second. As a result, the source sent a new RTCP APP ACT packet in order to allow R4 to achieve synchronization as soon as possible, although the detected playout asynchrony between all the other receivers was lower than τ_{max} at that moment.

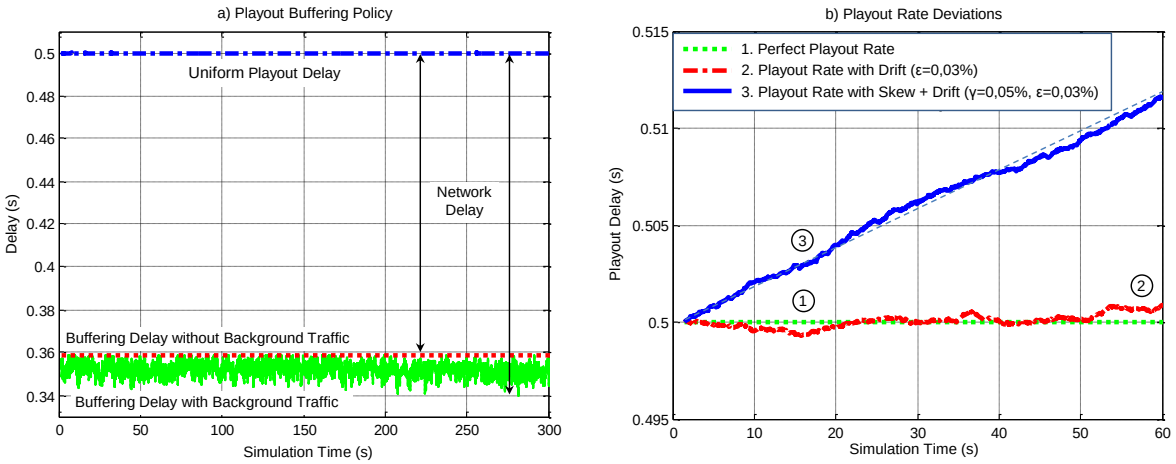
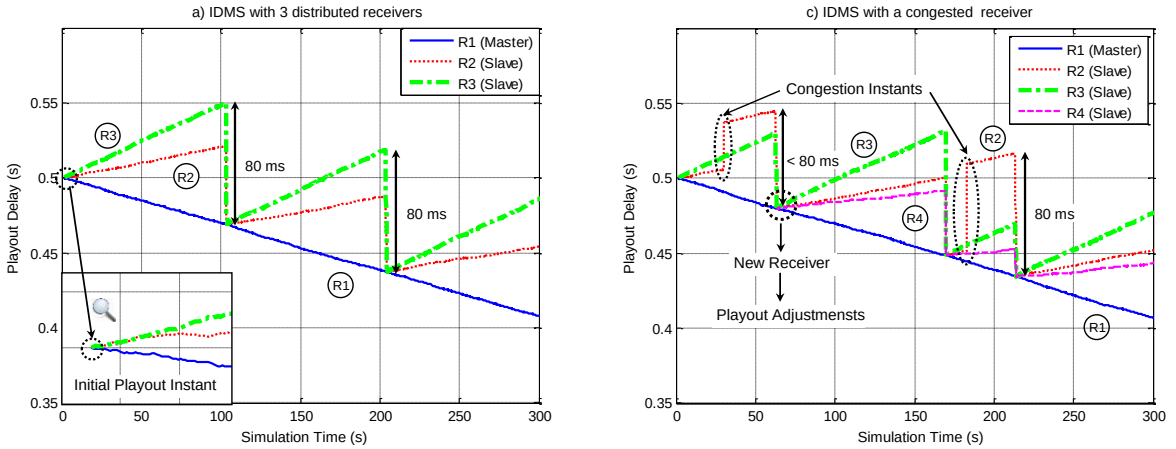


Fig.9. Intra-Stream Synchronization Approach (Playout Rate Parameters).



