Design, Development and Assessment of Control Schemes for IDMS in an Evolved and Standardized RTCP-based Solution

Mario Montagud[1], Fernando Boronat[1], Hans Stokking[2], Pablo Cesar[3],
[1]Universitat Politècnica de València (UPV) - IGIC Institute
Grao de Gandia, Valencia (Spain)
mamontor@posgrado.upv.es; fboronat@dcom.upv.es
[2]TNO: Netherlands Organisation for Applied Scientific Research TNO
Brassersplein 2, Delft, the Netherlands
hans.stokking@tno.nl
[3]CWI: Centrum Wiskunde & Informatica
Science Park 123, Amsterdam 1098 XG, Netherlands
p.s.cesar@cwi.nl

Abstract — Currently, several media sharing applications that allow social interactions between distributed groups of users are gaining momentum. This paper focuses on a key factor for providing a satisfying feeling of togetherness in those networked scenarios, which is to ensure synchronous playout between distributed users. This research problem is known as Inter-Destination Media Synchronization (IDMS). Particularly, this paper discusses the benefits of extending RTP/RTCP protocols for IDMS purposes. Accordingly, newly defined RTCP messages and control techniques have been added to a previous version of a centralized RTP/RTCP-based IDMS solution to enable an adaptive, highly accurate and standardized solution. Moreover, as different control schemes for IDMS exist, and each one is best suited for specific use cases, our IDMS solution has been extended to be able to adopt each one of them. Simulation results prove the satisfactory responsiveness of our IDMS solution, as well as its consistent behavior when using each one of the deployed control schemes.

***Keywords* — Distributed Media Consumption; Inter-Destination Media Synchronization; RTP/RTCP; Simulation**

List of Abbreviations:

AMP	Adaptive Media Playout
APP	(RTCP) Application-Defined Packet
CBR	Constant Bit Rate
CoD	Content on Demand
DCS	Distributed Control Scheme
ETSI	European Telecommunications Standards Institute
GoP	Group of Picture
GPS	Global Positioning System
IANA	Internet Assigned Numbers Authority
IDMS	Inter-Destination Multimedia Synchronization
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPTV	Internet Protocol Television
FTP	File Transfer Protocol
M/S	Master/Slave
MMOG	Massive Multiplayer Online Games
MU	Media Unit
NTP	Network Time Protocol
PTP	Precise Time Protocol
RFC	Request For Comments
QoE	Quality of Experience
QoS	Quality of Service
RR	(RTCP) Receiver Report
RTP	Real-Time Transport Protocol
RTCP	RTP Control Protocol
RTSP	Real Time Streaming Protocol
RTT	Round Trip Time
SDES	(RTCP) Source Description
SIP	Session Initiation Protocol
SMS	Synchronization Maestro Scheme
SR	(RTCP) Sender Report
SSM	Source Specific Multicast
TISPAN	Telecoms & Internet Converged Services & Protocols for Advanced Networking
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VBR	Variable Bit Rate
VoD	Video on Demand
VoIP	Voice over IP
XR	(RTCP) Extended Report

1. Introduction.

Recent advances on media streaming and social networking, as well as the proliferation of heterogeneous connected devices, open a new media landscape. Shared media experiences are gaining momentum, allowing geographically distributed users to concurrently consume the same media content while interacting with each other (e.g., using text, audio or video chat).

This emerging media consumption paradigm faces several challenges [1]. Particularly, this paper concentrates on a key aspect, which is the provisioning of synchronous playout between distributed users. This research problem is known as Inter-Destination Media Synchronization (IDMS), and is becoming relevant in a large number of use cases [2]. However, as current distribution networks cause delay differences, additional adaptive solutions for equalizing them are required [2, 3].

In [3], the authors presented the design, implementation and evaluation of a centralized IDMS solution, which is based on simple extensions to the RTP/RTCP standard protocols [4]. Experimental tests proved its satisfactory performance in both real, but controlled, scenarios [5] and simulated ones [3]. However, with the advent of distributed media consumption, further requirements emerge. First, IDMS needs to be provided in open large-scale scenarios, while guaranteeing interoperability when different types of (third-party) consumer devices are involved. Second, some IDMS use cases require very strict synchronization levels [2], so highly accurate IDMS solutions should be provided. Furthermore, our study in [2] pointed out that a centralized approach is not always the best choice for IDMS, as in some cases distributed solutions are more suitable.

The goal of this work is to fit the above requirements. Accordingly, the contributions of this paper are four-fold. First, the rationale for using and extending RTP/RTCP for IDMS purposes is discussed. New standard compliant RTCP extensions[1] for IDMS have been designed to be able to guarantee interoperability in large-scale deployments, as well as compatibility between third-party implementations and infrastructure. As the second contribution, these RTCP extensions, in combination with further designed control algorithms and techniques, have been implemented in our IDMS solution to enable the achievement of stringent synchronization levels. Third, our RTP/RTCP-based IDMS solution has been extended to be able to adopt different architectural schemes. This allows our IDMS solution to be adaptive and flexible enough to be efficiently deployed in different network environments. Fourth, the consistent behavior and satisfactory responsiveness of the evolved version of our RTP/RTCP-based IDMS solution is proved, for each one of the adopted schemes, through extensive simulation studies.

1.1. *IDMS Nomenclature and Classification.*

Different entities can be distinguished in the IDMS control process:

[1] Both ETSI [6] and IETF [7] standardization bodies have adopted our RTP/RTCP-based technology for IDMS.

1. *The Media Server(s)*: The sender(s) of the media stream(s).
2. *The Sync Clients*: The sync entities that render the media content in a synchronous way. They must also send informative control reports, including their current timing information, to allow for an overall IDMS control.
3. *Master*: A specific Sync Client whose timing information is selected as the IDMS reference.
4. *Sync Manager*: The sync entity responsible for collecting the informative reports from the Sync Clients. Then, it calculates the timing discrepancies between the Sync Clients and, if needed, sends back to them new control messages to notify of the required adjustments to achieve IDMS.

Additionally, the concepts of "solution", "schemes", "algorithms" and "techniques" are repeatedly used in the IDMS context. To help to understand this paper and its contributions, the relationships between these concepts are briefly explained. First, an IDMS solution can adopt different architectural schemes to exchange synchronization information. This will determine the involved sync entities and their communication processes to achieve IDMS. Second, different algorithms can be implemented in an IDMS solution to carry out the synchronization control. Third, an IDMS solution can make use of several (server- or client-based) control/adjustment techniques [8] to maintain and/or restore the media synchronicity.

Regarding the IDMS schemes, three main models can be distinguished (see Fig. 1): two centralized schemes (Master/Slave, or M/S, Scheme and Synchronization Maestro Scheme or SMS) and one distributed scheme (Distributed Control Scheme or DCS). Readers are referred to [2] to find an exhaustive description, classification and a qualitative comparison between these IDMS schemes.

Furthermore, independently of the control scheme in use, two main approaches can be differentiated based on the location of the sync entities: network-based and terminal-based. In the first case, the sync entities are deployed at the network side, and the IDMS control processes are managed by network entities, under control of the service provider or the operator. This way, the end users' terminals do not have to implement any IDMS functionality. The other approach consists of deploying the IDMS functionality at the end-users' terminals. By using this approach, the Sync Clients are located in the end-users' terminals, and the Sync Manager functionality can be deployed as a separate independent entity (see Fig. 1b1), or either as part of the Media Server (see Fig. 1b2) or of a Sync Client (see Fig. 1b3).

Network-based IDMS solutions have been neither deployed nor tested in any real scenarios yet. Only a design approach was proposed in [9] to meet the need for IDMS in advanced and large-scale IPTV services. So, this paper mainly focuses on terminal-based IDMS solutions, which have been more extensively adopted up to date.

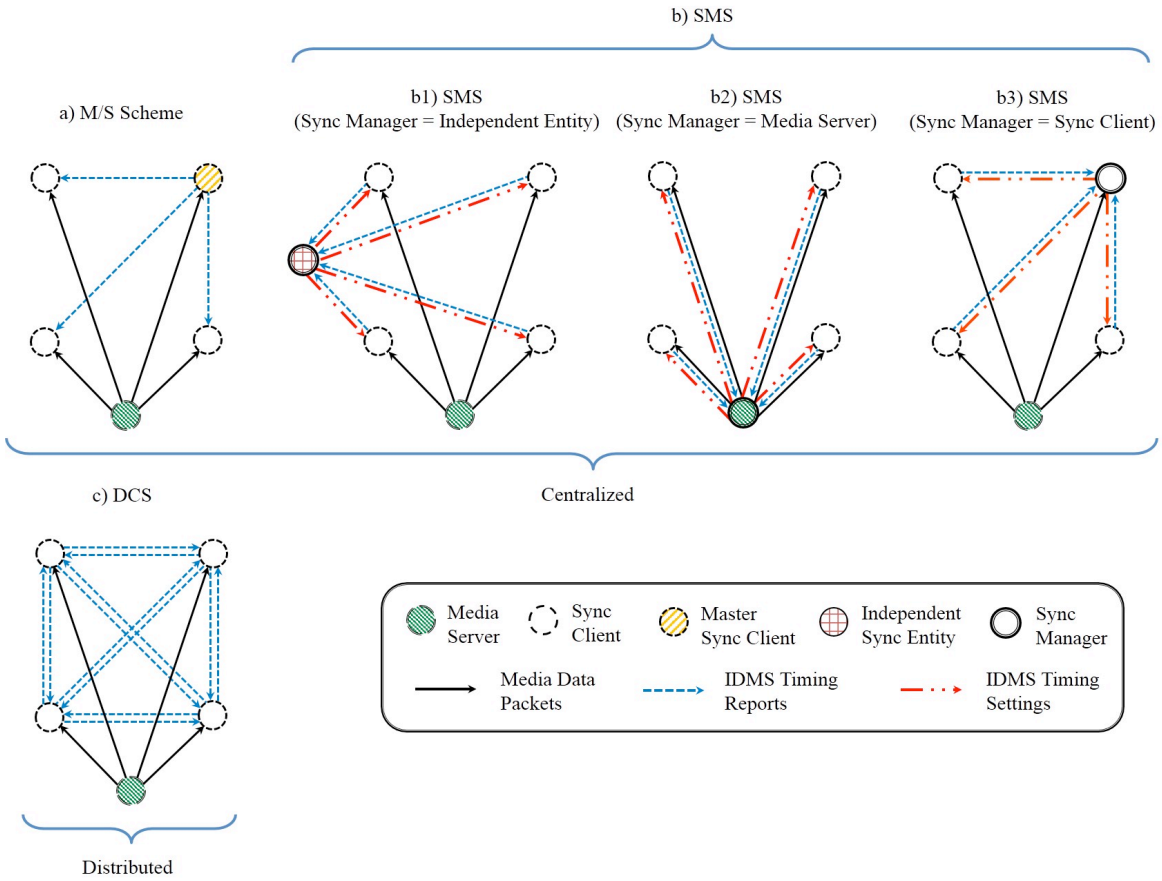

Fig. 1. Control Schemes for IDMS.

A hybrid architecture can also be used. Herein, the Sync Clients can be divided in sub-level domains, each one with a local Sync Manager. On a multi-domain level perspective, the local Sync Managers can further communicate with a higher hierarchic Sync Manager that is responsible of controlling the IDMS processes of the sub-level domains. A similar architecture is utilized for managing the consistency in some Massive Multiplayer Online Games (MMOG) (e.g., [10, 11]). However, even though considering its relevancy, this hybrid design is out of scope of this work. The appropriateness of such an architecture for IDMS will be analyzed in future works.

To conclude this sub-section, Table 1 provides a list of all the symbols used in this paper (mostly in Section 4) and their meaning. This also aids in understanding the paper.

1.2. *Structure of the Paper.*

The structure of the paper is as follows. In the next section, the state-of-the-art regarding IDMS is reviewed. Likewise, the suitability of each IDMS scheme for specific requirements is briefly analyzed, based on our conclusions in [2]. In Section 3, the rationale for choosing RTP/RTCP to provide IDMS is discussed. Then, Section 4 describes the design and operation of our evolved IDMS solution, for each control scheme, as well as the newer aspects and functionalities that have been added to improve its performance. Section 5 gives performance results, and, finally, Section 6 provides our conclusions and a discussion of future work.

Table 1 Nomenclature Used in this Work

Symbol	Units	Meaning
θ	MU/s	Media Server Nominal Rate
γ^k_i	%	Playout Rate Skew (Deviation) of the *i-th* Sync Client belonging to the *k-th* Group
$\omega^k_i(t)$	%	Playout Rate Drift of the *i-th* Sync Client belonging to the *k-th* Group
ε	%	Maximum Playout Rate Drift (Fluctuation)
$\mu^k_i(t)$	MU/s	Playout (Service) Rate of the *i-th* Sync Client belonging to the *k-th* Group
t_n	Time Unit	Transmission time of the *n-th* MU
$r^k_{n,i}$	Time Unit	Reception time of the *n-th* MU in the *i-th* Sync Client belonging to the *k-th* Group
$d^k_{n,i}$	ms	Playout delay of the *n-th* MU in the *i-th* Sync Client belonging to the *k-th* Group
$p^k_{n,i}$	Time Unit	Playout time of the *n-th* MU in the *i-th* Sync Client belonging to the *k-th* Group
τ^k_{max}	ms	Maximum allowable asynchrony threshold in the *k-th* Group
$\Delta^k_{n,i}$	ms	Detected asynchrony for the *n-th* MU in the *i-th* Sync Client belonging to the *k-th* Group
$\delta^k_{n,i}$	ms	Distributed asynchrony for the *n-th* MU in the *i-th* Sync Client belonging to the *k-th* Group
$\varphi^k_{n,i}$	%	Playout Factor for the *n-th* MU in the *i-th* Sync Client belonging to the *k-th* Group
T_{RTCP}	ms	RTCP Report Interval

2. Related Work

Due to the increased relevancy of IDMS, several works are addressing this topic. Some illustrative papers discuss a number of use cases [2], propose qualitative comparisons between synchronization solutions [8], and report on working prototype implementations (e.g., [1], [5], [12], etc.).