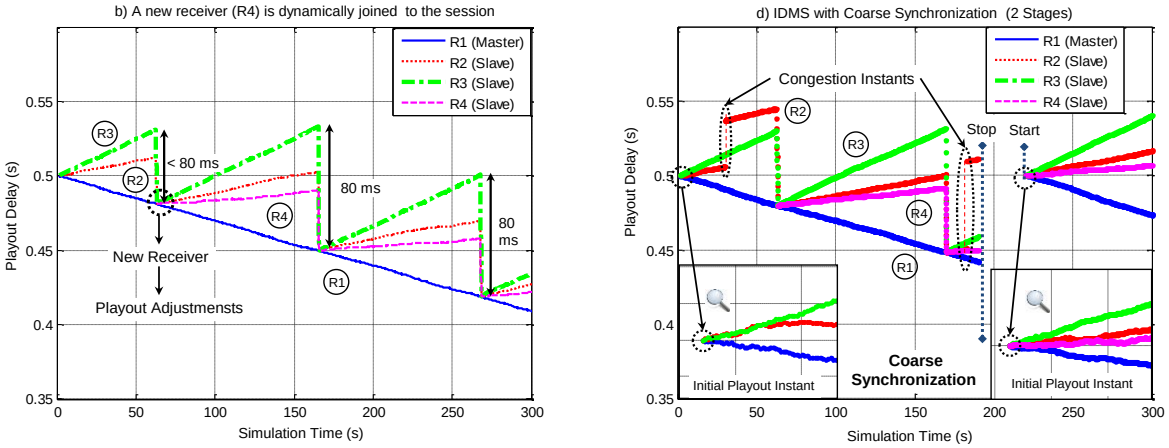


Fig.10. Playout Delay evolution to Acquire IDMS (Cluster 1).

Discontinuities in the media playback and loss of synchronization may also result from dynamic changes in network conditions and from congestion situations at the end nodes (e.g. due to CPU overload). For that reason, we have also included an optional *congestion module* that runs over the local playout process of each node. This module has been configured by adopting an Exponential ON/OFF distribution in which the active period (ON) implies severe congestion situations, and the inactive period (OFF) symbolizes no congestion cases (i.e. normal playout process). This way, every time the active period is triggered, the MU being played out at this moment is stretched until this congestion period ends, causing a probable *freezing effect*. Such effect has been reflected in Fig.10c and in Fig.10d. In such cases, a congestion module with the following parameters (t_{ON}*=40 ms,* t_{OFF}*=120 sec*) was applied to the playout process of R2. As a result, it can be observed that the playout asynchrony between the distributed receivers increased every time that receiver was congested. For example, the third reactive playout adjustments performed by the receivers occurred significantly before in Fig.10c (at *220-th* second, approximately) than in Fig.10b (at *270-th* second), in which congestion situations were not considered.

Next, the multimedia session was divided into two stages, as in Fig.1, so as to corroborate the *Coarse Synchronization* capabilities of the implemented IDMS solution. As we can observe in Fig.10d, the media session was stopped at *180-th* second and was re-started at *220-th* second. As a result, an additional *Coarse Synchronization* phase was launched to ensure concurrently synchronized playout points in all the distributed receivers at the beginning of the playout in the second stage.

4) Master Selection Policies and Reactive Playout Adjustments.

4.1) Synchronization to the Fastest Receiver. Fig. 11a illustrates the playout delay evolution of 3 receivers (Table 2) to acquire IDMS when the fastest receiver was selected as the synchronization reference, by using aggressive adjustments. In that case, R1 played out the incoming MUs with a higher rate because it had the highest (positive) skew of all of them (Table 2). Thus, every time τ_{max}

was exceeded, the receivers had to adjust their playout timing according to the target playout point, calculated taking into consideration the collected playout information of the fastest one ($\gamma_{master}=\gamma_{max}=\gamma_1$). We can appreciate that, in that simulated session, slower (slave) receivers had to *'skip'* zero, one or two MUs[10] ($\tau_{max}=2\cdot s_{n,i}=80$ *ms*), as a consequence of the reception of each *'action message'*. However, there were not any *'pauses'* in the receivers' playout processes during the session. The same process, but enabling the AMP mechanism, is illustrated in Fig. 11b. In such a case, lagged receivers were more closely and fine-grained synchronized than using aggressive adjustments because they smoothly fasted up their playout rate to minimize the estimated asynchrony. Thus, the source would send a minor number of RTCP APP ACT messages if the multimedia session had a long duration. Additionally, Fig.12a is the same case as the previous one, but here all the receivers in both clusters have been considered. In such a case, the fastest receivers in each cluster were selected as the master synchronization reference (R1 and R6, respectively). Thus, the synchronization actions (playout asynchrony monitoring and reactive playout adjustments) were performed separately for each cluster.

In the previous graphs, in which the synchronization to the fastest receiver policy was employed, we can notice a significant reduction in the playout delay in all the receivers as the media session advanced in time, which caused an inherent progressive emptying of their buffer occupancy.

We can notice a major number of playout adjustments in the receivers belonging to C1 because their playout rate deviation parameters were higher than those of the ones in Cluster 2 (Table 2). So, we mainly focus on the measurements in the receivers belonging to C1, and present statistics of the playout adjustments in that cluster, for each one of the adopted master reference selection policies. Concretely, Table 3 summarizes the total number and the percentage of adjusted MUs for each receiver in C1 when using both aggressive (i.e. *skipped/paused* MUs) and smooth adjustments (i.e. MUs that were involved by the AMP process) during the 10-minute session. Also, the fourth column of Table 3 indicates the maximum variation of the buffer occupancy during the session lifetime.

Using the above master reference selection policy (IDMS to the fastest receiver), we can observe that the buffer fullness level (measured in time, not in MUs) was reduced more than 100 ms for all the receivers (-184.2 ms for R1) during the 10 minute session. Note that the maximum reduction is limited by the initial buffering delay (b_{ini}). Thus, this strategy may not be appropriate if the playout rate of the master is significantly faster than the source nominal rate, because the playout buffers may suffer underflow, and its application would require the use of novel adaptive techniques (e.g. buffer fullness monitoring) or *Coarse Synchonization* actions to avoid such situations, without need of increasing significantly the initial playout delay (d_{ini}) for all the receivers, by means of increasing the buffering delay in all of them (left for further study).

[10] Only entire MUs can be skipped, each one with a duration of $1/\theta$=40 ms/MU.

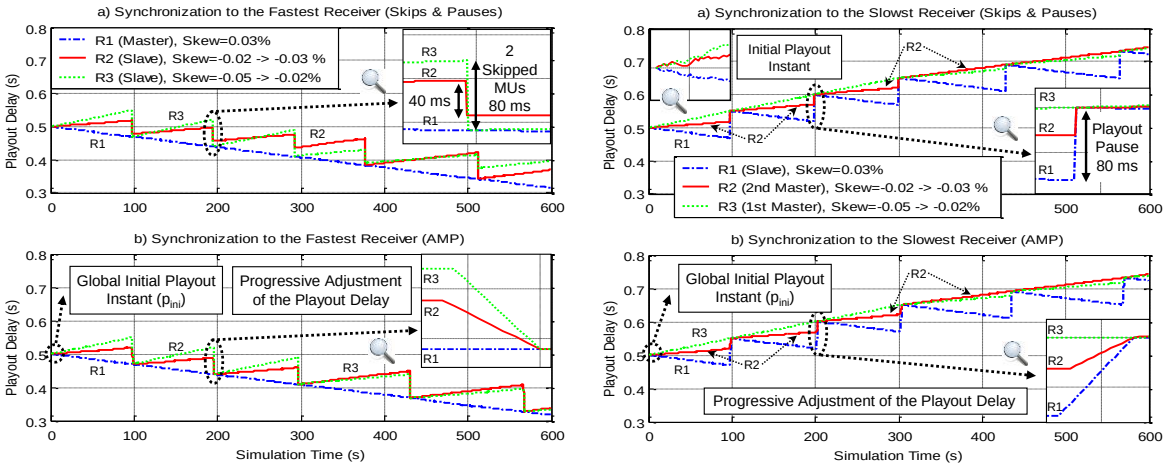


Fig.11. Playout Delay evolution to Acquire IDMS (Fastest/Slowest).

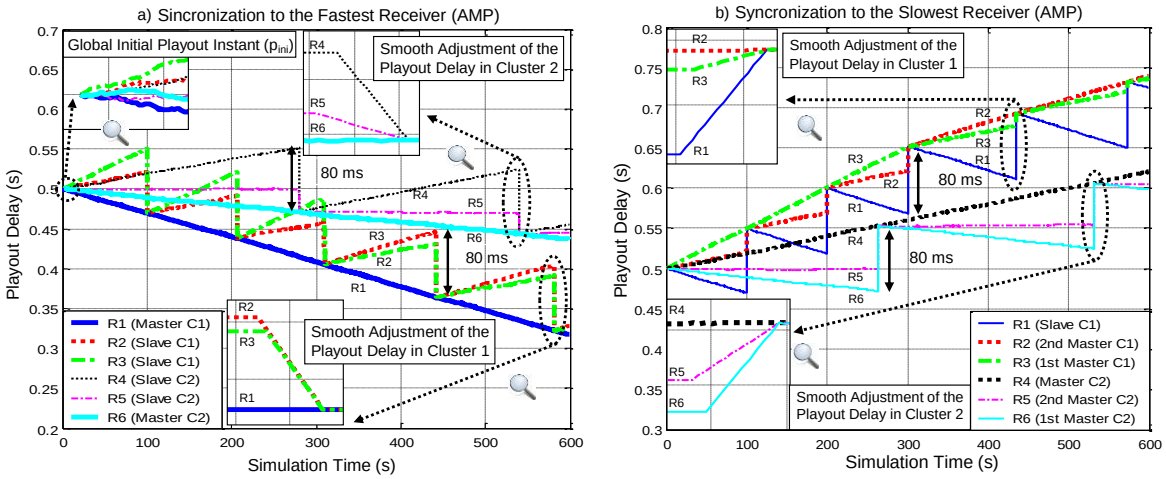


Fig.12. Playout Delay evolution to Acquire IDMS (C-to-C).