On one hand, most of the existing IDMS solutions (compiled in [2, 8]) were designed for specific use cases, such as audio/video streaming (e.g., [1], [5], [13], [14]), conferencing (e.g., [15], [16], [17], [18]), gaming (e.g., [10], [12], [17], [20], [21], [22]), collaborative virtual environments (e.g., [11], [12]) or synchronous e-learning (e.g., [15], [17]), involving different types of streams, such as video, audio, haptic media (e.g., [16], [17]), or users' events (e.g., [1], [20], [21]). Likewise, most of them define new (application-specific) proprietary protocols that may increase the network load and make compatibility between implementations more difficult.

Even though a specific IDMS solution may be highly dependent on the context and space for which it is going to be deployed, our intention was to design an inter-operable (i.e., using standardized components) and wider-applicable (i.e., easy to deploy and valid for different use cases) IDMS solution, by using and extending RTP/RTCP protocols. This is discussed and described in Sections 3 and 4, respectively.

On the other hand, different solutions have adopted different architectural approaches (more details can be found in [2, 8]), as can be appreciated in the classification in Table 2.

An exhaustive qualitative comparison among IDMS schemes was provided in [2]. First, that study pointed out that SMS is the best scheme in terms of consistency, coherence, and security. Likewise, this scheme can provide satisfactory responsiveness in terms of flexibility, traffic overhead, causality, and fairness. But, on the contrary, scalability and interactivity were identified as the main weaknesses of using SMS. Despite of the interactivity limitation, several works have

made use of SMS for IDMS (see Table 2) and have shown its feasibility for keeping the asynchrony within allowable limits [2, 8]. Accordingly, SMS is the most appropriate option for IDMS in those scenarios in which coherence is essential (i.e., all the Sync Clients must be synchronized almost simultaneously), the network delay is not excessively large, and the number of participants is not too high.

Second, the same study pointed out that M/S Scheme can provide the best performance in terms of scalability, traffic overhead, interactivity and causality, but presents serious drawbacks in terms of robustness, coherence, flexibility and fairness. Therefore, this scheme can be a suitable option in those scenarios in which the bandwidth available is scarce[2] (since only the master Sync Client sends IDMS reports), and also in those use cases in which a single participant (the master) has a certain priority level over the other participants (e.g., the teacher in synchronous e-learning scenarios).

Third, that study identified the appropriateness of DCS for IDMS for those use cases in which there is a need for high-performance in terms of robustness, fairness, flexibility, scalability and interactivity. Contrarily, DCS performs worse than the centralized schemes in terms of traffic overhead (because all the Sync Clients send IDMS reports in a multicast way), consistency and security. Therefore, DCS can be a suitable solution for controlled environments in which bandwidth availability is not a problem (see footnote 2), and security aspects can be ensured.

A noteworthy limiting factor for DCS and M/S schemes is the necessary support of multicast feedback capabilities in the distributed Sync Clients. In some media streaming technologies, e.g. those in which Source Specific Multicast (SSM) with unicast feedback ([23]) is employed, this is not possible. In such cases, only the Distribution Source (which could be co-located with the Media Server or be an independent entity) can transmit data in a multicast way. So, it could prevent the deployment of an IDMS solution based on DCS or M/S schemes in large-scale environments, such as IPTV broadcast distribution channels. In other controlled scenarios, where small groups of users are consuming media content synchronously, independently of other users or groups of users, the adoption of DCS or M/S schemes would be a feasible option. This multicast limitation is not an issue when using SMS, because the feedback reports are sent by the Sync Clients in a unicast way to the Sync Manager.

Table 2 Classification of IDMS Solutions based on the Adopted Control Scheme

	CENTRALIZED		**DISTRIBUTED**
	M/S Scheme	**SMS**	**DCS**
Terminal-Based	[1], [13], [14]	[15], [12], [16], [17], [5], [3]	[19], [20], [21], [22], [18], [1]
Network-Based	-	[9]	-

[2] However, the traffic overhead added by the IDMS control messages will not be very high compared to the bandwidth consumption by the media stream to be synchronized.

Summarizing, it was concluded in [2] that SMS is, in general, the best-ranked scheme for IDMS. However, DCS and M/S schemes were also identified as suitable options for specific IDMS use cases. On one hand, M/S Scheme outperforms SMS in terms of scalability, traffic overhead, interactivity and causality. On the other hand, DCS can provide better performance than SMS in terms of robustness, scalability, interactivity, flexibility and fairness. That is why we decided on the deployment of DCS and M/S schemes in the evolved version of our IDMS solution, which was previously only based on the use of SMS [3]. The consistent behavior and effectiveness of the three control schemes for IDMS will be proved in Section 5.

3. Candidate Protocols for IDMS: Rationale for Choosing RTP/RTCP

Nowadays, RTP/RTCP standard protocols [4] are extensively used in interactive streaming services such as VoD/CoD (Video/Content on Demand), VoIP (Voice over IP), videoconferencing and IPTV (Internet Protocol Television). All these services would be benefited by the provisioning of IDMS [2], an IDMS solution based on the RTP/RTCP capabilities would ease and promote deployment, as well as ensure interoperability.

These protocols provide metadata useful for media synchronization. On one hand, the timestamps, sequence numbers, and Payload Type (PT) identification mechanisms provided by RTP packets enable intra-stream synchronization. On the other hand, the mapping time information and source identification parameters provided by each RTCP Sender Report (SR) and Source Description (SDES) reports, respectively, are helpful to achieve inter-stream synchronization [4].

Further modifications and/or additions to RTP/RTCP are allowed to include profile-specific information required by particular purposes, and the guidelines for doing so are specified in RFC 5968 [24]. Likewise, RFC 3611 [25] allows the definition of new RTCP Extended Report (XR) blocks for exchanging additional QoS metrics regarding media transmission or reception.

Some of the existing IDMS solutions do not provide an accurate or a continuous synchronization control [8]. On one hand, some of them are only based on compensating the variability of network delays between distributed clients, but do not take into account the variable delays that are originated at the end-system side (e.g., due to buffer settings, de-packetizing, decoding, reassembling, scaling, rendering, etc.) [2]. This cannot provide high synchronization accuracy because the end-to-end delay variability must be compensated for. On the other hand, some other solutions uniquely rely on synchronizing specific control events (e.g., stream content or position updates), but such solutions can neither provide high synchronization accuracy because of the delay variability during a media session's lifetime.

The above limitations can be overcome by extending the RTCP control channel. New RTCP feedback messages can be defined to include timing information about reception and/or

[3] At the client side, the original sampling rate, i.e. the rate of advancement of RTP timestamps, can be determined by the Payload Type (PT) field included in RTP packets [4].

presentation times for specific RTP packets, which can transport a fragment of one, one, or several, application-layer Media Units or MUs (e.g., video frames or audio samples). First, the inclusion of presentation times allows synchronizing media streams at the packet level but, at the same point, above the transport level, thus enabling an end-to-end synchronization (high accuracy). Second, the regular exchange of RTCP messages for IDMS, according to the dynamic and bounded RTCP timing rules [4], allows for adaptive and continuous IDMS monitoring and adjustment processes.

The appropriateness of alternative protocols for IDMS, such as SIP (Session Initiation Protocol) [26], RTSP (Real Time Streaming Protocol) [27], Diameter [28], and H.248 [29], was also assessed. SIP and RTSP could be extended with synchronization parameters, but those protocols are not supported by network elements transporting the actual media streams, only by user's terminals. This fact limits a future implementation of a network-based IDMS approach [9]. As well, RTSP is commonly used for CoD services, but not for live multicast services. Only an RTSP-based IDMS solution is clearly not sufficient. Contrarily, the downside of extending Diameter and H.248 is that these protocols are not commonly supported by user's terminals, being only valid for in-network synchronization [9]. The above limitations are overcome by using RTP/RTCP.

4. Development of a Interoperable, Accurate and Flexible IDMS Solution

The operation of the evolved and extended version of our RTCP-based IDMS solution, as well as the novel aspects that have been added to it, are described in this section.

4.1. *Evolved and Standardized RTCP-based IDMS Solution*

A preliminary version of our IDMS solution extended the RTCP Receiver Reports (RRs) in order to include the playout point of each Sync Client (sequence number of the MU being played out and its playout time) [5]. Then, an additional field was added to allow for an independent IDMS management for different groups of Sync Clients [3]. Also, new RTCP Application-Defined (APP) packets were defined in order to estimate network delays and to exchange playout settings instructions. Their format can be found in [3].

The satisfactory performance of our RTCP-based IDMS solution was tested, both objectively and subjectively, within our University infrastructure [5]. Besides, an enhanced version of our IDMS solution was implemented and evaluated in a simulation framework [3]. The proposed RTCP extensions in these works were targeted to achieve IDMS for our own-designed or vendor-specific applications (e.g., video sharing or surveillance), while: i) adding a minimal traffic overhead; and ii) operating within controlled scenarios. Both in our University and in the simulation framework we have a minimal control over the networking and communication devices.

Nowadays, there is an increasing demand of IDMS services in a large set of use cases [2]. The operation of these use cases does not have to be limited to controlled scenarios, but they are often deployed over large-scale environments (e.g., IPTV networks), using a variety of networking and

consumer devices. Therefore, interoperability between third-party implementations and infrastructure is needed. Besides, some of these use cases require very strict synchronization levels [2], so highly accurate IDMS technology should be provided.

In addition, further extensions to RTCP have been recently, and are still being, proposed in different contexts (e.g., for monitoring, synchronization, retransmission or adjustment techniques). These extensions may significantly increase the complexity of the protocol, probably leading to incompatibility issues.

Hence, given the suitability of these protocols to provide IDMS and the above emerging requirements for enabling interoperability, high accuracy and protocol maintenance, a need for an evolved RTP/RTCP-based IDMS solution arises. This is the goal of our IDMS standardization efforts [6, 7], derived from the initial RTCP route in [5].

4.2. *Definition of new RTCP messages for IDMS*

RFC 3550 [4] allows the definition of "profile-specific" extensions to RTCP RR for *"giving additional feedback information"*. Accordingly, it seems a suitable option for including the playout point of each Sync Client [5, 3]. On one hand, these "profile-specific" extensions introduce lower overhead than the definition of a new RTCP report or packet. On the other hand, these "profile-specific" extensions can lead to complexity and compatibility problems. First, we cannot obviate that several RTCP extensions have been, are being, and will be proposed in different contexts. If they were included as extensions of the current RTCP reports, this may lead to rather complex and large packets. Also, if several "profile-specific" extensions are appended to existing packets, the efficiency of the feedback channel will decrease, because not all the metrics defined in these different extensions need to be reported in all applications. Second, these kinds of extensions are difficult to signalize and may not be backwards compatible with other RTP profiles. Hence, these kinds of extensions would break the operation and cause inconsistences in most third-party RTP end-systems and middle boxes, lowering the chances of successful deployment.

A similar problem arises when considering the inclusion of IDMS setting instructions in extended RTCP SRs. Here it seems more reasonable to define Application-Defined or APP packets [4], as in [5, 3]. However, as pointed out in [24], APP packets should be used for private or vendor-specific extensions that do not need to interoperate with others, or for experimental purposes, but they are inappropriate for interoperable and standardized solutions.

Common guidelines must be followed to avoid an extension creep that can only harm interoperability and future evolution of the RTCP protocol at large. Where compatibility and protocol maintenance is pursued, the guidelines in RFC 5968 [24] must be followed for choosing the most suited RTCP extension points for IDMS.

Accordingly, a newly-defined RTCP XR block for IDMS, called IDMS report, has been specified to enable Sync Clients to report on packet reception and/or presentation times for specific

RTP packets. Concretely, this IDMS report is composed of the default XR headers [25], followed by useful metadata for IDMS [7]: i) the PT of the media stream this block reports on; ii) the SSRC of the source of the media stream this block reports on; iii) the Synchronization Group Identifier to which the sender of this report belongs; iv) the RTP timestamp belonging to the RTP packet to which the report refers; v) the packet reception time (64 bits); and, optionally, vi) the packet presentation time (32-bit central word).

But besides using RTCP for monitoring purposes, control of the timing of the Sync Clients is also needed for IDMS. Therefore, a new RTCP packet type for IDMS [7], called IDMS Settings packet, has been defined to allow the Sync Manager to provide a common target playout point to which all the Sync Clients belonging to a specific synchronization group must match.

Additionally, in order to signalize the use of the above RTCP messages for IDMS and to manage the group membership (i.e., to allow the co-existence of several independent groups of Sync Clients in an IDMS-enabled session), a new Session Description Protocol (SDP) parameter, called *rtcp-idms*, has been specified [7]. The use of this SDP attribute for groups establishment and management discussed in [7].

But apart from avoiding interoperability issues, the metadata info included in the newly defined RTCP reports and packets for IDMS enables better flexibility, scalability and accuracy than using the RTCP messages previously proposed in [5, 3]. This is discussed below:

i) Flexibility and Accuracy: The IDMS report allows reporting on both packet reception and presentation times. RTP packet arrival times are more accessible to Sync Clients and therefore relatively easier to report on. If the capabilities of the communication devices are in some way homogeneous (i.e., the buffer settings, decoding, processing and rendering delays will take roughly the same amount of time in all the Sync Clients), an acceptable accuracy can be achieved when reporting only on reception times. But if all the Sync Clients have the ability to report on, and hence synchronize on, actual presentation times, this will enable highly accurate end-to-end synchronization.

However, the use of packet presentation times requires the Sync Clients to track RTP packets to their ultimate presentation time (at the application layer). This can be seen as a form of layer-violation in some media players because the RTP plane (transport layer) could not have access to the application layer, thus implying more difficult implementation requirements. Because of the previous discussion, reporting on packet arrival times is mandatory, but reporting on presentation times is optional in our evolved version of our IDMS solution [7]. In our simulation-based prototype, the second option is supported by all the involved sync entities. Accordingly, syncing on RTP “arrival” times has not been considered in this paper.

ii) Accuracy: The RTCP IDMS settings packets include a 64-bit presentation timestamp for indicating the target presentation times, instead of the 32-bit parameter in the IDMS report. This

allows a higher level of granularity for those use cases presenting stringent synchronization requirements, such as audio beamforming or networked video walls [2].

Besides, when including the RTP sequence number to identify the last received/presented MU in each Sync Client, as in [5, 3], it is difficult to infer the asynchrony between them when Variable Bit Rate (VBR) coding mechanisms are used. This is because the rate of advancement of the sequence numbers can depend on the temporal and spatial compression methods employed in those coding mechanisms, as well as on its configured parameters (e.g., Group of Pictures or GOP size and pattern, quantizer scale, etc.). Therefore, a constant advance rate of the sequence numbers cannot be assumed, unlike when using Constant Bit Rate (CBR) traffic patterns.