4.2) Synchronization to the Slowest Receiver. Fig.11c illustrates the playout delay evolution of 3 receivers in C1 to acquire IDMS when the source selected the slowest receiver as the synchronization reference ($\gamma_{master}=\gamma_{min}$), by using aggressive adjustments. In that case, we can observe that slave (faster) receivers were more closely synchronized to the target playout point than in the previous case because they *'paused'* specific MUs an exact value of $\Delta_{n,i}$ seconds. We can notice in the third column of Table 3 that R1 (fastest) had to make significant pauses ($\Delta_{max}=82.2$ *ms*) every time an *action message* was received, waiting for the slowest one (master). Nonetheless, there were not any *'skipped'* MUs when this algorithm was employed. In addition, as the playout rates of R2 and R3 were slower than the source rate, we can appreciate in Table 3 (fourth column) that using this policy (synchronization to the slowest receiver) the playout buffers were gradually flooded with MUs. So, large capacities of the playout buffer would be required, with the possible degradation of the real-time (loss of continuity) perception. Thus, as discussed in Section 6.4, this

strategy may not be appropriate if the playout rate of the master is significantly slower than the source nominal rate, because the playout buffers may progressively suffer overflow, and its application would require the use of novel adaptive buffering control to avoid such situations. As an example, in a 90-minute on-line football match session, Coarse Sync techniques could be triggered at the half time or when the game is stopped by any action (corner, fault, …).

The responsiveness of such policy when AMP adjustments were used is illustrated in Fig.11d. This graph clearly reflects the effect of the M/S switching technique: initially R3 was the slowest one, but the playout rate skews of R2 and R3 were intentionally changed so as to convert R2 as the new master (Table 2). As in the previous policy, Fig.12b also shows the same process when all the receivers belonging to both clusters were active in the session (C-to-C). In such a case, the slowest receivers in each one of the clusters were selected as the synchronization master references.

4.3) Synchronization to the Mean Playout Point. Fig. 13a shows the evolution of the playout processes of all the receivers in C1 when they were synchronized to the mean playout point, by using aggressive adjustments. This solution provided more fine-grained and less frequent playout adjustments than above ones (R2 only had to make a small pause of 8.9 ms during the session, as shown in Table 3). The lower graph (Fig.13b) illustrates the same process when the AMP mechanism was enabled. In that case, advanced/lagged receivers smoothly slowed down/fasted up their playout timing to synchronize, thus avoiding long-term playout discontinuities (*skips/pauses*). However, this solution cannot guarantee buffer overflow/underflow situations because, as discussed in Section VI, the existence of inaccurate (extremely advanced/lagged) receivers would have a quantitative impact on the calculation of target (mean) playout point. So, as in the previous policies, novel dynamic and adaptive techniques should be adopted.

4.4) Synchronization to the Source Nominal Rate. This policy solves the above undesired situations. We can notice in the third column of Table 3 that R1 (fastest) had to make small pauses (Δ_{max}*=23.7 ms*), that might be imperceptible by users, and slow receivers (R2 and R3) had to *'skip'* a small number of MUs during the session when this policy was selected, by using aggressive adjustments. So, using this method, accurate receivers will not have to make significant adjustments in their playout timing. We can observe in Fig. 13c that the playout delay evolution was kept quite uniform in the receivers belonging to C1 during the session, which is a desired feature in real-time multimedia services. As it can be appreciated in Table 3, the variation of the buffer occupancy in all those receivers was bounded to a lower value than $\pm\tau_{max}$ (80 ms) during the session. Thus, starvation situations would be avoided if the network conditions are quite stable. Finally, the lower graph (Fig.13d) illustrates the playout rate variation for all the receivers (belonging to C1) when this policy was selected, by using AMP as reactive synchronization technique (smooth adjustments). This figure corroborates that the playout rate was varied within perceptually tolerable ranges in order to acquire IDMS. In addition, the last column in Table 3

indicates the maximum playout factor for all the adopted master selection policies in all the receivers belonging to C1, and reflects that it was inferior to 25% of the source nominal rate in all the cases ($|\varphi_{max}|\leq 0.25$) and, therefore, it would not have noticed by users, as indicated in Section 6.5.

Generally, using smooth adjustments (AMP), long-term discontinuities were avoided, although the total number of adjusted MUs was greater than the total number of *skipped/paused* MUs when aggressive adjustments were employed (fifth and third column in Table 3, respectively). However, in none of the master selection policies the percentage of adjusted MUs was higher than 0.4 % of the total MUs, which is a very low value that might be unnoticeable by users.