Even though a VBR coding mechanism is employed, if a constant frame rate is assumed (i.e., the Media Server transmits MUs with a constant rate, e.g. θ=25 MU/s), a solution for that is to use the RTP generation timestamp to identify specific RTP packets in the IDMS messages. This also helps to solve fragmentation issues when application-layer MUs (e.g., video frames) are carried in several RTP packets, since all of them will have the same timestamp.

iii) Flexibility and Scalability: Despite that an 8-bit field for allocating the group identifier could suffice [5, 3], since it enables 2^8=256 different groups of Sync Clients, the use of a 32-bit field in this evolved version minimizes the probability of collision (i.e., two groups choose the same identifier). Moreover, the process of such an identifier being randomly generated with 32 bits is well-known from numerous telecommunication protocols and systems [4].

These newly defined RTCP messages for IDMS have been adopted in the evolved version of our IDMS solution, being also implemented in our simulation framework. Their exchange process in each one of the deployed control schemes, as well as the operation of the designed IDMS algorithms and techniques, according to the new metadata included in these messages, are described in next sub-sections.

4.3. *Negotiation and Use of Common Wall-clock Sources*

Most IDMS solutions ([2, 8]) do require wall-clock synchronization mechanisms between the involved sync entities. This is necessary for: i) coherently stamping and interpreting timelines along the end-to-end media distribution chain; ii) correlating the timing information in the IDMS reports; and iii) calculating one-way delays.

Commonly, RTP/RTCP use NTP-based timestamps to facilitate media stream synchronization and to provide estimates of delay and other statistical parameters. If the involved sync entities employ different un-coordinated wall-clock references, it is rather difficult to achieve tight synchronization between different Sync Clients. In some cases, the use of NTP for clock synchronization cannot meet the stringent synchronization levels required in some IDMS use cases [2]. In some other cases, it could be possible that not all the involved sync entities support NTP, but do support other technologies. Therefore, the use of other (more accurate) clock synchronization

mechanisms, such as GPS (Global Positioning System) or PTP (Precise Time Protocol), must also be enabled for IDMS. To help the sync entities sort out these timing issues, an SDP attribute called *clocksource*, specified in [30], can be used. This attribute allows the sync entities to declare if they support clock synchronization, which sources they support, which source is currently used, the last time the Sync Client synchronized with that source, and the synchronization frequency. This SDP parameter can be used as an indication of clock sync accuracy and allows the involved sync entities in an IDMS-enabled session to negotiate the use of a common (or related) wall-clock source.

4.4. *Exchange of IDMS Control Messages between Sync Entities.*

Typically, during an RTP session, after the Media Server starts sending RTP data packets (encapsulating MUs), each distributed *i-th* Sync Client regularly sends feedback RTCP RR to inform about QoS (e.g., network delay, RTT, jitter or losses) [4]. Additionally, each *i-th* Sync Client must include an IDMS report in each compound RTCP packet it sends, including its local playout point: i) the original RTP timestamp of the MUs being played at that moment (t'_i); ii) its reception time (r_i); iii) optionally, its presentation time (p_i); and iv) the *k-th* group identifier to which the Sync Client belongs.

Depending on the employed IDMS scheme, the IDMS reports will be sent by all the Sync Clients (unicast in SMS or multicast in DCS) or only by the master Sync Client (multicast in the M/S Scheme). Accordingly, these IDMS reports will only be received by the Sync Manager (in SMS), or by all the Sync Clients (in DCS and M/S schemes). When using DCS, if the involved Sync Clients belong to different groups, each Sync Client must only register the information of the incoming IDMS reports from all the other Sync Clients belonging to the same group/s, despite it may receive IDMS reports from all the Sync Clients in the multicast session.

This way, once the overall playout information in each *k-th* group has been collected, the playout time discrepancy (asynchrony) between the Sync Clients belonging to that *k-th* group can be computed. If SMS or DCS is employed, the Sync Manager or each Sync Client, respectively, must compare the local playout delays of each *i-th* Sync Client belonging to that *k-th* group, d_i^k. It can be calculated as the time difference between the presentation time of the current MU being played out by that Sync Client (i.e., the one reported in the IDMS report) and the RTP generation timestamp[4] of that MU (more accurately of one of the RTP packets encapsulating that MU):

$$d_i^k = (p_i^k - t'^k_i) \tag{1}$$

The maximum asynchrony (Δ^k_{max}) in each *k-th* group will be given by the difference between the most lagged and the most advanced playout points of all the active Sync Clients in that group (G_k):

$$d^k_{\max/\min} = \max/\min\{d_i^k, \forall i \in G_k\} \tag{2}$$

[4] For that purpose, the generation RTP Timestamp (32-bits) must be mapped to its associated NTP-based wall clock timestamp (64-bits): $t'_i \rightarrow t_i$. If the sync entity calculating the playout delay is not the Media Server, it can easily be done by using the mapping time info included in the RTCP SR packets sent by the Media Server.

$$\Delta^{k}_{\max} = (d^{k}_{\max} - d^{k}_{\min}) \quad (3)$$

If the M/S Scheme is employed, each slave Sync Client must calculate the asynchrony with the master every time an IDMS report from it is received.

4.5. *Master Reference Selection Policies*

Every time the detected asynchrony in each *k-th* group exceeds an allowed threshold τ_{max} (i.e., $\Delta^{k}_{max} \geq \tau_{max}$), reactive techniques must be triggered to restore the synchronicity. Accordingly, the first decision consists of the selection of the IDMS master reference to synchronize with.

Using M/S scheme this is implicit, since the temporal reference for IDMS is dictated by the timing information included in the IDMS reports from the master Sync Client.

Using SMS, the Sync Manager can employ several dynamic policies to select the master reference for IDMS [3]. Possible strategies include: synchronization to the slowest Sync Client; synchronization to the fastest Sync Client; synchronization to the mean playout point; and synchronization to the nominal rate of the Media Server. Additionally, some variations to the above policies could be employed to account for variable network conditions (e.g., adding some temporal margin to smooth out extreme jitter/congestion situations).

Using DCS, each Sync Client must locally, and independently of the other Sync Clients, decide the reference timing to synchronize with by using the above first three master selection policies.

In the evolved version of our IDMS solution, the processes of comparing the IDMS timing of the distributed Sync Clients belonging to each *k-th* group (Eqs. 1, 2 and 3), and determining the reference IDMS timing to synchronize with (i.e., the processes of obtaining d^{k}_{IDMS}), are more accurate and simpler compared with the previous version of our IDMS solution in [3], because no translation from RTP sequence numbers to timestamps is needed.

The first strategy consists of selecting the playout timing of the most lagged (slowest) Sync Client in each *k-th* group as the IDMS reference, which is the one with the largest playout delay (i.e., $d^{k}_{IDMS} = d^{k}_{master} = d^{k}_{max}$). Using this method there will not be any skips in the Sync Clients' playout processes, avoiding the subsequent discontinuities, since (faster) slave Sync Clients will be forced to pause their playout processes (waiting for the slowest one). This policy is 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 compete with each other (e.g., networked quiz shows), this policy is appropriate to guarantee fairness between them. However, if the playout process of the master Sync Client is extremely lagged (e.g., due to any problem, such as network congestion or end-system overload), the use of this policy could result in the progressive filling of the playout buffers of all Sync Clients, which could eventually overflow. This way, loss of real time sensation would be noticed, affecting the overall QoE. To avoid such situations, additional adjustment techniques, such as buffer fullness monitoring and control, should be deployed (left for further study).

In the second method, the playout timing of the most advanced (fastest) Sync Client in each *k-th* group is selected as the IDMS reference, which is the one with the lowest playout delay (i.e., $d^k_{IDMS} = d^k_{master} = d^k_{min}$). As an example, in collaborative networked scenarios, the efficiency of the overall work may be improved by adjusting the lagged playout timings to the earliest one. Nevertheless, if there are slow Sync Clients under bad conditions (e.g., long network or processing delays), they could be constantly skipping MUs to achieve synchronization, and the continuity of their playout processes could be seriously affected. In such a case, if the playout process of the master Sync Client is extremely advanced, the playout buffers of all the Sync Clients in the same group may suffer underflow as the session advances in time. Hence, additional adaptive buffering control techniques should also be included when using this policy. This policy could not be applied in live streams, because of the inability to speed up a live stream, unless bigger buffers are employed at the Sync Client side at the beginning of the session.

The above policies are dynamic processes, since the master and slave roles of the Sync Clients can be exchanged during the session lifetime, depending on several uncontrollable network and end-system factors, allowing M/S switching techniques [8].

Another solution for selecting the IDMS reference is to define a virtual playout point (i.e., a fictitious Sync Client), obtained as the mean point of all the gathered playout processes in each *k-th* group (i.e., $d^k_{IDMS} = d^k_{master} = d^k_{mean}$). Using this method, the playout processes of the Sync Clients will be more continuous and smoother since the number and the values of the IDMS adjustments will be lower than in the previous methods. However, its use does not guarantee playout buffer overflow or underflow avoidance, because the playout rate imperfections and end-system situations (e.g., bandwidth availability, network load, CPU congestion, etc.) are in fact unpredictable. Also in this case, adaptive buffering control techniques should be deployed in future studies.

In all the previous discussed policies, the reporting of an erroneous playout point by a Sync Client, either accidental or malicious, may lead to undesired behavior. According to the adopted model, extremely advanced or lagged playout points will produce high adjustments on the Sync Clients' playout processes with the subsequent significant loss of real-time/continuity perception. Therefore, in any implementation, with the aim of avoiding faulty behavior, it would be advisable that the Sync Manager in SMS, or the distributed Sync Clients in DCS and M/S schemes, consider inconsistent playout information (exceeding configured limits) as a malfunction service and reject that information in the calculation of the IDMS target playout point.

Additionally, a fourth strategy can be adopted, but only when using SMS and if the Sync Manager and the Media Server are co-located. It consists of the synchronization to the nominal rate of the Media Server. In such a policy, the Sync Manager will act as a virtual master Sync Client with an ideal target playout timing, which is defined as the ideal playout timing when there is neither network nor end-systems' delay variability. Using this policy, if network conditions are quite stable, underflow and overflow situations will be avoided. This is because the playout states

of the deviated Sync Clients in each group would be adjusted to this ideal playout point every time an asynchrony situation is detected. Furthermore, this technique is beneficial for accurate Sync Clients because the smaller the deviations are in their playout states the smaller the IDMS adjustments that would be needed.

4.6. *IDMS Target Playout Point and Asynchrony Calculation*

In this sub-section, the calculation processes of the target playout point for IDMS in each one of the deployed schemes, and of the playout asynchrony with the selected IDMS reference, are described. Both processes have been also simplified in this evolved version of our IDMS solution.

When using SMS, if a de-synchronized (i.e., out of sync) situation in a specific *k-th* group is detected, the Sync Manager will calculate a target playout point for IDMS considering the output timing of the selected IDMS reference in that *k-th* group, following one of the above master selection policies. Then, it will include such information in a new RTCP IDMS settings packet (i.e., $p^k_{IDMS}=t'^k_{IDMS}+d^k_{IDMS}$), which will be sent as a compound RTCP packet. This packet can either reflect/forward the timing info of the IDMS report sent by the selected master Sync Client in that *k-th* group or even include playout hints for a specific (recent or even future) sent MU by inferring the timing evolution of the master Sync Client.

Let us consider the case that the *i-th* Sync Client belonging to the *k-th* group is playing a specific MU, which is identified by the RTP generation timestamp (t'^k_i)[5] and by a given MU sequence number identifier (MU^k_i), at p^k_i instant (local playout point). That Sync Client would consume the successive MUs with a (possibly deviated)[6] playout rate of μ^k_i MU/s. So, using SMS, it would play out the MU^k_{IDMS} (i.e., the MU to which the RTP timestamp carried in the RTCP IDMS packet refers[7]) at p'^k_{IDMS} instant, which possibly does not match with p^k_{IDMS} instant (target presentation time included in the RTCP IDMS settings packet).

Let $\Delta^k_{n,i}$ denote the asynchrony, for each *n-th* MU, between the evolution of the local playout point of the *i-th* Sync Client (p'^k_{IDMS}) and the IDMS target playout point (p^k_{IDMS}) in each *k-th* group:

$$\Delta^k_{n,i} = p^k_{IDMS} - p'^k_{IDMS} = p^k_{IDMS} - \left[p^k_i + \frac{1}{\mu^k_i}(MU^k_{IDMS} - MU^k_i) \right] \quad (4)$$

This asynchrony can also be easily calculated as the time difference between the playout delay of the selected master reference for IDMS ($d^k_{master}=d^k_{IDMS}=p^k_{IDMS}-t'^k_{IDMS}$) and the current local playout delay for that *i-th* Sync Client belonging to the *k-th* group ($d^k_{n,i}$):

$$\Delta^k_{n,i} = (d^k_{IDMS} - d^k_{n,i}) \quad (5)$$

[5] We are assuming here that each RTP packet encapsulating a part of the same MU will include the same (or very close) RTP timestamp.
[6] The playout rate deviations and their effect on media synchronization are explained in [3].
[7] We are also assuming here that the MUs are captured/generated periodically, e.g. θ≈25MU/s, despite that their size is not constant and the temporal pattern of transmitted RTP packets is variable. Therefore, we can identify which MU (i.e., its sequence number identifier) the RTP generation timestamp included in the RTCP IDMS settings packet (t'_{IDMS}) refers to, since the rate of advancements of RTP timestamps will be inferred by the PT field of the media stream, and the RTP and NTP Timestamps can be mapped according to the info included in each RTCP SR packet.

If DCS or M/S schemes are used, the target playout point for IDMS will not be received in a RTCP IDMS settings packet. Instead, it will be locally computed by each Sync Client in DCS, or by slave Sync Clients in M/S Scheme, independently of the other Sync Clients, using Eq. 5.

Independently on the IDMS scheme in use, Sync Clients must adjust their playout processes to acquire synchronization. It can be done by following two possible reactive methods. The first one is based on simple reactive actions such *'skips & pauses'* (aggressive playout adjustments), while the second one makes use of Adaptive Media Playout or AMP (smooth playout adjustments). These IDMS adjustment techniques are explained in next sub-sections.

The timing diagrams for the RTCP message exchanges in this evolved version of our RTCP-based IDMS solution using SMS and DCS are illustrated in Figures 2a and 2b, respectively. The diagram for the M/S Scheme has not been illustrated because its operation is similar to the one when using DCS, but in such a case only the master Sync Client multicasts IDMS reports.

4.7. *Fault Tolerance*

If a specific IDMS report (just one) sent by a Sync Client belonging to a specific group is lost, the IDMS control algorithm will not be affected drastically in any of the IDMS control schemes. This is because the Sync Manager in SMS, all the distributed Sync Clients in that group in DCS, or all the slave Sync Clients in that group in the M/S Scheme, will wait for the reception of the next IDMS report from that Sync Client, since the RTCP messages are sent regularly, as specified in [4].

If several successive IDMS reports are lost, the Sync Manager, in SMS, or the distributed Sync Clients, in DCS, could have to wait for an excessive period of time in order to collect the overall IDMS timing. So, a control timer has been included to manage the triggering of necessary playout adjustments. This way, the playout adjustments can be triggered either as a result of a detected asynchrony (exceeding a threshold) or as a timeout event of this monitoring timer. In case of such a timeout event, the required IDMS adjustments will be calculated according to the collected reports from the other Sync Clients. Every time playout setting instructions are sent to the Sync Clients (SMS) or directly performed by the Sync Clients (DCS), the timer is being reset.

The situation with loss of several successive IDMS reports is more problematic in case that the M/S Scheme is employed. This is because in that case only the master Sync Client reports on IDMS timing. Therefore, when the control timer runs out, the slave Sync Clients can decide to simply enforce IDMS adjustments according to the reference timing carried in the last received IDMS report from the master Sync Client.

As a summary, the flow chart of the designed overall IDMS algorithm for SMS (implemented in the Sync Manager) and DCS (implemented in each Sync Client) is sketched in Fig. 3. Using the M/S Scheme, the IDMS algorithm is executed by all the slave Sync Clients but, in such a case, they only have to collect the IDMS timing from the master Sync Client to compute the asynchrony.

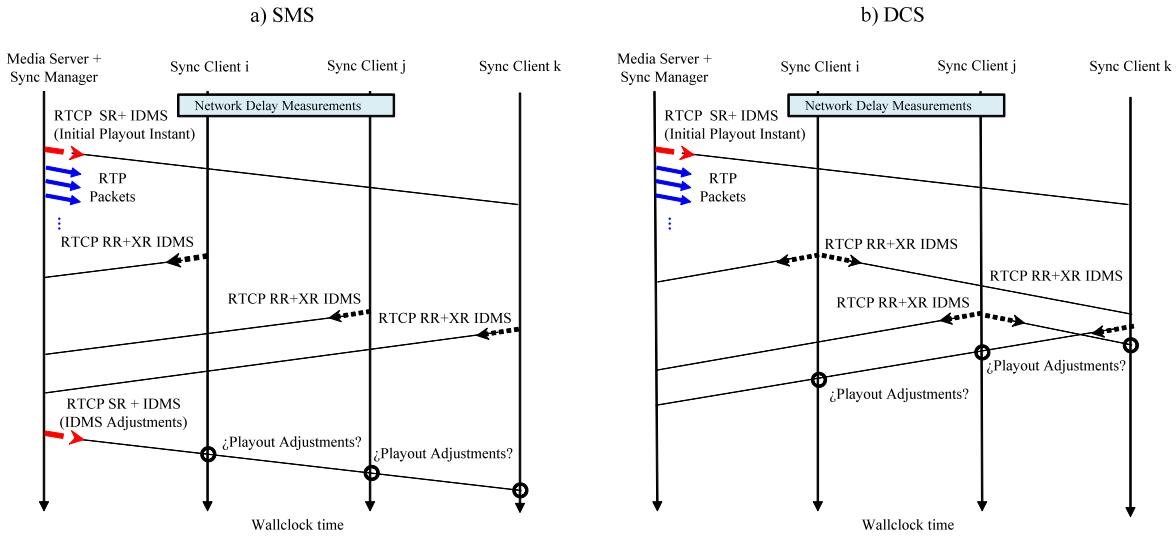

Fig. 2. RTP/RTCP Message Exchanges for IDMS.

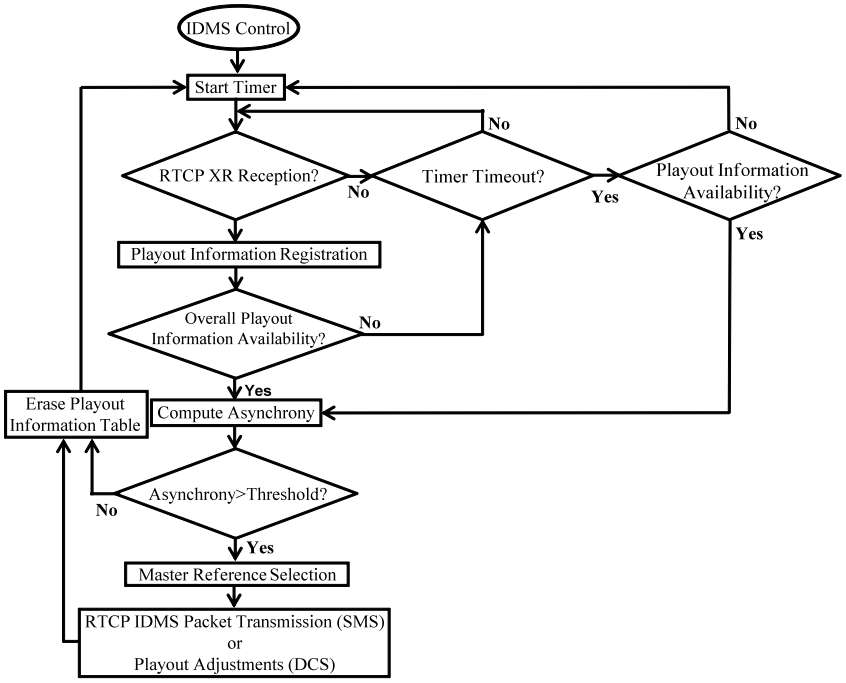

Fig. 3. Flow Chart of the designed DCS and SMS for IDMS.

4.8. *Aggressive playout Adjustments (Skips & Pauses)*

If $\Delta^{k}_{n,i}>0$ (see Eq. 5), the playout process of the *i-th* Sync Client belonging to the *k-th* group is advanced with respect to the selected IDMS reference. So, using aggressive adjustments, it must *'pause'* (stop playing) its playout process during $\Delta^{k}_{n,i}$ seconds to synchronize, causing a probable *freezing effect* (Fig. 4)[8]. Otherwise, if $\Delta^{k}_{n,i}<0$, the playout process of that Sync Client is lagged with respect to the IDMS reference. 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[9] (Fig. 4), thus probably causing a noticeable playout discontinuity/disruption.

[8] The modeling of the playout rate deviations (skew -γ- and drift -ε-) and their effect over the media synchronization are described in [3]
[9] We are assuming that only entire MUs can be skipped using our deployed aggressive adjustment policy.

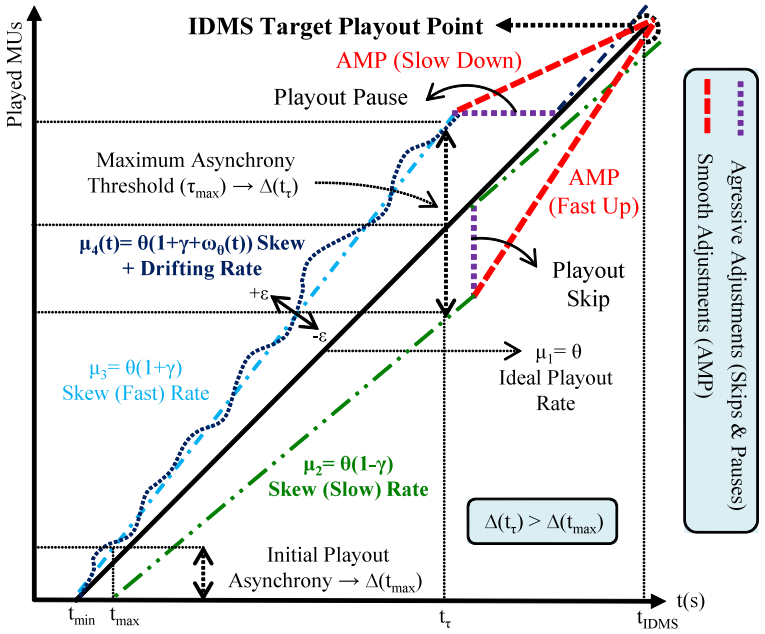

Fig. 4. Playout Rate Imperfections & Playout Adjustments for IDMS.

As shown in the evaluation section, those reactive playout actions will result in an overall synchronization status (within acceptable limits).

4.9. *Smooth playout Adjustments (Adaptive Media Playout or AMP)*

The above reactive playout adjustments could originate a noticeable degradation of the user perceived QoE. On one hand, some information may not be presented to the users, e.g. some important scenes may not be visualized (due to the *skipped* MUs). On the other hand, a sensation of loss of continuity may be noticed, e.g. a freezing effect on the display (due to the *paused* MUs).

Playout disruptions can also be originated due to network and end-systems fluctuations (e.g., congestion). To mitigate the above effects, several AMP techniques have been proposed in the past (e.g., [31, 33]). They consist of adjusting the media playout rate (i.e., playing the media faster/slower than normal), within perceptually tolerable ranges, to recover from undesired situations (e.g., buffer underflow/overflow or asynchrony situations) while providing glitch-free audio/visual quality.

Previous works on AMP solutions have been mostly focused on improving the intra-stream sync quality (e.g., [31, 32]) and, occasionally, for inter-stream sync purposes (e.g., [33]). However, our work in [3] proposed to extend the use of AMP for IDMS purposes. By using AMP, distributed Sync Clients were able to smoothly acquire an overall synchronization status every time an asynchrony threshold between their playout states was crossed. That AMP technique for IDMS has been enhanced in this work to be able to also operate with VBR traffic patterns, as well as adapted to be applied in each one of the deployed IDMS control schemes.

The operation of AMP for video simply consists of adjusting the display duration for each video frame. Nevertheless, AMP for audio involves signal processing in conjunction with time scaling techniques to stretch or widen an audio sequence while preserving the pitch of the signal.

Note that in this work we are only considering a single video stream, so we are not dealing with AMP for audio streaming. This is however a future work to be addressed when our IDMS solution is implemented in a real framework. This is because when IDMS is required in multimedia sessions involving multiple related streams, e.g. audio and video streams, the right decision may be to perform IDMS only on the audio stream (master stream for IDMS), and then enforce local inter-stream synchronization with respect the master stream for IDMS, as humans are more sensitive to degradation in audio quality than in video quality.

The operation of the enhanced and extended AMP technique for IDMS is as follows. Initially, the playout controller of each *i-th* Sync Client belonging to a specific *k-th* group must play out the buffered MUs at a non-adaptive playout rate given by $\mu^k_{n,i}=1/(t'_{n+1}-t'_n)$, as they were generated by the Media Server. 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.

As discussed, active Sync Clients (all of them in SMS and in DCS and the master in the M/S Scheme) include their current local playout point (t'^k_i, r^k_i, p^k_i) in each IDMS report they send to allow the Sync Manager (in SMS) or the distributed Sync Clients (DCS and M/S schemes) to collect the overall playout status.

Once an RTCP IDMS settings packet is received using SMS, the target playout point for IDMS (d^k_{IDMS}, MU^k_{IDMS}) is registered and processed. At this point, the AMP process will attempt to either increase (fast up) or decrease (slow down) the video playout rate in order to minimize $\Delta^k_{n,i}$ (see Eqs. 4 and 5) among all the remaining MUs to reach the IDMS target playout point. This can be accomplished by means of increasing/decreasing the playout time of each *n-th* MU a value of $\delta^k_{n,i}=(\Delta^k_{n,i})/(MU^k_{IDMS}-MU^k_i)$ seconds. This way, the local playout delay of the Sync Client can smoothly match the one of the IDMS reference in that *k-th* group (d^k_{IDMS}), as can be seen in Fig. 4.

As all RTP packets encapsulating a specific MU share the same timestamp, this evolved AMP technique can be applied independently of the variable proportion of RTP packets per MU, and independently of the total number of RTP packets to reach the IDMS target playout point. Therefore, our AMP technique can be applied for VBR streams (with constant MU rate). This could not be assumed when using the previous version [3], as that was based on sequence number instead of on timestamps, thus being only valid for CBR streams.

A key factor to perform AMP is to determine the allowed ratio within which the video playout speed can be varied without degrading the media quality. Previous relative 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* [31, 32]. Thus, we assume in our simulation tests that video playout adjustments up to 25% lead to

unnoticeable quality impairments, and we define a *playout factor* ($\varphi^k_{n,i}$) for each *n-th* MU in each *i-th* Sync Client belonging to the *k-th* group to specify this variation ratio. The optimal value of the $\varphi^k_{n,i}$ is computed, as smooth as possible, combining Equations 4 and 6, and using Eq. 7:

$$p^k_{IDMS} = p^k_i + \frac{1}{\mu^k_i(1+\varphi^k_{n,i})}(MU^k_{IDMS} - MU^k_i) \tag{6}$$

$$\varphi^k_{n,i} = \frac{1}{1+(\delta^k_{n,i}\cdot\mu^k_i)} - 1 \tag{7}$$

Note that if the calculated $\varphi^k_{n,i}$ is higher than 25%, it will be bounded to that maximum scaling ratio (i.e., $|\varphi_{max}| \leq 0.25$). To avoid such a situation, proper values for the initial buffering delay, τ_{max}, and the IDMS target playout point (MU^k_{IDMS}, t^k_{IDMS}) must be set.

The flow chart of the AMP algorithm using SMS is sketched in Fig. 5[10]. Note that in this figure, the playout buffer model at the client side, with a capacity of *C MUs* (not in bytes), is simplified by grouping the functionality of the jitter buffer (in which RTP packets are queued to compensate the effect of the network jitter and to de-packetize the encoded video frames), the decoder buffer (in which the encoded frames will temporarily wait for their decoding processes), and the render or display buffer (in which the decoded video frames are rendered to the visualization display).