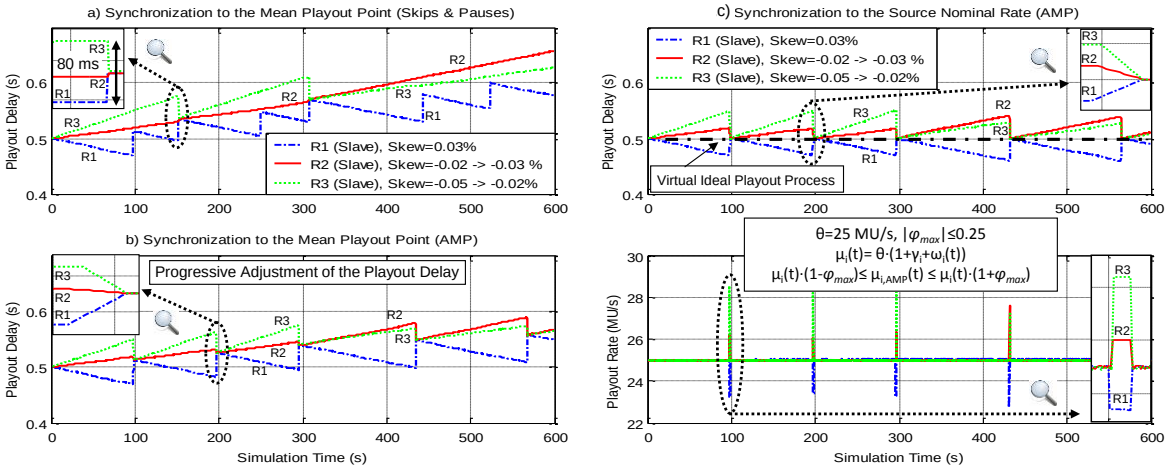


Fig.13. Playout Delay evolution to Acquire IDMS (Mean/Source).

Receiver		Aggressive Adjustments		Smooth Adjustments	
	Master Selection Policy	- Skipped (%) / + Paused (Δ_{max}) MUs	Buffer Fullness Variation (ms)	Total Adjusted MUs (%)	φ_{max}
R1	Fastest	0 / 0	- 184.2	-	-
	Slowest	0 / +4 (82.2)	+ 215.8	61 (0.4)	- 0.16
	Mean	0 / + 6 (54.7)	+ 77.9	56 (0.37)	- 0.09
	Source	0 / + 5 (23.7)	$\leq \lvert\tau_{max}\rvert$	43 (0.3)	- 0.08
R2	Fastest	- 7 (0.05) / 0	-129.8	57 (0.38)	+ 0. 23
	Slowest	0 / + 3 (21.9)	+ 223.8	64 (0.4)	- 0.08
	Mean	0 / + 1 (8.9)	+ 158.2	55 (0.37)	+ 0.06
	Source	- 3 (0.02) / 0	$\leq \lvert\tau_{max}\rvert$	48 (0.32)	+ 0.11
R3	Fastest	- 8 (0.05) / 0	- 104.9	52 (0.35)	+ 0.24
	Slowest	0 / + 2 (14.5)	+ 235.4	62 (0.4)	- 0.05
	Mean	- 2 (0.01) / 0	+ 127. 8	53 (0.35)	+ 0.1
	Source	- 4 (0.025) / 0	$\leq \lvert\tau_{max}\rvert$	49 (0.33)	+ 0. 12

a. 14967 MUs were sent during the multimedia session

Table 3: Summary of Playout Adjustments in Cluster 1

5) Buffer Fullness Variation. To conclude the evaluation, a new ideal receiver with a perfect playout rate (without deviations) joined the multicast session, belonging to C1. The playout controller of that receiver was able to play out the incoming MUs with the same rate as they were generated by the multimedia source. As a result, we measured the buffer delay for the incoming MUs in that receiver, for each one of the adopted master reference selection policies. We can observe in Fig. 14 that the buffer delay was kept quite uniform during the multimedia session (the appreciated fluctuation is due to the jitter delays in the simulated scenario) until the reception of a new *'action message'* from the maestro. At this moment, that receiver had to adjust their playout timing in order to correct the estimated playout time discrepancy, according to the target playout point included in that control message. As expected, when the synchronization to the slowest receiver policy was employed, the buffer delay for the incoming MUs was progressively increasing. This means an inherent filling of the playout buffer that could even overflow if the multimedia session had a long duration. On the contrary, when the synchronization to the fastest receiver policy was applied, the buffer delay in this receiver was progressively decreasing during the session lifetime, because the playout rate of the master receiver was higher than the source nominal rate ($\gamma_{master}=\gamma_1>0$). As a consequence, the buffer occupancy was gradually emptying. For the remaining two policies, we can observe a smooth evolution of the buffer delay, especially when the synchronization to the source nominal rate was employed. So, the buffer fullness level was moderately stable during the multimedia session. However, as previously discussed, although the synchronization to the mean playout point policy may minimize the number and the values of the playout adjustments in all the receivers, this algorithm is significantly affected by the existence of receivers with considerably deviated playout timings because the IDMS target playout point is calculated by averaging the collected playout point of all the active receivers in the session. In this case, the mean playout rate was slightly slower than the nominal playout rate, resulting in an increasing playout delay that, depending on the session duration, could culminate in an overflow situation.