Unlike in SMS, in which the Sync Clients receive the necessary playout adjustments in RTCP IDMS settings packets from the Sync Manager, in DCS and M/S schemes, the Sync Clients must locally compute (apart from carrying out) the required playout adjustments based on the received IDMS reports. Therefore, the AMP flow chart for DCS and M/S schemes is similar to the one in Fig. 5, but in these cases the IDMS target playout point will be directly and locally computed by the Sync Clients, and not received in an RTCP IDMS settings packet. Accordingly, the Sync Clients must choose the degree of the playout adjustments to achieve IDMS. On one hand, high values (near the maximum limit, i.e. $|\varphi_{max}| \approx 0.25$) of the playout factor will result in a nearly immediate synchronization status (depending, of course, on the allowed τ_{max}). On the other hand, low values of the playout factor will originate even more unnoticeable adjustments, but the overall synchronization status will be reached later.

In this work, for simplicity, a linear adjustment policy has been adopted. The number of MUs involved in the AMP process (N^{AMP}) in DCS and M/S schemes must be enough to allow an extremely deviated (advanced or lagged) Sync Client, with an asynchrony with the selected IDMS reference (d^k_{IDMS}) near the allowed threshold (i.e., $\Delta^k_{max} \approx \tau_{max}$), to adjust its playout timing without exceeding the maximum playout factor (i.e., $|\varphi_{max}| \leq 0.25$). That number is given by the Equations in (8):

[10] The dependency of the *k-th* group has been omitted in the notation of this figure. This is because the playout controller does not need to know about the group membership. It is the responsibility of the RTCP agent of each Sync Client to filter and send the RTCP messages from/to the group it belongs to.

$$N_{advanced}^{AMP} \geq \left\lceil \frac{\tau_{\max}}{\frac{1}{\mu_i^k \cdot (1-\varphi_{\max})} - \frac{1}{\mu_i^k}} \right\rceil ; \quad N_{lagged}^{AMP} \geq \left\lceil \frac{\tau_{\max}}{\frac{1}{\mu_i^k} - \frac{1}{\mu_i^k \cdot (1+\varphi_{\max})}} \right\rceil \tag{8}$$

In all the IDMS schemes, the AMP process will be finished once the IDMS target playout point (d^k_{IDMS}) is reached and will not be triggered again until a new out of sync situation, or a timeout event of the control timer, is detected.

The smoothed and linear playout adjustments to acquire IDMS using the adopted AMP technique are illustrated in Fig. 6. Note that when using DCS and M/S Schemes, the slave Sync Clients will match the IDMS target playout point at different instants, because the asynchrony situation will not be simultaneously detected at each one of them (and they may further choose different IDMS target playout points). Using SMS, however, all the Sync Clients will be synchronized almost simultaneously to the IDMS target playout point (see Fig. 4), because the required adjustments are indicated to them in the RTCP IDMS settings packet.

This way, using this evolved and extended AMP technique, the Sync Clients are able to achieve IDMS, using each one of the deployed schemes, while avoiding noticeable or annoying discontinuities in their playout processes.

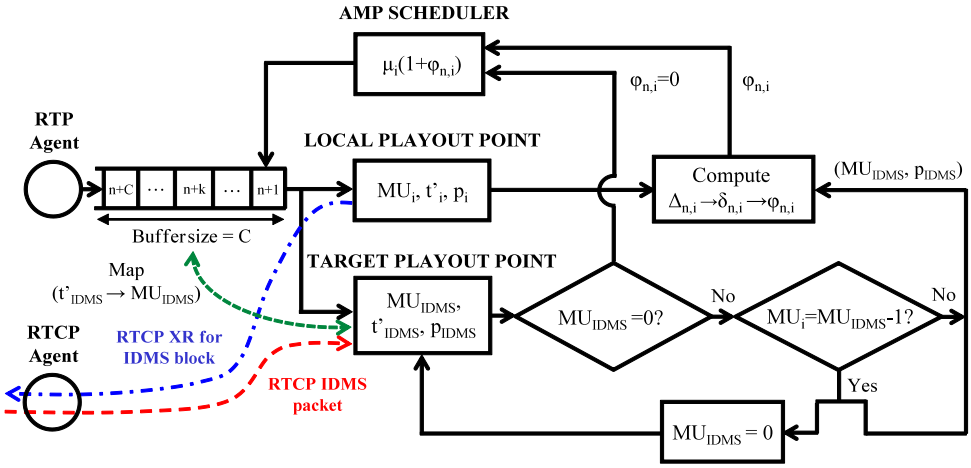

Fig. 5. Operation of the AMP Technique for IDMS.

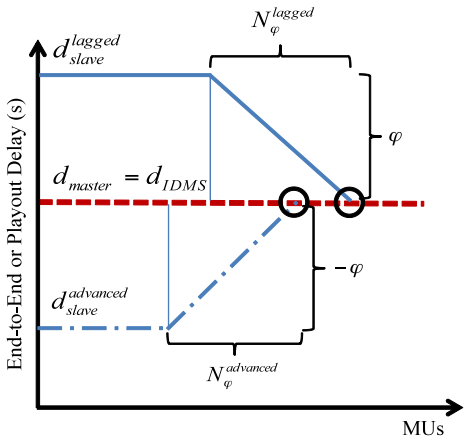

Fig. 6. Smoothed and Linear Adjustments to Acquire IDMS (DCS and M/S Scheme).

4.10. *Coherence Adjustment Technique for the DCS-based operation of the IDMS solution*

This sub-section is only focused on the DCS-based operation of our IDMS solution and describes a novel technique for enabling better coherence when this scheme is employed.

Let us assume that *"i-th Sync Client"* detects an asynchrony situation and starts its AMP process. During this period, the RTCP timer for that Sync Client expires and it multicasts an IDMS report including its (currently or recently adjusted) local playout point. It could be possible that *"j-th Sync Client"* is still waiting for that IDMS report from *"i-th Sync Client"* to complete all its group registries and compute the asynchrony in the group they belong to. In such a case, the asynchrony situation will not be detected by *"j-th Sync Client"* because it has been (partially) corrected by *"i-th Sync Client"* prior to send its next IDMS report to the group. This is not a serious constraint because, anyway, the overall asynchrony in that group will still be kept below the allowed threshold. Nevertheless, if *"j-th Sync Client"* was not selected as the IDMS reference, its playout process would not be synchronized to the IDMS target playout point selected by the other Sync Clients in that group. This means that all the Sync Clients in that group may not be synchronized simultaneously, and hence there will remain a residual asynchrony among them, lowering the overall accuracy of synchronization.

Consequently, a simple technique is proposed to solve this situation, thus providing better coherence when using DCS in our IDMS solution. It consists of using a bit of one of the fields reserved *"for future use"* in the IDMS report [7] to indicate that an out of sync situation has been recently detected and corrected by the sender of this report (by setting it to *'1'*). This way, once that IDMS report is received by the other *Sync Clients* belonging to the same group, they will be aware of such out of sync situation, and they will also adjust their playout processes to acquire a more fine-grained synchronization, using the IDMS timing information of the last cycle (i.e., the one computed the last time all the IDMS reports from that group were gathered).

The effect of this problem and the satisfactory responsiveness of the proposed technique are shown in Section 5.

5. Evaluation

5.1. *Simulation Scenario and Setup*

Modeling and simulations were conducted using NS-2 [34]. We have tested our IDMS solution in the multicast scenario shown in Fig. 7. The simulation setup is identical to the one in [3]. This is because our goal in this section is to prove the satisfactory responsiveness of the evolved and extended version of our IDMS solution in the same networked environment we used to evaluate the previous version in [3]. More specifically, we want to test the correct exchange of the newly designed and adopted RTCP messages, for each one of the deployed control schemes, as well as the correct performance of the newly adopted and designed algorithms and techniques for IDMS. This

way, the obtained results when using SMS are similar to the ones in [3] (although now our IDMS solution can be applied for VBR streams). But, apart from SMS, this section also provides results for DCS and M/S schemes, making them comparable in terms of several factors, such as interactivity, coherence, traffic overhead and computational load.

The simulation scenario has seven distributed Sync Clients belonging to two different logical synchronization groups (Group 1 - G1 - and Group 2 - G2 -). The Media Server transmitted a stream with a rate of θ=25 *MU/s*. All the links were bidirectional, their propagation delay were set to 10 ms, and their capacity was configured as shown in the figure. In addition, several aspects were considered to originate a significant end-to-end delay variability between the involved Sync Clients.

First, heavy and fluctuating background traffic (concretely, different cross-traffic flows following CBR/UDP, FTP/TCP and Pareto/UDP patterns) over the network topology was configured in order to force significant jitter variability for each individual Sync Client. The intensity of the background traffic was set in order to assure that the total amount of network traffic (RTP/RTCP streams + background traffic) was near the links' capacity at some instants during the session.

Second, the Sync Clients were strategically placed such that significant network delay variability from the Media Server to each one of them exists (see Figure 7 and Table 3). In G2, they were placed close together (but far from the Media Server) to show the benefits of DCS, compared to SMS, in such a case.

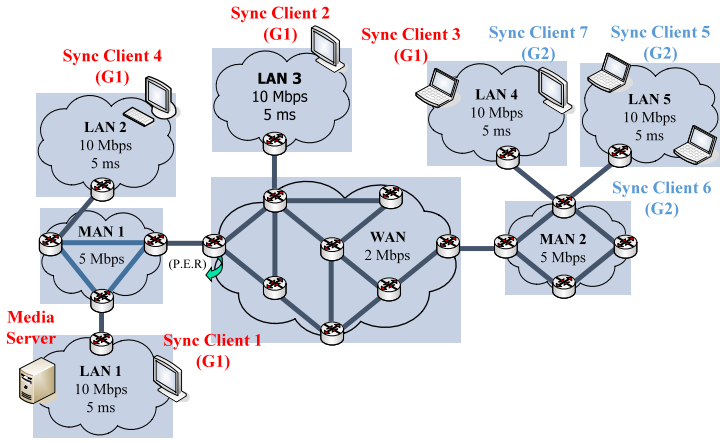

Fig. 7. Simulated Scenario.

Table 3 Sync Clients' Parameters and Aggrupation

Sync Client (SC)	Group	Mean RTT[1] (ms)	Playout Rate Skew, γ (%)	Playout Rate Drift, ε (%)
SC1	G1	~10	0.03 %	0.02 %
SC2	G1	~125	- 0.02 % → - 0.03 %	0.02 %
SC3	G1	~288	- 0.05 % → - 0.02 %	0.02 %
SC4	G1	~44	- 0.015 %	0.02 %
SC5	G2	~288	0 %	0.02 %
SC6	G2	~288	- 0.02 %	0.02 %
SC7	G2	~288	0.01 %	0.02 %

[1] RTT: Round Trip Time

Third, different playout rate imperfections (skews and drifts) were configured in the Sync Clients' playout processes (see Table 3). These values were set larger than customary deviations in inexpensive oscillators, which can vary between 10-100 *ppm* [35], in order to force higher asynchronies between the involved Sync Clients, and to test if these asynchronies were successfully handled by our IDMS solution[11]. Moreover, in order to corroborate the M/S switching capabilities of our IDMS solution (when using SMS and DCS) [8], those values were intentionally changed in two of the Sync Clients belonging to G1 at the midpoint of the simulation (*300-th* second), as can be appreciated in Table 3.

Fourth, we included an optional functionality to simulate congestion situations at the Sync Client side (e.g., due to CPU overload, decoding delays, audio/video card processing delays, etc.), which can cause abrupt discontinuities in the media playback and, therefore, a critical loss of synchronization. This congestion module runs on top of the local playout processes of each Sync Client, and it 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 that moment is stretched until this congestion period ends, causing a probable *freezing effect*. The congestion module was enabled in some of the simulations to Sync Client 2 (G1), with t_{ON}*=40 ms and* t_{OFF}*=120 sec*, in order to force higher playout times discrepancies between the Sync Clients in that group.

With respect to wall-clock synchronization, all the involved sync entities made use of the global scheduler clock of the simulator as the absolute shared, as well as local, clock source.

The duration of each simulation was set to 10 minutes and the value of τ_{max} was set to 80 ms in order to trigger playout corrections slightly before reaching an asynchrony of 100 ms, as that can be already perceivable and annoying in many IDMS use cases [2].

5.2. *Simulation Results*

5.2.1. *M/S Scheme*

The goal of this sub-section is to show the satisfactory responsiveness of the deployed M/S Scheme for IDMS in our RTCP-based solution. When using M/S Scheme, the value of τ_{max} (maximum allowable asynchrony threshold) was set to 50 ms in all the slave Sync Clients with the aim to keep the asynchrony bounds below 100 ms in each group. This is because, as discussed in Section 4, when using M/S Scheme, each slave Sync Client can only compute the asynchrony between its playout process and the one of the master Sync Client. So, the worst case would occur when the playout points of one slave Sync Client is extremely lagged and of another is extremely advanced, compared to the one of the master Sync Client.

[11] The effect of the delay variability and the playout rate imperfections over the local and global media synchronization (especially IDMS) are described in [3].

The playout (or end-to-end) delay evolution for the Sync Clients belonging to G1 (SC1, SC2 and SC3) and to G2 (SC5, SC6 and SC7) to acquire IDMS when using M/S Scheme is shown in Figures 8 and 9 when using aggressive and smooth adjustments, respectively, as reactive techniques. As can be appreciated, SC2 and SC7 were the master Sync Clients in G1 and G2, respectively.

It can be observed that the asynchrony between the playout states of the Sync Clients in each group progressively increased mainly due to the configured deviations in their local playout processes (Table 3). In G1 (upper graph in Fig. 8), it can be seen that every time lagged Sync Clients, SC3 in this case, detect an asynchrony between their local playout point and that of the master SC in that group (SC2) exceeding the allowed threshold (τ_{max}=*50 ms*), they adjusted their playout timing by skipping just one MU[12] to get in sync (see zoom view). Therefore, there was a residual asynchrony of around 10 ms (i.e., τ_{max}-$1/\theta \approx$ 50-40 ms) that was not corrected. Conversely, advanced Sync Clients, SC1 in this case, had to pause their playout process every time τ_{max} was crossed in order to minimize the detected asynchrony compared to the master (SC2). In such a case, advanced Sync Clients in G1 did acquire a more fine-grained synchronization than lagged ones because they paused specific MUs the exact time corresponding to the value of the detected asynchrony (Δ^{k}_{max}) in order to minimize it.

In G2 (lower graph in Fig. 8), the master Sync Client (SC7) was the most advanced (i.e., the one with the lowest playout delay: $d_{master}=d_7 \leq d_6 \leq d_5$). Therefore, slave Sync Clients (SC5 and SC6) had to skip MUs to reduce the detected asynchrony every time τ_{max} was exceeded.