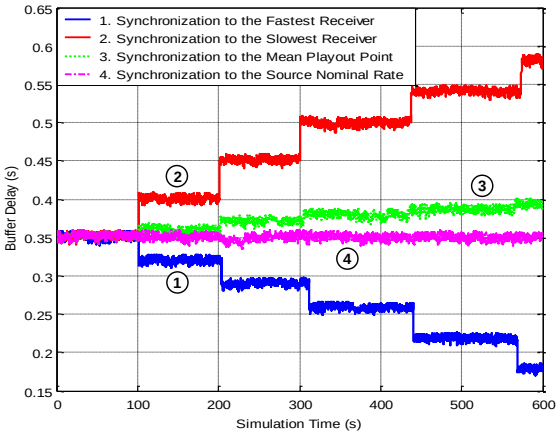


Fig.14. Buffer Delay evolution to Acquire IDMS.

6) Traffic Overhead. The number of RTCP RR EXT packets sent by each receiver during the 10 minute session in each one of the simulations was around 2% of the total RTP data packets sent by the media server (~15000 RTP packets). These amounts are similar to our measurements in real scenarios ([5] and [10]). Note that the same number of RTCP RR packets would have been sent even if our IDMS approach was not applied. The number of RTCP APP ACT packets sent by the source depends on the detected asynchrony between the playout states of the receivers in each cluster. As an example, when the synchronization to the slowest and to the fastest receiver were employed, 5 packets were sent to Cluster 1 and only 2 packets to Cluster 2 (because the playout deviations in that cluster were minor). These amounts are significantly lower than the ones obtained in [5], maybe because in that case each receiver had to play out three different media streams (we also carried out inter-stream synchronization in that work) and the computers could become overloaded at some instants. Thus, the traffic overhead added by our IDMS approach is very low since it is only due to the extension included in each RTCP RR packet (96 bits) sent by the receivers and the size of each RTCP APP ACT packet (192 bits) sent by the maestro to correct the receivers' playout timing, but only if an *out of sync* situation is detected.

8. Conclusions and Future Work

The simultaneous synchronization of the playout processes among distributed receivers (IDMS) is an important requirement in a variety of emerging multimedia applications, such as Social TV or networked multi-player games. This paper has presented the implementation and evaluation of a preliminary version of an RTCP-based Inter-Destination Multimedia Synchronization (IDMS) approach in a simulation framework (NS-2). For that purpose, we previously developed a more complete and accurate implementation of RTP/RTCP protocols for that network simulator. Then, this RTP/RTCP code has been extended to include the implementation of the preliminary version of our IDMS approach, which was presented in [5], by means of: *i)* extending and defining new

RTCP packets; *ii)* designing a proper playout buffering policy; *iii)* implementing new control timers; *iv)* including new methods to distribute and gather the playout timing of the receivers, to control the playout asynchrony, and to report playout setting instructions; and *iv)* implementing reactive techniques (playout adjustments actions) to synchronize. Moreover, the following enhancements have been included to our previous IDMS approach [10]: *i)* adoption of several dynamic master reference selection policies; *ii)* inclusion of coarse synchronization phases; *iii)* IDMS management for independent clusters (logical groups) of receivers; and *iv)* design of an AMP scheme to minimize playout interruptions.

Simulation results have been very satisfactory since they prove the ability of the coordination of both IDMS and AMP techniques to keep the asynchrony between the playout states of the receivers in each one of the clusters within acceptable limits while minimizing perceptible playout discontinuities (skips/pauses), and hardly increasing the network and computational load. The implementation of the IDMS approach in this simulation framework will allow us to test its performance and responsiveness in more heterogeneous scenarios, under different network conditions. Additionally, a comparison between the results of the real test-beds ([5] and [10]) and the simulation studies could also be performed in the near future.

Despite the operational idea of the IDMS approach is valid for both stored media content (e.g. Video-on-Demand -VoD- services) and live media content (e.g. IMS-based broadcast IPTV channels), as discussed in [6], there still remain some challenges to be addressed in order to perform IDMS in live broadcast services, such as interactive Social TV.

Our future work will address the following issues: *i)* a more exhaustive (objective and subjective) assessment of the proposed techniques for IDMS, implementing them in real synchronous media sharing applications in order to perform real-world experiments to analyze the effect of de-synchronization and of the application or not of our adaptive techniques over the user experience (QoE); *ii)* design of a coordinated AMP scheme to dynamically control both buffer fullness level and playout asynchrony between distributed receivers: *iii)* study of source overload and scalability issues regarding the number of clusters and receivers in each one of them; *iv)* optimization of the feedback reporting mechanisms; *v)* adaptation of our IDMS solution to be applied in event-based applications, such as networked games; *vi)* comparison between the different synchronization schemes or architectures (SMS, DCS, M/S and network-based) that have been adopted by the research community in the literature; *vii)* comparison between our IDMS approach and other proposed solutions to acquire IDMS, as the ones presented in [21] and [28], by implementing them in the simulation framework; *viii)* make our IDMS approach valid to synchronize bidirectional (interactive) services; and *ix)* possible standardization of the proposed IDMS solution by the IETF.

The source code and several simulation examples will be soon available at the web page of author's research group: http://personales.gan.upv.es/~fboronat/

ACKNOWLEDGMENT

This work has been financed, partially, by Universitat Politècnica de Valencia (UPV), under its R&D Support Program in PAID-05-11-002-331 Project and in PAID-01-10.

Authors also would like to thank the anonymous reviewers that helped to significantly improve the quality of the paper with their constructive comments.

REFERENCES

[1] F. Boronat, J. Lloret, and M. García, "Multimedia group and inter-stream synchronization techniques: A comparative study", *Inf. Syst.* 34, 1, 108-131, March 2009.

[2] M. Chen, "A low-latency lip-synchronized videoconferencing system, SIGCHI Conference on Human Factors in Computing Systems", *CHI'03*, ACM, 464-471, New York, 2003.

[3] Y. Ishibashi, S. Tasaka, H. Ogawa, "Media Synchronization Quality of Reactive Control Schemes", *IEICE Transactions on Communications*, Vol.E86-B, No.10, 3103-3113, October 2003.

[4] M. O. Van Deventer, H. Stokking, O. A. Niamut, F. A. Walraven, V. B. Klos, "Advanced Interactive Television Service Require Synchronization", *IWSSIP 2008*, Bratislava, June 2008.

[5] F. Boronat, M. Montagud, and J. C. Guerri, "Multimedia group synchronization approach for one-way cluster-to-cluster applications", IEEE 34th Conference on Local Computer Networks, *LCN 2009*, 177-184, Zürich, October 2009.

[6] I. Vaishnavi, P. Cesar, D. Bulterman, O. Friedrich, S. Gunkel, D. Geerts, "From IPTV to synchronous shared experiences challenges in design: Distributed media synchronization", Signal Processing: Image Communication, Vol. 26, Issue 7, pp. 370-377, August 2011.

[7] R. Steinmetz, "Human Perception of Jitter and Media Synchronization", *IEEE Journal On Selected Areas In Communications*, Vol.14, No.1, 61-72, January 1996.

[8] D. Geerts, I.Vaishnavi, R. Mekuria, O. van Deventer, and P. Cesar, "Are we in sync?: synchronization requirements for watching online video together", *CHI '11*, New York (USA), May 2011.

[9] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC-3550, July 2003.

[10] F. Boronat, J. C. Guerri, and J. Lloret, "An RTP/RTCP based approach for multimedia group and inter-stream synchronization", *Multimedia Tools and Applications Journal*, Vol. 40 (2), 285-319, June 2008.

[11] M. Montagud, and F. Boronat, "A new network simulator 2 (NS-2) module based on RTP/RTCP protocols to achieve multimedia group synchronization", *SIMUTOOLS 2010*, Torremolinos, Spain, March 2010.

[12] F. Boronat, M. Montagud, and V.Vidal, "A More Realistic RTP/RTCP-Based Simulation Platform for Video Streaming QoS Evaluation", J. Mobile Multimedia, 7 (1&2), pp. 66-88 (2011).

[13] Y. Su, Y. Yang, M. Lu, H. Chen, "Smooth Control of Adaptive Media Playout for Video Streaming", *IEEE Transactions on Multimedia*, Vol.1, No. 7, 1331-1339, November 2009.

[14] H. Chuang, C. Huang, and T. Chiang, "Content-Aware Adaptive Media Playout Controls for Wireless Video Streaming", *IEEE Transactions on Multimedia*, Vol.9, No. 6, 1273-1283, October 2007.

[15] S. Park, J. Kim, "An adaptive media playout for intra-media synchronization of networked-video applications", *Journal of Visual Communications & Image Representation*, Vol. 19, 106-120, 2008.

[16] N. Laoutaris, and I. Stavrakakis, "Intrastream synchronization for continuous media streams: a survey of playout schedulers", *IEEE Network Magazine*, 16 (3), 30-40, 2002.

[17] Y. Ishibashi, A. Tsuji, and S. Tasaka, "A Group Synchronization Mechanism for Stored Media in Multicast Communications", *Proc. of the INFOCOM '97*, Washington April 1997.

[18] Y. Ishibashi, and S. Tasaka, "A distributed control scheme for group synchronization in multicast communications", *Proc. of International Symposium Communications*, Kaohsiung (Taiwan), pp. 317-323, November 1999.