We can observe in both graphs that, unlike in our previous SMS-based IDMS solution [3], in which the playout adjustments were performed almost simultaneously by all the Sync Clients as a result of a reception of a new control message (containing playout setting instructions) from the Sync Manager, when using M/S Scheme the playout adjustments are not performed simultaneously by all the slave Sync Clients (lower performance in terms of *coherence* [2]). This is because when using M/S Scheme, each one of the Sync Clients will only perform IDMS adjustments if the asynchrony between its local playout point and that of the master exceeds the allowed threshold, independently of the other slave Sync Clients. No knowledge about the overall synchronization status is available to the Sync Clients.

Figure 9 illustrates the same process, but using AMP. In such a case, the above long-term playout discontinuities were avoided for both lagged (skips) and advanced (pauses) slave Sync Clients. Also, lagged slave Sync Clients were more fine-grained synchronized than using aggressive adjustments because they smoothly increased their playout rate to minimize the detected asynchrony. Consequently, using AMP, the number of playout adjustments to acquire IDMS during the session for lagged slave Sync Clients was lower than using aggressive adjustments, because in

[12] We are assuming that only entire MUs can be skipped, each one with a duration of $1/\theta$=1/(25 MU/s) = 40 ms/MU.

this case there was no residual asynchrony after the adjustment processes for synchronizing (see zoom view in the lower graph of the Fig. 9).

We can also notice in the upper and lower graphs, in both figures, a significant increase and decrease, respectively, of the playout delay in all the Sync Clients as the session advanced in time, which caused an inherent progressive filling or emptying, respectively, of their playout buffer occupancy. Thus, if the master Sync Client is significantly advanced or lagged, the playout buffers of all Sync Clients may suffer overflow or underflow, respectively, if the media session had a long duration. Therefore, the use of M/S Scheme for IDMS would require the use of additional adaptive techniques, e.g. buffer fullness monitoring and control, to avoid such situations. This is left for further study.

To conclude the evaluation of M/S Scheme for IDMS, Fig. 10 corroborates that the playout adjustments in Fig. 9 were performed by all the slave Sync Clients within perceptually tolerable ranges. This way, using AMP and the M/S Scheme for IDMS, long-term playout discontinuities or disruptions were avoided while maintaining the playout factor (i.e., the percentage of the playout speed adjustment) within allowable bounds (i.e., $|\varphi^k_{max}| \leq 0.25$) during the session's lifetime.

5.2.2. *SMS*

In this sub-section, the performance of SMS for IDMS is assessed. Figure 11 illustrates the playout delay evolution for both groups of Sync Clients using SMS, when the Sync Manager, which was collocated with the Media Server, selected the slowest Sync Client in each group as the IDMS reference, and enabling the AMP mechanism in the receivers' playout processes. This graph shows that our IDMS solution is capable of independently managing the playout processes of Sync Clients belonging to different groups. It can be observed that faster Sync Clients in each group smoothly slowed down their playout rate every time τ_{max} was exceeded in their own group. Also, this graph clearly reflects the effect of the M/S switching capabilities ([8]) in G1: initially SC3 was the slowest one, but the playout rate deviations of SC2 and SC3 were intentionally changed at *300-th* second (see Table 3) in order to force SC2 to become the new master in that group.

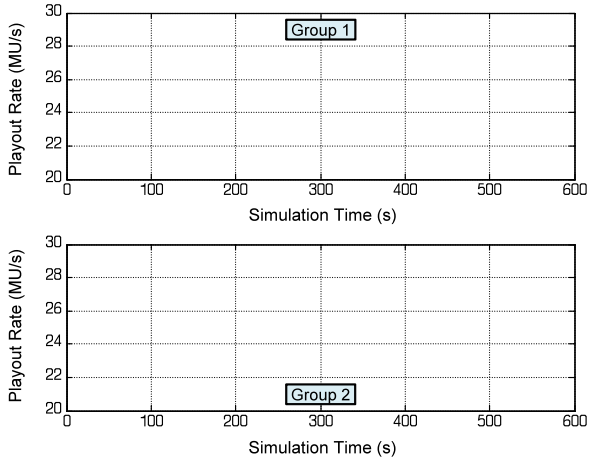

Fig. 10. Playout Rate Variation: M/S Scheme, Smooth Adjustments (AMP).

Despite that this policy (sync to the slowest Sync Client) can guarantee fairness in those scenarios where distributed users can compete among themselves (e.g., in battle MMOG or in networked quiz shows), it may not be appropriate if the master Sync Client is extremely lagged, because the playout buffers of all Sync Clients may progressively become flooded with MUs (overflow). As can be observed in Fig. 11, the playout delay progressively increased in both groups as the session advanced in time. Consequently, the application of this policy would require the use of novel adaptive buffering control techniques to avoid such a situation (left for further work).

Figure 12 illustrates the playout delay evolution of the Sync Clients belonging to G1 when using SMS, using the master selection policy of syncing to the fastest Sync Client, and using AMP as reactive technique. Here, lagged Sync Clients must speed up their media playout timing every time IDMS settings instructions are received from the Sync Manager, as a consequence of the detection of an asynchrony situation.

In such a case, three additional situations were considered. First, a new Sync Client (SC4) joined the on-going IDMS-enabled session at *30-th* second. Therefore, the Sync Manager sent a new IDMS settings packet to allow that *late-joiner* to acquire IDMS as soon as possible, despite that the asynchrony in that group was lower than the allowed threshold at that moment (see the zoom view at the bottom left). Second, the congestion module was enabled in SC2, therefore causing discontinuities (i.e., disruptions) in its playout process. As a result, the playout asynchrony between the distributed Sync Clients significantly increased every time SC2 suffered congestion (this can be clearly appreciated in both zoom views). Third, the media session was stopped/paused at *300-th* and re-started again at *320-th* second. It can be appreciated in the zoom views of the figure that all the Sync Clients were perfectly synchronized at the beginning of the two stages in which the session was divided (*Coarse Sync* [5, 3]), despite of the variable network delays from the Media Server to each one of them (see Table 3), due to the transmission of an initial RTCP IDMS Settings packet by the Sync Manager.

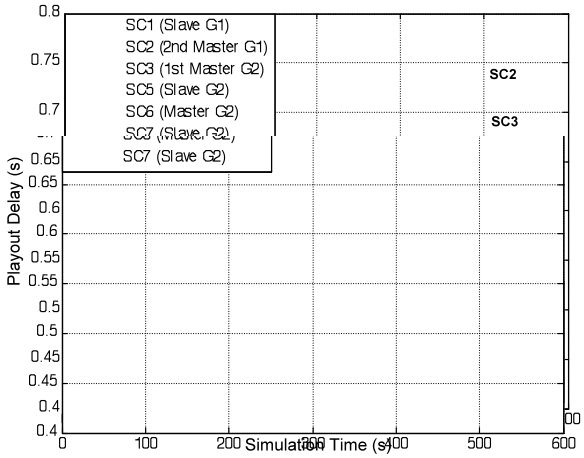

Fig. 11. Playout Delay Evolution: SMS to the Slowest with Smooth Adjustments (AMP).

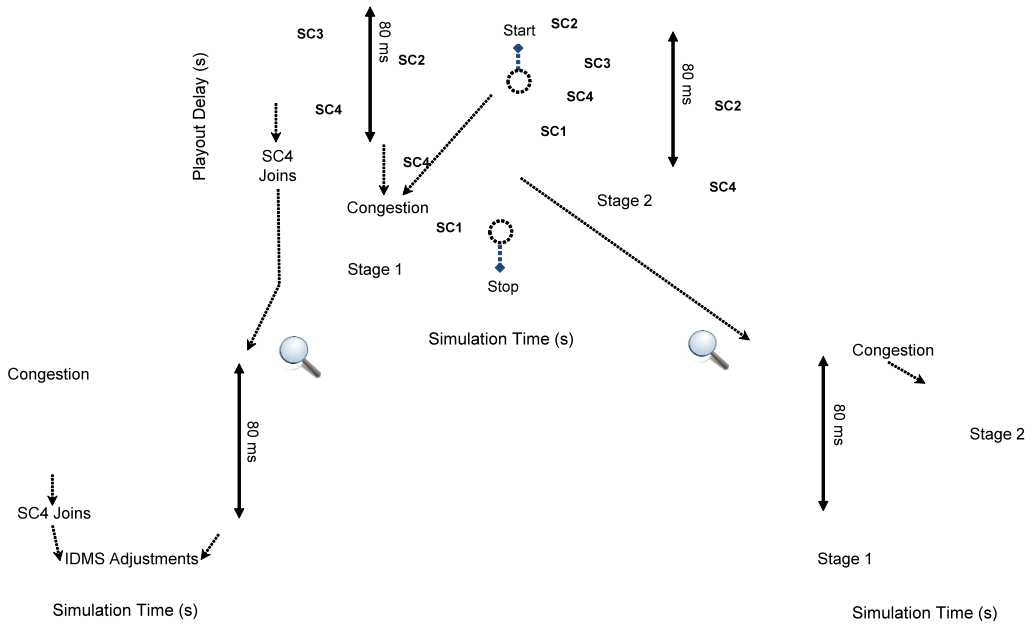

Fig. 12. SMS to the Fastest Sync Client with Coarse Sync, Latecomer Accommodation and Congestion Situations, using AMP as reactive technique.

Figure 13 illustrates the evolution of the playout processes of all the Sync Clients belonging to both groups when they were synchronized to the mean playout point, using SMS, and using AMP as reactive technique. For clarity, those processes have been illustrated in different graphs for each group. In this case, advanced/lagged Sync Clients had to smoothly slow down/fast up their playout rate to synchronize, avoiding long-term playout discontinuities. This master selection policy provided more fine-grained and less frequent playout adjustments than the previous ones. However, it cannot guarantee either buffer overflow or underflow situations because the existence of inaccurate (extremely advanced/lagged) Sync Clients cannot be predicted and would have a quantitative impact on the calculation of IDMS target (mean) playout point. So, as in the previous policies, novel adaptive techniques should also be adopted as future work.

The *sync to the Media Server nominal rate* policy was adopted in [3] to minimize the occurrence of buffer underflow/overflow situations (if network conditions are quite stable), but it is only applicable when using SMS and if the Media Server and the Sync Manager are the same entity. In such a case, the Media Server also acts as a virtual Sync Client with an ideal playout timing to which the playout processes of all the distributed Sync Clients in each group must match (every time τ_{max} is exceeded). This process is illustrated in Fig. 14 for the Sync Clients in both groups. It can be appreciated that the Sync Clients in each group were adjusted to the ideal playout timing every time an out of sync situation in that group was detected, resulting in a quite uniform playout delay evolution for all of them, which is a desirable feature in real-time multimedia services. The advantage of this policy is that accurate Sync Clients (e.g., SC5) do not have to

enforce significant adjustments in their playout timing for synchronizing. However, this policy does not take into account potential fluctuations of the end-to-end delay, e.g. due to bandwidth and network load constraints or congestion situations. Therefore, as in the previous policies, optimized and adaptive variations of this policy must be devised in future studies to be able to smooth out the effect of jitter and to keep the playout buffers' occupancy into stable and safe ranges.

5.2.3. *DCS*

In this sub-section, the effectiveness of our RTCP-based IDMS solution using DCS is proved. Figure 15 illustrates the same process as Fig. 11 (i.e., group-based IDMS to the slowest Sync Client), but using DCS as the control scheme. Both graphs seem quite similar. However, we can observe in Fig. 15 that the Sync Clients belonging to G1 (in which they were sparser than in G2) were not always simultaneously synchronized (e.g., at *200-th* and at *300-th* seconds). This is because when the SC1 detected an out of sync situation (after gathering the IDMS reports from the other Sync Clients in that group), SC2 was still waiting for the IDMS report from SC1. This way, when SC1 sent an IDMS report, including its local playout point, SC2 did not detect that out of sync situation because SC1 had already begun (or even finished) its adjustment process.

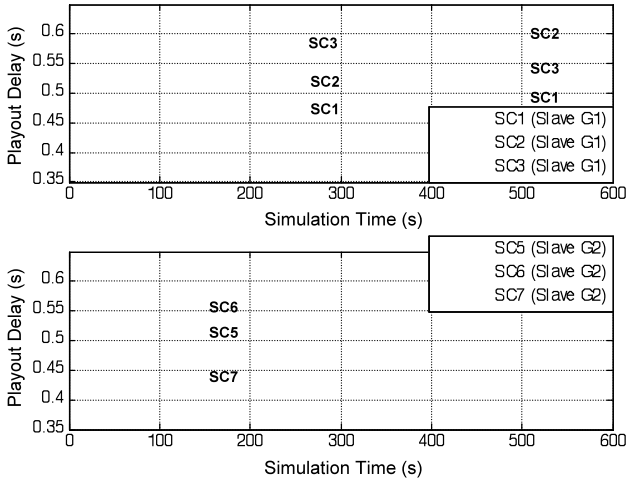

Fig. 13. Playout Delay Evolution in both groups: SMS to the Mean with AMP.

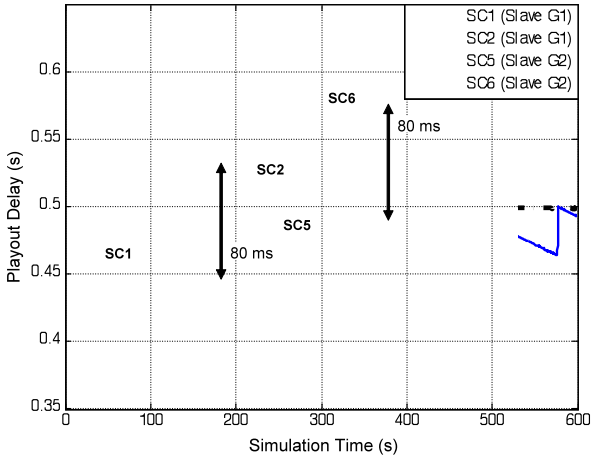

Fig. 14. Playout Delay Evolution: SMS to the Sender using AMP.

The same situation can be clearly seen in Fig. 16, but using DCS as the control scheme, using the master selection policy to the mean playout point, and using AMP as reactive technique. In this case, all the Sync Clients in G1 had to perform IDMS adjustments every time the asynchrony threshold was exceeded, since the IDMS target playout point was calculated as the mean of the IDMS timings from all of them. However, it can be seen that the 3 Sync Clients did not always simultaneously enforce IDMS adjustments due to the previous described situation. As discussed in Section 4.10, this issue is not a serious constraint because, anyway, the overall asynchrony in each group is kept below the allowed threshold (80 ms) for the entire duration of the session.

In order to overcome the above issue, the *coherence* adjustment technique (Section 4.10) was enabled in the next simulation, of which the results are shown in Fig. 17. In this case, all the Sync Clients were synchronized almost simultaneously every time τ_{max} was exceeded in each group because, despite that in some cases they did not detect the out of sync situation, they did receive the IDMS reports indicating the triggering of IDMS adjustments from the Sync Clients sending such reports. This resulted in a more fine-grained synchronization and a clearly better performance in terms of coherence (i.e., all the SCs were synchronized almost simultaneously to the same reference IDMS timing) [2].

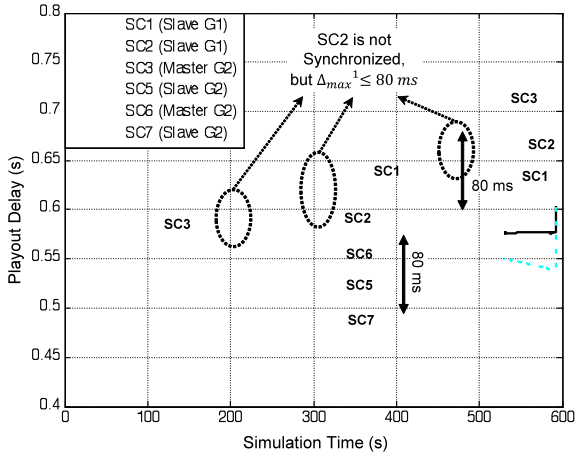

Fig. 15. Playout Delay Evolution: DCS to the Slowest, using AMP.

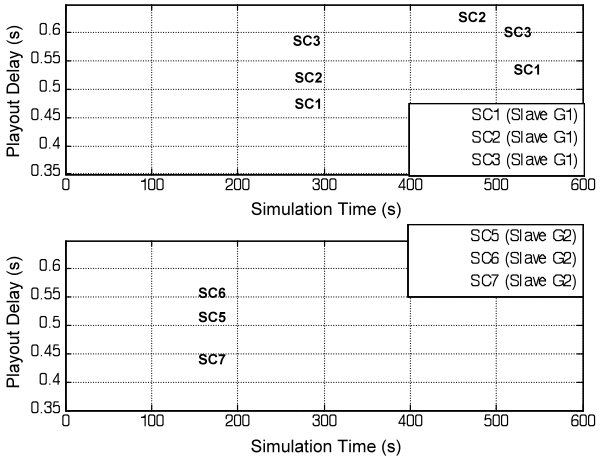

Fig. 16. Playout Delay Evolution: DCS to the Mean, using AMP.

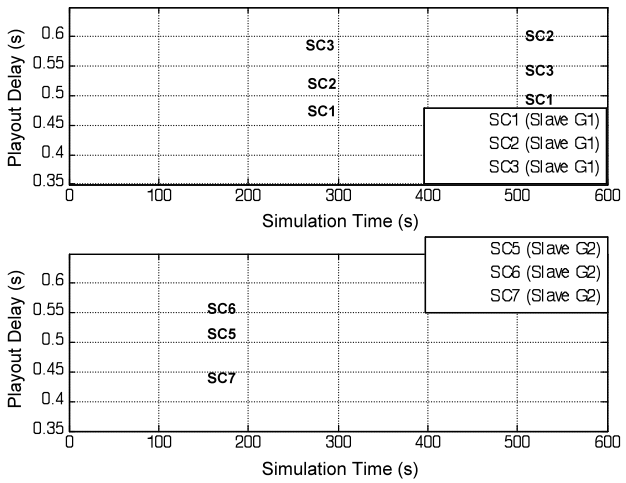

Fig. 17. Playout Delay Evolution: DCS to the Mean, using AMP + Coherence.

5.2.4. *Comparison among SMS and DCS*

This sub-section aims to provide some comparison results between SMS and DCS regarding interactivity, coherence, traffic overhead and computational load. M/S Scheme is not compared since a different asynchrony threshold was set when using this scheme, as discussed earlier.

5.2.4.1 *Interactivity*

As can be appreciated, Figures 13 (SMS) and 17 (DCS) are almost indistinguishable, since they show the evolution of the playout delay for the same groups of Sync Clients, when the same master selection policy and the same adjustment technique are employed, although using different control schemes. To clarify the differences among them, zoom views of the adjustments processes in both figures are presented in Figures 18 (for G1) and 19 (for G2) for both SMS (left graphs) and DCS (right graphs). It can be observed that asynchrony situations were corrected earlier using DCS than using SMS, in both groups. Therefore, these graphics confirm that DCS outperforms SMS in terms of interactivity. This occurs because, using DCS, each Sync Client started the playout adjustments once it collected the IDMS reports from all the other Sync Clients in its own group. Using SMS, larger delays occur when exchanging the control information for IDMS. First, the Sync Manager must collect all the IDMS reports from the Sync Clients and, as shown in Fig. 7 and in Table 3, there is a significant network delay (*RTT≈288 ms*) between the Sync Clients and the Sync Manager (which is collocated with the Media Server). Second, the Sync Manager may not be able to send an immediate RTCP IDMS settings packet, since it may have to adhere to some specific bounded timing rules, as specified in [4]. Third, a copy of that IDMS settings packet has to be received by all the Sync Clients to be able to enforce the required playout adjustments.

An advantage of DCS for IDMS is that the synchronization delay (i.e., the time interval needed to exchange the IDMS messages) can be significantly reduced if the Sync Clients are quite close to each other, as in G2 (see Fig. 7), as can be appreciated in Fig. 19.

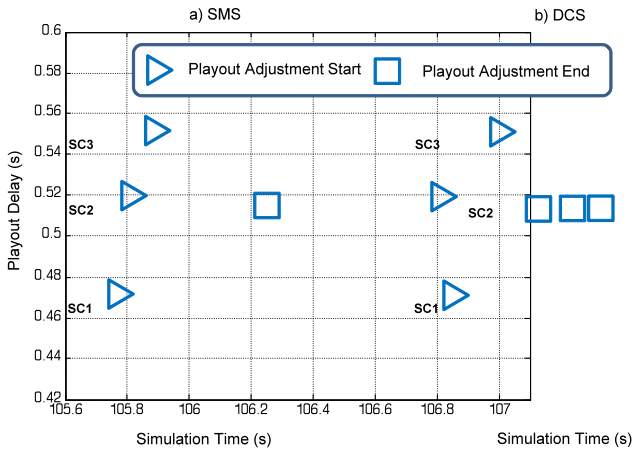

Fig. 18. Zoom View of the Playout Adjustments (AMP) in Group 1.

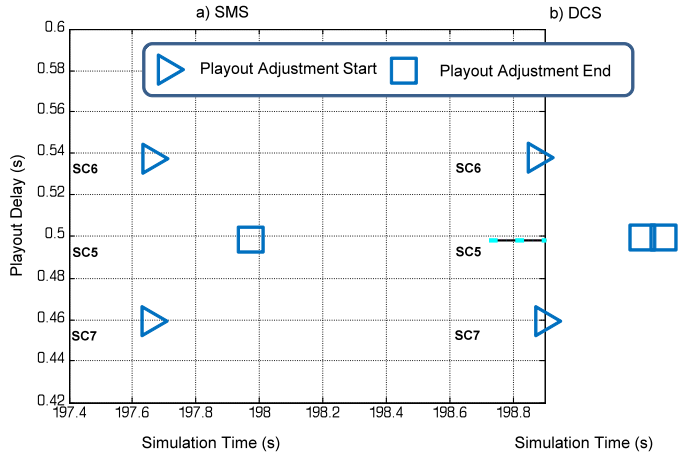

Fig. 19. Zoom View of the Playout Adjustments (AMP) in Group 2.

For instance, the maximum playout asynchrony in G2 during the session using the SMS was 82.3 ms (mean value 39.4 ms) whilst the one using the DCS was 81.2 ms (mean value 38.9 ms).

As discussed, the RTCP report interval [4], T_{RTCP}, plays a key role regarding the interactivity of our IDMS solution when using each one of the deployed control schemes. Next, the RTCP reporting rules are briefly reviewed to help to understand and justify these delay differences.

According to RFC 3550 [4], the total amount of control traffic added by RTCP should be limited to 5 % of the allocated RTP session bandwidth ($BW_{session}$). Besides, the T_{RTCP} must be dynamically adjusted according to the number of participants and their role (senders or receivers), as well as to the available bandwidth. If the proportion of senders constitute less than one quarter of the membership (i.e., $n_{senders} \leq \frac{1}{4} \cdot n_{participants}$, *where* $n_{participants}=n_{senders}+n_{receivers}$), this percentage is further divided into two parts, where 25 % must be dedicated to active senders (i.e., the Media Server), and the remainder can be consumed by receivers (i.e., the Sync Clients). Accordingly, the T_{RTCP} is computed as shown in the following formulas, where $avg(RTCP_{size})$ is the average size of all received and sent (compound) RTCP packets (including IDMS messages and transport and network layer headers, e.g. UDP and IP):

$$T_{RTCP}^{sender} = \frac{n_{senders} \cdot avg(RTCP_{size})}{0.25 \cdot 0.05 \cdot BW_{session}}; \quad T_{RTCP}^{receiver} = \frac{n_{receivers} \cdot avg(RTCP_{size})}{0.75 \cdot 0.05 \cdot BW_{session}} \tag{9}$$

If this proportion is not met, the RTCP bandwidth is equally shared between senders and receivers.

Besides, a minimum RTCP report interval, T^{min}_{RTCP}, is recommended in RFC 3550 [4] to avoid excessive RTCP packets when the number of participants is small. Two options can be used for that minimum bound: *i)* 5 s; and *ii)* $360/BW_{session}(kbps)$. Therefore, the maximum value between the computed T_{RTCP} and the selected value for the T^{min}_{RTCP} will be used for the RTCP report interval. After that, this parameter is randomized within the range *[0.5; 1.5]* to prevent floods of RTCP reports.

The recommended T^{min}_{RTCP} in [4] is, however, dropped in RFC 4585 [36]. Instead, an optional attribute, called *trr-int*, is defined to specify the minimum period interval (in ms) between two successive regular RTCP packets. This is an offset parameter to the computed T_{RTCP} to restrain from sending RTCP reports too frequently, while allowing more flexibility to transmit early RTCP packet when media stream related events need to be reported to the media sender. However, as it is an optional attribute, it may be set to zero if an application would benefit from a high frequent exchange of regular RTCP reports, as it is useful for IDMS. Therefore, this has been the policy we have adopted in this work.

Note that, even without considering either T^{min}_{RTCP} from [4] or *trr-int* from [36], the differences in terms of interactivity between SMS and DCS are significant (see Figures 18 and 19), especially for the IDMS use cases with stringent sync requirements [2]. Obviously, the differences would have been significantly larger if any of the above mechanisms (T^{min}_{RTCP} or *trr-int*) were adopted. It is also important to emphasize that if any of these mechanisms had been adopted, the interactivity of our IDMS solution would also have been affected, since large delays would be introduced when gathering the IDMS reports from the Sync Clients in each one of the deployed schemes. This is because the RTCP report interval in each Sync Client would be larger in such a case and, therefore, asynchrony situations would be detected later.

In our simulated scenario, the Sync Manager is implemented within the single Media Server resources (i.e., $n_{senders}=1$) and the bandwidth for the RTP Session was configured to $BW_{session}=200$ *Kbps*. If an approximate value of $avg(RTCP_{size})\approx1000$ bits is assumed, hence, according to formulas in (9) and after randomization, a delay of up to 0.6 s could be accumulated between the instant at which an asynchrony situation is detected by the Sync Manager and the instant at which it can transmit the RTCP IDMS settings packet. This gives the maximum Sync Manager delay because of the bounded RTCP reporting rules. Notice that this delay would be significantly larger (up to 5 s or slightly superior) if either T^{min}_{RTCP} or *trr-int* would have been considered. Also, such an effect would be much worse if the Sync Manager functionality would be implemented as a part of an RTP

receiver in a large-scale session (involving lots of Sync Clients), as can be derived from equations in (9).

Even though the maximum playout asynchrony in each group may slightly increase during this additional Sync Manager delay, the issue here is that, when using SMS, the out of sync situation will not be repaired during this time interval.

5.2.4.2 *Coherence*

Regarding coherence, it can be also appreciated in Figures 18 and 19 that using SMS all the Sync Clients finished their adjustment processes almost simultaneously, because all of them were synchronized at the same target playout point included in the RTCP IDMS settings packet. Using DCS, however, each Sync Client started to adjust its playout process once it collected the overall playout status in its own group. Thus, the IDMS adjustment period did not begin/finish at the same time in all of them. It can be appreciated in the right graphs of both figures.

Additionally, Figures 20 and 21 show that the playout rate was varied within tolerable limits (AMP) during the session lifetime in both SMS and DCS, when using the master selection policy to the mean playout point. So, according to conclusions in [31, 32], this may result in good performance in terms of QoE, since the playout rate adjustments would be unnoticeable to users. However, the zoom views in both figures show that, using SMS, the playout adjustments were performed almost simultaneously in all Sync Clients (Fig. 20), whilst this did not occur when using DCS (Fig. 21). This also confirms that SMS is superior to DCS in terms of coherence.

Even though asynchrony situations and playout adjustments below the allowed threshold may be unnoticeable to users, coherence is an important feature to enable high accuracy after synchronizing, since all the Sync Clients will be almost simultaneously adjusted to the same IDMS reference. This can be especially important when physically close-by Sync Clients need to be synchronized. This kind of synchronization is commonly referred to as Inter-Device Media Synchronization (IDES). Typical examples of IDES are: multiple devices within the same home environment (e.g., one TV in the living room and one in the kitchen, second-screen applications, etc.); an audio system containing multiple networked speakers to be synchronized; and a video wall consisting of various networked displays. In such situations, a single user may readily notice small delay differences (and subsequent playout adjustments), which may spoil the experience. Therefore, the synchronization requirements become stricter and coherence is more relevant here. For example, recent research studies regarding IDES show that the delay between two synchronized video streams in multiple-screen applications should not exceed 15 milliseconds [37].