[19] M. Mauve, J. Vogel, V. Hilt, and W. Effelsberg, "Local-Lag and Timewarp: Providing Consistency for Replicated Continuous Applications", *IEEE Transactions on Multimedia*, Vol.6, No.1, February 2004.

[20] C. Hesselman, D. Abbadessa, W. Van Der Beek, et alter, "Sharing enriched multimedia experiences across heterogeneous network infrastructures", IEEE Communications Magazine, Vol.48, no.6, pp.54-65, June 2010.

[21] I. Vaishnavi, et al., "From IPTV to synchronous shared experiences challenges in design: Distributed media synchronization", Signal Process. Image Commun. (To be published in 2011).

[22] Y. Ishibashi, and S. Tasaka, "A group synchronization mechanism for live media in multicast communications", *IEEE GLOBECOM'97*, pp. 746–752, November 1997.

[23] H.M. Stokking, M.O. van Deventer, O.A. Niamut, F.A. Walraven, and R.N. Mekuria, "IPTV inter-destination synchronization: A network-based approach", *ICIN'2010* , Berlin, October 2010.

[24] Y. Ishibashi, and S. Tasaka, "A distributed control scheme for causality and media synchronization in networked multimedia games", Proc. 11th International Conference on Computer Communications and Networks, Miami (USA), pp. 144-149, October 2002.

[25] T. Hashimoto, Y. Ishibashi, "Group Synchronization Control over Haptic Media in a Networked Real-Time Game with Collaborative Work", *Netgames'06*, Singapore, October 2006.

[26] Y. Kurokawa, Y. Ishibashi, and T. Asano, "Group synchronization control in a remote haptic drawing system", *Proc. of IEEE International Conference on Multimedia and Expo*, Beijing (China), pp. 572-575, July 2007.

[27] Y. Ishibashi, K. Tomaru, S. Tasaka, and K. Inazumi, "Group synchronization in networked virtual environments", Proc. Of the 38th IEEE International Conference on Communications, Alaska (USA), pp. 885–890, May 2003.

[28] T. Nunome, and S. Tasaka, "Inter-destination synchronization quality in a multicast mobile ad hoc network", *Proc. of IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications*, Berlin (Germany), pp. 1366-1370, September 2005.

[29] J. Ridoux, and R.Veitch, "Principles of Robust Timing over the Internet", *Com. of the ACM*, Vol. 53 (No. 5), May 2010.

[30] P. V. Rangan, S. Ramanathan, and S. Sampathkumar, "Feedback techniques for continuity and synchronization in multimedia information retrieval", *ACM Trans. Inf. Syst*, 13, 2, 145-176, April 1995.

[31] J. C. Guerri, M. Esteve, C. E. Palau, and V. Casares, "Feedback Flow Control with Hysteresial Techniques for Multimedia Retrievals", *Multimedia Tools and Applications*, 13, 3, pp. 307-332, March 2001.

[32] A. Begen, C. Perkins, and J. Ött, "On the use of RTP for Monitoring and Fault Isolation in IPTV", *IEEE Network*, Volume 24 Issue 2, March/April 2010.

[33] J. Ott, and C. Perkins, "Guidelines on Extending the RTP Control Protocol (RTCP)", RFC 5968, September 2010.

[34] F. Ferrari, A. Meier, and L. Thiele, "Accurate Clock Models for Simulating Wireless Sensor Networks", *SIMUTools 2010*, Torremolinos (Spain), March 2010.

SHORT BIOGRAPHIES

Mario Montagud was born in Gandia (Spain). He studied Telecommunications Engineering at Polytechnics University of Valencia (UPV). Since then, he is researching about Multimedia Systems, Multimedia Synchronization Protocols and Simulation Techniques with a research scholarship to obtain his phD degree. He is also involved in several local and regional research projects. His topics of interest include Network Protocols, QoS, Multimedia Systems, Multimedia Synchronization Protocols and Simulation Techniques. He is IEEE member since 2009, is involved in several IPCs of international conferences and is Assistant Editor of two international Journals (Network Protocols and Algorithms -NPA-, and Advances in Networks and Communications -ANC-).

Fernando Boronat, was born in Gandia, (SPAIN) and went to the Polytechnic University of Valencia (UPV) in Spain, where he studied Telecommunications Engineering. After doing his military Service, in 1994, he worked for a couple of years for Telecommunication Companies before moving back to the UPV in 1996 where he is Assistant Professor in the Communications Department at the Gandia Campus in the UPV. He obtained his phD degree in 2004 and his topics of interest are Communication networks, Multimedia systems and Multimedia Synchronization

Protocols. He is IEEE member since 1993 and is involved in several IPCs of national and international conferences.