5.2.4.3 *Playout Asynchrony Distribution*

To complement the previous results, Fig. 22 shows the distribution of the playout asynchrony in G1 for the same master selection policy (sync to the mean playout point), when using SMS and

DCS (enabling and disabling the coherence adjustment technique), and employing aggressive and smooth playout adjustments.

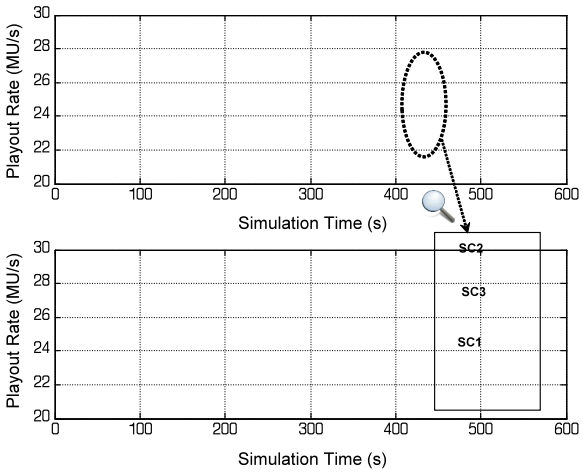

Fig. 20. Playout Rate Variation: SMS to the Mean, using AMP.

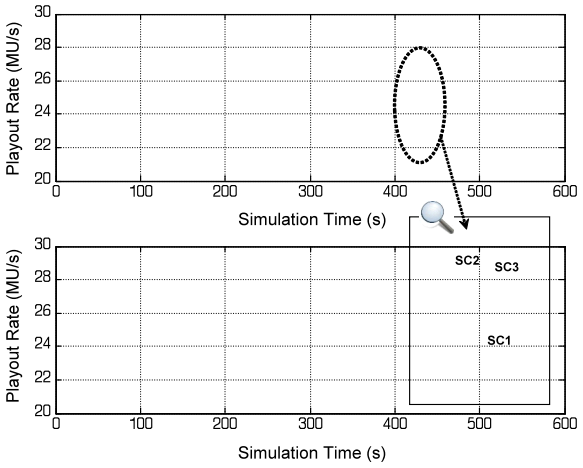

Fig. 21. Playout Rate Variation: DCS to the Mean, using AMP + Coherence.

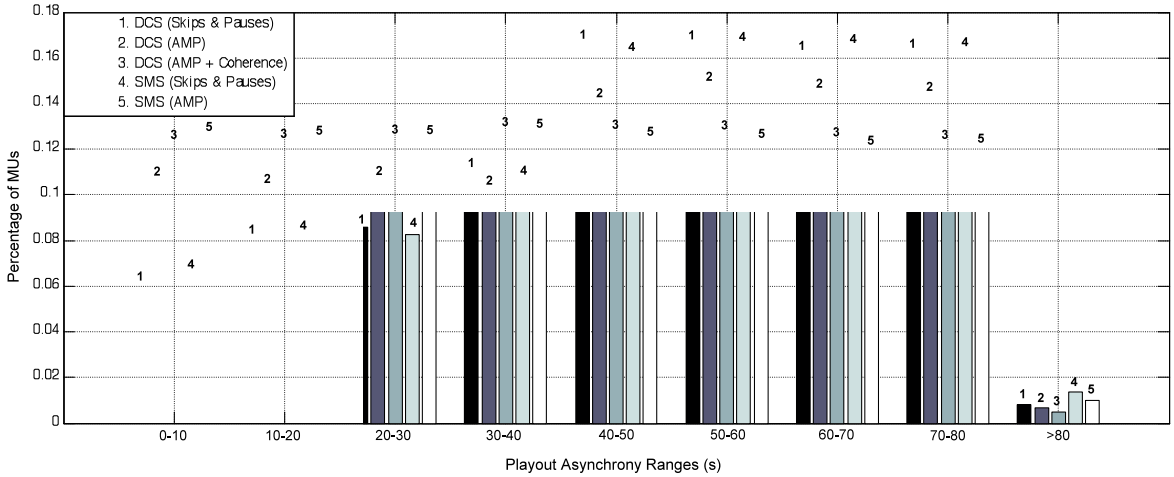

Fig. 22. Playout Asynchrony Distribution: SMS vs DCS (Group 1).

Regarding the adjustment techniques, it can be appreciated that the Sync Clients were more accurately synchronized using AMP than using aggressive adjustments in both schemes, because the percentage of MUs played out with low asynchrony values was significantly higher, which corroborates our conclusions in [3]. Conversely, large asynchrony values (near the threshold) were more probable using aggressive adjustments than using smooth adjustments.

Regarding the IDMS schemes, it can be appreciated that the percentage of MUs played out with low asynchrony ranges was significantly higher when SMS was employed. This is because in SMS all the Sync Clients were synchronized almost simultaneously once they received the RTCP IDMS settings packets (high performance in terms of coherence), as shown in Figures 18, 19 and 20. That percentage was significantly enhanced when enabling the coherence technique in DCS, as previously discussed. Conversely, the percentage of MUs that were played out with an asynchrony larger than the allowed threshold of 80 ms (i.e., out of sync MUs) was larger when using SMS than using DCS. This reflects the lower performance in terms of interactivity of SMS compared to DCS. Note that this percentage would have been significantly larger if any of the above-discussed mechanisms for reducing the T_{RTCP} (i.e., either T^{min}_{RTCP} or *trr-int*) had been adopted, especially for SMS.

Even though the percentages of the playout asynchrony distribution depend on several factors, such as the network topology under test, available bandwidth, number of Sync Clients and their characteristics, the values from Fig. 22 are representative for illustrative purposes, corroborating the differences between the use of different control schemes and adjustment techniques for IDMS.

5.2.4.4 *Traffic Overhead*

As discussed above, according to [4], the total amount of RTCP traffic must be fixed to 5 % of the allocated $BW_{session}$. Therefore, the traffic overhead added by our IDMS solution will be always significantly lower than this percentage when using each one of the deployed control schemes.

When using both SMS and DCS, the number of IDMS reports sent (multicast) by each Sync Client 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 approx.). This percentage is similar to our previous measurements in [5, 3], but in those cases the Sync Clients sent extended RTCP RR packets. Using DCS, only one RTCP IDMS settings packet is sent per each stage in which the session is divided (to indicate its starting instant), since each Sync Client locally calculates and performs the required adjustments according to the incoming IDMS reports. Contrarily, using SMS, the Sync Manager must multicast a new IDMS Settings packet every time an asynchrony situation is detected, or when a latecomer joins the session. Thus, the number of RTCP IDMS settings packets sent by the Sync Manager (only in SMS) depends on the detected asynchrony between the playout states of the Sync Clients in each group. As an example, when the synchronization to the mean playout point was employed (Fig. 13), 5 packets were sent to G1 and only 3 packets to G2 (because the playout

deviations in that group were minor). These amounts are significantly lower than the ones obtained in [5], in which specific RTCP APP packets were sent by the Sync Manager, probably because in that case each Sync Client had to play out 3 different media streams (we also carried out inter-stream sync in that work) and the end-systems could become overloaded at some instants.

The newly defined RTCP reports in this evolved version of our IDMS solution are larger than the ones used in the previous version [3, 5]. The IDMS report has a length of 288 bits vs 96 bits of extension of the RTCP RR in the previous version of our IDMS solution, whilst the RTCP IDMS settings packet has a length of 224 bits vs 192 bits of each specific RTCP APP defined in the previous version of our IDMS solution. However, this only minimally affects the frequency of sending RTCP reports (see equations in (8)) since, as discussed, the total amount of RTCP traffic is bounded to 5 %. In either case, the traffic overhead for IDMS still remains very low, because we have not defined a new proprietary protocol either, but we have taken advantage of the RTP/RTCP extension capabilities for designing our own adaptive IDMS solution.

5.2.4.5 *Computational Load*

Regarding the computational load of our RTP/RTCP-based IDMS solution, when using DCS each Sync Client must only process a mean percentage of IDMS reports from the Sync Clients given by $(N_G\text{-}1)/N_T$ per each RTCP report interval, where N_G is the number of Sync Clients belonging to a specific group *G,* and N_T is the total number of Sync Clients in the session. Using SMS, the Sync Manager must process all the IDMS reports from all the Sync Clients in the session (N_T) but, contrarily, distributed Sync Clients have to process a significantly lower number of IDMS messages from the Sync Manager.

6. Conclusions and Future Work

In this work, the rationale for using and extending RTP/RTCP standard protocols for IDMS purposes has been discussed, in comparison with other candidate protocols. Besides, a preliminary version of a centralized RTP/RTCP-based IDMS solution has been enhanced to avoid compatibility constraints and to be able to constitute an interoperable, accurate and standardized IDMS solution. This evolved version of the IDMS solution makes use of newly-defined RTCP packets, which results in a simplified operation and enhanced responsiveness to achieve IDMS. Besides, because of the suitability of different control schemes for specific IDMS use cases, our IDMS solution has been extended to adopt both centralized (SMS and M/S Scheme) and distributed schemes (DCS). This enhances the flexibility of our IDMS solution to be efficiently deployed in a wide range of scenarios and use cases. The adoption of these control schemes has implied the adaption and extension of the previously designed control algorithms and techniques for IDMS, which were uniquely SMS-based, as well as the inclusion of some novel aspects, such as the coherence adjustment technique. Simulation tests have shown the consistent behavior of the control schemes and the satisfactory responsiveness of the presented RTCP-based IDMS solution, which is able to

achieve an overall synchronization status (in independent groups of disjoint clients) while minimizing the occurrence of long-term playout discontinuities or disruptions (such as *skips & pauses*), and hardly increasing the network and computational load.

Despite that simulation tests cannot definitively validate the performance of our RTP/RTCP-based IDMS solution, they show the satisfactory responsiveness of the designed algorithms and techniques, as well as the consistent behavior of the deployed control schemes for IDMS, according to our qualitative study in [2]. These simulation studies are not meant to be conclusive, but the implementation in a simulation framework provides us more flexibility for double checking assumptions, for evaluating the performance of our IDMS solution in various network environments, and for identifying critical issues during the design and validation processes, thus providing a valuable feedback information about the right direction of our research work.

As future work, we will assess the impact of the application of the proposed IDMS techniques by implementing them in a real media framework (e.g., Gstreamer [38]) in order to perform real-world assessments in present-day network environments, analyzing the effects on the user experience (QoE) of different levels of out of sync situations and how they are avoided by using our IDMS solution. We also plan to include further functionalities to our IDMS solution to be able to trigger the IDMS adjustments based on the instantaneous importance of the media content being delivered. Finally, we will to continue with our effort in the standardization process of our IDMS solution, by tackling some important issues such as interactivity and scalability.

ACKNOWLEDGMENT

This work has been financed, partially, by Universitat Politècnica de València (UPV), under its R&D Support Program in PAID-11-02-331 Project, PAID-01-10 and PAID-00-12. TNO's work has been partially funded by European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. ICT-2011-8-318343 (STEER project). CWI's work has been partially funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. ICT-2011-7-287723 (Reverie project).

REFERENCES

[1] 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.

[2] M. Montagud, F.Boronat, H. Stokking, R. van Brandenburg, "Inter-destination multimedia synchronization: schemes, use cases and standardization", MMSJ, 18(6), 459-482, November 2012.

[3] M. Montagud and F. Boronat, "Enhanced adaptive RTCP-based Inter-Destination Multimedia Synchronization approach for distributed applications", Computer Networks Journal, Volume 56, Issue 12, pp. 2912-2933, August 2012.

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

[5] 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.

[6] ETSI TS 183 063 V3.5.2 (2011-03), Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); IMS-based IPTV stage 3 specification.

[7] R. van Brandenburg, H. Stokking, M.O. Van Deventer, F. Boronat, M. Montagud, K. Gross, "RTCP for inter-destination media synchronization", draft-ietf-avtcore-idms-09, IETF Audio/Video Transport Core Maintenance Working Group, Internet Draft, March 2013.

[8] 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.

[9] 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.

[10] M. Roccetti, S. Ferretti, and C. Palazzi; "The Brave New World of Multiplayer Online Games: Synchronization Issues with Smart Solution", 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC '08), Washington, USA, 587-592, May 2008

[11] C. Fleury, T. Duval, V. Gouranton, B. Arnaldi, "Architectures and Mechanisms to efficiently Maintain Consistency in Collaborative Virtual Environments", Workshop on Software Engineering and Architectures for Realtime Interactive Systems, SEARIS 2010, Massachussets, USA, March 2010.

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

[13] 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.

[14] 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.

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

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

[17] Y. Kurokawa, Y. Ishibashi, and T. Asano, "Group synchronization control in a remote haptic drawing system", *IEEE ICME*, Beijing (China), pp. 572-575, July 2007.

[18] K. Hosoya, Y. Ishibashi, S. Sugawara, K.E. Psannis, "Group synchronization control considering difference of conversation roles", IEEE ISCE '09, 948-952, Kyoto (Japan), May 2009.

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

[20] C. Diot and L. Gautier, "A Distributed Architecture for Multiplayer Interactive Applications on the Internet", IEEE Network, Vol.13, N. 4, pp. 6-15, August 1999.

[21] 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.

[22] C.E. Palazzi, et alter, "On maintaining interactivity in event delivery synchronization for mirrored game architectures", IEEE Global Telecommunications Conference, Dallas (USA), 157-165, December 2004.

[23] J. Ott, J. Chesterfield, E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, February 2010.

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

[25] T. Friedman, R. Caceres, A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.

[26] J. Rosenberg, et alter, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[27] H. Schulzrinne, A. Rao, R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

[28] P. Calhoun, et alter, "Diameter Base Protocol", September 2003.

[29] C. Groves, et alter, "Gateway Control Protocol Version 1", RFC 3525, June 2003.

[30] A. Williams, K. Gross, R. van Brandenburg, and H. Stokking, "RTP Clock Source Signalling", draft-williams-avtcore-clksrc-04, IETF Audio/Video Transport Working Group, Internet Draft, May 2013.

[31] 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.

[32] 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.

[33] 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.

[34] The Network Simulator 2 (NS-2), www.isi.edu/nsnam/ns/

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

[36] J. Ott, et alter, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.

[37] Y. Bang, et alter, "Wireless Network Synchronization for Multichannel Multimedia Services", IEEE ICACT 2009, Piscataway, NJ, USA, February 2009.

[38] GStreamer: open source multimedia framework, www.gstreamer.net/